APLIKASI PENCATATAN AKADEMIK MAHASISWA DENGAN *BLOCKCHAIN*

SKRIPSI

Oleh:

RAIHAN HIDAYATULLAH DJUNAEDI NIM. 2041720108

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

HALAMAN PENGESAHAN

APLIKASI PENCATATAN AKADEMIK MAHASISWA DENGAN *BLOCKCHAIN*

Disusun oleh:

RAIHAN HIDAYATULLAH DJUNAEDI. NIM. 2041720108

Laporan Akhir ini telah diuji pada tanggal 19 Juli 2024
Disetujui oleh:

1.	Pembimbing Utama	:	Pramana Yoga Saputra, S.Kom., MMT. NIP. 198805042015041001	.....
2.	Pembimbing Pendamping	:	Yan Watequlis Syaifudin, ST., M.MT., Ph.D. NIP. 198101052005011005	.....
3.	Penguji Utama	:	Muhammad Afif Hendrawan., S.Kom., M.T. NIP. 199111282019031013	.....
4.	Penguji Pendamping	:	Dian Hanifudin Subhi, S.Kom., M.Kom. NIP. 198806102019031018	.....

Mengetahui,

Ketua Jurusan Teknologi Informasi	Ketua Program Studi D4 Teknik Informatika
Rosa Andrie Asmara, ST., MT., Dr. Eng.	Dr. Ely Setyo Astuti, S.T., M.T.
NIP. 198010102005011001	NIP. 197605152009122001

PERNYATAAN

Dengan ini saya menyatakan bahwa pada Skripsi ini tidak terdapat karya, baik seluruh maupun sebagian, yang sudah pernah diajukan untuk memperoleh gelar akademik di Perguruan Tinggi manapun, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini serta disebutkan dalam daftar sitasi/pustaka.

Malang, 17 Juli 2024

Raihan Hidayatullah Djunaedi

ABSTRAK

Djunaedi, Raihan Hidayatullah. "Aplikasi Pencatatan Akademik Mahasiswa dengan Blockchain." **Pembimbing: (1) Pramana Yoga Saputra, S.Kom., MMT., (2) Yan Watequlis Syaifudin, ST., M.MT., Ph.D.**

Skripsi, Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2024.

Dalam era digital, pencatatan data akademik, termasuk Satuan Kredit Kegiatan Mahasiswa (SKKM), memerlukan keamanan, integritas, dan transparansi yang tinggi. Namun, sistem pencatatan yang ada di Politeknik Negeri Malang masih menghadapi tantangan seperti potensi manipulasi data dan proses verifikasi yang tidak efisien. Penelitian ini bertujuan untuk mengatasi permasalahan tersebut dengan mengembangkan dan mengimplementasikan sistem pencatatan SKKM berbasis teknologi blockchain. Pada tahap awal, sistem ini menggunakan Hyperledger Besu sebagai jaringan blockchain privat, di mana smart contract ditulis dengan Solidity untuk mengotomatiskan proses verifikasi dan validasi. Teknologi blockchain memastikan integritas dan keamanan data melalui pencatatan transaksi yang tidak dapat diubah. Selain itu, sistem ini juga memanfaatkan InterPlanetary File System (IPFS) untuk penyimpanan file yang terdesentralisasi, memastikan keamanan bukti partisipasi mahasiswa. Hasil pengujian menunjukkan bahwa fungsi menambah SKKM memiliki waktu respons rata-rata tertinggi sebesar 26,57 detik, sedangkan fungsi mengedit SKKM memiliki waktu respons rata-rata terendah sebesar 3,23 detik. Waktu respons untuk fungsi-fungsi lainnya berkisar antara 8,99 detik hingga 17,97 detik. Secara keseluruhan, sistem frontend menunjukkan performa yang baik dalam menangani transaksi pengguna. Hasil pengujian backend menunjukkan bahwa waktu respons rata-rata untuk transaksi blockchain cukup konsisten. Fungsi menambah SKKM memiliki waktu respons rata-rata tertinggi sebesar 5,84 detik, sementara deploy smart contract memiliki waktu respons rata-rata terendah sebesar 4,44 detik. Sistem backend menunjukkan efisiensi yang baik dalam memproses transaksi.

Kata Kunci: Blockchain, Hyperledger Besu, Smart Contract, InterPlanetary File System, Satuan Kredit Kegiatan Mahasiswa

ABSTRACT

Djunaedi, Raihan Hidayatullah. *"Student Academic Record application with Blockchain".* ***Supervisor: Pramana Yoga Saputra, S.Kom., MMT., Co-Supervisor: Yan Watequlis Syaifudin, ST., M.MT., Ph.D.***

Thesis, Informatics Engineering Study Program, Department of Information Technology, Malang State Polytechnic, 2024.

In the digital era, recording academic data, including Student Activity Credit Units, requires high security, integrity, and transparency. However, the existing recording system at State Polytechnic of Malang still faces challenges such as the potential for data manipulation and an inefficient verification process. This research aims to overcome these problems by developing and implementing a blockchain technology-based Student Activity Credit Units recording system. In the initial stage, the system uses Hyperledger Besu as a private blockchain network, where smart contracts are written with Solidity to automate the verification and validation process. Blockchain technology ensures data integrity and security through immutable transaction records. In addition, the system also utilizes the InterPlanetary File System for decentralized file storage, ensuring the security of proof of student participation. The test results show that the function of adding Student Activity Credit Units has the highest average response time of 26.57 seconds, while the function of editing Student Activity Credit Units has the lowest average response time of 3.23 seconds. Response times for other functions ranged from 8.99 seconds to 17.97 seconds. Overall, the frontend system shows good performance in handling user transactions. The backend test results show that the average response time for blockchain transactions is quite consistent. The add Student Activity Credit Units function has the highest average response time of 5.84 seconds, while deploy smart contract has the lowest average response time of 4.44 seconds. The backend system shows good efficiency in processing transactions.

Keywords: *Blockchain, Hyperledger Besu, Smart Contract, InterPlanetary File System, Student Activity Credit Unit*

KATA PENGANTAR

Puji Syukur kami panjatkan kehadirat Allah SWT/Tuhan YME atas segala rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi dengan judul "APLIKASI PENCATATAN AKADEMIK MAHASISWA DENGAN *BLOCKCHAIN*." Skripsi ini penulis susun sebagai persyaratan untuk menyelesaikan studi program Diploma IV Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang.

Kami menyadari bahwasannya dengan tanpa adanya dukungan dan kerja sama dari berbagai pihak, kegiatan laporan akhir ini tidak akan dapat berjalan baik. Untuk itu, kami ingin menyampaikan rasa terima kasih kepada:

1. Bapak Rosa Andrie Asmara, ST., MT., Dr. Eng , selaku Ketua Jurusan Teknologi Informasi
2. Ibu Dr. Ely Setyo Astuti, S.T., M.T., selaku Ketua Program Studi DIV Teknik Informatika
3. Bapak Pramana Yoga Saputra, S.Kom., MMT., selaku Dosen Pembimbing Utama yang telah memberikan arahan, bimbingan, dan motivasi selama penulisan skripsi ini.
4. Bapak Yan Watequlis Syaifudin, ST., M.MT., Ph.D., selaku Dosen Pembimbing Pendamping yang telah memberikan saran dan masukan yang sangat berguna.
5. Orang Tua dan Keluarga, yang selalu memberikan doa, dukungan, dan kasih sayang yang tiada henti.
6. Teman-teman terdekat yang selalu hadir untuk menemani, mendengarkan dan memberikan semangat untuk saya sebagai penulis untuk menyelesaikan skripsi saya.
7. Dan seluruh pihak yang telah membantu dan mendukung lancarnya pembuatan Laporan Akhir dari awal hingga akhir yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan akhir ini, masih banyak terdapat kekurangan dan kelemahan yang dimiliki penulis baik itu sistematika penulisan maupun penggunaan bahasa. Untuk itu penulis mengharapkan saran dan kritik dari berbagai pihak yang bersifat membangun

demi penyempurnaan laporan ini. Semoga laporan ini berguna bagi pembaca secara umum dan penulis secara khusus. Akhir kata, penulis ucapkan banyak terima kasih.

Malang, 02 Juli 2024

Raihan Hidayatullah Djunaedi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I. PENDAHULUAN

1.1 Latar Belakang

Pendidikan tinggi adalah aspek kunci dalam pembangunan manusia dan masyarakat. Dalam era digital saat ini, data akademik mahasiswa, termasuk nilai, jadwal kuliah, transkrip akademik, dan informasi penting lainnya, menjadi bagian integral dari pencatatan dan pemantauan kemajuan akademik mahasiswa. Tantangan utama dalam pencatatan data akademik adalah menjaga keamanan, integritas, dan transparansi data tersebut. (Habib et al., 2023).

Masalah yang sering dihadapi dalam pencatatan data akademik adalah potensi manipulasi dan risiko keamanan data (Amo-Filva et al., 2023). Penggunaan *database* tradisional sering kali memiliki kerentanan terhadap manipulasi data oleh pihak yang tidak berwenang. Data yang tersimpan secara terpusat di satu *server* berisiko diubah atau dihapus, sehingga integritas data tidak terjamin. Selain itu, proses verifikasi dan penyimpanan data bergantung pada otoritas pusat, yang dapat menimbulkan risiko kesalahan manusia, penyalahgunaan wewenang, dan keterlambatan proses (Bari & Patel, 2023).

Topik ini dipilih karena relevansinya terhadap isu-isu kontemporer dalam pencatatan data akademik. Dalam era digital saat ini, pencatatan data akademik menjadi semakin penting dan kompleks, terutama dalam hal keamanan dan integritas data. Ini mencakup berbagai aspek, mulai dari data pribadi mahasiswa hingga data akademik seperti nilai dan sertifikat. Penelitian ini berfokus pada isu spesifik terkait pencatatan dan pengelolaan data Satuan Kredit Kegiatan Mahasiswa (SKKM). SKKM adalah sistem yang digunakan oleh beberapa universitas di Indonesia untuk memberikan apresiasi terhadap kegiatan ekstrakurikuler dan nonkurikuler yang dilakukan oleh mahasiswa. Setiap kegiatan yang diakui oleh universitas diberikan nilai kredit tertentu, dan mahasiswa diharuskan mengumpulkan jumlah kredit minimum selama masa studi mereka. Namun, pencatatan data SKKM sering kali menimbulkan masalah keamanan dan integritas data. Misalnya, ada risiko bahwa data SKKM bisa diubah atau dipalsukan, atau bahwa data pribadi mahasiswa yang terkait dengan SKKM bisa bocor atau disalahgunakan. Oleh karena itu, penting untuk mengembangkan strategi dan sistem yang efektif untuk mengelola dan melindungi data ini.

Dalam menyusun bagian mengenai masalah yang sering dihadapi dalam pencatatan data akademik, beberapa temuan dari penelitian terdahulu, yang mencakup periode tahun 2018 hingga 2021, telah diidentifikasi. *A Novel Blockchain-based Education Records Verification Solution* menyebutkan bahwa proses penyimpanan, manajemen, dan otentikasi catatan dan kredensial siswa saat ini sering kali tidak efisien dan rentan terhadap kesalahan dan penyalahgunaan. Sistem ini juga rentan terhadap kerusakan atau kehilangan dokumen, terutama jika otoritas penerbit berhenti beroperasi. Oleh karena itu, diperlukan solusi yang dapat memastikan keandalan dan keamanan catatan pendidikan (Han et al., 2018). *SmartCert BlockChain Imperative for Educational Certificates* menyoroti masalah kurangnya catatan yang dapat diandalkan, dilindungi, dan dipertahankan dalam sistem saat ini untuk mengkonfirmasi sertifikat pendidikan. Selain itu, kehilangan catatan fisik dan risiko pemalsuan data menjadi perhatian utama. Penelitian ini menunjukkan bahwa *blockchain* dapat menjadi solusi untuk mencegah pemalsuan data dan dokumen resmi yang dikeluarkan oleh institusi pendidikan (Kanan et al., 2019). *Academic Records Verification Platform Based on Blockchain Technology* menyoroti masalah konsumsi waktu besar, biaya tinggi, ketergantungan pada pihak ketiga, dan kurangnya transparansi dalam manajemen kredensial akademik. Keterbatasan ini dapat diatasi dengan pengembangan platform verifikasi catatan akademik berbasis *blockchain* yang memungkinkan verifikasi kredensial tanpa bergantung pada pihak ketiga terpusat (Huang, 2020). *Revitalizing blockchain technology potentials for smooth academic records management and verification in low-income countries* menunjukkan bahwa sistem pendidikan tinggi di negara berpenghasilan rendah rentan terhadap lingkungan politik dan ekonomi sekitarnya. Proses verifikasi dan akreditasi sertifikat menjadi panjang dan mahal. Studi ini mengusulkan solusi berbasis *blockchain* untuk manajemen catatan akademik yang dapat mengurangi biaya dan meningkatkan efisiensi (Alnafrah & Mouselli, 2021).

Salah satu solusi yang menjanjikan untuk mengatasi masalah ini adalah menggunakan teknologi *blockchain*. *Blockchain* adalah jaringan digital *peer-to-peer* yang memungkinkan pengelolaan dan pemrosesan transaksi pelanggan tanpa adanya mekanisme kontrol dan manajemen terpusat (Thakur et

al., 2023). Penggunaan *blockchain* dalam pencatatan akademik dapat memberikan manfaat signifikan dalam hal keamanan, integritas, dan transparansi data. Dengan *blockchain*, catatan akademik dapat dijamin tidak dapat dimanipulasi oleh pihak yang tidak berwenang, sementara juga mengurangi ketergantungan pada pihak ketiga dalam proses verifikasi dan penyimpanan data.

Untuk menerapkan penggunaan *blockchain* dalam pencatatan akademik, diperlukan pengembangan sistem yang dapat mengintegrasikan teknologi *blockchain* ke dalam infrastruktur pendidikan tinggi. Ini melibatkan pembuatan aplikasi yang memungkinkan data akademik untuk diakses, dimasukkan, dan disimpan dalam *blockchain* (Pathak et al., 2022). Selain itu, perlu ada mekanisme otomatis untuk mengelola dan memproses data secara efisien dan aman. Salah satu teknologi yang memungkinkan hal ini adalah *smart contract*. Kontrak pintar (*smart contract*) adalah kontrak digital yang dijalankan secara otomatis ketika kondisi-kondisi tertentu terpenuhi. Dalam skripsi ini, *smart contract* digunakan untuk mengelola proses pencatatan SKKM.

Dengan menerapkan teknologi *blockchain* dalam pencatatan data akademik, khususnya pada Satuan Kredit Kegiatan Mahasiswa (SKKM), diharapkan dapat menciptakan sistem yang lebih modern, aman, dan transparan dalam pengelolaan dan pencatatan data pendidikan tinggi. Implementasi *blockchain* pada Satuan Kredit Kegiatan Mahasiswa (SKKM) menawarkan solusi yang lebih baik untuk mengatasi kerentanan sistem saat ini terhadap pemalsuan dan manipulasi data. Teknologi ini, dengan tingkat keamanan yang tinggi dan transparansi yang ia sediakan, memungkinkan pencatatan dan verifikasi data akademik menjadi lebih dapat dipercaya dan efisien. Harapan utama dari penerapan ini adalah pengurangan risiko pemalsuan dan manipulasi data akademik. Penggunaan *blockchain* dapat meningkatkan kepercayaan dalam integritas data, serta mengurangi biaya administrasi yang terkait dengan verifikasi dan penyimpanan data. Ini tidak hanya memberikan keamanan yang lebih baik tetapi juga menjamin transparansi proses, yang sangat penting dalam lingkungan akademis. Dengan demikian, adopsi solusi berbasis *blockchain* dalam manajemen data akademik diharapkan dapat memberikan kontribusi signifikan terhadap peningkatan kualitas pendidikan tinggi. Teknologi ini membuka jalan bagi sebuah sistem yang tidak

hanya lebih efisien tetapi juga lebih adil dan dapat diakses oleh semua pihak yang terlibat dalam ekosistem pendidikan.

1.2 Rumusan Masalah

- Apakah implementasi *blockchain* dapat meningkatkan keamanan data dan mencegah manipulasi data akademik?
- Apakah teknologi *blockchain* dapat mengurangi ketergantungan pada pihak ketiga dalam proses verifikasi dan penyimpanan data akademik?
- Apakah *blockchain* dapat meningkatkan keamanan, integritas, dan transparansi dalam pencatatan data akademik SKKM?

1.3 Batasan Masalah

- Penelitian ini akan mempertimbangkan bagaimana teknologi *blockchain* dapat diintegrasikan ke dalam pencatatan data akademik SKKM, dengan fokus pada peningkatan keamanan, integritas, dan transparansi data.
- Penelitian ini akan mempertimbangkan penggunaan *smart contract* dalam konteks pencatatan dan pengelolaan data akademik SKKM.
- Penelitian ini akan menggunakan *private network* sebagai model jaringan.
- Penelitian ini tidak akan mencakup pengujian langsung terhadap keamanan dan performa *blockchain* secara keseluruhan, tetapi akan fokus pada pengujian keamanan, integritas, dan transparansi data dan fungsi *smart contract* dalam konteks pencatatan data akademik.
- Penelitian ini akan secara khusus membahas implementasi *blockchain* pada kasus data akademik, dengan fokus pada pencatatan Satuan Kredit Kegiatan Mahasiswa (SKKM) di Politeknik Negeri Malang (POLINEMA).

1.4 Tujuan

Tujuan dari dilakukannya skripsi dengan judul "**APLIKASI PENCATATAN AKADEMIK MAHASISWA DENGAN *BLOCKCHAIN***", adalah sebagai berikut:

- Mengatasi potensi manipulasi dan risiko keamanan data dalam pengelolaan data akademik Satuan Kredit Kegiatan Mahasiswa (SKKM) dengan menggunakan teknologi *blockchain*.

- Mengurangi ketergantungan pada pihak ketiga dalam proses verifikasi dan penyimpanan data akademik Satuan Kredit Kegiatan Mahasiswa (SKKM) dengan menggunakan teknologi *blockchain*.
- Meningkatkan keamanan, integritas, dan transparansi data dalam pencatatan data akademik Satuan Kredit Kegiatan Mahasiswa (SKKM) melalui penerapan teknologi *blockchain*, yang mencakup pengembangan sistem dan aplikasi untuk mengakses, memasukkan, dan menyimpan data akademik dalam *blockchain*. Selain itu, penggunaan *smart contract* untuk mendukung proses pencatatan dan verifikasi data akademik, sehingga setiap perubahan dan verifikasi data dapat tercatat dengan transparan dan tidak dapat diubah.

1.5 Manfaat

Manfaat dari penelitian ini, adalah sebagai berikut:

- Memberikan pemahaman yang lebih baik tentang potensi penggunaan teknologi *blockchain* dalam pendidikan tinggi, khususnya dalam pencatatan akademik mahasiswa.
- Meningkatkan keamanan, integritas, dan transparansi data akademik Satuan Kredit Kegiatan Mahasiswa (SKKM) di perguruan tinggi dengan menggunakan teknologi *blockchain*.
- Mengurangi risiko manipulasi data melalui pencatatan yang tidak dapat diubah dan transparan, serta mengurangi ketergantungan pada pihak ketiga dalam proses verifikasi dan penyimpanan data akademik.
- Meningkatkan efisiensi dalam proses pencatatan dan verifikasi data akademik dengan mendukung transparansi dan keandalan melalui penggunaan *smart contract*.
- Menyediakan solusi yang lebih modern dan aman untuk pengelolaan data akademik, yang dapat diadopsi oleh institusi pendidikan tinggi lainnya untuk meningkatkan kualitas dan keamanan layanan pendidikan mereka.

BAB II. LANDASAN TEORI

2.1 Studi Literatur

Beberapa penelitian terdahulu telah mencoba mengatasi masalah keamanan dan integritas data dalam pengelolaan data akademik.

Tabel 2.1 Tabel Studi Literatur

Judul / Penulis	Rangkuman	Hasil	Pemahaman
A Novel Blockchain-based Education Records Verification Solution (Han et al., 2018)	● Mengusulkan solusi verifikasi catatan pendidikan berbasis *blockchain* ● Terintegrasi penyimpanan data terdesentralisasi dan akses aman	● Individu dapat menjadi penjaga catatan pendidikan mereka. ● *Blockchain* memungkinkan penyimpanan data yang terdesentralisasi dan aman.	● Teknologi *Blockchain* dapat digunakan untuk penyimpanan, manajemen, dan otentikasi catatan siswa dan kredensial di pendidikan tinggi. ● *Holberton School of Software Engineering* menggunakan *blockchain* untuk memasukkan sertifikat digital ke dalam *blockchain Bitcoin*.
Academic Records Verification Platform Based on Blockchain Technology (Huang, 2020)	● Mengembangkan platform yang memungkinkan verifikasi catatan akademik secara efisien dan aman menggunakan teknologi *blockchain*. ● Dirancang untuk mengatasi masalah	● Catatan akademik dapat disimpan dalam bentuk yang aman dan tahan terhadap manipulasi. ● Proses verifikasi menjadi lebih cepat dan efisien, karena pihak yang	● Teknologi *blockchain* memiliki potensi untuk merevolusi sektor pendidikan dengan meningkatkan keamanan, transparansi, dan efisiensi dalam

Judul / Penulis	Rangkuman	Hasil	Pemahaman
	pemalsuan dan penipuan dalam catatan akademik serta mempermudah proses verifikasi bagi institusi pendidikan dan pemberi kerja	berwenang dapat mengakses catatan yang telah diverifikasi melalui *blockchain*	pengelolaan dan verifikasi catatan akademik. ● Dalam konteks ini, platform yang dikembangkan dalam artikel ini menunjukkan bagaimana teknologi ini dapat diaplikasikan untuk mengatasi masalah yang ada dalam sistem pendidikan saat ini
Blockchain System in the Higher Education (Raimundo & Rosário, 2021)	● Tinjauan Literatur *Bibliometrik* Sistematis (LRSB) dilakukan ● Mengidentifikasi bidang utama penelitian dan tantangan dalam aplikasi *blockchain* di pendidikan tinggi	● *Blockchain* meningkatkan efisiensi, efektivitas, kontrol privasi, dan keamanan di pendidikan tinggi. ● Solusi yang dibutuhkan untuk kinerja, keamanan, *interoperabilitas*, dan kontrol akses.	● Teknologi *Blockchain* meningkatkan aplikasi pengetahuan dan kepercayaan dalam pendidikan tinggi *online*. ● Ini diintegrasikan ke dalam protokol organisasi dan praktik pembelajaran.
Blockchain to improve Academic Governance (Bhagwat et al., 2020)	● Mengembangkan prototipe yang berfungsi penuh untuk tata kelola akademik ● Menerapkan verifikasi transkrip dan mengusulkan perluasan ke	● Hasil yang menggembirakan dari kasus penggunaan verifikasi transkrip. ● Perluasan untuk mencakup lebih banyak kasus penggunaan	● Teknologi *Blockchain* meningkatkan sistem tata kelola akademik ● Prototipe diperluas untuk mencakup lebih banyak kasus penggunaan

Judul / Penulis	Rangkuman	Hasil	Pemahaman
	kasus penggunaan lainnya	yang diusulkan	
Docschain: Blockchain-Based IoT Solution for Verification of Degree Documents (Rasool et al., 2020)	● *DocsChain* memungkinkan penyerahan massal beberapa dokumen gelar. ● Kamera *IoT/WoT* yang diaktifkan *OCR* digunakan untuk mengumpulkan informasi digital dari fotokopi dokumen gelar.	● Dokumen gelar dapat diverifikasi menggunakan teknologi *OCR*. ● *DocsChain* memungkinkan pengiriman massal beberapa dokumen gelar.	● *DocsChain* adalah solusi berbasis *blockchain* untuk verifikasi gelar. ● Ini memungkinkan penyerahan massal beberapa dokumen gelar.
Revitalizing blockchain technology potentials for smooth academic records management and verification in low-income countries (Alnafrah & Mouselli, 2021)	● Mengusulkan platform berbasis *blockchain* hibrida nasional ● Kelayakan yang diuji di Suriah dan Sudan	● Teknologi *Blockchain* memastikan validitas dan keamanan catatan akademik. ● Solusi yang diusulkan dapat diimplementasikan secara internasional.	● Teknologi *Blockchain* dapat memecahkan masalah merusak catatan akademik. ● Platform *Blockchain* dapat memfasilitasi kerjasama akademik internasional.
UniChain: A Design of Blockchain-Based System for Electronic Academic Records Access and Permissions Management (Daraghmi et al., 2019)	● Desain *UniChain*, sistem berbasis *blockchain* untuk mengelola Catatan Akademik Elektronik (*EAR*) ● Implementasi *UniChain* menggunakan klien *Go*	● Kinerja *UniChain* dievaluasi dengan dataset besar pada latensi rendah. ● Efisiensi proposal ditunjukkan dalam menangani *dataset* besar.	● *UniChain* adalah sistem berbasis *blockchain* untuk mengelola Catatan Akademik Elektronik (*EAR*). ● *UniChain* menyediakan akses aman ke *EAR* sambil menjaga privasi

Judul / Penulis	Rangkuman	Hasil	Pemahaman
	Ethereum dan antarmuka berbasis web *EARS*		siswa.
CredenceLedger: A Permissioned Blockchain for Verifiable Academic Credentials (Arenas & Fernandez, 2018)	● Makalah ini menjelaskan penerapan *Blockchain* yang diizinkan untuk verifikasi kredensial akademik yang terdesentralisasi. ● Solusi yang diusulkan, *CredenceLedger*, menyimpan bukti data ringkas dari kredenSIAL akademik digital dalam buku besar *Blockchain*.	● Hasil yang diperoleh adalah penciptaan sistem *Blockchain* yang diizinkan yang disebut *CredenceLedger*. ● *CredenceLedger* menyimpan bukti data ringkas kredenSIAL akademik digital dalam buku besar *Blockchain* untuk verifikasi yang mudah.	● *Blockchain* adalah jaringan konsensus *P2P* terdesentralisasi. ● *Kriptografi* asimetris digunakan untuk keamanan transaksi.
EduRSS: A Blockchain-Based Educational Records Secure Storage and Sharing Scheme (Li & Han, 2019)	● *EduRSS* adalah skema berbasis *blockchain* untuk penyimpanan dan berbagi catatan pendidikan. Skema ini menggabungkan teknologi *blockchain*, *server* penyimpanan, dan teknik kriptografi untuk menciptakan lingkungan yang aman dan dapat diandalkan	● *EduRSS* menunjukkan bahwa skema ini aman dan memiliki biaya komputasi yang lebih rendah dibandingkan dengan skema *CP-ABE* dan *MA-CPABE*	● *EduRSS* menunjukkan potensi teknologi *blockchain* dalam sektor pendidikan, khususnya dalam hal penyimpanan dan berbagi catatan pendidikan. Namun, masih ada tantangan yang perlu diatasi, seperti integrasi dengan sistem pendidikan yang

Judul / Penulis	Rangkuman	Hasil	Pemahaman
			ada dan hambatan hukum dan regulasi
Blockchain-based Education Project (Guustaaf et al., 2021)	● Melakukan studi literatur sistematis tentang proyek-proyek pendidikan berbasis *blockchain*. ● Mereka berfokus pada protokol yang digunakan dalam proyek ini dan mengeksplorasi beberapa proyek pendidikan tinggi berbasis *blockchain*. ● Selain itu, mereka juga menganalisis fitur *blockchain* yang saat ini digunakan dan layanan yang akan ditawarkan oleh proyek pendidikan yang ada, untuk meningkatkan implementasi teknologi di bidang pendidikan dengan menggunakan fitur *blockchain*	● Teknologi *blockchain* sangat praktis jika digunakan di bidang pendidikan tinggi untuk melakukan sertifikasi digital, pencatatan, dll. ● Proyek-proyek di bidang pendidikan tinggi yang berbasis *blockchain* ditujukan untuk menyelesaikan masalah yang terjadi pada pendidik saat ini. ● Beberapa proyek yang ditemukan antara lain *Sony Global Education*, *Edgecoin*, *APPII*, *Tutellus*, *SuccessLife*, *TeachMePlease*, *GradBase*, *ODEM*, *Blockcerts*, *Parchment*, *Echolink*, dan *Origin-Stamp*	● Teknologi *blockchain* memiliki potensi besar untuk digunakan dalam bidang pendidikan, terutama dalam hal sertifikasi digital dan pencatatan. ● Selain itu, proyek-proyek pendidikan berbasis *blockchain* dapat membantu menyelesaikan berbagai masalah yang dihadapi oleh pendidik saat ini. Namun, masih ada kebutuhan untuk penelitian lebih lanjut untuk mengeksplorasi dan memahami lebih dalam tentang bagaimana teknologi ini dapat diimplementasikan dengan efektif dalam pendidikan
SmartCert BlockChain Imperative for	● Sistem otentikasi elektronik	● Implementasi sistem berbasis	● Teknologi *Blockchain*

Judul / Penulis	Rangkuman	Hasil	Pemahaman
Educational Certificates (Kanan et al., 2019)	menggunakan teknologi *blockchain* diimplementasikan ● Sistem *database* yang kuat untuk dokumen resmi yang dibuat	*blockchain* yang aman ● Peningkatan keamanan dan kerahasiaan dokumen dan sertifikat	memastikan keamanan dan kerahasiaan data. ● Universitas cenderung mengadopsi *Blockchain* untuk keamanan sertifikat.
Application of Blockchain Technology in Online Education (Sun et al., 2018)	● Teknologi *Blockchain* telah terintegrasi dengan pendidikan *online*. ● Temuan penelitian menunjukkan tren perkembangan pendidikan *online*.	● Teknologi *Blockchain* memberikan solusi untuk masalah pendidikan *online*. ● Integrasi teknologi *blockchain* merupakan tren yang menjanjikan dalam pengembangan pendidikan *online*.	● Teknologi *Blockchain* dapat memberikan catatan pembelajaran tepercaya dan tidak dapat dirusak untuk pendidikan *online*. ● Kontrak pintar dapat meningkatkan efisiensi berbagi kursus dalam pendidikan *online*.

Berdasarkan studi literatur di atas, penelitian ini memiliki perbedaan dengan penelitian-penelitian terdahulu yang umumnya membahas penerapan teknologi *blockchain* dalam verifikasi catatan akademik secara luas dan teoritis. Penelitian ini secara khusus meneliti penerapan teknologi *blockchain* pada Satuan Kredit Kegiatan Mahasiswa (SKKM) di Politeknik Negeri Malang. Salah satu perbedaan utama adalah penggunaan *Hyperledger Besu* sebagai platform *blockchain*, yang memungkinkan pembuatan *private network* yang lebih aman dan terkontrol karena akses yang terbatas hanya untuk pihak yang berwenang. Selain itu, penelitian ini juga menggunakan *InterPlanetary File System* (IPFS) untuk penyimpanan terdesentralisasi, yang bertujuan untuk meningkatkan efisiensi dan keamanan dalam proses verifikasi data SKKM. Dengan menyimpan *hash*

sertifikat di *blockchain* dan *file* sertifikat di *IPFS*, keaslian sertifikat dapat diverifikasi tanpa mahasiswa perlu mencetak dan menyimpan dokumen fisik, sehingga pendekatan ini tidak hanya meningkatkan efisiensi proses verifikasi namun juga memperkuat keamanan dan integritas data karena tidak bergantung pada satu *server* pusat. Penggunaan *smart contract* dalam penelitian ini mendukung proses pencatatan dan verifikasi data akademik, memungkinkan otomatisasi proses sehingga setiap perubahan dan verifikasi data dapat tercatat dengan transparan dan tidak dapat diubah. Penelitian ini juga mengembangkan aplikasi berbasis web yang memungkinkan akses, pengelolaan, dan verifikasi data SKKM. Dengan demikian, penelitian ini memberikan kontribusi praktis yang lebih spesifik dan terfokus pada pencatatan data SKKM, berbeda dengan studi literatur yang umumnya bersifat teoritis dan mencakup penerapan teknologi *blockchain* secara umum dalam verifikasi catatan akademik. Harapannya, penelitian ini dapat meningkatkan keamanan, integritas, dan transparansi dalam pencatatan data akademik, serta memberikan model implementasi teknologi *blockchain* yang dapat diadopsi oleh institusi pendidikan tinggi lainnya untuk meningkatkan kualitas layanan mereka.

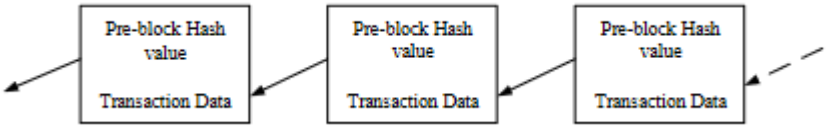
2.2 Dasar Teori

2.2.1 *Blockchain*

Konsep *blockchain* ditemukan oleh seseorang dengan nama samaran Satoshi Nakamoto dalam tesisnya "*Bitcoin: A Peer-to-Peer Electronic Cash System*" yang diterbitkan pada tahun 2008 (Nakamoto, 2008). Dalam tesis ini, Nakamoto mengusulkan sistem *Bitcoin*, tetapi tidak menyebutkan istilah *blockchain*. Sebagai gantinya, blok dan rantai dideskripsikan sebagai struktur data untuk mencatat sejarah buku besar transaksi *Bitcoin*. "Blok" mengacu pada data yang didistribusikan, sedangkan "rantai" berarti rangkaian kronologis blok yang disusun dengan metode kriptografi. Bersama-sama, blok dan rantai membentuk sebuah buku besar transaksi yang berkesinambungan. Secara garis besar, teknologi *blockchain* juga mengacu pada teknik akuntansi terdistribusi berdasarkan struktur *blockchain*, termasuk konsensus terdistribusi, perlindungan privasi dan keamanan, komunikasi *peer-to-peer* (P2P), protokol jaringan, dan kontrak pintar.

Teknologi *blockchain* melibatkan tiga konsep dasar: transaksi, blok, dan rantai. Transaksi adalah operasi buku besar seperti memasukkan atau menghapus sebuah item, yang selalu mengarah pada perubahan status buku besar, blok mencatat hasil dari semua data transaksi selama suatu periode; rantai adalah rangkaian kronologis blok yang mencerminkan semua perubahan status buku besar.

Teknologi *blockchain* diimplementasikan dengan cara berikut. Pertama, harus ada buku besar terdistribusi dalam jaringan yang hanya mengizinkan penambahan data baru. Dengan kata lain, tidak ada data yang dapat dihapus dari buku besar, sehingga memastikan data tidak dapat dirusak. Seperti yang ditunjukkan pada Gambar 2.1, blok-blok dihubungkan ke dalam sebuah rantai dalam urutan kronologis, dengan setiap blok mempertahankan nilai *Hash* dari blok sebelumnya. Ketika sebuah transaksi buku besar baru terjadi, seluruh sistem akan mencatat blok data transaksi, dan menghubungkannya ke rantai melalui algoritma tanda tangan digital kurva elips (*Elliptic Curve Digital Signature Algorithm* (ECDSA)) (Johnson et al., 2001) dalam kriptografi. Dengan cara ini, data tidak dapat dipalsukan atau dipalsukan. Sementara itu, data transaksi akan disiarkan ke seluruh jaringan, dan dikonfirmasi oleh semua *node* jaringan, membuatnya tidak dapat dihapus.



Gambar 2.1 Struktur *Blockchain* (Sun et al., 2018)

Dari sudut pandang teknis, teknologi *blockchain* memiliki beberapa fitur utama. Teknologi ini terdesentralisasi, tidak percaya, dapat diandalkan, dipelihara secara kolektif, privasi aman, dapat dilacak, tidak dapat diubah, dan tidak dapat dipisahkan.

1. Terdesentralisasi: proses verifikasi data, penyimpanan, pemeliharaan, dan transmisi pada *blockchain* didasarkan pada sistem terdistribusi dan bukan sistem terpusat. Sementara itu, kerusakan pada satu *node* tidak akan mempengaruhi data seluruh jaringan (Pilkington, 2015).

2. Tidak percaya: Dalam teknologi *blockchain*, semua blok dihubungkan bersama melalui nilai kriptografi *Hash* di satu sisi, semua *node* dapat melakukan transaksi dengan aman tanpa pengawasan pihak ketiga, di sisi lain (Raths, 2016).
3. Dapat dilacak: Ini berarti bahwa semua transaksi di *blockchain* diatur secara kronologis, dan sebuah blok terhubung dengan dua blok yang berdekatan dengan nilai *hash* kriptografi. Oleh karena itu, setiap transaksi dapat dilacak dengan nilai *hash* kriptografi (Underwood, 2016).
4. Dapat diandalkan: Basis data dalam teknologi *blockchain* mengadopsi penyimpanan terdistribusi, yang berarti setiap *node* dapat memperoleh salinan semua data transaksi. Mode penyimpanan ini melindungi integritas dan keandalan data. Selain itu, setiap data transaksi dicatat dan disimpan pada stempel waktu dan sangat mudah dilacak ke sumbernya (Tschorsch & Scheuermann, 2016).
5. Dipelihara secara kolektif: Data pada *blockchain* dipelihara secara kolektif oleh semua *node*. Kesalahan dari satu *node* tidak berdampak pada data seluruh jaringan (Sun et al., 2018).
6. Privasi aman: Menurut algoritma tanda tangan digital, data ditransmisikan menggunakan *public key* dan *private key*, tanpa mengungkapkan identitas *node*. Pengguna sama sekali tidak terlihat dalam proses transfer. Dengan keandalan dan keamanan yang tinggi, teknologi *blockchain* menawarkan solusi yang ideal untuk masalah pembelajaran *online* (Tschorsch & Scheuermann, 2016).
7. Tidak dapat diubah: *Blockchain* tidak dapat diubah, karena semua transaksi yang tersimpan dalam sebuah blok terhubung dengan blok sebelumnya melalui satu kunci *hash* dan juga terhubung dengan blok berikutnya melalui kunci *hash* yang lain. Mengubah dengan transaksi apapun akan menghasilkan nilai *hash* yang berbeda dan dengan demikian akan terdeteksi oleh semua *node* lain dalam jaringan (Sharples & Domingue, 2016).
8. Tidak dapat dipisahkan: Teknologi *blockchain* dan mata uang digital tidak dapat dipisahkan, dengan kata lain, setiap jaringan *blockchain* memiliki

properti mata uang digital. Oleh karena itu, semua transaksi pada *blockchain* tidak memerlukan partisipasi dari perantara apapun (Raths, 2016).

9. Menjadi Efisien: Efisiensi berarti semua data transaksi secara otomatis dijalankan melalui prosedur yang telah ditetapkan sebelumnya. Oleh karena itu, teknologi *blockchain* dapat mempercepat penyelesaian transaksi keuangan tertentu dengan mengurangi jumlah pihak ketiga yang terlibat (H. Wang et al., 2016).
10. Mekanisme konsensus: Mekanisme konsensus mengacu pada persetujuan bersama dari semua *node* yang terkait dengan jaringan *blockchain* (Grech & Camilleri, 2017). Dengan demikian, ia tidak bergantung pada mediator. *Proof-of-work* (POW), *proof-of-stake* (POS), *delegated-proof-of-stake* (DPOS) adalah beberapa teknik mekanisme konsensus.

Untuk mengimplementasikan aplikasi berbasis *blockchain*, tiga jenis *blockchain* harus dipertimbangkan sesuai dengan kebutuhan aplikasi (Alhadhrami et al., 2017):

1. Publik: *Blockchain* publik sepenuhnya terdesentralisasi dan tanpa izin, yang berarti dapat dilihat oleh semua orang dan tidak ada pemilik tunggal. Proses konsensus terbuka untuk semua orang yang ingin berpartisipasi. Contoh dari *blockchain* publik adalah *Bitcoin*.
2. Pribadi: Jenis *blockchain* ini dimiliki oleh satu entitas tunggal yang mengontrol partisipasi dalam jaringan transaksi. Oleh karena itu, ini juga disebut sebagai *blockchain* berizin. Pada jenis *blockchain* ini, tidak diperlukan algoritma konsensus untuk membuat blok.
3. Hibrida (konsorsium): *Blockchain* ini bersifat publik dan berizin, yang berarti bahwa *blockchain* ini hanya bersifat publik untuk sekelompok peserta tertentu.

Dengan menggunakan *blockchain*, setiap transaksi akademik tidak hanya dijamin keamanannya, tetapi juga dapat ditelusuri dan diverifikasi dengan mudah, memastikan transparansi dan akuntabilitas dalam sistem pendidikan. Keistimewaan lain dari teknologi ini adalah sifatnya yang terdistribusi dan terdesentralisasi, yang dicapai melalui penyimpanan data terdistribusi dan

pemeliharaan kolektif oleh seluruh jaringan. Dibandingkan dengan *database* tradisional yang terpusat, *blockchain* mengurangi risiko kehilangan data yang dapat terjadi akibat serangan siber pada satu titik atau *node* tertentu. Menyusul eksplorasi umum mengenai potensi dan aplikasi *blockchain* dalam konteks pendidikan tinggi, fokus selanjutnya adalah pada pencatatan akademik mahasiswa. Dalam hal ini, *blockchain* menawarkan pendekatan revolusioner yang mengatasi banyak keterbatasan sistem pencatatan tradisional, memastikan bahwa catatan akademik mahasiswa dijaga dengan standar keamanan dan integritas yang tertinggi.

2.2.2 Pencatatan Akademik Mahasiswa

Arsip mahasiswa yang disimpan oleh institusi pendidikan tinggi merupakan bagian dari dokumen yang merekam kegiatan mahasiswa selama proses studinya. Dokumen adalah unit rekaman terkecil yang berisi informasi yang direkam dalam media apapun dan memiliki arti bagi siapa saja yang menafsirkannya. *ISO 15489* (2001) mendefinisikan dokumen sebagai informasi yang terekam, atau objek yang diperlakukan sebagai satu kesatuan. Dokumen dibuat oleh unit kerja suatu organisasi dan akan digunakan sebagai media operasional organisasi dan sumber bukti. Dengan demikian, semua dokumen yang merekam kegiatan mahasiswa dari setiap unit kerja di universitas merupakan bagian dari catatan mahasiswa dan juga akan berfungsi sebagai media operasional organisasi dan sumber bukti perusahaan (Standardization, 2001).

Dokumen ini mencakup informasi seperti data pribadi lain yang diperlukan untuk identifikasi mahasiswa. Data yang disimpan dalam pencatatan akademik mahasiswa mencakup berbagai aspek, seperti

1. Informasi Pribadi Mahasiswa: Data mahasiswa disimpan dalam tabel *database* terpisah dengan kolom-kolom seperti nama lengkap, tanggal lahir, alamat, nomor identifikasi mahasiswa, dan informasi kontak. Akses ke data ini biasanya dikendalikan melalui peran dan izin di dalam *database*.
2. Riwayat Pendidikan: Data pendidikan sebelumnya dan saat ini juga disimpan dalam tabel terpisah dengan kolom-kolom yang sesuai, dan izin akses dikendalikan oleh sistem manajemen basis data.

3. Transkrip Akademik: Informasi tentang kursus, nilai, kredit semester, IPK, dan hasil ujian disimpan dalam tabel terpisah dengan catatan terbaru yang diperbarui. Data ini dapat diakses oleh pihak yang memiliki izin.
4. Pencapaian dan Penghargaan: Data penghargaan akademik, beasiswa, prestasi dalam kegiatan ekstrakurikuler, dan pengakuan lainnya disimpan dalam tabel khusus dengan izin akses yang sesuai.
5. Kehadiran dan Rekam Jejak Akademik: Data tentang kehadiran di kelas, partisipasi dalam kegiatan-kegiatan akademik, dan catatan lainnya disimpan dalam tabel terpisah, dengan izin akses yang sesuai.
6. Kualifikasi Tambahan: Informasi tentang sertifikasi, kursus tambahan, *workshop*, dan kualifikasi profesional disimpan dalam tabel tambahan yang memiliki izin akses.
7. Catatan Keuangan: Data keuangan, termasuk biaya pendidikan, pembayaran, beasiswa, dan bantuan keuangan, disimpan dalam tabel keuangan dengan izin akses yang sesuai.
8. Sertifikasi Akademik dan Dokumen Resmi: Dokumen-dokumen akademik, seperti ijazah, sertifikat kelulusan, dan dokumen akademik resmi lainnya, disimpan dalam direktori atau tabel khusus dengan izin akses yang tepat.
9. Rekomendasi dan Ulasan: Catatan tentang rekomendasi akademik atau profesional, serta ulasan atau *feedback* dari dosen atau staf pengajar, dapat disimpan dalam tabel ulasan yang terkait dengan izin akses yang sesuai.
10. Proyek Akhir dan Penelitian: Informasi tentang skripsi, tesis, proyek penelitian, dan karya ilmiah lain yang dihasilkan oleh mahasiswa, disimpan dalam tabel penelitian terpisah dengan izin akses yang sesuai.

Untuk catatan akademik, sekolah dan universitas memiliki dua tuntutan: menyediakan dan meninjau catatan akademik. Gambar 2.2 menyajikan prosedur verifikasi catatan tradisional untuk universitas dan sekolah. Jika siswa meminta verifikasi berkali-kali, staf sekolah akan mengulangi proses ini berulang kali.

Gambar 2.2 Prosedur Verifikasi Catatan Tradisional Universitas dan Sekolah (Huang, 2020)

Ditambah lagi, kantor pendaftaran perlu mengatur dokumen yang mereka terima setiap kali dalam bentuk yang berbeda dan memeriksanya, transkrip dikirim oleh sekolah. Dengan demikian, agar lebih mudah, dokumen harus dikirim sekali saja. Selain itu, untuk mencegah kebocoran informasi siswa, mereka harus meninjau dokumen pada platform yang aman sehingga privasi siswa tidak akan bocor. Oleh karena itu, sekolah membutuhkan alat yang fasilitatif dan aman untuk menghemat banyak upaya dalam verifikasi siswa.

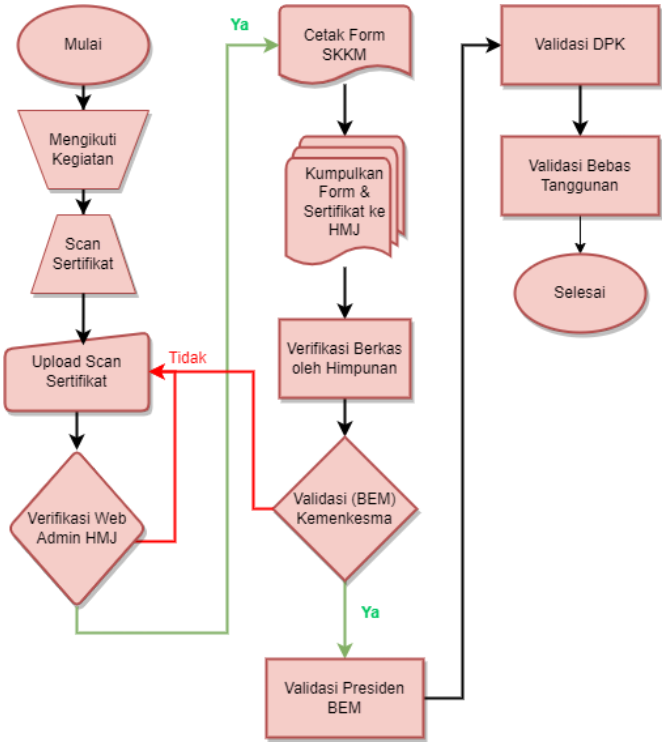
2.2.3 Proses Bisnis Satuan Kredit Kegiatan Mahasiswa (SKKM) di Politeknik Negeri Malang (POLINEMA)

Kegiatan kemahasiswaan mengembangkan kemampuan berpikir, bakat, minat, keterampilan, dan kepribadian seseorang sebagai bagian integral dari proses pembelajaran. Kegiatan kemahasiswaan mengacu pada kegiatan yang bersifat kompetitif baik di tingkat lokal maupun nasional. Kegiatan kemahasiswaan dirancang untuk memenuhi minat dan bakat mahasiswa, meningkatkan nilai, dan meningkatkan kesejahteraan mahasiswa. Sertifikat penghargaan merupakan bukti keikutsertaan dalam kegiatan seperti pelatihan, *workshop*, seminar, dan kompetisi (Utomo et al., 2023).

Proses verifikasi keikutsertaan mahasiswa dalam kegiatan di Politeknik Negeri Malang dimulai dengan mahasiswa yang mengikuti berbagai kegiatan akademik maupun non-akademik yang diselenggarakan di kampus. Setelah kegiatan, mahasiswa harus melakukan scan bukti keikutsertaan kegiatan tersebut. Kemudian, mahasiswa perlu *login* ke *website* SKKM dan memasukkan data yang diperlukan. Langkah berikutnya adalah mencetak *FORM* SKKM setiap semester, yang mana data yang telah dimasukkan dan disimpan di *website* tersebut harus dicetak menggunakan tombol CETAK pada menu POINKU. Setelah mencetak *FORM* SKKM, mahasiswa wajib melampirkan sertifikat asli atau bukti keikutsertaan lainnya seperti piagam, surat keputusan, surat tugas, atau kartu tanda anggota yang telah diverifikasi. Sertifikat atau bukti keikutsertaan ini harus

diserahkan ke Badan Eksekutif Mahasiswa (BEM) untuk proses verifikasi. BEM akan melakukan pengecekan dan memberikan *ACC* atau verifikasi. Setelah itu, mahasiswa dapat mengambil kembali berkas-berkas yang telah diverifikasi dari BEM. Seluruh proses ini diakhiri dengan pengambilan berkas kembali di BEM, menandakan bahwa mahasiswa telah menyelesaikan proses verifikasi keikutsertaan dalam kegiatan yang diwajibkan oleh Politeknik Negeri Malang . Detail langkah-langkahnya adalah sebagai berikut:

1. Mengikuti Kegiatan: Mahasiswa mengikuti kegiatan akademik atau non-akademik.
2. *Scan* Bukti Keikutsertaan: Setelah kegiatan, mahasiswa melakukan scan bukti keikutsertaan.
3. *Login* Web dan Input Data: Mahasiswa *login* ke *website* SKKM dan memasukkan data yang diperlukan.
4. Verifikasi Web *Admin* HMJ: Setelah *upload scan* sertifikat, ada verifikasi oleh web *admin* HMJ (Himpunan Mahasiswa Jurusan).
5. Cetak *Form* SKKM: Mahasiswa mencetak *FORM* SKKM setiap semester.
6. Kumpulkan *Form* & Sertifikat Asli ke HMJ: Mahasiswa mengumpulkan *FORM* SKKM dan sertifikat asli ke HMJ.
7. Verifikasi Berkas oleh Himpunan: Berkas yang dikumpulkan diverifikasi oleh Himpunan.
8. Validasi (BEM) Kemekesma: Setelah verifikasi oleh Himpunan, berkas tersebut diberikan validasi oleh BEM Kemekesma.
9. Validasi Presiden BEM: Presiden BEM memberikan validasi terakhir.
10. Validasi DPK: Berkas yang telah diverifikasi dan divalidasi kemudian diberikan validasi oleh DPK.
11. Validasi Bebas Tanggungan: Sebelum proses selesai, dilakukan validasi bebas tanggungan.



Gambar 2.3 *Flowchart*: Alur Birokrasi SKKM

Proses yang dijelaskan dalam rincian yang diberikan melibatkan beberapa langkah di mana mahasiswa di Politeknik Negeri Malang harus memindai dan menyerahkan bukti partisipasi dalam kegiatan, masuk ke situs web untuk memasukkan data, mencetak formulir setiap semester, dan menyerahkan sertifikat asli atau bukti partisipasi lainnya untuk diverifikasi. Proses ini berpotensi memiliki implikasi keamanan yang dapat diatasi oleh teknologi *blockchain*.

Berikut adalah beberapa alasan mengapa proses yang ada saat ini rentan terhadap masalah keamanan:

1. Penyimpanan Data Terpusat: Data disimpan di *server* terpusat ketika siswa mengunggah bukti partisipasi mereka ke situs web. Sistem yang terpusat dapat menjadi satu titik kegagalan, sehingga menjadi target yang menarik bagi para peretas.
2. Perusakan Data: Dengan sistem terpusat, ada risiko akses yang tidak sah dan merusak data. Misalnya, seseorang berpotensi mengubah catatan partisipasi atau status verifikasi kegiatan.

3. Pemalsuan Dokumen: Dokumen fisik seperti sertifikat atau bukti partisipasi dapat dipalsukan. Meskipun dokumen-dokumen ini dipindai dan diunggah, keaslian dokumen asli mungkin masih dipertanyakan.
4. Proses Verifikasi: Proses verifikasi melibatkan pemeriksaan manusia, yang dapat mengalami kesalahan atau manipulasi. Ada juga risiko kehilangan atau kerusakan dokumen fisik selama proses verifikasi.

Berdasarkan permasalahan tersebut, Mengimplementasikan teknologi *blockchain* dalam proses verifikasi keikutsertaan mahasiswa di Politeknik Negeri Malang dapat memberikan solusi yang lebih aman, transparan, dan efisien. Dengan *blockchain*, data disimpan dalam bentuk yang terdistribusi, tahan terhadap perubahan yang tidak sah, dan proses verifikasi dapat diotomatisasi, mengurangi risiko kesalahan dan penipuan. Ini akan meningkatkan integritas dan keandalan sistem verifikasi SKKM di institusi pendidikan.

2.2.4 *Blockchain* untuk Satuan Kredit Kegiatan Mahasiswa (SKKM)

Penyimpanan catatan tradisional dalam *database* menunjukkan beberapa keterbatasan dalam hal audit dan verifikasi oleh pihak eksternal. Walaupun *blockchain* menyediakan sebuah buku besar yang tidak dapat diubah dan didistribusikan, *blockchain* dapat diperluas untuk menyediakan artefak catatan yang dapat diverifikasi, sementara *peer node* menyimpan salinan buku besar tersebut dan memvalidasi melalui protokol konsensus. Ketika mengadaptasi solusi berbasis *blockchain* untuk aplikasi praktis, ada beberapa tantangan yang perlu dipertimbangkan. Aplikasi *blockchain* memiliki kemungkinan biaya tambahan dan waktu pemrosesan yang lama jika protokol konsensus yang mahal diperlukan. Lebih lanjut, karena kekekalan *blockchain*, harus ada pertimbangan untuk data apa yang disimpan di dalamnya untuk privasi pengguna. Hal ini menginspirasi pendekatan penyimpanan catatan "*off-chain*", menggunakan kombinasi metodologi *on-chain* dan *off-chain* untuk memasukkan hanya apa yang diperlukan pada *blockchain* itu sendiri, seperti *hash* dari catatan tersebut, untuk menghemat ruang dan biaya dan untuk menjaga privasi data siswa (Badr et al., 2019).

Dalam proses tradisional verifikasi Satuan Kredit Kegiatan Mahasiswa (SKKM) di Politeknik Negeri Malang, mahasiswa harus mengikuti serangkaian langkah yang cukup panjang dan melibatkan berbagai entitas organisasi

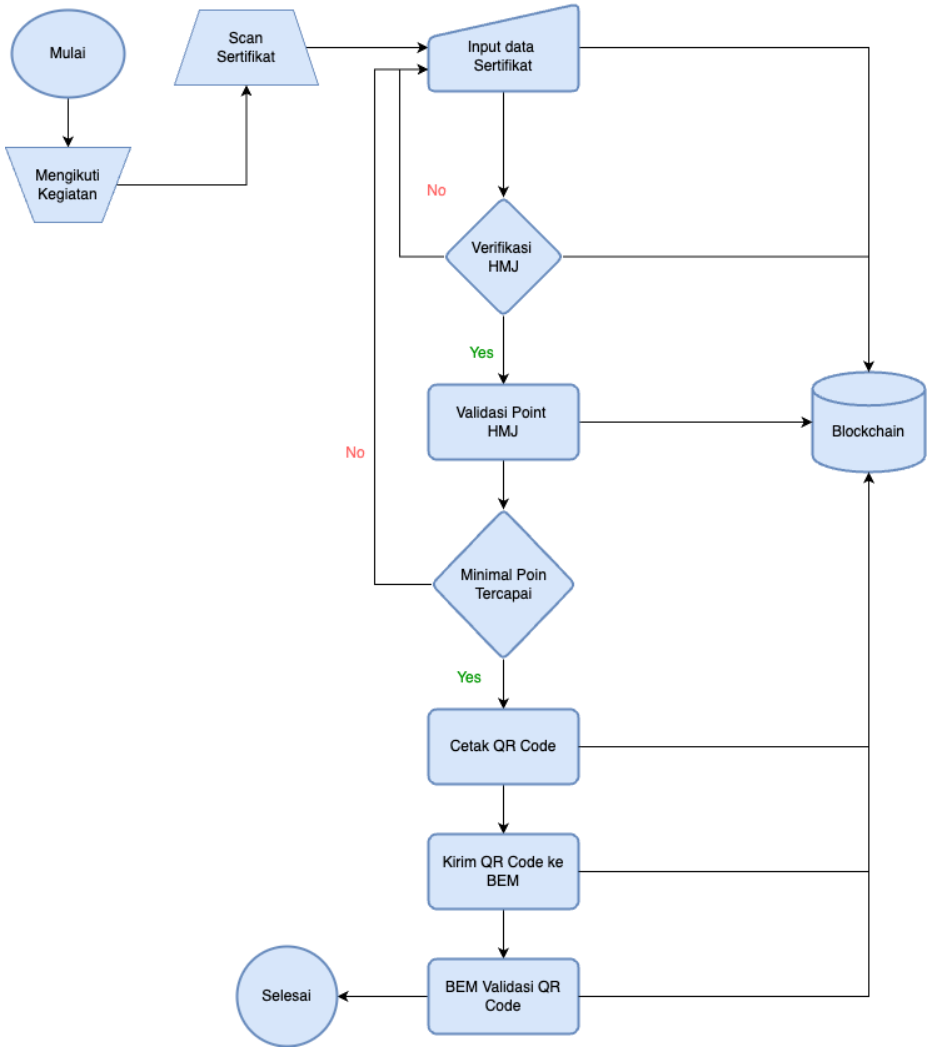
mahasiswa di kampus. Perbedaan utama adalah bahwa data di *blockchain* dienkripsi, memiliki jejak transparan, dan dapat diverifikasi dengan lebih mudah. Selain itu, akses ke data di *blockchain* dikendalikan melalui *smart contract, smart contract* adalah ketentuan perjanjian yang dapat dijalankan sendiri antara dua pihak atau lebih yang ditulis langsung ke dalam kode tanpa memerlukan otoritas pusat, sistem hukum, atau mekanisme penegakan eksternal.

Oleh karena itu, *smart contract* didefinisikan sebagai sebuah kode yang memberlakukan aturan logis untuk mengontrol aset digital. Kode ini berjalan di atas jaringan *blockchain*. *Smart contract* mendefinisikan perjanjian kontrak yang mengontrol siklus hidup transaksi yang diusulkan yang terdapat di negara dunia. Selain itu, *smart contract* mengemas transaksi yang diusulkan secara legal ke dalam kode rantai, yang kemudian disebarkan ke jaringan *blockchain* (Zhao et al., 2021). Sejak tahun 2016, telah ada peningkatan minat terhadap aplikasi *smart contract* berbasis *blockchain*. Semua aplikasi ini berfokus pada *programmability smart contract* selain keamanan, privasi, dan skalabilitas yang memungkinkan untuk mengatur izin dengan lebih canggih dan aman. Ini memberikan tingkat keamanan dan transparansi yang lebih tinggi dalam menyimpan data akademik mahasiswa. Berikut adalah perbandingan antara proses tradisional dengan penggunaan proses *blockchain*:

Tabel 2.2 Perbandingan Proses Tradisional dengan penggunaan Proses *Blockchain*

No	Alur SKKM	Proses Lama	Proses *Blockchain*
1	Mengikuti Kegiatan	✓	✓
2	Scan Bukti Keikutsertaan	✓	✓
3	Login Web dan Input Data	✓	✓
4	Verifikasi Web *Admin* HMJ	✓	X
5	Cetak *Form* SKKM	✓	X
6	Kumpulkan *Form* & Sertifikat Asli ke HMJ	✓	X
7	Verifikasi Berkas oleh Himpunan	✓	✓
8	Validasi (BEM) Kemekesma	✓	✓

No	Alur SKKM	Proses Lama	Proses *Blockchain*
9	Validasi Presiden BEM	✓	X
10	Validasi DPK	✓	X
11	Validasi Bebas Tanggungan	✓	X



Gambar 2.4 Flowchart: Alur Birokrasi SKKM *Blockchain*

Dengan menggunakan *blockchain*, proses verifikasi menjadi lebih efisien, transparan, dan aman. *Blockchain* mengurangi kebutuhan akan verifikasi manual dan validasi oleh berbagai pihak, mengotomatisasi proses dengan *smart contracts*, dan memastikan integritas data dengan teknologi enkripsi dan distribusi. Ini menghilangkan risiko pemalsuan dokumen dan memungkinkan verifikasi yang

cepat dan mudah oleh pihak yang berwenang tanpa perlu mengumpulkan dan memeriksa dokumen fisik.

2.2.5 *Ethereum*

Perjalanan mata uang digital terdesentralisasi dimulai dengan *Bitcoin*, yang diperkenalkan oleh Satoshi Nakamoto pada tahun 2009. Inovasi *Bitcoin* terletak pada dua konsepnya: "*bitcoin*" sebagai mata uang *peer-to-peer* yang terdesentralisasi dan teknologi *blockchain* yang mendasarinya yang memastikan konsensus pada urutan transaksi tanpa otoritas pusat (Buterin, 2014). *Ethereum* dibangun berdasarkan prinsip-prinsip ini, yang bertujuan untuk menggeneralisasi aplikasi *blockchain* di luar mata uang untuk memasukkan berbagai aset digital, kontrak pintar, dan organisasi terdesentralisasi (Buterin, 2014).

