APLIKASI MOBILE UNTUK E-VOTING DENGAN *BLOCKCHAIN*

SKRIPSI

Digunakan Sebagai Syarat Maju Ujian Diploma IV
Politeknik Negeri Malang

Oleh:

SILVIA PRADA APRILIA NIM. 2041720141

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
JULI 2024

HALAMAN PENGESAHAN

APLIKASI MOBILE UNTUK E-VOTING DENGAN BLOCKCHAIN

Disusun oleh:

SILVIA PRADA APRILIA NIM. 2041720141

Laporan Akhir ini telah diuji pada tanggal 19 Juli 2024

Disetujui oleh:

1.	Pembimbing Utama	:	Dwi Puspitasari, S.Kom., M.Kom. NIP. 19791115 200501 2 002	.....
2.	Pembimbing Pendamping	:	Yan Watequlis Syaifudin, ST., M.MT., Ph.D. NIP. 19810105 200501 1 005	.....
3.	Penguji Utama	:	XXXXXXXXXXX NIP. XXXX	.....
4.	Penguji Pendamping	:	XXXXXXXXXXX NIP. XXXX	.....

Mengetahui,

Ketua Jurusan Teknologi Informasi	Ketua Program Studi Teknik Informatika
Dr. Eng. Rosa Andrie Asmara, ST., MT.	Dr. Ely Setyo Astuti, S.T., M.T.
NIP. 19801010 200501 1 001	NIP. 19760515 200912 2 001

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, 19 Juli 2024

Silvia Prada Aprilia

ABSTRAK

Prada A., Silvia. "Aplikasi Mobile untuk E-Voting dengan Blockchain". **Pembimbing: (1) Dwi Puspitasari, S.Kom., M.Kom., (2) Yan Watequlis S.T., M.MT., Ph.D.**

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

Dalam negara demokratis seperti Indonesia, voting atau pemungutan suara adalah proses penting dalam pengambilan keputusan. Sistem voting manual sering dianggap kurang efisien dan rentan terhadap manipulasi. Seiring dengan perkembangan teknologi, penelitian mengenai e-voting telah banyak dilakukan untuk meningkatkan efisiensi dan keamanan proses pemungutan suara. Penelitian ini bertujuan untuk mengembangkan aplikasi mobile e-voting berbasis blockchain yang aman, transparan, dan efisien. Teknologi blockchain dipilih karena kemampuannya dalam menyediakan transparansi dan keamanan tinggi, yang penting untuk menjaga integritas proses pemungutan suara. Aplikasi ini dikembangkan menggunakan teknologi Go Ethereum (Geth) untuk membangun private blockchain. Metode pengumpulan data meliputi studi pustaka dan dokumentasi. Analisis data dilakukan dengan mengidentifikasi pola dan informasi penting dari data yang dikumpulkan. Pengujian sistem dilakukan melalui beberapa skenario untuk memastikan setiap komponen berfungsi sesuai dengan tujuan.

Oleh karena itu, implementasi aplikasi e-voting berbasis blockchain menunjukkan bahwa sistem ini dapat meningkatkan keamanan dan transparansi proses pemungutan suara. Uji coba sistem membuktikan bahwa hanya suara yang valid yang dicatat dalam blockchain dan setiap transaksi diverifikasi oleh node validator. Fitur autentikasi dua faktor dan enkripsi data pengguna turut meningkatkan keamanan sistem. Aplikasi mobile e-voting berbasis blockchain yang dikembangkan dalam penelitian ini terbukti efektif dalam meningkatkan efisiensi dan keamanan proses pemilihan. Sistem ini diharapkan dapat menjadi solusi inovatif untuk pemilihan umum yang lebih transparan dan dapat dipercaya.

Kata Kunci : E-voting, Blockchain, Aplikasi Mobile, Keamanan, Transparansi

ABSTRACT

Kartika P., Anggi**. ""Mobile Application for E-Voting with Blockchain".". **Supervisor: Dwi Puspitasari, S.Kom., Co–Supervisor: Yan Watequlis S.T., M.MT., Ph.D.

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

In a democratic country like Indonesia, voting is an important process in decision making. Manual voting systems are often considered less efficient and susceptible to manipulation. Along with technological developments, much research has been carried out on e-voting to increase the efficiency and security of the voting process. This research aims to develop a blockchain-based e-voting mobile application that is safe, transparent and efficient. Blockchain technology was chosen for its ability to provide high levels of transparency and security, which are important for maintaining the integrity of the voting process. This application was developed using Go Ethereum (Geth) technology to build a private blockchain. Data collection methods include literature study and documentation. Data analysis is carried out by identifying patterns and important information from the data collected. System testing is carried out through several scenarios to ensure each component functions as intended.

Therefore, the implementation of a blockchain-based e-voting application shows that this system can increase the security and transparency of the voting process. A test run of the system proved that only valid votes are recorded in the blockchain and each transaction is verified by validator nodes. Two-factor authentication features and user data encryption also increase system security. The blockchain-based e-voting mobile application developed in this research is proven to be effective in increasing the efficiency and security of the election process. It is hoped that this system can be an innovative solution for more transparent and trustworthy general elections.

Keywords: *E-voting, Blockchain, Mobile Application, Security, Transparency*

KATA PENGANTAR

Puji Syukur kami panjatkan kehadirat Allah SWT/Tuhan YME atas segala rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi dengan judul "APLIKASI MOBILE UNTUK E-VOTING 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 Dr. Eng. Rosa Andrie Asmara, ST., MT., selaku Ketua Jurusan Teknologi Informasi
2. Bapak Imam Dr. Ely Setyo Astuti, S.T., M.T., selaku Ketua Program Studi DIV Teknik Informatika
3. Ibu Dwi Puspitasari, S.Kom., M.Kom. selaku Dosen Pembimbing Utama
4. Bapak Yan Watequlis Syaifudin, ST., selaku Dosen Pembimbing Pendamping
5. 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, 19 Juli 2024

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I. PENDAHULUAN

1.1 Latar Belakang

Voting atau pemungutan suara sudah menjadi proses yang umum dalam sebuah negara demokratis, termasuk di Indonesia. Voting digunakan untuk mengumpulkan aspirasi, juga mengambil keputusan yang sebaik-baiknya sesuai dengan aspirasi masyarakat. Terlebih dalam sebuah negara demokrasi, voting digunakan untuk mengambil keputusan negara, diantaranya pemilihan wakil rakyat maupun pemimpin melalui pemilihan umum (Balqis et al., 2018). Pelaksanaan voting secara manual dengan cara mencoblos atau mencoreng kertas suara dinilai kurang efektif jika dilakukan pada lingkup yang luas. Banyaknya biaya untuk penyediaan kertas suara hingga keefektifan serta keabsahan penghitungan suara menjadi pertimbangan. Seiring dengan berkembangnya teknologi, mulai dilakukan penelitian-penelitian mengenai pelaksanaan voting yang lebih efektif dengan menggunakan elektronik voting (e-voting).

E-voting adalah sebuah sistem yang memanfaatkan perangkat elektronik dan mengolah informasi digital untuk membuat surat suara, memberikan suara, menghitung perolehan suara, menayangkan perolehan suara dan memelihara serta menghasilkan jejak audit (Saputra & Nasution, 2021). Implementasi e-voting dapat meningkatkan efisiensi dan kecepatan dalam proses pemilihan serta mengurangi biaya yang terkait dengan pemilihan, seperti biaya untuk mencetak kertas suara dan membangun tempat pemungutan suara. Dengan demikian, e-voting memiliki potensi untuk meningkatkan partisipasi pemilih dan kepercayaan pemilih dalam proses demokrasi. Namun, implementasi e-voting juga memerlukan perhatian terhadap keamanan, privasi, dan verifikasi suara, sehingga dapat diadopsi secara efektif dan dapat dipercaya oleh Masyarakat.

E-voting merupakan sistem pemungutan suara elektronik yang harus diseriusi dan menjamin transparansi, kepastian, keamanan, akuntabilitas, dan akurasi. Maka dari itu dibutuhkan aplikasi dengan keamanan yang kuat dan sistem penghitungan yang akurat. Berdasarkan penelitian yang ada, penerapan sistem e-voting masih terkendala dengan adanya *hacker* yang bisa membobol sistem (Karmanis, 2021). Penerapan *database* konvensional pada penyimpanan suara dapat menjadi masalah

karena rentannya data terkena manipulasi. E-voting konvensional, mengalami tiga masalah besar: manipulasi data, masalah keamanan, dan transparansi (Malkawi et al., 2021). Dengan demikian, diperlukan teknologi yang lebih kompleks untuk penyimpanan data pemilih dan suara agar tidak terjadi kecurangan-kecurangan dalam pemilihan.

Blockchain adalah sekelompok blok yang dihubungkan bersama melalui tautan. Tautan ini menghubungkan satu blok ke blok berikutnya, sedemikian rupa sehingga menghasilkan rantai blok yang tersusun (Liu & Wang, 2017). Implementasi teknologi *blockchain* dalam e-voting dapat memberikan tingkat transparansi, keamanan, dan efisiensi biaya yang lebih tinggi. *Blockchain* menyediakan platform untuk membangun aplikasi atau *smart contract* yang terdistribusi dan tidak dapat diubah. *Smart contract* mengotomatiskan transaksi dan memungkinkan para pihak mencapai kesepakatan secara langsung dan otomatis, tanpa memerlukan perantara (Hjalmarsson et al., 2018). Sehingga penggunaan *blockchain* memungkinkan semua pihak dapat melihat transaksi yang sedang berlangsung. Struktur *blockchain* juga akan menghilangkan kebutuhan akan pihak ketiga yang terpercaya karena kesepakatan dicapai secara langsung tanpa perantara.

Dengan demikian, penggunaan teknologi *blockchain* dalam aplikasi *mobile* e-voting dapat memberikan solusi yang inovatif dan aman untuk proses pemilihan yang lebih transparan dan dapat dipercaya. Dilakukannya penelitian ini dimaksudkan untuk mengimplementasikan teknologi *blockchain* pada aplikasi *mobile* e-voting yang bersifat transparan dan aman tanpa adanya manipulasi data. Implementasi aplikasi *mobile* e-voting berbasis *blockchain* diharapkan dapat mempermudah proses pemilihan umum dan menjaga keamanan serta kerahasiaan suara, sesuai dengan prinsip-prinsip pemilihan. Penggunaan platform *mobile* dalam penelitian ini dimaksudkan untuk mengintegrasikan fitur keamanan yang lebih kuat, seperti autentikasi dua faktor atau pemindaian sidik jari, yang akan meningkatkan tingkat keamanan dalam proses pemilihan. Selain itu, aplikasi *mobile* juga memungkinkan pemutakhiran dan pemantauan secara *real-time*, yang mendukung transparansi dan mengurangi risiko manipulasi data.

1.2 Rumusan Masalah

Terdapat beberapa permasalahan yang didapatkan dari latar belakang di atas, yaitu:

1. Bagaimana mengatasi manipulasi data perolehan suara pada aplikasi *mobile* e-voting?
2. Bagaimana transparansi data pemilihan dapat dicapai dengan tetap menjaga anonimitas pemilih pada aplikasi *mobile* e-voting?

1.3 Batasan Masalah

Beberapa permasalahan yang telah diangkat memiliki batasan sebagai berikut:

1. Teknologi yang digunakan adalah *private blockchain* yang terbatas aksesnya dan umumnya tidak terbuka untuk umum.
2. Penelitian berfokus pada pengembangan *blockchain* dan *smart contract*.
3. Pengembangan *blockchain* pada penelitian ini menggunakan Go Ethereum.

1.4 Tujuan

Tujuan dari dilakukannya penelitian dengan judul "**APLIKASI MOBILE UNTUK E-VOTING DENGAN BLOCKCHAIN**" adalah sebagai berikut:

1. Mengatasi manipulasi data perolehan suara pada aplikasi *mobile* e-voting menggunakan *blockchain*.
2. Melakukan transparansi data pemilihan dengan tetap menjaga anonimitas pemilih pada aplikasi *mobile* e-voting dengan menggunakan *blockchain*

1.5 Manfaat

Penelitian ini diharapkan dapat memberikan beberapa manfaat. Manfaat-manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Menciptakan sarana e-voting yang aman dan transparan menggunakan teknologi *blockchain*.
2. Meningkatkan efisiensi dan kecepatan dalam proses pemilihan serta mengurangi biaya yang terkait dengan pemilihan.
3. Meningkatkan partisipasi pemilih dan kepercayaan pemilih dalam proses demokrasi.

4. Memberikan solusi yang inovatif dan aman untuk proses pemilihan yang lebih transparan dan dapat dipercaya.

BAB II. LANDASAN TEORI

2.1 Studi Literatur

Sehubungan dengan penelitian yang dilakukan, penulis sedikit banyak terinspirasi dan mereferensi dari penelitian-penelitian terkait dengan topik yang diteliti. Dalam penelitian ini terdapat beberapa keterkaitan dengan penelitian yang telah dilakukan sebelumnya. Adapun penelitian yang berkaitan dengan penelitian ini sebagai berikut.