Arsitektur dan Komponen *Ethereum*. Arsitektur *Ethereum* memperkenalkan beberapa komponen utama yang membedakannya dari *Bitcoin*:

1. Akun *Ethereum*: Ada dua jenis akun di *Ethereum* - akun yang dimiliki secara eksternal (dikontrol oleh *private key*) dan akun kontrak (dikontrol oleh kode kontraknya). Setiap akun menyimpan saldo *Ether* (ETH), mata uang kripto asli platform ini, dan dapat berinteraksi dengan akun lain melalui transaksi dan pesan (Buterin, 2014).
2. Kontrak Cerdas: Ini adalah kontrak yang dapat dijalankan sendiri dengan ketentuan yang ditulis langsung ke dalam kode. Kontrak ini secara otomatis memberlakukan dan melaksanakan ketentuan perjanjian ketika kondisi yang telah ditentukan terpenuhi. Kontrak pintar memungkinkan aplikasi yang kompleks seperti platform keuangan terdesentralisasi (DeFi), sistem token, dan banyak lagi (Buterin, 2014).
3. Pesan dan Transaksi: *Ethereum* memperluas konsep transaksi dengan menyertakan pesan, yang dapat memanggil fungsi kontrak, mentransfer nilai, dan membawa data. Fleksibilitas ini memungkinkan interaksi yang canggih antara akun dan kontrak (Buterin, 2014).
4. Mesin Virtual *Ethereum* (EVM): *EVM* adalah lingkungan komputasi terdesentralisasi yang menjalankan kontrak pintar. *EVM* memproses *bytecode* kontrak, menangani transisi status, dan memastikan eksekusi

logika kontrak secara deterministik di semua *node* jaringan (Buterin, 2014).

5. *Blockchain* dan Penambangan: Mirip dengan *Bitcoin*, *Ethereum* menggunakan *blockchain* untuk mencatat semua transaksi. Para penambang memvalidasi transaksi, mengamankan jaringan, dan dihargai dengan *ETH* atas usaha mereka. Akan tetapi, *Ethereum* juga menggunakan protokol *GHOST* (Greedy Heaviest Observed Subtree) yang telah dimodifikasi untuk meningkatkan efisiensi dan keamanan validasi blok (Buterin, 2014).

Meskipun *Ethereum* menawarkan kemajuan yang signifikan, *Ethereum* juga menghadapi tantangan seperti skalabilitas, biaya transaksi, dan potensi risiko sentralisasi. Sifat *Turing-complete* dari platform ini memperkenalkan kompleksitas seperti menangani *loop* yang tak terbatas dan memastikan efisiensi komputasi. *Ethereum* mengatasi masalah ini melalui mekanisme seperti biaya gas, yang membatasi langkah komputasi untuk mencegah penyalahgunaan dan memastikan penggunaan sumber daya yang adil. *Ethereum* mewakili evolusi transformatif dalam teknologi *blockchain*, yang memungkinkan spektrum yang luas dari aplikasi terdesentralisasi dan kontrak pintar. Infrastrukturnya yang fleksibel dan kuat membuka jalan bagi inovasi di bidang keuangan, tata kelola, identitas, dan lainnya. Seiring dengan perkembangannya, platform ini menjanjikan untuk mendefinisikan ulang cara kita berinteraksi dengan sistem digital dan mendorong masa depan digital yang lebih terdesentralisasi dan adil (Buterin, 2014).

2.2.6 Kontrak Pintar (Smart Contract)

Kontrak Pintar adalah protokol transaksi yang terkomputerisasi yang mengeksekusi ketentuan dari sebuah perjanjian secara otomatis ketika kondisi tertentu terpenuhi. Konsep ini pertama kali diperkenalkan pada tahun 1990-an oleh Nick Szabo sebagai cara untuk memfasilitasi, memverifikasi, atau menegakkan negosiasi atau kinerja kontrak secara otomatis menggunakan teknologi komputer. Kontrak pintar dapat dianggap sebagai kemajuan besar dalam teknologi *blockchain*. Klausul kontrak yang tertanam dalam *smart contract* akan ditegakkan secara otomatis ketika kondisi tertentu terpenuhi (contohnya,

salah satu pihak yang melanggar kontrak akan dihukum secara otomatis). Kontrak pintar, sebagai sebuah konsep, sudah ada sebelum *Ethereum*, tetapi tidak diimplementasikan secara praktis atau digunakan secara luas dalam lingkungan *blockchain* sampai *Ethereum* diciptakan (Zheng et al., 2020).

Smart contract pada dasarnya diimplementasikan di atas *blockchain*. Klausul kontrak yang disetujui diubah menjadi program komputer yang dapat dieksekusi. Hubungan logis antara klausul kontrak juga telah dipertahankan dalam bentuk aliran logis dalam program (misalnya, pernyataan jika-lalu-jika). Eksekusi dari setiap pernyataan kontrak dicatat sebagai sebuah transaksi yang tidak dapat diubah yang disimpan dalam *blockchain*. Kontrak pintar menjamin kontrol akses dan penegakan kontrak yang tepat. Secara khusus, pengembang dapat memberikan izin akses untuk setiap fungsi dalam kontrak. Ketika kondisi apapun dalam *smart contract* terpenuhi, pernyataan yang dipicu akan secara otomatis menjalankan fungsi yang sesuai dengan cara yang dapat diprediksi. Sebagai contoh, Alice dan Bob menyetujui hukuman atas pelanggaran kontrak. Jika Bob melanggar kontrak, denda yang sesuai (seperti yang ditentukan dalam kontrak) akan secara otomatis dibayarkan (dipotong) dari deposit Bob (Zheng et al., 2020).

Keseluruhan siklus hidup *smart contract* terdiri dari empat fase yang berurutan seperti yang diilustrasikan pada Gambar 2.3:

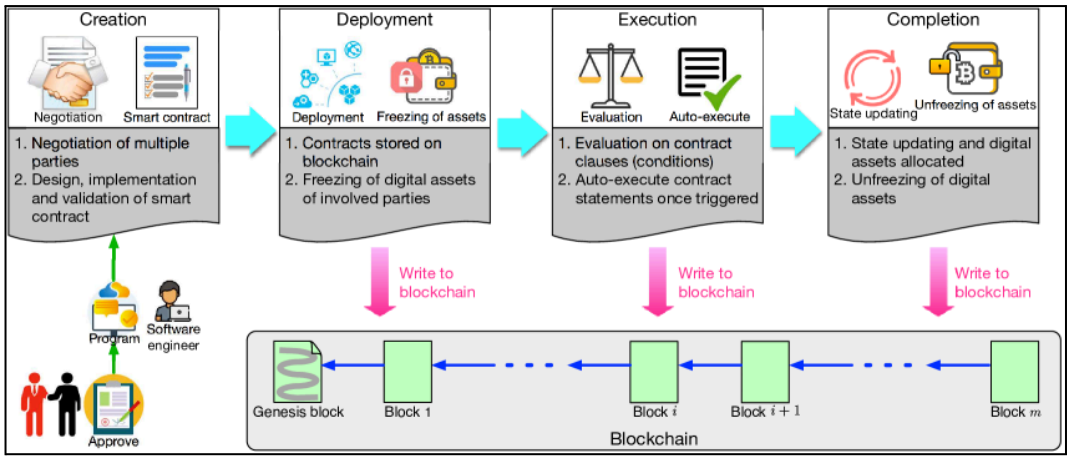
1. Pembuatan kontrak pintar. Beberapa pihak yang terlibat pertama-tama bernegosiasi mengenai kewajiban, hak, dan larangan dalam kontrak. Setelah beberapa putaran diskusi dan negosiasi, kesepakatan dapat dicapai. Pengacara atau konselor akan membantu para pihak untuk menyusun perjanjian kontrak awal. Insinyur perangkat lunak kemudian mengubah perjanjian yang ditulis dalam bahasa alami ini menjadi kontrak pintar yang ditulis dalam bahasa komputer termasuk bahasa deklaratif dan bahasa aturan berbasis logika (Idelberger et al., 2016). Serupa dengan pengembangan perangkat lunak komputer, prosedur konversi *smart contract* terdiri dari desain, implementasi, dan validasi (yaitu pengujian). Perlu disebutkan bahwa pembuatan *smart contract* merupakan proses berulang yang melibatkan beberapa putaran negosiasi dan iterasi.

Sementara itu, proses ini juga melibatkan banyak pihak, seperti pemangku kepentingan, pengacara, dan insinyur perangkat lunak.

2. Penyebaran kontrak pintar. *Smart contract* yang telah divalidasi kemudian dapat digunakan ke platform di atas *blockchain*. Kontrak yang disimpan di *blockchain* tidak dapat dimodifikasi karena kekekalan *blockchain*. Setiap perubahan membutuhkan pembuatan kontrak baru. Setelah *smart contract* digunakan pada *blockchain*, semua pihak dapat mengakses kontrak melalui *blockchain*. Selain itu, aset digital dari kedua belah pihak yang terlibat dalam *smart contract* dikunci melalui pembekuan dompet digital yang sesuai (Sillaber & Waltl, 2017). Sebagai contoh, transfer koin (baik masuk maupun keluar) pada dompet yang relevan dengan kontrak diblokir. Sementara itu, para pihak dapat diidentifikasi melalui dompet digital mereka.
3. Eksekusi kontrak pintar. Setelah penerapan *smart contract*, klausul kontrak dipantau dan dievaluasi. Setelah kondisi kontrak tercapai (misalnya, penerimaan produk), prosedur kontrak (atau fungsi) akan dieksekusi secara otomatis. Perlu dicatat bahwa *smart contract* terdiri dari sejumlah pernyataan deklaratif dengan koneksi logis. Ketika sebuah kondisi dipicu, pernyataan yang sesuai akan dieksekusi secara otomatis, akibatnya transaksi akan dieksekusi dan divalidasi oleh para penambang di dalam *blockchain* (Koulu, 2016). Transaksi yang dilakukan dan status yang telah diperbarui akan disimpan pada *blockchain* setelahnya.
4. Penyelesaian kontrak pintar. Setelah *smart contract* dieksekusi, status baru dari semua pihak yang terlibat diperbarui. Dengan demikian, transaksi selama pelaksanaan kontrak pintar serta status yang diperbarui disimpan dalam *blockchain*. Sementara itu, aset digital telah ditransfer dari satu pihak ke pihak lain (misalnya, transfer uang dari pembeli ke pemasok). Akibatnya, aset digital dari pihak-pihak yang terlibat telah dibuka. Kontrak pintar kemudian telah menyelesaikan seluruh siklus hidup (Zheng et al., 2020).

Perlu disebutkan bahwa selama penerapan, eksekusi, dan penyelesaian *smart contract*, sebuah urutan transaksi telah dieksekusi (masing-masing sesuai

dengan pernyataan dalam *smart contract*) dan disimpan dalam *blockchain*.



Gambar 2.5 Siklus *Smart Contract* (Zheng et al., 2020)

2.2.7 *Hyperledger Besu*

Hyperledger Besu adalah klien *Ethereum* yang mengimplementasikan spesifikasi *Enterprise Ethereum*, yang dikelola oleh *Enterprise Ethereum Alliance* (EEA), sebuah keanggotaan yang terdiri dari *blockchain* dan bisnis yang sudah ada dari seluruh dunia. Awalnya dikenal sebagai *Pegasys Pantheon*, kemudian bergabung dengan *Hyperledger Foundation* dan berganti nama menjadi *Hyperledger Besu* pada tahun 2019 (Fan et al., 2022). *Hyperledger Besu*, adalah proyek *blockchain* pertama yang diajukan ke *Hyperledger* yang dapat beroperasi pada *blockchain* publik. *Besu* mewakili minat perusahaan yang terus meningkat untuk membangun kasus penggunaan jaringan publik dan berizin untuk aplikasi mereka. Keputusan desain dan arsitektur proyek ini ditujukan untuk antarmuka yang bersih dan modularitas, dengan tujuan menjadikan *Hyperledger Besu* sebagai platform untuk pengembangan dan penyebaran terbuka. *Besu* dirancang untuk menjadi semodular mungkin, dengan pemisahan antara algoritma konsensus dan fitur-fitur *blockchain* utama lainnya, membuat komponen-komponen ini mudah untuk di *upgrade* atau diimplementasikan. Dengan menciptakan antarmuka yang bersih antara elemen-elemen dalam klien (misalnya, jaringan, penyimpanan, EVM, dll.) (*Announcing Hyperledger Besu – Hyperledger Foundation*, 2019).

Hyperledger Besu dikembangkan di bawah lisensi Apache 2.0 dan ditulis dalam bahasa *Java*. Ia dapat dijalankan di jaringan publik *Ethereum* atau di jaringan pribadi yang memiliki izin, serta jaringan uji coba seperti *Rinkeby*,

Ropsten, dan *Görli*. *Hyperledger Besu* mencakup beberapa algoritma konsensus termasuk *PoW*, *PoA*, dan *IBFT*, dan memiliki skema perizinan yang komprehensif yang dirancang khusus untuk digunakan dalam lingkungan konsorsium (*Announcing Hyperledger Besu – Hyperledger Foundation*, 2019).

Hyperledger Besu adalah salah satu dari beberapa klien *Ethereum*. Klien *Ethereum* adalah perangkat lunak yang mengimplementasikan protokol *Ethereum*. Klien *Ethereum* berisi (*Announcing Hyperledger Besu – Hyperledger Foundation*, 2019) :

1. Lingkungan eksekusi untuk memproses transaksi dalam *blockchain Ethereum* .
2. Penyimpanan untuk menyimpan data yang terkait dengan eksekusi transaksi.
3. Jaringan *peer-to-peer* (P2P) untuk berkomunikasi dengan *node Ethereum* lainnya di jaringan untuk menyinkronkan status.
4. *API* untuk pengembang aplikasi agar dapat berinteraksi dengan *blockchain*.

Hyperledger Besu mengimplementasikan spesifikasi *Enterprise Ethereum Alliance* (EEA). Spesifikasi *EEA* dibuat untuk membuat antarmuka umum di antara berbagai proyek sumber terbuka dan tertutup dalam *Ethereum*, untuk memastikan pengguna tidak terkunci oleh vendor, dan untuk membuat antarmuka standar untuk aplikasi pembangunan tim. *Besu* mengimplementasikan fitur-fitur perusahaan yang selaras dengan spesifikasi klien *EEA* (*Announcing Hyperledger Besu – Hyperledger Foundation*, 2019). Fitur-fitur *Hyperledger Besu* meliputi:

1. Mesin Virtual *Ethereum* (EVM): *EVM* adalah mesin virtual lengkap Turing yang memungkinkan penyebaran dan eksekusi kontrak pintar melalui transaksi dalam *blockchain Ethereum*.
2. Algoritma Konsensus: *Hyperledger Besu* mengimplementasikan berbagai algoritma konsensus yang terlibat dalam validasi transaksi, validasi blok, dan produksi blok (yaitu penambangan dalam *Proof of Work*). Algoritma tersebut meliputi:
    a. Bukti Otoritas: *Hyperledger Besu* mengimplementasikan beberapa protokol Bukti Otoritas. Protokol konsensus *Proof of Authority*

digunakan ketika para partisipan saling mengenal satu sama lain dan terdapat tingkat kepercayaan di antara mereka - dalam jaringan konsorsium yang memiliki izin, misalnya.

i. *IBFT 2.0*: Dalam jaringan *IBFT 2.0*, transaksi dan blok divalidasi oleh akun yang disetujui, yang dikenal sebagai *validator*. *Validator* bergiliran membuat blok berikutnya. *Validator* yang ada mengusulkan dan memberikan suara untuk menambah atau menghapus *validator*. *IBFT 2.0* memiliki finalitas langsung. Ketika menggunakan *IBFT 2.0*, tidak ada garpu dan semua blok yang valid disertakan dalam rantai utama.

ii. *Clique*: *Clique* lebih toleran terhadap kesalahan daripada *IBFT 2.0*. *Clique* dapat mentolerir hingga setengah dari *validator* yang gagal. Jaringan *IBFT 2.0* membutuhkan lebih dari atau sama dengan ⅔ *validator* yang beroperasi untuk membuat blok. *Clique* tidak memiliki hasil akhir yang langsung. Implementasi yang menggunakan *Clique* harus menyadari adanya *fork* dan reorganisasi rantai yang terjadi.

b. *Proof of Work* (Ethash): *Proof of Work* digunakan untuk aktivitas penambangan di mainnet *Ethereum*.

3. Penyimpanan: *Hyperledger Besu* menggunakan *database* nilai kunci *RocksDB* untuk menyimpan data rantai secara lokal. Data ini dibagi menjadi beberapa sub-kategori:

a. *Blockchain*: Data *blockchain* terdiri dari header blok yang membentuk "rantai" data yang digunakan untuk memverifikasi status *blockchain* secara kriptografis; badan blok yang berisi daftar transaksi yang dipesan yang disertakan dalam setiap blok; dan tanda terima transaksi yang berisi metadata yang terkait dengan eksekusi transaksi termasuk log transaksi.

b. Status Dunia: Setiap *header* blok mereferensikan status dunia melalui *hash stateRoot*. Status dunia adalah pemetaan dari alamat

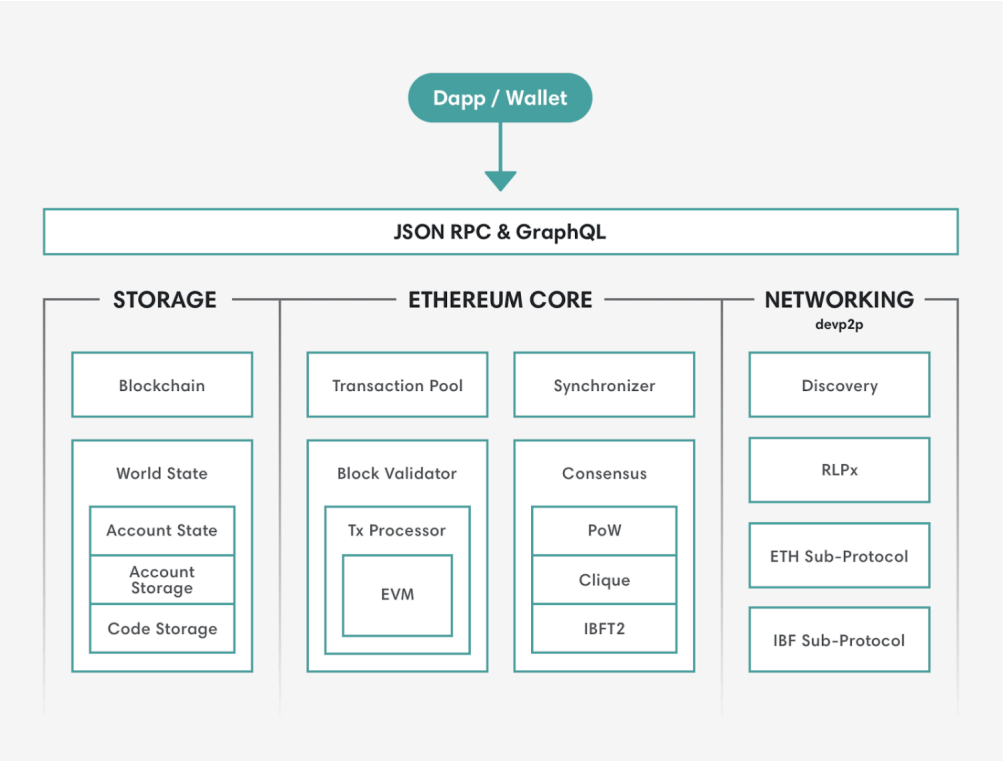
ke akun. Akun yang dimiliki secara eksternal berisi saldo eter, sementara akun *smart contract* juga berisi kode yang dapat dieksekusi dan penyimpanan.

4. Jaringan *P2P*: *Hyperledger Besu* mengimplementasikan protokol jaringan *devp2p Ethereum* untuk komunikasi antar-klien dan sub-protokol tambahan untuk *IBFT2*:
    a. *Discovery*: Protokol berbasis *UDP* untuk menemukan rekan-rekan di jaringan
    b. *RLPx*: Protokol berbasis *TCP* untuk komunikasi antar rekan melalui berbagai "sub-protokol":
        i. Sub-protokol *ETH* (Ethereum Wire Protocol): Digunakan untuk menyinkronkan status *blockchain* di seluruh jaringan dan menyebarkan transaksi baru.
        ii. Sub-protokol *IBF*: Digunakan oleh protokol konsensus *IBFT2* untuk memfasilitasi keputusan konsensus.
5. *API* yang berhadapan dengan pengguna: *Hyperledger Besu* menyediakan *API JSON-RPC* mainnet *Ethereum* dan *EEA* melalui protokol *HTTP* dan *WebSocket* serta *API GraphQL*.
    a. JSON-RPC
        i. Layanan *JSON-RPC HTTP*
        ii. Layanan *JSON-RPC WebSocket*
    b. GraphQL
6. Pemantauan *Hyperledger Besu* memungkinkan untuk memonitor kinerja *node* dan jaringan.
    a. Performa *node* dipantau menggunakan *Prometheus* atau metode *API JSON-RPC debug_metrics*.
    b. Performa Jaringan dipantau dengan alat *Alethio* seperti *Block Explorer* dan *EthStats Network* Monitor.
7. Privasi: Privasi dalam *Hyperledger Besu* mengacu pada kemampuan untuk menjaga kerahasiaan transaksi antara pihak-pihak yang terlibat. Pihak lain tidak dapat mengakses konten transaksi, pihak pengirim, atau daftar pihak

yang berpartisipasi. *Besu* menggunakan *Private Transaction Manager* untuk mengimplementasikan privasi.

8. Perizinan: Jaringan yang diijinkan hanya mengijinkan *node* dan akun tertentu untuk berpartisipasi dengan mengaktifkan izin *node* dan/atau ijin akun di jaringan.

Hyperledger Besu mencakup antarmuka baris perintah serta *API* berbasis *HTTP* dan *WebSocket* untuk menjalankan, memelihara, dan memantau *node* dalam jaringan *Ethereum*. *API* klien *Besu* mendukung fungsionalitas *Ethereum* yang khas seperti kontrak pintar dan pengembangan *dapp*, penyebaran, dan kasus penggunaan operasional. Alat-alat seperti *Truffle*, *Hardhat*, *Remix*, dan *web3js* memungkinkan aktivitas-aktivitas ini. Klien mengimplementasikan *API JSON-RPC standard*, membuat integrasi dengan perkakas ekosistem menjadi sederhana. Klien juga mendukung pembuatan jaringan konsorsium pribadi dan berizin. *Hyperledger Besu* tidak mendukung manajemen kunci di dalam klien karena masalah keamanan. Sebagai gantinya, dapat menggunakan *EthSigner* atau dompet yang kompatibel dengan *Ethereum* untuk mengelola kunci pribadi. *EthSigner* menyediakan akses ke tempat penyimpanan kunci dan menandatangani transaksi melalui alat seperti *Hashicorp Vault* dan *Microsoft Azure*. Izin simpul dan akun berbasis kontrak pintar dan konfigurasi *local* tersedia di dalam *Besu*. Transaksi privat tersedia dengan menggunakan metode tanpa pengetahuan pada klien (termasuk penggunaan protokol *Aztec*). Pendekatan *off-chain* membutuhkan penggunaan *Orion*, sebuah manajer transaksi privat *open source* yang dikembangkan secara terpisah oleh *PegaSys*. Pada tingkat tinggi, arsitektur untuk *Hyperledger Besu* terlihat seperti apa:



Gambar 2.6 Arsitektur *Hyperledger Besu* (*Announcing Hyperledger Besu – Hyperledger Foundation*, 2019)

2.2.8 Private Network

Private network, atau dikenal juga sebagai jaringan *blockchain* privat atau *permissioned blockchain*, adalah jaringan *blockchain* yang beroperasi secara independen dan terisolasi dari jaringan publik (mainnet) seperti *Ethereum*. Akses ke *private network* dibatasi hanya untuk *node-node* yang telah diberikan izin oleh administrator jaringan (*Private Networks | Besu Documentation*, 2024). Karakteristik utama *private network* meliputi:

1. Akses Terbatas (Permissioned): Tidak seperti *blockchain* publik yang terbuka bagi siapa saja, *private network* hanya memperbolehkan entitas yang telah diverifikasi untuk bergabung dan berpartisipasi. Hal ini memberikan kontrol yang lebih besar atas siapa yang dapat mengakses dan berkontribusi pada jaringan, memastikan bahwa hanya pihak yang berwenang yang dapat melihat dan mengelola data sensitif seperti informasi akademik.
2. Privasi: Transaksi dan data yang tersimpan di *private network* bersifat rahasia dan hanya dapat dilihat oleh peserta yang berwenang. Fitur ini

sangat penting untuk melindungi informasi sensitif seperti data akademik mahasiswa, termasuk nilai, transkrip, dan informasi pribadi lainnya.

3. Kontrol Terpusat: Administrator jaringan memiliki otoritas penuh atas *private network*, termasuk pengelolaan *node*, penetapan aturan konsensus, dan konfigurasi parameter jaringan. Hal ini memungkinkan pengaturan yang lebih fleksibel dan sesuai dengan kebutuhan spesifik institusi pendidikan.
4. Skalabilitas Tinggi: *Private network* dapat dioptimalkan untuk mencapai tingkat skalabilitas yang lebih tinggi dibandingkan dengan jaringan publik. Dengan jumlah *node* yang terbatas dan terkendali, *private network* dapat memproses transaksi lebih cepat dan efisien, yang penting untuk aplikasi seperti pencatatan akademik yang membutuhkan kecepatan dan *responsivitas*.
5. Biaya Rendah: Tidak adanya biaya transaksi (gas) yang terkait dengan public *blockchain* membuat *private network* menjadi pilihan yang lebih hemat biaya untuk banyak aplikasi *enterprise*, termasuk di bidang pendidikan.
6. Kustomisasi: *Private network* dapat dengan mudah disesuaikan dengan kebutuhan spesifik organisasi, mulai dari aturan validasi transaksi hingga mekanisme konsensus yang digunakan. Fleksibilitas ini memungkinkan institusi pendidikan untuk mengadaptasi *private network* sesuai dengan kebutuhan unik mereka.

Keunggulan *Private Network* untuk Pencatatan Akademik. Dalam konteks pencatatan akademik mahasiswa, *private network* menawarkan sejumlah keunggulan yang signifikan:

1. Privasi dan Keamanan Data yang Ditingkatkan: *Private network* memberikan perlindungan yang lebih baik terhadap data sensitif mahasiswa, seperti nilai, transkrip, dan informasi pribadi, dengan membatasi akses hanya kepada pihak yang berwenang.
2. Integritas Data: Teknologi *blockchain* yang mendasari *private network* memastikan *immutability* (ketidakberubahan) data, sehingga catatan

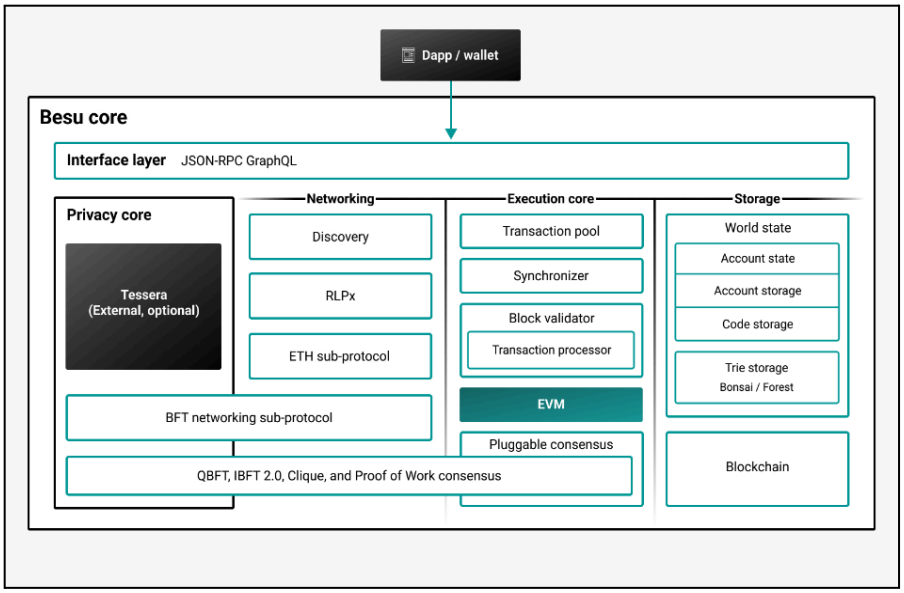
akademik terlindungi dari manipulasi dan pemalsuan, menjamin keakuratan dan keaslian informasi.

3. Efisiensi dan Transparansi: *Private network* dapat mengotomatisasi berbagai proses akademik, seperti verifikasi nilai dan penerbitan transkrip, melalui penggunaan *smart contract*, mengurangi kesalahan manusia dan meningkatkan efisiensi.
4. Kepatuhan terhadap Regulasi: *Private network* memudahkan institusi pendidikan untuk mematuhi peraturan privasi data yang ketat, seperti *GDPR*, dengan membatasi akses ke data dan menerapkan mekanisme kontrol yang ketat.
5. Fleksibilitas dan Pengendalian Penuh: Administrator jaringan memiliki kendali penuh atas *private network*, termasuk kemampuan untuk menyesuaikan aturan konsensus, izin akses, dan berbagai parameter lainnya sesuai kebutuhan spesifik institusi.

Mekanisme Konsensus dalam *Private Network*. Mekanisme konsensus adalah proses yang digunakan untuk mencapai kesepakatan tentang keadaan jaringan *blockchain*, termasuk urutan transaksi yang valid. *Hyperledger Besu* mendukung beberapa mekanisme konsensus yang dapat digunakan dalam *private network*, masing-masing dengan karakteristik dan keunggulannya sendiri (*Private Networks | Besu Documentation*, 2024):

1. *IBFT 2.0* (Istanbul Byzantine Fault Tolerance 2.0): Mekanisme ini adalah pilihan yang direkomendasikan untuk *private network Besu* karena menawarkan finalitas instan (transaksi dianggap valid segera setelah ditambahkan ke *blockchain*), toleransi kesalahan yang tinggi (dapat berfungsi bahkan jika beberapa *node* gagal), dan keamanan yang kuat. *IBFT 2.0* sangat cocok untuk aplikasi *enterprise* yang membutuhkan kepastian dan keandalan.
2. *QBFT* (Quorum-Based Byzantine Fault Tolerance): *QBFT* adalah varian dari *IBFT 2.0* yang dioptimalkan untuk kinerja yang lebih tinggi. *QBFT* juga menawarkan finalitas instan, toleransi kesalahan, dan keamanan yang kuat, menjadikannya pilihan yang baik untuk *private network* yang membutuhkan throughput transaksi yang tinggi.

3. *Clique* (PoA): *Clique* adalah implementasi *Proof of Authority* (PoA) yang digunakan oleh *Geth*, klien *Ethereum* lainnya. Dalam *Clique*, *validator* dipilih berdasarkan jumlah suara yang mereka terima dari *validator* lain. *Clique* lebih toleran terhadap kesalahan daripada *IBFT 2.0*, tetapi tidak memiliki finalitas instan.
4. *Proof of Authority* (PoA): *PoA* adalah mekanisme konsensus yang memberikan otoritas kepada sejumlah *node* tertentu (*validator*) untuk membuat blok baru. *PoA* lebih cepat dan efisien daripada *Proof of Work* (PoW) dan cocok untuk *private network* di mana identitas *validator* diketahui dan dipercaya. *Besu* mendukung dua varian *PoA*: *Clique* dan *IBFT 2.0*.
5. *Ethash* (Proof of Work): *Ethash* adalah mekanisme konsensus yang sama dengan yang digunakan di jaringan publik *Ethereum*. Ethash menggunakan *Proof of Work* (PoW), di mana *node-node* bersaing untuk memecahkan teka-teki kriptografi untuk membuat blok baru. Meskipun *PoW* dapat digunakan di *private network* kecil untuk tujuan pengembangan, mekanisme ini umumnya tidak disarankan untuk *private network* skala besar karena konsumsi energi yang tinggi dan skalabilitas yang terbatas.



Gambar 2.7 Arsitektur *Besu* untuk *Private Network* (*Private Networks | Besu Documentation*, 2024)

Dengan demikian, *private network* dengan *Hyperledger Besu* menawarkan solusi yang ideal untuk aplikasi pencatatan akademik mahasiswa. Kemampuannya dalam menjaga privasi dan keamanan data, memastikan integritas data, meningkatkan efisiensi dan transparansi, serta memberikan kontrol penuh kepada institusi pendidikan menjadikannya pilihan yang tepat untuk mengatasi tantangan dalam sistem pencatatan akademik tradisional. Dalam penelitian ini, mekanisme konsensus *QBFT* (Quorum-Based Byzantine Fault Tolerance) dipilih untuk *private network Hyperledger Besu*. *QBFT* menawarkan finalitas instan, toleransi kesalahan yang tinggi, dan keamanan yang kuat, menjadikannya pilihan yang ideal untuk aplikasi pencatatan akademik yang membutuhkan kepastian dan keandalan. Selain itu, *QBFT* juga memiliki kinerja yang baik, sehingga dapat memproses transaksi dengan cepat dan efisien.

Alasan memilih *QBFT* dibandingkan mekanisme konsensus lainnya seperti *PoW* (Proof of Work), *PoS* (Proof of Stake), *IBFT* (Istanbul Byzantine Fault Tolerance), dan *Clique* adalah sebagai berikut:

- *PoW* (Proof of Work): *PoW* membutuhkan daya komputasi yang sangat tinggi dan tidak efisien untuk lingkungan *private network*. *PoW* juga tidak memberikan finalitas instan dan memiliki waktu pemrosesan transaksi yang lebih lambat.
- *PoS* (Proof of Stake): Meskipun lebih efisien daripada *PoW*, *PoS* tidak menawarkan finalitas instan dan masih bergantung pada partisipasi *staker* yang dapat bervariasi dalam kecepatan pemrosesan.
- *IBFT* (Istanbul Byzantine Fault Tolerance): *IBFT* adalah mekanisme konsensus yang kuat dan tahan terhadap kesalahan, namun tidak memiliki finalitas instan seperti *QBFT*. *IBFT* juga bisa lebih kompleks dalam pengelolaannya dan memiliki waktu latensi yang sedikit lebih tinggi.
- *Clique*: *Clique* adalah mekanisme konsensus yang menggunakan metode *proof-of-authority*, yang lebih cepat dan efisien daripada *PoW* atau *PoS*, namun tidak memiliki toleransi kesalahan sebaik *QBFT*. *Clique* juga tidak memberikan finalitas instan, yang berarti bahwa ada kemungkinan blok-blok bisa berubah sampai mencapai finalitas.

QBFT dipilih karena kemampuannya untuk menawarkan finalitas transaksi instan, yang sangat penting dalam aplikasi pencatatan akademik di mana kepastian dan keandalan data sangat kritis. *QBFT* juga lebih efisien dalam penggunaan sumber daya dan memiliki kinerja yang lebih baik dalam pemrosesan transaksi dibandingkan dengan *IBFT* dan *Clique*. Meskipun *private network* menawarkan berbagai keunggulan, penting untuk memahami bahwa keamanan dan integritas data tetap menjadi perhatian utama. Oleh karena itu, penerapan prinsip-prinsip kriptografi yang kuat sangat penting dalam melindungi data sensitif yang tersimpan di *blockchain*.

2.2.9 Crypto Wallet

Crypto wallet, atau dompet mata uang kripto, adalah perangkat lunak atau perangkat keras yang dirancang untuk menyimpan kunci kriptografi yang diperlukan untuk berinteraksi dengan jaringan *blockchain*. Kunci ini terdiri dari kunci publik dan kunci privat. Kunci publik berfungsi sebagai alamat dompet yang dapat dibagikan untuk menerima aset kripto, sedangkan kunci privat adalah rahasia yang digunakan untuk otorisasi transaksi dan membuktikan kepemilikan aset kripto. Tanpa kunci privat, pengguna tidak dapat mengakses atau mentransfer aset kripto mereka, sehingga keamanan kunci privat menjadi sangat penting (Crypto.com, n.d.). *Crypto wallet* memiliki peran krusial dalam ekosistem *blockchain* karena memungkinkan pengguna untuk menyimpan, mengirim, dan menerima aset kripto secara aman. Selain itu, beberapa dompet juga menawarkan fungsionalitas tambahan seperti interaksi dengan aplikasi terdesentralisasi (dApps) dan *staking* mata uang kripto untuk mendapatkan imbalan. Sebagai contoh, beberapa dompet memungkinkan pengguna untuk berpartisipasi dalam mekanisme konsensus *blockchain* seperti *proof-of-stake* (PoS), di mana mereka dapat mengunci sejumlah aset kripto mereka dalam jangka waktu tertentu untuk mendukung keamanan dan operasional jaringan, dan sebagai imbalannya, mereka menerima hadiah dalam bentuk aset kripto tambahan (Phillips, 2018) .

Jenis-jenis *Crypto Wallet*. *Crypto wallet* dapat dikategorikan menjadi dua jenis utama berdasarkan konektivitasnya dengan internet:

1. Dompet Panas (Hot Wallet):

Dompet panas terhubung ke internet dan dapat berupa aplikasi seluler, ekstensi peramban, atau perangkat lunak desktop. Contoh populer termasuk *MetaMask*, *Exodus*, dan *Trust Wallet*. Keunggulan dompet panas terletak pada kemudahan penggunaan dan kenyamanan untuk transaksi sehari-hari. Dompet ini biasanya memiliki antarmuka pengguna yang ramah dan mudah digunakan, yang membuatnya ideal untuk pengguna baru atau mereka yang sering melakukan transaksi kripto. Karena terhubung ke internet, dompet panas lebih rentan terhadap serangan siber seperti *phishing*, *malware*, dan *hacking*. Oleh karena itu, pengguna perlu menerapkan langkah-langkah keamanan tambahan seperti otentikasi dua faktor (2FA) dan memastikan perangkat mereka bebas dari virus dan *malware* (Sun et al., 2018).

2. Dompet Dingin (Cold Wallet):

   Dompet dingin tidak terhubung ke internet dan biasanya berbentuk perangkat keras fisik seperti *Ledger Nano S* atau *Trezor*, atau bahkan bisa berupa dompet kertas (paper wallet) di mana kunci publik dan kunci privat dicetak di selembar kertas. Karena isolasi dari internet, dompet dingin dianggap lebih aman untuk penyimpanan jangka panjang aset kripto dalam jumlah besar. Dompet ini melindungi kunci privat dari serangan *online*, sehingga mengurangi risiko pencurian digital. Penggunaannya kurang praktis untuk transaksi rutin karena memerlukan langkah tambahan untuk menghubungkan perangkat ke komputer atau mengakses kunci privat dari kertas. Selain itu, pengguna harus berhati-hati untuk tidak kehilangan perangkat keras atau kertas yang berisi kunci privat mereka, karena kehilangan ini dapat menyebabkan hilangnya akses ke aset kripto (Khan et al., 2019).

Fitur-fitur Penting *Crypto Wallet*

1. Keamanan:
   - Enkripsi: Dompet kripto harus menggunakan enkripsi tingkat tinggi untuk melindungi kunci privat dan data pengguna. Enkripsi ini memastikan bahwa hanya pemilik kunci privat yang dapat mengakses dan menggunakan dompet mereka.

- Otentikasi Dua Faktor (2FA): Banyak dompet kripto menawarkan fitur *2FA* untuk menambah lapisan keamanan ekstra. Pengguna harus mengautentikasi identitas mereka melalui dua metode berbeda sebelum dapat mengakses dompet mereka.
- *Backup Seed Phrase* (Frasa Pemulihan): Dompet kripto biasanya memberikan *seed phrase* yang terdiri dari 12-24 kata acak yang dapat digunakan untuk memulihkan dompet jika perangkat pengguna hilang atau rusak. *Seed phrase* ini harus disimpan dengan aman dan tidak dibagikan dengan siapa pun (Crypto.com, n.d.).

2. Kemudahan Penggunaan:
    - Antarmuka Pengguna yang Intuitif: Antarmuka pengguna yang sederhana dan intuitif penting untuk memastikan pengguna dapat dengan mudah menavigasi dan menggunakan dompet kripto mereka. Ini termasuk desain yang ramah pengguna, panduan penggunaan, dan dukungan pelanggan yang responsif.
    - Proses Transaksi yang Sederhana: Dompet kripto harus memungkinkan pengguna untuk melakukan transaksi dengan cepat dan mudah, dengan langkah-langkah yang jelas dan sederhana. Ini termasuk kemampuan untuk mengirim dan menerima aset kripto, memeriksa saldo, dan melihat riwayat transaksi (Phillips, 2018).
3. Dukungan Multi-Mata Uang Kripto:

    Kemampuan untuk menyimpan dan mengelola berbagai jenis mata uang kripto dalam satu dompet akan memberikan fleksibilitas dan kenyamanan bagi pengguna. Dompet multi-mata uang memungkinkan pengguna untuk ber diversifikasi dalam portofolio aset digital mereka. Contoh, Dompet seperti *Exodus* dan *Trust Wallet* mendukung banyak jenis mata uang kripto, termasuk *Bitcoin*, *Ethereum*, *Litecoin*, dan banyak token *ERC-20* lainnya. Ini memungkinkan pengguna untuk mengelola semua aset digital mereka dari satu aplikasi (Suratkar et al., 2020).

4. Integrasi dengan *dApps*:

   Dompet yang terintegrasi dengan platform *dApps* akan memungkinkan pengguna untuk memanfaatkan berbagai layanan dan aplikasi yang dibangun di atas jaringan *blockchain*. *dApps* mencakup berbagai layanan seperti keuangan terdesentralisasi (DeFi), *game*, media sosial, dan banyak lagi. Contoh, *MetaMask* adalah contoh dompet yang terintegrasi dengan ekosistem *Ethereum*, memungkinkan pengguna untuk berinteraksi dengan berbagai *dApps* langsung dari dompet mereka. Pengguna dapat berpartisipasi dalam pinjaman, staking, perdagangan, dan banyak aktivitas lainnya tanpa meninggalkan aplikasi dompet (Phillips, 2018).

5. Staking dan Fitur Tambahan:

   Beberapa dompet menawarkan fitur tambahan seperti *staking*, di mana pengguna dapat mengunci aset kripto mereka untuk mendukung operasi jaringan *blockchain* dan mendapatkan imbalan. *Staking* membantu menjaga keamanan jaringan dan memvalidasi transaksi. *Trust Wallet* dan *Exodus* adalah contoh dompet yang mendukung *staking* untuk berbagai mata uang kripto seperti *Tezos*, *Cosmos*, dan lainnya. Pengguna dapat dengan mudah memulai staking melalui antarmuka dompet dan menerima hadiah staking langsung ke dompet mereka (Suratkar et al., 2020).

Dalam konteks skripsi ini, *crypto wallet* memiliki peran sentral dalam aplikasi pencatatan akademik mahasiswa berbasis *blockchain*. Setiap mahasiswa akan memiliki *crypto wallet* yang terkait dengan identitas digital mereka di *blockchain*. *Wallet* ini akan digunakan untuk menyimpan kunci privat yang memungkinkan mahasiswa untuk mengakses, memverifikasi, dan mengelola data akademik mereka yang tercatat di *blockchain*. Dengan menggunakan *crypto wallet*, mahasiswa memiliki kendali penuh atas data akademik mereka dan dapat memastikan bahwa hanya pihak yang berwenang yang dapat mengaksesnya. Mahasiswa dapat menggunakan kunci privat mereka untuk menandatangani dan memverifikasi keaslian data tersebut, yang kemudian dapat diakses oleh pihak ketiga seperti perusahaan atau institusi pendidikan lain dengan izin mahasiswa. Penggunaan *crypto wallet* dalam sistem ini tidak hanya meningkatkan keamanan

dan privasi data akademik, tetapi juga memungkinkan transparansi dan integritas yang lebih baik. *Blockchain* memastikan bahwa sekali data dicatat, data tersebut tidak dapat diubah tanpa jejak, yang mengurangi risiko manipulasi atau pemalsuan data. Dengan demikian, *crypto wallet* memberikan landasan yang kuat untuk membangun sistem pencatatan akademik yang lebih transparan dan terpercaya. Untuk mendukung implementasi ini, penting untuk memahami konsep *private network*. *Private network* dalam *blockchain* memberikan lapisan keamanan tambahan dengan membatasi akses hanya kepada pihak-pihak yang berwenang.

2.2.10 Desentralisasi dan Kriptografi

Desentralisasi dan Kriptografi, Fondasi Keamanan dan Kepercayaan dalam *Blockchain* untuk Pencatatan Akademik. *Blockchain*, sebagai teknologi yang mendasari *cryptocurrency* dan berbagai aplikasi terdesentralisasi lainnya, memiliki dua pilar utama yang menopang keamanan dan integritasnya: desentralisasi dan kriptografi. Kedua prinsip ini bekerja sama untuk menciptakan sistem yang tahan terhadap manipulasi, sensor, dan titik kegagalan tunggal, menjadikannya solusi yang menjanjikan untuk menjaga keamanan dan integritas data sensitif seperti catatan akademik mahasiswa. Desentralisasi, Membangun Kepercayaan melalui Jaringan Terdistribusi.

Desentralisasi mengacu pada distribusi kontrol dan otoritas di seluruh jaringan, bukan terpusat pada satu entitas tunggal. Dalam konteks *blockchain*, ini berarti tidak ada satu pihak pun yang memiliki kendali penuh atas jaringan, termasuk pemerintah, perusahaan, atau individu (Nakamoto, 2008). Manfaat Desentralisasi:

1. Keamanan yang Ditingkatkan: Desentralisasi membuat *blockchain* lebih tahan terhadap serangan. Untuk mengubah data di *blockchain*, penyerang harus mengendalikan mayoritas *node* di jaringan, yang sangat sulit dilakukan karena *node-node* ini tersebar di berbagai lokasi dan dioperasikan oleh entitas yang berbeda (Zheng et al., 2017).
2. Transparansi yang Lebih Tinggi: Semua transaksi tercatat di buku besar publik yang dapat dilihat oleh siapa saja yang memiliki akses ke jaringan. Ini meningkatkan transparansi dan akuntabilitas, karena setiap perubahan

pada *blockchain* dapat diaudit. Misalnya, dalam konteks pencatatan akademik, semua pihak yang berwenang (mahasiswa, dosen, administrator) dapat melihat riwayat perubahan nilai atau data akademik lainnya, memastikan tidak ada manipulasi data yang tidak sah (Crosby et al., 2016).

3. Ketahanan Sensor yang Kuat: Karena data didistribusikan di seluruh jaringan, *blockchain* lebih tahan terhadap sensor. Bahkan jika beberapa *node* dihapus atau diblokir, *blockchain* akan tetap berfungsi selama mayoritas *node* tetap online. Ini memastikan bahwa catatan akademik mahasiswa tetap dapat diakses dan diverifikasi meskipun ada upaya untuk mengganggu jaringan (W. Wang et al., 2019).

Desentralisasi dalam *Private Network*. Meskipun *private network* lebih terpusat daripada *public blockchain* karena partisipasi dibatasi hanya untuk entitas yang berwenang, prinsip desentralisasi tetap berlaku dalam beberapa aspek. Misalnya, *private network* biasanya memiliki beberapa *node validator* yang bertanggung jawab untuk memvalidasi transaksi dan mencapai konsensus. Ini mencegah satu entitas tunggal untuk mengontrol jaringan secara sepihak dan memastikan integritas data (*Private Networks | Besu Documentation*, 2024).

Kriptografi, Menjaga Keamanan Data dengan Matematika. Kriptografi adalah ilmu tentang teknik matematika yang digunakan untuk mengamankan komunikasi dan data. Dalam *blockchain*, kriptografi digunakan untuk melindungi data dari akses yang tidak sah, memverifikasi keaslian transaksi, dan memastikan integritas data (Narayanan et al., 2016). Teknik Kriptografi yang Digunakan dalam *Blockchain*:

1. *Hashing*: *Hashing* adalah proses mengubah data input (seperti transaksi atau blok) menjadi nilai *hash* yang unik dan berukuran tetap. Hash berfungsi sebagai "sidik jari digital" untuk data tersebut. Setiap perubahan kecil pada data akan menghasilkan nilai *hash* yang sangat berbeda. Ini memungkinkan *blockchain* untuk mendeteksi manipulasi data dengan mudah (Zheng et al., 2017).
2. Tanda Tangan Digital: Tanda tangan digital digunakan untuk memverifikasi keaslian transaksi. Setiap transaksi ditandatangani secara

digital oleh pengirim menggunakan kunci privat mereka, dan tanda tangan ini dapat diverifikasi menggunakan kunci publik pengirim. Ini memastikan bahwa hanya pengirim yang sah yang dapat melakukan transaksi (Narayanan et al., 2016).

3. Enkripsi: Enkripsi digunakan untuk melindungi data dari akses yang tidak sah. Data di enkripsi menggunakan kunci, dan hanya pihak yang memiliki kunci yang sesuai yang dapat mendekripsi dan membaca data tersebut. Dalam konteks pencatatan akademik, enkripsi dapat digunakan untuk melindungi informasi pribadi mahasiswa yang sensitif (W. Wang et al., 2019).

Kriptografi dalam *Private Network*. Kriptografi memainkan peran penting dalam mengamankan *private network*. Hashing digunakan untuk menjaga integritas data, memastikan bahwa catatan akademik tidak dapat diubah tanpa terdeteksi. Tanda tangan digital digunakan untuk memverifikasi keaslian transaksi, memastikan bahwa hanya pihak yang berwenang yang dapat melakukan perubahan pada catatan akademik. Enkripsi, meskipun opsional, dapat memberikan lapisan keamanan tambahan untuk melindungi data yang sangat sensitif (*Private Networks | Besu Documentation*, 2024).

Desentralisasi dan kriptografi adalah dua pilar utama yang menopang keamanan dan kepercayaan dalam *blockchain*, termasuk *private network*. Desentralisasi mendistribusikan kontrol dan otoritas, sementara kriptografi melindungi data dan memverifikasi transaksi. Dengan memahami prinsip-prinsip ini, kita dapat lebih menghargai bagaimana *blockchain* dapat digunakan untuk membangun aplikasi yang aman, transparan, dan dapat dipercaya, seperti aplikasi pencatatan akademik mahasiswa, sehingga memastikan bahwa data akademik mahasiswa terlindungi dengan baik dan dapat diakses hanya oleh pihak-pihak yang berwenang. Penerapan prinsip-prinsip ini, terutama dalam konteks *private network* seperti yang dibangun dengan *Hyperledger Besu*, membutuhkan *tools* dan *framework* yang memudahkan pengembangan aplikasi *blockchain* yang aman dan terukur. Salah satu platform yang menyediakan *tools* dan *framework* yang komprehensif untuk membangun aplikasi *web3*, termasuk aplikasi berbasis *blockchain*, adalah *Thirdweb*.

2.2.11 *Thirdweb*

Thirdweb, Platform Pengembangan *Web3* yang Memberdayakan Aplikasi Pencatatan Akademik yang Aman dan Terdesentralisasi. *Thirdweb*, sebuah platform pengembangan aplikasi *web3* terkemuka, hadir sebagai solusi komprehensif untuk menyederhanakan dan mempercepat proses pembuatan aplikasi *blockchain*. Dalam konteks pencatatan akademik mahasiswa, *Thirdweb* menawarkan sejumlah fitur dan alat yang sangat relevan untuk membangun aplikasi yang aman, terdesentralisasi, dan mudah digunakan. Platform ini menjembatani kesenjangan antara teknologi *blockchain* yang kompleks dengan kebutuhan pengembangan aplikasi yang *user-friendly*, memungkinkan institusi pendidikan untuk memanfaatkan potensi *blockchain* tanpa harus memiliki keahlian teknis yang mendalam (*Thirdweb Docs*, n.d.).

Fitur Utama *Thirdweb*, Mendorong Inovasi dalam Pengembangan Aplikasi *Blockchain*. *Thirdweb* menawarkan rangkaian fitur yang komprehensif yang dirancang untuk menyederhanakan dan meningkatkan setiap tahap pengembangan aplikasi *web3* (*Thirdweb Docs*, n.d.), termasuk :

1. *Smart Contract SDK* yang Kuat dan Fleksibel: *Thirdweb SDK* (Software Development Kit) adalah pustaka perangkat lunak yang menyediakan abstraksi tingkat tinggi untuk berinteraksi dengan *smart contract* di berbagai jaringan *blockchain* yang kompatibel dengan *Ethereum Virtual Machine* (EVM), termasuk *private network Hyperledger Besu*. *SDK* ini menyederhanakan operasi seperti membaca dan menulis data ke *smart contract*, mengirim transaksi, dan menangani *event*, sehingga pengembang dapat fokus pada logika aplikasi mereka. Dengan dukungan untuk berbagai bahasa pemrograman populer seperti *JavaScript* dan *TypeScript*, *SDK* ini memberikan fleksibilitas bagi pengembang untuk memilih bahasa yang paling sesuai dengan preferensi dan keahlian mereka.
2. *Deployment* dan *Hosting Smart Contract* yang Mudah dan Cepat: *Thirdweb* menyederhanakan proses *deployment smart contract* ke berbagai jaringan *blockchain*, termasuk *private network*. Pengembang dapat dengan mudah mengunggah kode *smart contract* mereka ke platform *Thirdweb*, yang kemudian akan menangani proses *deployment* secara otomatis. Selain

itu, *Thirdweb* juga menyediakan layanan hosting yang dapat diandalkan untuk memastikan *smart contract* selalu tersedia dan dapat diakses oleh aplikasi, menghilangkan kebutuhan untuk mengelola infrastruktur *server* sendiri.

3. *Dashboard* Intuitif dan *Analytics* yang Mendalam: *Thirdweb Dashboard* adalah antarmuka yang intuitif dan mudah digunakan yang memungkinkan pengembang untuk memantau kinerja aplikasi mereka secara *real-time*. *Dashboard* ini menampilkan berbagai metrik dan analitik penting, seperti jumlah pengguna aktif, volume transaksi, dan penggunaan *smart contract*. Dengan informasi ini, pengembang dapat mengidentifikasi area yang perlu ditingkatkan, melacak tren penggunaan, dan membuat keputusan berdasarkan data untuk mengoptimalkan kinerja aplikasi.
4. Komponen *UI* yang Siap Pakai dan Dapat Dikustomisasi: *Thirdweb* menyediakan berbagai komponen *UI* (*User* Interface) yang dapat digunakan kembali, seperti tombol koneksi dompet, tampilan *NFT*, dan lainnya. Komponen-komponen ini tidak hanya mempercepat proses pengembangan *frontend*, tetapi juga memastikan konsistensi desain dan pengalaman pengguna yang optimal. Pengembang dapat dengan mudah mengintegrasikan komponen-komponen ini ke dalam aplikasi mereka dan mengkustomisasi tampilan dan perilaku sesuai dengan kebutuhan spesifik mereka.
5. Integrasi Dompet yang Luas dan Aman: *Thirdweb* mendukung integrasi dengan berbagai dompet kripto populer seperti *MetaMask*, WalletConnect, dan *Coinbase Wallet*. Ini memudahkan pengguna untuk terhubung ke aplikasi kita dan berinteraksi dengan *smart contract* menggunakan dompet yang mereka pilih. *Thirdweb* juga menerapkan standar keamanan yang ketat untuk melindungi data pengguna dan mencegah akses yang tidak sah.
6. Otentikasi Pengguna yang Aman dan Fleksibel: *Thirdweb* menyediakan berbagai opsi otentikasi pengguna yang aman, seperti *login* dengan dompet kripto, tanda tangan pesan, dan otentikasi berbasis *email*. Pengembang dapat memilih metode otentikasi yang paling sesuai dengan

kebutuhan aplikasi mereka dan memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data sensitif.

7. Penyimpanan Data Terdesentralisasi dengan *IPFS*: *Thirdweb* terintegrasi dengan *IPFS* (InterPlanetary File System), sebuah protokol penyimpanan terdesentralisasi yang memungkinkan kita untuk menyimpan data aplikasi secara aman dan tahan sensor. Dengan *IPFS*, data tidak disimpan di satu *server* pusat, melainkan didistribusikan ke berbagai *node* di jaringan, meningkatkan ketahanan terhadap serangan dan gangguan. Hal ini sangat penting untuk menjaga integritas dan ketersediaan data akademik mahasiswa dalam jangka panjang.

Keunggulan *Thirdweb* dalam Pengembangan Aplikasi Pencatatan Akademik. Dalam konteks aplikasi pencatatan akademik, *Thirdweb* menawarkan sejumlah keunggulan yang signifikan (*Thirdweb Docs*, n.d.):

1. Kemudahan Penggunaan: *Thirdweb* SDK menyederhanakan kompleksitas pengembangan *blockchain*, memungkinkan institusi pendidikan untuk fokus pada logika bisnis aplikasi tanpa harus memiliki keahlian *blockchain* yang mendalam. Ini berarti institusi dapat lebih cepat mengadopsi teknologi *blockchain* dan mengimplementasikan solusi pencatatan akademik yang efisien.
2. Fokus pada Pengalaman Pengguna: Komponen *UI* yang siap pakai dan dapat dikustomisasi dari *Thirdweb* memungkinkan pengembang untuk membuat antarmuka pengguna yang intuitif dan mudah digunakan bagi mahasiswa, dosen, dan administrator. Antarmuka yang *user-friendly* akan meningkatkan adopsi dan penggunaan aplikasi, sehingga memberikan manfaat yang lebih besar bagi semua pihak yang terlibat.
3. Keamanan dan Privasi: Fitur otentikasi pengguna yang aman dan integrasi dengan *private network Hyperledger Besu* memastikan bahwa data akademik mahasiswa terlindungi dengan baik. Ini sangat penting untuk menjaga kepercayaan antara institusi pendidikan, mahasiswa, dan pihak terkait lainnya.
4. Skalabilitas: *Thirdweb* dirancang untuk mendukung aplikasi yang dapat berkembang seiring dengan pertumbuhan jumlah pengguna dan transaksi.

Ini berarti aplikasi pencatatan akademik yang dibangun dengan *Thirdweb* dapat dengan mudah disesuaikan dengan kebutuhan yang berubah dan meningkatnya jumlah mahasiswa.

5. Integrasi yang Mudah: *Thirdweb* dapat dengan mudah diintegrasikan dengan berbagai teknologi dan platform lain yang digunakan dalam lingkungan pendidikan, seperti sistem informasi akademik yang sudah ada. Hal ini memudahkan proses adopsi dan implementasi *blockchain* dalam sistem yang sudah ada, sehingga mengurangi gangguan pada operasional sehari-hari.

Dalam penelitian ini, *Thirdweb* digunakan sebagai platform pengembangan utama untuk membangun aplikasi pencatatan akademik mahasiswa berbasis *private network Hyperledger Besu*. *Thirdweb* SDK dimanfaatkan untuk menyederhanakan interaksi dengan *smart contract*, komponen *UI* digunakan untuk membangun antarmuka pengguna yang intuitif, dan fitur otentikasi pengguna digunakan untuk memastikan keamanan aplikasi. Selain itu, *Thirdweb* juga memfasilitasi *deployment* dan hosting *smart contract* di *private network*, serta menyediakan *dashboard* dan analitik yang berguna untuk memantau kinerja aplikasi. Dengan memanfaatkan *Thirdweb*, proses pengembangan aplikasi menjadi lebih efisien dan fokus pada implementasi logika bisnis pencatatan akademik, tanpa harus terbebani oleh kompleksitas teknis *blockchain*. Hal ini memungkinkan institusi pendidikan untuk lebih cepat mengadopsi teknologi *blockchain* dan menciptakan solusi pencatatan akademik yang lebih aman, transparan, dan efisien. Lebih lanjut, *Thirdweb* juga menyediakan integrasi dengan *IPFS* (InterPlanetary File System), sebuah protokol penyimpanan terdesentralisasi yang memungkinkan penyimpanan data aplikasi secara aman dan tahan sensor.

2.2.12 *IPFS* (InterPlanetary File System)

InterPlanetary File System (IPFS) adalah protokol dan jaringan *peer-to-peer* (P2P) yang dirancang untuk merevolusi cara kita menyimpan, berbagi, dan mengakses data. *IPFS* menghadirkan paradigma baru dalam pengelolaan data dengan menggantikan pendekatan tradisional *location addressing* (pengalamatan berdasarkan lokasi) dengan *content addressing*

(pengalamatan berdasarkan konten). Dalam sistem tradisional, file diidentifikasi berdasarkan alamat *server* atau *path file*, sedangkan di *IPFS*, setiap *file* diidentifikasi berdasarkan kontennya, yang direpresentasikan dalam bentuk *hash* kriptografi unik yang disebut *Content Identifier* (CID) (Benet, 2014).

Cara Kerja *IPFS* yaitu mendistribusikan Data untuk Ketahanan dan Efisiensi *IPFS* beroperasi sebagai jaringan terdistribusi yang terdiri dari ribuan *node* di seluruh dunia. Ketika sebuah file diunggah ke *IPFS*, file tersebut tidak disimpan di satu *server* pusat, melainkan dipecah menjadi beberapa bagian yang lebih kecil (chunks) dan didistribusikan ke berbagai *node* di jaringan. Setiap *node* menyimpan sebagian dari *file* dan dapat menyediakan bagian tersebut kepada *node* lain yang memintanya (Benet, 2014). Proses pengambilan *file* dari *IPFS* juga unik. Alih-alih meminta *file* berdasarkan alamat *server* atau *path*, pengguna hanya perlu mengetahui *CID* dari *file* tersebut. *CID* ini bertindak sebagai alamat unik yang mengidentifikasi konten *file* secara spesifik. Ketika pengguna meminta *file* dengan *CID* tertentu, *node-node* yang menyimpan bagian-bagian dari *file* tersebut akan merespons dan mengirimkan bagian-bagian tersebut kepada pengguna. *IPFS* kemudian akan menyatukan kembali bagian-bagian tersebut menjadi *file* asli (Benet, 2014).

Keunggulan *IPFS*, Lebih dari Sekadar Penyimpanan. *IPFS* menawarkan sejumlah keunggulan yang membuatnya menjadi solusi penyimpanan yang menarik, terutama dalam konteks aplikasi pencatatan akademik mahasiswa:

1. Desentralisasi dan Ketahanan Terhadap Sensor: Karena data tidak disimpan di satu *server* pusat, *IPFS* lebih tahan terhadap sensor dan gangguan. Bahkan jika beberapa *node offline* atau diblokir, *file* tetap dapat diakses dari *node* lain yang menyimpan salinannya. Ini memastikan bahwa bukti keikutsertaan SKKM mahasiswa tetap tersedia dan tidak dapat dihapus atau dimanipulasi oleh pihak yang tidak berwenang.
2. Integritas Data yang Tinggi: Penggunaan *hash* kriptografi dalam content addressing menjamin integritas data. Setiap perubahan pada *file*, sekecil apapun, akan menghasilkan *CID* yang berbeda. Dengan demikian, keaslian dan keakuratan bukti keikutsertaan SKKM mahasiswa dapat diverifikasi dengan mudah (W. Wang et al., 2019).

3. Efisiensi Penyimpanan: *IPFS* dapat secara signifikan mengurangi duplikasi data. Jika beberapa pengguna mengunggah *file* yang sama, *IPFS* hanya akan menyimpan satu salinan dari *file* tersebut dan memberikan tautan ke salinan tersebut kepada semua pengguna yang memintanya. Ini menghemat ruang penyimpanan dan bandwidth.
4. Ketersediaan Data Jangka Panjang: *IPFS* dapat digunakan untuk menyimpan data secara permanen. *File-file* tidak terikat pada lokasi fisik tertentu dan dapat tetap tersedia selama ada *node* yang menyimpan salinannya. Ini sangat penting untuk menjaga arsip catatan akademik mahasiswa dalam jangka panjang.
5. Dukungan untuk Konten Dinamis: *IPFS* tidak hanya dapat menyimpan file statis, tetapi juga mendukung konten dinamis seperti situs web dan aplikasi web. Ini membuka peluang untuk membangun aplikasi pencatatan akademik yang lebih interaktif dan kaya fitur.

Keterbatasan *IPFS*, Pertimbangan Penting dalam Implementasi. Meskipun memiliki banyak keunggulan, *IPFS* juga memiliki beberapa keterbatasan yang perlu dipertimbangkan:

1. Ketergantungan pada *Node* yang Online: Untuk mengakses file di *IPFS*, setidaknya satu *node* yang menyimpan file tersebut harus online. Jika semua *node* yang menyimpan *file offline*, maka file tersebut tidak dapat diakses. Ini dapat menjadi tantangan dalam menjaga ketersediaan data jangka panjang.
2. Potensi Risiko Keamanan: Seperti halnya teknologi baru lainnya, *IPFS* masih memiliki potensi risiko keamanan yang perlu diatasi. Misalnya, serangan *Sybil*, di mana penyerang membuat banyak *node* palsu untuk memanipulasi jaringan, atau manipulasi *gateway*, di mana *gateway IPFS* yang tidak terpercaya dapat menyajikan konten yang salah.
3. Penggunaan *IPFS Storage Thirdweb* dalam Aplikasi Pencatatan SKKM

Dalam penelitian ini, *IPFS Storage Thirdweb* digunakan untuk menyimpan bukti keikutsertaan mahasiswa, seperti sertifikat atau surat keterangan. *Thirdweb* menyediakan antarmuka yang mudah digunakan untuk berinteraksi dengan *IPFS*, sehingga memudahkan pengembang untuk mengintegrasikan *IPFS* ke dalam

aplikasi mereka. Dengan menggunakan *IPFS* Storage *Thirdweb*, bukti keikutsertaan mahasiswa tersimpan secara terdesentralisasi, terjamin integritas nya, dan dapat diakses dengan mudah oleh pihak yang berwenang. Penggunaan *IPFS* dalam aplikasi pencatatan SKKM ini merupakan langkah penting dalam memastikan keamanan, integritas, dan ketersediaan data jangka panjang. Namun, untuk menghadirkan pengalaman pengguna yang optimal dan antarmuka yang intuitif, diperlukan sebuah *framework frontend* yang handal dan mudah digunakan. Dalam hal ini, *ReactJS* dipilih sebagai solusi yang tepat untuk membangun antarmuka pengguna aplikasi ini.

2.2.13 *React*

React, Mendorong Inovasi Antarmuka Pengguna dalam Aplikasi Pencatatan Akademik Berbasis *Blockchain*. *React*, sebuah pustaka *JavaScript open-source* yang dikembangkan oleh *Facebook*, telah menjadi pilihan utama bagi para pengembang dalam membangun antarmuka pengguna (UI) yang dinamis, interaktif, dan responsif. Dalam konteks aplikasi pencatatan akademik mahasiswa yang terdesentralisasi, *React* memainkan peran penting dalam menghadirkan pengalaman pengguna yang intuitif dan efisien, sekaligus memanfaatkan kekuatan teknologi *blockchain* seperti *Hyperledger Besu*.

Arsitektur Berbasis Komponen, Membangun *UI* yang Modular dan Dapat Digunakan Kembali. Salah satu konsep utama yang membuat *React* begitu populer adalah arsitektur berbasis komponen. Dalam paradigma ini, *UI* dipecah menjadi komponen-komponen yang lebih kecil dan independen, masing-masing dengan logika dan tampilannya sendiri. Komponen-komponen ini dapat dianggap sebagai blok bangunan dasar *UI*, yang dapat disusun dan digabungkan untuk membentuk antarmuka pengguna yang lebih kompleks. Misalnya, dalam aplikasi pencatatan akademik, komponen dapat berupa formulir input nilai, tabel data mahasiswa, grafik visualisasi nilai, atau elemen navigasi (Odeniran et al., 2024). Keuntungan dari arsitektur berbasis komponen ini sangat signifikan. Pertama, komponen dapat digunakan kembali di berbagai bagian aplikasi, mengurangi duplikasi kode dan meningkatkan efisiensi pengembangan. Kedua, karena setiap komponen memiliki tanggung jawab yang jelas, lebih mudah untuk mengidentifikasi dan memperbaiki masalah, sehingga meningkatkan kualitas kode

dan mengurangi risiko terjadinya *bug*. Ketiga, aplikasi yang dibangun dengan komponen lebih mudah untuk diubah dan diperluas seiring dengan pertumbuhan kebutuhan. Misalnya, jika institusi pendidikan ingin menambahkan fitur baru ke aplikasi pencatatan akademik, mereka dapat melakukannya dengan mudah dengan menambahkan komponen baru tanpa harus mengubah seluruh struktur aplikasi (Odeniran et al., 2024).

Pendekatan Deklaratif, Menyederhanakan Pengelolaan *UI* yang Kompleks *React* menggunakan pendekatan deklaratif untuk membangun *UI*. Artinya, pengembang hanya perlu mendeklarasikan bagaimana *UI* seharusnya terlihat berdasarkan state (keadaan) aplikasi. *State* adalah objek *JavaScript* yang menyimpan data yang mempengaruhi tampilan *UI*. Ketika state berubah, *React* secara otomatis memperbarui *UI* tanpa perlu memanipulasi *DOM* (Document Object Model) secara manual (Lazuardy & Anggraini, 2022).

Pendekatan deklaratif ini memiliki beberapa keuntungan. Pertama, memudahkan pengembang untuk memahami dan mengelola kompleksitas *UI*, terutama dalam aplikasi yang besar dan dinamis. Pengembang tidak perlu khawatir tentang bagaimana memperbarui setiap elemen *DOM* secara manual, karena *React* akan menangani hal ini secara otomatis. Kedua, pendekatan deklaratif membuat kode lebih mudah dibaca dan diuji, karena logika *UI* terpisah dari logika manipulasi *DOM* (Lazuardy & Anggraini, 2022).

Virtual DOM, Meningkatkan Performa *Rendering UI*. *React* menggunakan *Virtual DOM* untuk mengoptimalkan kinerja rendering *UI*. *Virtual DOM* adalah representasi ringan dari *DOM* yang sebenarnya, yang disimpan dalam memori. Ketika state aplikasi berubah, *React* akan menghitung perbedaan antara *Virtual DOM* sebelum dan sesudah perubahan, dan hanya memperbarui bagian *DOM* yang benar-benar berubah (Lazuardy & Anggraini, 2022). Penggunaan *Virtual DOM* ini secara signifikan meningkatkan kinerja aplikasi, terutama dalam aplikasi yang sering memperbarui *UI*, seperti aplikasi pencatatan akademik yang menampilkan data *real-time*. Dengan hanya memperbarui bagian *DOM* yang diperlukan, *React* mengurangi jumlah operasi *DOM* yang mahal, sehingga aplikasi dapat berjalan lebih cepat dan responsif (Odeniran et al., 2024).

JSX (*JavaScript* XML), Menulis Kode *UI* yang Lebih Intuitif. *JSX* adalah ekstensi sintaksis untuk *JavaScript* yang memungkinkan pengembang untuk menulis kode yang terlihat seperti *HTML* di dalam *JavaScript*. *JSX* membuat penulisan komponen *React* menjadi lebih intuitif dan mudah dibaca, karena pengembang dapat menulis struktur UI dan logika komponen dalam satu tempat (*React*, n.d.).

React, dengan arsitektur berbasis komponen, pendekatan deklaratif, *Virtual DOM*, dan *JSX*, telah membuktikan dirinya sebagai pustaka *JavaScript* yang andal dan efisien dalam membangun antarmuka pengguna yang dinamis dan interaktif. Dalam konteks aplikasi pencatatan akademik mahasiswa berbasis *blockchain*, *React* berperan penting dalam menciptakan pengalaman pengguna yang intuitif dan responsif, memungkinkan mahasiswa, dosen, administrator, dll untuk berinteraksi dengan data akademik secara mudah dan efisien. Pemanfaatan *React* dalam pengembangan aplikasi pencatatan akademik mahasiswa tidak hanya meningkatkan pengalaman pengguna, tetapi juga membuka peluang untuk mengintegrasikan fitur-fitur canggih seperti visualisasi data yang interaktif, notifikasi *real-time*, dan personalisasi tampilan. Dengan memanfaatkan kemampuan *React* dalam mengelola *state* dan memperbarui *UI* secara efisien, aplikasi pencatatan akademik dapat menyajikan informasi yang relevan dan terkini kepada pengguna, meningkatkan transparansi, dan memfasilitasi pengambilan keputusan yang lebih baik. Keunggulan *React* dalam membangun antarmuka pengguna yang modern dan responsif merupakan langkah penting dalam menciptakan aplikasi pencatatan akademik mahasiswa yang efektif.

BAB III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan di laboratorium Teknologi Informasi, Gedung Pasca Sarjana Politeknik Negeri Malang selama periode Desember 2023 hingga Mei 2024. Rentang waktu ini mencakup seluruh tahapan penelitian, mulai dari perancangan sistem, pengembangan aplikasi, implementasi di *private network Hyperledger Besu*, hingga pengujian dan evaluasi. Data Satuan Kredit Kegiatan Mahasiswa (SKKM) yang digunakan dalam penelitian ini untuk data pribadi tidak berasal dari Politeknik Negeri Malang, melainkan merupakan data simulasi atau data yang dikumpulkan secara independen oleh peneliti. Data yang berasal dari Politeknik Negeri Malang ini mencakup kegiatan wajib dan kegiatan pilihan yang relevan dengan konteks pencatatan akademik mahasiswa.

3.2 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data Satuan Kredit Kegiatan Mahasiswa (SKKM) dari Politeknik Negeri Malang. Data ini mencakup kegiatan wajib dan kegiatan pilihan yang relevan dengan konteks pencatatan akademik mahasiswa, seperti yang dijelaskan di bawah ini:

1. Kegiatan Wajib:
    a. Orientasi Pendidikan (ORDIK)
    b. Latihan Dasar Kepemimpinan (LDK)
    c. Mentoring Keagamaan (sebagai peserta)
2. Kegiatan Pilihan:
    a. Kepengurusan organisasi selain organisasi kemahasiswaan intra
    b. Keanggotaan organisasi internal kampus
    c. Kepanitiaan
    d. Kejuaraan/kompetisi/perlombaan
    e. Penelitian, pengabdian masyarakat, seminar, kuliah tamu, dan kegiatan ilmiah lainnya
    f. Penghargaan akademik dan non-akademik
    g. Hak paten dan hak cipta

h. Pertandingan persahabatan antar kampus/jurusan dengan pihak lain/industri/institusi
i. Kegiatan penunjang akademik

Data SKKM ini diperoleh dari sumber resmi Politeknik Negeri Malang, seperti sistem informasi akademik atau unit kemahasiswaan. Untuk menjaga privasi mahasiswa, data yang digunakan dalam penelitian ini telah dianonimkan, yaitu dengan menghapus atau mengubah informasi yang dapat mengidentifikasi individu mahasiswa secara langsung, seperti nama, NIM, atau informasi pribadi lainnya.

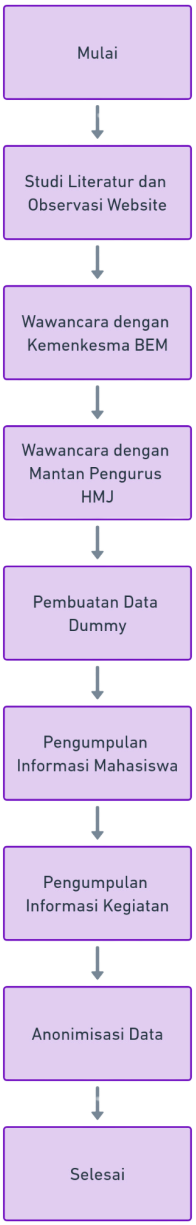
3.2.1 Proses Pengumpulan Data

Berikut adalah proses pengumpulan data :

1. Studi Literatur dan Observasi *Website*: Peneliti melakukan studi literatur terkait regulasi dan pedoman SKKM di Politeknik Negeri Malang. Selain itu, peneliti juga mengamati *website* resmi SKKM Polinema (https://skkm.polinema.ac.id/) untuk memahami jenis kegiatan yang diakui, bobot SKS, serta prosedur pengajuan dan validasi SKKM.
2. Wawancara dengan Narasumber:
    a. Kemenkesma BEM: Peneliti melakukan wawancara dengan perwakilan dari Kementerian Kesejahteraan Mahasiswa (Kemenkesma) BEM Polinema untuk mendapatkan informasi lebih lanjut mengenai proses validasi SKKM, kriteria penilaian, dan kendala yang sering dihadapi dalam proses tersebut.
    b. Mantan Pengurus HMJ: Peneliti juga mewawancarai mantan pengurus Himpunan Mahasiswa Jurusan (HMJ) untuk mendapatkan perspektif dari pihak yang pernah terlibat langsung dalam proses verifikasi SKKM di tingkat jurusan. Wawancara ini bertujuan untuk menggali informasi mengenai tantangan dan potensi perbaikan dalam proses verifikasi SKKM.
3. Data *Dummy*: Berdasarkan informasi yang diperoleh dari studi literatur, observasi *website*, dan wawancara, peneliti membuat data *dummy* SKKM yang mencakup:

a. Informasi Mahasiswa: NIM (Nomor Induk Mahasiswa) fiktif, nama jurusan, dan angkatan.
b. Informasi Kegiatan: Nama kegiatan, jenis kegiatan (wajib atau pilihan), tanggal pelaksanaan, jumlah kredit poin, dan bukti keikutsertaan (dalam bentuk dokumen digital simulasi) dll.

Data *dummy* ini digunakan untuk menguji dan mengevaluasi aplikasi pencatatan SKKM berbasis *blockchain* yang dikembangkan dalam penelitian ini. Setelah data-data tersebut terkumpul, langkah selanjutnya adalah melakukan pengolahan data untuk mendapatkan informasi yang relevan dan mendukung analisis penelitian.



Gambar 3.1 Flowchart: Proses Pengumpulan Data SKKM

3.3 Teknik Pengolahan Data

Dalam penelitian ini, teknik pengolahan data digunakan untuk mengolah dan menganalisis data yang telah dikumpulkan melalui berbagai metode seperti studi literatur, observasi *website* SKKM Polinema, dan wawancara dengan narasumber. Langkah-langkah yang dilakukan dalam teknik pengolahan data ini bertujuan untuk mendapatkan informasi yang relevan dan mendukung analisis penelitian.

3.3.1 Reduksi Data

Tahap reduksi data dilakukan untuk menyaring dan merangkum data yang telah dikumpulkan sehingga hanya data yang relevan dan penting yang akan dianalisis lebih lanjut. Proses reduksi data meliputi beberapa langkah berikut:

1. Penyaringan Data: Data yang tidak relevan atau tidak berhubungan langsung dengan fokus penelitian disaring dan dihapus.
2. Pengelompokan Data: Data yang telah disaring kemudian dikelompokkan berdasarkan kategori yang relevan dengan penelitian. Kategori-kategori ini termasuk jenis kegiatan SKKM (wajib dan pilihan), tahapan proses SKKM, serta masalah dan kendala yang dihadapi dalam proses tersebut.
3. Penghapusan Data Duplikat: Data yang terduplikasi akan dihapus untuk memastikan keakuratan dan konsistensi data yang digunakan dalam analisis.

3.3.2 Penyajian Data

Setelah data direduksi, langkah selanjutnya adalah menyajikan data dalam format yang mudah dipahami dan dianalisis. Penyajian data dilakukan dengan cara berikut:

1. Pembuatan Tabel: Data yang telah dikelompokkan disusun dalam bentuk tabel untuk memudahkan identifikasi dan perbandingan antar kategori. Misalnya, tabel yang mencantumkan jenis kegiatan SKKM, jumlah kredit poin, dan frekuensi kegiatan.
2. Pembuatan Grafik dan Diagram (opsional): Data divisualisasikan dalam bentuk grafik atau diagram untuk memperjelas distribusi dan hubungan antar data. Contoh visualisasi meliputi diagram batang untuk menunjukkan frekuensi jenis kegiatan SKKM, diagram lingkaran untuk menggambarkan proporsi kegiatan wajib dan pilihan, serta diagram alur untuk memvisualisasikan tahapan proses SKKM.
3. Deskripsi Naratif: Selain visualisasi, data juga dijelaskan secara naratif untuk memberikan konteks dan pemahaman yang lebih baik terhadap data tersebut. Deskripsi ini mencakup penjelasan mendetail mengenai setiap kategori data dan bagaimana data tersebut berhubungan dengan penelitian.

3.3.3 Analisis Data

Setelah data disajikan, tahap berikutnya adalah melakukan analisis data untuk mengidentifikasi pola, tema, dan wawasan yang relevan dengan tujuan penelitian. Analisis data dilakukan melalui beberapa pendekatan kualitatif berikut:

1. Analisis Tematik:
    - Identifikasi Tema: Data dari studi literatur, observasi *website*, dan wawancara dianalisis untuk mengidentifikasi tema-tema utama terkait proses SKKM. Tema-tema ini termasuk tahapan proses SKKM, peran masing-masing pihak yang terlibat, kendala yang dihadapi, dan harapan pengguna terhadap sistem pencatatan SKKM.
    - Pengkodean Data: Data diberi kode berdasarkan tema yang telah diidentifikasi. Pengkodean ini membantu dalam menganalisis lebih lanjut untuk menemukan pola atau hubungan antara tema-tema tersebut. Setiap bagian data yang relevan akan diberi kode sesuai dengan tema yang diidentifikasi untuk memudahkan analisis mendalam.
2. Analisis SWOT:
    - *Strengths* (Kekuatan): Mengidentifikasi kekuatan dalam penerapan *blockchain* untuk pencatatan SKKM, seperti keamanan data dan transparansi.
    - *Weaknesses* (Kelemahan): Mengidentifikasi kelemahan atau keterbatasan yang mungkin dihadapi dalam penerapan *blockchain*, seperti biaya implementasi, kebutuhan teknologi yang tinggi, atau resistensi dari pihak yang terlibat.
    - *Opportunities* (Peluang): Mengidentifikasi peluang yang bisa dimanfaatkan untuk meningkatkan proses SKKM dengan teknologi *blockchain*, seperti dukungan dari pihak akademik atau peluang pengembangan lebih lanjut.
    - *Threats* (Ancaman): Mengidentifikasi ancaman atau risiko yang mungkin muncul dalam penerapan *blockchain*, seperti masalah regulasi, keamanan siber, atau resistensi terhadap perubahan.
3. Analisis Konten:

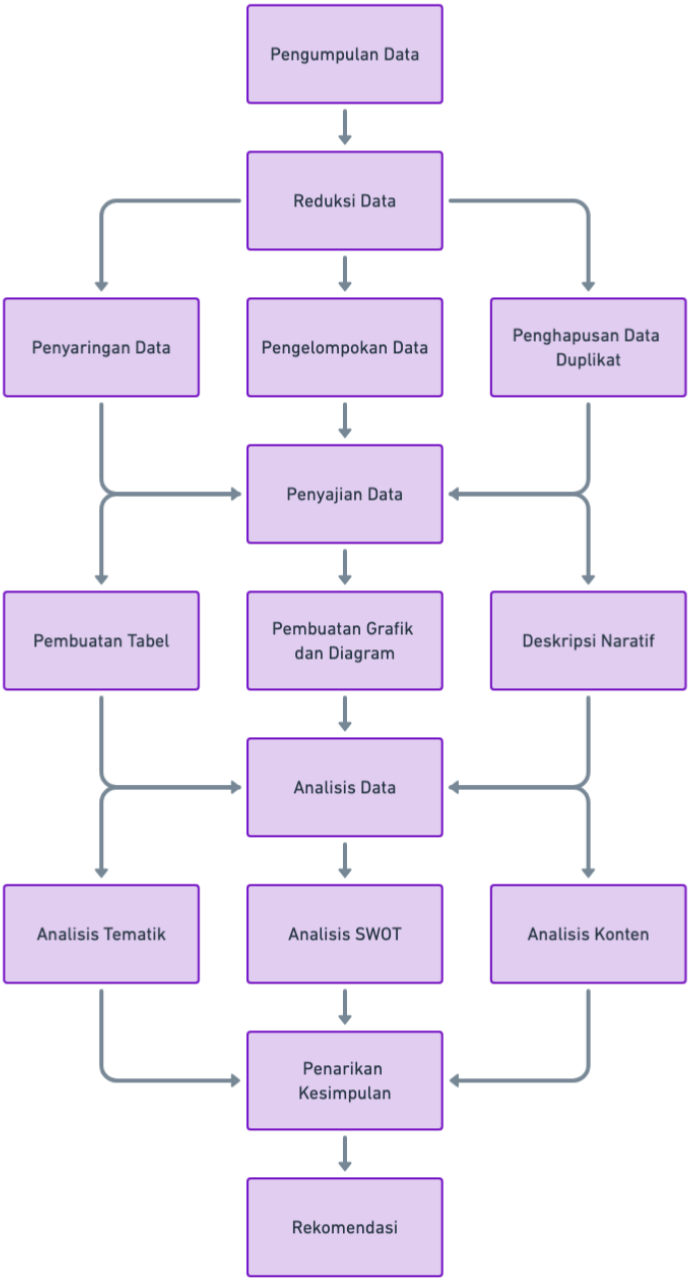
- Pendalaman Konten: Data dari wawancara dan data *dummy* dianalisis secara mendalam untuk mengidentifikasi pandangan, pendapat, dan pengalaman narasumber terkait proses SKKM. Analisis ini akan mencakup pemahaman mendalam tentang bagaimana narasumber melihat proses SKKM saat ini dan harapan mereka terhadap perbaikan dengan teknologi *blockchain*.
- Interpretasi Data: Menafsirkan data untuk memahami makna di balik pandangan dan pengalaman narasumber. Interpretasi ini juga mencakup evaluasi pengalaman pengguna dan tingkat kepuasan terhadap aplikasi pencatatan SKKM berbasis *blockchain* yang diuji dengan data *dummy*.

3.3.4 Penarikan Kesimpulan dan Rekomendasi

Berdasarkan hasil analisis data, langkah terakhir adalah menarik kesimpulan dan memberikan rekomendasi yang relevan dengan tujuan penelitian. Langkah-langkah ini meliputi:

1. Kesimpulan: Berdasarkan analisis tematik, SWOT, dan konten, saya akan menarik kesimpulan mengenai efektivitas penggunaan *blockchain* dalam pencatatan SKKM. Kesimpulan ini akan mencakup manfaat yang diperoleh, seperti peningkatan keamanan dan transparansi data, serta tantangan dan keterbatasan yang dihadapi dalam implementasinya.
2. Rekomendasi: Berdasarkan temuan penelitian, saya akan memberikan rekomendasi untuk perbaikan atau implementasi sistem *blockchain* dalam pencatatan SKKM di masa mendatang. Rekomendasi ini mencakup saran untuk mengatasi tantangan yang diidentifikasi, strategi untuk memanfaatkan peluang yang ada, dan langkah-langkah untuk meningkatkan penerimaan dan dukungan dari pihak yang terlibat.

Dengan mengikuti urutan langkah-langkah ini, saya berharap penelitian ini dapat memberikan pemahaman yang komprehensif dan mendalam tentang penerapan *blockchain* dalam pencatatan SKKM, serta menghasilkan sistem yang lebih efektif, efisien, transparan, dan aman.



Gambar 3.2 Flowchart: Proses Pengolahan Data SKKM

3.4 Desain Sistem

Desain sistem pencatatan SKKM berbasis *blockchain* ini merupakan hasil dari analisis mendalam terhadap proses SKKM yang ada di Politeknik Negeri Malang. Sistem ini bertujuan untuk mengatasi masalah-masalah yang teridentifikasi, seperti kurangnya transparansi, sentralisasi, dan potensi manipulasi

data, dengan memanfaatkan teknologi *blockchain* yang aman, transparan, dan efisien.

3.4.1 Arsitektur Sistem

Arsitektur sistem ini dirancang dengan cermat untuk mengintegrasikan berbagai komponen teknologi yang saling mendukung, menciptakan ekosistem pencatatan SKKM yang handal dan terdesentralisasi. Berikut adalah komponen-komponen utama dalam arsitektur sistem ini:

1. *Node Hyperledger Besu*: *Node-node Besu* merupakan tulang punggung dari *private network* yang digunakan dalam sistem ini. *Node-node* ini bertanggung jawab untuk menjalankan *private network*, menyimpan salinan *blockchain*, dan mengeksekusi *smart contract*. Politeknik Negeri Malang dapat mengelola *node-node* ini di *server* internal mereka, memastikan kontrol penuh atas jaringan dan data yang tersimpan.
2. *Smart Contract*: *Smart contract*, yang ditulis dalam bahasa *Solidity*, merupakan inti dari sistem ini. Kontrak pintar ini berisikan logika bisnis yang mengatur seluruh proses pencatatan SKKM, mulai dari pendaftaran kegiatan, pengajuan bukti keikutsertaan, verifikasi oleh HMJ (Himpunan Mahasiswa Jurusan), validasi oleh BEM (Badan Eksekutif Mahasiswa), hingga pembuatan dan validasi *QR Code*. *Smart contract* ini memastikan otomatisasi dan integritas data, sehingga setiap langkah dalam proses SKKM tercatat secara transparan dan tidak dapat diubah.
3. Aplikasi Web (Frontend): Aplikasi web berbasis *React JS* menyediakan antarmuka pengguna yang intuitif dan responsif bagi mahasiswa, HMJ, dan BEM. Mahasiswa dapat dengan mudah mengakses aplikasi ini melalui perangkat mereka untuk mendaftar kegiatan, mengunggah bukti keikutsertaan, melihat status pengajuan, dan menghasilkan *QR Code*. HMJ dan BEM dapat menggunakan aplikasi ini untuk memverifikasi dan memvalidasi pengajuan SKKM, serta mengelola data kegiatan dan mahasiswa.
4. *Thirdweb SDK*: *Thirdweb SDK* adalah perangkat pengembangan perangkat lunak (SDK) yang menyederhanakan interaksi antara aplikasi web dengan *smart contract* dan *blockchain*. *SDK* ini menyediakan fungsi-fungsi yang

mudah digunakan untuk membaca dan menulis data ke *smart contract*, mengirim transaksi, dan menangani *event*, sehingga mempercepat proses pengembangan aplikasi.

5. *IPFS Storage*: *InterPlanetary File System* (IPFS) adalah protokol penyimpanan terdesentralisasi yang digunakan untuk menyimpan bukti keikutsertaan mahasiswa. Dengan menggunakan *IPFS*, bukti-bukti ini didistribusikan ke seluruh jaringan, sehingga tidak bergantung pada satu *server* pusat dan lebih tahan terhadap kerusakan atau kehilangan data. Selain itu, *IPFS* memastikan integritas data dengan menggunakan *content addressing*, di mana setiap *file* diidentifikasi dengan *hash* unik yang tidak dapat diubah.

3.4.2 Gambaran *Smart Contract*

Smart contract dalam sistem pencatatan SKKM berbasis *blockchain* ini dirancang untuk mengotomatiskan dan meningkatkan transparansi dalam proses pengajuan, verifikasi, dan validasi kegiatan mahasiswa. Kontrak pintar ini berjalan diatas *private network Hyperledger Besu*, memanfaatkan teknologi *blockchain* untuk menjamin integritas, keamanan, dan keaslian data.

1. Data yang Dikelola oleh *Smart Contract*:
    a. Data Pengguna (*User*): Data ini mencakup informasi pengguna dalam sistem, termasuk mahasiswa, admin, HMJ, dan BEM. Informasi yang dikelola mencakup nama lengkap, pengidentifikasi unik (misalnya NIM), jurusan atau departemen, peran dalam sistem, dan status apakah pengguna terdaftar dalam sistem.
    b. Data Pengajuan SKKM (*SKKMRequest*): Data ini mencakup informasi terkait pengajuan SKKM oleh mahasiswa. Informasi yang dikelola mencakup nama kegiatan dalam Bahasa Indonesia dan Inggris, nomor sertifikat, kategori dan jenis kegiatan, prestasi yang dicapai, dasar penilaian, *hash* dari sertifikat yang disimpan di *IPFS*, jumlah poin kredit yang diperoleh, status pengajuan, pesan jika SKKM tidak terverifikasi, dan waktu pengajuan.

c. Data *QR Code*: Data ini mencakup informasi terkait *QR Code* yang dihasilkan untuk SKKM. Informasi yang dikelola mencakup data *QR Code*, status *QR Code*, dan waktu pembuatan *QR Code*.
d. Pemetaan (Mappings):
    - Pemetaan alamat dompet ke data pengguna.
    - Pemetaan *identifier* (misalnya NIM) ke nilai *boolean* (untuk memastikan keunikan).
    - Pemetaan alamat dompet mahasiswa ke nomor sertifikat yang pernah diajukan.
    - Pemetaan alamat dompet mahasiswa ke *boolean* (untuk memastikan apakah mahasiswa pernah membuat *QR Code*).
    - Pemetaan alamat dompet mahasiswa ke timestamp pembuatan *QR Code* terakhir.
e. *Array*:
    - Menyimpan semua pengajuan SKKM.
    - Menyimpan semua data pengguna.
    - Menyimpan semua data *QR Code*.
f. Variabel Lain:
    - Alamat dompet *super admin* (pemilik kontrak).
    - Jumlah poin minimal untuk menghasilkan *QR Code*.

Data-data ini dikelola oleh *smart contract* dengan cara mencatat, memperbarui, dan memverifikasi data tersebut sesuai dengan logika dan aturan yang telah ditentukan dalam kontrak pintar. Dengan demikian, *smart contract* memastikan bahwa data yang dikelola tetap aman, terverifikasi, dan dapat diakses oleh pihak yang berwenang sesuai dengan kebutuhan sistem.

2. Fungsi Utama *Smart Contract*:
    a. Pengelolaan Pengguna: Fungsi ini memungkinkan penambahan, pembaruan, dan penghapusan data pengguna, termasuk mahasiswa, admin, HMJ, dan BEM, sehingga memastikan data pengguna selalu *up-to-date* dan akurat dalam sistem.

b. Pengajuan SKKM: Fungsi ini memungkinkan mahasiswa untuk mengajukan, mengedit, dan menghapus pengajuan SKKM, serta memastikan bahwa semua pengajuan dicatat dengan benar dan dapat diperbarui sebelum diverifikasi.
c. Verifikasi dan Validasi SKKM: Fungsi ini memungkinkan HMJ untuk memverifikasi pengajuan SKKM dan memvalidasi pengajuan yang telah diverifikasi, sehingga memastikan setiap pengajuan telah melalui proses pengecekan yang ketat sebelum dianggap sah.
d. Pembuatan dan Validasi *QR Code*: Fungsi ini memungkinkan mahasiswa untuk membuat *QR Code* setelah memenuhi jumlah poin minimal dan mengirimkan *QR Code* tersebut untuk divalidasi oleh BEM, sehingga memastikan keabsahan *QR Code* yang dihasilkan.
e. Pengaturan Poin Minimal: Fungsi ini memungkinkan admin untuk mengatur jumlah poin minimal yang diperlukan mahasiswa untuk menghasilkan *QR Code*, sehingga memberikan fleksibilitas dalam menentukan standar yang harus dicapai mahasiswa.

3. Mekanisme Keamanan:
   a. Pembatasan Akses: Penggunaan *modifier onlySuperAdmin*, *onlyAdmin*, *onlyStudent*, *onlyHMJ*, dan *onlyBEM* untuk memastikan bahwa hanya pihak yang berwenang yang dapat melakukan tindakan tertentu dalam *smart contract*. Misalnya, hanya super admin yang dapat menambahkan pengguna baru, dan hanya HMJ yang dapat memverifikasi SKKM.
   b. Verifikasi Identitas: Setiap transaksi di dalam *smart contract* memerlukan alamat dompet unik pengguna yang diverifikasi sebelum melakukan tindakan apa pun. Ini memastikan bahwa hanya pengguna yang terdaftar dan terverifikasi yang dapat mengakses dan mengubah data.
   c. Konsistensi Data: Penggunaan struktur data dan *mapping* untuk menjaga konsistensi dan keunikan data, seperti memastikan bahwa

nomor sertifikat tidak dapat digunakan lebih dari sekali oleh mahasiswa yang sama.

d. Transaksi Atomik: Setiap fungsi dalam *smart contract* dirancang untuk bersifat atomik, yang berarti bahwa transaksi harus selesai sepenuhnya atau tidak sama sekali. Ini memastikan bahwa tidak ada data yang terjebak dalam keadaan setengah jadi.
e. Validasi dan Verifikasi: Proses verifikasi dan validasi berlapis untuk setiap pengajuan SKKM dan *QR Code* memastikan bahwa hanya data yang sah yang dapat diubah statusnya. Pengajuan harus diverifikasi oleh HMJ sebelum dapat divalidasi, dan *QR Code* harus melalui validasi BEM.
f. Penggunaan Kriptografi: Penyimpanan *hash IPFS* untuk sertifikat memastikan bahwa dokumen digital dapat diverifikasi keasliannya tanpa memerlukan dokumen fisik. Setiap dokumen yang diunggah ke *IPFS* akan menghasilkan *hash* unik, sehingga setiap perubahan pada dokumen akan menghasilkan *hash* yang berbeda, memastikan integritas data.
g. Log Peristiwa: Penggunaan *event logging* untuk mencatat setiap tindakan penting, seperti penambahan pengguna, pengajuan SKKM, verifikasi, validasi, dan pembuatan *QR Code*. Ini memberikan jejak audit yang lengkap untuk setiap transaksi yang terjadi dalam sistem.
h. Pembatasan Fungsi Tertentu: Fungsi-fungsi tertentu hanya dapat dijalankan dalam kondisi tertentu. Misalnya, pengajuan SKKM hanya dapat diedit jika statusnya belum diverifikasi, dan *QR Code* hanya dapat divalidasi jika sudah dikirim ke BEM.
i. Proteksi Terhadap Serangan *Reentrancy*: *Smart contract* dirancang untuk menghindari serangan *reentrancy* dengan memodifikasi status dan kondisi sebelum melakukan panggilan eksternal atau transfer dana.
j. Update Kontrak: *Smart contract* menyediakan mekanisme untuk memperbarui informasi pengguna dan pengaturan poin minimal,

memungkinkan sistem untuk beradaptasi dengan perubahan kebutuhan tanpa mengorbankan keamanan.

Dengan mekanisme keamanan ini, *smart contract* SKKM memastikan bahwa data akademik yang dikelola tetap aman, terverifikasi, dan hanya dapat diakses oleh pihak yang berwenang sesuai dengan aturan yang telah ditetapkan.

3.4.3 Alur Proses

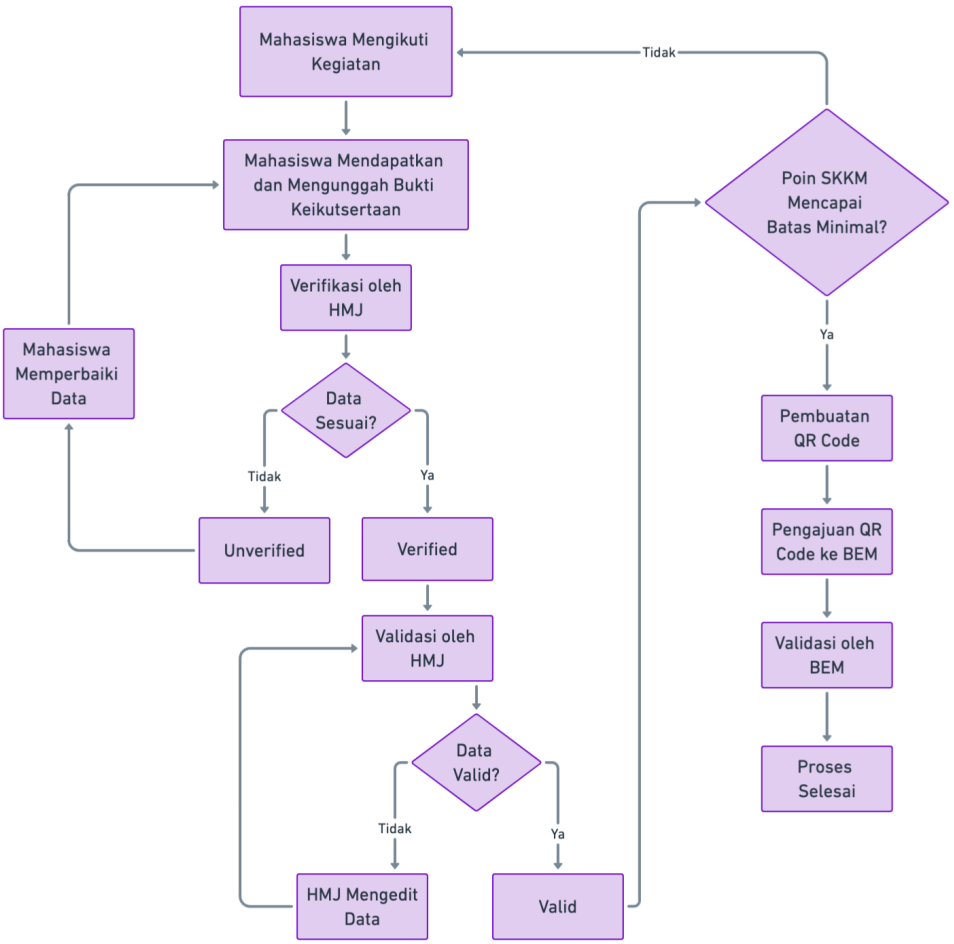
Alur proses pencatatan SKKM dalam sistem yang diusulkan ini dirancang untuk meningkatkan efisiensi, transparansi, dan akuntabilitas, serta mengakomodasi peran mahasiswa, HMJ, dan BEM dalam proses tersebut. Berikut adalah langkah-langkah detail dalam alur proses ini:

1. Mahasiswa Mengikuti Kegiatan: Mahasiswa berpartisipasi dalam kegiatan SKKM, baik di dalam maupun di luar kampus.
2. Mahasiswa Mendapatkan dan Mengunggah Bukti Keikutsertaan: Setelah menyelesaikan kegiatan, mahasiswa mendapatkan bukti keikutsertaan (sertifikat, surat keterangan, dll.) dan mengunggahnya ke aplikasi dalam format digital (*scan* atau foto).
3. Verifikasi oleh HMJ:
    a. HMJ menerima notifikasi tentang pengajuan SKKM baru.
    b. HMJ mengakses aplikasi untuk melihat detail pengajuan dan bukti keikutsertaan.
    c. HMJ melakukan verifikasi data, memastikan kesesuaian antara data yang diinputkan dengan bukti keikutsertaan.
    d. Jika data tidak sesuai, HMJ dapat menolak (Unverified) pengajuan dengan memberikan pesan perbaikan kepada mahasiswa.
    e. Mahasiswa dapat memperbaiki data dan mengajukan ulang (Revised).
    f. Jika data sesuai, HMJ menyetujui (Verified) pengajuan.
4. Validasi oleh HMJ:
    a. Setelah SKKM terverifikasi, HMJ melakukan validasi lebih lanjut.
    b. HMJ memeriksa kategori kegiatan, jenis kegiatan, capaian, dasar penilaian, dan poin kredit yang diajukan mahasiswa.

c. Jika ada ketidaksesuaian, HMJ dapat mengedit data untuk menyesuaikan poin kredit yang sesuai.
d. HMJ mengkonfirmasi validasi (Valid) SKKM setelah semua data dipastikan benar.

5. Pembuatan *QR Code* (Jika Memenuhi Syarat):
    a. Mahasiswa yang poin SKKM-nya telah mencapai atau melebihi batas minimal yang ditentukan dapat membuat *QR Code* melalui aplikasi.
    b. *QR Code* ini mewakili seluruh sertifikat yang valid pada saat itu. Jika ada sertifikat baru yang valid, mahasiswa perlu membuat *QR Code* baru.
6. Pengajuan *QR Code* ke BEM:
    a. Mahasiswa menyalin (copy) *QR Code* yang telah dibuat.
    b. Mahasiswa membuka menu validasi di aplikasi dan menempelkan (paste) *QR Code*.
    c. Mahasiswa mengirimkan (Send to BEM) *QR Code* untuk validasi oleh BEM.
7. Validasi oleh BEM:
    a. BEM menerima notifikasi tentang pengajuan *QR Code*.
    b. BEM mengakses aplikasi untuk melihat detail pengajuan dan data mahasiswa.
    c. BEM melakukan validasi *QR Code* dan data SKKM terkait.

Dengan alur ini, proses pencatatan SKKM menjadi lebih terstruktur, transparan, dan akuntabel. *Smart contract* mencatat setiap tahap proses, memastikan integritas data, dan memudahkan pelacakan riwayat SKKM mahasiswa.



Gambar 3.3 Flowchart: Alur Proses Sistem

3.4.4 Output Sistem

Output sistem pencatatan SKKM berbasis *blockchain* ini terdiri dari beberapa informasi dan fitur yang dihasilkan dari setiap tahapan alur proses:

1. Data SKKM Tercatat di *Blockchain*: Setiap pengajuan SKKM yang berhasil divalidasi oleh BEM akan dicatat secara permanen di *smart contract* yang berjalan di atas *blockchain Hyperledger Besu*. Data lengkap seperti informasi kegiatan, bukti keikutsertaan (dalam bentuk hash IPFS), poin SKKM yang diperoleh, serta riwayat verifikasi dan validasi disimpan di *smart contract*. *Blockchain* mencatat *hash* dari data ini untuk memastikan integritas dan keamanan.
2. Notifikasi Status Pengajuan: Mahasiswa, HMJ, dan BEM akan menerima notifikasi otomatis melalui aplikasi tentang perubahan status pengajuan

SKKM, seperti "*Pending*", "*Verified*", "*Unverified*", "*Valid*", atau "*Invalid*". Notifikasi ini membantu pengguna untuk memantau perkembangan pengajuan SKKM mereka dan mengambil tindakan yang diperlukan.

3. *QR Code*: Mahasiswa yang telah memenuhi syarat poin SKKM minimum dapat menghasilkan *QR Code* melalui aplikasi. *QR Code* ini berisi data *QR Code* yang terkait dengan data SKKM yang tersimpan di *smart contract*. Ketika *QR Code* ini divalidasi melalui aplikasi, semua data SKKM valid yang terkait dengan mahasiswa tersebut hingga saat *QR Code* dihasilkan akan ditampilkan. Hal ini memastikan bahwa pencapaian SKKM mahasiswa tidak dapat dipalsukan.
4. *Dashboard Monitoring*: Aplikasi menyediakan *dashboard* bagi HMJ dan BEM untuk memantau dan melihat jumlah status pengajuan SKKM. *Dashboard* ini membantu HMJ dan BEM dalam melakukan tugas mereka secara efisien dan transparan.
5. Riwayat Transaksi: Sistem mencatat semua transaksi yang terjadi di *blockchain*, termasuk pengajuan SKKM, verifikasi, validasi, dan pembuatan *QR Code*. Riwayat transaksi ini dapat digunakan untuk audit dan memastikan akuntabilitas dalam pencatatan SKKM.

Output sistem ini dirancang untuk memberikan informasi yang lengkap dan transparan tentang proses pencatatan SKKM, memudahkan pemantauan dan evaluasi, serta meningkatkan kepercayaan semua pihak yang terlibat dalam proses tersebut.

3.4.5 Desain Antarmuka

Desain antarmuka pengguna (UI) aplikasi pencatatan SKKM berbasis *blockchain* ini mengutamakan kemudahan penggunaan, aksesibilitas, dan pengalaman pengguna yang positif. Berikut adalah beberapa elemen kunci dalam desain *UI* aplikasi ini:

1. Halaman *Dashboard*:
    a. Mahasiswa: Menampilkan ringkasan data SKKM dan statusnya dan juga dapat melihat *Event*.

   b. HMJ: Menampilkan jumlah daftar pengajuan SKKM yang perlu diverifikasi dan validasi, sudah diverifikasi dan divalidasi.
   c. BEM: Menampilkan jumlah daftar pengajuan SKKM yang perlu divalidasi dan sudah di validasi.
2. Halaman Pengajuan SKKM:
   a. Formulir Pengajuan: Menyediakan formulir yang mudah diisi untuk mahasiswa mengunggah bukti keikutsertaan mereka. Formulir ini mencakup kolom untuk memilih kegiatan yang diikuti, mengunggah file bukti (sertifikat, surat keterangan, dll.).
   b. Pratinjau Bukti: Menampilkan pratinjau bukti keikutsertaan yang diunggah sebelum dikirimkan untuk verifikasi.
3. Halaman Verifikasi dan Validasi SKKM:
   a. Tampilan Detail Pengajuan: Menyediakan tampilan detail pengajuan SKKM, termasuk informasi kegiatan, bukti keikutsertaan, dan status verifikasi/validasi.
   b. Opsi Verifikasi/Validasi: Memberikan opsi bagi HMJ dan BEM untuk menyetujui atau menolak pengajuan SKKM, dengan kolom untuk memberikan alasan atau umpan balik jika diperlukan.
4. Halaman Pembuatan dan Validasi *QR Code*:
   a. Tombol Generate *QR Code*: Tombol yang dapat diklik oleh mahasiswa untuk membuat *QR Code* setelah memenuhi syarat poin SKKM minimum.
   b. Tampilan *QR Code*: Menampilkan *QR Code* yang dihasilkan, yang dapat diunduh atau disalin oleh mahasiswa.
   c. Formulir Validasi *QR Code*: Menyediakan formulir untuk Mahasiswa untuk memasukkan atau memindai *QR Code* untuk mengirimkan ke BEM untuk di validasi ataupun memverifikasi keaslian data SKKM yang tersimpan di *blockchain*.
5. Elemen *UI* Tambahan:
   a. Notifikasi: Memberikan notifikasi kepada pengguna tentang perubahan status pengajuan SKKM, validasi *QR Code*, atau informasi penting lainnya.

b. Histori Aktivitas: Menampilkan riwayat aktivitas pengguna, seperti pengajuan SKKM, verifikasi, dan validasi.
c. Bantuan dan Panduan: Menyediakan informasi bantuan dan panduan pengguna untuk membantu pengguna memahami cara menggunakan aplikasi.

Dengan desain *UI*/*UX* yang berpusat pada pengguna, aplikasi pencatatan SKKM berbasis *blockchain* ini diharapkan dapat meningkatkan partisipasi mahasiswa, mempermudah proses administrasi, dan meningkatkan transparansi dalam pengelolaan SKKM.

3.4.6 Teknologi yang Digunakan

Pengembangan aplikasi pencatatan SKKM ini memanfaatkan kombinasi teknologi *blockchain*, *smart contract*, *framework frontend* modern, dll. untuk menciptakan solusi yang aman, transparan, dan mudah digunakan. Berikut adalah rincian teknologi yang digunakan:

1. *Blockchain*:
    a. *Hyperledger Besu*: Klien *Ethereum enterprise* yang digunakan untuk membangun dan mengelola *private network blockchain*. *Hyperledger Besu* dipilih karena kemampuannya dalam menyediakan fitur-fitur *enterprise* seperti *permissioning* dan *private transactions*, serta dukungan untuk berbagai mekanisme konsensus yang cocok untuk *private network*, termasuk *QBFT*.
2. *Smart Contract*:
    a. *Solidity*: Bahasa pemrograman yang digunakan untuk menulis *smart contract* yang mengatur logika bisnis aplikasi pencatatan SKKM. *Solidity* adalah bahasa yang populer dan matang untuk pengembangan *smart contract* di *Ethereum* dan platform yang kompatibel dengan *EVM*.
3. *Frontend*:
    a. *ReactJS*: Pustaka *JavaScript* yang digunakan untuk membangun antarmuka pengguna (UI) aplikasi web. *ReactJS* dipilih karena arsitektur berbasis komponennya yang memudahkan pengembangan *UI* yang modular, dapat digunakan kembali, dan

mudah dipelihara. Selain itu, *ReactJS* juga memiliki komunitas yang besar dan dukungan yang luas, sehingga memudahkan dalam mencari solusi dan bantuan jika diperlukan.

4. *Web3*:
    a. *Thirdweb SDK*: Perangkat pengembangan perangkat lunak (SDK) yang menyederhanakan interaksi antara aplikasi web dengan *smart contract* dan *blockchain*. *Thirdweb SDK* menyediakan fungsi-fungsi yang mudah digunakan untuk membaca dan menulis data ke *smart contract*, mengirim transaksi, dan menangani *event*, sehingga mempercepat proses pengembangan aplikasi.
    b. *Web3.js* atau *ethers.js* (Opsional): Jika diperlukan fungsionalitas yang lebih spesifik atau kustomisasi yang lebih lanjut, *library Web3.js* atau *ethers.js* dapat digunakan sebagai alternatif atau pelengkap untuk *Thirdweb SDK*.
5. Penyimpanan Bukti Keikutsertaan:
    a. *IPFS* (InterPlanetary File System): Protokol penyimpanan terdesentralisasi yang digunakan untuk menyimpan bukti keikutsertaan mahasiswa. *IPFS* memastikan integritas data dengan menggunakan *content addressing* dan meningkatkan ketersediaan data dengan mendistribusikannya ke berbagai *node* di jaringan.
6. Basis Data (Opsional):
    a. *MySQL*: Basis data relasional yang digunakan untuk menyimpan data kegiatan yang tidak perlu disimpan di *blockchain*. *MySQL* dipilih karena kehandalannya, skalabilitas nya, kemudahan penggunaannya, dan dukungan komunitas yang luas.

Dengan teknologi-teknologi ini, sistem pencatatan SKKM berbasis *blockchain* ini diharapkan dapat memberikan solusi yang inovatif dan efektif untuk meningkatkan transparansi, efisiensi, dan keamanan dalam pengelolaan SKKM di Politeknik Negeri Malang.

3.5 Pengujian dan Evaluasi Sistem

Pengujian dan evaluasi merupakan tahapan krusial dalam penelitian ini. Tujuan utamanya adalah untuk memvalidasi hipotesis dan menjawab pertanyaan

penelitian tentang bagaimana cara mengatasi potensi manipulasi, mengurangi ketergantungan pada pihak ketiga, dan meningkatkan keamanan, integritas, serta transparansi data dalam pencatatan SKKM berbasis *blockchain*. Tahapan ini bertujuan untuk memastikan bahwa sistem yang dikembangkan berfungsi sesuai dengan yang diharapkan, memenuhi kebutuhan pengguna, dan memberikan manfaat yang signifikan bagi Politeknik Negeri Malang.

3.5.1 Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk memverifikasi apakah setiap fitur dan fungsi dalam aplikasi bekerja sesuai dengan spesifikasi yang telah ditentukan. Pengujian ini mencakup:

1. Pengujian Unit: Setiap komponen aplikasi, seperti *smart contract* untuk mengelola data SKKM, fungsi-fungsi dalam *Thirdweb SDK* yang digunakan untuk berinteraksi dengan *blockchain*, dan komponen *UI ReactJS* yang membentuk antarmuka pengguna, akan diuji secara individual. Tujuannya adalah untuk memastikan bahwa setiap komponen berfungsi sesuai harapan dan bebas dari kesalahan (bugs).
2. Pengujian Integrasi: Setelah pengujian unit selesai, komponen-komponen yang berbeda akan diintegrasikan dan diuji bersama-sama. Pengujian ini akan memastikan bahwa interaksi antara *frontend* (ReactJS) dan *backend* (smart contract dan Thirdweb SDK) berjalan lancar, serta memastikan bahwa aplikasi dapat berkomunikasi dengan *private network Hyperledger Besu* tanpa masalah.
3. Pengujian Sistem: Pada tahap ini, seluruh aplikasi akan diuji secara keseluruhan, mencakup alur kerja lengkap dari awal (pengajuan SKKM oleh mahasiswa) hingga akhir (validasi oleh pihak berwenang dan pembuatan laporan). Pengujian ini akan memastikan bahwa semua komponen aplikasi bekerja sama dengan baik dan pengguna (mahasiswa, HMJ, BEM) dapat menggunakan aplikasi ini untuk mencatat, memverifikasi, dan mengelola SKKM dengan mudah dan efektif.
4. Pengujian Skenario: Aplikasi akan diuji dalam berbagai skenario penggunaan yang realistis, seperti pengajuan SKKM untuk berbagai jenis kegiatan (seminar, workshop, kompetisi), proses verifikasi dan validasi

oleh HMJ dan BEM, serta pembuatan dan validasi *QR Code* untuk setiap SKKM yang tervalidasi. Pengujian ini akan membantu mengidentifikasi potensi masalah atau kesalahan yang mungkin muncul dalam situasi penggunaan yang sebenarnya.

3.5.2 Pengujian Non-Fungsional

Pengujian non-fungsional dilakukan untuk mengevaluasi kualitas aplikasi di luar fungsionalitas dasarnya. Pengujian ini mencakup:

1. Pengujian Keamanan Jaringan: Mengingat sensitivitas data akademik, pengujian keamanan menjadi sangat penting. Pengujian ini akan mencakup simulasi kondisi jaringan yang tidak ideal, seperti jumlah *node* yang tidak mencukupi untuk mencapai konsensus *QBFT*, atau *node* yang mengalami kegagalan sementara. Selain itu, pengujian akan dilakukan terhadap integritas data untuk memastikan bahwa data SKKM yang tercatat di *blockchain* tidak dapat diubah secara tidak sah, bahkan oleh pengguna dengan peran tinggi seperti *super admin*.
2. Pengujian Kegunaan (opsional): Pengujian ini melibatkan pengguna potensial (mahasiswa, HMJ, BEM) untuk mencoba aplikasi secara langsung. Umpan balik dari pengguna akan dikumpulkan mengenai kemudahan penggunaan, kejelasan navigasi, dan desain antarmuka secara keseluruhan. Tujuannya adalah untuk memastikan bahwa aplikasi mudah digunakan dan memenuhi kebutuhan pengguna.

3.5.3 Evaluasi Sistem

Evaluasi sistem dilakukan untuk mengukur tingkat keberhasilan aplikasi dalam mencapai tujuan penelitian. Evaluasi ini mencakup:

1. Evaluasi Efisiensi: Menganalisis seberapa efisien aplikasi dalam menyederhanakan proses pengajuan, verifikasi, dan validasi SKKM, serta menghasilkan *QR Code* dan laporan SKKM. Perbandingan akan dilakukan dengan metode pencatatan SKKM tradisional yang ada di POLINEMA, yang melibatkan lebih banyak langkah manual dan koordinasi antar pihak. Efisiensi akan diukur dari pengurangan jumlah langkah yang diperlukan, waktu tunggu yang lebih singkat, serta minimnya intervensi manual dalam proses.

2. Evaluasi Efektivitas: Mengukur seberapa efektif aplikasi dalam meningkatkan transparansi, akuntabilitas, dan kepercayaan dalam proses pencatatan SKKM.
3. Evaluasi Kepuasan Pengguna: Mengukur tingkat kepuasan pengguna terhadap aplikasi, termasuk kemudahan penggunaan, fitur yang disediakan, dan manfaat yang dirasakan.

Dengan melakukan pengujian dan evaluasi yang komprehensif, diharapkan aplikasi pencatatan SKKM berbasis *blockchain* ini dapat terbukti efektif, efisien, aman, dan memberikan manfaat yang signifikan bagi Politeknik Negeri Malang.

BAB IV. ANALISIS DAN PERANCANGAN SISTEM

4.1 Deskripsi Sistem

Sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis *blockchain* ini dirancang untuk mengembangkan sistem pencatatan yang lebih transparan, aman, dan efisien di Politeknik Negeri Malang (POLINEMA). Sistem yang ada saat ini menggunakan *database* tradisional yang rentan terhadap manipulasi data oleh otoritas pusat. Dalam sistem ini, admin atau pihak yang memiliki akses penuh ke *database* dapat mengubah, menghapus, atau menambahkan data secara langsung, sehingga menimbulkan risiko terhadap integritas dan kepercayaan data. Oleh karena itu, penelitian ini mengembangkan sebuah sistem baru yang memanfaatkan teknologi *blockchain* untuk mengatasi permasalahan tersebut.

Sistem ini memanfaatkan teknologi *blockchain*, sebuah buku besar digital terdistribusi yang mencatat transaksi secara aman dan transparan. Dalam konteks ini, setiap kegiatan SKKM yang diikuti oleh mahasiswa akan dicatat sebagai sebuah transaksi di *blockchain*. Data yang dicatat meliputi informasi detail kegiatan, bukti keikutsertaan, status verifikasi dan validasi, serta poin SKKM yang diperoleh mahasiswa.

Dengan menggunakan *blockchain*, sistem ini dapat menjamin integritas data SKKM, karena setiap transaksi yang tercatat di *blockchain* tidak dapat diubah atau dihapus. Selain itu, sistem ini juga meningkatkan transparansi karena semua pihak yang berwenang dapat melihat dan memverifikasi data SKKM yang tersimpan di *blockchain*.

4.1.1 Komponen Utama Sistem

Sistem pencatatan SKKM berbasis *blockchain* ini terdiri dari beberapa komponen utama yang saling terintegrasi untuk menciptakan sistem pencatatan yang lebih aman, transparan, dan efisien :

1. *Private Blockchain Network* (*Hyperledger Besu*):

   Jaringan *blockchain* pribadi ini berfungsi sebagai fondasi sistem, menyimpan data SKKM mahasiswa secara aman, transparan, dan tidak dapat diubah (immutable). *Hyperledger Besu* dipilih karena kemampuannya dalam menyediakan fitur-fitur *enterprise* seperti

pengaturan izin akses (permissioning) dan privasi data, yang sangat penting dalam menjaga kerahasiaan informasi mahasiswa. Setiap *node* dalam jaringan menyimpan salinan lengkap *blockchain* yang berisi data SKKM, dan mekanisme konsensus *QBFT* (Quorum-Based Byzantine Fault Tolerance) digunakan untuk mencapai kesepakatan tentang status *blockchain* dan memvalidasi transaksi baru secara efisien dan aman. Jaringan ini hanya dapat diakses oleh pihak-pihak yang berwenang di POLINEMA untuk menjaga keamanan dan kerahasiaan data.

2. *Smart Contract*:

   Smart contract adalah program komputer yang berjalan di atas *blockchain* dan berfungsi sebagai inti dari sistem ini. *Smart contract* mengatur alur kerja proses pencatatan SKKM. *Smart contract* ditulis dalam bahasa *Solidity* dan memastikan otomatisasi serta integritas data, sehingga setiap langkah dalam proses SKKM tercatat secara transparan dan tidak dapat diubah setelah dicatat di *blockchain*. Hal ini meningkatkan kepercayaan semua pihak terhadap sistem dan mengurangi potensi kecurangan atau manipulasi data.

3. Aplikasi Web (Frontend):

   Aplikasi berbasis web ini dibangun menggunakan *ReactJS* dan berfungsi sebagai antarmuka bagi pengguna untuk berinteraksi dengan sistem. Melalui aplikasi ini, mahasiswa dapat mendaftar kegiatan, mengunggah bukti partisipasi, melihat status pengajuan, dan membuat *QR Code* untuk verifikasi SKKM mereka. HMJ dan BEM dapat menggunakan aplikasi ini untuk memverifikasi dan memvalidasi pengajuan SKKM, memberikan umpan balik kepada mahasiswa, serta mengelola data kegiatan dan mahasiswa. *ReactJS* dipilih karena kemampuannya dalam membangun antarmuka pengguna yang modern, interaktif, dan responsif, sehingga meningkatkan pengalaman pengguna.

4. *IPFS* (InterPlanetary File System):

   IPFS digunakan sebagai sistem penyimpanan terdistribusi untuk menyimpan bukti partisipasi mahasiswa. *IPFS* memastikan bahwa bukti partisipasi tidak dapat diubah dan dapat diakses secara publik namun tetap

aman. Dalam sistem ini, bukti keikutsertaan mahasiswa diunggah ke *IPFS* dan *hash* dari *file* tersebut disimpan dalam *smart contract* di *blockchain*. Ini memastikan bahwa *file* tidak dapat diubah tanpa mengubah *hash*, sehingga menjaga integritas data. Menggunakan *hash* dari *file* di *IPFS* dalam *smart contract* memberikan keamanan yang tinggi dan efisiensi penyimpanan, karena *file* asli disimpan di jaringan *IPFS* yang terdistribusi, bukan di *blockchain* itu sendiri.

5. *Thirdweb SDK*:

   Thirdweb SDK menyediakan seperangkat alat dan fungsi yang memudahkan pengembangan aplikasi *web3* yang berinteraksi dengan *smart contract* di *blockchain*. *SDK* ini menyediakan abstraksi tingkat tinggi untuk operasi *blockchain* seperti membaca dan menulis data, mengirim transaksi, dan menangani *event*. Ini mempercepat proses pengembangan dengan menyederhanakan interaksi antara aplikasi *frontend* dan *smart contract* di *blockchain Hyperledger Besu*, memudahkan implementasi fitur-fitur seperti pengajuan, verifikasi, validasi SKKM, dan pembuatan *QR Code*.