Penelitian yang dilakukan oleh (Setia & Susanto, 2019). mengenai pengujian pengamanan hasil pemilihan, dilakukan dengan bahasa pemrograman Solidity untuk berinteraksi dengan Ethereum *blockchain*. Pada penelitian ini *smart contract* menghasilkan kode unik pada setiap pemilihan baru. Sehingga manipulasi hasil pemilihan tidak akan bisa dilakukan karena setiap pemilih hanya memiliki satu account dan satu *address blockchain*. Dari hasil pengujian tersebut dapat disimpulkan bahwa *smart contract blockchain* bisa digunakan untuk membuktikan hasil pemilihan yang aman.

Di sisi lain, penelitian yang dilakukan oleh (Arianto et al., 2021) membangun *prototype* sistem pemilihan suara berbasis teknologi *blockchain*. Prototipe sistem yang dibangun dilakukan pengujian sistem menggunakan teknik UAT (*User Acceptance Testing*) didapatkan hasil yang cukup memuaskan. *Prototype* sistem telah cukup memudahkan pengguna untuk dioperasikan, pengguna cukup percaya perihal keamanan sistem dan memiliki performa yang baik dalam hal kelengkapan informasi yang disajikan. Namun sistem yang dibangun juga memiliki kelemahan. Berdasarkan uji sistem yang dilakukan *prototype* sistem mendapatkan persentase yang rendah sebesar 52% manajemen data. Hal ini disebabkan oleh proses eksekusi transaksi yang memerlukan waktu 30 detik untuk dapat diproses dan dikirim ke jaringan *blockchain*. Pengambilan data dari *smart contract blockchain* juga tidak dapat secepat pengambilan data seperti sistem tersentralisasi.

(Hu et al., 2019) juga melakukan penelitian terkait e-voting menggunakan aplikasi *blockchain* berbasis *website* dengan memanfaatkan *multichain* untuk mengelola satu atau lebih *blockchain*. Dari penelitian dan pengujian sistem yang telah dibuat, aplikasi dapat membantu memudahkan penyelenggaraan pemungutan

suara. Teknologi *blockchain* yang digunakan mampu membantu menyimpan data hasil pemungutan suara yang transparan dan dapat diakses oleh publik. Teknologi ini juga membantu menjaga kerahasiaan dari *voter*. Data pemungutan suara juga tidak dapat diubah, digandakan atau dihapus. Selain itu, teknologi blockchain juga membantu melakukan verifikasi dan memilah data vote yang valid.

Penerapan sistem pemungutan suara elektronik dengan teknologi blockchain *Proof-of-Authority* (POA) Go-Ethereum yang dilakukan oleh (Hjalmarsson et al., 2018). POA menggunakan algoritma yang memberikan transaksi yang relatif cepat melalui mekanisme konsensus berdasarkan identitas sebagai taruhannya. Hasil penelitian menggunakan *blockchain* pribadi Ethereum, ratusan transaksi per detik dapat dikirim ke *blockchain*, memanfaatkan setiap aspek *smart contract* untuk meringankan beban pada *blockchain*.

Secara keseluruhan, penelitian-penelitian tersebut menunjukkan bahwa teknologi *blockchain* mampu memberikan keamanan, transparansi, dan anonimitas dalam sistem e-voting. Namun, terdapat beberapa kendala terkait keterbatasan kinerja, waktu eksekusi transaksi, serta biaya yang lebih tinggi terutama ketika menggunakan Ethereum dalam jaringan *private*. Penggunaan *website* sebagai platform *user interface* memiliki kelebihan dalam hal aksesibilitas dan kemudahan penggunaan.

Dengan menggali hasil penelitian-penelitian terdahulu, penulis dapat melihat berbagai pendekatan yang telah dilakukan dalam pengembangan aplikasi *mobile* e-voting berbasis *blockchain*, serta mempertimbangkan kelebihan dan kelemahan masing-masing pendekatan untuk diadaptasi dalam penelitian yang sedang dilakukan.

2.2 Dasar Teori

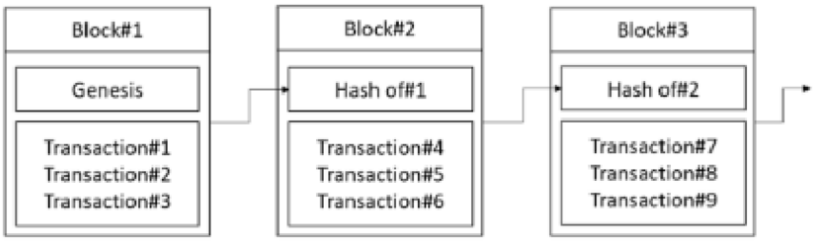
2.2.1 Elektronik Voting (E-Voting)

Electronic Voting (e-voting) atau pemungutan suara elektronik adalah proses pemungutan suara yang memungkinkan pemilih untuk memberikan suara secara aman dan rahasia melalui internet (Karmanis, 2021). Penerapan e-voting bisa beragam, mulai dari penggunaan terminal khusus di tempat pemungutan suara, aplikasi *mobile* khusus, hingga sistem *online* yang dapat diakses dari jarak jauh.

Proses penghitungan suara dapat dilakukan lebih cepat dengan bantuan teknologi, mengurangi waktu yang dibutuhkan untuk mengumumkan hasil pemilihan. E-voting dapat memiliki antarmuka pengguna yang intuitif, membuatnya lebih mudah digunakan oleh berbagai kelompok usia dan latar belakang. Dengan bantuan teknologi, kesalahan dalam proses pemungutan dan penghitungan suara dapat diminimalisir.

2.2.2 Blockchain

Blockchain adalah sekelompok blok yang dihubungkan bersama melalui tautan. Tautan ini menghubungkan satu blok ke blok berikutnya, sedemikian rupa sehingga menghasilkan rantai blok yang tersusun (Liu & Wang, 2017). Setiap blok berkontribusi pada pembangunan blok berikutnya. ini menandakan bahwa hubungan antar blok sangat kuat. Setiap blok memiliki data transaksi, *nonce*, *signature* dari blok sebelumnya, dan *signature* dari blok yang sama. *Signature* bersifat unik dan sesuai dengan data dan *nonce* blok. Proses pembuatan *signature* dilakukan dengan menggunakan algoritma hashing. Setiap *signature* blok berkontribusi untuk membangun *signature* dari blok berikutnya. Metode koneksi ini merupakan kunci kekuatan teknologi *blockchain* (Malkawi et al., 2021).



Gambar 2. 1 Ilustrasi *Blockchain*

Dalam bukunya yang berjudul "*Blockchain in Libraries*", (Meth, 2019) secara tersirat menjelaskan bahwa *blockchain* dapat dikelompokkan ke dalam dua jenis:

- *Public Blockchain*: *Blockchain* Publik memungkinkan partisipasi dari semua orang. Setiap individu dapat menjadi simpul (*node*), membaca, menulis, dan melakukan pembaruan pada *blockchain* dengan menggunakan alamat pribadi masing-masing. Ini mengindikasikan bahwa jenis *blockchain* ini terbuka untuk umum. Siapa pun dengan akses internet dan perangkat komputasi yang mendukung perangkat lunak *blockchain*

dapat menggunakan kunci publik atau kunci pribadi mereka untuk berpartisipasi.

- *Private Blockchain*: Pemilik dari *Private Blockchain* memiliki kendali yang signifikan atas desain dan operasionalnya. Dalam model ini, individu yang ingin menjadi *node* harus mendapatkan izin dari pemilik *blockchain*. Hanya anggota yang diizinkan yang dapat mengakses dan menyimpan data dalam *blockchain* ini. Konsekuensinya, keamanan *blockchain* pribadi ini lebih terjaga dan cenderung bersifat eksklusif. Dalam konteks ini, para peserta dalam jaringan *blockchain* dapat diidentifikasi, dan blok dapat diubah sesuai dengan aturan yang ditetapkan oleh pemiliknya.

2.2.3 Go Ethereum

Ethereum Blockchain (Khoury et al., 2018) merupakan platform komputasi terbuka dan terdistribusi yang memungkinkan para pengembang perangkat lunak untuk membuat dan menjalankan aplikasi terdesentralisasi (DApps). Platform ini menggunakan bahasa pemrograman yang lengkap, sehingga memungkinkan pengembang untuk menulis skrip yang kompleks. Keunggulan utama Ethereum adalah desentralisasi dan keamanan yang diwarisi dari teknologi *blockchain*.

Ethereum tidak hanya menyimpan catatan transaksi keuangan seperti Bitcoin, tetapi juga untuk menyimpan data lainnya dengan konsep *smart contract* (Jri Fadli et al., 2020). *Smart contract* ini mirip dengan kelas dalam pemrograman, di mana satu *smart contract* dapat memiliki banyak objek dan fungsi. Objek dan fungsi ini dieksekusi di Mesin Virtual Ethereum (EVM) oleh *node* pada *blockchain* Ethereum. Setiap *node* yang bekerja pada *smart contract* diberi insentif dengan sejumlah kecil ether, yaitu cryptocurrency Ethereum. Hal ini memungkinkan pengembangan sistem desentralisasi untuk berbagai keperluan, tidak hanya untuk mata uang digital.

2.2.4 Smart Contract

Smart contract adalah skrip yang ditambatkan secara desentralisasi pada *blockchain* atau infrastruktur serupa yang memungkinkan pelaksanaan proses yang telah ditentukan sebelumnya secara transparan. Dengan menggunakan *smart*

contract, aset seperti uang dapat diprogram, sehingga membuka potensi penerapan yang sebelumnya tidak dapat diakses (Ante, 2021). Smart contract memungkinkan pengguna untuk melakukan transaksi *peer-to-peer* tanpa melibatkan pihak ketiga. *Smart contract* merupakan salah satu fitur kunci dalam teknologi blockchain yang memungkinkan otomatisasi eksekusi *contract* dan transaksi tanpa memerlukan intervensi pihak ketiga. Konsep ini memungkinkan *contract* untuk dieksekusi secara otomatis ketika kondisi yang ditetapkan terpenuhi, sehingga memungkinkan transparansi, keamanan, dan keandalan yang tinggi dalam eksekusi *contract*.

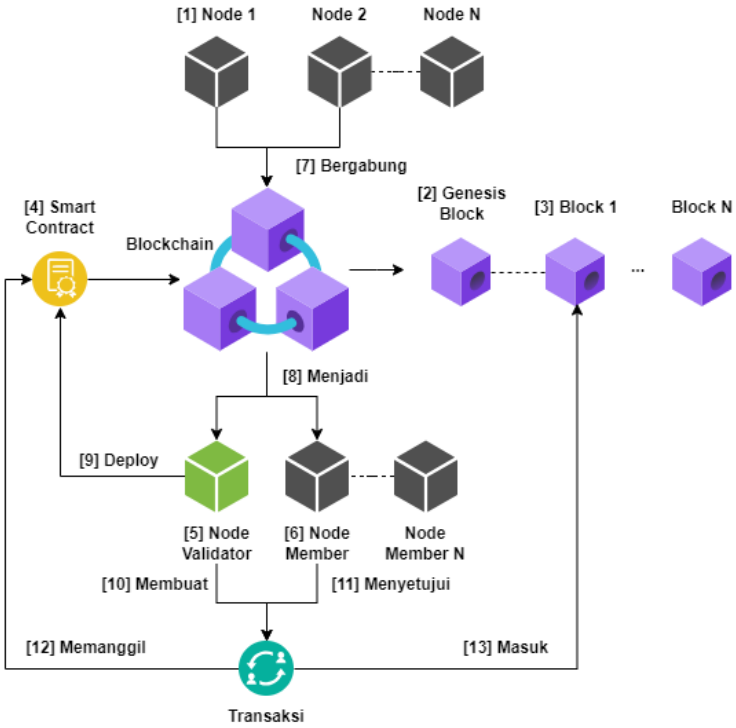
2.2.5 Web 3.0

Web 3.0, web generasi berikutnya, adalah arsitektur terdesentralisasi yang memanfaatkan teknologi *blockchain* untuk menawarkan peningkatan keamanan, privasi, dan otonomi kepada penggunanya. Konsep Web 3.0 pertama kali diusulkan oleh Gavin Wood, salah satu pendiri Ethereum pada tahun 2014. Dia percaya bahwa Web 3.0 adalah Internet berbasis *blockchain* dan tujuan Web 3.0 adalah untuk menguranginya ketergantungan pada institusi terpusat. Dengan pengayaan konsep Web 3.0, Web 3.0 bisa merekonstruksi Internet kontemporer dari dua aspek. Dari perspektif kepemilikan data, Web 3.0 tidak hanya merupakan jaringan yang dapat dibaca dan ditulis, namun juga memungkinkan pengguna untuk memiliki data dan aset. Dari perspektif manajemen data, Web 3.0 adalah sistem ekonomi baru dibangun bersama dan dibagikan oleh pengguna dan pembangun (Shen et al., 2023).