Dengan mengintegrasikan komponen-komponen ini, sistem pencatatan SKKM berbasis *blockchain* ini diharapkan dapat memberikan solusi yang lebih aman, transparan, efisien, dan akuntabel dibandingkan dengan sistem yang ada saat ini.

4.2 Analisis Kebutuhan Fungsional

Aplikasi pencatatan SKKM berbasis *blockchain* ini akan digunakan oleh lima jenis pengguna utama, masing-masing dengan peran, kebutuhan, dan harapan yang berbeda terhadap sistem. Pemahaman mengenai kebutuhan dan harapan pengguna ini diperoleh melalui observasi dan analisis sistem yang ada, studi literatur, serta wawancara dengan narasumber yang relevan seperti Kemenkesma BEM dan mantan pengurus HMJ. Observasi terhadap sistem yang ada (website SKKM POLINEMA) menunjukkan bahwa proses verifikasi SKKM masih manual, melibatkan banyak pihak, dan memakan waktu lama. Studi literatur terkait sistem pencatatan akademik dan teknologi *blockchain* mengungkapkan bahwa transparansi dan keamanan data merupakan hal yang penting dalam sistem

pencatatan akademik. Wawancara dengan Kemenkesma BEM dan mantan pengurus HMJ memberikan informasi mendalam tentang pengalaman, masalah, dan harapan mereka terhadap sistem yang ada. Analisis kebutuhan fungsional dilakukan untuk mengidentifikasi fungsi-fungsi utama yang harus dimiliki oleh sistem pencatatan SKKM berbasis *blockchain*. Fungsi-fungsi ini akan menjadi dasar dalam perancangan *smart contract*, aplikasi web, dan komponen-komponen lain dalam sistem. Berikut adalah kebutuhan dari masing-masing pengguna:

4.2.1 Mahasiswa

Mahasiswa adalah pengguna utama sistem ini. Mereka akan menggunakan sistem untuk mengunggah bukti partisipasi, serta memantau status dan riwayat SKKM mereka.

- Kebutuhan:
    - Mahasiswa dapat mengajukan SKKM untuk kegiatan yang telah diikuti dengan mengunggah bukti partisipasi (dalam format digital) ke *IPFS* dan mengirimkan *hash IPFS* ke *smart contract*.
    - Kemudahan dalam tidak perlu mencetak sertifikat fisik dan menyerahkannya secara manual untuk verifikasi.
    - Transparansi dan efisiensi dalam proses verifikasi dan validasi SKKM secara digital.
    - Kemudahan dalam mengakses riwayat SKKM dan melihat total poin yang telah diperoleh.
    - Sistem yang mudah digunakan dan intuitif, memungkinkan mahasiswa mengunggah bukti partisipasi secara digital tanpa perlu mencetak sertifikat fisik.
    - Proses verifikasi dan validasi SKKM yang cepat dan transparan, sehingga mahasiswa dapat melihat status pengajuan mereka secara *real-time*.
    - Keamanan data SKKM yang terjamin, memastikan bahwa informasi pribadi dan akademik mahasiswa terlindungi.

4.2.2 HMJ (Himpunan Mahasiswa Jurusan)

HMJ bertanggung jawab untuk memverifikasi keaslian bukti partisipasi yang diajukan oleh mahasiswa, serta melakukan validasi awal terhadap SKKM.

- Kebutuhan:
    - Akses ke daftar pengajuan SKKM yang belum diverifikasi dan melakukan verifikasi terhadap setiap pengajuan.
    - Kemudahan dalam melihat bukti partisipasi yang diunggah oleh mahasiswa secara digital, sehingga tidak perlu menerima sertifikat fisik untuk verifikasi.
    - HMJ dapat memberikan status "Terverifikasi" atau "Tidak Terverifikasi" untuk setiap pengajuan SKKM, beserta alasan atau umpan balik jika memberikan status tidak terverifikasi diperlukan.
    - HMJ dapat melakukan validasi SKKM yang telah diverifikasi, mengatur kategori, tipe kegiatan, pencapaian, basis penilaian, dan poin SKKM.
    - Alur verifikasi dan validasi yang efisien untuk mengurangi waktu dan usaha yang diperlukan dalam proses manual.
    - Sistem yang efisien dan mudah digunakan untuk mengelola pengajuan SKKM, memungkinkan seluruh proses dilakukan secara digital.
    - Alur verifikasi yang jelas dan terstruktur, memastikan proses verifikasi dan validasi dapat berjalan lancar dan cepat.
    - Kemampuan untuk berkomunikasi dengan mahasiswa melalui sistem, memberikan umpan balik secara langsung dan mempercepat proses komunikasi.
    - Proses verifikasi dan validasi yang terotomatisasi untuk mengurangi potensi kesalahan manusia dan meningkatkan akurasi data.

4.2.3 BEM (Badan Eksekutif Mahasiswa)

BEM bertanggung jawab untuk melakukan validasi akhir terhadap *QR Code* yang telah dihasilkan oleh mahasiswa setelah divalidasi oleh HMJ.

- Kebutuhan:
    - Akses ke daftar *QR Code* yang telah dihasilkan oleh mahasiswa dan perlu divalidasi.
    - Kemampuan untuk memvalidasi *QR Code*.

- Kemampuan untuk melihat data SKKM melalui *QR Code*.
- Sistem yang memberikan informasi yang lengkap dan akurat tentang SKKM mahasiswa, termasuk data yang terkait dengan *QR Code*.
- Proses validasi *QR Code* yang mudah dan cepat, sehingga mahasiswa dapat segera menggunakan *QR Code* yang valid.
- Laporan yang komprehensif tentang SKKM mahasiswa, termasuk jumlah *QR Code* yang telah divalidasi dan status validasi masing-masing mahasiswa.

4.2.4 *Admin*

Admin bertanggung jawab untuk mengelola pengguna (mahasiswa, HMJ, BEM), kegiatan SKKM, dan keseluruhan sistem.

- Kebutuhan:
  - Menambahkan, mengedit, dan menghapus pengguna.
  - Mengelola data pengguna, termasuk memastikan bahwa setiap pengguna memiliki peran yang sesuai.
  - Melihat daftar pengguna.
  - Mengatur poin minimal SKKM yang harus dicapai mahasiswa.
  - Antarmuka admin yang mudah digunakan dan intuitif.
  - Sistem yang handal dan aman untuk mengelola data pengguna dan kegiatan SKKM.
  - Kemampuan untuk dengan mudah mengatur dan memperbarui poin minimal SKKM yang harus dicapai oleh mahasiswa.
  - Dapat membuat info poin tentang kategori-kategori kegiatan yang ada di SKKM. Ini dilakukan menggunakan *database* tradisional karena tidak berhubungan langsung dengan *blockchain* (opsional).
  - *Admin* dapat membuat daftar pilihan departemen yang ada di POLINEMA untuk keperluan manajemen data (opsional).

4.2.5 *Super Admin*

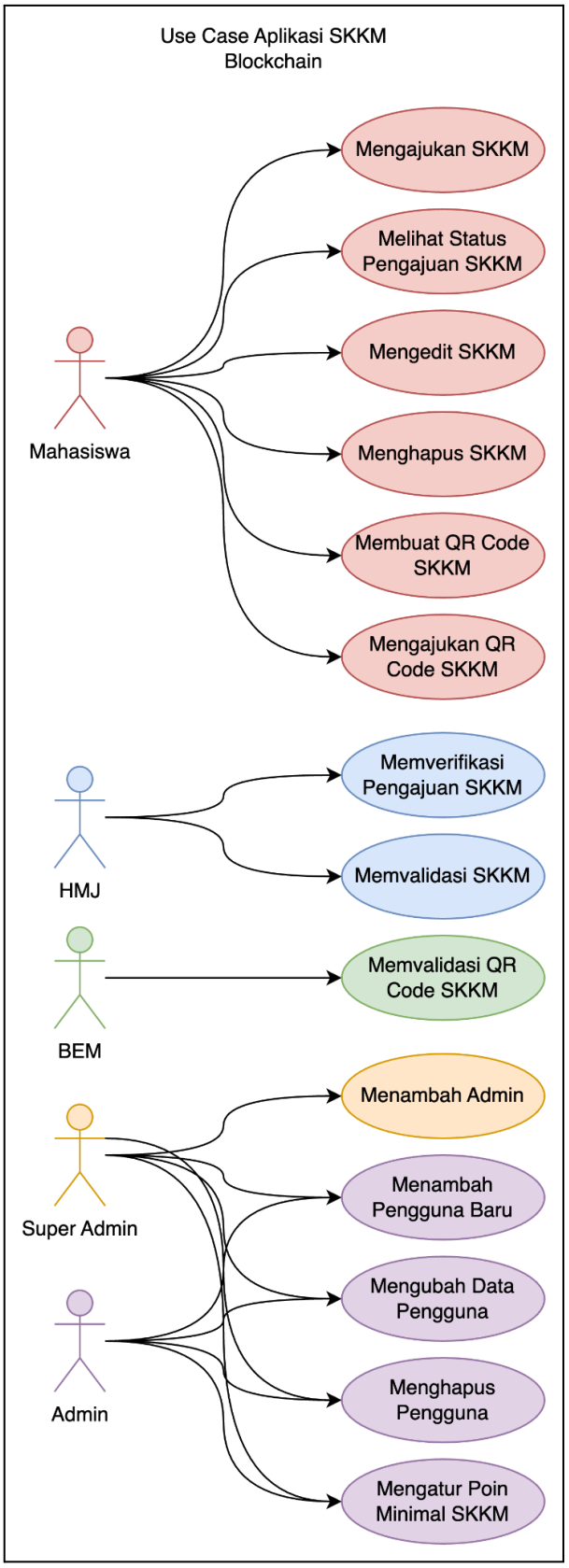
Super admin adalah inisiator awal yang melakukan *deployment smart contract* dan memiliki hak istimewa lain untuk fungsi-fungsi yang dapat diakses oleh admin biasa.

- Kebutuhan:
    - Melakukan *deployment smart contract* ke jaringan *blockchain*.
    - Menambahkan pengguna awal ke dalam sistem, termasuk admin, HMJ, dan BEM.
    - Proses *deployment* yang lancar dan bebas dari kesalahan, memastikan bahwa *smart contract* dapat beroperasi dengan baik dari awal.
    - *Smart contract* yang berfungsi dengan baik setelah *deployment*, tanpa *bug* atau kerentanan yang dapat mengganggu operasional sistem.
    - Memastikan bahwa *super admin*, meskipun melakukan *deployment smart contract*, tidak dapat mengubah data-data yang ada, menjaga integritas dan keamanan sistem.
    - Kemampuan untuk mengkonfigurasi awal sistem dan memastikan pengguna utama telah ditambahkan dengan benar.

Dengan mengidentifikasi kebutuhan fungsional ini, pengembangan sistem pencatatan SKKM berbasis *blockchain* dapat difokuskan pada implementasi fitur-fitur yang sesuai dengan kebutuhan pengguna dan tujuan dari sistem. Selain itu, analisis kebutuhan fungsional ini juga akan menjadi dasar dalam perancangan *use case* diagram dan skenario *use case* untuk menggambarkan interaksi antara pengguna dan sistem.

4.2.6 *Use Case* Diagram

Use case diagram menggambarkan interaksi antara aktor (pengguna) dengan sistem, serta fungsi-fungsi (use case) yang dapat dilakukan oleh masing-masing aktor. Berikut adalah *use case* diagram untuk sistem pencatatan SKKM berbasis *blockchain*:



Gambar 4.1 Use case Aplikasi SKKM *Blockchain*

4.2.7 Skenario *Use Case*

Skenario *use case* akan menjelaskan langkah-langkah detail dalam setiap *use case*. Berikut adalah skenario *use case* sesuai dengan *use case* 4.1 yang telah diidentifikasi:

1. Skenario *Use Case* : Mengajukan SKKM

Tabel 4.1 Skenario *Use Case* : Mengajukan SKKM

Nama *Use Case*	Mengajukan SKKM
Aktor	Mahasiswa
Deskripsi	Mahasiswa mengajukan SKKM untuk kegiatan yang telah diikuti, termasuk bukti partisipasi dalam format digital
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ Kategori kegiatan SKKM yang diikuti telah terdaftar di sistem. ➔ Mahasiswa memiliki bukti partisipasi dalam kegiatan SKKM dalam format digital.
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. ➔ Mahasiswa mengklik menu "*Submit* SKKM". ➔ Mahasiswa diarahkan ke halaman pengajuan SKKM. ➔ Mahasiswa mengisi formulir pengajuan SKKM dengan memilih kegiatan yang diikuti, nomor sertifikat, capaian, dasar penilaian, mengunggah bukti partisipasi (dalam format digital, dan melengkapi informasi lainnya (jika diperlukan). ➔ Mahasiswa mengklik tombol "*Submit*". ➔ Sistem menampilkan *hash*

	IPFS dari bukti partisipasi. ➔ Sistem mengirimkan file asli ke *IPFS*. Kemudian *hash* bukti partisipasi *IPFS* dan data pengajuan lainnya ke *smart contract*. ➔ *Smart contract* membuat objek *SKKMRequest* baru dengan status "*Pending*" dan menyimpannya dalam *array requests*. ➔ *Smart contract* memancarkan *event SKKMSubmitted* (untuk menampilkan notifikasi). ➔ Sistem menampilkan notifikasi bahwa pengajuan SKKM telah berhasil.
Alur Alternatif	➔ Nomor sertifikat sudah digunakan: Jika nomor sertifikat sudah digunakan sebelumnya oleh mahasiswa yang sama, *smart contract* akan menolak transaksi dan *frontend* akan menampilkan pesan kesalahan. ➔ Terjadi kesalahan dalam pengiriman data ke *smart contract*: Sistem akan menangani kesalahan pengiriman transaksi dan menampilkan pesan kesalahan.
Post Kondisi	➔ Pengajuan SKKM mahasiswa, termasuk *hash IPFS* bukti partisipasi, tercatat dalam *array requests* di *blockchain* dengan status "*Pending*".

2. Skenario *Use Case*: Melihat Status Pengajuan SKKM

Tabel 4.2 Skenario *Use Case*: Melihat Status Pengajuan SKKM

Nama *Use Case*	Melihat Status Pengajuan SKKM
Aktor	Mahasiswa
Deskripsi	Mahasiswa melihat status pengajuan SKKM yang telah diajukan, termasuk detail SKKM dan alasan penolakan jika ada.
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ Mahasiswa telah mengajukan setidaknya satu SKKM.
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. kemudian mahasiswa melihat daftar pengajuan SKKM yang telah diajukan. ➔ Mahasiswa mengklik menu "*MyPoints*" ➔ Mahasiswa diarahkan ke halaman menu *My Points*. kemudian mahasiswa dapat melihat lebih detail SKKM miliknya. ➔ Sistem mengirimkan alamat *Ethereum* mahasiswa (msg.sender) ke *smart contract* untuk mendapatkan daftar SKKM yang terkait dengan mahasiswa tersebut. ➔ *Smart contract* mengembalikan *array SKKMRequest* yang berisi detail semua SKKM yang diajukan oleh mahasiswa. ➔ Sistem menampilkan detail pengajuan SKKM, termasuk status verifikasi dan validasi,

	serta alasan jika pengajuan ditolak.
Alur Alternatif	➔ Tidak ada SKKM yang diajukan: Jika fungsi *getSKKMByStudent* tidak mengembalikan data, sistem akan menampilkan pesan "Tidak ada data yang tersedia".
Post Kondisi	➔ Mahasiswa mengetahui status pengajuan SKKM mereka.

3. Skenario *Use Case*: Mengedit SKKM

Tabel 4.3 Skenario *Use Case*: Mengedit SKKM

Nama *Use Case*	Mengedit SKKM
Aktor	Mahasiswa
Deskripsi	Mahasiswa mengedit data SKKM yang telah diajukan namun belum diverifikasi atau sudah diunverifikasi oleh HMJ, termasuk mengganti bukti partisipasi jika diperlukan.
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ SKKM yang diajukan memiliki status "*Pending*", "*Unverified*", atau "*Revised*". ➔ Mahasiswa memiliki bukti partisipasi dalam kegiatan SKKM dalam format digital (jika perlu mengganti).
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. ➔ Mahasiswa mengklik menu "*MyPoints*" ➔ Mahasiswa diarahkan ke halaman menu *My Points*.

	➔ Mahasiswa memilih SKKM yang ingin diedit dari daftar SKKM yang sudah diajukan. ➔ Mahasiswa mengklik *icon* "Pensil". ➔ Mahasiswa diarahkan ke halaman edit SKKM. ➔ Mahasiswa mengubah informasi yang diperlukan seperti kegiatan yang diikuti, nomor sertifikat, capaian, dasar penilaian, mengunggah bukti partisipasi baru (jika perlu), dan melengkapi informasi lainnya (jika diperlukan). ➔ Mahasiswa mengklik tombol "*Submit*". ➔ *Smart contract* memeriksa apakah *ID* SKKM valid dan milik mahasiswa yang melakukan permintaan. ➔ *Smart contract* memeriksa apakah status SKKM adalah "*Pending*" atau "*Unverified*". ➔ *Smart contract* memverifikasi bahwa nomor sertifikat baru (jika diubah) belum pernah digunakan oleh mahasiswa yang sama. ➔ Sistem menampilkan *hash IPFS* dari bukti partisipasi yang baru (jika diunggah). ➔ Sistem mengirimkan *file* asli ke *IPFS*. Kemudian *hash* bukti partisipasi *IPFS* dan data edit SKKM lainnya ke *smart contract*. ➔ *Smart contract* memperbarui data SKKM yang terkait dengan *ID* tersebut dalam *array requests*.

	➔ *Smart contract* mengubah status SKKM menjadi "*Pending*" jika status awalnya adalah "*Pending*", atau "*Revised*" jika status awalnya adalah "*Unverified*". ➔ *Smart contract* memancarkan *event SKKMEdited* (untuk notifikasi). ➔ Sistem menampilkan notifikasi bahwa edit SKKM telah berhasil.
Alur Alternatif	➔ Data tidak valid: Jika data yang diubah tidak valid (misalnya, nomor sertifikat sudah digunakan), *smart contract* akan menolak transaksi dan *frontend* akan menampilkan pesan kesalahan. ➔ Status SKKM tidak valid: Jika status SKKM bukan "*Pending*" atau "*Unverified*", *smart contract* akan menolak transaksi dan *frontend* akan menampilkan pesan kesalahan. ➔ *ID* SKKM tidak valid: Jika *ID* SKKM tidak valid atau tidak milik mahasiswa yang melakukan permintaan, *smart contract* akan menolak transaksi dan *frontend* akan menampilkan pesan kesalahan. ➔ Terjadi kesalahan dalam pengiriman data ke *smart contract*: *Frontend*/*backend* akan menangani kesalahan pengiriman transaksi dan menampilkan pesan kesalahan.
Post Kondisi	➔ SKKM yang telah diedit mahasiswa tercatat dalam

	array requests di *blockchain* dengan status "*Pending*" jika status awal adalah "*Pending*", atau "*Revised*" jika status awalnya adalah "*Unverified*".

4. Skenario *Use Case*: Menghapus SKKM

Tabel 4.4 Skenario *Use Case*: Menghapus SKKM

Nama *Use Case*	Menghapus SKKM
Aktor	Mahasiswa
Deskripsi	Menghapus SKKM yang sudah diajukan namun belum diverifikasi oleh HMJ.
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ SKKM yang diajukan memiliki status "*Pending*".
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. ➔ Mahasiswa mengklik menu "*MyPoints*" ➔ Mahasiswa diarahkan ke halaman menu *My Points*. ➔ Mahasiswa memilih SKKM yang ingin dihapus dari daftar SKKM yang sudah diajukan. ➔ Mahasiswa mengklik *icon* "Sampah". ➔ Sistem menampilkan *pop-up* konfirmasi penghapusan. ➔ Mahasiswa mengklik tombol "*Confirm*" pada *pop-up* konfirmasi. ➔ Sistem mengirimkan *ID* SKKM yang akan dihapus ke *smart contract* melalui transaksi *blockchain*.

	➔ *Smart contract* memeriksa apakah *ID* SKKM valid dan milik mahasiswa yang melakukan permintaan. ➔ *Smart contract* memeriksa apakah status SKKM adalah "*Pending*". ➔ *Smart contract* menghapus SKKM dari *array requests*. ➔ *Smart contract* memancarkan *event SKKMDeleted* (untuk notifikasi). ➔ Sistem menampilkan notifikasi bahwa penghapusan SKKM telah berhasil.
Alur Alternatif	➔ Gagal mengirim data ke *smart contract*: *Frontend*/*backend* menangani kesalahan pengiriman transaksi dan menampilkan pesan kesalahan. ➔ Status SKKM bukan "*Pending*": Jika status SKKM bukan "*Pending*", *smart contract* akan menolak transaksi dan *frontend*/*backend* akan menampilkan pesan kesalahan. ➔ *ID* SKKM tidak valid: Jika *ID* SKKM tidak valid atau tidak milik mahasiswa yang melakukan permintaan, *smart contract* akan menolak transaksi dan *frontend*/*backend* akan menampilkan pesan kesalahan.
Post Kondisi	➔ SKKM yang telah dihapus tidak lagi tercatat dalam *array requests* di *blockchain*.

5. Skenario *Use Case*: Membuat *QR Code* SKKM

Tabel 4.5 Skenario *Use Case*: Membuat *QR Code* SKKM

Nama *Use Case*	Membuat *QR Code* SKKM
Aktor	Mahasiswa
Deskripsi	Mahasiswa membuat *QR Code* SKKM setelah poin SKKM mencapai batas minimal yang ditentukan. *QR Code* ini akan berisi string unik yang dapat digunakan untuk mengakses data SKKM mahasiswa.
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ Poin SKKM mahasiswa telah mencapai atau melebihi batas minimal yang ditentukan dalam *smart contract* (minimalPoin). ➔ Mahasiswa belum memiliki *QR Code* SKKM yang belum divalidasi oleh BEM (*hasGeneratedQRCode* bernilai *false* atau *QR code* terakhir memiliki status selain "*Generated*" atau "*SendToBEM*").
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. ➔ Mahasiswa mengklik menu "*MyPoints*". ➔ Mahasiswa diarahkan ke menu *MyPoints*. ➔ Mahasiswa mengklik tombol " *QR Code*". ➔ Mahasiswa mengklik tombol "*Generate QR Code*". ➔ Sistem menghasilkan string unik untuk *QR Code*: String ini bisa berupa kombinasi angka,

	huruf, atau karakter lain yang unik. ➔ Mahasiswa mengklik tombol "Send *QR Code*". ➔ Sistem mengirimkan permintaan pembuatan *QR Code* ke *smart contract*: Frontend memanggil fungsi *generateQRCode* dengan data yang sesuai. ➔ *Smart contract* memeriksa ulang apakah poin SKKM mahasiswa mencukupi. ➔ *Smart contract* menyimpan string unik dalam *array qrcodes* dengan status "*Generated*" dan mencatat timestamp pembuatan. ➔ *Smart contract* memperbarui *hasGeneratedQRCode* menjadi *true*. ➔ *Smart contract* memancarkan *event* *QRCodeGenerated* (untuk notifikasi). ➔ Sistem membuat *QR code* yang berisi string unik tersebut dan menampilkannya kepada mahasiswa. ➔ Mahasiswa dapat menyalin atau mengunduh *QR Code*.
Alur Alternatif	➔ Poin SKKM belum mencukupi: Sistem menampilkan pesan kesalahan bahwa poin SKKM belum mencapai batas minimal. ➔ Mahasiswa sudah memiliki *QR Code* yang belum divalidasi: Sistem menampilkan pesan kesalahan bahwa mahasiswa sudah memiliki *QR Code* yang

	belum divalidasi dan harus menunggu validasi dari BEM sebelum membuat yang baru. ➔ Terjadi kesalahan dalam pengiriman data ke *smart contract* atau kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai.
Post Kondisi	➔ *QR Code* yang berisi string unik, beserta timestamp pembuatan, tercatat dalam *array qrcodes* di *blockchain* dengan status "*Generated*". ➔ *Event QRCodeGenerated* dipancarkan, yang dapat digunakan untuk mengirimkan notifikasi (opsional). ➔ Nilai *hasGeneratedQRCode* untuk mahasiswa diperbarui menjadi *true*.

6. Skenario *Use Case*: Mengajukan *QR Code* ke BEM

Tabel 4.6 Skenario *Use Case*: Mengajukan *QR Code* ke BEM

Nama *Use Case*	Mengajukan *QR Code* ke BEM
Aktor	Mahasiswa
Deskripsi	Mahasiswa mengajukan *QR Code* SKKM yang telah dibuat ke BEM untuk validasi, dengan tampilan *progress bar* yang menunjukkan status proses.
Pre Kondisi	➔ Mahasiswa telah *login* ke dalam sistem. ➔ Mahasiswa telah memiliki *QR Code* SKKM dengan status "*Generated*".

	➔ Mahasiswa sudah menyalin *QR Code* tersebut
Alur Normal	➔ Mahasiswa masuk ke halaman *dashboard*. ➔ Mahasiswa mengklik menu "*Validation*". ➔ Mahasiswa diarahkan ke halaman *Validation*. ➔ Mahasiswa memasukkan string unik ke *input QR Code* ke inputan. ➔ Mahasiswa mengklik tombol "Check *QR Code*". ➔ Sistem memanggil fungsi getSKKMByQRCode pada *smart contract*. ➔ *Smart contract* mengembalikan hasil: ◆ Jika *QR code* valid: *Smart contract* mengembalikan data SKKM yang terkait dengan *QR code* tersebut. Sistem menampilkan pesan "*QR Code* Found" pada *progress bar*. ◆ Jika *QR code* tidak valid: *Smart contract* tidak mengembalikan data. Sistem menampilkan pesan "Invalid *QR Code*" pada *progress bar*. ➔ Sistem memeriksa kepemilikan *QR Code*: ◆ Jika *QR code* valid dan milik mahasiswa yang *login* dan mahasiswa belum mengirim ke

	BEM: Sistem menampilkan tombol "*Send To BEM*" dan *progress bar* berubah menjadi "*QR Code* Found". ◆ Jika *QR code* valid tetapi bukan milik mahasiswa yang *login*: Sistem akan menyembunyikan tombol "*Send To BEM*". ➔ Mahasiswa mengklik tombol "*Send To BEM*" ➔ Sistem mengirimkan data *QR Code* ke *smart contract*. ➔ *Smart contract submitQRCode* dipanggil. ➔ Sistem menampilkan notifikasi bahwa pengajuan *QR code* telah berhasil dan *progress bar* berubah menjadi "*Send To BEM*".
Alur Alternatif	➔ Jika mahasiswa menempel *QR Code* SKKM yang tidak valid: Sistem akan menampilkan pesan kesalahan dan tidak mengizinkan pengajuan.
Post Kondisi	➔ *QR Code* SKKM mahasiswa tercatat di *blockchain* dengan status "*SendToBEM*". ➔ *Event QRCodeSubmitted* dipancarkan, yang dapat digunakan untuk mengirimkan notifikasi ke BEM. ➔ Progress bar menampilkan status "*Validated by BEM*" setelah *QR code* divalidasi oleh BEM.

7. Skenario *Use Case*: Memverifikasi Pengajuan SKKM

Tabel 4.7 Skenario *Use Case*: Memverifikasi Pengajuan SKKM

Nama *Use Case*	Memverifikasi Pengajuan SKKM
Aktor	HMJ
Deskripsi	HMJ memverifikasi keaslian bukti partisipasi dan data SKKM yang diajukan oleh mahasiswa.
Pre Kondisi	➔ HMJ telah *login* ke dalam sistem. ➔ Terdapat pengajuan SKKM yang belum diverifikasi (status "*Pending*" atau "*Revised*").
Alur Normal	➔ HMJ masuk ke halaman *dashboard* HMJ. ➔ HMJ mengklik menu "*Verify* SKKM". ➔ HMJ diarahkan ke halaman *Verify* SKKM. ➔ HMJ melihat daftar pengajuan SKKM yang belum diverifikasi. ➔ HMJ memilih pengajuan SKKM yang akan diverifikasi. ➔ HMJ melihat informasi kegiatan, data mahasiswa, dan bukti partisipasi yang diunggah. ➔ HMJ melakukan verifikasi keaslian bukti partisipasi dan kesesuaian data. ➔ HMJ mengklik tombol "Verifikasi" atau "Tolak" sesuai hasil verifikasi. ➔ Jika SKKM ditolak, HMJ mengisi alasan penolakan pada kolom yang disediakan dan mengklik tombol "Kirim".

	➔ Sistem mengirimkan status verifikasi ke *smart contract* verifySKKM. ➔ *Smart contract* memeriksa apakah *ID* SKKM valid dan statusnya "*Pending*" atau "*Revised*". ➔ *Smart contract* memperbarui status SKKM menjadi "Verified" atau "*Unverified*" sesuai input. ➔ Jika SKKM ditolak, *smart contract* menyimpan alasan penolakan dalam field unverifiedMessage. ➔ *Smart contract* memancarkan *event* SKKMVerified. ➔ Sistem menampilkan notifikasi bahwa verifikasi SKKM telah berhasil.
Alur Alternatif	➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai. ➔ Jika status SKKM bukan "*Pending*" atau "*Revised*", *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan. ➔ Jika *ID* SKKM tidak valid, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan.
Post Kondisi	➔ Status pengajuan SKKM dalam *array requests* di *blockchain* diperbarui menjadi "*Verified*" atau "*Unverified*" sesuai hasil verifikasi. Jika

	ditolak, alasan penolakan dicatat dalam field unverifiedMessage. ➔ *Event* SKKMVerified dipancarkan, yang dapat digunakan untuk mengirimkan notifikasi kepada mahasiswa tentang status verifikasi SKKM mereka, termasuk alasan penolakan jika ada.

8. Skenario *Use Case*: Memvalidasi SKKM

Tabel 4.8 Skenario *Use Case*: Memvalidasi SKKM

Nama *Use Case*	Memvalidasi SKKM
Aktor	HMJ
Deskripsi	HMJ memvalidasi SKKM yang telah diverifikasi sebelumnya dan dapat menyesuaikan data poin SKKM jika diperlukan.
Pre Kondisi	➔ HMJ telah *login* ke dalam sistem. ➔ Terdapat pengajuan SKKM dengan status "Verified" (terverifikasi, namun belum divalidasi).
Alur Normal	➔ HMJ masuk ke halaman *dashboard* HMJ. ➔ HMJ mengklik menu "*Validate* SKKM". ➔ HMJ diarahkan ke halaman *Validate* SKKM. ➔ HMJ melihat daftar SKKM yang telah terverifikasi. ➔ HMJ memilih SKKM yang akan divalidasi. ➔ HMJ memeriksa kembali data SKKM dan memastikan

	kesesuaian dengan bukti partisipasi. ➔ HMJ dapat melakukan penyesuaian terhadap kategori kegiatan, jenis kegiatan, capaian, dasar penilaian, dan poin SKKM yang diberikan jika diperlukan. ➔ HMJ mengklik tombol "Validasi". ➔ Sistem mengirimkan data validasi ke *smart contract* validateSKKM. ➔ *Smart contract* memeriksa apakah *ID* SKKM valid dan statusnya "Verified". ➔ *Smart contract* memperbarui data SKKM sesuai input. ➔ *Smart contract* memperbarui status SKKM menjadi "Valid" dan mencatat timestamp validasi. ➔ *Smart contract* memancarkan *event* SKKMValidated. ➔ Sistem menampilkan notifikasi bahwa validasi SKKM telah berhasil.
Alur Alternatif	➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai. ➔ Jika status SKKM bukan "Verified", *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan. ➔ Jika *ID* SKKM tidak valid, *smart contract* akan menolak

	transaksi dan sistem akan menampilkan pesan kesalahan.
Post Kondisi	➔ Data SKKM, termasuk status ("Valid") dan timestamp validasi, diperbarui dalam *array requests* di *blockchain*. ➔ Poin SKKM mahasiswa, jika ada perubahan, juga diperbaharui dalam *array requests* di *blockchain*. ➔ *Event* SKKMValidated dipancarkan, yang dapat digunakan untuk mengirimkan notifikasi.

9. Skenario *Use Case*: Memvalidasi *QR Code* SKKM

Tabel 4.9 Skenario *Use Case*: Memvalidasi *QR Code* SKKM

Nama *Use Case*	Memvalidasi *QR Code* SKKM
Aktor	BEM
Deskripsi	BEM memvalidasi *QR Code* SKKM yang diajukan oleh mahasiswa
Pre Kondisi	➔ BEM telah *login* ke dalam sistem. ➔ Terdapat pengajuan *QR Code* SKKM dari mahasiswa yang perlu divalidasi (status *QR code* adalah "*SendToBEM*").
Alur Normal	➔ BEM masuk ke halaman *dashboard* BEM. ➔ BEM mengklik menu "*Validate QR Code*". ➔ BEM diarahkan ke halaman *Validate QR Code*. ➔ BEM memilih *QR Code* SKKM yang akan divalidasi. ➔ BEM mengklik tombol "*Validate*".

	➔ Sistem Sistem mengirimkan *ID QR code* yang akan divalidasi ke *smart contract* melalui transaksi *blockchain*. ➔ *Smart contract* memeriksa apakah *ID QR code* valid dan statusnya "*SendToBEM*". ➔ *Smart contract* memperbarui status *QR code* menjadi "*ValidatedByBEM*". ➔ *Smart contract* memancarkan *event* *QRCodeValidatedByBEM*. ➔ Sistem menampilkan notifikasi bahwa *QR code* SKKM telah berhasil divalidasi.
Alur Alternatif	➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai. ➔ Jika *QR code* tidak valid atau sudah divalidasi, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan.
Post Kondisi	➔ Status *QR Code* SKKM diperbaharui menjadi "*ValidatedByBEM*" dalam *array qrcodes* di *blockchain*. ➔ *Event* *QRCodeValidatedByBEM* dipancarkan, yang dapat digunakan untuk mengirimkan notifikasi kepada mahasiswa.

10. Skenario *Use Case*: Menambah Pengguna Baru

Tabel 4.10 Skenario *Use Case*: Menambah Pengguna Baru

Nama *Use Case*	Menambah Pengguna Baru
Aktor	*Admin* atau *Super Admin*
Deskripsi	*Admin* atau *Super Admin* menambahkan pengguna baru (Mahasiswa, HMJ, BEM, atau *Admin*) ke dalam sistem dengan informasi lengkap dan peran yang sesuai.
Pre Kondisi	➔ *Admin* atau *Super Admin* telah *login* ke dalam sistem.
Alur Normal	➔ *Admin* atau *Super Admin* masuk ke halaman *dashboard Admin*. ➔ *Admin* atau *Super Admin* mengklik menu "*User*". ➔ *Admin* atau *Super Admin* diarahkan ke halaman *User*. ➔ *Admin* atau *Super Admin* mengklik tombol "Add *User*". ➔ *Admin* atau *Super Admin* diarahkan ke halaman formulir tambah pengguna. ➔ *Admin* atau *Super Admin* mengisi formulir dengan informasi lengkap mengenai pengguna baru, seperti nama, NIM/NIP, departemen, peran (Mahasiswa, HMJ, BEM, atau *Admin*) dan *ETH Amount*. ➔ *Admin* atau *Super Admin* mengklik tombol "Add *User*". ➔ Sistem mengirimkan data yang telah diisi ke *smart contract* melalui transaksi *blockchain*. ➔ *Smart contract* memeriksa apakah pengguna sudah ada (*users[user].exists*).

	➔ *Smart contract* memeriksa apakah NIM/NIP sudah terdaftar (*identifiers[identifier]*). ➔ *Smart contract* memeriksa apakah pengirim transaksi memiliki hak untuk menambahkan pengguna dengan peran tersebut. ➔ Jika semua kondisi terpenuhi, *smart contract* membuat objek *User* baru dan menyimpannya dalam *array users* dan *allUsers*. ➔ *Smart contract* menandai NIM/NIP sebagai sudah terdaftar. ➔ Jika ada *ETH* yang dikirim, *smart contract* mentransfer *ETH* tersebut ke alamat pengguna baru. ➔ *Smart contract* memancarkan *event UserAdded*. ➔ Sistem menampilkan notifikasi bahwa pengguna baru berhasil ditambahkan.
Alur Alternatif	➔ Jika data tidak lengkap, format data tidak sesuai, atau NIM/NIP sudah terdaftar, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan yang spesifik. ➔ Jika pengirim transaksi adalah *Admin* dan mencoba menambahkan pengguna dengan peran *Admin*, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan.

	➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai.
Post Kondisi	➔ Pengguna baru terdaftar dalam *array users* dan *allUsers* di *blockchain*: Data pengguna baru disimpan secara permanen di *blockchain*. ➔ NIM/NIP pengguna baru ditandai sebagai sudah terdaftar untuk mencegah duplikasi NIM/NIP. ➔ Jika ada, *ETH* ditransfer ke alamat pengguna baru sebagai pemberian insentif atau transfer awal *ETH* kepada pengguna baru. ➔ *Event UserAdded* dipancarkan untuk mengirimkan notifikasi atau melakukan tindakan lain setelah pengguna berhasil ditambahkan.

11. Skenario *Use Case*: Mengubah Data Pengguna

Tabel 4.11 Skenario *Use Case*: Mengubah Data Pengguna

Nama *Use Case*	Mengubah Data Pengguna
Aktor	*Admin* atau *Super Admin*
Deskripsi	Mengubah data pengguna yang sudah terdaftar di sistem
Pre Kondisi	➔ *Admin* atau *Super Admin* telah *login* ke dalam sistem. ➔ Pengguna yang akan diubah datanya sudah terdaftar di sistem (memiliki entri dalam

	array users di *smart contract*).
Alur Normal	➔ *Admin* atau *Super Admin* masuk ke halaman *dashboard Admin*. ➔ *Admin* atau *Super Admin* mengklik menu "*User*". ➔ *Admin* atau *Super Admin* diarahkan ke halaman *User*. ➔ *Admin* atau *Super Admin* melihat daftar pengguna yang terdaftar. ➔ *Admin* atau *Super Admin* memilih pengguna yang datanya akan diubah. ➔ *Admin* atau *Super Admin* mengklik *icon* "Pensil". ➔ *Admin* atau *Super Admin* diarahkan ke halaman edit pengguna. ➔ *Admin* atau *Super Admin* mengubah informasi pengguna yang diperlukan, seperti nama, NIM/NIP, departemen, atau peran. ➔ *Admin* atau *Super Admin* mengklik tombol "Simpan Perubahan". ➔ Sistem mengirimkan data pengguna yang telah diubah ke *smart contract* updateUser melalui transaksi *blockchain*. ➔ *Smart contract* memeriksa apakah pengguna ada (*users*[user].exists). ➔ *Smart contract* memeriksa apakah NIM/NIP baru valid (unik dan belum digunakan oleh pengguna lain). ➔ *Smart contract* memeriksa apakah pengirim transaksi

	memiliki hak untuk mengubah peran pengguna (jika peran diubah). ➔ Jika semua kondisi terpenuhi, *smart contract* memperbarui data pengguna dalam *array users* dan *allUsers*. ➔ *Smart contract* memancarkan *event UserUpdated* (opsional, untuk notifikasi). ➔ Sistem menampilkan notifikasi bahwa data pengguna berhasil diubah.
Alur Alternatif	➔ Jika data tidak lengkap, format data tidak sesuai, atau NIM/NIP baru sudah digunakan oleh pengguna lain, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan yang spesifik. ➔ Jika pengirim transaksi adalah *Admin* dan mencoba mengubah peran pengguna lain menjadi *Admin*, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan. ➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai.
Post Kondisi	➔ Data pengguna berhasil diperbarui dalam *array users* dan *allUsers* di *blockchain*. ➔ *Event* UserUpdated (opsional) dipancarkan, yang dapat digunakan untuk mengirimkan

	notifikasi atau melakukan tindakan lain setelah data pengguna berhasil diubah.

12. Skenario *Use Case*: Menghapus Pengguna

Tabel 4.12 Skenario *Use Case*: Menghapus Pengguna

Nama *Use Case*	Menghapus Pengguna
Aktor	*Admin* atau *Super Admin*
Deskripsi	Menghapus pengguna dari sistem
Pre Kondisi	➔ *Admin* atau *Super Admin* telah *login* ke dalam sistem. ➔ Pengguna yang akan dihapus sudah terdaftar di sistem (memiliki entri dalam *array users* di *smart contract*).
Alur Normal	➔ *Admin* atau *Super Admin* masuk ke halaman *dashboard Admin*. ➔ *Admin* atau *Super Admin* mengklik menu "*User*". ➔ *Admin* atau *Super Admin* diarahkan ke halaman *User*. ➔ *Admin* atau *Super Admin* melihat daftar pengguna yang terdaftar. ➔ *Admin* atau *Super Admin* memilih pengguna yang akan dihapus. ➔ Sistem menampilkan dialog konfirmasi penghapusan. ➔ *Admin* atau *Super Admin* mengklik tombol "Hapus". ➔ Sistem mengirimkan permintaan penghapusan ke *smart contract deleteUser*: Sistem mengirimkan alamat *Ethereum* pengguna yang akan

	dihapus ke *smart contract* melalui transaksi *blockchain*. ➔ *Smart contract* memeriksa apakah pengguna ada (*users[user].exists*). ➔ *Smart contract* menghapus data pengguna dari *array users* dan *allUsers*. ➔ *Smart contract* menandai NIM/NIP pengguna sebagai tidak terdaftar lagi (menghapus dari mapping *identifiers*). ➔ *Smart contract* memancarkan *event UserDeleted*. ➔ Sistem menampilkan notifikasi bahwa pengguna berhasil dihapus.
Alur Alternatif	➔ Jika pengguna yang akan dihapus tidak ditemukan dalam *array users*, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan. ➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai.
Post Kondisi	➔ Pengguna berhasil dihapus dari *array users* dan *allUsers* di *blockchain*. ➔ NIM/NIP pengguna tidak lagi terdaftar dalam mapping *identifiers*. ➔ *Event UserDeleted* dipancarkan, yang dapat digunakan untuk memperbarui tampilan daftar pengguna di sistem atau melakukan

	tindakan lain yang diperlukan.

13. Skenario *Use Case*: Mengatur Poin Minimal SKKM

Tabel 4.13 Skenario *Use Case*: Mengatur Poin Minimal SKKM

Nama *Use Case*	Mengatur Poin Minimal SKKM
Aktor	*Admin* atau *Super Admin*
Deskripsi	Mengatur jumlah poin minimal SKKM yang harus dicapai mahasiswa sebelum mereka dapat membuat *QR Code*
Pre Kondisi	➔ *Admin* atau *Super Admin* telah *login* ke dalam sistem.
Alur Normal	➔ *Admin* atau *Super Admin* masuk ke halaman *dashboard Admin*. ➔ *Admin* atau *Super Admin* mengklik menu “Points”. ➔ *Admin* atau *Super Admin* diarahkan ke halaman Points. ➔ *Admin* atau *Super Admin* mengubah nilai poin minimal SKKM pada kolom yang disediakan. ➔ *Admin* atau *Super Admin* mengklik tombol "Simpan Perubahan". ➔ Sistem mengirimkan nilai poin minimal SKKM yang baru ke *smart contract* setMinimalPoin melalui transaksi *blockchain*. ➔ *Smart contract* memeriksa apakah pengirim transaksi adalah *Admin*/*Super Admin*. ➔ *Smart contract* memperbarui nilai minimalPoin dengan nilai baru.

	➔ *Smart contract* memancarkan *event* MinimalPoinUpdated. ➔ Sistem menampilkan notifikasi bahwa poin minimal SKKM berhasil diubah.
Alur Alternatif	➔ Jika pengirim transaksi bukan *Admin* atau *Super Admin*, *smart contract* akan menolak transaksi dan sistem akan menampilkan pesan kesalahan. ➔ Gagal mengirim data ke *smart contract* atau terjadi kesalahan pada *smart contract*: Sistem menangani kesalahan dan menampilkan pesan kesalahan yang sesuai.
Post Kondisi	➔ Poin minimal SKKM berhasil diperbarui dalam variabel minimalPoin di *blockchain*. ➔ *Event* MinimalPoinUpdated dipancarkan, yang dapat digunakan untuk memperbarui tampilan poin minimal di sistem atau melakukan tindakan lain yang diperlukan.

4.3 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional dilakukan untuk mengidentifikasi karakteristik dan kualitas sistem yang diharapkan, di luar fungsi-fungsi utamanya. Kebutuhan non-fungsional ini penting untuk memastikan bahwa sistem pencatatan SKKM berbasis *blockchain* dapat berjalan dengan baik dan memberikan pengalaman pengguna yang memuaskan. Berikut adalah beberapa kebutuhan non-fungsional yang diidentifikasi:

4.3.1 Keamanan

Mengingat sensitivitas data akademik dan potensi risiko penyalahgunaan, keamanan menjadi fokus utama dalam perancangan sistem ini. Berikut adalah

beberapa aspek keamanan yang perlu diperhatikan dan bagaimana sistem ini mengatasinya

- *Confidentiality* (Kerahasiaan): Data SKKM bersifat sensitif dan harus dijaga kerahasiaannya. Sistem harus memastikan bahwa hanya pihak yang berwenang (Mahasiswa, HMJ, BEM, dan admin) yang dapat mengakses data SKKM sesuai dengan peran mereka. Mekanisme autentikasi yang kuat seperti penggunaan dompet digital (misalnya, *MetaMask*) akan diimplementasikan untuk melindungi data dari akses yang tidak sah.
- *Integrity* (Integritas): Data SKKM harus terlindungi dari perubahan yang tidak sah. Teknologi *blockchain* memastikan integritas data dengan menyimpannya dalam blok-blok yang terhubung secara kriptografis, sehingga setiap perubahan akan terdeteksi. *Smart contract*, yang berjalan di atas *blockchain*, juga berkontribusi pada integritas data karena kode dan logikanya tidak dapat diubah setelah disebarkan. Setiap interaksi dengan *smart contract*, termasuk perubahan data SKKM, akan dicatat sebagai transaksi baru di *blockchain*, menciptakan jejak audit yang transparan dan tidak dapat diubah.

4.3.2 Kegunaan (Usability)

Sistem harus mudah digunakan dan memberikan pengalaman yang positif bagi pengguna. Berikut adalah beberapa aspek kegunaan yang perlu diperhatikan:

- *Ease of Use* (Kemudahan Penggunaan): Antarmuka pengguna harus intuitif, mudah di navigasi, dan mudah dipahami oleh semua jenis pengguna, termasuk mereka yang kurang familiar dengan teknologi *blockchain*.
- *Learnability* (Kemudahan Pembelajaran): Sistem harus mudah dipelajari oleh pengguna baru. Dokumentasi pengguna yang jelas dan panduan penggunaan yang interaktif dapat membantu pengguna untuk memahami cara menggunakan sistem dengan cepat.
- *User Satisfaction* (Kepuasan Pengguna): Sistem harus memberikan pengalaman pengguna yang positif dan memuaskan. Umpan balik pengguna harus dikumpulkan secara berkala untuk mengidentifikasi area yang perlu ditingkatkan.

4.3.3 *Maintainability* (Kemampuan Pemeliharaan)

Sistem harus mudah dipelihara dan diperbarui untuk memastikan keberlanjutan dan adaptasi terhadap perubahan kebutuhan. Berikut adalah beberapa aspek kemampuan pemeliharaan yang perlu diperhatikan:

- Sistem harus mudah dipelihara dan diperbarui. Kode *smart contract* harus terstruktur dengan baik, didokumentasikan dengan jelas, dan mudah dipahami oleh pengembang lain.
- Dokumentasi teknis yang lengkap akan disediakan, termasuk diagram alur, penjelasan fungsi, dan contoh penggunaan, untuk memudahkan pemeliharaan dan pengembangan di masa depan.
- Prosedur pembaruan *smart contract* akan didokumentasikan dengan jelas, termasuk langkah-langkah pengujian dan *deployment*.

4.3.4 *Testability* (Kemampuan Pengujian)

Sistem harus mudah diuji untuk memastikan kualitas dan keandalannya. Berikut adalah beberapa aspek kemampuan pengujian yang perlu diperhatikan:

- Sistem harus mudah diuji untuk memastikan kualitas dan keandalannya. Rencana pengujian yang komprehensif akan dibuat, termasuk:
    - Pengujian unit untuk setiap fungsi dalam *smart contract*.
    - Pengujian integrasi untuk memastikan interaksi yang benar antara *smart contract* dan *frontend*/*backend*.
    - Pengujian sistem untuk menguji keseluruhan sistem dalam lingkungan yang mirip dengan produksi.
- Otomatisasi pengujian akan dipertimbangkan untuk meningkatkan efisiensi dan mengurangi risiko kesalahan manusia.

Dengan mengidentifikasi dan memenuhi kebutuhan non-fungsional ini, diharapkan sistem pencatatan SKKM berbasis *blockchain* dapat mencapai tingkat keamanan, kegunaan, kinerja, dan kemampuan pemeliharaan yang tinggi, sehingga memberikan manfaat yang optimal bagi semua pengguna dan menjadi solusi yang efektif dan efisien dalam pengelolaan SKKM di POLINEMA.

4.4 Perancangan Sistem

Perancangan sistem ini mencakup beberapa aspek penting, mulai dari perancangan *smart contract* yang menjadi inti logika sistem, perancangan basis

data untuk penyimpanan data di luar *blockchain*, perancangan antarmuka pengguna yang intuitif, hingga perancangan jaringan *blockchain* yang aman dan terukur.

4.4.1 Perancangan *Smart Contract*

Smart contract akan menjadi inti dari sistem pencatatan SKKM berbasis *blockchain* ini. *Smart contract* akan ditulis menggunakan bahasa *Solidity*, sebuah bahasa pemrograman yang dirancang khusus untuk mengembangkan *smart contract* pada platform *Ethereum*. *Smart contract* ini akan berjalan di atas jaringan *blockchain Hyperledger Besu* yang bersifat *private*, sehingga hanya pihak yang berwenang yang dapat mengakses dan berinteraksi dengannya.

Smart contract ini akan memiliki beberapa fungsi utama yang berkaitan dengan pengelolaan SKKM, seperti:

- `addUser`: Fungsi untuk menambahkan pengguna baru (Mahasiswa, HMJ, BEM, atau *Admin*) ke dalam sistem.
- `updateUser`: Fungsi untuk mengubah data pengguna yang sudah terdaftar.
- `deleteUser`: Fungsi untuk menghapus pengguna dari sistem.
- `submitSKKM`: Fungsi untuk mahasiswa mengajukan SKKM dengan mengirimkan *hash* bukti keikutsertaan yang telah diunggah ke *IPFS*.
- `editSKKM`: Fungsi untuk mahasiswa mengubah data SKKM yang telah diajukan.
- `deleteSKKM`: Fungsi untuk mahasiswa menghapus SKKM yang telah diajukan.
- `verifySKKM`: Fungsi untuk HMJ melakukan verifikasi SKKM, memberikan status "*Verified*" atau "*Unverified*", dan menambahkan pesan jika tidak terverifikasi.
- `validateSKKM`: Fungsi untuk HMJ melakukan validasi akhir SKKM, mengubah status menjadi "Valid", dan memperbarui data SKKM jika diperlukan.
- `generateQRCode`: Fungsi untuk menghasilkan *QR Code* yang berisi string unik setelah poin SKKM mahasiswa mencapai batas minimal.

- *submitQRCode*: Fungsi untuk mahasiswa mengajukan *QR Code* ke BEM untuk validasi.
- validateQRCodeByBEM: Fungsi untuk BEM melakukan validasi *QR Code* yang diajukan oleh mahasiswa.

Selain fungsi-fungsi utama tersebut, *smart contract* juga akan memiliki fungsi-fungsi pendukung lainnya, seperti:

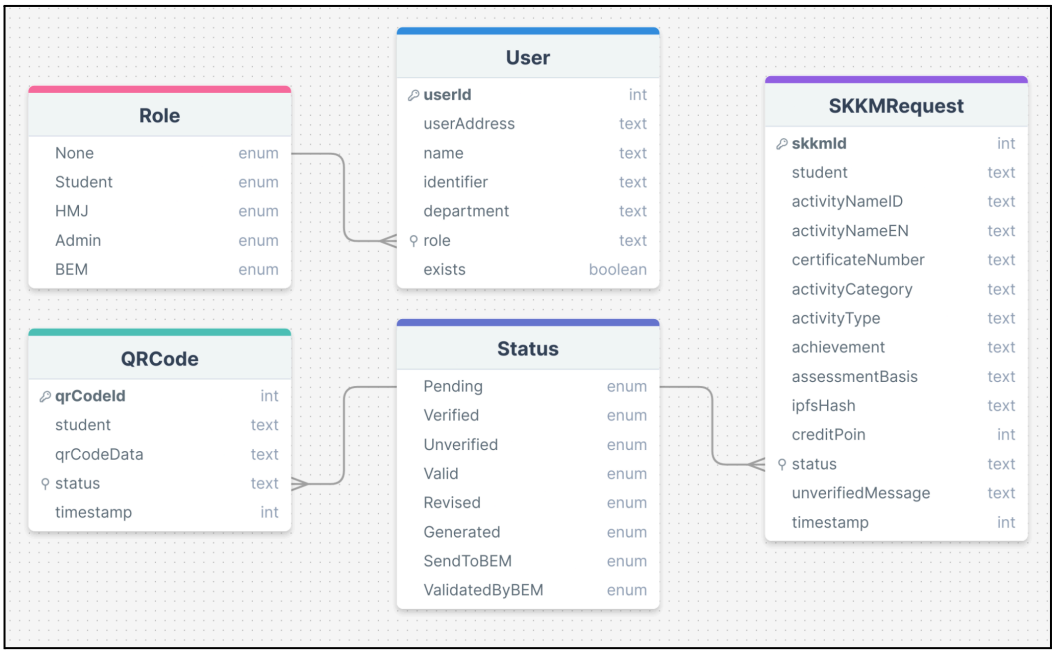
- *getSKKMByStudent*: Mendapatkan daftar SKKM berdasarkan mahasiswa.
- getSKKMByQRCode: Mendapatkan daftar SKKM berdasarkan *QR Code*.
- getQRNotValidatedByBEM: Mendapatkan daftar *QR Code* yang belum divalidasi oleh BEM.
- getSKKMNotVerifiedByHMJ: Mendapatkan daftar SKKM yang belum diverifikasi oleh HMJ.
- getSKKMNotValidatedByHMJ: Mendapatkan daftar SKKM yang sudah diverifikasi tapi belum divalidasi oleh HMJ.
- setMinimalPoin: Mengatur jumlah poin minimal SKKM yang harus dicapai mahasiswa.
- getQRCodeByStudent: Mendapatkan *QR Code* berdasarkan mahasiswa.
- getAllSKKM: Mendapatkan semua SKKM.
- getAllUsers: Mendapatkan semua pengguna.
- getAllQRCodes: Mendapatkan semua *QR Code*.

Smart contract ini akan dilengkapi dengan *modifier*. *Modifier* ini akan digunakan pada fungsi-fungsi yang relevan untuk memastikan bahwa hanya pengguna dengan peran yang tepat yang dapat melakukan tindakan tertentu. Berikut untuk mengatur hak akses setiap pengguna sesuai dengan peran mereka dalam sistem:

- onlySuperAdmin: Hanya dapat dipanggil oleh *Super Admin*.
- onlyAdmin: Hanya dapat dipanggil oleh *Admin* atau *Super Admin*.
- onlyStudent: Hanya dapat dipanggil oleh Mahasiswa.
- onlyHMJ: Hanya dapat dipanggil oleh HMJ.

- `onlyBEM`: Hanya dapat dipanggil oleh BEM.

Dengan perancangan *smart contract* yang lebih sesuai ini, diharapkan sistem pencatatan SKKM berbasis *blockchain* dapat berfungsi dengan baik dan memenuhi kebutuhan pengguna.



Gambar 4.2 *Schema Smart Contract*

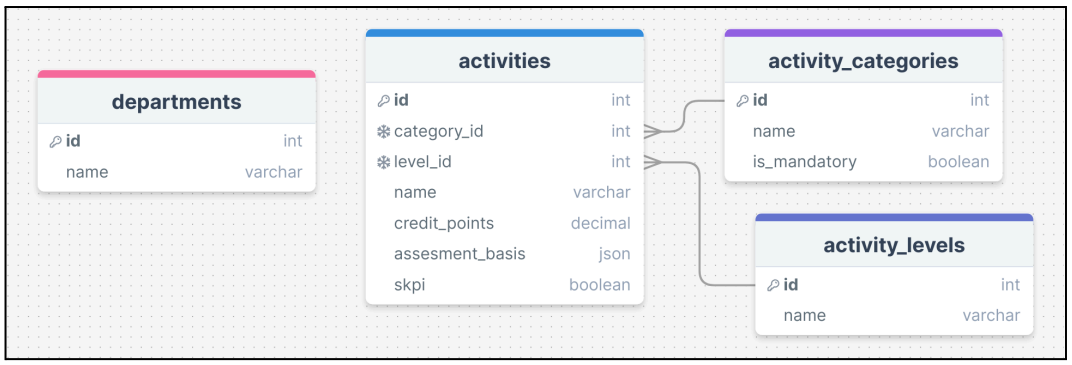
4.4.2 Perancangan Basis Data

Basis data *MySQL* akan digunakan untuk menyimpan data yang bersifat statis dan tidak perlu dicatat di *blockchain*, seperti data departemen, kategori kegiatan, level kegiatan, beserta detail setiap kegiatan SKKM. *MySQL* dipilih karena kehandalannya, skalabilitas nya, dan dukungan komunitas yang luas.

Berikut ini adalah penjelasan detail mengenai tabel-tabel yang terdapat dalam basis data:

1. Tabel departments:
    - `department_id`: *ID* unik untuk setiap departemen.
    - `department_name`: Nama departemen.
2. Tabel activity_categories:
    - `category_id`: *ID* unik untuk setiap kategori kegiatan.
    - `category_name`: Nama kategori kegiatan.
    - `is_mandatory` : Menunjukkan apakah kegiatan dalam kategori ini wajib (true) atau pilihan (false).

3. Tabel activity_levels:
    - `level_id`: *ID* unik untuk setiap level kegiatan.
    - `level_name` : Nama level kegiatan.
4. Tabel activities:
    - `activity_id` : *ID* unik untuk setiap kegiatan SKKM.
    - `category_id` : *ID* kategori kegiatan.
    - `level_id` : *ID* level kegiatan.
    - `activity_name` : Nama kegiatan.
    - `credit_points`: Jumlah poin SKKM yang diberikan.
    - `assessment_basis_id` : *Array* berisi dasar penilaian (contoh: ["SERTIFIKAT", "SK"]). *MySQL* mendukung tipe data *JSON* untuk menyimpan *array* dan objek *JSON*.
    - `skpi` (*BOOLEAN NOT NULL*): Menunjukkan apakah kegiatan ini termasuk dalam SKPI (*true* atau *false*).



Gambar 4.3 *Schema SQL*

4.4.3 Perancangan Antarmuka Pengguna (UI)

Antarmuka pengguna (UI) aplikasi akan dibangun menggunakan ReactJS, sebuah pustaka *JavaScript* yang populer untuk membangun *UI* yang interaktif dan dinamis. *UI* akan dirancang agar mudah digunakan dan intuitif, dengan navigasi yang jelas dan tampilan informasi yang terstruktur. Desain *UI* akan disesuaikan dengan kebutuhan dan preferensi pengguna, yaitu mahasiswa, HMJ, dan BEM. Berikut adalah beberapa halaman utama dalam aplikasi web:

1. Halaman *Dashboard*: Untuk Mahasiswa, menampilkan ringkasan SKKM mahasiswa dan menampilkan semua aktivitas yang dicatat di *blockchain*. Untuk HMJ, BEM, halaman *dashboard* akan menampilkan jumlah daftar

pengajuan SKKM yang perlu diverifikasi atau divalidasi. Untuk *Admin* / *Super Admin*, halaman *dashboard* akan menampilkan jumlah Pengguna.

2. Halaman Detail Kegiatan: Menampilkan daftar SKKM yang telah diajukan oleh mahasiswa, dengan detail lebih lengkap. Menyediakan opsi untuk mengedit atau menghapus SKKM, dan juga sebagai pembuatan *QR Code*.
3. Halaman Pengajuan SKKM: Memungkinkan mahasiswa untuk memasukkan data kegiatan yang telah diikuti, mengunggah bukti partisipasi, dan mengirimkan pengajuan SKKM.
4. Halaman *Validation*: Menampilkan formulir untuk memasukkan string unik *QR Code*, menunjukkan status validasi *QR code*, dan juga untuk mengirim *QR Code* ke BEM untuk di validasi
5. Halaman Verifikasi SKKM: Memungkinkan HMJ untuk melihat detail pengajuan SKKM, termasuk bukti partisipasi, dan memberikan status verifikasi.
6. Halaman Validasi SKKM: Memungkinkan HMJ untuk melihat detail SKKM yang telah diverifikasi oleh HMJ dan memberikan status validasi.
7. Halaman Validasi *QR Code*: Menampilkan daftar *QR Code* SKKM yang diajukan mahasiswa dan memungkinkan BEM untuk melakukan validasi.

UI juga akan dilengkapi dengan fitur-fitur pendukung seperti notifikasi, histori aktivitas, dan bantuan pengguna untuk meningkatkan pengalaman pengguna. Dengan perancangan *UI* yang disesuaikan ini, diharapkan aplikasi dapat memberikan pengalaman pengguna yang lebih baik dan sesuai dengan alur kerja yang telah ditentukan dalam skenario *use case*.

4.4.4 Perancangan Jaringan *Blockchain*

Jaringan *blockchain* yang digunakan dalam sistem ini adalah *private network* berbasis *Ethereum* yang dibangun menggunakan *Hyperledger Besu*. *Private network* dipilih karena memberikan kontrol penuh atas siapa saja yang dapat bergabung dalam jaringan, sehingga meningkatkan keamanan dan privasi data SKKM. *Hyperledger Besu* dipilih sebagai klien *Ethereum* karena kemampuannya dalam mendukung fitur-fitur *enterprise* seperti *permissioning* (pengaturan izin akses) dan *private transactions*. Selain itu, *Hyperledger Besu* juga mendukung berbagai mekanisme konsensus, termasuk *QBFT*

(Quorum-Based Byzantine Fault Tolerance) yang dipilih untuk digunakan dalam sistem ini. *QBFT* menawarkan finalitas instan, toleransi kesalahan yang tinggi, dan keamanan yang kuat, menjadikannya pilihan yang ideal untuk aplikasi pencatatan SKKM yang membutuhkan kepastian dan keandalan.

a. Konfigurasi Jaringan

   Jaringan *blockchain* akan terdiri dari beberapa *node* yang terhubung satu sama lain. Setiap *node* akan menjalankan instance *Hyperledger Besu* dan menyimpan salinan lengkap dari *blockchain*. *Node-node* ini akan berkomunikasi satu sama lain untuk memvalidasi transaksi dan mencapai konsensus tentang status *blockchain*. Konfigurasi jaringan akan diatur dalam file `qbftConfigFile.json`.

b. Pembuatan Kunci dan Konfigurasi *Node*

   Untuk setiap *node* dalam jaringan, akan dibuat sepasang kunci publik dan privat menggunakan script `generateNodeKeys.sh`. Kunci-kunci ini akan digunakan untuk mengidentifikasi dan mengamankan *node* dalam jaringan. Script `setupQBFT.sh` akan digunakan untuk menyalin file konfigurasi dan kunci-kunci yang diperlukan ke direktori masing-masing *node*.

c. Inisialisasi dan Penambahan *Validator Node*

   Node pertama yang dijalankan akan menjadi *bootnode*, yaitu *node* yang menjadi titik awal bagi *node* lain untuk bergabung dalam jaringan. Script `startAllNodes.sh` akan memulai *bootnode* dan *node-node* lainnya, serta menyimpan informasi penting seperti *enode URL* dan alamat *node* dan juga sekaligus menjalankan quorum explorer untuk memonitor *node-node* tersebut. Jika ingin menjalankan satu *node* tertentu, bisa menggunakan script `startNodes.sh` dengan memberikan argumen nama *node* tertentu. Script ini akan memastikan *node* yang bersangkutan belum berjalan dan kemudian menjalankannya.

   Setelah jaringan berjalan, script `confirmNetwork.sh` akan digunakan untuk memeriksa status jaringan dari satu *node* tertentu, memastikan apakah *node* tersebut berfungsi dengan benar sebagai *validator*. Untuk memeriksa status semua *node* dalam jaringan, script

`confirmAllNetwork.sh` dapat digunakan. Script ini akan memastikan bahwa semua *node* dalam jaringan berfungsi dengan benar dan mencapai konsensus. *Validator node* adalah *node* yang bertanggung jawab untuk memvalidasi transaksi dan mencapai konsensus. Jika diperlukan, *validator node* tambahan dapat ditambahkan ke jaringan menggunakan script `addValidator.sh`. Script ini akan memulai *node* baru, menambahkannya ke daftar *bootnode*, dan mengusulkan penambahan *validator* baru dari mayoritas *node* yang ada.

d. Penghentian dan Penghapusan *Validator Node*

Untuk menghentikan semua *node* dalam jaringan, dapat digunakan script `stopAllNodes.sh`. Jika ingin menghentikan satu *node* tertentu, bisa menggunakan script `stopNodes.sh` dengan memberikan argumen nama *node* tertentu. Script ini akan memastikan *node* yang bersangkutan dihentikan dengan benar. Jika ingin menghapus *validator node* dari jaringan, dapat digunakan script `removeValidator.sh`.

Dengan perancangan jaringan *blockchain* yang tepat, sistem pencatatan SKKM ini dapat berjalan dengan aman, efisien, dan terukur, serta mampu mengakomodasi kebutuhan pengelolaan SKKM di Politeknik Negeri Malang (POLINEMA) yang terus berkembang.

4.4.5 Perancangan Keamanan

Keamanan merupakan aspek krusial dalam sistem pencatatan SKKM berbasis *blockchain* ini, mengingat data SKKM bersifat sensitif dan harus dilindungi dari akses yang tidak sah, manipulasi, atau kehilangan. Oleh karena itu, perancangan keamanan sistem ini akan mencakup beberapa lapisan perlindungan, baik pada tingkat jaringan *blockchain*, *smart contract*, basis data, maupun aplikasi web.

1. Keamanan Jaringan *Blockchain* (*Hyperledger Besu*):
    a. *Private Network*: Penggunaan *private network Hyperledger Besu* memastikan bahwa hanya *node-node* yang terdaftar dan memiliki izin yang dapat bergabung dalam jaringan. Hal ini membatasi akses dari pihak luar yang tidak berwenang.

b. *QBFT* Consensus: Mekanisme konsensus *QBFT* (Quorum-Based Byzantine Fault Tolerance) memberikan keamanan yang tinggi terhadap serangan Byzantine fault, yaitu serangan di mana *node-node* dalam jaringan memberikan informasi yang salah atau bertentangan. *QBFT* memastikan bahwa hanya transaksi yang valid yang akan dicatat di *blockchain*.

2. Keamanan *Smart Contract*:
    a. Pengujian Unit dan Integrasi: *Smart contract* akan diuji secara menyeluruh dengan berbagai skenario untuk memastikan bahwa fungsi-fungsi *smart contract* berjalan sesuai dengan yang diharapkan dan tidak memiliki bug atau kesalahan logika.
    b. Pembatasan Akses: *Smart contract* akan dilengkapi dengan *modifier* untuk membatasi akses ke fungsi-fungsi tertentu. Misalnya, hanya admin yang dapat menambah, mengubah, atau menghapus pengguna, sedangkan mahasiswa hanya dapat mengajukan dan melihat SKKM mereka sendiri.
3. Keamanan Aplikasi Web (ReactJS):
    a. Autentikasi Pengguna: Aplikasi web akan menggunakan mekanisme autentikasi yang kuat, seperti penggunaan dompet digital (misalnya, *MetaMask*), untuk memastikan bahwa hanya pengguna yang sah yang dapat mengakses sistem.

Dengan menerapkan berbagai lapisan keamanan ini, diharapkan sistem pencatatan SKKM berbasis *blockchain* ini dapat memberikan perlindungan yang kuat terhadap berbagai ancaman keamanan, sehingga menjamin keamanan dan integritas data SKKM.

4.4.6 Koneksi ke *MetaMask*

MetaMask adalah dompet kripto populer yang berfungsi sebagai jembatan antara aplikasi web (frontend) dan jaringan *blockchain Ethereum*. *MetaMask* memungkinkan pengguna untuk berinteraksi dengan *smart contract* di *blockchain*, menandatangani transaksi, dan mengelola aset kripto mereka. Dalam konteks aplikasi pencatatan SKKM berbasis *blockchain*, *MetaMask* akan digunakan oleh

mahasiswa, HMJ, BEM, admin dan super admin untuk berinteraksi dengan *smart contract* yang telah di-*deploy* di jaringan *Hyperledger Besu*.

1. Instalasi dan Konfigurasi *MetaMask*
    a. Unduh dan Pasang: *MetaMask* tersedia sebagai ekstensi browser (misalnya, Google Chrome, Mozilla Firefox) atau aplikasi *mobile*. Pengguna perlu mengunduh dan memasang *MetaMask* sesuai dengan perangkat yang mereka gunakan.
    b. Buat atau Impor Dompet: Setelah terpasang, pengguna dapat membuat dompet baru atau mengimpor dompet yang sudah ada menggunakan *seed phrase* (frasa pemulihan). Dompet ini akan digunakan untuk menyimpan kunci privat pengguna dan berinteraksi dengan *blockchain*.
    c. Hubungkan ke Jaringan: *MetaMask* perlu dihubungkan ke jaringan *blockchain* yang sesuai. Dalam kasus ini, pengguna perlu menambahkan jaringan *Hyperledger Besu* ke *MetaMask*. Informasi jaringan seperti *RPC URL*, *Chain ID*, dan nama jaringan dapat diperoleh dari konfigurasi *Hyperledger Besu*.
    d. Tambah Akun: Pengguna dapat menambahkan beberapa akun ke *MetaMask*. Setiap akun memiliki alamat *Ethereum* yang berbeda dan dapat digunakan untuk berinteraksi dengan *smart contract*.
2. Integrasi *MetaMask* dengan Aplikasi Web
    a. Deteksi *MetaMask*: Aplikasi web perlu mendeteksi apakah *MetaMask* terpasang di browser pengguna. Jika tidak terpasang, aplikasi dapat menampilkan pesan yang meminta pengguna untuk memasang *MetaMask*.
    b. Koneksi ke *MetaMask*: Aplikasi web akan meminta izin pengguna untuk terhubung ke *MetaMask*. Setelah pengguna memberikan izin, aplikasi dapat mengakses akun *MetaMask* yang dipilih oleh pengguna dan berinteraksi dengan *blockchain*.
    c. Sign dan Kirim Transaksi: Ketika pengguna melakukan tindakan yang memerlukan interaksi dengan *smart contract* (misalnya, mengajukan SKKM, memverifikasi SKKM, membuat *QR Code*),

aplikasi web akan membuat transaksi yang sesuai dan meminta *MetaMask* untuk menandatanganinya. Setelah ditandatangani, transaksi akan dikirim ke jaringan *blockchain Hyperledger Besu* untuk diproses.

d. Tangani *Event*: Aplikasi web perlu mendengarkan *event* yang dihasilkan oleh *smart contract*, seperti *event* yang menunjukkan bahwa SKKM telah diverifikasi atau *QR Code* telah dibuat. Dengan menangani *event* ini, aplikasi dapat memperbarui tampilan antarmuka pengguna secara dinamis dan memberikan umpan balik yang relevan kepada pengguna.

Dengan mengintegrasikan *MetaMask* ke dalam aplikasi web, pengguna dapat dengan mudah berinteraksi dengan sistem pencatatan SKKM berbasis *blockchain* tanpa harus memiliki pengetahuan teknis yang mendalam tentang *blockchain* atau *smart contract*. *MetaMask* akan menangani kompleksitas teknis di balik layar, sementara pengguna dapat fokus pada tindakan yang ingin mereka lakukan dalam aplikasi.

4.4.7 Pembuatan dan *Deployment Smart Contract*

Pembuatan dan *deployment smart contract* adalah tahapan krusial dalam pengembangan aplikasi berbasis *blockchain*. Tahapan ini melibatkan penulisan kode *smart contract*, pengujian, dan penerapannya ke jaringan *blockchain*. Dalam konteks aplikasi pencatatan SKKM, *smart contract* akan menjadi inti dari sistem, mengatur semua logika bisnis dan interaksi data di *blockchain*. *Thirdweb CLI* menyederhanakan proses ini dengan menyediakan antarmuka yang mudah digunakan dan terintegrasi dengan baik.

1. Persiapan Lingkungan Pembuatan dan *Deployment*

   Sebelum membuat dan menyebarkan *smart contract*, beberapa persiapan lingkungan perlu dilakukan untuk memastikan proses berjalan lancar:

   a. *Node Hyperledger Besu*: Peneliti memastikan *node Hyperledger Besu* telah terinstal dan berjalan, terhubung ke jaringan *blockchain* yang diinginkan.

b. *Thirdweb CLI*: Dengan *Thirdweb CLI*, peneliti dapat dengan mudah membuat, mengelola, dan *men-deploy smart contract* ke berbagai jaringan *blockchain* yang kompatibel dengan *Ethereum Virtual Machine* (EVM), termasuk *Hyperledger Besu*.

c. *Text Editor* atau *IDE*: Peneliti dapat memilih *text editor* atau *Integrated Development Environment* (*IDE*) yang sesuai dengan preferensi dan kebutuhan mereka. Beberapa opsi populer untuk pengembangan *Solidity* (bahasa pemrograman *smart contract*) meliputi *Visual Studio Code*, *Remix* (IDE berbasis web).

d. Akun *Ethereum* dan *Test Ether*: Peneliti memerlukan akun *Ethereum* yang akan digunakan untuk mendeploy *smart contract*.

2. Pembuatan dan *Deployment Smart Contract* dengan *Thirdweb*

a. Membuat Proyek *Smart Contract*: Buka terminal atau command prompt dan arahkan ke direktori tempat ingin membuat proyek *smart contract*. Kemudian menjalankan perintah `npx thirdweb create contract`.

b. Menyesuaikan *Smart Contract* (Opsional): Jika memilih template *smart contract*, peneliti dapat menyesuaikan kode *Solidity* yang dihasilkan sesuai dengan kebutuhan spesifik aplikasi pencatatan SKKM.

c. *Deployment Smart Contract*: Setelah kode *smart contract* siap, jalankan perintah `npx thirdweb deploy`. *Thirdweb* akan secara otomatis mendeteksi jaringan *blockchain* yang telah di konfigurasi di *node Hyperledger Besu*.

3. Verifikasi *Deployment*

a. *Dashboard Thirdweb*: Setelah *deployment* berhasil, peneliti dapat melihat dan mengelola *smart contract* melalui *dashboard Thirdweb*.

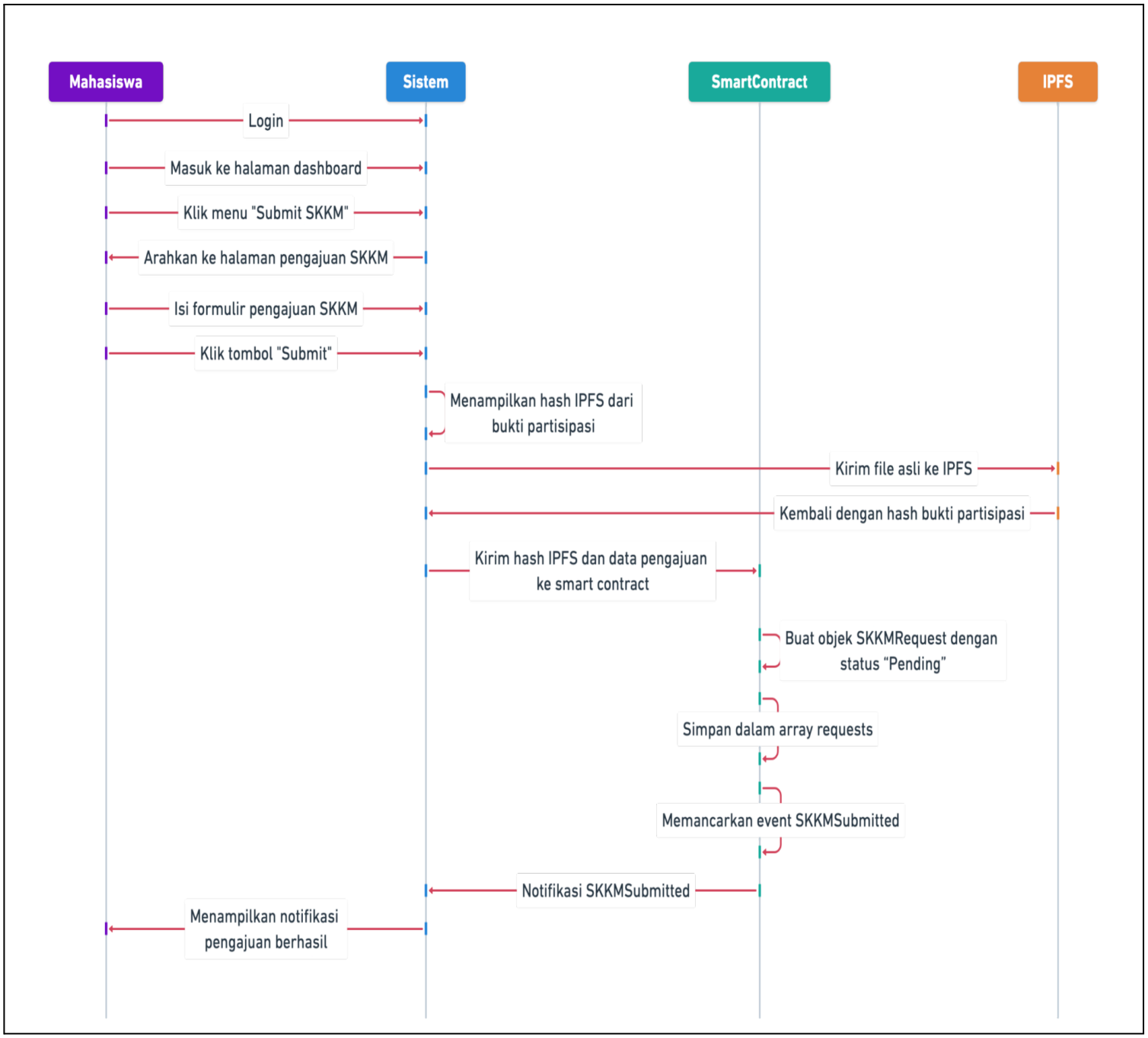
4.5 Diagram Urutan (*Sequence Diagram*)

Diagram-diagram ini menggambarkan urutan pesan yang dipertukarkan antara aktor-aktor tersebut, memberikan pemahaman yang jelas tentang alur kerja dan logika bisnis dari setiap *use case*.

4.5.1 Mahasiswa

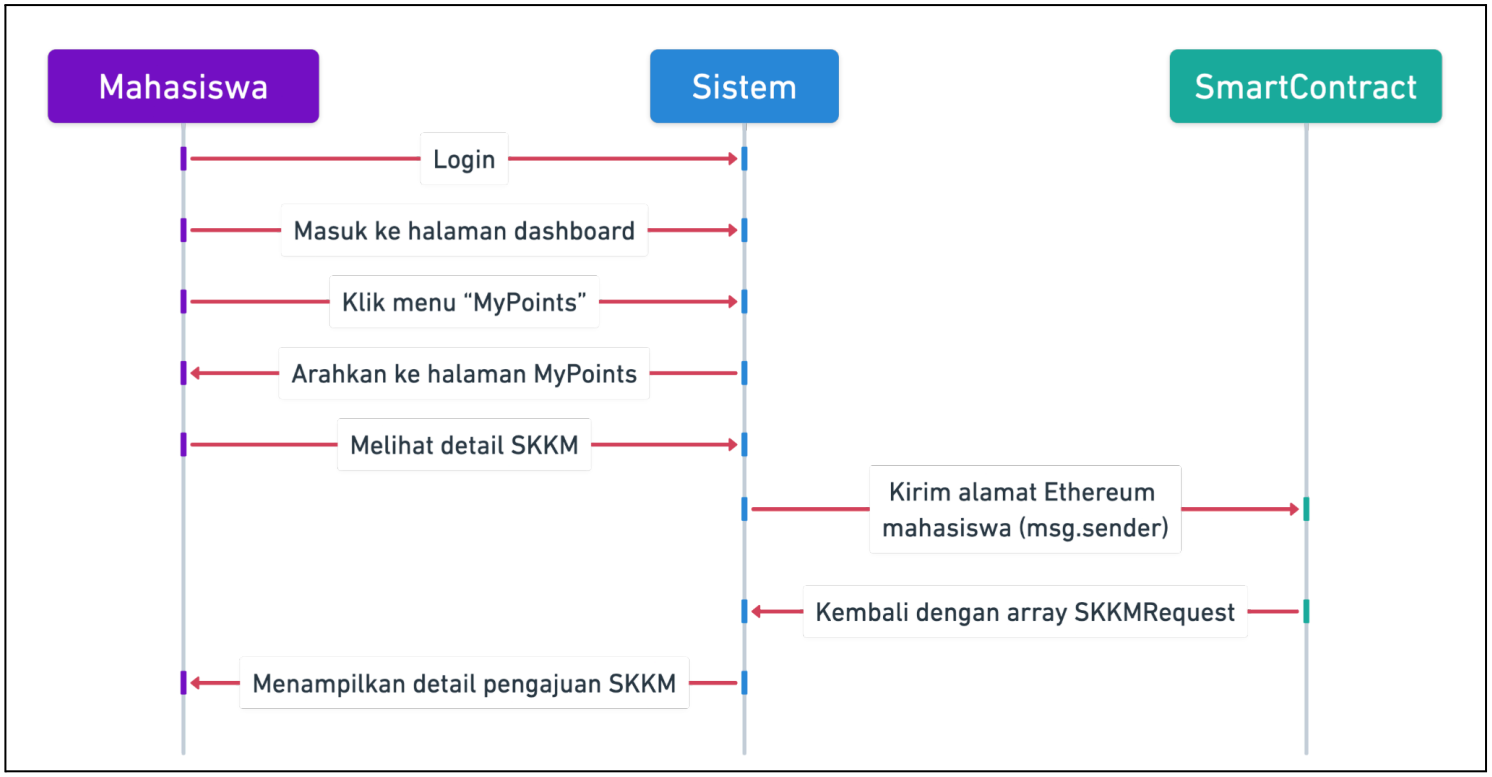
Berikut adalah *sequence diagram* yang menggambarkan interaksi antara mahasiswa dengan sistem dalam berbagai aktivitas yang didukung oleh aplikasi pencatatan SKKM berbasis *blockchain* :

1. Mengajukan SKKM



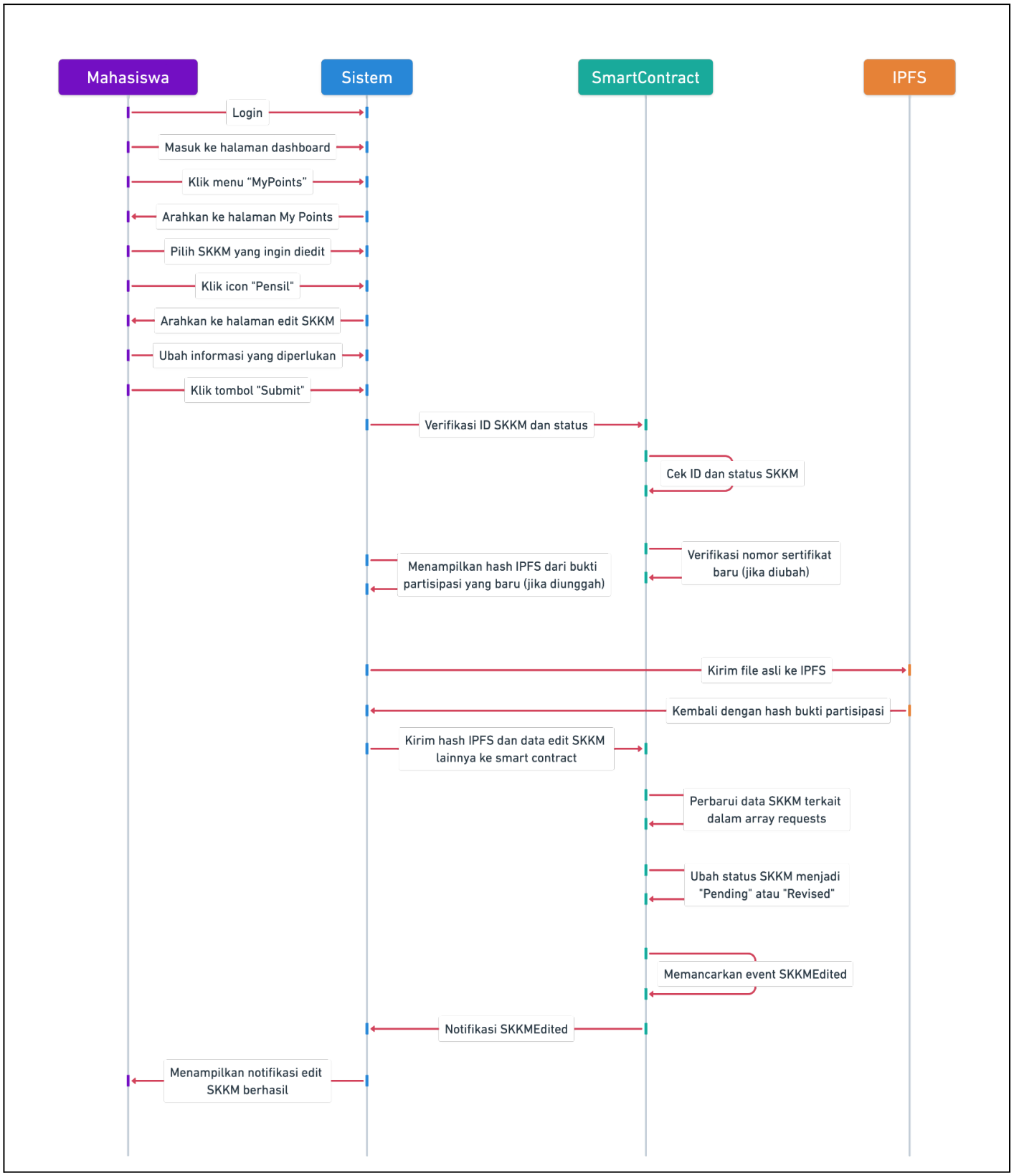
Gambar 4.4 *Sequence Diagram*: Mengajukan SKKM

2. Melihat Status Pengajuan SKKM



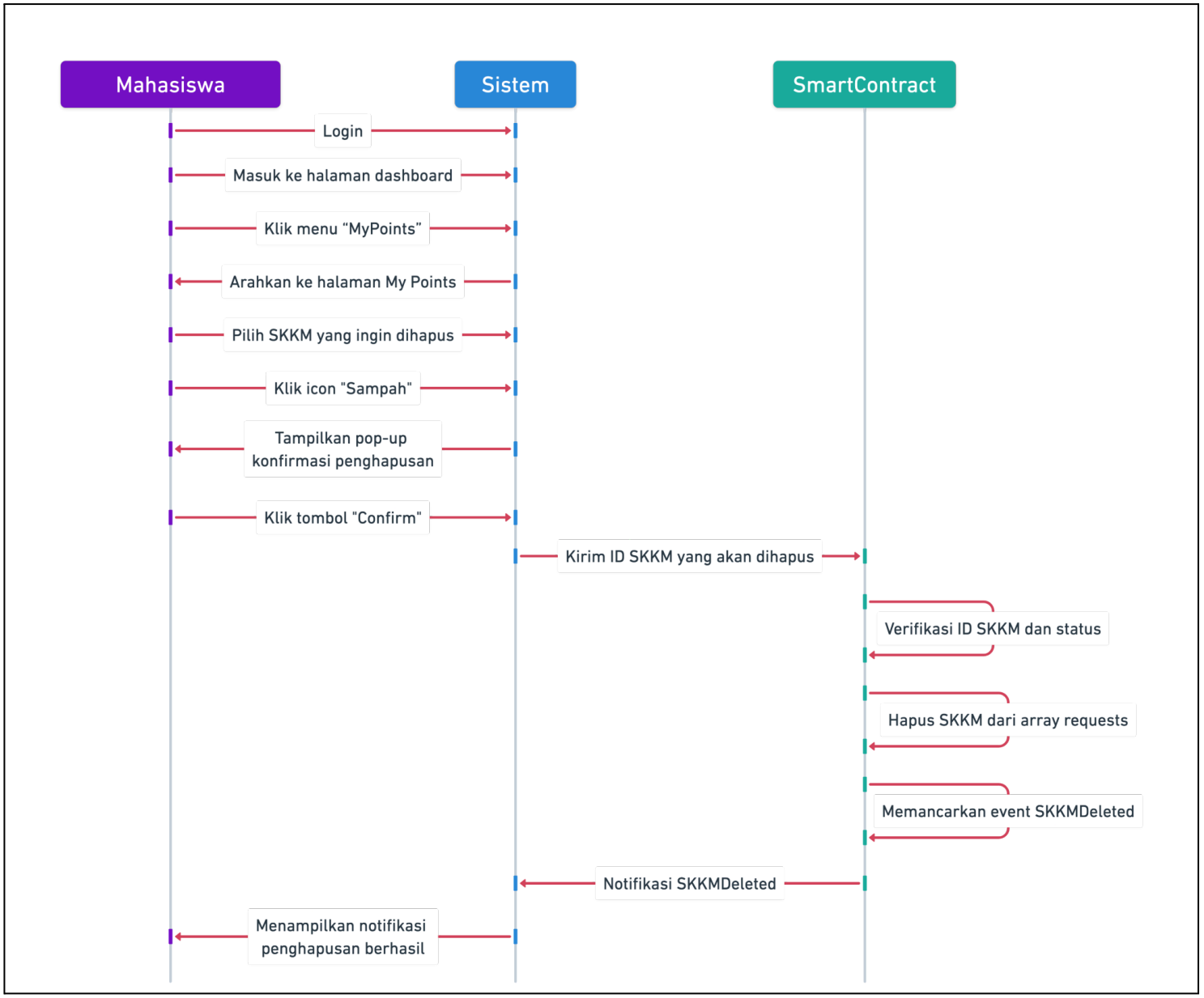
Gambar 4.5 *Sequence Diagram*: Melihat Status Pengajuan SKKM

3. Mengedit SKKM



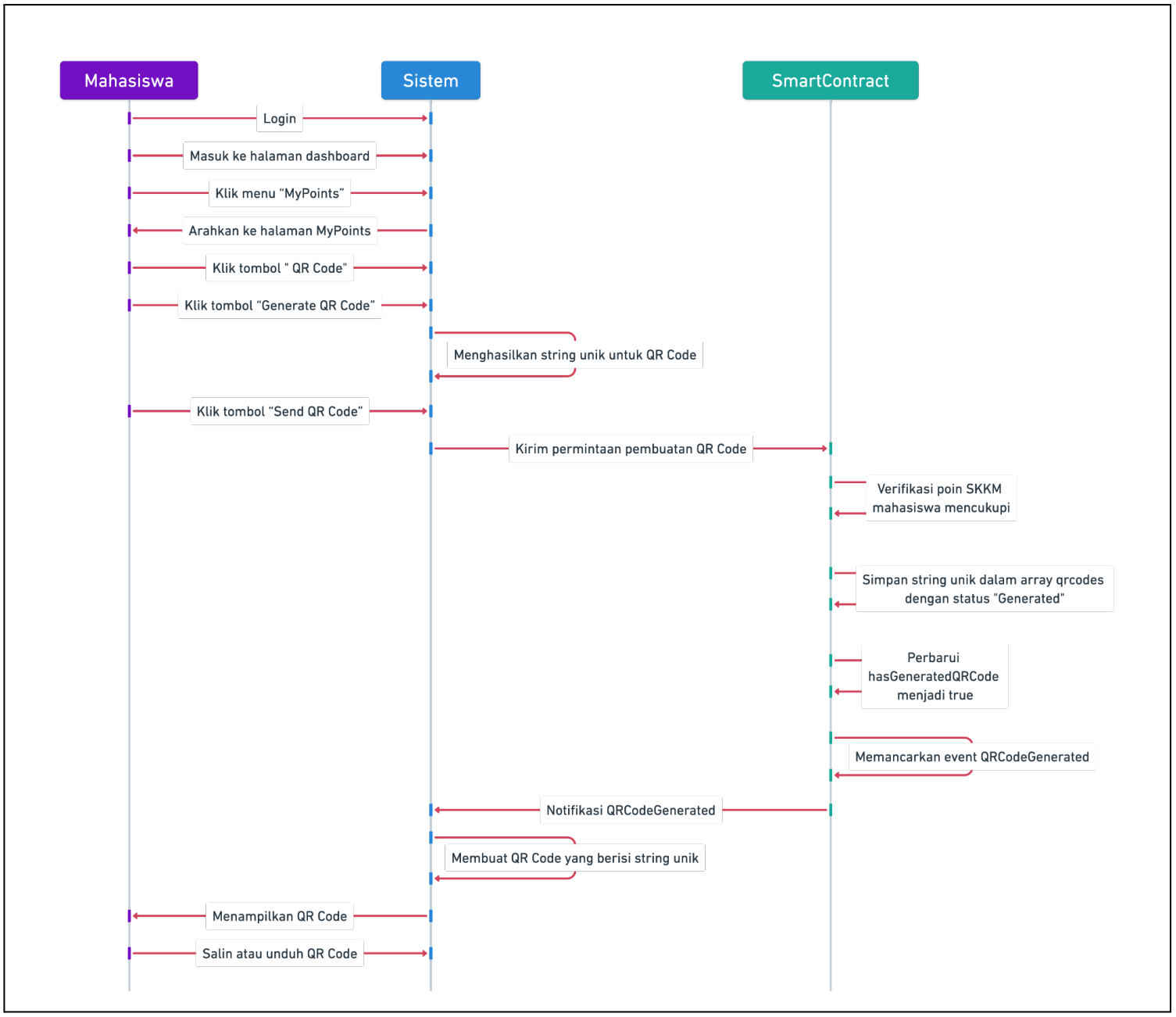
Gambar 4.6 *Sequence Diagram*: Mengedit SKKM

4. Menghapus SKKM



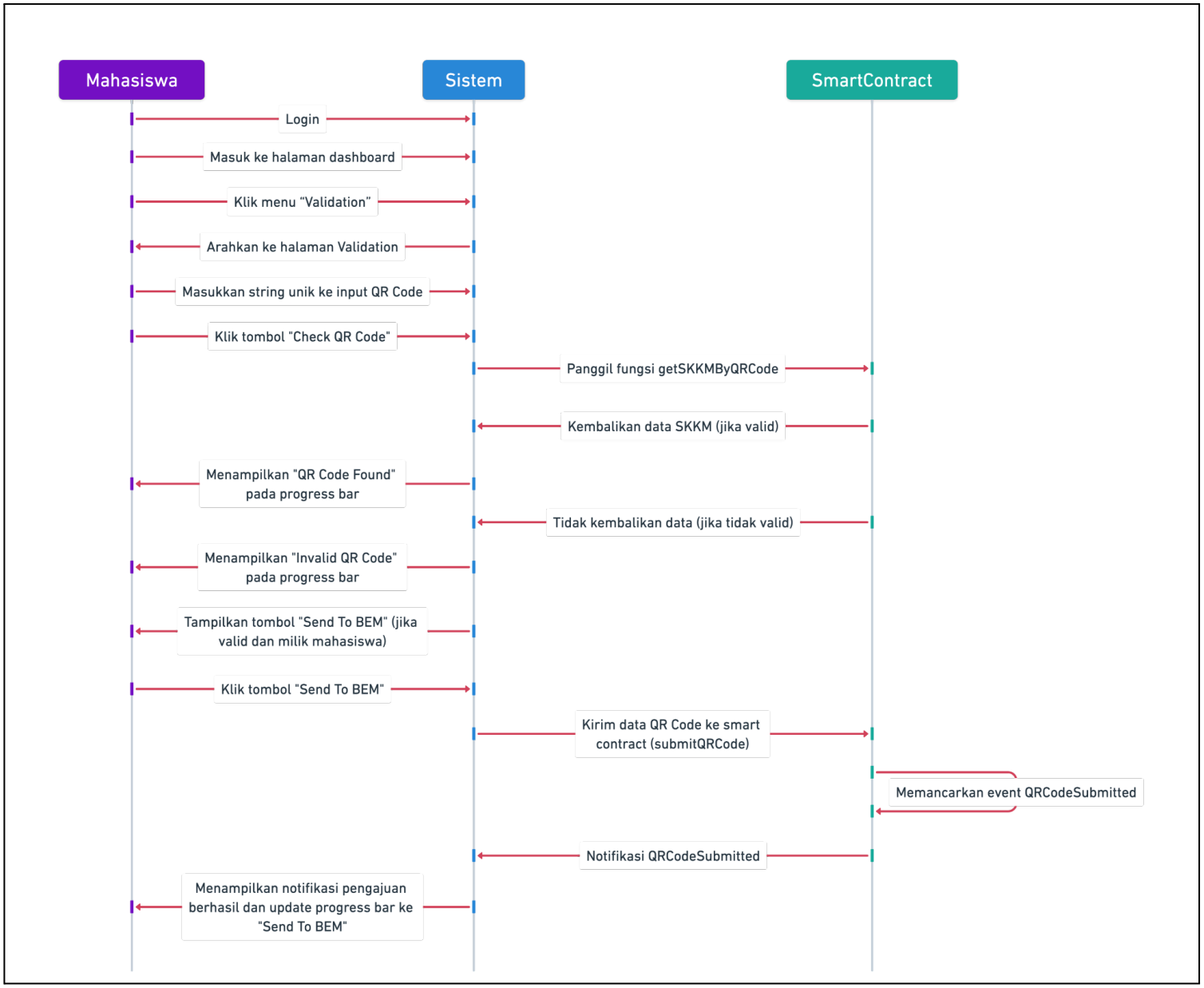
Gambar 4.7 *Sequence Diagram*: Menghapus SKKM

5. Membuat *QR Code* SKKM



Gambar 4.8 *Sequence Diagram*: Membuat *QR Code* SKKM

6. Mengajukan *QR Code* ke BEM

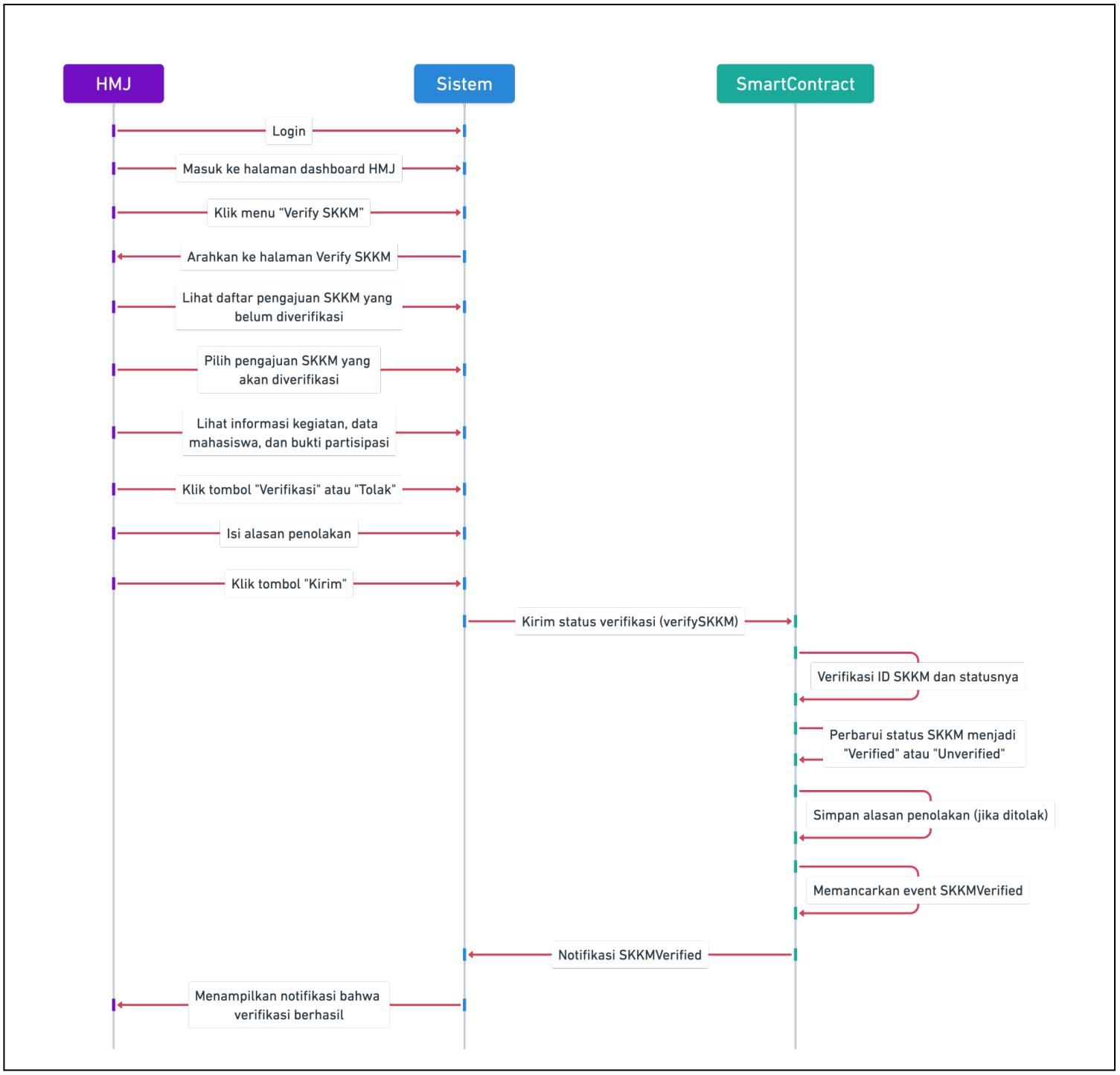


Gambar 4.9 *Sequence Diagram*: Mengajukan *QR Code* ke BEM

4.5.2 HMJ

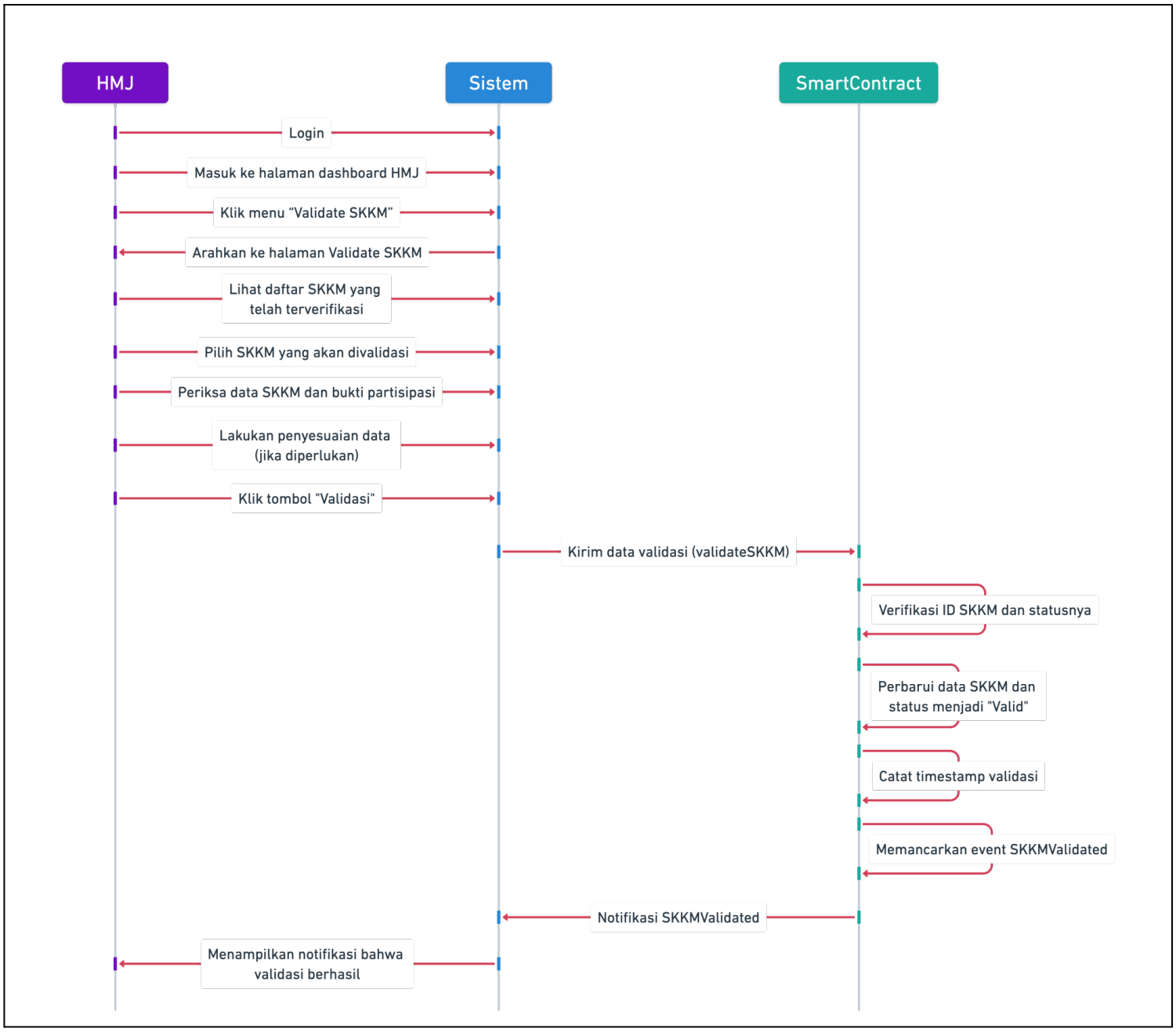
Berikut adalah *sequence diagram* yang menggambarkan interaksi antara HMJ (Himpunan Mahasiswa Jurusan) dengan sistem dalam aktivitas terkait pengecekan SKKM :

1. Memverifikasi Pengajuan SKKM



Gambar 4.10 *Sequence Diagram*: Memverifikasi Pengajuan SKKM

2. Memvalidasi SKKM

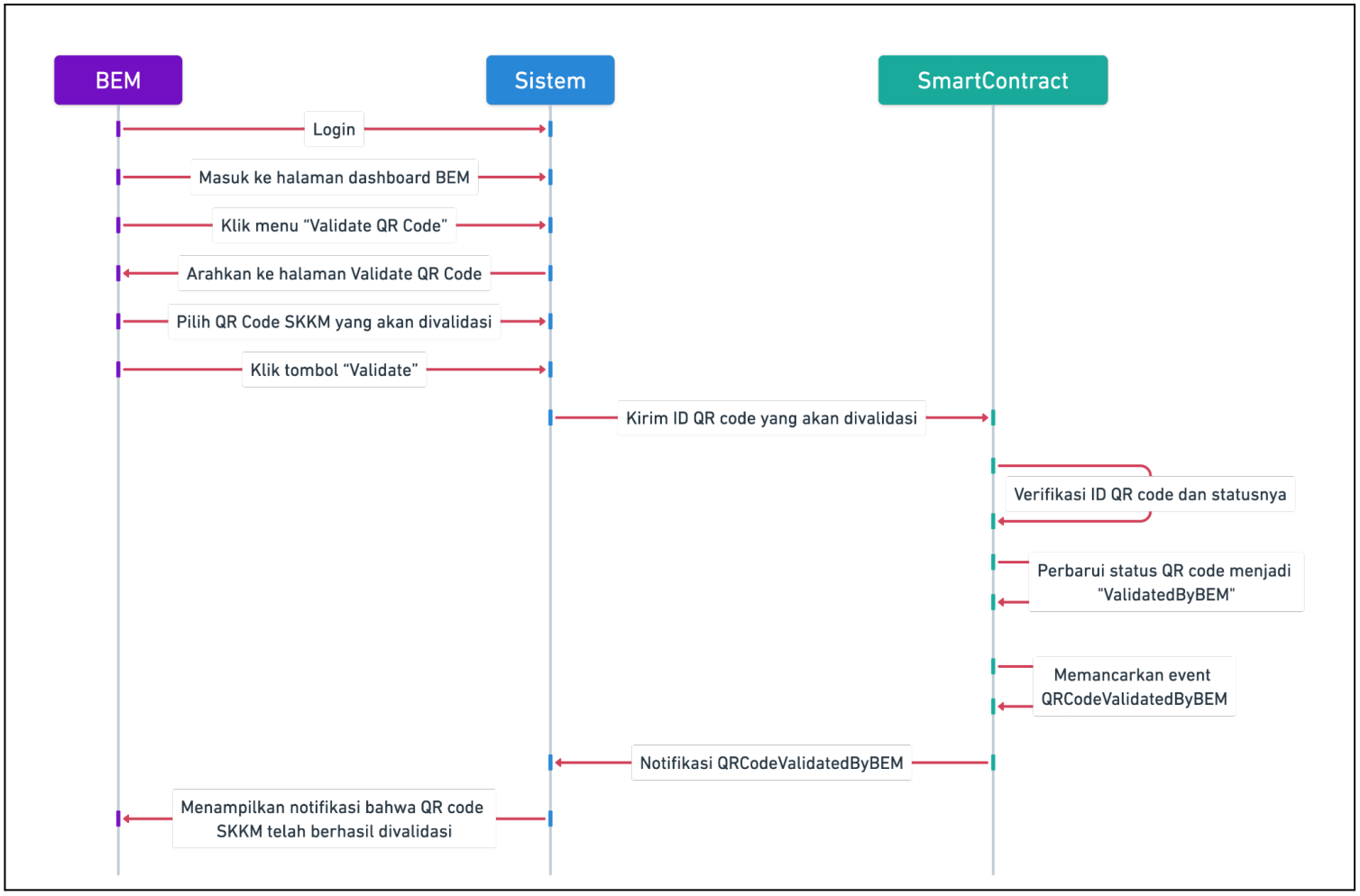


Gambar 4.11 *Sequence Diagram*: Memvalidasi SKKM

4.5.3 BEM

Berikut adalah *sequence diagram* yang menggambarkan interaksi antara BEM (Badan Eksekutif Mahasiswa) dengan sistem dalam aktivitas terkait validasi *Qr Code* SKKM :

1. Memvalidasi *QR Code* SKKM

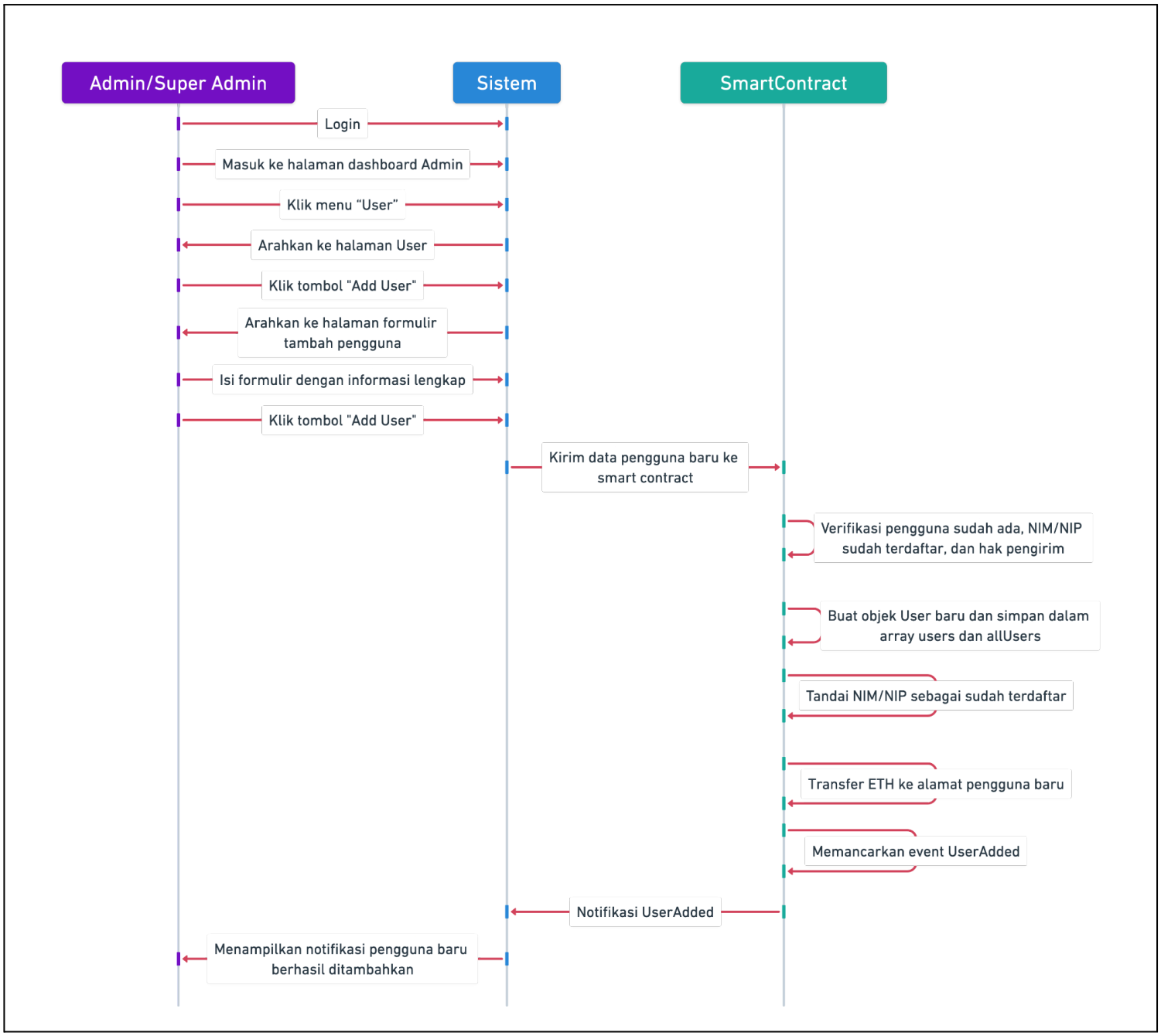


Gambar 4.12 *Sequence Diagram*: Memvalidasi *QR Code* SKKM

4.5.4 *Admin* dan *Super Admin*

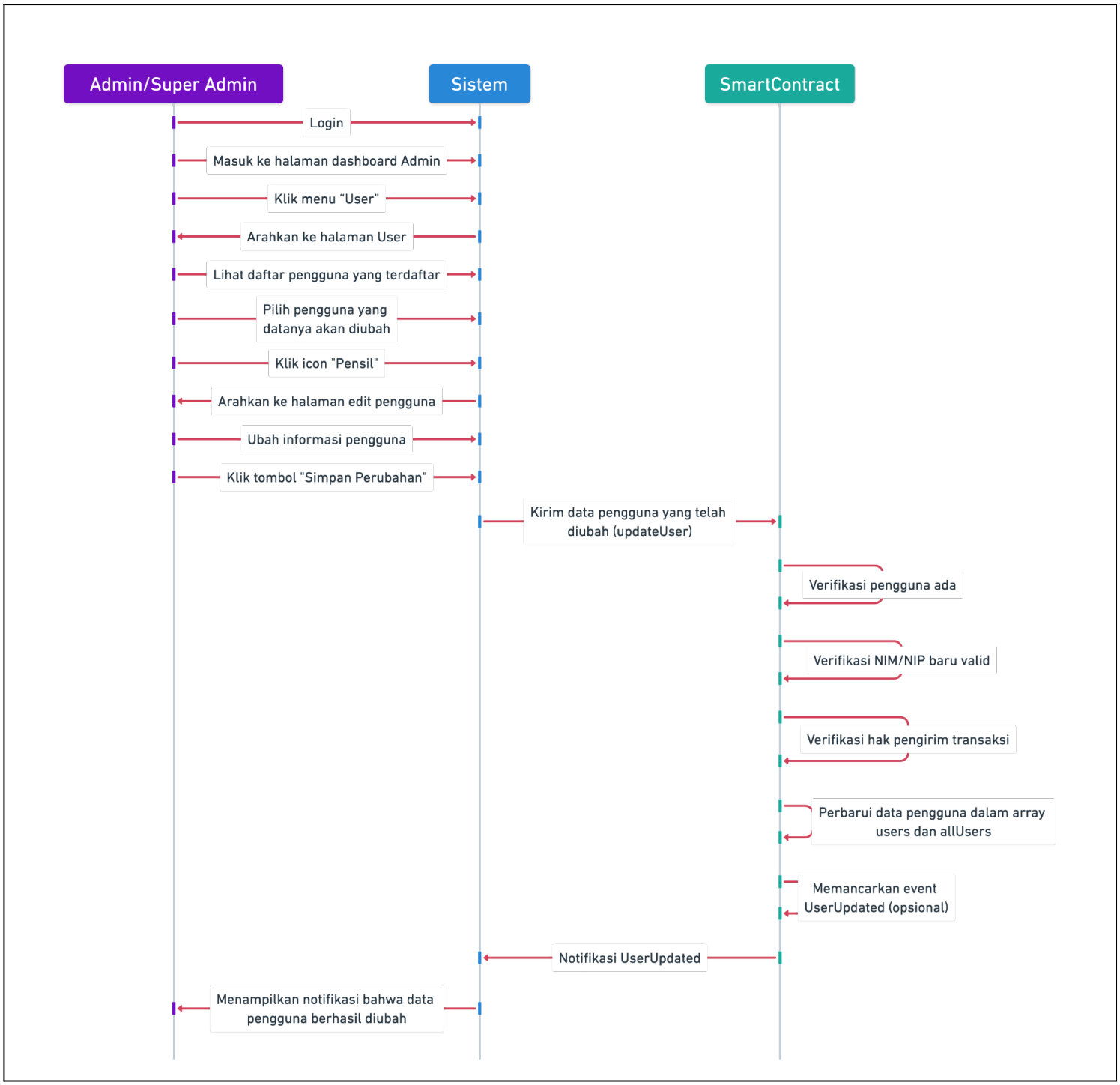
Berikut adalah *sequence diagram* yang menggambarkan interaksi antara *Admin* / *Super Admin* dengan sistem dalam aktivitas terkait pengelolaan sistem :

1. Menambah Pengguna Baru



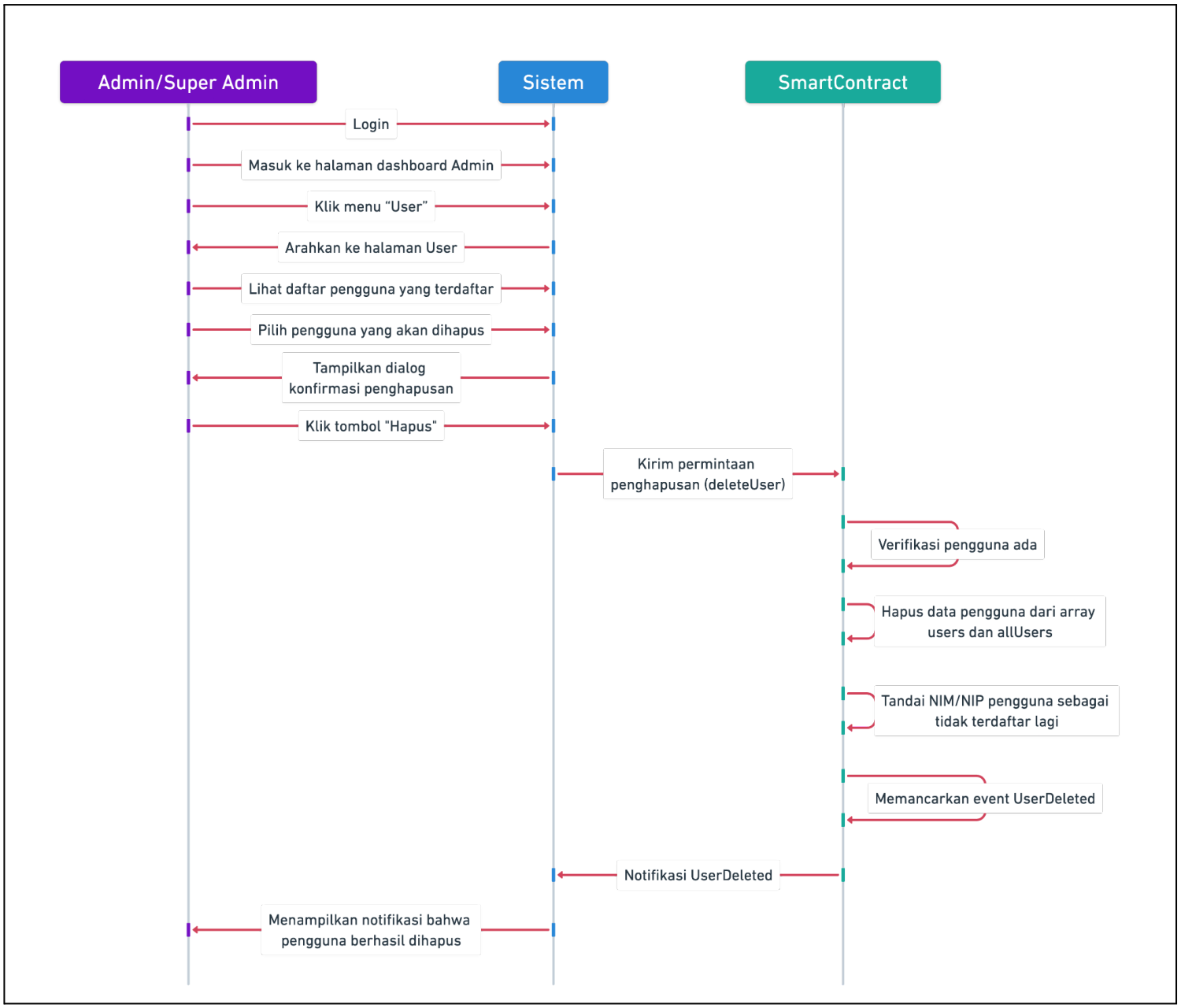
Gambar 4.13 *Sequence Diagram*: Menambah Pengguna Baru

2. Mengubah Data Pengguna



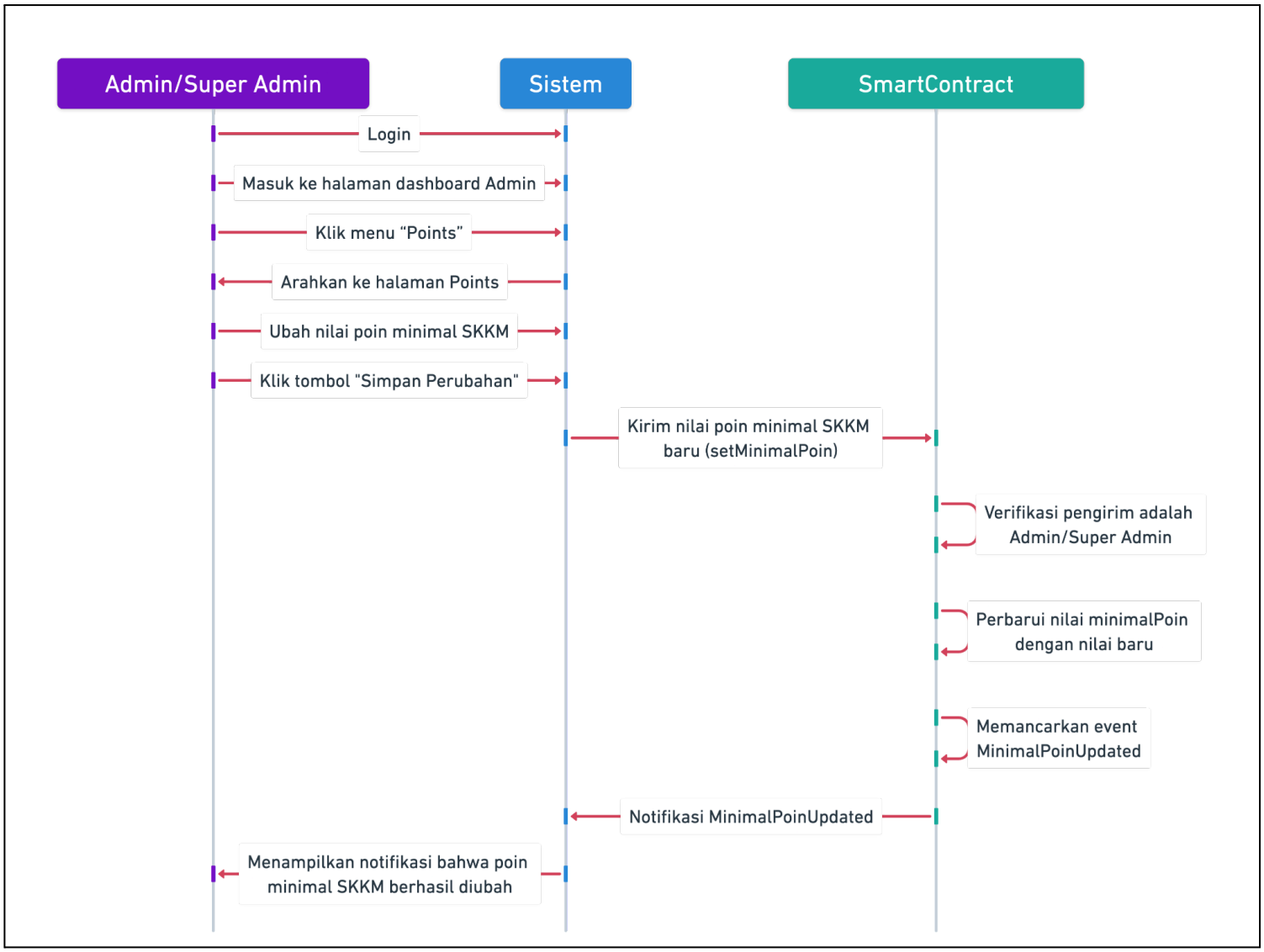
Gambar 4.14 *Sequence Diagram*: Mengubah Data Pengguna

3. Menghapus Pengguna



Gambar 4.15 *Sequence Diagram*: Menghapus Pengguna

4. Mengatur Poin Minimal SKKM



Gambar 4.16 *Sequence Diagram*: Mengatur Poin Minimal SKKM

4.6 Pengujian dan Evaluasi

Pengujian dan evaluasi merupakan tahapan penting untuk memastikan bahwa sistem pencatatan SKKM berbasis *blockchain* berfungsi dengan baik, memenuhi kebutuhan pengguna, dan memiliki kualitas yang memadai. Berikut adalah rencana pengujian dan evaluasi yang akan dilakukan:

4.6.1 Pengujian Fungsionalitas

Pengujian fungsionalitas bertujuan untuk memastikan bahwa semua fitur dan fungsi sistem bekerja sesuai dengan yang diharapkan. Pengujian ini akan mencakup:

1. Pengujian Unit (Unit Testing): Menguji setiap fungsi *smart contract* secara terpisah untuk memastikan bahwa fungsi tersebut bekerja sesuai dengan yang diharapkan. Misalnya, menguji fungsi submitSKKM untuk memastikan bahwa pengajuan SKKM dicatat dengan benar di *blockchain*.
2. Pengujian Integrasi (Integration Testing): Menguji interaksi antara *smart contract*, aplikasi *frontend*, dan *IPFS* untuk memastikan bahwa mereka bekerja sama dengan baik. Misalnya, menguji apakah *frontend* dapat memanggil fungsi *smart contract* dengan benar dan apakah bukti partisipasi dapat diunggah dan diambil dari *IPFS* dengan benar.
3. Pengujian Sistem (System Testing): Menguji keseluruhan sistem, termasuk alur kerja dari awal hingga akhir (misalnya, dari pengajuan SKKM hingga validasi *QR code*), untuk memastikan bahwa sistem memenuhi kebutuhan pengguna dan berfungsi sesuai dengan yang diharapkan.