2.2.6 Proof of Authority (PoA) pada Go Ethereum

PoA adalah algoritma konsensus dalam teknologi blockchain yang menawarkan solusi efektif dan efisien untuk menjaga privasi data. Algoritma PoA menggunakan identitas terpercaya untuk memastikan keamanan dengan memvalidasi node yang dipilih secara subyektif sebagai node yang dapat dipercaya. Algoritma ini mengandalkan sejumlah kecil validator blok, membuat sistem ini sangat mudah untuk diukur dan diperluas. Blok dan transaksi diverifikasi oleh rekan-rekan yang sudah disetujui sebelumnya dan bertindak sebagai moderator dalam sistem (Asad et al., 2020). Go Ethereum (Geth) adalah salah satu implementasi populer dari klien Ethereum yang memungkinkan penerapan

algoritma PoA dalam blockchain privat. Berikut adalah gambaran penerapan konsesus PoA dalam jaringan private blockchain Go Ethereum



Gambar 2. 2 Penerapan PoA pada private blockchain Go Ethereum

Berikut penjelasan tahapan yang terjadi dari setiap angka yang terdapat pada Gambar 2.2

1) Node

   Node 1, Node 2, hingga Node N adalah peserta yang dapat bergabung dalam mengelola jaringan blockchain. Mereka bisa berfungsi sebagai node validator atau node member biasa.

2) Blok Genesis (Genesis Block)

   Blok pertama dalam blockchain yang berisi konfigurasi awal jaringan, termasuk daftar node validator yang dipercaya.

3) Blok-Blok Selanjutnya (Block 1 hingga Block N)

   Blok-blok ini mencatat transaksi-transaksi yang terjadi setelah Blok Genesis. Blok-blok tersebut dibuat dan divalidasi oleh node validator.

4) Smart Contract

   Smart contract adalah program yang berjalan di atas blockchain dan otomatis mengeksekusi perintah berdasarkan kondisi tertentu. Smart contract dapat digunakan untuk menetapkan aturan dan mekanisme PoA.

5) Node Validator

   Node-node yang telah dipercaya untuk memvalidasi transaksi dan membuat blok baru. Mereka bertanggung jawab untuk memastikan integritas dan keabsahan blok yang ditambahkan ke blockchain.

6) Node Member

   Node-node yang berpartisipasi dalam jaringan tetapi tidak memiliki wewenang untuk membuat blok baru. Mereka dapat mengirim transaksi dan menyetujui blok yang dibuat oleh node validator.

7) Bergabung ke Jaringan Blockchain

   Masing-masing node akan bergabung ke dalam jaringan blockchain sesuai dengan ketentuan dari genesis block.

8) Menjadi Node Validator atau Node Member

   Proses di mana node tertentu dapat menjadi validator atau member setelah memenuhi syarat yang telah ditentukan dalam jaringan.

9) Deploy Smart Contract

   Proses pemasangan smart contract di jaringan blockchain oleh node validator atau pihak yang berwenang lainnya.

10) Pembuatan Blok oleh Node Validator

    Node validator membuat blok baru yang berisi transaksi-transaksi yang telah diverifikasi. Blok ini kemudian ditambahkan ke blockchain setelah disetujui.

11) Persetujuan Blok oleh Node Member

    Node member menyetujui blok baru yang dibuat oleh node validator sebelum blok tersebut secara resmi ditambahkan ke blockchain.

12) Eksekusi Transaksi

    Transaksi dipanggil dan dieksekusi berdasarkan aturan yang ditentukan dalam smart contract. Node-node dalam jaringan dapat memanggil dan mengirim transaksi ini.

13) Penambahan Blok ke Blockchain

    Blok-blok yang telah divalidasi dan disetujui oleh jaringan ditambahkan ke rantai blok (blockchain).

BAB III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilakukan di Laboratorium Teknologi Informasi, Gedung Pasca Sarjana Politeknik Negeri Malang. Penelitian dilaksanakan selama enam bulan mulai bulan Januari hingga Juli 2024.

3.2 Teknik Pengumpulan Data

Teknik pengumpulan data yang digunakan untuk mendukung penelitian ini adalah sebagai berikut:

3.2.1 Studi Pustaka

Studi Pustaka dilakukan dengan mengumpulkan beberapa literatur seperti jurnal artikel, buku dan sumber lain yang diperlukan untuk merancang dan mengimplementasikan sistem. Peneliti juga mengambil referensi dari penelitian terdahulu mengenai penerapan *blockchain* pada aplikasi mobile e-voting sebagai bahan pertimbangan dan adaptasi.

3.2.2 Dokumentasi

Metode dokumentasi, merupakan teknik pengumpulan data dengan mempelajari data-data yang telah didokumentasikan. Pada penelitian ini peneliti mencari data mengenai konsep, teori, kelemahan, dan keunggulan teknologi *blockchain* yang ingin diimplementasikan.

3.3 Teknik Pengolahan Data

Penelitian ini menggunakan beberapa teknik pengolahan data untuk mengelola informasi yang terkumpul:

3.3.1 Analisis Data

Analisis data bertujuan untuk memahami, mengidentifikasi pola, dan mengekstrak informasi yang relevan dari data yang telah dianalisis. Dalam konteks ini, penulis melakukan analisis informasi yang diperoleh dari studi literatur dan dokumen terkait. Proses ini mendukung pemahaman konsep-konsep kunci, temuan

sebelumnya, dan paradigma yang berkaitan dengan implementasi aplikasi e-voting menggunakan teknologi *blockchain*.

3.3.2 Klasifikasi Data

Pada tahap klasifikasi data, penulis mengelompokkan informasi dari literatur dan dokumentasi ke dalam kategori-kategori atau tema-tema yang signifikan, seperti keamanan, keandalan, skalabilitas, dan kebutuhan pengguna. Klasifikasi ini bertujuan untuk memudahkan analisis lebih lanjut terhadap pengembangan sistem serta memahami kebutuhan pengguna dengan lebih baik.

3.3.3 Interpretasi Data

Interpretasi data dalam penelitian ini dilakukan dengan melakukan tafsir terhadap temuan dari studi literatur dan dokumen yang telah diklasifikasikan secara teoritis. Tujuannya adalah untuk mendukung argumen dan rekomendasi dalam konteks aplikasi e-voting dengan *blockchain*. Interpretasi data bertujuan untuk memberikan makna dan konteks yang lebih dalam terhadap informasi atau hasil yang diperoleh dari data yang telah diolah.

3.4 Deskripsi Sistem

Aplikasi dikembangkan menggunakan Go Ethereum (Geth) dirancang untuk menyediakan platform pemungutan suara yang aman, transparan, dan efisien. Sistem ini memanfaatkan teknologi blockchain untuk memastikan integritas dan keandalan proses pemungutan suara. Pemilih dapat memberikan suaranya jika data pemilih telah ditambahkan oleh admin dan terdapat dalam jaringan blockchain sebagai data yang valid. Admin memiliki kemampuan untuk mengatur pemilu, termasuk menentukan daftar pemilih dan kandidat. Pemilih dapat memberikan suara mereka melalui aplikasi, yang kemudian dienkripsi dan disimpan di blockchain. Hasil pemilu diumumkan dan ditampilkan secara transparan di aplikasi, memungkinkan pemilih untuk melihat jumlah suara yang diterima oleh setiap kandidat.

3.5 Uji Coba Sistem

Uji coba sistem dilakukan setelah seluruh tahap pembuatan sistem selesai. Tujuan dari uji coba sistem ini adalah untuk mengetahui seluruh sistem dapat

berjalan sesuai dengan fungsi yang telah ditentukan. Sistem akan diuji menggunakan beberapa teknik pengujian yaitu:

a. Pengujian smart contract bertujuan untuk memastikan bahwa kontrak pintar (smart contract) yang digunakan dalam sistem berjalan sesuai dengan yang diharapkan dan bebas dari kesalahan atau kerentanan.
b. Pengujian jaringan blockchain bertujuan untuk memastikan bahwa jaringan blockchain yang digunakan dapat mendukung operasional sistem dengan baik.
c. Pengujian fungsional sistem bertujuan untuk memastikan bahwa setiap komponen dari sistem e-voting bekerja sesuai dengan fungsinya dan memberikan hasil yang diharapkan.

BAB IV. ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Kebutuhan Sistem

Tujuan utama dari analisis ini adalah untuk mengidentifikasi dan mendefinisikan semua elemen penting yang diperlukan agar sistem dapat berfungsi secara optimal dan aman. Analisis yang dilakukan berupa analisis kebutuhan fungsional dan non-fungsional sistem, termasuk fitur-fitur utama seperti login pemilih, verifikasi identitas, pemungutan suara, dan penghitungan suara. Selain itu, analisis mencakup evaluasi infrastruktur teknologi yang dibutuhkan untuk mengimplementasikan private blockchain menggunakan Go Ethereum, termasuk konfigurasi node, konsensus, dan mekanisme penyimpanan data. Dengan analisis yang mendalam ini, diharapkan dapat dihasilkan sebuah desain sistem yang tidak hanya efektif dan efisien, tetapi juga aman dan dapat diandalkan untuk menyelenggarakan e-voting secara transparan dan akuntabel.

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional mengacu pada fungsi-fungsi spesifik yang harus dilakukan oleh sistem. Berikut adalah kebutuhan yang berhubungan langsung dengan apa yang harus dilakukan oleh perangkat lunak atau sistem.

Tabel 4. 1 Tabel Analisis Kebutuhan Fungsional

No	Pengguna	Kebutuhan
1.	Administrator	Sistem dapat menambahkan pemilih baru ke dalam sistem.
		Sistem dapat menambahkan kandidat baru ke dalam sistem.
		Sistem dapat melakukan update data pemilih.
		Sistem dapat melakukan update data kandidat.
		Sistem dapat menghapus data pemilih yang tidak valid
		Sistem dapat menghapus data kandidat yang tidak valid
		Sistem dapat menampilkan history data pemilih
		Sistem dapat menampilkan history data kandidat
		Sistem dapat menampilkan hasil pemungutan suata secara *real time*

		Sistem dapat menampilkan history pemungutan suara
2	Pemilih	Sistem dapat melakukan autentikasi sesuai dengan data pemilih yang telah ditambahkan
		Sistem dapat menampilkan data kandidat yang valid
		Sistem dapat menampilkan hasil pemilihan terkini secara *real time*
		Sistem dapat menampilkan profil pengguna
		Sistem dapat melukan voting dengan memasukkan password pemilih terlebih dahulu
		Sistem dapat mendeteksi pemilih yang telah melakukan pemilihan
		Sistem tidak dapat melakukan pemilihan jika pemilih telah melakukan pemilihan sebelumnya
		Sistem dapat menampilkan keterangan bahwa pemilih telah melakukan pemilihan

4.1.2 Kebutuhan Non Fungsional

Kebutuhan nonfungsional menggambarkan karakteristik atau atribut kualitas dari sistem. Kebutuhan ini lebih fokus pada bagaimana sistem tersebut bekerja daripada apa yang dikerjakannya. Berikut adalah kebutuhan nonfungsional yang berhubungan dengan sistem.

Tabel 4. 2 Tabel Analisis Kebutuhan Nonfungsional

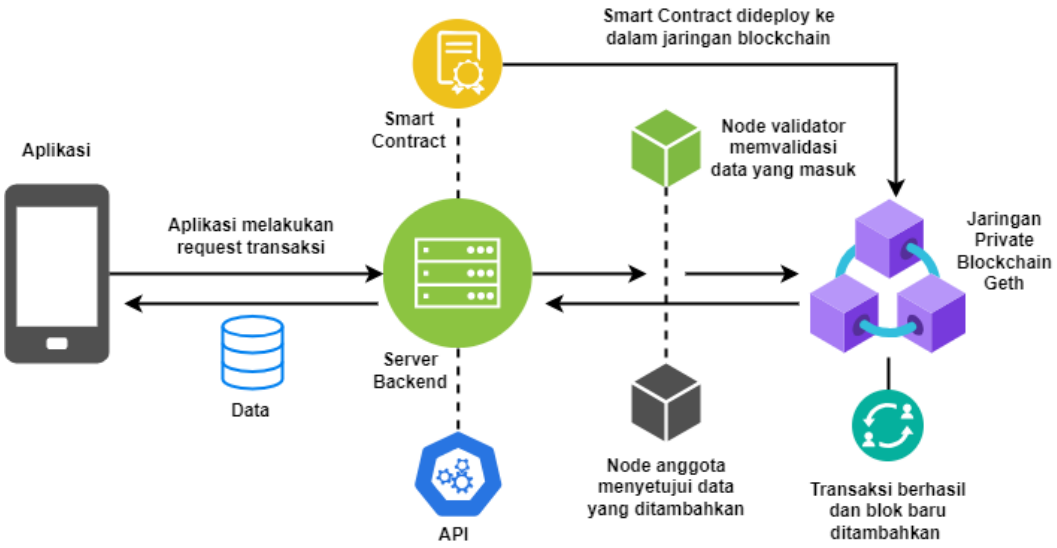
No	Atribut Kualitas	Kebutuhan
1	Keamanan	Data suara dienkripsi untuk mencegah akses yang tidak sah
		Sistem menggunakan mekanisme keamanan blockchain untuk memastikan integritas data.
2	Kinerja	Waktu respons aplikasi cepat, terutama dalam proses pemungutan suara.
		Blockchain dapat memproses dan memvalidasi transaksi suara dengan efisien.
3	Kegunaan	Antarmuka pengguna intuitif dan mudah digunakan bagi pemilih yang kurang familiar dengan teknologi.
4	Transparasi	Hasil perolehan suara secara real-time dapat diketahui oleh seluruh pihak

5	Kompabilitas	Aplikasi kompatibel dengan berbagai perangkat mobile yang umum digunakan

4.2 Perancangan Sistem

Pada tahap ini, berbagai aspek teknis dan fungsional dari sistem dirancang untuk memastikan aplikasi dapat berfungsi sesuai dengan kebutuhan yang telah diidentifikasi. Tujuan dari perancangan sistem ini adalah untuk memberikan gambaran menyeluruh tentang bagaimana sistem akan diimplementasikan, mulai dari arsitektur hingga antarmuka pengguna.