4.6.2 Pengujian Keamanan

Pengujian keamanan bertujuan untuk mengidentifikasi dan memperbaiki kerentanan keamanan dalam sistem. Pengujian ini akan mencakup:

1. Pengujian Integritas Data: Memastikan bahwa data SKKM tidak dapat diubah atau dimanipulasi tanpa izin. Pengujian ini akan mencakup:
    a. Perubahan Data Tanpa Izin: Mencoba mengubah data SKKM (misalnya, nama kegiatan, poin, status) melalui fungsi-fungsi yang tidak sesuai dengan peran pengguna. Misalnya, mencoba mengubah data SKKM sebagai super admin padahal seharusnya

hanya mahasiswa yang bisa melakukannya. Verifikasi bahwa perubahan tersebut ditolak oleh *smart contract*.

b. Manipulasi Data oleh Pengguna yang Tidak Berwenang: Mencoba mengubah data SKKM dengan menggunakan akun pengguna yang tidak memiliki izin untuk melakukan perubahan tersebut. Misalnya, mencoba mengubah data SKKM milik mahasiswa lain. Verifikasi bahwa perubahan tersebut ditolak oleh *smart contract*.

2. Pengujian Konsensus: Memastikan bahwa mekanisme konsensus *QBFT* (Quorum-Based Fault Tolerance) bekerja dengan baik. Pengujian ini akan mencakup:
    a. Skenario *Node* Gagal: Mensimulasikan kegagalan beberapa *node* dalam jaringan *blockchain* dan memverifikasi bahwa sistem tetap dapat mencapai konsensus dan memproses transaksi.

4.6.3 Evaluasi Kegunaan

Evaluasi kegunaan bertujuan untuk mengukur seberapa mudah dan efektif sistem digunakan oleh pengguna. Evaluasi ini akan mencakup:

1. Pengujian Kegunaan (Usability Testing): Melakukan pengujian kegunaan dengan melibatkan pengguna potensial (mahasiswa, HMJ, BEM) untuk mencoba aplikasi dan memberikan umpan balik tentang kemudahan penggunaan, navigasi, dan desain antarmuka. Umpan balik ini akan digunakan untuk memperbaiki desain UI/UX.
2. Survei Pengguna (*User* Survey): Mengumpulkan umpan balik dari pengguna melalui survei untuk mengukur tingkat kepuasan pengguna terhadap sistem secara keseluruhan. Survei dapat mencakup pertanyaan tentang kemudahan penggunaan, fungsionalitas, kinerja, dan keamanan sistem.

BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi *Backend*

Bagian ini menjelaskan secara komprehensif tentang penerapan sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis *blockchain*. Implementasi *backend* mencakup beberapa tahapan penting, mulai dari konfigurasi jaringan *blockchain* privat menggunakan *Hyperledger Besu*, memulai *node*, penambahan dan penghapusan *validator node*, hingga *validator* jaringan untuk memastikan operasi yang optimal. Selain itu, akan dijelaskan pula bagaimana *smart contract* yang dibangun menggunakan *Thirdweb* diintegrasikan ke dalam jaringan *blockchain* ini, serta bagaimana fungsi-fungsi utama dari *smart contract* tersebut diimplementasikan dan diuji. Implementasi ini bertujuan untuk menciptakan sistem yang aman, transparan, dan efisien dalam pengelolaan data akademik mahasiswa.

5.1.1 Implementasi *Private Blockchain Network* (*Hyperledger Besu*)

Implementasi jaringan *blockchain* privat menggunakan *Hyperledger Besu* melibatkan beberapa langkah penting untuk memastikan jaringan berjalan dengan baik, aman, dan dapat diandalkan. Langkah-langkah ini meliputi konfigurasi jaringan, memulai *node*, konfirmasi dan *monitoring* jaringan, menghentikan *node*, serta penambahan dan penghapusan *validator node*.

5.1.1.1 Konfigurasi Jaringan

1. Membuat File Konfigurasi Jaringan:

   File konfigurasi jaringan `qbftConfigFile.json` mendefinisikan berbagai parameter penting seperti *ID* rantai (chain *ID*), periode blok, panjang *epoch*, dan waktu tunggu permintaan. Berikut adalah isi dari file konfigurasi:

```
{
   "genesis": {
       "config": {
           "chainId": 1337,
           "berlinBlock": 0,
           "contractSizeLimit": 2147483647,
           "qbft": {
               "blockperiodseconds": 5,
               "epochlength": 30000,
               "requesttimeoutseconds": 10
           }
```

```
        },
        "nonce": "0x0",
        "timestamp": "0x58ee40ba",
        "gasLimit": "0x1fffffffffffff",
        "difficulty": "0x1",
        "mixHash":
"0x63746963616c2062797a616e74696e65206661756c7420746f6c6572
616e6365",
        "coinbase":
"0x0000000000000000000000000000000000000000",
        "alloc": {
            "1acA7a10b106CE0897a603148F0634838cc7AFcc": {
                "comment": "Super Admin",
                "balance": "900000000000000000000000"
            }
        }
    },
    "blockchain": {
        "nodes": {
            "generate": true,
            "count": 4
        }
    }
}
```

Tabel 5.1 Penjelasan script `qbftConfigFile.json`

Bagian	Penjelasan
`genesis`	Menyediakan konfigurasi untuk blok genesis, yaitu blok pertama dalam *blockchain*. Blok genesis menentukan parameter awal dan aturan yang akan diikuti oleh jaringan *blockchain*.
`config`	Berisi konfigurasi untuk jaringan *blockchain* yang mencakup berbagai parameter penting yang mengatur cara kerja jaringan.
`chainId`	*ID* unik untuk jaringan *blockchain*, dalam hal ini 1337. *Chain ID* ini digunakan untuk membedakan jaringan ini dari jaringan *blockchain* lain.
`berlinBlock`	Menunjukkan bahwa blok *Berlin* berlaku mulai dari blok pertama (blok 0). *Berlin* adalah *upgrade* protokol *Ethereum* yang mengoptimalkan biaya gas untuk beberapa operasi dan memperkenalkan beberapa perubahan lainnya.

Bagian	Penjelasan
`contractSizeLimit`	Batas ukuran kontrak dalam *byte*, yaitu 2147483647. Nilai ini dipilih untuk memastikan kontrak cerdas yang besar dapat disimpan dan dieksekusi dalam jaringan tanpa masalah. Angka ini merupakan batas maksimum yang memungkinkan untuk ukuran kontrak cerdas di jaringan ini.
`qbft`	Konfigurasi untuk konsensus *QBFT* (Quorum Byzantine Fault Tolerant). *QBFT* digunakan untuk mencapai konsensus di jaringan dengan toleransi kesalahan *Bizantium*, memastikan jaringan tetap berfungsi meskipun ada *node* yang berperilaku buruk atau mengalami kegagalan.
`blockperiodseconds`	Waktu dalam detik antara produksi blok, yaitu 5 detik. Periode blok yang pendek membantu dalam konfirmasi transaksi yang cepat.
`epochlength`	Panjang *epoch* dalam jumlah blok, yaitu 30000 blok. *Epoch length* menentukan seberapa sering proses rotasi *validator* terjadi, yang membantu dalam menjaga keamanan jaringan dengan memperbarui *validator* secara berkala.
`requesttimeoutseconds`	Waktu tunggu untuk permintaan dalam detik, yaitu 10 detik. Waktu tunggu ini mengatur seberapa lama *node* akan menunggu untuk menerima respons sebelum menganggap permintaan gagal.
`nonce`	Nomor yang digunakan sekali untuk blok genesis, yaitu 0x0. *Nonce* ini membantu dalam pembuatan *hash* unik untuk blok genesis.
`timestamp`	Waktu pembuatan blok genesis, yaitu 0x58ee40ba. *Timestamp* ini memastikan bahwa setiap blok genesis memiliki penanda waktu yang unik.
`gasLimit`	Batas maksimum gas yang bisa digunakan dalam satu blok, yaitu 0x1fffffffffffff. Gas *limit* ini menentukan berapa banyak komputasi yang dapat dilakukan dalam satu blok dan membantu mengatur biaya transaksi dalam jaringan.

Bagian	Penjelasan
`difficulty`	Tingkat kesulitan untuk menambang blok, yaitu 0x1. Meskipun dalam jaringan *QBFT* tingkat kesulitan ini tidak sepenting dalam jaringan *proof-of-work*, tetap ada untuk kesesuaian format.
`mixHash`	Nilai *hash* untuk membedakan blok genesis dari yang lain, yaitu 0x63746963616c2062797a616e74696e65206661756c7420746f6c6572616e6365. *MixHash* ini membantu dalam memastikan keunikan blok genesis.
`coinbase`	Alamat penerima hadiah dari penambangan blok, yaitu 0x0000000000000000000000000000000000000000. Dalam jaringan *QBFT*, alamat coinbase biasanya tidak digunakan karena tidak ada hadiah penambangan seperti dalam jaringan *proof-of-work*.
`alloc`	Alokasi awal akun dengan saldo dan kunci privat. Alokasi ini memungkinkan beberapa akun untuk memiliki saldo awal tanpa perlu menambang atau menerima transaksi dari luar.
1acA7a10b106C E0897a603148F 0634838cc7AFcc	Alamat akun yang dialokasikan dengan saldo awal. Alamat ini biasanya digunakan untuk akun yang memiliki peran khusus dalam jaringan.
`comment`	Komentar tentang akun, yaitu "*Super Admin*". Komentar ini memberikan konteks atau deskripsi tambahan tentang peran akun tersebut dalam jaringan.
`balance`	Saldo awal untuk akun dalam *wei*, yaitu 90000000000000000000000000. Saldo ini memungkinkan akun untuk melakukan transaksi dan operasi lain di jaringan tanpa perlu menerima dana dari luar.
`blockchain`	Konfigurasi untuk pembuatan *node blockchain*, yang menentukan bagaimana *node* akan diatur dan dihasilkan dalam jaringan.
`nodes`	Pengaturan untuk *node blockchain* yang akan dihasilkan.

Bagian	Penjelasan
`generate`	Mengindikasikan apakah *node* akan dihasilkan secara otomatis, dalam hal ini *true*. Pengaturan ini memungkinkan konfigurasi otomatis *node* untuk jaringan yang lebih mudah.
`count`	Jumlah *node* yang akan dihasilkan, dalam hal ini 4. Jumlah ini menentukan skala awal dari jaringan *blockchain* yang akan dibangun.

2. Script `generateNodeKeys.sh`

```
#!/bin/bash
besu operator generate-blockchain-config
--config-file=qbftConfigFile.json --to=networkFiles
--private-key-file-name=key
```

Tabel 5.2 Penjelasan script `generateNodeKeys.sh`

Penjelasan	Output

• Script ini digunakan untuk menghasilkan konfigurasi *blockchain* dan kunci *node* berdasarkan file qbftConfigFile.json.
• besu operator generate-*blockchain*-config: Perintah untuk menghasilkan konfigurasi *blockchain* dan kunci *node*.
• --config-file=qbftConfigFile.json: Menunjuk pada file konfigurasi yang berisi detail jaringan dan *node*.
• --to=networkFiles: Menentukan direktori keluaran tempat file konfigurasi *blockchain* dan kunci *node* akan
	• Direktori networkFiles/ akan dibuat, berisi:
	  ○ genesis.json: File genesis yang mendefinisikan blok awal *blockchain*.
	  ○ Direktori keys/ berisi pasangan kunci publik dan privat untuk setiap *node*.

Penjelasan	Output
disimpan. ● --*private*-key-file-name =key: Menentukan nama file kunci privat yang akan dihasilkan.	

3. Script `setupQBFT.sh`

```
#!/bin/bash


SOURCE_DIR="networkFiles"


cp "$SOURCE_DIR/genesis.json" .


NODE_COUNTER=1
for NODE_DIR in "$SOURCE_DIR/keys"/*; do
   if [ -d "$NODE_DIR" ]; then
       NODE_DEST_DIR="Node-${NODE_COUNTER}/data"
       mkdir -p "$NODE_DEST_DIR"
       cp "$NODE_DIR/key" "$NODE_DEST_DIR/"
       cp "$NODE_DIR/key.pub" "$NODE_DEST_DIR/"
       NODE_COUNTER=$((NODE_COUNTER + 1))
   fi
done


echo "Setup completed successfully."
```

Tabel 5.3 Penjelasan script `setupQBFT.sh`

Penjelasan	Output
● Script ini digunakan untuk menyalin file genesis.json dan kunci *node* dari direktori networkFiles ke direktori masing-masing *node*. ● cp "$SOURCE_DIR/genesis.json" .: Menyalin file genesis.json ke direktori root.	● Direktori *Node*-1, *Node*-2, *Node*-3, dan *Node*-4 akan dibuat, masing-masing berisi direktori data yang berisi file kunci (key dan key.pub).

Penjelasan	Output
• Loop melalui setiap direktori *node* di networkFiles/keys dan menyalin kunci ke direktori data yang sesuai untuk setiap *node*. • NODE_COUNTER: Menghitung nomor *node* untuk membuat direktori yang sesuai (*Node*-1, *Node*-2, *Node*-3, *Node*-4).	

5.1.1.2 Memulai *Node*

1. Script `startAllNodes.sh`

```
#!/bin/bash


LOG_DIR="logs"
BOOTNODES_FILE="bootnodes.txt"
NODE_ADDRESS_DIR="NodeAddresses"


mkdir -p "${LOG_DIR}"
mkdir -p "${NODE_ADDRESS_DIR}"


start_node() {
   local NODE_DIR=$1
   local NODE_NUMBER=$2
   local P2P_PORT=$((30303 + NODE_NUMBER - 1))
   local RPC_HTTP_PORT=$((8545 + NODE_NUMBER - 1))


   NODE_PID=$(ps -ef | grep 'besu' | grep "$NODE_DIR" |
grep -v 'grep' | awk '{print $2}')
   if [ -n "$NODE_PID" ]; then
       echo "Node $NODE_DIR is already running with PID
$NODE_PID."
       return
   fi


   echo "Starting $NODE_DIR..."
   if [ "$NODE_NUMBER" -eq 1 ]; then
       nohup besu --data-path="${NODE_DIR}/data" \
           --genesis-file="${NODE_DIR}/../genesis.json" \
```

```
           --p2p-port=${P2P_PORT} \
           --rpc-http-enabled \

--rpc-http-api=EEA,WEB3,ETH,NET,TRACE,DEBUG,ADMIN,TXPOOL,PE
RM,QBFT \
           --host-allowlist="*" \
           --rpc-http-port=${RPC_HTTP_PORT} \
           --rpc-http-cors-origins="*" \
           --min-gas-price=0 >
"${LOG_DIR}/node${NODE_NUMBER}.log" 2>&1 &
   else
       nohup besu --data-path="${NODE_DIR}/data" \
           --genesis-file="${NODE_DIR}/../genesis.json" \
           --bootnodes="$(cat $BOOTNODES_FILE)" \
           --p2p-port=${P2P_PORT} \
           --rpc-http-enabled \

--rpc-http-api=EEA,WEB3,ETH,NET,TRACE,DEBUG,ADMIN,TXPOOL,PE
RM,QBFT \
           --host-allowlist="*" \
           --rpc-http-port=${RPC_HTTP_PORT} \
           --rpc-http-cors-origins="*" \
           --min-gas-price=0 >
"${LOG_DIR}/node${NODE_NUMBER}.log" 2>&1 &
   fi


   sleep 10


   NODE_ADDRESS=$(grep -o "Node address [^ ]*"
"${LOG_DIR}/node${NODE_NUMBER}.log" | awk '{print $3}')
   echo "$NODE_ADDRESS" >
"${NODE_ADDRESS_DIR}/${NODE_DIR}.address"


   echo "$NODE_DIR started successfully."
}


start_node "Node-1" 1


ENODE_URL=$(grep -o "enode://[^@]*@[^:]*@[0-9]*"
"${LOG_DIR}/node1.log")


if [ -z "$ENODE_URL" ]; then
   echo "Failed to capture Node-1 enode URL."
   exit 1
fi


echo "Node-1 started with enode URL: $ENODE_URL"
echo "$ENODE_URL" > "$BOOTNODES_FILE"


NODE_COUNTER=2
```

```
for NODE_DIR in Node-*; do
   if [ "$NODE_DIR" != "Node-1" ] && [ -d "$NODE_DIR" ];
then
       start_node "$NODE_DIR" "$NODE_COUNTER"
       NODE_COUNTER=$((NODE_COUNTER + 1))
   fi
done


echo "All nodes started successfully."
```

Tabel 5.4 Penjelasan script `startAllNodes.sh`

Penjelasan	Output
• Script ini digunakan untuk memulai semua *node* dalam jaringan *blockchain*. *Node* pertama akan berfungsi sebagai *bootnode*. • nohup besu ... & menjalankan setiap *node* dalam mode background dan menyimpan lognya. • sleep 10 memberikan jeda waktu agar *node* bisa mulai dengan benar. • grep -o "*enode*://[^@]*@[^:]*:[0-9]*" menangkap *URL enode* dari log *node* pertama untuk digunakan sebagai *bootnode* untuk *node* lainnya. • Setiap *node* akan disimpan lognya di direktori logs dan alamat nodenya disimpan di direktori NodeAddresses.	• Semua *node* akan berjalan sebagai *validator* dalam jaringan *blockchain*. • File log untuk setiap *node* akan dibuat di direktori logs. • Alamat *node* disimpan di direktori NodeAddresses.

2. Script `startNodes.sh`

```
#!/bin/bash


if [ -z "$1" ]; then
  echo "Usage: $0 <Node-X>"
  exit 1
fi


NODE_DIR="$1"
GENESIS_FILE="genesis.json"
LOG_DIR="logs"
BOOTNODES_FILE="bootnodes.txt"
NODE_ADDRESS_DIR="NodeAddresses"


if [ ! -d "$NODE_DIR" ];then
  echo "Node directory $NODE_DIR does not exist. Add
Validator first!"
  exit 1
fi


NODE_NUMBER=$(echo $NODE_DIR | grep -o '[0-9]*')
P2P_PORT=$((30303 + NODE_NUMBER - 1))
RPC_HTTP_PORT=$((8545 + NODE_NUMBER - 1))


NODE_PID=$(ps -ef | grep 'besu' | grep "$NODE_DIR" | grep
-v 'grep' | awk '{print $2}')
if [ -n "$NODE_PID" ];then
   echo "Node $NODE_DIR is already running with PID
$NODE_PID."
   exit 1
fi


echo "Starting $NODE_DIR..."
nohup besu --data-path="${NODE_DIR}/data" \
   --genesis-file="${GENESIS_FILE}" \
   --bootnodes="$(cat $BOOTNODES_FILE)" \
   --p2p-port=${P2P_PORT} \
   --rpc-http-enabled \

--rpc-http-api=EEA,WEB3,ETH,NET,TRACE,DEBUG,ADMIN,TXPOOL,PE
RM,QBFT \
   --host-allowlist="*" \
   --rpc-http-port=${RPC_HTTP_PORT} \
   --rpc-http-cors-origins="*" \
   --min-gas-price=0 > "${LOG_DIR}/node${NODE_NUMBER}.log"
2>&1 &


sleep 10


NODE_ADDRESS=$(grep -o "Node address [^ ]*"
"${LOG_DIR}/node${NODE_NUMBER}.log" | awk '{print $3}')
```

```
echo "$NODE_ADDRESS" >
"${NODE_ADDRESS_DIR}/${NODE_DIR}.address"


echo "$NODE_DIR started successfully."
```

Tabel 5.5 Penjelasan script `startNodes.sh`

Penjelasan	Output
• Script ini digunakan untuk memulai satu *node* tertentu dalam jaringan *blockchain*. • nohup besu ... & menjalankan *node* dalam mode background dan menyimpan lognya. • sleep 10 memberikan jeda waktu agar *node* bisa mulai dengan benar. • grep -o "*Node* address [^]*" menangkap alamat *node* dari log dan menyimpannya ke dalam direktori NodeAddresses.	• *Node* yang ditentukan akan berjalan sebagai *validator* dalam jaringan *blockchain*. • File log untuk *node* tersebut akan dibuat di direktori logs. • Alamat *node* disimpan di direktori NodeAddresses.

5.1.1.3 Konfirmasi dan Monitoring Jaringan

1. Script `confirmAllNetwork.sh`

```
#!/bin/bash


LOG_DIR="logs"
NODE_ADDRESS_DIR="NodeAddresses"


confirm_network() {
   local RPC_URL=$1
   local NODE_NAME=$2



PAYLOAD='{"jsonrpc":"2.0","method":"qbft_getValidatorsByBlo
ckNumber","params":["latest"], "id":1}'
```

```
   RESPONSE=$(curl -s -X POST --data "$PAYLOAD" $RPC_URL |
jq)


   echo "Response from JSON-RPC API for $NODE_NAME
($RPC_URL):"
   echo "$RESPONSE"


   VALIDATORS=$(echo "$RESPONSE" | jq '.result | length')


   if [ -z "$VALIDATORS" ]; then
       VALIDATORS="stopped"
   fi


   echo "Number of validators for $NODE_NAME: $VALIDATORS"


   if [ "$VALIDATORS" != "stopped" ] && [ "$VALIDATORS" -ge
4 ]; then
       echo "The private network is working correctly with
at least four validators for $NODE_NAME."
   else
       echo "The private network does not have four
validators for $NODE_NAME."
   fi


   echo "validators[$NODE_NAME]=$VALIDATORS" >>
$LOG_DIR/validator_count.log
   echo ""
}


NODE_COUNTER=1
> $LOG_DIR/validator_count.log
for NODE_DIR in Node-*; do
   if [ -d "$NODE_DIR" ]; then
       RPC_HTTP_PORT=$((8545 + NODE_COUNTER - 1))
       RPC_URL="http://localhost:${RPC_HTTP_PORT}"
       confirm_network "$RPC_URL" "$NODE_DIR"
       NODE_COUNTER=$((NODE_COUNTER + 1))
   fi
done


echo "Network confirmation completed for all nodes."


echo "Summary of validators count for each node:"
cat $LOG_DIR/validator_count.log | while read line; do
   echo "$line"
done
```

Tabel 5.6 Penjelasan script `confirmAllNetwork.sh`

Penjelasan	Output
• Script ini digunakan untuk mengkonfirmasi status semua *node* dalam jaringan *blockchain*. • curl digunakan untuk mengirim permintaan *JSON-RPC* ke setiap *node*. • jq digunakan untuk memparse dan menampilkan response *JSON*. • Script ini memeriksa apakah setiap *node* memiliki setidaknya empat *validator* dan menyimpan hasilnya ke dalam file validator_count.log.	• Status jaringan untuk setiap *node* akan ditampilkan. • Informasi jumlah *validator* untuk setiap *node* akan disimpan di file validator_count.log. • Summary dari jumlah *validator* untuk setiap *node* akan ditampilkan di akhir.

2. Script `confirmNetwork.sh`

```
#!/bin/bash


if [ -z "$1" ]; then
   echo "Usage: $0 <Node-X>"
   exit 1
fi


NODE_DIR="$1"
NODE_NUMBER=$(echo $NODE_DIR | grep -o '[0-9]*')
RPC_HTTP_PORT=$((8545 + NODE_NUMBER - 1))
RPC_URL="http://localhost:${RPC_HTTP_PORT}"


PAYLOAD='{"jsonrpc":"2.0","method":"qbft_getValidatorsByBlockNumb
er","params":["latest"], "id":1}'


RESPONSE=$(curl -s -X POST --data "$PAYLOAD" $RPC_URL | jq)


echo "Response from JSON-RPC API for $NODE_DIR ($RPC_URL):"
echo "$RESPONSE"
```

```
VALIDATORS=$(echo "$RESPONSE" | jq '.result | length')


if [ -z "$VALIDATORS" ]; then
   VALIDATORS="stopped"
fi


echo "Number of validators for $NODE_DIR: $VALIDATORS"


if [ "$VALIDATORS" != "stopped" ] && [ "$VALIDATORS" -ge 4 ];
then
   echo "The private network is working correctly with at least
four validators for $NODE_DIR."
else
   echo "The private network does not have four validators for
$NODE_DIR."
fi
```

Tabel 5.7 Penjelasan script `confirmNetwork.sh`

Penjelasan	Output
• Script ini digunakan untuk mengkonfirmasi status satu *node* tertentu dalam jaringan *blockchain*. • curl digunakan untuk mengirim permintaan *JSON-RPC* ke *node* yang ditentukan. • jq digunakan untuk memparse dan menampilkan response *JSON*. • Script ini memeriksa apakah *node* yang ditentukan memiliki setidaknya empat *validator*.	• Status jaringan untuk *node* yang ditentukan akan ditampilkan. • Informasi jumlah *validator* untuk *node* yang ditentukan akan ditampilkan.

3. Quorum Explorer: Monitoring Jaringan Secara Visual

Quorum Explorer digunakan untuk *monitoring* jaringan secara visual. Langkah-langkah untuk menjalankan Quorum Explorer:

a. Clone repository Quorum Explorer

```
git clone https://github.com/Consensys/quorum-explorer.git
```

b. Konfigurasi config.json

```
{
 "algorithm": "qbft",
 "nodes": [
   {
     "name": "rpcnode",
     "client": "besu",
     "rpcUrl": "http://localhost:8545",
     "privateTxUrl": ""
   },
   {
     "name": "validator1",
     "client": "besu",
     "rpcUrl": "http://localhost:8546",
     "privateTxUrl": ""
   },
   {
     "name": "validator2",
     "client": "besu",
     "rpcUrl": "http://localhost:8547",
     "privateTxUrl": ""
   },
   {
     "name": "validator3",
     "client": "besu",
     "rpcUrl": "http://localhost:8548",
     "privateTxUrl": ""
   }
 ]
}
```

c. Buat file .env.local di root direktori proyek Quorum Explorer

```
QE_BASEPATH="/explorer"
QE_CONFIG_PATH="src/config/config.json"
NODE_ENV=development
DISABLE_AUTH=true
NEXTAUTH_URL=http://localhost:25000
NEXTAUTH_URL_INTERNAL=http://localhost:25000
```

d. Instal dependensi dan jalankan Quorum Explorer

```
npm install
npm run dev
```

Tabel 5.8 Penjelasan quorum-explorer

Penjelasan	Output
• Quorum Explorer adalah alat *monitoring* visual yang memungkinkan pengguna untuk melihat status jaringan, *node*, dan transaksi dalam waktu nyata. • File konfigurasi config.json menentukan algoritma konsensus (*QBFT*) dan daftar *node* yang akan dimonitor dalam jaringan. • File .env.local mengatur variabel lingkungan yang diperlukan untuk menjalankan Quorum Explorer dalam mode pengembangan.	• Quorum Explorer akan berjalan di http://localhost:25000. • Pengguna dapat memonitor status jaringan, *node*, dan transaksi secara visual melalui antarmuka Quorum Explorer.

5.1.1.4 Menghentikan *Node*

1. Script `stopAllNodes.sh`

```
#!/bin/bash


echo "Stopping all besu nodes..."


PIDS=$(ps -ef | grep 'besu' | grep -v 'grep' | awk '{print
$2}')


if [ -z "$PIDS" ]; then
   echo "No besu nodes are running."
else
   for PID in $PIDS; do
       echo "Stopping besu node with PID $PID"
       kill -9 $PID
   done
```

```
   echo "All besu nodes stopped successfully."
fi
```

Tabel 5.9 Penjelasan script `stopAllNodes.sh`

Penjelasan	Output
• Script ini digunakan untuk menghentikan semua *node* besu yang sedang berjalan. • ps -ef | grep 'besu' | grep -v 'grep' | awk '{print $2}' mendapatkan daftar process *ID* (PID) dari semua proses besu yang berjalan. • kill -9 $PID menghentikan setiap proses besu berdasarkan PID yang ditemukan.	• Semua *node* besu akan dihentikan. • Menampilkan pesan bahwa semua *node* besu telah dihentikan dengan sukses.

2. Script `stopNodes.sh`

```
#!/bin/bash


if [ -z "$1" ]; then
   echo "Usage: $0 <Node-X>"
   exit 1
fi


NODE_DIR="$1"


# Mendapatkan PID dari node yang berjalan berdasarkan
direktori data
NODE_PID=$(ps -ef | grep 'besu' | grep "$NODE_DIR" | grep
-v 'grep' | awk '{print $2}')


if [ -z "$NODE_PID" ]; then
   echo "Node $NODE_DIR is not running."
else
   echo "Stopping Besu node with PID $NODE_PID from
directory $NODE_DIR..."
   kill -9 $NODE_PID
   echo "Node $NODE_DIR stopped successfully."
fi
```

Tabel 5.10 Penjelasan script `stopAllNodes.sh`

Penjelasan	Output
• Script ini digunakan untuk menghentikan *node* tertentu dalam jaringan *blockchain*. • Script akan mencari PID (Process *ID*) dari proses *Besu* yang berjalan berdasarkan direktori *node* yang diberikan. • Jika *node* berjalan, maka script akan menghentikan proses tersebut menggunakan perintah kill -9.	• Status apakah *node* sedang berjalan atau tidak akan ditampilkan. • Jika *node* sedang berjalan, maka *node* tersebut akan dihentikan dan pesan konfirmasi akan ditampilkan.

5.1.1.5 Penambahan dan Penghapusan *Validator Node*

1. Script `addValidator.sh`

```
#!/bin/bash


LOG_DIR="logs"
BOOTNODES_FILE="bootnodes.txt"
GENESIS_FILE="genesis.json"
NODE_ADDRESS_DIR="NodeAddresses"


mkdir -p "${LOG_DIR}"


find_available_port() {
 local PORT=$1
 while lsof -i :${PORT} >/dev/null; do
   PORT=$((PORT + 1))
 done
 echo ${PORT}
}


NODE_COUNTER=1
while [ -d "Node-${NODE_COUNTER}" ]; do
   NODE_COUNTER=$((NODE_COUNTER + 1))
done


NODE_DIR="Node-${NODE_COUNTER}"
```

```
P2P_PORT=$(find_available_port $((30303 + $((NODE_COUNTER -
1)))))
RPC_HTTP_PORT=$(find_available_port $((8545 +
$((NODE_COUNTER - 1)))))


mkdir -p "${NODE_DIR}/data"


echo "Starting new validator node (Node-${NODE_COUNTER})
with P2P port ${P2P_PORT} and RPC HTTP port
${RPC_HTTP_PORT}..."
nohup besu --data-path="${NODE_DIR}/data" \
    --genesis-file="${GENESIS_FILE}" \
    --bootnodes="$(head -n 1 $BOOTNODES_FILE)" \
    --p2p-port=${P2P_PORT} \
    --rpc-http-enabled \

--rpc-http-api=EEA,WEB3,ETH,NET,TRACE,DEBUG,ADMIN,TXPOOL,PE
RM,QBFT \
    --host-allowlist="*" \
    --rpc-http-host="0.0.0.0" \
    --rpc-http-cors-origins="*" \
    --rpc-http-port=${RPC_HTTP_PORT} >
"${LOG_DIR}/node${NODE_COUNTER}.log" 2>&1 &


sleep 10


ENODE_URL=$(grep -m 1 -o "enode://[^@]*@[^:]*:[0-9]*"
"${LOG_DIR}/node${NODE_COUNTER}.log")


if [ -z "$ENODE_URL" ]; then
   echo "Failed to capture enode URL for the new validator
node."
   echo "Log content for debugging:"
   cat "${LOG_DIR}/node${NODE_COUNTER}.log"
   exit 1
fi


echo "New validator node started with enode URL:
$ENODE_URL"


echo "$ENODE_URL" >> "$BOOTNODES_FILE"


NODE_ADDRESS=$(grep -m 1 -o "Node address [^ ]*"
"${LOG_DIR}/node${NODE_COUNTER}.log" | awk '{print $3}')


echo "$NODE_ADDRESS" >
"${NODE_ADDRESS_DIR}/Node-${NODE_COUNTER}.address"
```

```
if [ -z "$NODE_ADDRESS" ]; then
   echo "Node address not found in the log file. Here is
the log content:"
   cat "${LOG_DIR}/node${NODE_COUNTER}.log"
   echo "Failed to capture node address for the new
validator node."
   exit 1
fi


echo "New validator node address: $NODE_ADDRESS"


EXISTING_NODE_COUNT=$((NODE_COUNTER - 1))
MAJORITY_COUNT=$(( (EXISTING_NODE_COUNT / 2) + 1 ))


echo "Proposing new validator from ${MAJORITY_COUNT}
nodes..."
VALIDATOR_VOTE_COUNT=0
for i in $(seq 1 $EXISTING_NODE_COUNT); do
   if [ $VALIDATOR_VOTE_COUNT -ge $MAJORITY_COUNT ]; then
       break
   fi
   RPC_PORT=$((8544 + i))
   echo "Proposing new validator from node with RPC port
${RPC_PORT}..."
   RESPONSE=$(curl -s -m 10 -X POST --data
'{"jsonrpc":"2.0","method":"qbft_proposeValidatorVote","par
ams":["'"${NODE_ADDRESS}"'", true], "id":1}'
http://127.0.0.1:${RPC_PORT})
   if echo "$RESPONSE" | grep -q '"result":true'; then
       VALIDATOR_VOTE_COUNT=$((VALIDATOR_VOTE_COUNT + 1))
       echo "Validator vote succeeded from node with RPC
port ${RPC_PORT}."
   else
       echo "Validator vote failed from node with RPC port
${RPC_PORT}. Response: $RESPONSE"
   fi
   echo ""
done


if [ $VALIDATOR_VOTE_COUNT -ge $MAJORITY_COUNT ]; then
   echo "New validator node added successfully
(Node-${NODE_COUNTER})."
else
   echo "Failed to get majority vote for the new validator
node (Node-${NODE_COUNTER})."
   exit 1
fi
```

Tabel 5.11 Penjelasan script `addValidator.sh`

Penjelasan	Output
• Script ini digunakan untuk menambahkan *node* baru sebagai *validator* ke jaringan *blockchain*. • find_available_port menemukan port yang tersedia untuk P2P dan *RPC*. • nohup besu ... & memulai *node* baru dan menyimpan lognya. • grep -m 1 -o "*enode*://[^@]*@[^:]*:[0-9]*" menangkap *URL enode* dari log *node* baru. • curl -s -m 10 -X POST ... mengusulkan penambahan *validator* baru dari *node* yang sudah ada.	• *Node* baru akan dimulai sebagai *validator*. • *URL enode* dan alamat *node* disimpan di file yang sesuai. • Usulan penambahan *validator* baru diajukan dan diterima oleh mayoritas *node* yang ada.

2. Script `removeValidator.sh`

```
#!/bin/bash


if [ -z "$1" ]; then
   echo "Usage: $0 <validator-address>"
   exit 1
fi


VALIDATOR_ADDRESS=$1
LOG_DIR="logs"
NODE_ADDRESS_DIR="NodeAddresses"


mkdir -p "${LOG_DIR}"
```

```
NODE_DIR=""
for FILE in ${NODE_ADDRESS_DIR}/*.address; do
   if grep -q "${VALIDATOR_ADDRESS}" "$FILE"; then
       NODE_DIR=$(basename "$FILE" .address)
       break
   fi
done


if [ -z "$NODE_DIR" ]; then
   echo "No node directory found for validator address
${VALIDATOR_ADDRESS}."
   exit 1
fi


EXISTING_NODE_COUNT=$(ls -d Node-* | wc -l)
MAJORITY_COUNT=$(( (EXISTING_NODE_COUNT / 2) + 1 ))


echo "Discarding any pending votes for validator
${VALIDATOR_ADDRESS} from ${MAJORITY_COUNT} nodes..."
for i in $(seq 1 $EXISTING_NODE_COUNT); do
   RPC_PORT=$((8544 + i))
   echo "Discarding vote from node with RPC port
${RPC_PORT}..."
   RESPONSE=$(curl -s -m 10 -X POST --data
'{"jsonrpc":"2.0","method":"qbft_discardValidatorVote","par
ams":["'"${VALIDATOR_ADDRESS}"'"], "id":1}'
http://127.0.0.1:${RPC_PORT})
   if echo "$RESPONSE" | grep -q '"result":true'; then
       echo "Vote discarded from node with RPC port
${RPC_PORT}."
   else
       echo "Failed to discard vote from node with RPC port
${RPC_PORT}. Response: $RESPONSE"
   fi
   echo ""
done


echo "Proposing to remove validator ${VALIDATOR_ADDRESS}
from ${MAJORITY_COUNT} nodes..."
VALIDATOR_VOTE_COUNT=0
for i in $(seq 1 $EXISTING_NODE_COUNT); do
   if [ $VALIDATOR_VOTE_COUNT -ge $MAJORITY_COUNT ]; then
       break
   fi
   RPC_PORT=$((8544 + i))
   echo "Proposing to remove validator from node with RPC
port ${RPC_PORT}..."
   RESPONSE=$(curl -s -m 10 -X POST --data
'{"jsonrpc":"2.0","method":"qbft_proposeValidatorVote","par
ams":["'"${VALIDATOR_ADDRESS}"'", false], "id":1}'
http://127.0.0.1:${RPC_PORT})
   if echo "$RESPONSE" | grep -q '"result":true'; then
       VALIDATOR_VOTE_COUNT=$((VALIDATOR_VOTE_COUNT + 1))
```

```
        echo "Validator removal vote succeeded from node
with RPC port ${RPC_PORT}."
    else
        echo "Validator removal vote failed from node with
RPC port ${RPC_PORT}. Response: $RESPONSE"
    fi
    echo ""
done


if [ $VALIDATOR_VOTE_COUNT -ge $MAJORITY_COUNT ]; then
    echo "Validator ${VALIDATOR_ADDRESS} removed
successfully."
    pkill -f "${NODE_DIR}/data" || true


    NODE_PATH="${NODE_DIR}"
    if [ -d "$NODE_PATH"]; then
        rm -rf "$NODE_PATH"
        echo "Node directory ${NODE_PATH} removed
successfully."
    else
        echo "Node directory ${NODE_PATH} not found."
    fi
    ADDRESS_FILE="${NODE_ADDRESS_DIR}/${NODE_DIR}.address"
    if [ -f "$ADDRESS_FILE"]; then
        rm "$ADDRESS_FILE"
        echo "Node address file ${ADDRESS_FILE} removed
successfully."
    else
        echo "Node address file ${ADDRESS_FILE} not found."
    fi
    LOG_FILE="${LOG_DIR}/node${NODE_DIR##*-}.log"
    if [ -f "$LOG_FILE"]; then
        rm "$LOG_FILE"
        echo "Log file ${LOG_FILE} removed successfully."
    else
        echo "Log file ${LOG_FILE} not found."
    fi
else
    echo "Failed to get majority vote for removing validator
${VALIDATOR_ADDRESS}."
    exit 1
fi
```

Tabel 5.12 Penjelasan script `removeValidator.sh`

Penjelasan	Output
• Script ini digunakan untuk menghapus *validator* dari jaringan *blockchain*. • grep -q	• *Validator* dihapus dari jaringan *blockchain*. • Direktori *node*, file alamat, dan file log dihapus.

Penjelasan	Output
"${VALIDATOR_ADDRESS}" "$FILE" menemukan direktori *node* yang sesuai dengan alamat *validator*. ● curl -s -m 10 -X POST ... mengusulkan penghapusan *validator* dari *node* yang sudah ada. ● pkill -f "${NODE_DIR}/data" || *true* memastikan tidak ada proses background yang berjalan untuk *node* tersebut. ● Direktori *node*, file alamat, dan file log dihapus setelah *validator* dihapus dari jaringan.	

5.1.1.6 Menambahkan *Network* di *Metamask*

Untuk menghubungkan *Metamask* dengan jaringan *blockchain* lokal yang telah dibangun menggunakan *Hyperledger Besu*, diperlukan penambahan network secara manual di *Metamask*. Berikut adalah langkah-langkah lengkap untuk menambahkan *network* tersebut:

1. Menambahkan *Network* di *Metamask*
    a. Install *Metamask* (Jika Belum Terpasang):
        - Kunjungi situs resmi *Metamask* di https://metamask.io/ dan tambahkan ekstensi *Metamask* ke browser.
        - Setelah terpasang, buka ekstensi dan buat *wallet* baru atau impor *wallet* yang sudah ada dengan menggunakan *seed phrase*.
    b. Buka *Metamask*:

- Klik ikon *Metamask* di browser untuk membuka ekstensi *Metamask*.

c. Pilih *Network*:
   - Klik dropdown *network* yang berada di bagian atas *Metamask* (biasanya bertuliskan "*Ethereum* Mainnet" atau nama jaringan lainnya).

d. Tambahkan *Network* Baru:
   - *Scroll* ke bawah dan klik pada "Add *Network*" atau "*Custom RPC*".

e. Isi Detail *Network*:
   - *Network* Name: Masukkan nama jaringan, misalnya "LedgerKu".
   - *New RPC URL*: Masukkan *URL RPC* dari *node* pertama, misalnya http://localhost:8545.
   - *Chain ID*: Masukkan *Chain ID* yang digunakan oleh jaringan, misalnya 1337 untuk jaringan lokal.
   - *Currency Symbol*: Memasukkan simbol mata uang, misalnya "*ETH*".
   - *Block Explorer URL*: Kosongkan atau isi dengan *URL block explorer* jika ada misalnya “http://localhost:25000/explorer/nodes”.

f. Simpan *Network*:
   - Klik "Save" untuk menyimpan konfigurasi network.

g. Verifikasi *Network*:
   - Setelah *network* ditambahkan, pastikan *Metamask* terhubung ke *network* yang baru saja ditambahkan dengan melihat nama *network* di bagian atas *Metamask*.

Penambahan *network* ini memungkinkan wallet *Metamask* untuk berinteraksi dengan jaringan *blockchain* lokal yang telah dibangun, sehingga dapat digunakan untuk keperluan pengembangan dan pengujian *smart contract* secara lokal.

Gambar 5.1 Alur Implementasi *Private Blockchain Network* (*Hyperledger Besu*)

5.1.2 Implementasi *Smart Contract*

Bagian ini akan membahas implementasi *smart contract*, yang menjadi komponen penting dalam aplikasi ini. Pembahasan akan mencakup proses pembuatan *smart contract*, penjelasan *smart contract*, hingga deploy *smart contract*.

5.1.2.1 Pembuatan *Smart Contract*

Langkah pertama dalam implementasi *smart contract* adalah membuat contract menggunakan *Thirdweb*. *Thirdweb CLI* digunakan untuk membuat *smart contract* baru dengan langkah-langkah sebagai berikut:

1. Instalasi *Thirdweb CLI*:

   Pastikan bahwa *Thirdweb CLI* telah terinstal di sistem. Jika belum, instalasi dilakukan dengan perintah berikut:

   ```
   npm install -g @thirdweb-dev/cli
   ```

2. Membuat Contract Baru:

   Gunakan *Thirdweb CLI* untuk membuat *smart contract* baru dengan perintah berikut:

   ```
   npx thirdweb create --contract
   ```

   Ikuti instruksi di *CLI* untuk memberikan nama contract dan memilih template yang sesuai. Setelah selesai, sebuah project baru dengan struktur dasar contract akan dibuat. Langkah-langkah detail adalah:

   a. *Project Name* : Berikan nama folder yang sesuai, misalnya “web3”.
   b. *Framework* : Memilih framework yang ingin digunakan misalnya “Hardhat”.
   c. Nama *Contract*: Berikan nama yang sesuai, misalnya "SKKM".
   d. *Template*: Pilih template yang paling sesuai dengan kebutuhan, misalnya "Empty Contract".
   e. Konfigurasi: Ikuti petunjuk untuk mengkonfigurasi *contract*, seperti menetapkan variabel-variabel awal yang diperlukan.

Smart contract ini, bernama "SKKM.sol", menjadi inti dari sistem pencatatan SKKM berbasis *blockchain*. *Smart contract* ini mengatur semua

transaksi dan interaksi yang terjadi dalam sistem. Berikut adalah kode lengkap dari *smart contract* tersebut:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;


contract SKKM {
   address public superAdmin;
   uint public minimalPoin;


   enum Role {
       None,
       Student,
       HMJ,
       Admin,
       BEM
   }


   enum Status {
       Pending,
       Verified,
       Unverified,
       Valid,
       Revised,
       Generated,
       SendToBEM,
       ValidatedByBEM
   }


   struct SKKMRequest {
       uint skkmId;
       address student;
       string activityNameID;
       string activityNameEN;
       string certificateNumber;
       string activityCategory;
       string activityType;
       string achievement;
       string assessmentBasis;
       string ipfsHash;
       uint creditPoin;
       Status status;
       string unverifiedMessage;
       uint timestamp;
   }


   struct User {
       uint userId;
       address userAddress;
       string name;
       string identifier;
       string department;
```

```
    Role role;
    bool exists;
}


struct QRCode {
    uint qrCodeId;
    address student;
    string qrCodeData;
    Status status;
    uint timestamp;
}


mapping(address => User) public users;
mapping(string => bool) private identifiers;
mapping(address => mapping(string => bool))
    private studentCertificateNumbers;
SKKMRequest[] public requests;
User[] public allUsers;
QRCode[] public qrcodes;
mapping(address => bool) public hasGeneratedQRCode;
mapping(address => uint) public lastQRCodeTimestamp;


event SKKMSubmitted(
    uint requestId,
    address indexed student,
    string ipfsHash,
    string certificateNumber,
    address triggeredBy
);


event SKKMVerified(
    uint requestId,
    address indexed verifier,
    bool isVerified,
    string message,
    address triggeredBy
);


event SKKMValidated(
    uint requestId,
    address indexed validator,
    bool isValid,
    address triggeredBy
);


event SKKMEdited(
    uint requestId,
    address indexed student,
    string ipfsHash,
    string certificateNumber,
    address triggeredBy
);
```

```
    event SKKMDeleted(
        uint requestId,
        address indexed student,
        address triggeredBy
    );
    event UserDeleted(address indexed user, address triggeredBy);
    event MinimalPoinUpdated(uint newMinimalPoin, address
triggeredBy);
    event QRCodeGenerated(
        uint qrCodeId,
        address indexed student,
        string qrCodeData,
        uint timestamp,
        address triggeredBy
    );
    event QRCodeSubmitted(
        uint qrCodeId,
        address indexed student,
        string qrCodeData,
        uint timestamp,
        address triggeredBy
    );


    event UserAdded(
        uint userId,
        address indexed user,
        string name,
        string identifier,
        string department,
        Role role,
        address triggeredBy
    );


    event QRCodeValidatedByBEM(
        uint qrCodeId,
        address indexed validator,
        address indexed student,
        address triggeredBy
    );


    modifier onlySuperAdmin() {
        require(
            msg.sender == superAdmin,
            "Only super admin can perform this action"
        );
        _;
    }


    modifier onlyAdmin() {
        require(
            users[msg.sender].role == Role.Admin || msg.sender ==
superAdmin,
```

```
            "Only admin can perform this action"
        );
        _;
    }


    modifier onlyStudent() {
        require(
            users[msg.sender].role == Role.Student,
            "Only students can perform this action"
        );
        _;
    }


    modifier onlyHMJ() {
        require(
            users[msg.sender].role == Role.HMJ,
            "Only HMJ can perform this action"
        );
        _;
    }


    modifier onlyBEM() {
        require(
            users[msg.sender].role == Role.BEM,
            "Only BEM can perform this action"
        );
        _;
    }


    constructor() {
        superAdmin = msg.sender;
    }


    function addUser(
        address user,
        Role role,
        string memory name,
        string memory identifier,
        string memory department
    ) external payable {
        require(
            msg.sender == superAdmin ||
                (role != Role.Admin && users[msg.sender].role ==
Role.Admin),
            "Only admin or super admin can perform this action"
        );
        require(!users[user].exists, "User already exists");
        require(!identifiers[identifier], "Identifier already
exists");


        User memory newUser = User({
            userId: allUsers.length,
```

```
            userAddress: user,
            name: name,
            identifier: identifier,
            department: department,
            role: role,
            exists: true
        });


        users[user] = newUser;
        allUsers.push(newUser);


        identifiers[identifier] = true;


        if (msg.value > 0) {
            payable(user).transfer(msg.value);
        }


        emit UserAdded(
            newUser.userId,
            user,
            name,
            identifier,
            department,
            role,
            msg.sender
        );
    }


    function updateUser(
        address user,
        Role role,
        string memory name,
        string memory identifier,
        string memory department
    ) external onlyAdmin {
        require(users[user].exists, "User does not exist");
        require(
            !identifiers[identifier] ||
                keccak256(bytes(identifier)) ==
                keccak256(bytes(users[user].identifier)),
            "Identifier already exists"
        );


        User storage existingUser = users[user];
        existingUser.name = name;
        existingUser.identifier = identifier;
        existingUser.department = department;
        existingUser.role = role;


        allUsers[existingUser.userId] = existingUser;
```

```
        identifiers[identifier] = true;
    }


    function deleteUser(address user) external onlyAdmin {
        require(users[user].exists, "User does not exist");


        delete identifiers[users[user].identifier];
        uint userId = users[user].userId;
        delete users[user];


        for (uint i = userId; i < allUsers.length - 1; i++) {
            allUsers[i] = allUsers[i + 1];
            users[allUsers[i].userAddress].userId = i;
        }
        allUsers.pop();


        emit UserDeleted(user, msg.sender);
    }


    function submitSKKM(
        string memory activityNameID,
        string memory activityNameEN,
        string memory certificateNumber,
        string memory activityCategory,
        string memory activityType,
        string memory achievement,
        string memory assessmentBasis,
        string memory ipfsHash,
        uint creditPoin
    ) external onlyStudent {
        require(

!studentCertificateNumbers[msg.sender][certificateNumber],
            "Certificate number already exists"
        );


        uint requestId = requests.length;
        requests.push(
            SKKMRequest({
                skkmId: requestId,
                student: msg.sender,
                activityNameID: activityNameID,
                activityNameEN: activityNameEN,
                certificateNumber: certificateNumber,
                activityCategory: activityCategory,
                activityType: activityType,
                achievement: achievement,
                assessmentBasis: assessmentBasis,
                ipfsHash: ipfsHash,
                creditPoin: creditPoin,
                status: Status.Pending,
```

```
                unverifiedMessage: "",
                timestamp: 0
            })
        );
        studentCertificateNumbers[msg.sender][certificateNumber] =
true;
        emit SKKMSubmitted(
            requestId,
            msg.sender,
            ipfsHash,
            certificateNumber,
            msg.sender
        );
    }


    function editSKKM(
        uint requestId,
        string memory activityNameID,
        string memory activityNameEN,
        string memory newCertificateNumber,
        string memory activityCategory,
        string memory activityType,
        string memory achievement,
        string memory assessmentBasis,
        string memory ipfsHash,
        uint creditPoin
    ) external onlyStudent {
        require(requestId < requests.length, "Invalid request
ID");
        SKKMRequest storage request = requests[requestId];
        require(request.student == msg.sender, "Unauthorized");
        require(
            request.status == Status.Pending ||
                request.status == Status.Unverified ||
                request.status == Status.Revised,
            "Cannot edit request after verification"
        );


        require(
            keccak256(bytes(request.certificateNumber)) ==
                keccak256(bytes(newCertificateNumber)) ||

!studentCertificateNumbers[msg.sender][newCertificateNumber],
            "Certificate number already exists"
        );


        studentCertificateNumbers[msg.sender][
            request.certificateNumber
        ] = false;
        request.activityNameID = activityNameID;
        request.activityNameEN = activityNameEN;
        request.certificateNumber = newCertificateNumber;
        request.activityCategory = activityCategory;
        request.activityType = activityType;
        request.achievement = achievement;
```

```
        request.assessmentBasis = assessmentBasis;
        request.ipfsHash = ipfsHash;
        request.creditPoin = creditPoin;


        if (request.status == Status.Unverified) {
            request.status = Status.Revised;
        } else if (request.status == Status.Pending) {
            request.status = Status.Pending;
        }



studentCertificateNumbers[msg.sender][newCertificateNumber] =
true;
        emit SKKMEdited(
            requestId,
            msg.sender,
            ipfsHash,
            newCertificateNumber,
            msg.sender
        );
    }


    function deleteSKKM(uint requestId) external onlyStudent {
        require(requestId < requests.length, "Invalid request
ID");
        SKKMRequest storage request = requests[requestId];
        require(request.student == msg.sender, "Unauthorized");
        require(
            request.status == Status.Pending,
            "Cannot delete request after verification"
        );


        studentCertificateNumbers[msg.sender][
            request.certificateNumber
        ] = false;


        for (uint i = requestId; i < requests.length - 1; i++) {
            requests[i] = requests[i + 1];
            requests[i].skkmId = i;
        }
        requests.pop();


        emit SKKMDeleted(requestId, msg.sender, msg.sender);
    }


    function verifySKKM(
        uint requestId,
        bool isVerified,
        string memory message
    ) external onlyHMJ {
        require(requestId < requests.length, "Invalid request
```

```
ID");
        SKKMRequest storage request = requests[requestId];
        require(
            request.status == Status.Pending ||
                request.status == Status.Revised,
            "Request already processed"
        );
        request.status = isVerified ? Status.Verified :
Status.Unverified;
        request.unverifiedMessage = isVerified ? "" : message;
        emit SKKMVerified(
            requestId,
            msg.sender,
            isVerified,
            message,
            msg.sender
        );
    }


    function validateSKKM(
        uint requestId,
        string memory activityCategory,
        string memory activityType,
        string memory achievement,
        string memory assessmentBasis,
        uint creditPoin
    ) external onlyHMJ {
        require(requestId < requests.length, "Invalid request
ID");
        SKKMRequest storage request = requests[requestId];
        require(
            request.status == Status.Verified,
            "Request not verified by HMJ"
        );


        request.activityCategory = activityCategory;
        request.activityType = activityType;
        request.achievement = achievement;
        request.assessmentBasis = assessmentBasis;
        request.creditPoin = creditPoin;
        request.status = Status.Valid;
        request.timestamp = block.timestamp;
        emit SKKMValidated(requestId, msg.sender, true,
msg.sender);
    }


    function submitQRCode(string memory qrCodeData) external
onlyStudent {
        require(hasGeneratedQRCode[msg.sender], "Generate QR Code
first");
        QRCode storage qrCode = qrcodes[qrcodes.length - 1];
        require(qrCode.status == Status.Generated, "Invalid QR
Code status");
```

```
        qrCode.status = Status.SendToBEM;
        qrCode.qrCodeData = qrCodeData;
        qrCode.timestamp = block.timestamp;


        emit QRCodeSubmitted(
            qrCode.qrCodeId,
            msg.sender,
            qrCodeData,
            qrCode.timestamp,
            msg.sender
        );
    }


    function validateQRCodeByBEM(uint qrCodeId) external onlyBEM {
        require(qrCodeId < qrcodes.length, "Invalid QR Code ID");
        QRCode storage qrCode = qrcodes[qrCodeId];
        require(
            qrCode.status == Status.SendToBEM,
            "QR Code must be sent to BEM first"
        );


        qrCode.status = Status.ValidatedByBEM;
        emit QRCodeValidatedByBEM(
            qrCodeId,
            msg.sender,
            qrCode.student,
            msg.sender
        );
    }


    function getSKKMByStudent(
        address student
    ) external view returns (SKKMRequest[] memory) {
        uint256 count = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (requests[i].student == student) {
                count++;
            }
        }


        SKKMRequest[] memory result = new SKKMRequest[](count);
        uint256 index = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (requests[i].student == student) {
                result[index] = requests[i];
                index++;
            }
        }
        return result;
    }


    function getSKKMByQRCode(
```

```
        string memory qrCode
    ) external view returns (SKKMRequest[] memory) {
        address student = address(0);
        uint qrCodeTimestamp = 0;
        for (uint256 i = 0; i < qrcodes.length; i++) {
            if (
                keccak256(bytes(qrcodes[i].qrCodeData)) ==
                keccak256(bytes(qrCode))
            ) {
                student = qrcodes[i].student;
                qrCodeTimestamp = qrcodes[i].timestamp;
                break;
            }
        }


        require(student != address(0), "QR Code not found");


        uint256 count = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (
                requests[i].student == student &&
                requests[i].status == Status.Valid &&
                requests[i].timestamp <= qrCodeTimestamp
            ) {
                count++;
            }
        }


        SKKMRequest[] memory result = new SKKMRequest[](count);
        uint256 index = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (
                requests[i].student == student &&
                requests[i].status == Status.Valid &&
                requests[i].timestamp <= qrCodeTimestamp
            ) {
                result[index] = requests[i];
                index++;
            }
        }
        return result;
    }


    function getQRNotValidatedByBEM() external view returns
(QRCode[] memory) {
        uint256 count = 0;
        for (uint256 i = 0; i < qrcodes.length; i++) {
            if (qrcodes[i].status == Status.SendToBEM) {
                count++;
            }
        }


        QRCode[] memory result = new QRCode[](count);
```

```
        uint256 index = 0;
        for (uint256 i = 0; i < qrcodes.length; i++) {
            if (qrcodes[i].status == Status.SendToBEM) {
                result[index] = qrcodes[i];
                index++;
            }
        }
        return result;
    }


    function getSKKMNotVerifiedByHMJ()
        external
        view
        returns (SKKMRequest[] memory)
    {
        uint256 count = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (
                requests[i].status == Status.Pending ||
                requests[i].status == Status.Revised
            ) {
                count++;
            }
        }


        SKKMRequest[] memory result = new SKKMRequest[](count);
        uint256 index = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (
                requests[i].status == Status.Pending ||
                requests[i].status == Status.Revised
            ) {
                result[index] = requests[i];
                index++;
            }
        }
        return result;
    }


    function getSKKMNotValidatedByHMJ()
        external
        view
        returns (SKKMRequest[] memory)
    {
        uint256 count = 0;
        for (uint256 i = 0; i < requests.length; i++) {
            if (requests[i].status == Status.Verified) {
                count++;
            }
        }


        SKKMRequest[] memory result = new SKKMRequest[](count);
        uint256 index = 0;
        for (uint256 i = 0; i < requests.length; i++) {
```