4.2.1 Arsitektur Sistem



Gambar 4. 1 Arsitektur Sistem dengan Jaringan Blockchain

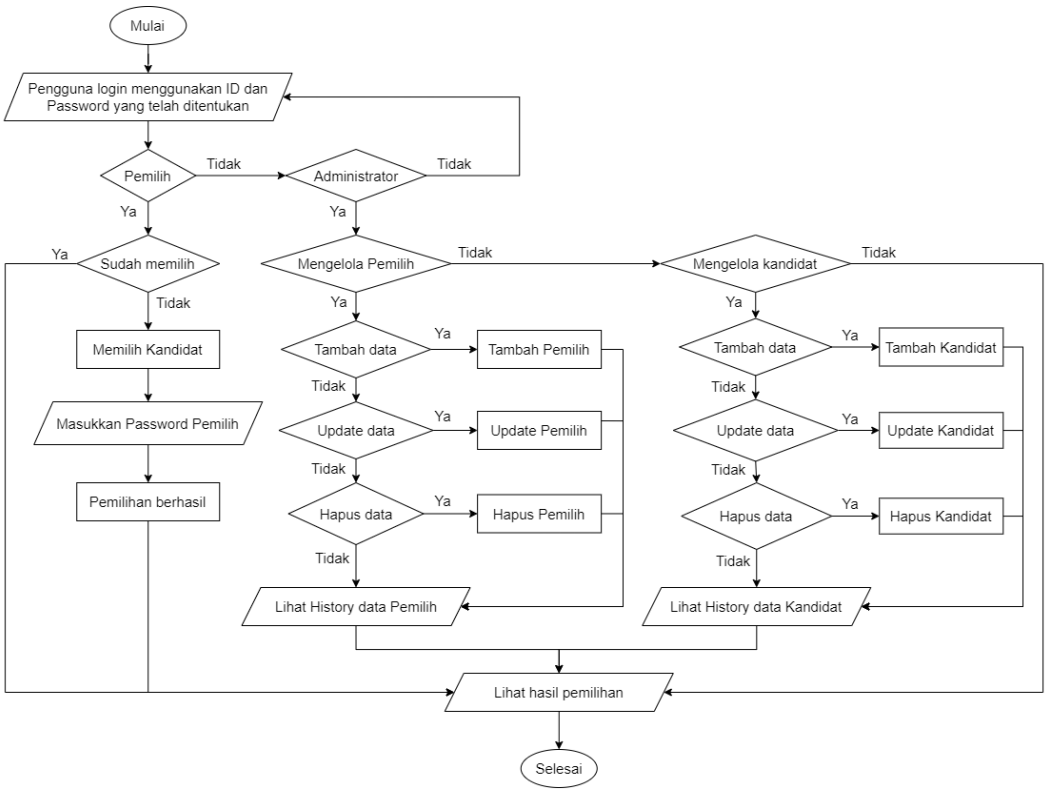
Arsitektur sistem terdiri dari beberapa komponen utama yang saling terintegrasi. Pertama, aplikasi mobile bertindak sebagai antarmuka pengguna, baik untuk pemilih maupun penyelenggara. Pemilih dapat melakukan login dan memberikan suara, sementara penyelenggara dapat memasukkan data pemilih yang valid sehingga mereka dapat memberikan suaranya. Aplikasi ini terhubung dengan server backend melalui API yang terhubung dengan *smart contract*, memastikan keamanan dalam memproses data pengguna dan transaksi. Server backend berfungsi sebagai penghubung antara aplikasi mobile dan jaringan *blockchain*. *Validator nodes* dalam jaringan bertugas memverifikasi setiap transaksi untuk memastikan bahwa hanya suara yang sah yang diterima dan dicatat. Setiap suara

yang diberikan akan dienkripsi dan dicatat sebagai transaksi di dalam *blockchain*, menghasilkan catatan yang tidak dapat diubah dan transparan.

Di sisi *blockchain*, Geth (Go-Ethereum) digunakan untuk membangun dan mengelola private blockchain, memastikan keamanan dan integritas data pemilih serta hasil pemilihan. Seluruh sistem dirancang untuk menjamin anonimitas, keamanan, dan transparansi proses pemilihan, serta mencegah kecurangan dan manipulasi data. Dengan demikian, baik pemilih maupun penyelenggara pemilihan dapat melihat hasil penghitungan suara secara langsung, memastikan proses yang cepat dan terbuka.

4.2.2 Flowchart Diagram

Flowchart berikut menggambarkan alur proses aplikasi mulai dari login pengguna hingga pemilihan selesai.

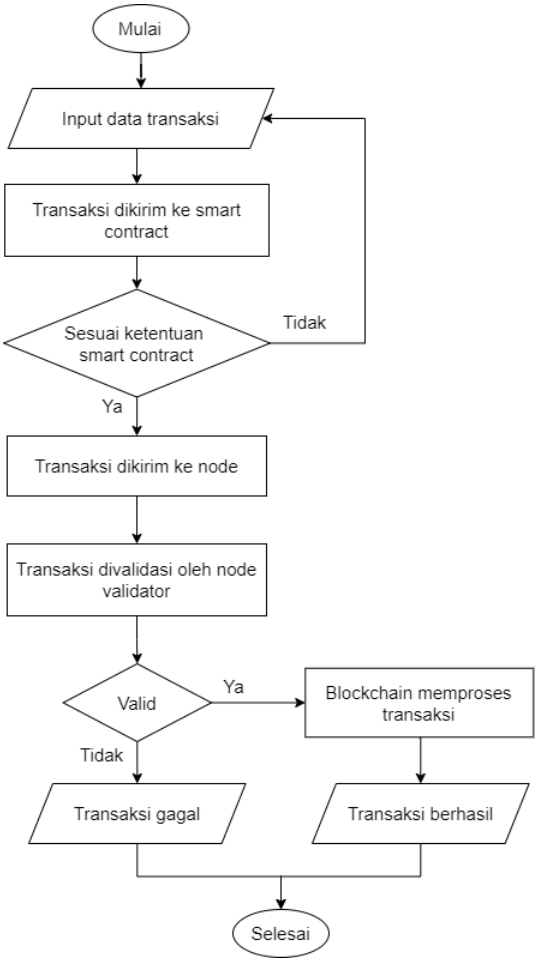


Gambar 4. 2 *Flowchart* Diagram Sistem

Proses dimulai dengan pengguna login menggunakan ID dan Password yang telah ditentukan. Setelah login, sistem akan memverifikasi apakah pengguna tersebut adalah Pemilih atau Administrator. Jika pengguna adalah Pemilih, sistem

akan memeriksa apakah Pemilih sudah melakukan pemilihan. Jika belum, Pemilih dapat memilih kandidat dan kemudian harus memasukkan Password untuk konfirmasi. Jika Password benar, pemilihan dinyatakan berhasil dan Pemilih dapat melihat hasil pemilihan.

Jika pengguna adalah Administrator, mereka memiliki opsi untuk mengelola data Pemilih dan Kandidat. Dalam mengelola Pemilih, Administrator dapat menambah, mengupdate, atau menghapus data Pemilih. Proses serupa berlaku untuk mengelola Kandidat, di mana Administrator dapat menambah, mengupdate, atau menghapus data Kandidat. Setelah menyelesaikan pengelolaan data, Administrator dapat kembali ke tahap sebelumnya atau memilih untuk melihat hasil pemilihan.



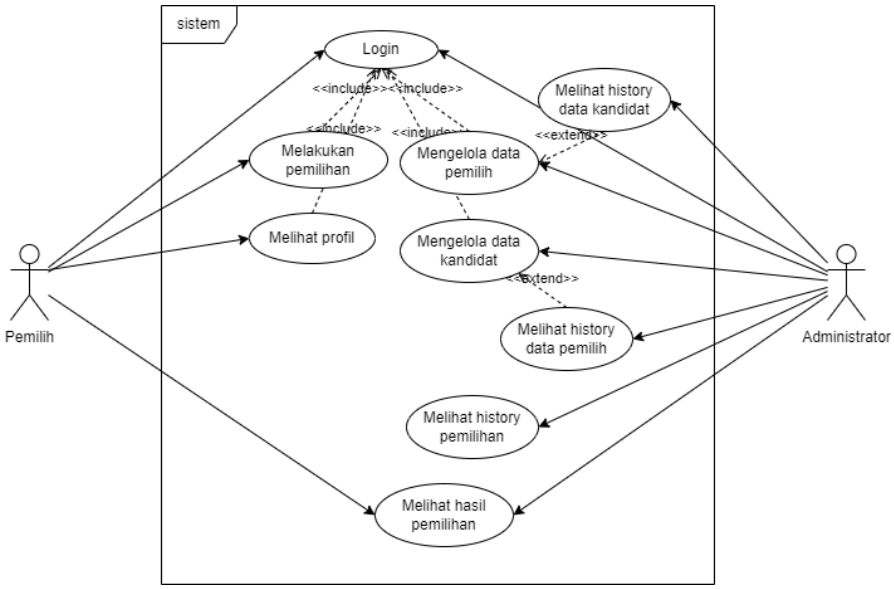
Gambar 4. 3 Flowchart Diagram Proses Transaksi

Alur penyimpanan transaksi pada private *blockchain* Ethereum dimulai dengan pengguna menginput data transaksi ke dalam sistem. Data transaksi

kemudian dikirim ke smart contract, yaitu kontrak pintar yang berisi kode yang mengeksekusi sendiri berdasarkan syarat-syarat yang telah ditentukan. Transaksi ini diperiksa untuk memastikan bahwa ia memenuhi ketentuan yang ada dalam smart contract tersebut. Jika transaksi tidak memenuhi syarat, pengguna harus mengoreksi data input dan mengirim ulang transaksi.

Jika transaksi sesuai dengan ketentuan smart contract, data transaksi kemudian dikirim ke node dalam jaringan blockchain. Node adalah komputer individu yang berpartisipasi dalam jaringan blockchain. Node validator kemudian memvalidasi transaksi tersebut untuk memastikan keaslian dan integritasnya. Jika transaksi dinyatakan valid oleh validator, blockchain akan memproses transaksi tersebut dan menambahkannya ke dalam buku besar blockchain, sehingga transaksi dianggap berhasil. Namun, jika transaksi dinyatakan tidak valid, transaksi tersebut gagal dan tidak akan ditambahkan ke dalam blockchain. Proses ini diakhiri setelah transaksi dinyatakan berhasil atau gagal.

4.2.3 Use Case Diagram



Gambar 4. 4 Use Case Diagram Sistem

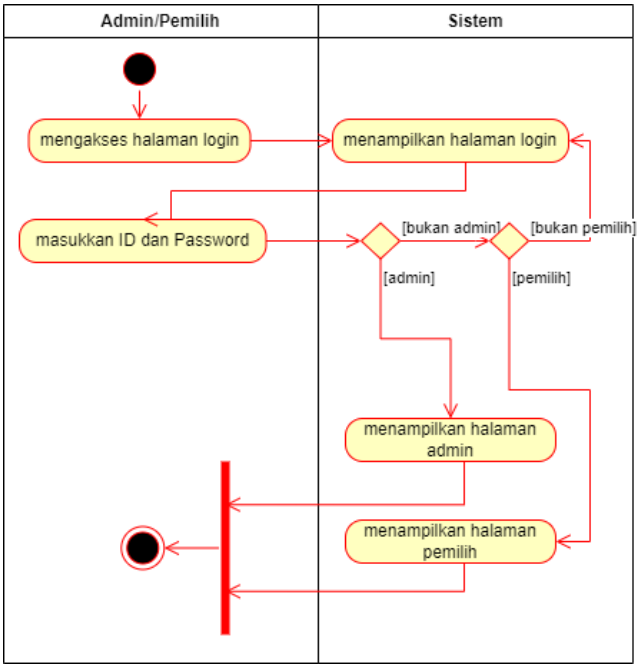
Use case diagram pada Gambar 4. 4 menggambarkan skenario penggunaan aplikasi dengan dua jenis aktor utama yaitu Pemilih dan Administrator. Pemilih dapat melakukan login, melakukan pemilihan, melihat profil dan melihat hasil pemilihan. Proses login adalah prasyarat untuk melakukan pemilihan dan melihat

profil pada aktor Pemilih. Administrator memiliki akses lebih luas, termasuk login, mengelola data pemilih, mengelola data kandidat, melihat history data kandidat, melihat history data pemilih, melihat history pemilihan dan melihat hasil pemilihan. Untuk mengelola data pemilih dan mengelola data kandidat di-include oleh proses login, sementara melihat history data kandidat dan melihat history data pemilih merupakan perluasan fungsi mengelola data kandidat dan pemilih.

4.2.4 Activity Diagram

Activity diagram pada dasarnya adalah rancangan aliran aktivitas atau aliran kerja yang digunakan pada sebuah sistem yang dijalankan. Diagram ini digunakan untuk mengelompokkan atau mendefinisikan aluran tampilan dari sistem. Berikut beberapa acitivity diagram yang mendefinisikan aktivitas yang dijalankan sistem.

1. Login

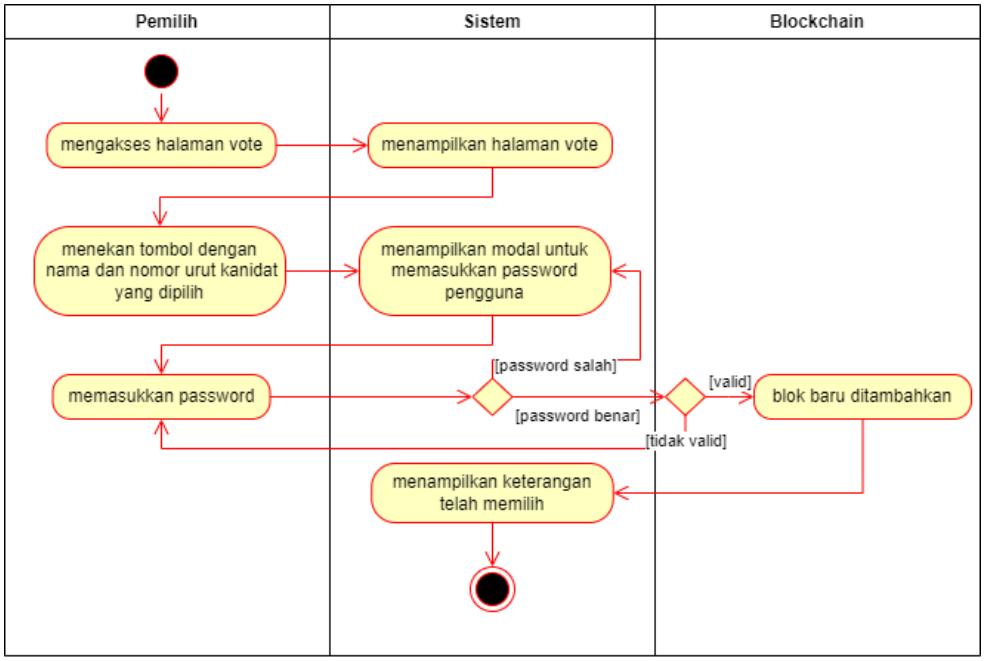


Gambar 4. 5 Activity Diagram Login

Proses dimulai ketika pengguna, baik admin maupun pemilih, mengakses halaman login dan memasukkan ID serta kata sandi mereka. Sistem kemudian menampilkan halaman login dan memverifikasi validitas ID serta kata sandi yang dimasukkan. Jika pengguna bukan admin maka sistem akan mengecek apakah penggunana merupaka pemilih, jika pengguna adalah pemilih maka sistem akan menampilkan halaman pemilih. Sebaliknya,

jika pengguna adalah admin, sistem akan menampilkan halaman admin. Setelah pengguna diarahkan ke halaman yang sesuai, proses login dianggap selesai.

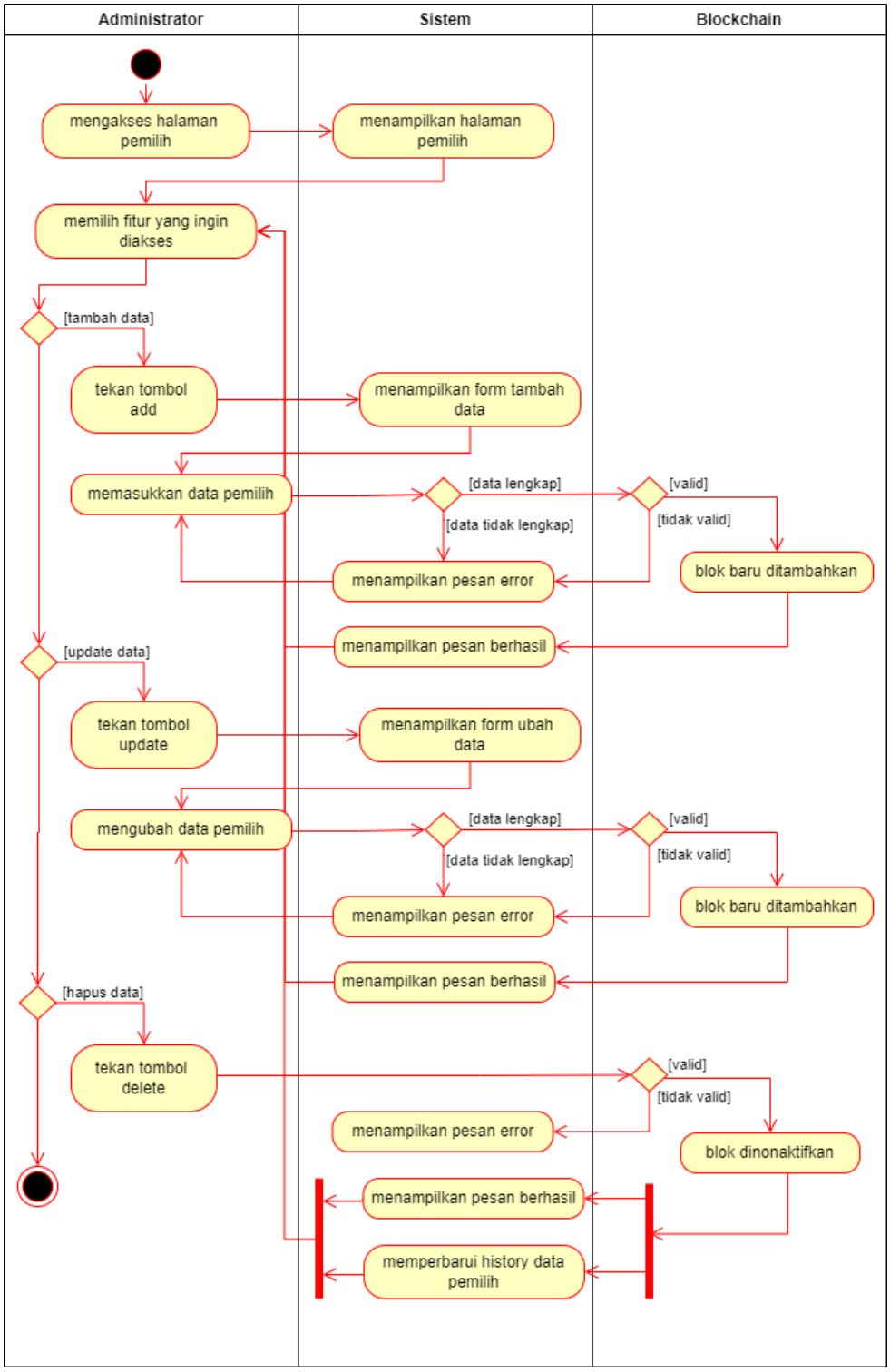
2. Pemilih melakukan pemilihan



Gambar 4. 6 Activity Diagram Pemilih Melakukan Pemilihan

Proses dimulai ketika pemilih mengakses halaman *vote*. Sistem kemudian menampilkan halaman *vote* tersebut kepada pemilih. Pemilih menekan tombol yang menunjukkan nama dan nomor urut kandidat yang dipilih, setelah itu sistem akan menampilkan modal untuk memasukkan password pengguna. Pemilih kemudian memasukkan password mereka. Jika password yang dimasukkan salah, sistem akan meminta pemilih untuk memasukkan password yang benar. Jika password yang dimasukkan benar, *validator* akan mengecek apakah data yang dimasukkan valid dan sesuai dengan ketentuan smart contract. Jika data valid maka blok baru akan ditambahkan ke dalam *blockchain* dan mencatat pilihan pemilih tersebut. Setelah blok baru ditambahkan, sistem akan menampilkan keterangan bahwa pemilih telah berhasil melakukan voting. Proses ini berakhir dengan tampilan keterangan tersebut, menandakan bahwa suara pemilih telah tercatat dengan aman pada jaringan *blockchain*.

3. Admin mengelola data Pemilih



Gambar 4. 7 Activity Diagram Admin mengelola data Pemilih

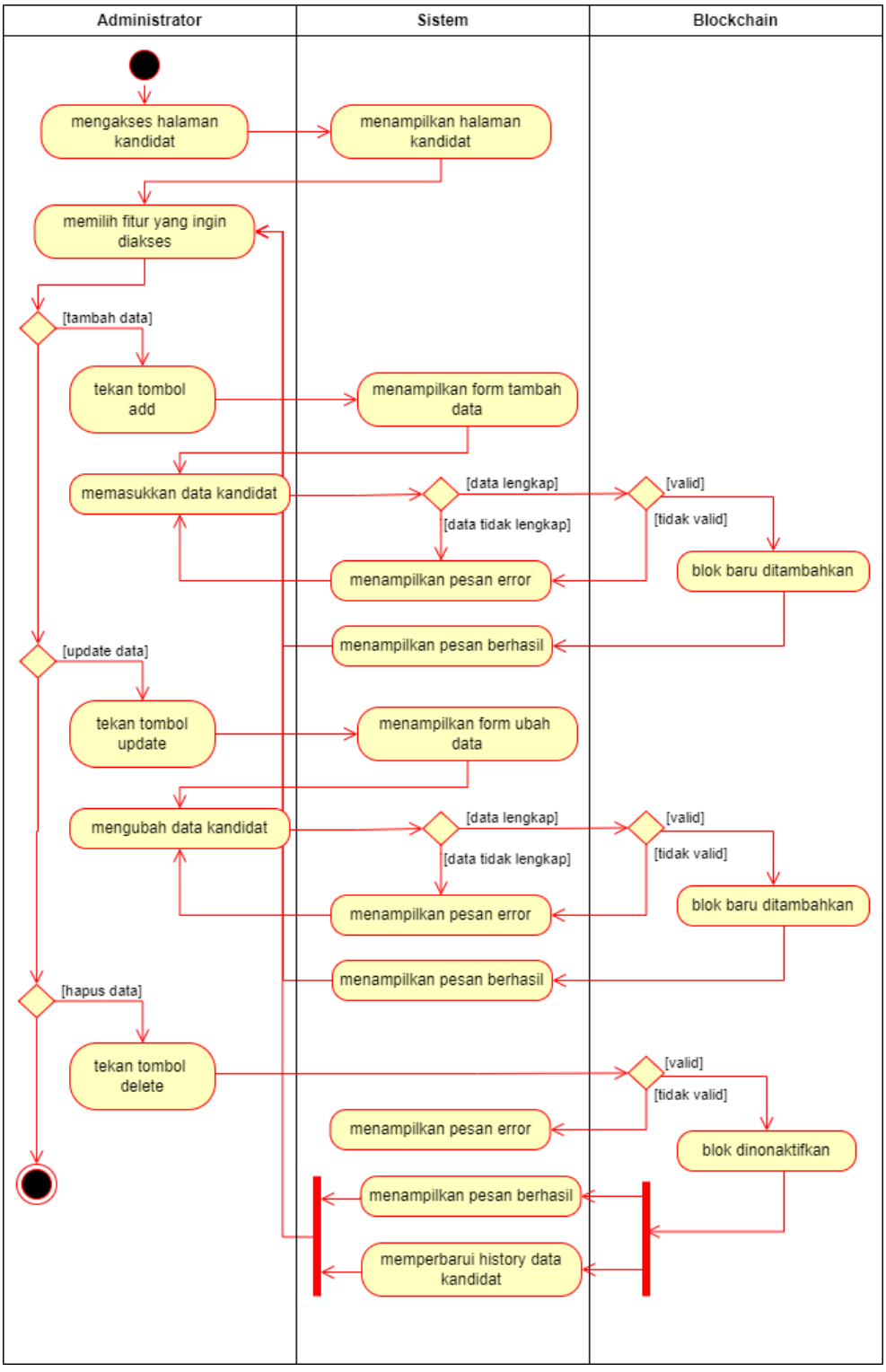
Pada diagram Gambar 4. 7 aktivitas dimulai dengan Pemilih mengakses halaman voter, dan sistem menampilkan halaman tersebut. Saat pemilih ingin menambah data, mereka mengisi form yang disediakan oleh sistem. Sistem

kemudian memeriksa kelengkapan data yang dimasukkan. Jika data tidak lengkap, sistem menampilkan pesan error. Sebaliknya, jika data lengkap, pesan berhasil ditampilkan dan data tersebut divalidasi di blockchain. Jika valid, blok baru ditambahkan ke blockchain, sedangkan jika tidak valid, data tidak ditambahkan.

Untuk mengubah data pemilih, proses serupa dilakukan. Pemilih mengakses form ubah data dan mengisi data yang diperlukan. Sistem memverifikasi kelengkapan data dan menampilkan pesan error jika data tidak lengkap. Jika data lengkap, sistem menampilkan pesan berhasil dan memvalidasi data di blockchain. Data yang valid akan ditambahkan sebagai blok baru, sementara data yang tidak valid tidak akan ditambahkan.

Pada proses penghapusan data pemilih, pemilih mengisi form penghapusan data. Sistem memverifikasi kelengkapan data dan menampilkan pesan error jika data tidak lengkap. Jika data lengkap, pesan berhasil ditampilkan, dan sistem memvalidasi data di blockchain. Setelah validasi, jika data valid, blok baru ditambahkan ke blockchain untuk mencatat penghapusan, dan sistem memperbarui riwayat data pemilih dengan menonaktifkan blok yang berkaitan.