```
            if (requests[i].status == Status.Verified) {
                result[index] = requests[i];
                index++;
            }
        }
        return result;
    }


    function setMinimalPoin(uint _minimalPoin) external onlyAdmin
{
        minimalPoin = _minimalPoin;
        emit MinimalPoinUpdated(_minimalPoin, msg.sender);
    }


    function generateQRCode(string memory qrCodeData) external
onlyStudent {
        uint totalPoin = 0;
        for (uint i = 0; i < requests.length; i++) {
            if (
                requests[i].student == msg.sender &&
                requests[i].status == Status.Valid
            ) {
                totalPoin += requests[i].creditPoin;
            }
        }
        require(totalPoin >= minimalPoin, "Minimal poin not
reached");


        uint qrCodeId = qrcodes.length;
        uint currentTimestamp = block.timestamp;
        qrcodes.push(
            QRCode({
                qrCodeId: qrCodeId,
                student: msg.sender,
                qrCodeData: qrCodeData,
                status: Status.Generated,
                timestamp: currentTimestamp
            })
        );
        lastQRCodeTimestamp[msg.sender] = currentTimestamp;
        hasGeneratedQRCode[msg.sender] = true;


        emit QRCodeGenerated(
            qrCodeId,
            msg.sender,
            qrCodeData,
            currentTimestamp,
            msg.sender
        );
    }


    function getQRCodeByStudent(
        address student
```

```
    ) external view returns (QRCode memory) {
        require(users[student].exists, "Student does not exist");
        require(hasGeneratedQRCode[student], "QR Code not
generated");
        QRCode storage qrCode = qrcodes[qrcodes.length - 1];
        return qrCode;
    }


    function getAllSKKM() external view returns (SKKMRequest[]
memory) {
        return requests;
    }


    function getAllUsers() external view returns (User[] memory) {
        return allUsers;
    }


    function getAllQRCodes() external view returns (QRCode[]
memory) {
        return qrcodes;
    }
}
```

5.1.2.2 Struktur Data

Struktur data dalam *smart contract* SKKM berfungsi untuk mengelompokkan berbagai variabel yang digunakan dalam sistem. Struktur data ini membantu dalam pengelolaan informasi pengguna, permintaan SKKM, dan *QR Code*. Berikut adalah penjelasan lengkap dari masing-masing struktur data yang digunakan dalam *smart contract* ini :

1. Role

Tabel 5.13 Struktur data : *Role*

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
Role	enum	Enumerasi, tipe data khusus yang digunakan untuk mendefinisikan variabel yang dapat memiliki beberapa nilai yang telah ditentukan sebelumnya.	Enumerasi untuk peran pengguna dalam sistem.
None	0	Angka 0 mewakili	Tidak memiliki

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
		peran "None".	peran.
Student	1	Angka 1 mewakili peran "Student".	Mahasiswa.
HMJ	2	Angka 2 mewakili peran "HMJ".	Himpunan Mahasiswa Jurusan.
Admin	3	Angka 3 mewakili peran "*Admin*".	*Admin*.
BEM	4	Angka 4 mewakili peran "BEM".	Badan Eksekutif Mahasiswa.

2. Status

Tabel 5.14 Struktur data : Status

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
Status	enum	Enumerasi, tipe data khusus yang digunakan untuk mendefinisikan variabel yang dapat memiliki beberapa nilai yang telah ditentukan sebelumnya.	Enumerasi untuk status dari permintaan SKKM dan *QR Code*.
Pending	0	Angka 0 mewakili status "*Pending*".	Permintaan dalam status menunggu verifikasi.
Verified	1	Angka 1 mewakili status "Verified".	Permintaan telah diverifikasi oleh HMJ.
Unverified	2	Angka 2 mewakili status "*Unverified*".	Permintaan tidak diverifikasi oleh HMJ.
Valid	3	Angka 3 mewakili status "Valid".	Permintaan telah divalidasi oleh HMJ dan BEM.
Revised	4	Angka 4 mewakili	Permintaan telah

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
		status "*Revised*".	direvisi oleh mahasiswa.
Generated	5	Angka 5 mewakili status "*Generated*".	*QR Code* telah dihasilkan.
SendToBEM	6	Angka 6 mewakili status "*SendToBEM*".	*QR Code* telah dikirim ke BEM untuk validasi.
ValidatedBy BEM	7	Angka 7 mewakili status "*ValidatedByBEM*".	*QR Code* telah divalidasi oleh BEM.

3. *User*

Tabel 5.15 Struktur data : *User*

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
User	struct	Struktur, tipe data yang digunakan untuk mengelompokkan beberapa variabel menjadi satu kesatuan.	Struktur data untuk pengguna sistem.
`userId`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	*ID* unik untuk setiap pengguna.
`userAddress`	address	Alamat *Ethereum*, tipe data khusus untuk alamat dompet *Ethereum*.	Alamat *Ethereum* dari pengguna.
`name`	string	String, tipe data untuk teks.	Nama pengguna.
`identifier`	string	String, tipe data untuk teks.	Identifikasi unik pengguna (misal: NIM).
`department`	string	String, tipe data untuk teks.	Departemen atau jurusan pengguna (misal: Teknik

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
			Informatika, Manajemen).
`role`	Role	Enumerasi, tipe data khusus yang digunakan untuk mendefinisikan variabel yang dapat memiliki beberapa nilai yang telah ditentukan sebelumnya.	Peran pengguna dalam sistem (None, Student, HMJ, *Admin*, BEM).
`exists`	bool	Boolean, tipe data untuk nilai benar atau salah.	Indikator apakah pengguna ada dalam sistem (*true* jika ada, *false* jika tidak ada).

4. *SKKMRequest*

Tabel 5.16 Struktur data : *SKKMRequest*

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
SKKMRequest	struct	Struktur, tipe data yang digunakan untuk mengelompokkan beberapa variabel menjadi satu kesatuan.	Struktur data untuk permintaan pencatatan SKKM.
`skkmId`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	*ID* unik untuk setiap permintaan SKKM.
`student`	address	Alamat *Ethereum*, tipe data khusus untuk alamat dompet *Ethereum*.	Alamat *Ethereum* dari mahasiswa yang mengajukan SKKM.

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
`activityNameID`	string	String, tipe data untuk teks.	Nama kegiatan dalam bahasa Indonesia.
`activityNameEN`	string	String, tipe data untuk teks.	Nama kegiatan dalam bahasa Inggris.
`certificateNumber`	string	String, tipe data untuk teks.	Nomor sertifikat kegiatan.
`activityCategory`	string	String, tipe data untuk teks.	Kategori kegiatan (misal: akademik, non-akademik).
`activityType`	string	String, tipe data untuk teks.	Jenis kegiatan (misal: seminar, workshop, lomba).
`achievement`	string	String, tipe data untuk teks.	Prestasi yang dicapai dalam kegiatan (misal: juara 1, peserta).
`assessmentBasis`	string	String, tipe data untuk teks.	Dasar penilaian kegiatan (misal: kehadiran, hasil karya).
`ipfsHash`	string	String, tipe data untuk teks.	Hash *IPFS* dari bukti keikutsertaan (misal: sertifikat, surat keterangan).
`creditPoin`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	Poin kredit yang diberikan untuk kegiatan tersebut.

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
`status`	Status	Enumerasi, tipe data khusus yang digunakan untuk mendefinisikan variabel yang dapat memiliki beberapa nilai yang telah ditentukan sebelumnya.	Status dari permintaan SKKM (*Pending*, Verified, *Unverified*, Valid, *Revised*, *Generated*, *SendToBEM*, *ValidatedByBEM*).
`unverifiedMessage`	string	String, tipe data untuk teks.	Pesan jika permintaan tidak diverifikasi (misal: alasan penolakan).
`timestamp`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	Waktu ketika permintaan dibuat atau diperbarui (dalam format UNIX timestamp).

5. QRCode

Tabel 5.17 Struktur data : QRCode

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
QRCode	struct	Struktur, tipe data yang digunakan untuk mengelompokkan beberapa variabel menjadi satu kesatuan.	Struktur data untuk *QR Code* yang dihasilkan.
`qrCodeId`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	*ID* unik untuk setiap *QR Code*.

Struktur Data	Tipe Data	Penjelasan Tipe Data	Deskripsi
`student`	address	Alamat *Ethereum*, tipe data khusus untuk alamat dompet *Ethereum*.	Alamat *Ethereum* dari mahasiswa yang menghasilkan *QR Code*.
`qrCodeData`	string	String, tipe data untuk teks.	Data yang terkandung dalam *QR Code* (misal: *hash* data SKKM).
`status`	Status	Enumerasi, tipe data khusus yang digunakan untuk mendefinisikan variabel yang dapat memiliki beberapa nilai yang telah ditentukan sebelumnya.	Status dari *QR Code* (*Generated*, *SendToBEM*, *ValidatedByBEM*).
`timestamp`	uint	Unsigned Integer, tipe data bilangan bulat tanpa tanda.	Waktu ketika *QR Code* dihasilkan atau diperbarui (dalam format UNIX timestamp).

5.1.2.3 *Event*

Event dalam *smart contract* SKKM digunakan untuk mencatat dan mengumumkan berbagai aktivitas yang terjadi dalam sistem, seperti pengajuan permintaan SKKM, verifikasi dan validasi, serta penambahan atau penghapusan pengguna. *Event* ini membantu dalam pelacakan dan auditing aktivitas dalam sistem. Berikut adalah penjelasan lengkap dari *event-event* yang digunakan dalam *smart contract* ini :

Tabel 5.18 *Event*

Event	Parameter	Deskripsi
`SKKMSubmitted`	uint requestId, address indexed student, string ipfsHash, string certificateNumber, address triggeredBy	*Event* yang dipicu saat permintaan SKKM diajukan oleh mahasiswa.
`SKKMVerified`	uint requestId, address indexed verifier, bool isVerified, string message, address triggeredBy	*Event* yang dipicu saat permintaan SKKM diverifikasi oleh HMJ.
`SKKMValidated`	uint requestId, address indexed *validator*, bool isValid, address triggeredBy	*Event* yang dipicu saat permintaan SKKM divalidasi oleh HMJ.
`SKKMEdited`	uint requestId, address indexed student, string ipfsHash, string certificateNumber, address triggeredBy	*Event* yang dipicu saat permintaan SKKM diedit oleh mahasiswa.
`SKKMDeleted`	uint requestId, address indexed student, address triggeredBy	*Event* yang dipicu saat permintaan SKKM dihapus oleh mahasiswa.
`UserDeleted`	address indexed user, address triggeredBy	*Event* yang dipicu saat pengguna dihapus dari sistem oleh admin atau super admin.
`MinimalPoinUpdated`	uint newMinimalPoin, address triggeredBy	*Event* yang dipicu saat minimal poin diubah oleh admin atau super admin.
`QRCodeGenerated`	uint qrCodeId, address indexed student, string qrCodeData, uint timestamp, address triggeredBy	*Event* yang dipicu saat *QR Code* dihasilkan oleh mahasiswa.

Event	Parameter	Deskripsi
`QRCodeSubmitted`	uint qrCodeId, address indexed student, string qrCodeData, uint timestamp, address triggeredBy	*Event* yang dipicu saat *QR Code* dikirim ke BEM oleh mahasiswa.
`UserAdded`	uint userId, address indexed user, string name, string identifier, string department, Role role, address triggeredBy	*Event* yang dipicu saat pengguna baru ditambahkan ke sistem oleh admin atau super admin.
`QRCodeValidatedByBEM`	uint qrCodeId, address indexed *validator*, address indexed student, address triggeredBy	*Event* yang dipicu saat *QR Code* divalidasi oleh BEM.

5.1.2.4 *Modifier*

Modifier dalam *smart contract* SKKM digunakan untuk membatasi akses ke fungsi-fungsi tertentu, sehingga hanya pengguna dengan peran atau hak akses tertentu yang dapat menjalankan fungsi tersebut. *Modifier* ini memastikan bahwa operasi dalam sistem dilakukan oleh pengguna yang berwenang. Berikut adalah penjelasan lengkap dari *modifier* yang digunakan dalam *smart contract* ini :

Tabel 5.19 *Modifier*

Modifier	Deskripsi
`onlySuperAdmin`	Membatasi akses hanya untuk super admin. *Modifier* ini memastikan bahwa hanya super admin yang dapat menjalankan fungsi yang dilindungi oleh *modifier* ini.
`onlyAdmin`	Membatasi akses hanya untuk admin atau super admin. *Modifier* ini memastikan bahwa hanya admin atau super admin yang dapat menjalankan fungsi yang dilindungi oleh *modifier* ini.
`onlyStudent`	Membatasi akses hanya untuk mahasiswa. *Modifier* ini memastikan bahwa hanya mahasiswa yang dapat menjalankan fungsi yang dilindungi oleh *modifier* ini.

Modifier	Deskripsi
`onlyHMJ`	Membatasi akses hanya untuk HMJ. *Modifier* ini memastikan bahwa hanya anggota HMJ yang dapat menjalankan fungsi yang dilindungi oleh *modifier* ini.
`onlyBEM`	Membatasi akses hanya untuk BEM. *Modifier* ini memastikan bahwa hanya anggota BEM yang dapat menjalankan fungsi yang dilindungi oleh *modifier* ini.

5.1.2.5 Fungsi Utama

Fungsi-fungsi utama dalam *smart contract* SKKM dirancang untuk mengelola berbagai operasi yang dilakukan oleh pengguna sistem, seperti menambahkan pengguna, mengajukan dan mengelola permintaan SKKM, serta menghasilkan dan memvalidasi *QR Code*. Berikut adalah penjelasan lengkap dari fungsi-fungsi utama yang ada dalam *smart contract* ini :

Tabel 5.20 Fungsi Utama

Fungsi	Deskripsi
`addUser`	Fungsi ini digunakan untuk menambahkan pengguna baru ke sistem dengan peran tertentu. Fungsi ini hanya dapat dijalankan oleh super admin atau admin. Saat pengguna baru ditambahkan, informasi seperti alamat *Ethereum*, nama, identifikasi unik (misal: NIM), dan departemen pengguna disimpan dalam sistem. Jika ada nilai ether yang dikirim bersama dengan transaksi, nilai tersebut akan ditransfer ke alamat pengguna yang baru ditambahkan. Fungsi ini juga memancarkan *event* `UserAdded` untuk mencatat penambahan pengguna baru.
`updateUser`	Fungsi ini digunakan untuk memperbarui informasi pengguna yang sudah ada dalam sistem. Fungsi ini hanya dapat dijalankan oleh admin atau super admin. Informasi yang dapat diperbarui termasuk nama, identifikasi unik, departemen, dan peran pengguna. Fungsi ini memastikan

Fungsi	Deskripsi
	bahwa identifikasi unik yang baru tidak sudah ada dalam sistem untuk mencegah duplikasi. Fungsi ini juga memperbarui daftar semua pengguna dan memancarkan *event* `UserUpdated` untuk mencatat perubahan informasi pengguna.
`deleteUser`	Fungsi ini digunakan untuk menghapus pengguna dari sistem. Fungsi ini hanya dapat dijalankan oleh admin atau super admin. Ketika pengguna dihapus, identifikasi unik mereka dihapus dari daftar identifikasi, dan pengguna tersebut dihapus dari daftar semua pengguna. Fungsi ini juga memancarkan *event* *`UserDeleted`* untuk mencatat penghapusan pengguna.
`submitSKKM`	Fungsi ini digunakan oleh mahasiswa untuk mengajukan permintaan SKKM baru. Fungsi ini memastikan bahwa nomor sertifikat yang diajukan belum pernah digunakan sebelumnya oleh mahasiswa yang sama. Informasi yang diajukan termasuk nama kegiatan, nomor sertifikat, kategori dan jenis kegiatan, prestasi, dasar penilaian, *hash IPFS* dari bukti keikutsertaan, dan poin kredit. Permintaan baru ditambahkan ke daftar semua permintaan SKKM dan *event* *`SKKMSubmitted`* dipancarkan untuk mencatat pengajuan permintaan baru.
`editSKKM`	Fungsi ini digunakan oleh mahasiswa untuk mengedit permintaan SKKM yang sudah ada. Fungsi ini memastikan bahwa hanya permintaan yang belum diverifikasi atau yang telah direvisi yang dapat diedit. Informasi yang dapat diedit termasuk nama kegiatan, nomor sertifikat, kategori dan jenis kegiatan, prestasi, dasar penilaian, *hash IPFS* dari bukti

Fungsi	Deskripsi
	keikutsertaan, dan poin kredit. Fungsi ini juga memastikan bahwa nomor sertifikat yang baru tidak sudah ada dalam sistem untuk mencegah duplikasi. *Event* `SKKMEdited` dipancarkan untuk mencatat perubahan permintaan SKKM.
`deleteSKKM`	Fungsi ini digunakan oleh mahasiswa untuk menghapus permintaan SKKM yang sudah ada. Fungsi ini memastikan bahwa hanya permintaan yang belum diverifikasi yang dapat dihapus. Permintaan yang dihapus dihapus dari daftar semua permintaan SKKM dan *event* `SKKMDeleted` dipancarkan untuk mencatat penghapusan permintaan.
`verifySKKM`	Fungsi ini digunakan oleh anggota HMJ untuk memverifikasi permintaan SKKM. Fungsi ini memastikan bahwa hanya permintaan yang sedang dalam status pending atau direvisi yang dapat diverifikasi. Anggota HMJ dapat memilih untuk memverifikasi atau menolak permintaan dengan memberikan pesan jika ditolak. Status permintaan diperbarui sesuai dengan verifikasi dan *event* `SKKMVerified` dipancarkan untuk mencatat hasil verifikasi.
`validateSKKM`	Fungsi ini digunakan oleh anggota HMJ untuk memvalidasi permintaan SKKM yang telah diverifikasi. Fungsi ini memastikan bahwa hanya permintaan yang telah diverifikasi oleh HMJ yang dapat divalidasi. Informasi yang dapat divalidasi termasuk kategori dan jenis kegiatan, prestasi, dasar penilaian, dan poin kredit. Status permintaan diperbarui menjadi valid dan *event* `SKKMValidated` dipancarkan untuk mencatat hasil validasi.

Fungsi	Deskripsi
submitQRCode	Fungsi ini digunakan oleh mahasiswa untuk mengirim *QR Code* yang dihasilkan ke BEM untuk validasi. Fungsi ini memastikan bahwa mahasiswa telah menghasilkan *QR Code* sebelum mengirimkannya. Status *QR Code* diperbarui menjadi dikirim ke BEM dan *event* `QRCodeSubmitted` dipancarkan untuk mencatat pengiriman *QR Code*.
validateQRCodeByBEM	Fungsi ini digunakan oleh anggota BEM untuk memvalidasi *QR Code*. Fungsi ini memastikan bahwa *QR Code* yang divalidasi telah dikirim ke BEM sebelumnya. Status *QR Code* diperbarui menjadi divalidasi oleh BEM dan *event* `QRCodeValidatedByBEM` dipancarkan untuk mencatat hasil validasi.
getSKKMByStudent	Fungsi ini digunakan untuk mengambil daftar permintaan SKKM berdasarkan alamat mahasiswa. Fungsi ini mengembalikan semua permintaan SKKM yang diajukan oleh mahasiswa yang bersangkutan.
getSKKMByQRCode	Fungsi ini digunakan untuk mengambil daftar permintaan SKKM berdasarkan data *QR Code*. Fungsi ini mencari *QR Code* yang sesuai dan mengembalikan semua permintaan SKKM yang valid dan dihasilkan sebelum *QR Code* tersebut.
getQRNotValidatedByBEM	Fungsi ini digunakan untuk mengambil daftar *QR Code* yang belum divalidasi oleh BEM. Fungsi ini mengembalikan semua *QR Code* yang statusnya masih dikirim ke BEM.
getSKKMNotVerifiedByHMJ	Fungsi ini digunakan untuk mengambil daftar permintaan SKKM yang belum diverifikasi oleh HMJ. Fungsi ini

Fungsi	Deskripsi
	mengembalikan semua permintaan SKKM yang statusnya masih pending atau direvisi.
`getSKKMNotValidatedByHMJ`	Fungsi ini digunakan untuk mengambil daftar permintaan SKKM yang belum divalidasi oleh HMJ. Fungsi ini mengembalikan semua permintaan SKKM yang statusnya telah diverifikasi tetapi belum divalidasi.
`setMinimalPoin`	Fungsi ini digunakan untuk mengatur poin minimal yang dibutuhkan untuk menghasilkan *QR Code*.
`generateQRCode`	Fungsi ini digunakan untuk mengatur poin minimal yang dibutuhkan untuk menghasilkan *QR Code*. Fungsi ini hanya dapat dijalankan oleh admin atau super admin. *Event* `MinimalPoinUpdated` dipancarkan untuk mencatat perubahan minimal poin.
`getQRCodeByStudent`	Fungsi ini digunakan oleh mahasiswa untuk menghasilkan *QR Code* setelah mencapai poin minimal yang dibutuhkan. Fungsi ini menghitung total poin kredit yang dimiliki mahasiswa dan memastikan bahwa poin tersebut memenuhi syarat minimal. *QR Code* baru ditambahkan ke daftar semua *QR Code* dan *event* `QRCodeGenerated` dipancarkan untuk mencatat pembuatan *QR Code*.
`getAllSKKM`	Fungsi ini digunakan untuk mengambil *QR Code* berdasarkan alamat mahasiswa. Fungsi ini mengembalikan *QR Code* yang dihasilkan oleh mahasiswa yang bersangkutan.
`getAllUsers`	Fungsi ini digunakan untuk mengambil semua pengguna yang ada dalam sistem. Fungsi ini mengembalikan daftar lengkap dari semua pengguna..

Fungsi	Deskripsi
`getAllQRCodes`	Fungsi ini digunakan untuk mengambil semua *QR Code* yang ada dalam sistem. Fungsi ini mengembalikan daftar lengkap dari semua *QR Code*.

5.1.2.6 Deploy *Smart Contract*

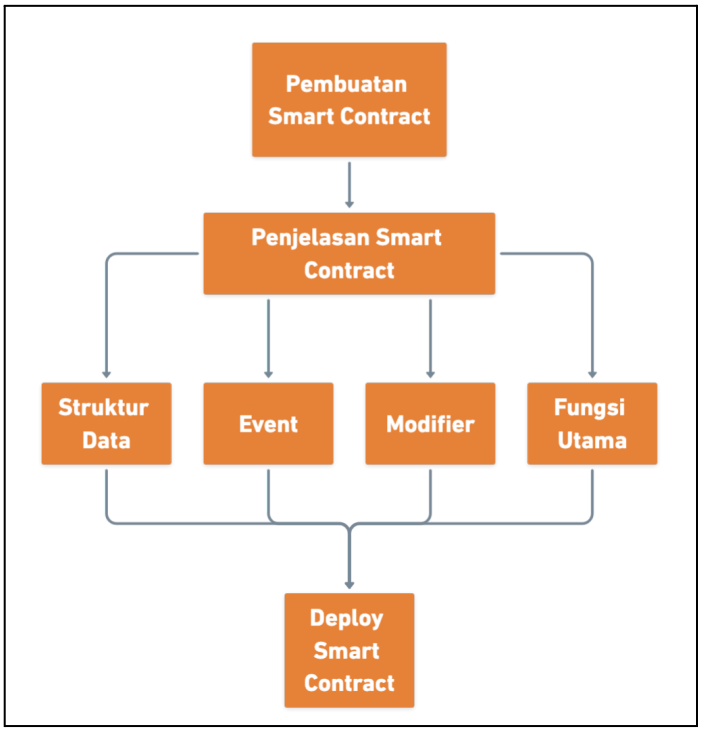
Setelah *smart contract* selesai dibuat, langkah selanjutnya adalah melakukan deploy ke jaringan *blockchain* menggunakan *Thirdweb*. *Thirdweb* menyediakan antarmuka yang memudahkan proses deploy dan pengelolaan *smart contract*. Berikut adalah langkah-langkah lengkap untuk melakukan deploy *smart contract* menggunakan *Thirdweb*:

1. Deploy *Smart Contract* Menggunakan *Thirdweb*
    a. Buka Terminal dan Navigasikan ke Direktori Proyek:
        - Buka terminal dan navigasikan ke direktori proyek tempat *smart contract* berada.
    b. Jalankan Perintah *Thirdweb* Deploy:
        - Jalankan perintah `npx thirdweb deploy` di terminal. Perintah ini akan membuka antarmuka *Thirdweb* di browser.
    c. Hubungkan *Wallet Metamask*:
        - Di antarmuka *Thirdweb*, pilih opsi untuk menghubungkan *wallet*.
        - *Metamask* akan meminta izin untuk terhubung. Pilih akun yang akan digunakan dan izinkan akses.
    d. Isi Informasi Jaringan di *Metamask*:
        - Jika belum terhubung ke jaringan yang benar, *Metamask* akan meminta untuk menambahkan jaringan secara manual.
        - Isi informasi jaringan seperti *Network Name*, *New RPC URL*, *Chain ID*, *Currency Symbol*, dan *Block Explorer URL*.

e. Konfirmasi Transaksi di *Metamask*:
   - Setelah jaringan terhubung, *Metamask* akan meminta konfirmasi untuk melakukan deploy *smart contract*.
   - Konfirmasi transaksi dengan mengklik tombol "*Confirm*" di *Metamask*.

f. Tunggu Proses *Deploy* Selesai:
   - *Thirdweb* akan menampilkan status proses *deploy*. Tunggu hingga proses selesai dan *smart contract* berhasil di-*deploy*.

Dengan langkah-langkah tersebut, *smart contract* berhasil di-deploy ke jaringan *blockchain* menggunakan *Thirdweb*, dan siap untuk diintegrasikan dengan aplikasi *frontend*.



Gambar 5.2 Alur Implementasi *Smart Contract*

5.2 Implementasi *Frontend*

5.2.1 Struktur Proyek *Frontend*

Struktur proyek *frontend* dirancang untuk memisahkan komponen-komponen utama aplikasi, sehingga memudahkan pengembangan dan pemeliharaan. Berikut adalah struktur *direktori* dan *file* dalam proyek *frontend* yang berhubungan dengan *Thirdweb* atau *blockchain*:

- abis: Direktori yang berisi file *ABI* (*Application Binary Interface*) untuk *smart contract*.
- *src*: Direktori utama untuk semua kode sumber aplikasi.
    - *components*/SKKM: Berisi komponen terkait SKKM seperti *Navbar*, *Sidebar*, dan *SpeedDial*.
    - *constants*: Berisi data yang belum di *database* kan, seperti *credit point* kegiatan wajib dan pilihan departemen untuk select *option*.
    - *pages*/SKKM: Berisi halaman terkait SKKM untuk berbagai peran seperti *Admin*, BEM, HMJ, dan Student.

5.2.2 Pemasangan Dependensi

Proyek *frontend* ini membutuhkan beberapa dependensi untuk bisa berjalan dengan baik. Langkah-langkah pemasangannya meliputi instalasi *Tailwind CSS*, *Thirdweb SDK*, dan dependensi tambahan seperti *React Router*, dll. Proses pemasangan dilakukan dengan menjalankan perintah pemasangan untuk masing-masing dependensi yang dibutuhkan.

5.2.3 Konfigurasi Lingkungan

Untuk menghubungkan aplikasi *frontend* dengan *smart contract* dan layanan lain, variabel lingkungan (*environment variables*) perlu diatur. Buat *file* *.env* di *root* proyek yang mencantumkan alamat kontrak dan *client ID*.

- *VITE_CONTRACT_ADDRESS*: Alamat kontrak *smart contract* yang telah di-*deploy*.
- *VITE_CLIENT_ID*: *ID* klien untuk menghubungkan aplikasi dengan *Thirdweb*.

5.2.4 Integrasi dengan *Smart Contract*

Frontend berinteraksi dengan *smart contract* yang telah di-*deploy* menggunakan *Thirdweb SDK*. Integrasi ini memungkinkan aplikasi *frontend* untuk memanggil fungsi-fungsi dalam *smart contract* seperti menambahkan pengguna, mengajukan SKKM, memvalidasi SKKM, dan lain-lain. Integrasi dilakukan dengan mengimpor *Thirdweb SDK* dan mengkonfigurasi *ThirdwebProvider* di *file* *Main.jsx*.

- *Client ID*: Diatur di Main.jsx untuk mengkonfigurasi ThirdwebProvider.

- *Contract Address*: Diimpor dan digunakan dalam setiap komponen yang membutuhkan interaksi dengan *smart contract*. Misalnya, di *pages/student/SubmitSKKM.jsx*, *contract address* digunakan untuk mengajukan permintaan SKKM.

5.3 Pengujian Sistem

5.3.1 Pengujian Jaringan *Blockchain*

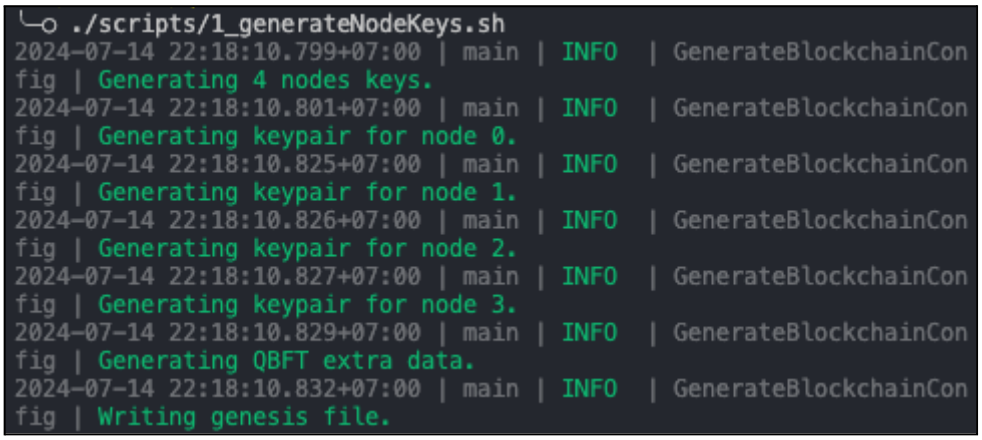
Pengujian sistem dilakukan secara komprehensif untuk memastikan bahwa sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis *blockchain* berfungsi dengan baik, aman, dan handal. Pada tahap ini, fokus pengujian adalah pada fungsionalitas sistem, yaitu untuk memastikan bahwa semua fitur dan fungsi sistem berjalan sesuai dengan yang diharapkan.

1. Pengujian Skrip *generateNodeKeys.sh*

Tabel 5.21 Pengujian Skrip: *generateNodeKeys.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *generateNodeKeys.sh* pada *terminal* atau *command prompt*.	Skrip berjalan tanpa *error*.	✓
2	Memeriksa apakah *direktori networkFiles* telah dibuat di lokasi yang ditentukan dalam skrip.	*Direktori networkFiles* berhasil dibuat.	✓
3	Memeriksa isi *direktori networkFiles*.	Terdapat *file genesis.json* dan *subdirektori* bernama *keys* di dalam *direktori networkFiles*.	✓
4	Memastikan bahwa *file genesis.json* berisi informasi konfigurasi yang sesuai.	File genesis.json berisi informasi seperti *chainId*, *algorithm*, *gasLimit*, dll.	✓
5	Memeriksa isi subdirektori *keys* di dalam *networkFiles*.	Terdapat pasangan file kunci (misalnya key1, key1.pub, key2, key2.pub, dll.) untuk	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
		setiap *node*.	
6	Memastikan bahwa setiap pasangan kunci terdiri dari kunci privat (key) dan kunci publik (key.pub).	Setiap pasangan kunci terdiri dari dua *file*: kunci privat (key) dan kunci publik (key.pub).	✓
7	Memeriksa format dan integritas kunci kriptografi yang dihasilkan.	Kunci kriptografi memiliki format yang benar dan valid.	✓



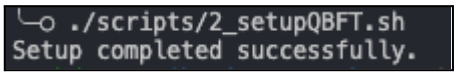
Gambar 5.3 Hasil Pengujian Skrip: *generateNodeKeys.sh*

2. Pengujian Skrip *setupQBFT.sh*:

Tabel 5.22 Pengujian Skrip: *setupQBFT.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *setupQBFT.sh* pada terminal atau *command prompt*.	Skrip berjalan tanpa *error*.	✓
2	Memverifikasi bahwa *direktori* untuk setiap *node* (Node-1, Node-2, Node-3, dan Node-4) telah dibuat.	*Direktori* untuk setiap *node* (Node-1, Node-2, Node-3, dan Node-4) berhasil dibuat.	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
3	Memeriksa isi *direktori* data di dalam setiap direktori *node*.	Setiap *direktori node* memiliki *subdirektori* data yang berisi *file genesis.json* dan pasangan kunci (key, key.pub) yang sesuai.	✓
4	Memastikan bahwa *file genesis.json* dan pasangan kunci yang sesuai telah disalin ke *direktori* data setiap *node*.	File *genesis.json* dan pasangan kunci (key, key.pub) yang sesuai berhasil disalin ke *direktori* data masing-masing *node*.	✓

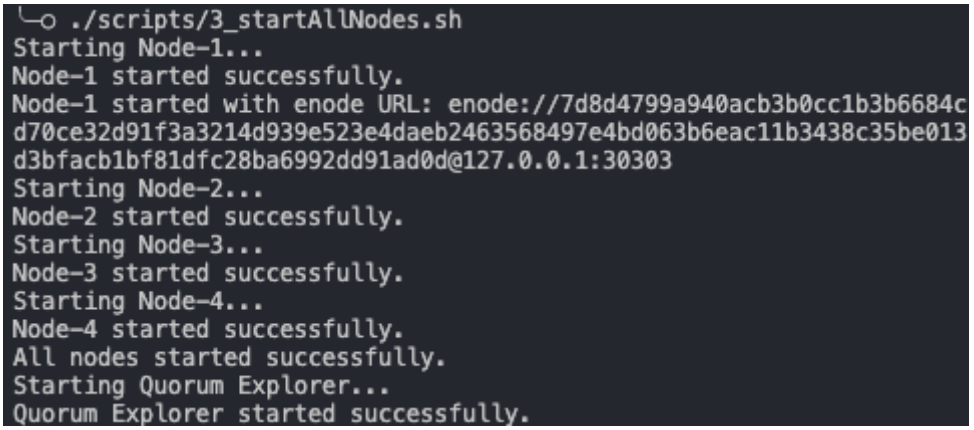


Gambar 5.4 Hasil Pengujian Skrip: *setupQBFT.sh*

3. Pengujian Skrip *startAllNodes.sh*:

Tabel 5.23 Pengujian Skrip: *startAllNodes.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *startAllNodes.sh* pada terminal atau *command prompt*.	Skrip berjalan tanpa *error*.	✓
2	Mengamati *output* di terminal untuk memastikan bahwa setiap *node* berhasil dimulai tanpa pesan kesalahan.	Semua *node* berhasil dimulai tanpa pesan kesalahan di terminal.	✓
3	Memeriksa *file* log (node1.log, node2.log, dst.) di direktori *logs* untuk melihat informasi lebih detail tentang proses *startup* setiap *node*.	File log menunjukkan bahwa setiap *node* telah berhasil memulai dan bergabung dalam jaringan.	✓

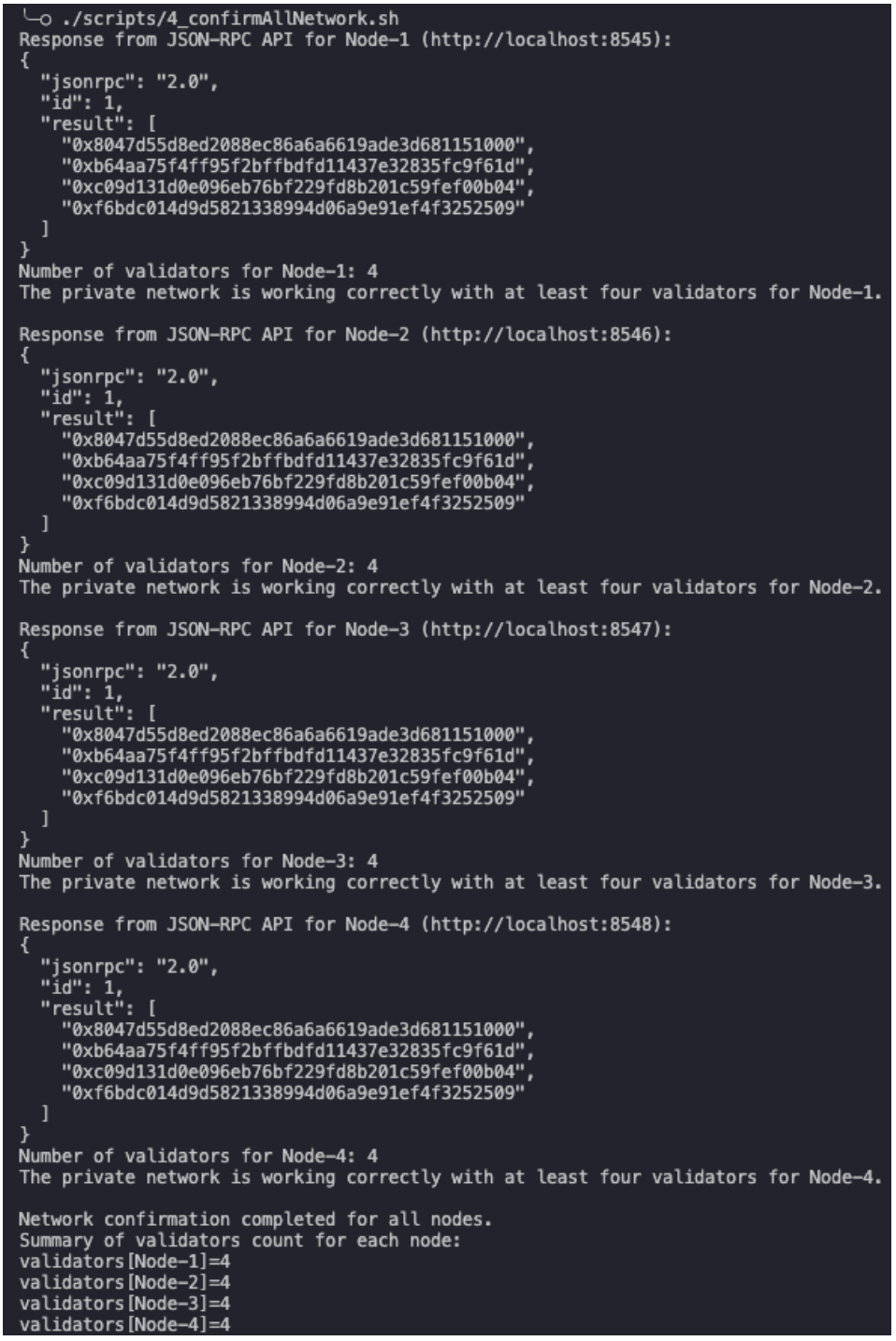


Gambar 5.5 Hasil Pengujian Skrip: *startAllNodes.sh*

4. Pengujian Skrip *confirmAllNetwork.sh*:

Tabel 5.24 Pengujian Skrip: *confirmAllNetwork.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *confirmAllNetwork.sh* pada terminal atau *command prompt*.	Skrip berjalan tanpa *error*.	✓
2	Memeriksa output skrip untuk memastikan bahwa setiap *node* merespons dengan daftar *validator* yang sama.	Semua *node* merespons dengan daftar *validator* yang identik, yang menunjukkan bahwa mereka telah mencapai konsensus.	✓
3	Memeriksa *file validator_count.log* di direktori *logs* untuk mencatat jumlah *validator* untuk setiap *node*.	*File validator_count.log* mencatat jumlah *validator* untuk setiap *node*.	✓
4	Memastikan skrip menampilkan ringkasan jumlah *validator* untuk setiap *node* di akhir proses.	Skrip menampilkan ringkasan jumlah *validator* untuk setiap *node* di akhir proses.	✓



Gambar 5.6 Hasil Pengujian Skrip: *confirmAllNetwork.sh*

5. Pengujian Skrip *addValidator.sh*:

Tabel 5.25 Pengujian Skrip: *addValidator.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *addValidator.sh* pada terminal atau *command prompt*.	Skrip berjalan tanpa *error*.	✓
2	Memverifikasi bahwa *node* baru berhasil dimulai dengan memeriksa *file* log yang sesuai di direktori *logs*.	*File* log untuk *node* baru menunjukkan bahwa *node* telah berhasil dimulai tanpa pesan kesalahan.	✓
3	Menjalankan skrip *confirmAllNetwork.sh* untuk memastikan bahwa *node* baru telah ditambahkan sebagai *validator*.	Semua *node*, termasuk *node* baru, merespons dengan daftar *validator* yang identik.	✓

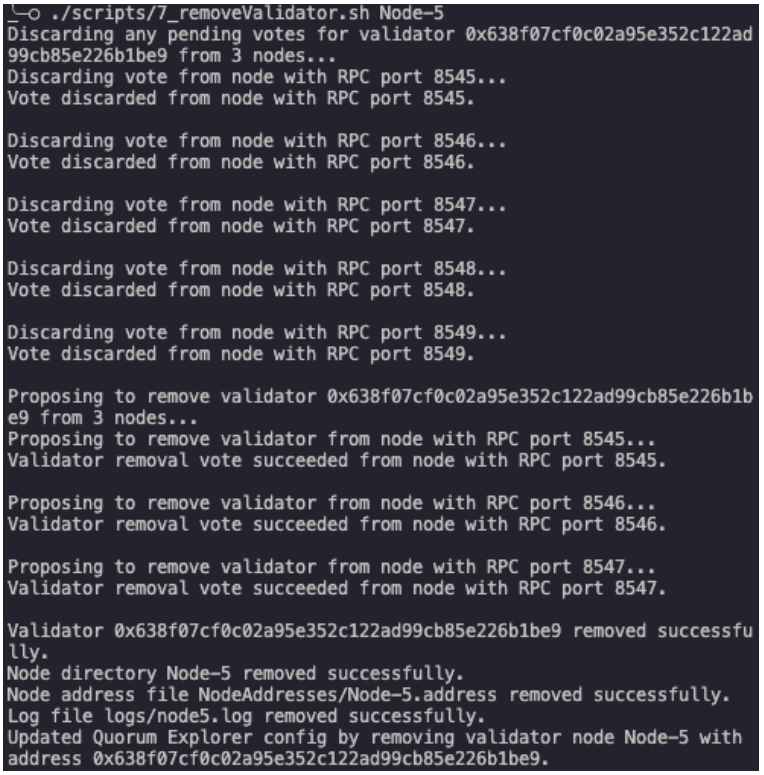
```
└─○ ./scripts/6_addValidator.sh
Starting new validator node (Node-5) with P2P port 30307 and RPC HTTP
port 8549...
New validator node started with enode URL: enode://e0474cf2fc389ec5e5c
720a7ccc94a22157c387cd2d62e06f1d77572528a55357f70f40305bd7d2336a72ece3
fb6bc90cfa8d9894e161bd841808010e1eb14c7@127.0.0.1:30307
New validator node address: 0x638f07cf0c02a95e352c122ad99cb85e226b1be9
Proposing new validator from 3 nodes...
Proposing new validator from node with RPC port 8545...
Validator vote succeeded from node with RPC port 8545.

Proposing new validator from node with RPC port 8546...
Validator vote succeeded from node with RPC port 8546.

Proposing new validator from node with RPC port 8547...
Validator vote succeeded from node with RPC port 8547.

New validator node added successfully (Node-5).
Validator with rpcUrl http://localhost:8549 already exists in the conf
ig file.
```

Gambar 5.7 Hasil Pengujian Skrip: *addValidator.sh*

6. Pengujian Skrip *removeValidator.sh*:

Tabel 5.26 Pengujian Skrip: *removeValidator.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *removeValidator.sh* <alamat_validator>, dengan <alamat_validator> adalah alamat *node* yang akan dihapus.	Skrip berjalan tanpa error dan *node* yang ditargetkan mulai proses penghapusan.	✓
2	Menggunakan skrip *confirmAllNetwork.sh* untuk memverifikasi bahwa *node* yang ditargetkan tidak lagi terdaftar sebagai *validator*.	Semua *node* lainnya melaporkan bahwa *node* yang ditargetkan tidak lagi terdaftar sebagai *validator*.	✓

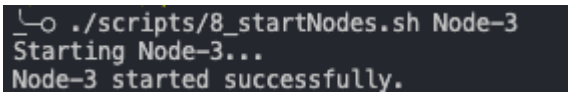


Gambar 5.8 Hasil Pengujian Skrip: *removeValidator.sh*

7. Pengujian Skrip *startNodes.sh*:

Tabel 5.27 Pengujian Skrip: *startNodes.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *startNodes.sh* <*Node*-X>, dengan <*Node*-X> adalah *node* yang ingin Anda mulai (misalnya, *Node*-2).	Skrip berjalan tanpa error dan *node* yang ditentukan mulai proses *startup*.	✓
2	Memverifikasi bahwa *node* berjalan tanpa kesalahan dengan memeriksa file log *nodeX.log* di *direktori* logs.	*File* log menunjukkan bahwa *node* telah dimulai tanpa kesalahan dan berfungsi dengan baik.	✓
3	Menggunakan skrip *confirmNetwork.sh* <*Node*-X> untuk memastikan *node* tersebut berfungsi sebagai *validator*.	*Node* yang ditentukan dilaporkan sebagai *validator* aktif oleh semua *node* lainnya dalam jaringan.	✓

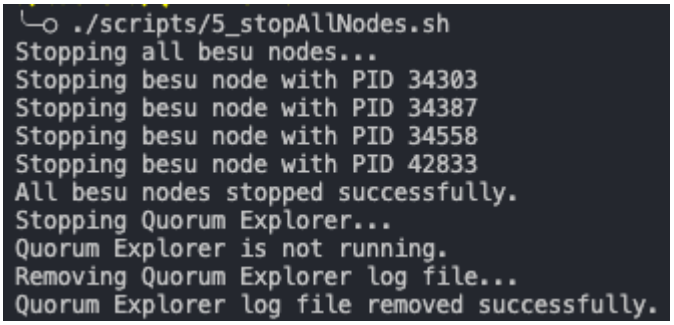


Gambar 5.9 Hasil Pengujian Skrip: *startNodes.sh*

8. Pengujian Skrip *stopAllNodes.sh*:

Tabel 5.28 Pengujian Skrip: *stopAllNodes.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *stopAllNodes.sh*.	Skrip berjalan tanpa error dan menampilkan pesan bahwa semua proses *node Besu* sedang dihentikan.	✓

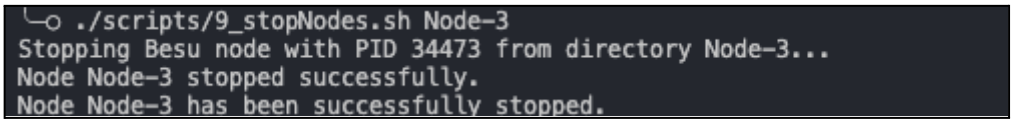


Gambar 5.10 Hasil Pengujian Skrip: *stopAllNodes.sh*

9. Pengujian Skrip *stopNodes.sh*:

Tabel 5.29 Pengujian Skrip: *stopNodes.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *stopNodes.sh* <*Node*-X>, dengan <*Node*-X> adalah *node* yang ingin dihentikan.	Skrip berjalan tanpa error dan menampilkan pesan bahwa *node* yang ditentukan sedang dihentikan.	✓
2	Menggunakan skrip *confirmAllNetwork.sh* untuk memastikan bahwa jaringan tetap mencapai konsensus meskipun ada satu *node* yang dihentikan.	Semua *node* yang tersisa melaporkan jumlah *validator* yang sesuai dan jaringan tetap mencapai konsensus.	✓



Gambar 5.11 Hasil Pengujian Skrip: *stopNodes.sh*

10. Pengujian Skrip *confirmNetwork.sh*:

Tabel 5.30 Pengujian Skrip: *confirmNetwork.sh*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menjalankan skrip *confirmNetwork.sh* <*Node*-X>, dengan <*Node*-X> adalah *node* yang ingin dicek (misalnya, Node-2).	Skrip berjalan tanpa error dan menampilkan pesan bahwa sedang memeriksa status *node* yang ditentukan.	✓
2	Memeriksa *output* skrip untuk respons dari *API JSON-RPC node* yang ditentukan.	*Output* skrip menampilkan respons *JSON-RPC* dari *node* yang berisi daftar *validator* yang terhubung dengan *node* tersebut.	✓
3	Verifikasi bahwa daftar *validator* mencakup *node-node validator* lainnya dalam jaringan, termasuk *node* itu sendiri.	Daftar *validator* mencakup alamat *node-node validator* lainnya dalam jaringan, termasuk *node* yang sedang dicek.	✓
4	Memeriksa pesan ringkasan dari skrip.	Jika *node* memiliki setidaknya empat *validator* yang terhubung, skrip menampilkan pesan "The *private network is working correctly with at least four validators for* <*Node*-X>." Jika tidak, skrip menampilkan pesan kesalahan "The *private network does not have four validators for* <*Node*-X>."	✓

```
└o ./scripts/10_confirmNetwork.sh Node-2
Response from JSON-RPC API for Node-2 (http://localhost:8546):
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": [
    "0x8047d55d8ed2088ec86a6a6619ade3d681151000",
    "0xb64aa75f4ff95f2bffbdfd11437e32835fc9f61d",
    "0xc09d131d0e096eb76bf229fd8b201c59fef00b04",
    "0xf6bdc014d9d5821338994d06a9e91ef4f3252509"
  ]
}
Number of validators for Node-2: 4
The private network is working correctly with at least four validators
 for Node-2.
```

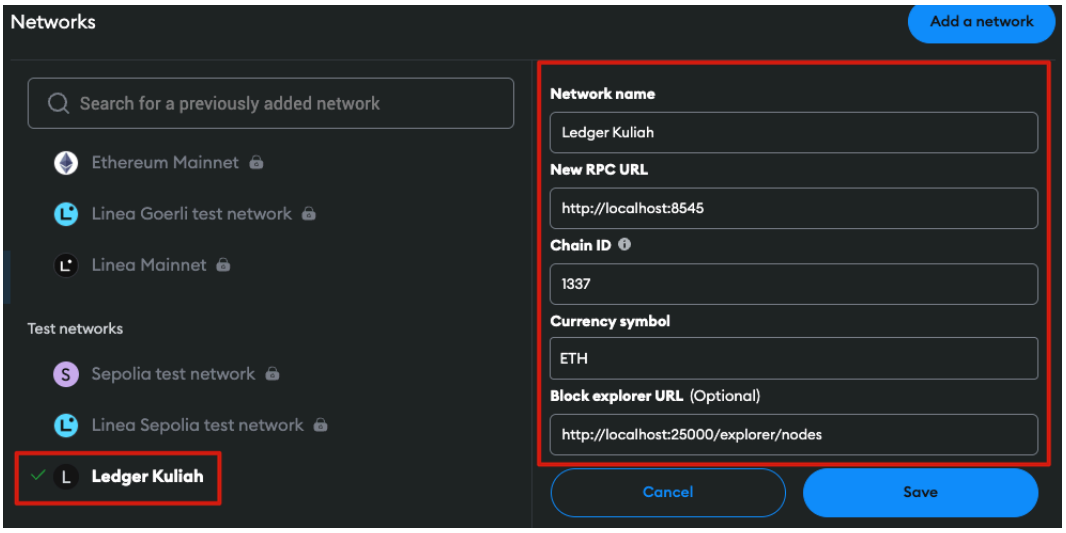
Gambar 5.12 Hasil Pengujian Skrip: *confirmNetwork.sh*

11. Pengujian Menambahkan Jaringan di *MetaMask*:

Tabel 5.31 Pengujian Menambahkan Jaringan di *MetaMask*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Membuka *MetaMask* dan memilih "*Add Network*" atau "*Custom RPC*".	*MetaMask* menampilkan *form* untuk memasukkan detail jaringan baru.	✓
2	Memasukkan detail jaringan:	*Network Name*: LedgerKu (atau nama lain yang dipilih)	✓
	- *New RPC URL*: *http://localhost:8545* (atau alamat RPC node yang lain)	*MetaMask* menerima dan menampilkan detail jaringan yang dimasukkan dengan benar.	✓
	- *Chain ID*: 1337 (atau Chain ID yang digunakan)		✓
	- *Currency* Symbol: *ETH* (atau simbol mata uang kripto lain)		✓
	- *Block Explorer URL*: (Opsional, misalnya http://localhost:25000/explorer/nodes)		✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
3	Menyimpan konfigurasi jaringan.	Jaringan berhasil disimpan dan ditambahkan ke daftar jaringan di *MetaMask*.	✓
4	Memastikan *MetaMask* terhubung ke jaringan yang baru saja ditambahkan.	*MetaMask* menampilkan nama jaringan yang baru ditambahkan di bagian atas antarmuka.	✓
5	Melakukan beberapa transaksi sederhana di jaringan melalui *MetaMask*.	Transaksi berhasil dikirim, diterima, dan ditampilkan dengan benar di *MetaMask* tanpa masalah.	✓



Gambar 5.13 Hasil Pengujian Menambahkan Jaringan di *MetaMask*

5.3.2 Pengujian *Smart Contract*

Pengujian *smart contract* dilakukan menggunakan framework pengujian seperti *Hardhat*. Langkah-langkah ini bertujuan untuk memastikan bahwa setiap fungsi dalam *smart contract* berfungsi dengan baik dan aman sebelum di-deploy ke jaringan *blockchain* yang sebenarnya. Pengujian dilakukan dengan menggunakan skrip pengujian yang telah dibuat dalam *JavaScript*. Berikut adalah tabel skenario pengujian, hasil yang diharapkan, dan hasil pengujian:

1. Pengujian *Deploy Smart Contract*

Tabel 5.32 Pengujian *Deploy Smart Contract*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mengimpor ethers dari *Hardhat* dan menginisialisasi kontrak SKKM.	Kontrak SKKM dapat diinisialisasi tanpa *error*.	✓
2	Mendapatkan instance dari SKKM dan men-*deploy*-nya menggunakan *owner*.	Kontrak berhasil di-*deploy* dan alamat *superAdmin* adalah alamat *owner*.	✓
3	Memastikan bahwa alamat *superAdmin* adalah alamat penginisialisasi.	*superAdmin* sama dengan alamat *owner*.	✓
4	Mendapatkan *receipt* dari transaksi *deploy*.	*Receipt* transaksi berhasil diperoleh tanpa error.	✓
5	Mendapatkan *hash* transaksi dari *receipt*.	*Hash* transaksi tercetak dengan benar di konsol.	✓
6	Mendapatkan nomor blok dari *receipt* transaksi *deploy*.	Nomor blok transaksi *deploy* tercetak dengan benar di konsol.	✓

```
└─○ npx hardhat test test/1_deploy-contract.test.js


  Inisialisasi Kontrak SKKM
Hash transaksi: 0x9efd004b8760fbd96c7c89ad13d85451a55227c50b0c03269808c3061f
6c42c8
Nomor blok: 1
    ✔ Smart Contract sudah berhasil di deploy dengan benar


  1 passing (394ms)
```

Gambar 5.14 Hasil Pengujian *Deploy Smart Contract*

2. Pengujian Menambah Pengguna

Tabel 5.33 Pengujian Menambah Pengguna

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menambahkan pengguna dengan peran *Student*, HMJ, *Admin*, dan BEM.	Pengguna berhasil ditambahkan dengan peran masing-masing tanpa error. Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓
2	Menambahkan pengguna yang sudah ada.	Transaksi gagal dengan pesan error "*User already exists*".	✓
3	Menambahkan pengguna dengan identifier yang sudah digunakan.	Transaksi gagal dengan pesan error "*Identifier already exists*".	✓

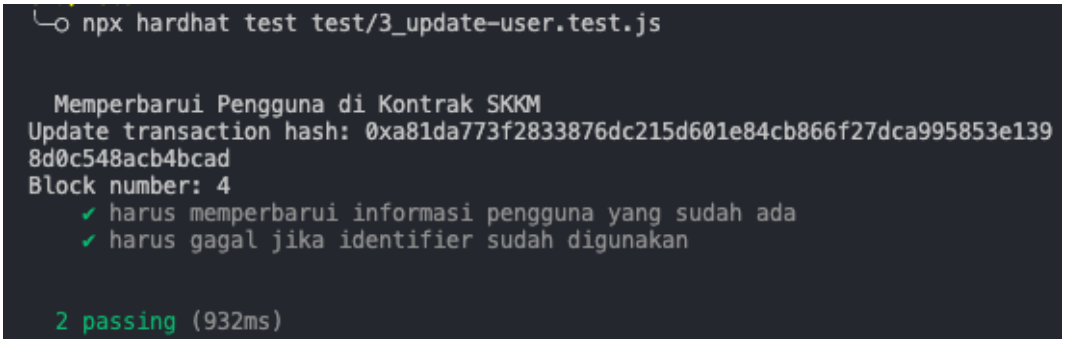
```
└─○ npx hardhat test test/2_add-user.test.js


  Menambahkan Pengguna ke Kontrak SKKM
Student added with transaction hash: 0xd3c71f067896d6b1b78a158e3c5a2e67892a7
d860baf93a6f39f5ff7c3c079a9
Block number: 2
HMJ added with transaction hash: 0xb7b2636f08984611096a1c8703440e6cb0373cba1
a053f8e8c472e35e6a4d2be
Block number: 3
Admin added with transaction hash: 0x869d5a8c22bd96eaf67776285c8232182b8d49f
b6fbe99b4e0836ad64b26335b
Block number: 4
BEM added with transaction hash: 0x6a1768bef8d0ea6ade93cfa5ee2f56c4e3ecfce4d
49c5ffc7322918aa6efb1ea
Block number: 5
    ✔ harus menambahkan pengguna dengan peran Student, HMJ, Admin, dan BEM (
40ms)
    ✔ harus gagal jika pengguna sudah ada
    ✔ harus gagal jika identifier sudah digunakan