4. Admin mengelola data Kandidat



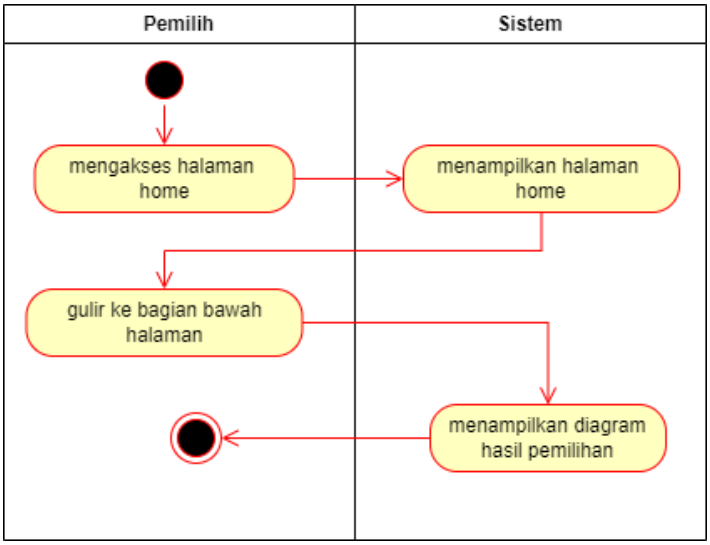
Gambar 4. 8 Activity Diagram Admin mengelola data Kandidat

Diagram activity pengelolaan kandidat mengikuti alur yang sama dengan diagram activity pemilih yang telah dijelaskan sebelumnya. Proses

dimulai dengan akses ke halaman terkait, di mana sistem menampilkan halaman tersebut. Untuk menambah, mengubah, atau menghapus data, pengguna harus mengisi formulir yang disediakan oleh sistem, yang kemudian akan memeriksa kelengkapan data yang dimasukkan. Pesan error akan ditampilkan jika data tidak lengkap. Jika data lengkap, sistem akan menampilkan pesan berhasil dan melakukan validasi data di blockchain. Blok baru akan ditambahkan ke blockchain untuk data yang valid, sementara data yang tidak valid tidak akan ditambahkan. Satu-satunya perbedaan antara kedua diagram ini adalah jenis data yang diolah, dalam diagram aktivitas kandidat, data yang diolah adalah data kandidat, sedangkan dalam diagram aktivitas pengelolaan pemilih data yang diolah adalah data pemilih.

5. Melihat Hasil Memilihan

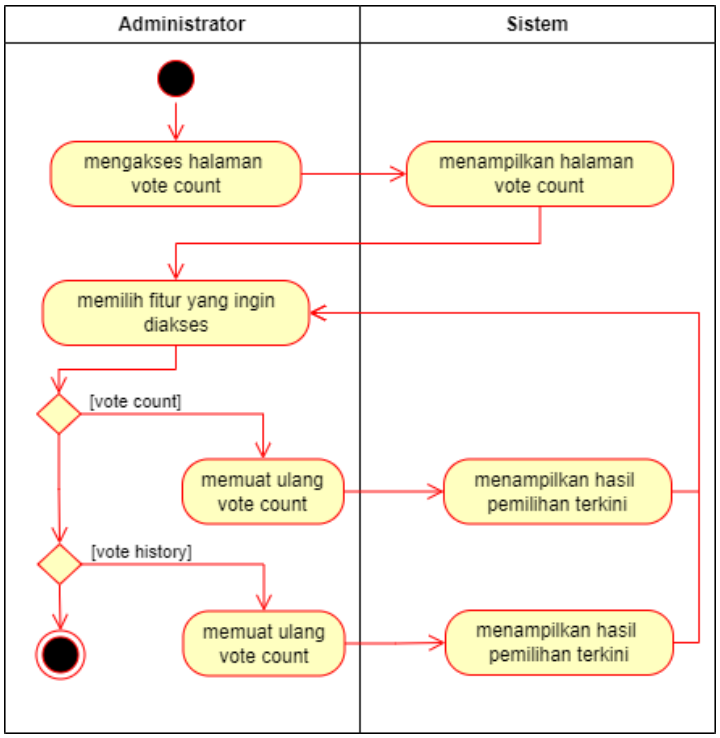
Diagram ini menggambarkan proses melihat hasil pemilihan untuk pengguna dengan peran admin dan pemilih. Berikut adalah menjelaskan langkah-langkah yang dilakukan oleh kedua jenis pengguna dari awal hingga akhir proses.



Gambar 4. 9 Activity Diagram pemilih melihat hasil pemilihan

Diagram pada Gambar 4. 9 di atas menggambarkan proses yang terjadi ketika seorang pemilih ingin melihat hasil pemilihan. Aktivitas dimulai ketika pemilih membuka halaman utama dari aplikasi atau sistem yang digunakan untuk pemilihan. Setelah pemilih mengakses halaman home, sistem akan menampilkan halaman utama tersebut kepada pemilih. Selanjutnya, pemilih

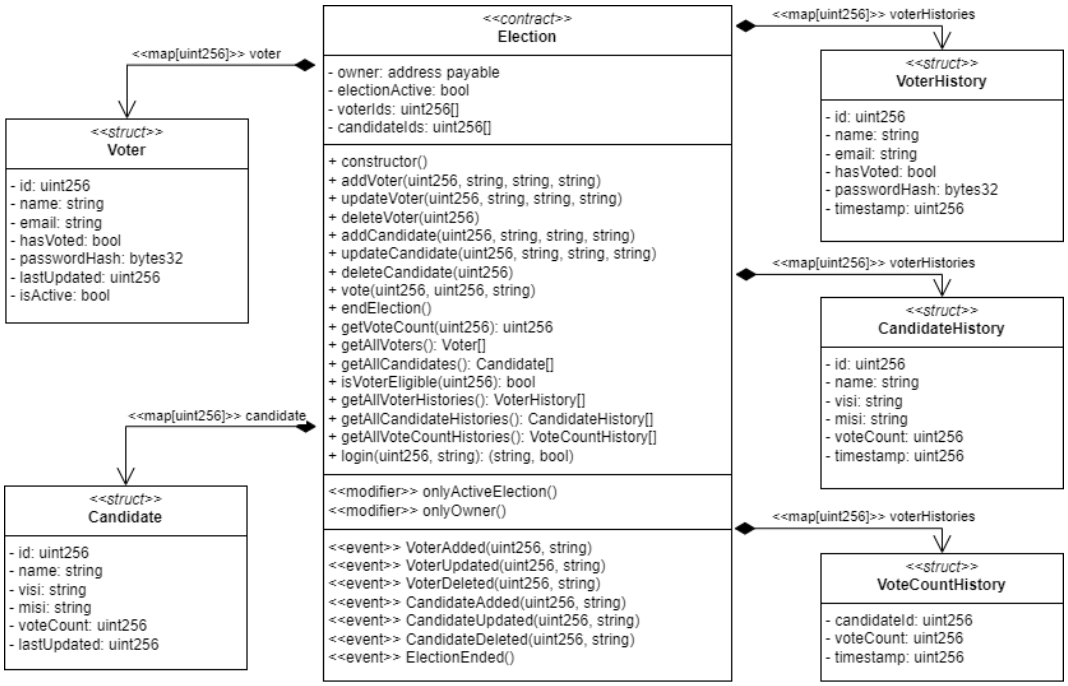
menggulir atau menelusuri halaman hingga bagian bawah untuk menemukan informasi lebih lanjut. Ketika pemilih mencapai bagian bawah halaman, sistem kemudian menampilkan diagram hasil pemilihan yang berisi informasi mengenai hasil pemilihan yang telah berlangsung.



Gambar 4. 10 Activity Diagram kandidat melihat hasil pemilihan

Diagram aktivitas di atas menggambarkan proses yang terjadi ketika seorang administrator ingin melihat hasil pemilihan. Aktivitas dimulai ketika administrator membuka halaman vote count dari aplikasi atau sistem yang digunakan untuk pemilihan. Setelah administrator mengakses halaman vote count, sistem akan menampilkan halaman tersebut kepada administrator. Selanjutnya, administrator memilih fitur yang ingin diakses di halaman tersebut, seperti melihat perhitungan suara atau riwayat pemilihan. Berdasarkan pilihan administrator, sistem akan menampilkan hasil yang sesuai. Jika administrator memilih fitur "vote count", sistem akan memuat ulang perhitungan suara terkini dan menampilkan hasil pemilihan terbaru. Jika administrator memilih fitur "vote history", sistem akan menampilkan riwayat pemilihan.

4.2.5 Desain Smart Contract



Gambar 4. 11 Desain Smart Contract

Smart contract Election dirancang untuk mengelola proses pemilihan di blockchain Ethereum. Kontrak ini dimiliki oleh sebuah variabel alamat owner dan memastikan bahwa hanya pemilik yang dapat menjalankan fungsi-fungsi tertentu. Kontrak ini memiliki beberapa pemetaan dan array, termasuk voters, candidates, voterHistories, candidateHistories, dan voteCountHistories.

Kontrak Election mendefinisikan beberapa tipe struct untuk mengatur data yaitu Voter, Candidate, VoterHistory, CandidateHistory, dan VoteCountHistory. Setiap Voter mencakup detail seperti id, name, email, hasVoted, passwordHash, lastUpdated, dan status isActive. Demikian juga, setiap Candidate berisi id, name, visi, misi, voteCount, dan lastUpdated. Struct *history* (VoterHistory, CandidateHistory dan VoteCountHistory) mencatat history dari pemilih dan kandidat, masing-masing, dengan *timestamp* untuk melacak perubahan dari waktu ke waktu.

Kontrak ini mencakup beberapa fungsi penting untuk mengelola pemilih dan kandidat, seperti addVoter, updateVoter, deleteVoter, addCandidate, updateCandidate, dan deleteCandidate. Kontrak ini juga memiliki fungsi vote yang memungkinkan pemilih yang memenuhi syarat untuk memberikan suara mereka,

memperbarui catatan pemilih dan kandidat. Fungsi endElection dapat dipanggil oleh pemilik untuk menonaktifkan pemilihan.

Untuk mengambil data, kontrak menyediakan fungsi seperti getVoteCount untuk mengambil jumlah suara saat ini dari seorang kandidat, getAllVoters dan getAllCandidates untuk daftar semua pemilih dan kandidat, isVoterEligible untuk memeriksa kelayakan pemilih, dan fungsi untuk mengambil data *history* (getAllVoterHistories, getAllCandidateHistories, dan getAllVoteCountHistories).

Modifier onlyActiveElection dan onlyOwner digunakan untuk membatasi eksekusi fungsi pada kondisi tertentu, memastikan integritas proses pemilihan. Event seperti VoterAdded, VoterUpdated, VoterDeleted, CandidateAdded, CandidateUpdated, CandidateDeleted, dan ElectionEnded dijalankan untuk mencatat tindakan dan perubahan signifikan dalam kontrak.

Selain itu, kontrak ini mencakup fungsi login yang memungkinkan pemilih untuk mengautentikasi diri mereka menggunakan id dan password mereka. Fungsi ini mengembalikan nama pemilih dan boolean yang menunjukkan keberhasilan autentikasi. Secara keseluruhan, smart contract Election menyediakan kerangka kerja yang komprehensif untuk mengelola pemilihan dengan aman, melacak data pemilih dan kandidat, serta mempertahankan catatan history di blockchain.

BAB V. IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

Berdasarkan dari proses perancangan pada tahapan sebelumnya yang sudah dilakukan, maka selanjutnya adalah melakukan tahap proses implementasi sistem dengan memaparkan secara detail sesuai dengan rancangan sebelumnya seperti pembuatan program, serta tampilan dari Aplikasi Mobile berbasis private blockchain menggunakan Go Ethereum. Implementasi ini mencakup berbagai aspek teknis, termasuk integrasi dengan blockchain private, pengembangan smart contract, dan pembuatan antarmuka pengguna yang intuitif. Pembuatan program melibatkan penulisan kode untuk aplikasi mobile menggunakan bahasa pemrograman Typescript yang sesuai untuk berbagai jenis sistem operasi mobile dan kemudian mengintegrasikannya dengan jaringan blockchain menggunakan Go Ethereum.

5.2 Implementasi Jaringan Go Ethereum

Pada implementasi jaringan langkah-langkah yang diambil mencakup inisialisasi lingkungan, pembuatan akun, konfigurasi file Genesis, dan pengoperasian node dalam jaringan. Berikut penjelasan proses tersebut secara rinci, dimulai dari persiapan lingkungan hingga pengoperasian node dalam jaringan private blockchain.

1. Instalasi Go Ethereum (Geth) dan Pembuatan Direktori

Istalasi Geth dilakuakan dengan mengakses halaman resmi Geth dan mengikuti langkah-langkah penginstallan yang telah disediakan. Setelah geth berhasil diinstall, selanjutnya membuat direktori root yang akan digunakan untuk menyimpan konfigurasi jaringan blockchain. Kemudian membuat file password.txt dan menuliskan password yang ingin digunakan dalam pembuatan blockchain. Kemudian menjalankan perintah berikut untuk membuat direktori node1 dan node2.

```
$mkdir -p ./node1/data
$mkdir -p ./node2/data
```

2. Pembuatan Akun Node

Pembuatan akun Node dilakukan dengan menjalankan perintah berikut

```
$geth --datadir "./node1/data" account new --password
password.txt
$geth --datadir "./node2/data" account new --password
password.txt
```

Perintah diatas digunakan untuk menjalankan membuat akun Ethereum baru yang disimpan pada direktori data ./node1/data menggunakan password dari file password.txt. Setelah perintah dijalankan maka akan didapatkan address dari akun yang telah dibuat. Simpan account address untuk digunakan pada pembuatan genesis blok.

3. Membuat Genesis Blok

File Genesis berisi konfigurasi awal blockchain, termasuk chainId, difficulty, gasLimit, serta alokasi Ether untuk kedua akun yang telah dibuat. Buat file baru dengan nama genesis.json dan buat konfigurasi seperti pada kode berikut.

```
{
  "config": {
    "chainId": $CHAIN_ID,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "difficulty": "1",
  "gasLimit": "8000000",
```

```
  "extradata":
"0x0000000000000000000000000000000000000000000000000000000000000000
000${INITIAL_SIGNER_ADDRESS}0000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000",
  "alloc": {
    "${FIRST_NODE_ADDRESS}": { "balance": "${ETHER_AMOUNT}" },
    "${SECOND_NODE_ADDRESS}": { "balance": "${ETHER_AMOUNT}" }
  }
}
```

Kode di atas adalah konfigurasi genesis file untuk sebuah blockchain Ethereum yang menggunakan protokol Clique, yaitu protokol konsensus Proof-of-Authority (PoA). Berikut kegunaan dari kode pada konfigurasi tersebut:

- config: Mengatur parameter jaringan, termasuk ID rantai dan blok di mana berbagai peningkatan Ethereum diaktifkan.
- chainId: ID unik untuk rantai blockchain.
- homesteadBlock, eip150Block, eip155Block, eip158Block, byzantiumBlock, constantinopleBlock, petersburgBlock, istanbulBlock, berlinBlock: Blok di mana peningkatan protokol Ethereum diaktifkan. Pada kode, semuanya diatur ke 0, yang berarti peningkatan ini aktif sejak blok pertama.
- clique: Konfigurasi khusus untuk konsensus Clique.
- period: Interval waktu dalam detik antara blok.
- epoch: Jumlah blok sebelum reset ulang RLP (Merkel Patricia Trie).
- difficulty: Tingkat kesulitan untuk menambang blok, diatur ke "1" karena ini adalah jaringan PoA dan tidak menggunakan penambangan tradisional.
- gasLimit: Batas gas per blok, yang dalam contoh ini diatur ke "8000000".
- extradata: Data tambahan untuk blok genesis. Bagian ini mencakup alamat penandatangan awal (validator) dari rantai ini.
- difficulty: Tingkat kesulitan untuk menambang blok, Pada kode diatas diatur ke "1" karena ini adalah jaringan PoA dan tidak menggunakan penambangan tradisional.
- gasLimit: Batas gas per blok, yang dalam kode diatas diatur ke "8000000".

- extradata: Data tambahan untuk blok genesis. Bagian ini mencakup signer address (validator) dari rantai blockcahin.
- ${INITIAL_SIGNER_ADDRESS}: Signer address (validator).
- alloc: Bagian ini menentukan alokasi awal Ether untuk address tertentu pada blok genesis.
- ${FIRST_NODE_ADDRESS} dan ${SECOND_NODE_ADDRESS}: Address node yang menerima alokasi awal Ether. Address node didapatkan dari hasil pembuatan akun yang telah dilakukan sebelumnya
- ${ETHER_AMOUNT}: Jumlah Ether yang dialokasikan ke alamat tersebut.

4. Inisialisasi Node

Setelah membuat file untuk konfigurasi genesis blok, Node akan diinisialisasi menggunakan file genesis. Inisialisasi dilakukan dengan menjalankan perintah berikut.

```
$geth --datadir "./node1/data" init genesis.json
$geth --datadir "./node2/data" init genesis.json
```

Perintah diatas digunakan untuk menginisialisasi direktori data node pada ./node1/data dengan file genesis.json

5. Membuat bootnode key dan Menjalankan bootnode

Bootnode key digunakan untuk menjalankan bootnode. Bootnode dijalankan untuk menghasilkan URL enode yang digunakan untuk menghubungkan node lain ke jaringan. Bootnode key dibuat menggunakan kode berikut.

```
bootnode -genkey boot.key
```

Setelah bootnode key selesai dibuat maka akan terbentuk file dengan nama boot.key yang dapat digunakan untuk memnjalankan bootnode. Kemudian, untuk menjalankan bootnode digunakan perintah berikut

```
bootnode -nodekey boot.key -verbosity 7 -addr "127.0.0.1:30301"
```

Perintah diatas digunakan untuk menjalankan bootnode dengan kunci privat boot.key dan mengatur tingkat verbositas log ke level 7, yang sangat detail, serta menentukan alamat dan port di mana bootnode akan mendengarkan koneksi. Pada perintah diatas 127.0.0.1:30301.

6. Menjalankan Node 1 dan Node 2

Setelah URL enode didapatkan, Node harus dijalankan menggunakan enode untuk menghubungkan node dalam jaringan. Untuk menjalankan node 1 dan node 2 digunakan kode berikut

```
// command untuk node 1 sebagai signer address

$geth --datadir "./node1/data" --port 30304 --bootnodes
"$BOOTNODE_URL" --authrpc.port 8547 --ipcdisable --allow-
insecure-unlock --http --
http.corsdomain="https://remix.ethereum.org" --http.api
web3,eth,debug,personal,net --networkid $NETWORK_ID --unlock
0x$FIRST_NODE_ADDRESS --password password.txt --mine --
miner.etherbase=0x$INITIAL_SIGNER_ADDRESS

// command untuk node 2 sebagai member address

$geth --datadir "./node2/data" --port 30306 --bootnodes
"$BOOTNODE_URL" --authrpc.port 8546 --networkid $NETWORK_ID --
unlock 0x$SECOND_NODE_ADDRESS --password password.txt
```

Node pertama dijalankan dengan menggunakan datadir yang telah diinisialisasi, menentukan network ID dan mengatur port dan URL bootnode, serta mengaktifkan beberapa API penting seperti web3, eth, debug, personal, dan net. Node kedua juga dijalankan dengan konfigurasi serupa, menggunakan URL bootnode yang sama untuk memastikan kedua node terhubung dalam jaringan yang sama. Dengan langkah-langkah tersebut, sebuah jaringan private blockchain Go Ethereum berhasil dibuat, dengan dua node yang terhubung melalui bootnode siap untuk melakukan operasi blockchain dan transaksi di dalam jaringan private yang telah dikonfigurasi.

5.3 Implementasi Smart Contract

Pada implementasi ini, akan dikembangkan smart contract sesuai dengan desain smart contract yang telah dibuat sebelumnya. Implementasi smart contract pada penelitian ini melibatkan pembuatan, deployment, dan interaksi dengan smart contract menggunakan Go Ethereum, serta bagaimana smart contract tersebut mengelola data pemilih dan suara secara efisien. Berikut beberapa komponen yang digunakan dalam pembuatan smart contract.

1. Contract dan Variabel

Dalam pembuatan smart contract, langkah pertama yang harus dilakukan adalah mendefinisikan kontrak itu sendiri. Kontrak untuk e-voting mencakup logika dan aturan yang mengatur seluruh proses pemilihan, termasuk penambahan pemilih, pemberian suara, dan penghitungan hasil. Pada penelitian ini, hanya satu smart contract yang digunakan, sehingga seluruh transaksi terpusat pada kontrak yang sama. Variabel dalam smart contract digunakan untuk menyimpan data yang relevan dengan proses e-voting. Dalam penelitian ini, hanya satu variabel yang didefinisikan, yaitu variabel `owner` yang menyimpan alamat (address) dari pemilik kontrak. Berikut kode untuk pendefinisian smart contract dan variabel sesuai dengan desain.

```
contract Election {
    address payable public owner;
// kode smart contract selengkapnya
}
```

2. Struktur Data

Struktur data dalam smart contract mencakup jenis-jenis data yang digunakan untuk menyimpan informasi secara efisien. Berikut struktur data yang digunakan dalam pembuatan smart contract sesuai dengan desain yang telah dibuat.

a. Voter

```
    struct Voter {
        uint256 id;
        string name;
        string email;
        bool hasVoted;
        bytes32 passwordHash;
        uint256 lastUpdated;
        bool isActive;
    }
```

Struktur data Voter digunakan untuk menyimpan data pemilih. Data yang disimpan berupa data.

- id: ID pemilih.
- name: Nama pemilih.

- email: Email pemilih.
- hasVoted: Status apakah pemilih sudah memberikan suara.
- passwordHash: Hash dari kata sandi pemilih.
- lastUpdated: Waktu terakhir data diperbarui.
- isActive: Status apakah pemilih aktif.

b. Candidate

```
struct Candidate {
    uint256 id;
    string name;
    string visi;
    string misi;
    uint256 voteCount;
    uint256 lastUpdated;
}
```

Struktur data Candidate digunakan untuk menyimpan data kandidat. Data yang disimpan berupa data.

- id: ID kandidat.
- name: Nama kandidat.
- visi: Visi kandidat.
- misi: Misi kandidat.
- voteCount: Jumlah suara yang diperoleh.
- lastUpdated: Waktu terakhir data diperbarui.

c. VoterHistory

```
struct VoterHistory {
    uint256 id;
    string name;
    string email;
    bool hasVoted;
    bytes32 passwordHash;
    uint256 timestamp;
}
```

Struktur data VoterHistory digunakan untuk menyimpan riwayat perubahan data pemilih. Data yang disimpan sama seperti data pada struktur data Voter namun pada VoterHistory terdapat tambahan timestamp utnuk menyimpan waktu perubahan data.

d. CandidateHistory

```
struct CandidateHistory {
    uint256 id;
    string name;
    string visi;
    string misi;
    uint256 voteCount;
    uint256 timestamp;
}
```

Struktur data CandidateHistory digunakan untuk menyimpan riwayat perubahan data kandidat. Data yang disimpan sama seperti data pada struktur data Candidate namun pada CandidateHistory terdapat tambahan timestamp utnuk menyimpan waktu perubahan data.

e. VoteCountHistory

```
struct VoteCountHistory {
    uint256 candidateId;
    uint256 voteCount;
    uint256 timestamp;
}
```

Struktur data VoteCountHistory digunakan untuk menyimpan riwayat jumlah suara yang diperoleh oleh kandidat. Data yang disimpan berupa data.

- candidateId: ID kandidat.
- voteCount: Jumlah suara.
- timestamp: Waktu perubahan data.

3. Event

Event digunakan untuk mencatat aktivitas atau perubahan yang terjadi selama eksekusi kontrak, yang dapat dideteksi oleh aplikasi yang berjalan di luar blockchain. Misalnya, ketika seorang pemilih memberikan suaranya, event dapat digunakan untuk mencatat tindakan tersebut sehingga aplikasi front-end dapat memperbarui antarmuka pengguna dengan informasi terbaru. Berikut adalah event yang digunakan pada smart contract

```
event VoterAdded(uint256 id, string name);
event VoterUpdated(uint256 id, string name);
event VoterDeleted(uint256 id, string name);
event CandidateAdded(uint256 id, string name);
```

```
    event CandidateUpdated(uint256 id, string name);
    event CandidateDeleted(uint256 id, string name);
    event ElectionEnded();
```

Fungsi dari masing-masing event diatas yaitu:

- VoterAdded: Mencatat dan menyimpan informasi saat seorang pemilih baru ditambahkan ke dalam sistem pemilihan.
- VoterUpdated: Mencatat dan menyimpan informasi saat data pemilih diperbarui.
- VoterDeleted: Mencatat dan menyimpan informasi saat seorang pemilih dihapus dari sistem pemilihan.
- CandidateAdded: Mencatat dan menyimpan informasi saat seorang kandidat baru ditambahkan ke dalam sistem pemilihan.
- CandidateUpdated: Mencatat dan menyimpan informasi saat data kandidat diperbarui.
- CandidateDeleted: Mencatat dan menyimpan informasi saat seorang kandidat dihapus dari sistem pemilihan.
- ElectionEnded: Mencatat dan menyimpan informasi saat pemilihan berakhir.
- Parameter: Tidak ada parameter, hanya menandakan bahwa pemilihan telah selesai.