  3 passing (954ms)
```

Gambar 5.15 Hasil Pengujian Menambah Pengguna

3. Pengujian Memperbarui Pengguna

Tabel 5.34 Pengujian Memperbarui Pengguna

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Memperbarui informasi pengguna yang sudah ada.	Informasi pengguna berhasil diperbarui tanpa *error*. Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓
2	Memperbarui informasi pengguna dengan identifier yang sudah digunakan oleh pengguna lain.	Transaksi gagal dengan pesan error "*Identifier already exists*".	✓

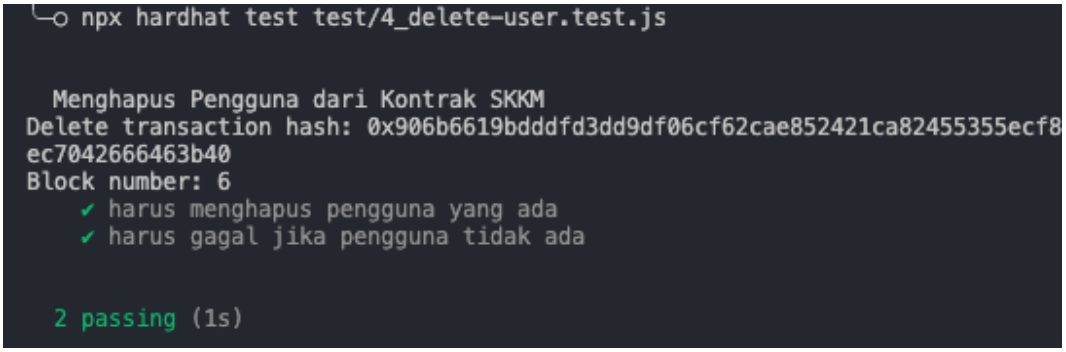


Gambar 5.16 Hasil Pengujian Memperbarui Pengguna

4. Pengujian Menghapus Pengguna

Tabel 5.35 Pengujian Menghapus Pengguna

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menghapus pengguna yang ada.	Pengguna berhasil dihapus tanpa *error*. Transaksi dicatat dengan *hash* dan nomor blok yang valid. Pengguna tidak lagi ada dalam mapping *users* dan *array allUsers*.	✓
2	Menghapus pengguna yang tidak ada.	Transaksi gagal dengan pesan error "*User does not exist*".	✓

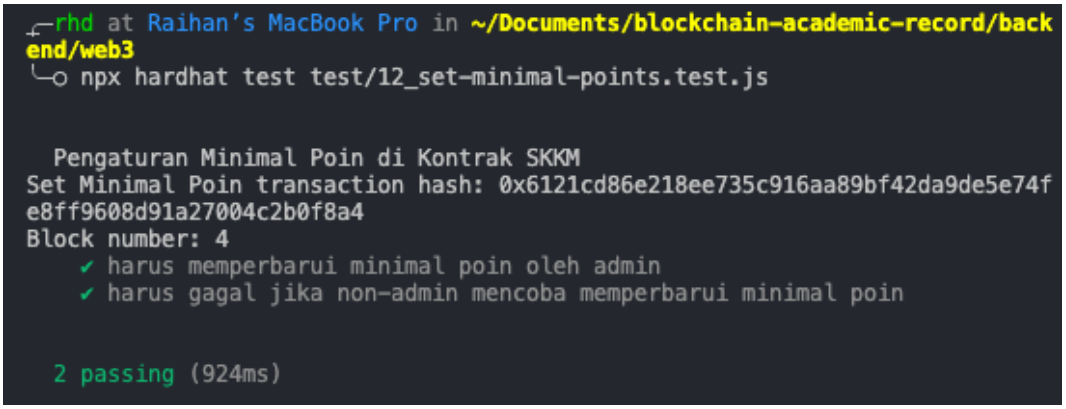


Gambar 5.17 Hasil Pengujian Menghapus Pengguna

5. Pengujian Mengatur Minimal Poin

Tabel 5.36 Pengujian Mengatur Minimal Poin

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	*Admin* memperbarui minimal poin.	Minimal poin berhasil diperbarui dan tercatat di kontrak.	✓
2	*Non-admin* mencoba memperbarui minimal poin.	Gagal memperbarui minimal poin dengan pesan kesalahan "*Only admin can perform this action*".	✓

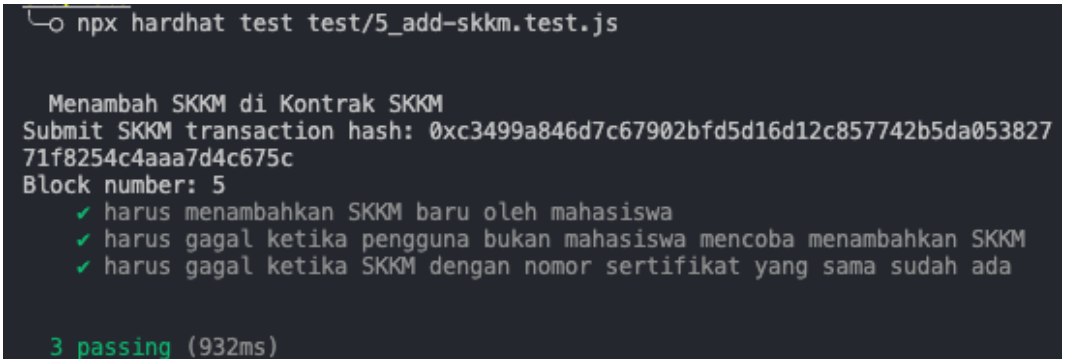


Gambar 5.18 Hasil Pengujian Mengatur Minimal Poin

6. Pengujian Menambah SKKM

Tabel 5.37 Pengujian Mengajukan SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mahasiswa menambahkan SKKM baru.	SKKM berhasil ditambahkan oleh mahasiswa. Transaksi dicatat dengan *hash* dan nomor blok yang valid. SKKM yang ditambahkan memiliki atribut yang sesuai seperti *student*, *activityNameID*, dan *creditPoin*.	✓
2	Pengguna bukan mahasiswa mencoba menambahkan SKKM.	Transaksi gagal dengan pesan error "*Only students can perform this action*".	✓
3	Menambahkan SKKM dengan nomor sertifikat yang sama.	Transaksi gagal dengan pesan error "*Certificate number already exists*".	✓

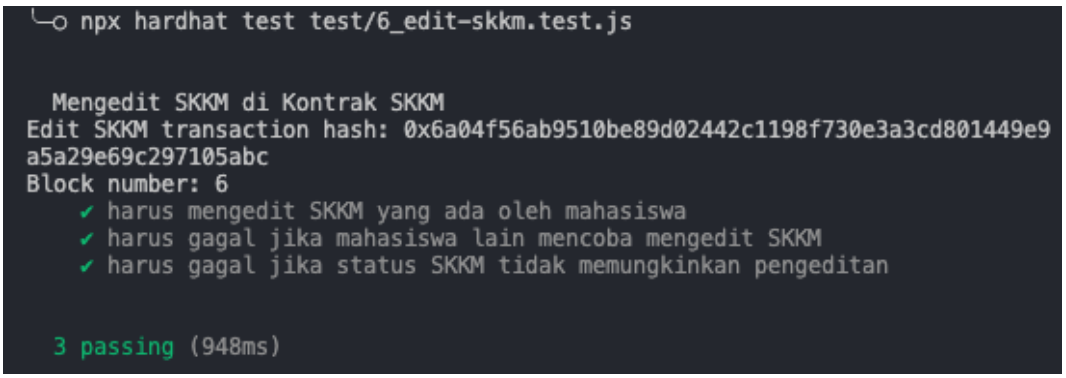


Gambar 5.19 Hasil Pengujian Mengajukan SKKM

7. Pengujian Memperbarui SKKM

Tabel 5.38 Pengujian Memperbarui SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mahasiswa mengedit SKKM yang ada.	SKKM berhasil diedit oleh mahasiswa yang membuatnya. Transaksi dicatat dengan *hash* dan nomor blok yang valid. Atribut SKKM yang diperbarui seperti *activityNameID* dan *creditPoin* sesuai dengan nilai baru yang diberikan.	✓
2	Mahasiswa lain mencoba mengedit SKKM.	Transaksi gagal dengan pesan *error* "*Unauthorized*".	✓
3	Mahasiswa mengedit SKKM yang telah diverifikasi.	Transaksi gagal dengan pesan *error* "*Cannot edit request after verification*".	✓

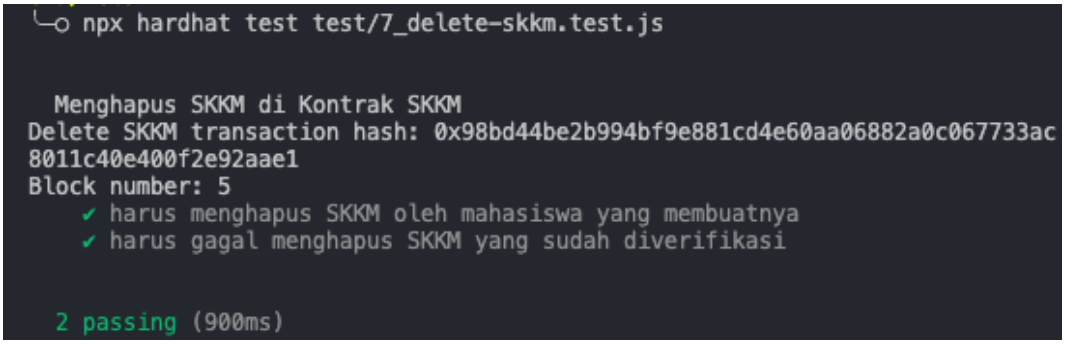


Gambar 5.20 Hasil Pengujian Memperbarui SKKM

8. Pengujian Menghapus SKKM

Tabel 5.39 Pengujian Menghapus SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mahasiswa yang membuat SKKM menghapus SKKM tersebut.	SKKM berhasil dihapus oleh mahasiswa yang membuatnya. Jumlah SKKM berkurang satu dan SKKM yang dihapus tidak lagi ditemukan dalam daftar SKKM. Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓
2	Mahasiswa mencoba menghapus SKKM yang sudah diverifikasi.	Transaksi gagal dengan pesan error "*Cannot delete request after verification*".	✓



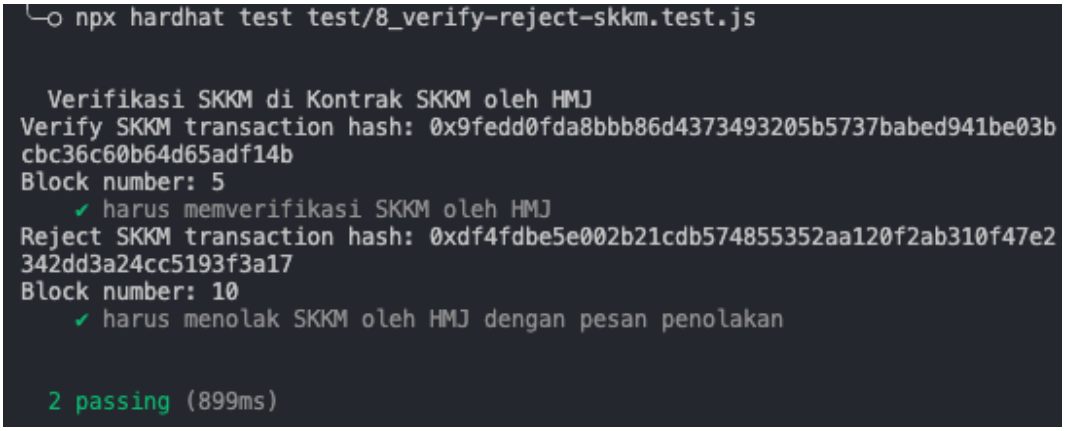
Gambar 5.21 Hasil Pengujian Menghapus SKKM

9. Pengujian Verifikasi dan Unverifikasi SKKM

Tabel 5.40 Pengujian Verifikasi dan Unverifikasi SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	HMJ memverifikasi SKKM yang diajukan oleh mahasiswa.	SKKM berhasil diverifikasi oleh HMJ. Status SKKM berubah menjadi "*Verified*". Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
2	HMJ menolak SKKM yang diajukan oleh mahasiswa dengan pesan penolakan.	SKKM berhasil ditolak oleh HMJ. Status SKKM berubah menjadi "*Unverified*" dan pesan penolakan disimpan dengan benar. Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓



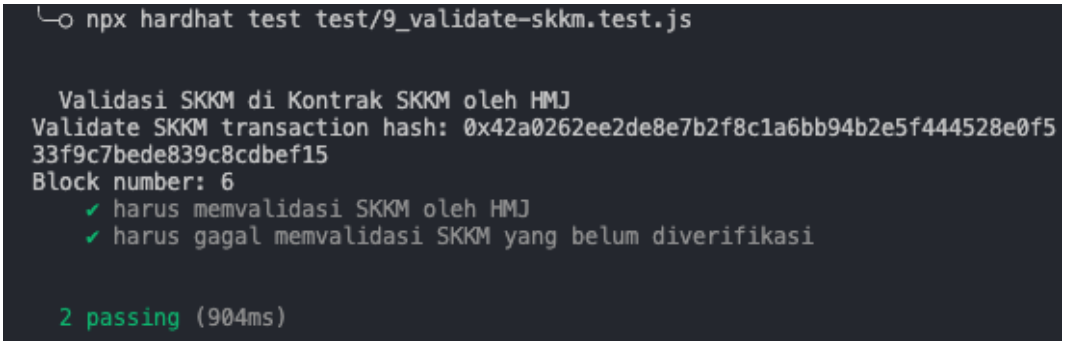
Gambar 5.22 Hasil Pengujian Verifikasi dan Unverifikasi SKKM

10. Pengujian Validasi SKKM

Tabel 5.41 Pengujian Validasi SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	HMJ memvalidasi SKKM yang telah diverifikasi.	SKKM berhasil divalidasi oleh HMJ. Status SKKM berubah menjadi "Valid". Kategori, tipe, pencapaian, penilaian, dan poin kredit yang divalidasi disimpan dengan benar. Transaksi dicatat dengan *hash* dan nomor blok yang valid.	✓
2	HMJ mencoba memvalidasi SKKM yang belum diverifikasi.	Validasi SKKM gagal dan mengembalikan pesan kesalahan "*Request not verified by* HMJ". Tidak ada perubahan pada status	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
		SKKM.	

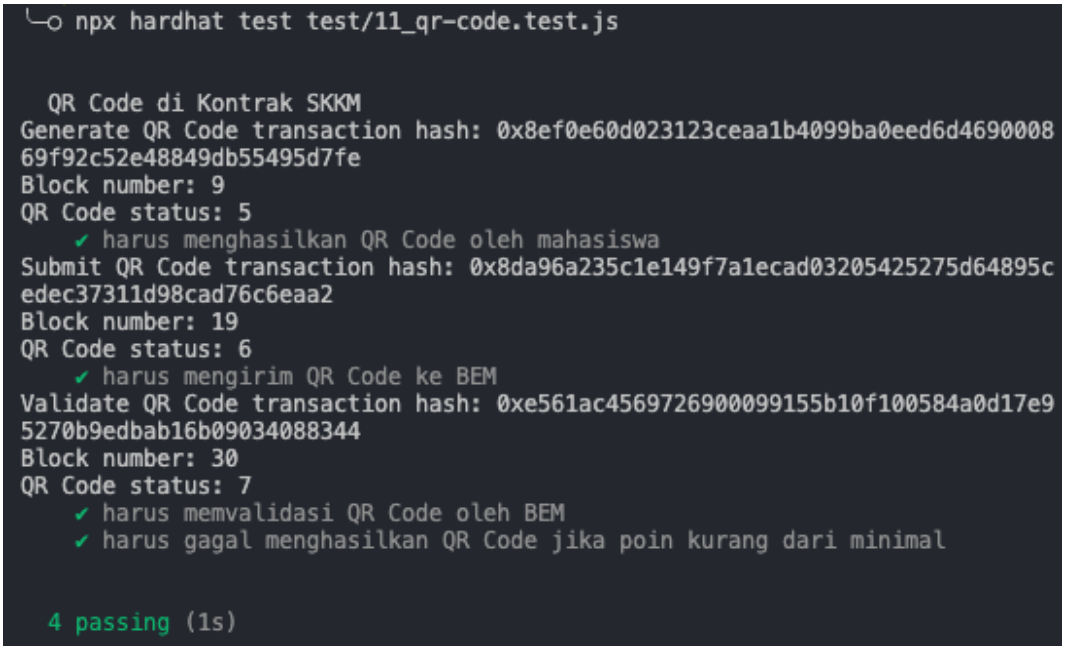


Gambar 5.23 Hasil Pengujian Validasi SKKM

11. Pengujian *QR Code*

Tabel 5.42 Pengujian *QR Code*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mahasiswa menghasilkan *QR Code* setelah memenuhi poin minimal.	*QR Code* berhasil dihasilkan dengan status "*Generated*".	✓
2	Mahasiswa mengirim *QR Code* ke BEM.	*QR Code* berhasil dikirim ke BEM dengan status "*SendToBEM*".	✓
3	BEM memvalidasi *QR Code*.	*QR Code* berhasil divalidasi oleh BEM dengan status "*ValidatedByBEM*".	✓
4	Mahasiswa mencoba menghasilkan *QR Code* tanpa memenuhi poin minimal.	Gagal menghasilkan *QR Code* dengan pesan kesalahan "Minimal poin *not reached*".	✓

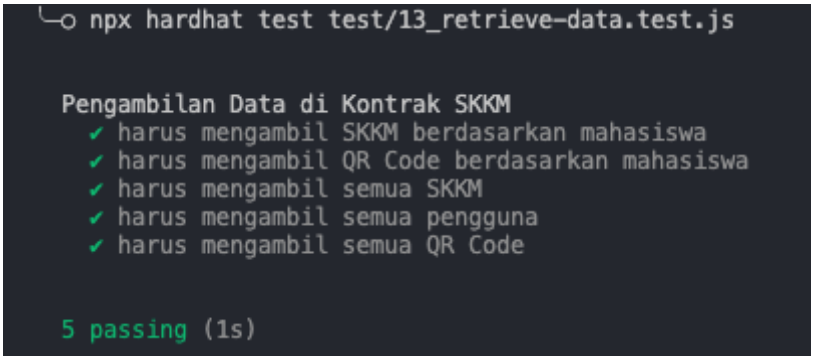


Gambar 5.24 Hasil Pengujian *QR Code*

12. Pengujian Mengambil Data

Tabel 5.43 Pengujian Mengambil Data

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Mengambil SKKM berdasarkan mahasiswa.	Data SKKM untuk setiap mahasiswa diambil dengan benar.	✓
2	Mengambil *QR Code* berdasarkan mahasiswa.	*QR Code* yang dihasilkan oleh mahasiswa diambil dengan benar.	✓
3	Mengambil semua SKKM.	Semua data SKKM yang ada di kontrak diambil dengan benar.	✓
4	Mengambil semua pengguna.	Semua data pengguna yang ada di kontrak diambil dengan benar.	✓
5	Mengambil semua *QR Code*.	Semua *QR Code* yang dihasilkan diambil dengan benar.	✓

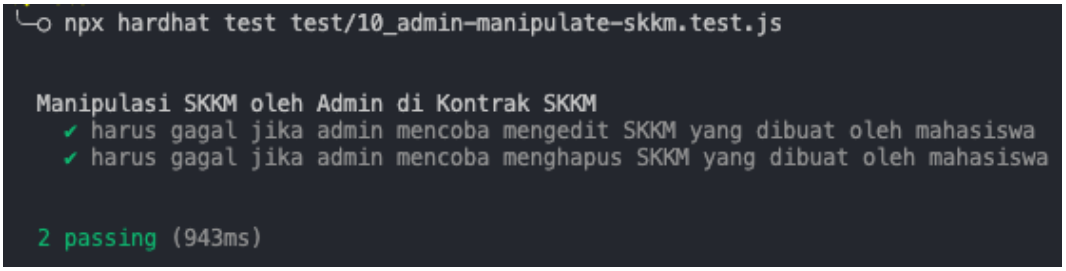


Gambar 5.25 Hasil Pengujian Mengambil Data

13. Pengujian Integritas Data: *Admin* Memanipulasi SKKM

Tabel 5.44 Pengujian Integritas Data: *Admin* Memanipulasi SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	*Admin* mencoba mengedit SKKM yang dibuat oleh mahasiswa.	Edit SKKM oleh *admin* gagal dan mengembalikan pesan kesalahan "*Only students can perform this action*". Tidak ada perubahan pada SKKM.	✓
2	*Admin* mencoba menghapus SKKM yang dibuat oleh mahasiswa.	Hapus SKKM oleh admin gagal dan mengembalikan pesan kesalahan "*Only students can perform this action*". Tidak ada SKKM yang dihapus.	✓



Gambar 5.26 Hasil Pengujian Integritas Data: *Admin* Memanipulasi SKKM

14. Pengujian Integritas Data: *Super Admin* Memanipulasi SKKM

Tabel 5.45 Pengujian Integritas Data: *Super Admin* Memanipulasi SKKM

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Menambahkan pengguna dengan peran mahasiswa.	Pengguna berhasil ditambahkan dengan informasi yang benar.	✓
2	Mengirim data SKKM oleh mahasiswa.	Data SKKM berhasil dikirim dan *hash* transaksi serta *hash* blok dapat diperoleh.	✓
3	Verifikasi konsistensi *hash* data sebelum dan setelah upaya manipulasi.	*Hash* data tetap konsisten, menunjukkan bahwa data tidak dimanipulasi oleh pihak yang tidak berwenang.	✓

Gambar 5.27 Hasil Pengujian Integritas Data: *Super Admin* Memanipulasi SKKM

5.3.3 Pengujian Frontend

Tabel 5.46 Pengujian Frontend

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	Pengujian Fungsionalitas: Memastikan semua komponen dan halaman berfungsi sesuai dengan yang diharapkan.	Semua komponen dan halaman berfungsi tanpa ada *error* atau *bug*. Semua fitur dapat dijalankan dengan baik.	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
2	Pengujian Integrasi: Memastikan *frontend* dapat berkomunikasi dengan *backend* dan *smart contract* dengan baik.	Komunikasi antara *frontend*, *backend*, dan *smart contract* berjalan lancar. Data dari input pengguna di *frontend* berhasil disimpan ke dalam *smart contract* di *blockchain* dan dapat diambil kembali untuk ditampilkan di *frontend*.	✓
3	Pengujian Antarmuka Pengguna: Memastikan bahwa antarmuka pengguna mudah digunakan dan navigasi aplikasi berjalan dengan baik tanpa ada kendala desain yang berarti.	Antarmuka pengguna mudah digunakan dan navigasi aplikasi berjalan dengan baik. Tata letak halaman dan navigasi diuji dan pengguna dapat menggunakan aplikasi dengan nyaman tanpa kebingungan atau masalah.	✓

5.3.4 Pengujian Waktu Respons

Pengujian waktu respons bertujuan untuk mengukur efisiensi sistem dalam menangani transaksi dari sudut pandang pengguna dan dari sisi *backend*. Pengujian ini dibagi menjadi dua bagian utama, yaitu pengujian waktu respons di *frontend* dan pengujian waktu respons transaksi di *blockchain*. Pengujian ini penting untuk memastikan bahwa sistem dapat merespons dengan cepat dan efisien, sehingga pengalaman pengguna tetap optimal dan integritas transaksi tetap terjaga.

5.3.4.1 Pengujian Waktu Respons *Frontend*

Pengujian waktu respons *frontend* dilakukan untuk mengukur waktu yang dibutuhkan dari saat pengguna melakukan tindakan, seperti menekan tombol, hingga transaksi dimulai dan masuk ke dalam jaringan *blockchain*. Waktu respons ini mencakup waktu yang diperlukan untuk menampilkan antarmuka pengguna, seperti *MetaMask*, hingga konfirmasi bahwa transaksi telah dimulai. Metode pengujian yang digunakan adalah dengan melakukan beberapa kali transaksi

untuk setiap fungsi dan mencatat waktu yang diperlukan dalam detik. Berikut adalah tabel hasil pengujian waktu respons *frontend*:

Tabel 5.47 Pengujian Waktu Respons *Frontend*

No	Fungsi	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	Menambah Pengguna	11.70	11.11	7.61	23.30	23.07
2	Mengedit Pengguna	11.32	11.24	11.28	23.16	23.08
3	Menghapus Pengguna	23.39	11.15	11.22	15.85	11.30
4	Mengatur Poin Minimal	23.19	11.02	11.10	15.26	23.05
5	Menambah SKKM	26.00	26.36	27.42	25.81	25.25
6	Mengedit SKKM	3.19	3.92	3.02	3.07	2.97
7	Menghapus SKKM	23.33	11.22	15.26	11.19	11.44
8	Verifikasi SKKM	11.19	11.18	11.39	11.20	15.41
9	Unverifikasi SKKM	23.86	23.25	23.35	7.20	7.28
10	Validasi SKKM	23.17	11.37	15.11	11.12	15.07
11	Generate *QR Code*	23.44	23.46	11.22	7.21	23.54
12	Send *QR Code*	11.61	22.84	8.60	7.21	23.54
13	*Validate QR Code*	11.20	7.20	7.24	11.20	7.12

5.3.4.2 Pengujian Waktu Respons Transaksi *Blockchain*

Pengujian waktu respons transaksi *blockchain* dilakukan untuk mengukur waktu yang dibutuhkan dari saat transaksi diterima oleh jaringan *blockchain*

hingga transaksi tersebut diselesaikan. Waktu respons ini mencakup waktu yang diperlukan untuk memvalidasi dan mencatat transaksi di *blockchain*. Metode pengujian yang digunakan adalah dengan melakukan transaksi secara langsung di *backend* menggunakan framework seperti Hardhat, dan mencatat waktu yang diperlukan untuk setiap transaksi dalam detik. Hasil pengujian ini memberikan gambaran mengenai efisiensi sistem dalam memproses transaksi di sisi *backend*. Berikut adalah tabel hasil pengujian waktu respons transaksi *blockchain*:

Tabel 5.48 Pengujian Waktu Respons Transaksi *Blockchain*

No	Fungsi	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
1	Deploy *Smart Contract*	4.058	8.038	4.039	4.039	4.038
2	Menambah Pengguna	4.052	4.04	8.031	4.036	4.046
3	Mengedit Pengguna	8.025	8.028	4.038	4.046	4.04
4	Menghapus Pengguna	8.031	4.029	8.037	4.036	4.031
5	Mengatur Poin Minimal	8.03	4.028	4.03	4.035	8.037
6	Menambah SKKM	8.031	4.043	4.032	4.04	8.032
7	Mengedit SKKM	4.036	4.038	8.037	4.035	4.039
8	Menghapus SKKM	4.035	8.027	4.027	4.03	4.031
9	Verifikasi SKKM	4.032	4.03	4.028	8.034	4.038
10	Unverifikasi SKKM	8.051	4.033	4.031	4.036	8.036

No	Fungsi	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
11	Validasi SKKM	4.03	4.027	4.034	8.037	4.038
12	Membuat *QR Code*	4.039	8.036	4.039	4.045	4.044
13	Mengirim *QR Code* ke BEM	8.035	4.026	4.028	4.034	4.032
14	Validasi *QR Code*	4.037	8.029	4.03	4.036	4.024

BAB VI. HASIL DAN PEMBAHASAN

6.1 Hasil Penelitian

Penelitian ini berhasil mengembangkan sebuah sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis *blockchain* di Politeknik Negeri Malang (POLINEMA) dengan mengimplementasikan *private blockchain network* menggunakan *Hyperledger Besu*, *smart contract* yang ditulis dalam *Solidity*, aplikasi web dengan *ReactJS*, dan penyimpanan *file* terdesentralisasi menggunakan *IPFS*. Hasil penelitian ini menunjukkan bahwa sistem yang dikembangkan mampu mengatasi beberapa masalah yang ada pada sistem pencatatan SKKM sebelumnya, yaitu:

1. Peningkatan Keamanan dan Integritas Data: Teknologi *blockchain* menjamin integritas data SKKM karena setiap transaksi yang tercatat tidak dapat diubah atau dimanipulasi. Selain itu, penggunaan *private blockchain network* dan mekanisme autentikasi yang kuat melindungi data SKKM dari akses yang tidak sah, sehingga meningkatkan keamanan dan kepercayaan terhadap data serta mengurangi risiko kecurangan.
2. Transparansi yang Lebih Baik: Semua pihak yang berwenang (mahasiswa, HMJ, BEM, dan *admin*) dapat melihat dan memverifikasi data SKKM yang tersimpan di *blockchain*, meningkatkan transparansi proses.
3. Efisiensi Proses: Otomatisasi proses verifikasi dan validasi SKKM melalui *smart contract* mengurangi ketergantungan pada proses manual, mempercepat waktu proses, dan mengurangi beban kerja administrasi.
4. Pengurangan Biaya Operasional: Dengan mengurangi kebutuhan akan dokumen fisik dan proses verifikasi manual, sistem ini dapat membantu mengurangi biaya operasional yang terkait dengan pencatatan SKKM.

6.1.1 Hasil Pengujian Jaringan *Blockchain*

Berikut adalah hasil pengujian jaringan *blockchain* yang dilakukan untuk memastikan bahwa skrip-skrip yang dibuat untuk mengelola *node-node* dalam jaringan berjalan dengan baik:

Tabel 6.1 Hasil Pengujian Jaringan *Blockchain*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	*generateNodeKeys.sh*	Skrip berhasil menghasilkan file konfigurasi *genesis.json* dan kunci kriptografi untuk setiap *node* dalam jaringan. Direktori *network files* berisi *file genesis.json* dan subdirektori *keys* dengan pasangan kunci untuk setiap *node*.	✓
2	*setupQBFT.sh*	Skrip berhasil menyalin *file genesis.json* dan kunci *node* ke direktori data masing-masing *node*, memastikan setiap *node* memulai dengan konfigurasi yang sama.	✓
3	*startAllNodes.sh*	Skrip berhasil memulai semua *node* dalam jaringan tanpa pesan kesalahan. Log menunjukkan bahwa setiap *node* berhasil bergabung dalam jaringan dan mendengarkan pada *port* yang benar.	✓
4	*confirmAllNetwork.sh*	Skrip memastikan bahwa semua *node* dalam jaringan telah mencapai konsensus dan berfungsi sebagai *validator*.	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
5	*addValidator.sh*	*Node* baru berhasil ditambahkan sebagai *validator* dan diakui oleh semua *node* lain dalam jaringan.	✓
6	*removeValidator.sh*	*Node* yang ditargetkan berhasil dihapus sebagai *validator*, dan jaringan melaporkan jumlah *validator* yang sesuai setelah penghapusan.	✓
7	*startNodes.sh*	*Node* yang ditentukan berhasil dimulai secara individual dan bergabung dengan jaringan sebagai *validator*.	✓
8	*stopAllNodes.sh*	Semua proses *node Besu* berhasil dihentikan.	✓
9	*stopNodes.sh*	*Node* yang ditentukan berhasil dihentikan, dan jaringan tetap mencapai konsensus dengan *node* yang tersisa.	✓
10	*confirmNetwork.sh*	Skrip menampilkan status dan daftar *validator* dari *node* yang ditentukan.	✓
11	Penambahan Jaringan ke *MetaMask*	Jaringan *blockchain* lokal berhasil ditambahkan ke *MetaMask*, dan transaksi dapat dilakukan tanpa masalah.	✓

Dengan hasil pengujian ini, dapat disimpulkan bahwa skrip-skrip untuk mengelola *node-node* dalam jaringan *blockchain* berjalan dengan baik dan sesuai dengan yang diharapkan.

6.1.2 Hasil Pengujian *Smart Contract*

Berikut adalah hasil pengujian *smart contract* dilakukan menggunakan *framework Hardhat*, dengan hasil sebagai berikut:

Tabel 6.2 Hasil Pengujian *Smart Contract*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
1	*Deploy Contract*	*Smart contract* berhasil di-*deploy* dengan benar, alamat *superAdmin* sesuai dengan alamat penginisialisasi, dan kontrak berhasil diinisialisasi tanpa kesalahan.	✓
2	Menambah Pengguna	Pengguna berhasil ditambahkan dengan peran (*Student*, HMJ, *Admin*, BEM), validasi memastikan tidak ada duplikasi pengguna atau *identifier*, dan transaksi penambahan pengguna tercatat di *blockchain*.	✓
3	Mengedit Pengguna	Informasi pengguna berhasil diperbarui, validasi untuk mencegah duplikasi identifier diterapkan, dan transaksi pembaruan pengguna tercatat di *blockchain*.	✓
4	Menghapus Pengguna	Pengguna yang ada berhasil dihapus, validasi memastikan pengguna yang sudah dihapus tidak ada lagi dalam sistem, dan	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
		transaksi penghapusan pengguna tercatat di *blockchain*.	
5	Menambah SKKM	Mahasiswa berhasil menambahkan SKKM baru, validasi untuk mencegah duplikasi nomor sertifikat diterapkan, dan transaksi penambahan SKKM tercatat di *blockchain*.	✓
6	Mengedit SKKM	SKKM yang ada berhasil diperbarui oleh mahasiswa yang membuatnya, validasi untuk mencegah perubahan oleh mahasiswa lain diterapkan, dan transaksi pembaruan SKKM tercatat di *blockchain*.	✓
7	Menghapus SKKM	Mahasiswa berhasil menghapus SKKM yang mereka buat, validasi untuk mencegah penghapusan SKKM yang sudah diverifikasi diterapkan, dan transaksi penghapusan SKKM tercatat di *blockchain*.	✓
8	Verifikasi SKKM	HMJ berhasil memverifikasi SKKM, opsi untuk menolak SKKM dengan pesan penolakan tersedia, dan transaksi verifikasi SKKM tercatat di *blockchain*.	✓
9	Validasi SKKM	HMJ berhasil memvalidasi SKKM yang sudah diverifikasi, dan transaksi validasi SKKM tercatat di	✓

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian
		blockchain.	
10	*QR Code*	Mahasiswa berhasil menghasilkan *QR Code*, *QR Code* berhasil dikirim ke BEM, dan BEM berhasil memvalidasi *QR Code*, serta semua transaksi terkait tercatat di *blockchain*.	✓
11	Pengaturan Minimal Poin	*Admin* berhasil memperbarui minimal poin yang diperlukan untuk menghasilkan *QR Code*, dan perubahan tercatat di *blockchain*.	✓
12	Pengambilan Data	Data SKKM, *QR Code*, dan pengguna berhasil diambil dari *blockchain* sesuai dengan yang diharapkan, memastikan data dapat diakses dan diverifikasi oleh pihak yang berwenang.	✓
13	Integritas Data: Manipulasi SKKM oleh *Admin*	*Admin* tidak dapat mengedit atau menghapus SKKM yang dibuat oleh mahasiswa, dan upaya manipulasi dicegah oleh sistem.	✓
14	Integritas Data: Manipulasi SKKM oleh *Super Admin*	Konsistensi data terjaga, upaya manipulasi oleh *super admin* dicegah, dan transaksi yang gagal tercatat di *blockchain*, memastikan integritas data tetap utuh.	✓

Berdasarkan hasil pengujian di atas, dapat disimpulkan bahwa *smart contract* SKKM berhasil diimplementasikan dan diuji dengan baik menggunakan

framework Hardhat. Semua fungsi serta integritas data telah berjalan sesuai dengan yang diharapkan.

6.1.3 Hasil Pengujian Waktu Respons

Pengujian waktu respons dilakukan untuk mengevaluasi kinerja sistem dalam hal kecepatan respon terhadap berbagai fungsi yang ada di *frontend* dan *backend*. Waktu respons diukur untuk memastikan bahwa sistem dapat menangani permintaan pengguna dan memproses transaksi *blockchain* dengan efisien.

6.1.3.1 Hasil Pengujian Waktu Respons *Frontend*

Berikut adalah hasil pengujian waktu respons *frontend* dalam bentuk rata-rata untuk setiap fungsi:

Tabel 6.3 Hasil Pengujian Waktu Respons *Frontend*

No	*Function*	Rata-rata Waktu Respons (detik)
1	Menambah Pengguna	15.76
2	Mengedit Pengguna	16.42
3	Menghapus Pengguna	14.18
4	Mengatur Poin Minimal	16.12
5	Menambah SKKM	26.57
6	Mengedit SKKM	3.23
7	Menghapus SKKM	14.49
8	Verifikasi SKKM	12.47
9	Unverifikasi SKKM	16.59
10	Validasi SKKM	15.57
11	Membuat *QR Code*	17.97
12	Mengirim *QR Code*	14.36
13	Validasi *QR Code*	8.99

Hasil pengujian menunjukkan bahwa fungsi menambah SKKM memiliki waktu respons rata-rata tertinggi sebesar 26.57 detik, sedangkan fungsi mengedit SKKM memiliki waktu respons rata-rata terendah sebesar 3.23 detik. Waktu respons untuk fungsi-fungsi lainnya berkisar antara 8.99 detik hingga 17.97 detik.

Secara keseluruhan, sistem *frontend* menunjukkan performa yang baik dalam menangani transaksi pengguna.

6.1.3.2 Hasil Pengujian Waktu Respons Transaksi *Blockchain*

Tabel 6.4 Hasil Pengujian Waktu Respons Transaksi *Blockchain*

No	*Function*	Rata-rata Waktu Respons (detik)
1	*Deploy Smart Contract*	4.44
2	Menambah Pengguna	4.84
3	Mengedit Pengguna	5.64
4	Menghapus Pengguna	5.23
5	Mengatur Poin Minimal	5.23
6	Menambah SKKM	5.84
7	Mengedit SKKM	4.84
8	Menghapus SKKM	4.83
9	Verifikasi SKKM	4.83
10	Unverifikasi SKKM	5.64
11	Validasi SKKM	4.83
12	Membuat *QR Code*	4.84
13	Mengirim *QR Code* ke BEM	4.83
14	Validasi *QR Code*	4.83

Hasil pengujian menunjukkan bahwa waktu respons rata-rata untuk transaksi *blockchain* cukup konsisten. Fungsi menambah SKKM memiliki waktu respons rata-rata tertinggi sebesar 5.84 detik, sementara *deploy smart contract* memiliki waktu respons rata-rata terendah sebesar 4.44 detik. Sistem *backend* menunjukkan efisiensi yang baik dalam memproses transaksi.

6.2 Pembahasan

Penelitian ini menggunakan teknologi *blockchain* untuk mencatat SKKM, menawarkan transparansi dan keamanan yang lebih baik dibandingkan dengan metode tradisional yang menggunakan sistem terpusat. Solusi berbasis *blockchain* ini menghilangkan kebutuhan akan otoritas pusat dan meningkatkan keandalan data melalui konsensus.

6.2.1 Perbandingan dengan Penelitian Sebelumnya

Penelitian ini memiliki beberapa perbedaan dan persamaan dengan penelitian-penelitian sebelumnya yang telah dilakukan. Berikut adalah perbandingan dengan beberapa penelitian terkait:

1. Fokus pada SKKM: Penelitian ini secara khusus berfokus pada pencatatan SKKM di POLINEMA.
2. Penggunaan *Hyperledger Besu*: Penelitian ini menggunakan *Hyperledger Besu* sebagai platform *blockchain*, yang memungkinkan pembuatan *private network* yang lebih aman dan terkontrol.
3. Penggunaan *IPFS*: Penelitian ini mengintegrasikan *IPFS* untuk penyimpanan bukti keikutsertaan mahasiswa, memberikan keuntungan dalam hal keamanan, integritas data, dan efisiensi waktu, serta mengurangi kebutuhan mahasiswa untuk mencetak dokumen fisik.
4. Pengembangan Aplikasi Web dengan *ReactJS*: Penelitian ini mengembangkan aplikasi web dengan *ReactJS* untuk memberikan antarmuka pengguna yang lebih modern, interaktif, dan mudah digunakan.

Tabel 6.3 Perbandingan Penelitian

No	Judul Penelitian	Persamaan	Perbedaan
1	*A Novel Blockchain-based Education Records Verification Solution* (Han et al., 2018)	Sama-sama menggunakan teknologi *blockchain* untuk menyimpan catatan akademik.	Penelitian ini fokus pada pencatatan SKKM, sedangkan penelitian (Han et al., 2018) lebih umum pada verifikasi catatan pendidikan. Penelitian ini menggunakan *Hyperledger Besu*, sedangkan (Han et al., 2018) tidak menyebutkan platform *blockchain* spesifik.

No	Judul Penelitian	Persamaan	Perbedaan
2	*UniChain: A Design of Blockchain-Based System for Electronic Academic Records Access and Permissions Management* (Daraghmi et al., 2019)	Sama-sama menggunakan *blockchain* untuk mengelola catatan akademik elektronik dan mengatur izin akses.	Penelitian ini fokus pada pencatatan SKKM dan menggunakan *Hyperledger Besu*, sedangkan (Daraghmi et al., 2019) lebih umum pada catatan akademik elektronik dan menggunakan *Ethereum*. Penelitian ini juga mengintegrasikan *IPFS* dan *Thirdweb*, sedangkan (Daraghmi et al., 2019) tidak menyebutkan penggunaan teknologi tersebut.
3	*Blockchain to improve Academic Governance* (Bhagwat et al., 2020)	Sama-sama menggunakan *blockchain* untuk meningkatkan tata kelola akademik, termasuk verifikasi transkrip.	Penelitian ini fokus pada pencatatan SKKM dan menggunakan *Hyperledger Besu*, sedangkan (Bhagwat et al., 2020) lebih umum pada tata kelola akademik dan tidak menyebutkan platform *blockchain* spesifik. Penelitian ini juga tidak secara khusus membahas verifikasi transkrip, melainkan berfokus pada pencatatan dan validasi kegiatan mahasiswa.

No	Judul Penelitian	Persamaan	Perbedaan
4	*Academic Records Verification Platform Based on Blockchain Technology* (Huang, 2020)	Sama-sama mengembangkan platform verifikasi catatan akademik berbasis *blockchain*.	Penelitian ini fokus pada pencatatan SKKM dan menggunakan *Hyperledger Besu*, sedangkan (Huang, 2020) lebih umum pada verifikasi catatan akademik dan tidak menyebutkan platform *blockchain* spesifik. Penelitian ini juga mengintegrasikan *IPFS* untuk penyimpanan bukti keikutsertaan, sedangkan (Huang, 2020) tidak menyebutkan penggunaan *IPFS*.
5	*Revitalizing blockchain technology potentials for smooth academic records management and verification in low-income countries* (Alnafrah & Mouselli, 2021)	Sama-sama membahas potensi teknologi *blockchain* untuk meningkatkan manajemen dan verifikasi catatan akademik.	Penelitian ini fokus pada pencatatan SKKM di POLINEMA, sedangkan (Alnafrah & Mouselli, 2021) lebih umum pada manajemen catatan akademik di negara berpenghasilan rendah. Penelitian ini menggunakan *Hyperledger Besu* dan *IPFS*, sedangkan (Alnafrah & Mouselli, 2021) mengusulkan platform *blockchain* hibrida nasional.

Dengan hasil perbandingan ini, dapat disimpulkan bahwa penelitian ini menawarkan beberapa peningkatan dan inovasi dalam hal transparansi, keamanan, efisiensi, dan kepercayaan dibandingkan dengan penelitian-penelitian sebelumnya.

6.2.2 Perbandingan dengan Sistem SKKM POLINEMA

Sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis *blockchain* yang dikembangkan dalam penelitian ini memiliki perbedaan yang

signifikan dibandingkan dengan sistem SKKM yang saat ini digunakan di Politeknik Negeri Malang (POLINEMA). Perbedaan ini terlihat dalam beberapa aspek, antara lain:

1. Transparansi dan Kepercayaan: Sistem *blockchain* memberikan transparansi yang lebih tinggi karena semua transaksi dan data SKKM tercatat dalam buku besar yang terdistribusi (distributed ledger) dan dapat diverifikasi oleh semua pihak yang berwenang. Dengan transparansi yang tinggi, sistem *blockchain* meningkatkan kepercayaan semua pihak terhadap integritas dan keakuratan data SKKM. Pengguna dapat melihat dan memverifikasi transaksi secara langsung melalui *transaction hash* yang tersedia di *blockchain explorer*, sehingga mengurangi kekhawatiran tentang potensi manipulasi data. Bukti transparansi ini dapat dilihat melalui pencatatan dan validasi *transaction hash* di *block explorer*.
2. Keamanan dan Potensi Manipulasi Data: Sistem *blockchain* menggunakan teknologi kriptografi canggih untuk melindungi data SKKM dari akses yang tidak sah dan manipulasi. Setiap perubahan pada data SKKM akan tercatat sebagai transaksi baru di *blockchain*, sehingga tidak dapat diubah atau dihapus secara diam-diam. Dalam sistem *blockchain*, setiap perubahan pada data dicatat sebagai transaksi baru, membuat potensi manipulasi data sangat rendah. Data tidak bisa diubah tanpa terdeteksi, sehingga integritas data tetap terjaga. Hasil pengujian yang mendukung ini dapat dilihat pada Tabel 5.44 dan Tabel 5.45.
3. Efisiensi dan Otomatisasi: Sistem *blockchain* mengotomatiskan banyak proses dalam pencatatan SKKM, seperti verifikasi dan validasi, menggunakan *smart contract*. Hal ini mengurangi ketergantungan pada proses manual yang memakan waktu dan rentan terhadap kesalahan manusia. Otomatisasi verifikasi dan validasi dengan *smart contract* mengurangi kebutuhan cetak dokumen fisik dan mengurangi tahap validasi manual. Ini tidak hanya mempercepat proses tetapi juga mengurangi kesalahan manusia. Mahasiswa tidak perlu mencetak dokumen fisik untuk keperluan validasi, sehingga lebih efisien. Efisiensi ini diobservasi melalui

implementasi sistem, yang menunjukkan pengurangan beban kerja administrasi dan waktu proses yang lebih cepat.

4. Desentralisasi: Sistem blockchain mendistribusikan data SKKM ke seluruh *node* dalam jaringan, sehingga tidak ada satu titik pusat kegagalan. Setiap *node* dalam jaringan memiliki salinan lengkap dari semua data, memastikan bahwa jika satu *node* mengalami kerusakan atau serangan, *node* lainnya tetap dapat melanjutkan operasi sistem tanpa gangguan. Hal ini meningkatkan ketahanan dan keandalan sistem, serta mengurangi risiko kehilangan data akibat kegagalan *server* tunggal. Implementasi desentralisasi ini mencakup konfigurasi *node-node* dalam jaringan *blockchain* yang saling berkomunikasi untuk menjaga konsistensi data dan mencapai konsensus. Implementasi ini dijelaskan lebih lanjut dalam sub bab 5.1.1 tentang implementasi *private blockchain network*, yang mencakup pengaturan dan distribusi *node* dalam jaringan.
5. Penyimpanan Bukti Keikutsertaan : Sistem blockchain menggunakan *IPFS* (InterPlanetary File System) untuk menyimpan bukti keikutsertaan mahasiswa secara terdistribusi. Alurnya adalah sebagai berikut: gambar bukti keikutsertaan diunggah ke *IPFS*, yang kemudian menghasilkan *hash* unik untuk setiap file. *Hash* ini berfungsi sebagai identifikasi unik yang memastikan bahwa setiap perubahan sekecil apa pun pada gambar akan menghasilkan *hash* yang berbeda. Gambar asli disimpan di *IPFS*, yang menyediakan *storage* terdesentralisasi dengan keamanan tinggi. *Hash* dari gambar ini disimpan di *blockchain*, memastikan bahwa meskipun beberapa *node* mengalami masalah, bukti keikutsertaan tetap dapat diakses melalui *hash* yang tersimpan di *blockchain*. Hal ini meningkatkan keamanan dan integritas data karena data dienkripsi dan didistribusikan di seluruh jaringan. Bukti transparansi ini dapat diobservasi melalui *hash* yang ditampilkan di web *frontend*, di mana pengguna dapat memverifikasi integritas data dengan membandingkan *hash* yang tersimpan di *blockchain* dengan *hash* yang dihasilkan oleh *IPFS*.

Berikut adalah tabel perbandingan antara sistem yang diusulkan dengan sistem SKKM POLINEMA yang ada saat ini:

Tabel 6.5 Perbandingan Sistem SKKM

No	Aspek Sistem	Sistem SKKM POLINEMA (saat ini)	Sistem SKKM Berbasis *Blockchain* (diusulkan)
1	Transparansi dan Kepercayaan	Rendah	Tinggi
2	Keamanan dan Potensi Manipulasi Data	Rendah	Tinggi
3	Efisiensi dan Otomatisasi	Rendah	Tinggi
4	Desentralisasi	Tidak	Ya
5	Penyimpanan Bukti Keikutsertaan	Terpusat	Terdistribusi (IPFS)

Dengan hasil perbandingan ini, dapat disimpulkan bahwa sistem pencatatan SKKM berbasis *blockchain* yang diusulkan menawarkan peningkatan yang signifikan dalam hal transparansi, keamanan, efisiensi, desentralisasi, dan penyimpanan data dibandingkan dengan sistem yang ada saat ini di POLINEMA.

BAB VII. KESIMPULAN DAN SARAN

7.1 Kesimpulan

Penelitian ini berhasil mengembangkan dan mengimplementasikan sebuah sistem pencatatan SKKM (Satuan Kredit Kegiatan Mahasiswa) berbasis blockchain di Politeknik Negeri Malang (POLINEMA). Sistem ini dibangun menggunakan teknologi *blockchain Hyperledger Besu* sebagai *private network*, *smart contract* yang ditulis dalam *Solidity*, aplikasi web dengan *ReactJS*, dan penyimpanan *file* terdesentralisasi menggunakan *InterPlanetary File System* (IPFS).

Hasil penelitian menunjukkan bahwa sistem yang dikembangkan mampu mengatasi berbagai masalah yang ada pada sistem pencatatan SKKM sebelumnya, serta memberikan beberapa peningkatan signifikan, di antaranya:

1. Peningkatan Keamanan dan Integritas Data:
    - Teknologi *blockchain* menjamin integritas data SKKM dengan memastikan setiap transaksi yang tercatat tidak dapat diubah atau dimanipulasi. Sistem menggunakan teknologi kriptografi canggih untuk melindungi data dari akses tidak sah, sehingga meningkatkan keamanan dan mengurangi risiko kecurangan. Implementasi ini berhasil menunjukkan bahwa *blockchain* dapat meningkatkan keamanan data akademik secara signifikan.
2. Transparansi yang Lebih Baik:
    - Semua transaksi tercatat dalam buku besar yang terdistribusi dan dapat diverifikasi oleh semua pihak yang berwenang, yang meningkatkan transparansi proses pencatatan SKKM. Dalam sistem terpusat sebelumnya, akses terbatas menyebabkan potensi manipulasi data yang lebih tinggi. Penggunaan *blockchain* memastikan bahwa transparansi dan keandalan data akademik meningkat.
3. Efisiensi Proses:
    - Otomatisasi proses verifikasi dan validasi SKKM melalui *smart contract* mengurangi ketergantungan pada proses manual, mempercepat waktu proses, dan mengurangi beban kerja

administrasi. Meskipun hasil pengujian waktu respons menunjukkan bahwa waktu rata-rata untuk setiap transaksi dalam sistem *blockchain* lebih lama dibandingkan dengan sistem *database* tradisional karena adanya proses membuka *MetaMask* dan menunggu transaksi sukses, efisiensi keseluruhan dalam mengurangi kesalahan manusia dan kebutuhan akan cetak dokumen fisik tetap tercapai.

4. Desentralisasi:
    - Sistem *blockchain* mendistribusikan data SKKM ke seluruh jaringan, sehingga tidak ada satu titik pusat kegagalan. Hal ini meningkatkan ketahanan sistem terhadap serangan atau kerusakan pada satu *node* karena data dan fungsi jaringan tersebar di banyak *node*. Sistem tetap berfungsi meskipun satu *node* mengalami masalah.
5. Penyimpanan Bukti Keikutsertaan:
    - Bukti keikutsertaan disimpan secara terdistribusi menggunakan *InterPlanetary File System* (IPFS), yang meningkatkan keamanan dan aksesibilitas serta memastikan data tidak hilang atau rusak. Penyimpanan desentralisasi memastikan bahwa data tetap tersedia dan aman meskipun sebagian jaringan mengalami masalah.

Selain itu, aplikasi web yang dikembangkan memberikan antarmuka yang intuitif dan mudah digunakan bagi mahasiswa, HMJ, BEM, dan *admin* untuk berinteraksi dengan sistem. Hasil pengujian keseluruhan menunjukkan bahwa sistem ini berfungsi dengan baik dan memenuhi kebutuhan fungsional serta non-fungsional yang telah ditentukan. Secara keseluruhan, sistem pencatatan SKKM berbasis *blockchain* ini mampu meningkatkan transparansi, keamanan, dan efisiensi dalam proses pencatatan SKKM di POLINEMA.

7.2 Saran

Berdasarkan hasil penelitian ini, terdapat beberapa saran untuk pengembangan lebih lanjut agar sistem pencatatan SKKM berbasis *blockchain* ini dapat lebih optimal:

1. Peningkatan Pengujian Keamanan: Meskipun pengujian keamanan telah dilakukan, disarankan untuk melakukan pengujian keamanan yang lebih komprehensif, termasuk pengujian penetrasi dan audit keamanan independen, untuk memastikan bahwa sistem ini tahan terhadap berbagai ancaman keamanan.
2. Pengembangan Fitur Tambahan: Sistem ini dapat dikembangkan lebih lanjut dengan menambahkan fitur-fitur seperti integrasi dengan sistem akademik yang sudah ada, notifikasi otomatis untuk mahasiswa dan pihak terkait, serta pelaporan dan analisis data SKKM yang lebih lengkap.
3. Evaluasi Jangka Panjang: Melakukan evaluasi jangka panjang terhadap sistem ini untuk memantau kinerja, keamanan, dan efektivitasnya dalam meningkatkan proses pencatatan SKKM di POLINEMA.
4. Sosialisasi dan Pelatihan: Melakukan sosialisasi dan pelatihan yang memadai kepada mahasiswa, HMJ, BEM, dan *admin* tentang cara menggunakan sistem ini secara efektif.
5. Eksplorasi Teknologi *Blockchain* Lainnya: Mengeksplorasi penggunaan teknologi *blockchain* lainnya, seperti *Ethereum* atau *Hyperledger Fabric*, untuk melihat apakah ada platform yang lebih sesuai dengan kebutuhan pencatatan SKKM di POLINEMA.
6. Implementasi di Lingkungan Produksi: Mengingat hasil pengujian yang positif, sistem ini disarankan untuk diimplementasikan di lingkungan produksi POLINEMA setelah melalui tahap pengujian dan evaluasi lebih lanjut. Implementasi ini akan melibatkan migrasi data SKKM yang ada ke sistem baru, pelatihan pengguna, dan pemantauan ketat untuk memastikan kelancaran transisi dan penggunaan sistem secara optimal.

Dengan penerapan saran-saran tersebut, diharapkan sistem pencatatan SKKM berbasis *blockchain* dapat memberikan kontribusi yang lebih besar dalam meningkatkan transparansi, keamanan, dan efisiensi dalam pencatatan dan pengelolaan data akademik di POLINEMA.

DAFTAR PUSTAKA

Alhadhrami, Z., Alghfeli, S., Alghfeli, M., Abedlla, J. A., & Shuaib, K. (2017). Introducing blockchains for healthcare. *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 1–4. https://doi.org/10.1109/ICECTA.2017.8252043

Alnafrah, I., & Mouselli, S. (2021). Revitalizing *blockchain* technology potentials for smooth academic records management and verification in low-income countries. *International Journal of Educational Development*, *85*, 102460. https://doi.org/10.1016/j.ijedudev.2021.102460

Amo-Filva, D., Escudero, D. F., Sanchez-Sepulveda, M. V., García-Holgado, A., García-Holgado, L., García-Peñalvo, F. J., Orehovački, T., Krašna, M., Pesek, I., Marchetti, E., Valente, A., Witfelt, C., Ružić, I., Fraoua, K. E., & Moreira, F. (2023). Security and Privacy in Academic Data Management at Schools: SPADATAS Project. In P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies* (pp. 3–16). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-34411-4_1

Announcing Hyperledger Besu – Hyperledger Foundation. (2019, August 29). https://www.hyperledger.org/blog/2019/08/29/announcing-hyperledger-besu

Arenas, R., & Fernandez, P. (2018). CredenceLedger: A Permissioned *Blockchain* for Verifiable Academic Credentials. *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 1–6. https://doi.org/10.1109/ICE.2018.8436324

Badr, A., Rafferty, L., Mahmoud, Q. H., Elgazzar, K., & Hung, P. C. K. (2019). A

Permissioned *Blockchain*-Based System for Verification of Academic Records. *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 1–5. https://doi.org/10.1109/NTMS.2019.8763831

Bari, H., & Patel, N. (2023). Generalized Immutable Ledger (GILED) using *Blockchain* Technology. *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 1–9. https://doi.org/10.1109/SCEECS57921.2023.10062983

Benet, J. (2014). *IPFS - Content Addressed, Versioned, P2P File System* (arXiv:1407.3561). arXiv. http://arxiv.org/abs/1407.3561

Bhagwat, M., Shah, J. C., Bilimoria, A., Parkar, P., & Patel, D. (2020). *Blockchain* to improve Academic Governance. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1–5. https://doi.org/10.1109/CONECCT50063.2020.9198665

Buterin, V. (2014). A next-generation *smart contract* and decentralized application platform. *White Paper*, *3*(37), 2–1.

Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). *Blockchain* technology: Beyond bitcoin. *Applied Innovation*, *2*(6–10), 71.

Crypto.com. (n.d.). *What Is a Crypto Wallet? A Beginner's Guide*. Retrieved July 4, 2024, from https://crypto.com/university/crypto-wallets

Daraghmi, E.-Y., Daraghmi, Y.-A., & Yuan, S.-M. (2019). UniChain: A Design of *Blockchain*-Based System for Electronic Academic Records Access and Permissions Management. *Applied Sciences*, *9*(22), Article 22.

https://doi.org/10.3390/app9224966

Fan, S., Lin, C., Khazaei, H., & Musilek, P. (2022). *Performance Analysis of Hyperledger Besu in Private Blockchain*.

Grech, A., & Camilleri, A. F. (2017). *Blockchain in Education*. pedocs. http://nbn-resolving.de/urn:nbn:de:0111-pedocs-150132

Guustaaf, E., Rahardja, U., Aini, Q., Maharani, H. W., & Santoso, N. A. (2021). *Blockchain*-based Education Project. *APTISI Transactions on Management*, *5*(1), Article 1. https://doi.org/10.33050/atm.v5i1.1433

Habib, Md. A., Hossen Manik, Md. M., & Zaman, S. (2023). A *Blockchain*-based Technique to Prevent Grade Tampering: A University Perspective. *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6. https://doi.org/10.1109/ECCE57851.2023.10101502

Han, M., Li, Z., He, J. (Selena), Wu, D., Xie, Y., & Baba, A. (2018). A Novel *Blockchain*-based Education Records Verification Solution. *Proceedings of the 19th Annual SIG Conference on Information Technology Education*, 178–183. https://doi.org/10.1145/3241815.3241870

Huang, S. (2020). Academic Records Verification Platform Based on *Blockchain* Technology. *2020 International Conference on Computer Science and Management Technology (ICCSMT)*, 203–206. https://doi.org/10.1109/ICCSMT51754.2020.00048

Idelberger, F., Governatori, G., Riveret, R., & Sartor, G. (2016). Evaluation of Logic-Based Smart Contracts for *Blockchain* Systems. In J. J. Alferes, L. Bertossi, G. Governatori, P. Fodor, & D. Roman (Eds.), *Rule Technologies.*

Research, Tools, and Applications (pp. 167–183). Springer International Publishing. https://doi.org/10.1007/978-3-319-42019-6_11

Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, *1*(1), 36–63. https://doi.org/10.1007/s102070100002

Kanan, T., Obaidat, A. T., & Al-Lahham, M. (2019). SmartCert BlockChain Imperative for Educational Certificates. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 629–633. https://doi.org/10.1109/JEEIT.2019.8717505

Khan, A. G., Zahid, A. H., Hussain, M., & Riaz, U. (2019). Security of *cryptocurrency* using hardware wallet and qr code. *2019 International Conference on Innovative Computing (ICIC)*, 1–10. https://ieeexplore.ieee.org/abstract/document/8966739/

Koulu, R. (2016). Blockchains and Online Dispute Resolution: Smart Contracts as an Alternative to Enforcement. *SCRIPTed: A Journal of Law, Technology and Society*, *13*, 40.

Lazuardy, M. F. S., & Anggraini, D. (2022). Modern front end web architectures with react. Js and next. Js. *Research Journal of Advanced Engineering and Science*, *7*(1), 132–141.

Li, H., & Han, D. (2019). EduRSS: A *Blockchain*-Based Educational Records Secure Storage and Sharing Scheme. *IEEE Access*, *7*, 179273–179289. https://doi.org/10.1109/ACCESS.2019.2956157

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*.

https://assets.pubpub.org/d8wct41f/31611263538139.pdf

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press. https://books.google.com/books?hl=en&lr=&id=LchFDAAAQBAJ&oi=fnd&pg=PP1&dq=Narayanan,+A.,+Bonneau,+J.,+Felten,+E.,+Miller,+A.,+%26+Goldfeder,+S.+(2016).+Bitcoin+and+*cryptocurrency*+technologies:+A+comprehensive+introduction.+Princeton+University+Press.&ots=AtlNaW5OiF&sig=YpvgW-lbJQW552-9Qhy9Q8jDeEc

Odeniran, Q., Wimmer, H., & Du, J. (2024). *JavaScript* Frameworks—A Comparative Study Between *React*.js and Angular.js. In *Interdisciplinary Research in Technology and Management*. CRC Press.

Pathak, S., Gupta, V., Malsa, N., Ghosh, A., & Shaw, R. N. (2022). *Smart Contract* for Academic Certificate Verification Using *Ethereum*. In R. N. Shaw, S. Das, V. Piuri, & M. Bianchini (Eds.), *Advanced Computing and Intelligent Technologies* (pp. 369–384). Springer Nature. https://doi.org/10.1007/978-981-19-2980-9_29

Phillips, D. (2018, December 2). *The Best Ethereum Wallets (ETH) [December 2018]*. BeInCrypto. https://beincrypto.com/the-best-*ethereum*-wallets-eth-december-2018/

Pilkington, M. (2015). *Blockchain Technology: Principles and Applications* (SSRN Scholarly Paper 2662660). https://papers.ssrn.com/abstract=2662660

Private networks | Besu documentation. (2024, June 24).

https://besu.hyperledger.org/*private*-networks

Raimundo, R., & Rosário, A. (2021). *Blockchain* System in the Higher Education. *European Journal of Investigation in Health, Psychology and Education*, *11*(1), Article 1. https://doi.org/10.3390/ejihpe11010021

Rasool, S., Saleem, A., Iqbal, M., Dagiuklas, T., Mumtaz, S., & Qayyum, Z. ul. (2020). Docschain: *Blockchain*-Based IoT Solution for Verification of Degree Documents. *IEEE Transactions on Computational Social Systems*, *7*(3), 827–837. https://doi.org/10.1109/TCSS.2020.2973710

Raths, D. (2016). How *blockchain* will disrupt the higher education transcript. *Campus Technology*, *16*(05), 2016.

React. (n.d.). Retrieved June 24, 2024, from https://react.dev/

Sharples, M., & Domingue, J. (2016). The *Blockchain* and Kudos: A Distributed System for Educational Record, Reputation and Reward. In K. Verbert, M. Sharples, & T. Klobučar (Eds.), *Adaptive and Adaptable Learning* (pp. 490–496). Springer International Publishing. https://doi.org/10.1007/978-3-319-45153-4_48

Sillaber, C., & Waltl, B. (2017). Life Cycle of Smart Contracts in *Blockchain* Ecosystems. *Datenschutz Und Datensicherheit - DuD*, *41*(8), 497–500. https://doi.org/10.1007/s11623-017-0819-7

Sun, H., Wang, X., & Wang, X. (2018). Application of *Blockchain* Technology in Online Education. *International Journal of Emerging Technologies in Learning (iJET)*, *13*(10), Article 10. https://doi.org/10.3991/ijet.v13i10.9455

Suratkar, S., Shirole, M., & Bhirud, S. (2020). Cryptocurrency Wallet: A Review.

2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), 1–7. https://doi.org/10.1109/ICCCSP49186.2020.9315193

Thakur, K., Pathan, A.-S. K., & Ismat, S. (2023). *Blockchain* Technology. In K. Thakur, A.-S. K. Pathan, & S. Ismat (Eds.), *Emerging ICT Technologies and Cybersecurity: From AI and ML to Other Futuristic Technologies* (pp. 125–145). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-27765-8_4

Thirdweb docs. (n.d.). Retrieved June 24, 2024, from https://portal.thirdweb.com/

Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys & Tutorials*, *18*(3), 2084–2123. https://doi.org/10.1109/COMST.2016.2535718

Underwood, S. (2016). *Blockchain* beyond bitcoin. *Communications of the ACM*, *59*(11), 15–17. https://doi.org/10.1145/2994581

Utomo, A. P., Prasetyo, J. A., Rhamadhani, D. A., & Ayatullah, M. D. (2023). Web-Based Student Activity Credit Unit Information System at State Polytechnic of Banyuwangi. *Journal of Telecommunication Network (Jurnal Jaringan Telekomunikasi)*, *13*(1), Article 1. https://doi.org/10.33795/jartel.v13i1.592

Wang, H., Chen, K., & Xu, D. (2016). A maturity model for *blockchain* adoption. *Financial Innovation*, *2*(1), 12. https://doi.org/10.1186/s40854-016-0031-z

Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., Wen, Y., & Kim, D. I. (2019). A Survey on Consensus Mechanisms and Mining Strategy

Management in *Blockchain* Networks. *IEEE Access*, *7*, 22328–22370. https://doi.org/10.1109/ACCESS.2019.2896108

Zhao, H., Zhang, M., Wang, S., Li, E., Guo, Z., & Sun, D. (2021). Security risk and response analysis of typical information and communication *blockchain* application architecture. *Neural Computing and Applications*, *33*(13), 7661–7671. https://doi.org/10.1007/s00521-020-05508-z

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of *Blockchain* Technology: Architecture, Consensus, and Future Trends. *2017 IEEE International Congress on Big Data (BigData Congress)*, 557–564. https://doi.org/10.1109/BigDataCongress.2017.85

Zheng, Z., Xie, S., Dai, H.-N., Chen, W., Chen, X., Weng, J., & Imran, M. (2020). An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, *105*, 475–491. https://doi.org/10.1016/j.future.2019.12.019