4. State Variable

State variable adalah variabel yang disimpan di blockchain dan mewakili status kontrak yang dapat berubah seiring waktu. Dalam konteks e-voting, state variable bisa mencakup daftar pemilih yang telah mendaftar, total suara yang diterima oleh setiap kandidat, dan status apakah pemilihan masih berlangsung atau sudah ditutup. Berikut state variabel yang digunakan pada smart contract sesuai dengan desain.

```
    bool public electionActive;
    uint256[] public voterIds;
    uint256[] public candidateIds;

    mapping(uint256 => Voter) public voters;
```

```
    mapping(uint256 => Candidate) public candidates;
    mapping(uint256 => VoterHistory[]) public voterHistories;
    mapping(uint256 => CandidateHistory[]) public
candidateHistories;
    mapping(uint256 => VoteCountHistory[]) public
voteCountHistories;
```

Fungsi dari masing-masing state variabel diatas yaitu

- Variabel electionActive: Menyimpan status apakah pemilihan sedang aktif atau tidak.
- Array voterIds: Menyimpan daftar ID semua pemilih yang terdaftar dalam pemilihan.
- Array candidateIds: Menyimpan daftar ID semua kandidat yang terdaftar dalam pemilihan.
- Mapping voters: Menyimpan informasi tentang pemilih. Mapping ini mengasosiasikan ID pemilih (tipe uint256) dengan struktur data Voter.
- Mapping candidates: Menyimpan informasi tentang kandidat. Mapping ini mengasosiasikan ID kandidat (tipe uint256) dengan struktur data Candidate.
- Mapping voterHistories: Menyimpan riwayat perubahan data pemilih. Mapping ini mengasosiasikan ID pemilih (tipe uint256) dengan array dari struktur data VoterHistory.
- Mapping candidateHistories: Menyimpan riwayat perubahan data kandidat. Mapping ini mengasosiasikan ID kandidat (tipe uint256) dengan array dari struktur data CandidateHistory.
- Mapping voteCountHistories: Menyimpan riwayat jumlah suara yang diterima oleh setiap kandidat. Mapping ini mengasosiasikan ID kandidat (tipe uint256) dengan array dari struktur data VoteCountHistory.

5. Modifier

Modifier digunakan untuk mengubah perilaku fungsi lain. Modifier sering digunakan untuk menerapkan aturan akses atau pra-kondisi sebelum eksekusi fungsi tertentu. Berikut modifier yang digunakan pada smart contract sesuai dengan desain.

```
    modifier onlyActiveElection() {
```

```
        require(electionActive, "Election is not active");
        _;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only contract owner can
call this function");
        _;
    }
```

Fungsi dari modifier diatas yaitu

- onlyActiveElection: Memastikan bahwa pemilihan sedang aktif sebelum eksekusi fungsi berlanjut. Ini berguna untuk mencegah operasi yang tidak diinginkan saat pemilihan tidak aktif.
- onlyOwner: Memastikan bahwa hanya pemilik kontrak yang dapat memanggil fungsi tertentu, memberikan kontrol akses dan keamanan.

6. Fungsi

Fungsi menjalankan logika tertentu saat dipanggil. Fungsi-fungsi ini mengimplementasikan aturan dan alur kerja yang telah ditetapkan untuk memastikan integritas dan keandalan proses pemilihan. Berikut fungsi yang digunakan pada smart contract sesuai dengan desain

- addVoter

```
function addVoter(uint256 _id, string memory _name, string
memory _email, string memory _password) public onlyOwner {
    require(voters[_id].id == 0, "Voter already
registered");

    voters[_id] = Voter({
        id: _id,
        name: _name,
        email: _email,
        hasVoted: false,
        passwordHash:
keccak256(abi.encodePacked(_password)),
        lastUpdated: block.timestamp,
```

```
        isActive: true
    });

    voterIds.push(_id);
    emit VoterAdded(_id, _name);
}
```

Fungsi diatas digunakan untuk Menambahkan pemilih baru ke dalam sistem menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang bisa memanggil fungsi ini. Fungsi akan memeriksa apakah pemilih sudah terdaftar. Jika belum, data pemilih akan ditambahkan ke dalam mapping voters, array voterIds, dan event VoterAdded.

- updateVoter

```
function updateVoter(uint256 _id, string memory _name,
string memory _email, string memory _password) public
onlyOwner {
    require(voters[_id].id != 0, "Voter not registered");

    // Record the current state to history before updating
    voterHistories[_id].push(VoterHistory({
        id: _id,
        name: voters[_id].name,
        email: voters[_id].email,
        hasVoted: voters[_id].hasVoted,
        passwordHash: voters[_id].passwordHash,
        timestamp: block.timestamp
    }));

    voters[_id].name = _name;
    voters[_id].email = _email;
    voters[_id].passwordHash =
keccak256(abi.encodePacked(_password));
    voters[_id].lastUpdated = block.timestamp;

    emit VoterUpdated(_id, _name);
```

```
}
```

Fungsi diatas digunakan untuk memperbarui informasi pemilih yang sudah terdaftar. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi akan memeriksa apakah pemilih sudah terdaftar. Jika sudah, data pemilih akan diperbarui dalam mapping voters, riwayat pemilih akan disimpan dalam voterHistories, dan event VoterUpdated akan dijalankan.

- deleteVoter

```
function deleteVoter(uint256 _id) public onlyOwner {
    require(voters[_id].id != 0, "Voter not registered");

    // Record the current state to history before deleting
    voterHistories[_id].push(VoterHistory({
        id: _id,
        name: voters[_id].name,
        email: voters[_id].email,
        hasVoted: voters[_id].hasVoted,
        passwordHash: voters[_id].passwordHash,
        timestamp: block.timestamp
    }));

    voters[_id].isActive = false; // Mark voter as inactive
instead of deleting

    emit VoterDeleted(_id, voters[_id].name);
}
```

Fungsi di atas digunakan untuk menandai pemilih sebagai tidak aktif. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi akan memeriksa apakah pemilih sudah terdaftar. Jika sudah, data pemilih akan ditandai sebagai tidak aktif dalam mapping voters, riwayat pemilih akan disimpan dalam voterHistories, dan event VoterDeleted akan dijalankan.

- addCandidate

```
function addCandidate(uint256 _id, string memory _name,
string memory _visi, string memory _misi) public onlyOwner
{
    require(candidates[_id].id == 0, "Candidate already
exists");

    candidates[_id] = Candidate({
        id: _id,
        name: _name,
        visi: _visi,
        misi: _misi,
        voteCount: 0,
        lastUpdated: block.timestamp
    });

    candidateIds.push(_id);
    emit CandidateAdded(_id, _name);
}
```

Fungsi diatas digunakan untuk menambahkan kandidat baru ke dalam sistem. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi akan memeriksa apakah kandidat sudah terdaftar. Jika belum, data kandidat akan ditambahkan ke dalam mapping candidates, array candidateIds, dan event CandidateAdded akan dijalankan.

- updateCandidate

```
function updateCandidate(uint256 _id, string memory _name,
string memory _visi, string memory _misi) public onlyOwner
{
require(candidates[_id].id != 0, "Candidate not
registered");

    // Record the current state to history before updating
    candidateHistories[_id].push(CandidateHistory({
        id: _id,
        name: candidates[_id].name,
        visi: candidates[_id].visi,
```

```
        misi: candidates[_id].misi,
        voteCount: candidates[_id].voteCount,
        timestamp: block.timestamp
    }));

    candidates[_id].name = _name;
    candidates[_id].visi = _visi;
    candidates[_id].misi = _misi;
    candidates[_id].lastUpdated = block.timestamp;

    emit CandidateUpdated(_id, _name);
}
```

Fungsi diatas digunakan untuk memperbarui informasi kandidat yang sudah terdaftar. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi akan memeriksa apakah kandidat sudah terdaftar. Jika sudah, data kandidat akan diperbarui dalam mapping candidates, riwayat kandidat akan disimpan dalam candidateHistories, dan event CandidateUpdated akan dijalankan.

- deleteCandidate

```
function deleteCandidate(uint256 _id) public onlyOwner {
    require(candidates[_id].id != 0, "Candidate not
registered");

    emit CandidateDeleted(_id, candidates[_id].name);

    // Remove candidate from candidateIds array
    for (uint256 i = 0; i < candidateIds.length; i++) {
        if (candidateIds[i] == _id) {
            candidateIds[i] =
candidateIds[candidateIds.length - 1];
            candidateIds.pop();
            break;
        }
    }
```

```
    delete candidates[_id];
}
```

Fungsi diatas digunakan untuk menghapus kandidat dari sistem. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi akan memeriksa apakah kandidat sudah terdaftar. Jika sudah, data kandidat akan dihapus dari mapping candidates, array candidateIds, dan event CandidateDeleted akan dipancarkan.

- login

```
function login(uint256 _voterId, string memory _password)
public view returns (string memory, bool) {
    Voter memory voter = voters[_voterId];
    if (voter.id != 0 && voter.passwordHash ==
keccak256(abi.encodePacked(_password))) {
        return (voter.name, true);
    } else {
        return ("", false);
    }
}
```

Fungsi diatas digunakan untuk memverifikasi kredensial pemilih untuk login. Fungsi akan memeriksa apakah ID pemilih valid dan apakah kata sandi yang diberikan sesuai dengan hash kata sandi yang tersimpan. Jika valid, fungsi ini akan mengembalikan nama pemilih dan status login berhasil (true). Jika tidak, fungsi ini akan mengembalikan string kosong dan status login gagal (false)

- isVoterEligible

```
function isVoterEligible(uint256 _voterId) public view
returns (bool) {
    return voters[_voterId].id != 0 &&
!voters[_voterId].hasVoted;
}
```

Fungsi diatas digunakan untuk memeriksa apakah pemilih memenuhi syarat untuk memberikan suara. Fungsi akan memeriksa apakah ID pemilih

valid dan apakah pemilih belum memberikan suara, kemudian mengembalikan nilai boolean yang menunjukkan kelayakan pemilih

- vote

```
function vote(uint256 _voterId, uint256 _candidateId,
string memory _password) public onlyActiveElection {
    require(voters[_voterId].id != 0, "Voter not
registered");
    require(!voters[_voterId].hasVoted, "Voter already
voted");
    require(candidates[_candidateId].id != 0, "Candidate
not found");
    require(voters[_voterId].passwordHash ==
keccak256(abi.encodePacked(_password)), "Invalid
password");

    // Record the current state to history before voting
    voterHistories[_voterId].push(VoterHistory({
        id: _voterId,
        name: voters[_voterId].name,
        email: voters[_voterId].email,
        hasVoted: voters[_voterId].hasVoted,
        passwordHash: voters[_voterId].passwordHash,
        timestamp: block.timestamp
    }));

    candidates[_candidateId].voteCount++;

    // Record vote count history

voteCountHistories[_candidateId].push(VoteCountHistory({
        candidateId: _candidateId,
        voteCount: candidates[_candidateId].voteCount,
        timestamp: block.timestamp
    }));

    voters[_voterId].hasVoted = true;
    voters[_voterId].lastUpdated = block.timestamp;
```

```
}
```

Fungsi diatas digunakan untuk mengizinkan pemilih memberikan suara kepada kandidat. Fungsi menggunakan modifier onlyActiveElection sehingga fungsi hanya dapat dipanggil saat pemilihan aktif. Fungsi akan memeriksa apakah pemilih dan kandidat terdaftar, apakah pemilih belum memberikan suara, dan apakah kata sandi pemilih valid. Jika valid, data pemilih dan kandidat akan diperbarui dalam mapping voters dan candidates, riwayat pemilih dan jumlah suara akan disimpan dalam voterHistories dan voteCountHistories.

- endElection

```
 function endElection() public onlyOwner {
    electionActive = false;
    emit ElectionEnded();
}
```

Fungsi diatas digunakan untuk mengakhiri pemilihan. Fungsi menggunakan modifier onlyOwner sehingga hanya pemilik kontrak yang dapat memanggilnya. Fungsi ini akan mengubah status pemilihan menjadi tidak aktif dan menjalankan event ElectionEnded.

- getVoteCount

```
function getVoteCount(uint256 _candidateId) public view
returns (uint256) {
    require(candidates[_candidateId].id != 0, "Candidate
not found");
    return candidates[_candidateId].voteCount;
}
```

Fungsi diatas digunakan untuk mengambil jumlah suara untuk kandidat tertentu. Fungsi akan memeriksa apakah kandidat terdaftar. Jika ya, fungsi ini akan mengembalikan jumlah suara kandidat tersebut.

- getAllVoters

```
function getAllVoters() public view returns (Voter[]
memory) {
    uint256 activeCount = 0;
```

```
    // Count active voters
    for (uint256 i = 0; i < voterIds.length; i++) {
        if (voters[voterIds[i]].isActive) {
            activeCount++;
        }
    }

    Voter[] memory allVoters = new Voter[](activeCount);
    uint256 index = 0;
    for (uint256 i = 0; i < voterIds.length; i++) {
        uint256 voterId = voterIds[i];
        if (voters[voterId].isActive) {
            allVoters[index] = voters[voterId];
            index++;
        }
    }
    return allVoters;
}
```

Fungsi diatas digunakan untuk mengambil daftar semua pemilih aktif. Fungsi ini akan menghitung jumlah pemilih aktif, membuat array baru untuk menyimpan pemilih aktif, dan mengembalikan array pemilih yang aktif.

- getAllCandidates

```
function getAllCandidates() public view returns
(Candidate[] memory) {
    Candidate[] memory allCandidates = new
Candidate[](candidateIds.length);
    uint256 index = 0;
    for (uint256 i = 0; i < candidateIds.length; i++) {
        uint256 candidateId = candidateIds[i];
        if (candidates[candidateId].id != 0) {
            allCandidates[index] = candidates[candidateId];
            index++;
        }
    }
    return allCandidates;
```

```
}
```

Fungsi diatas digunakan untuk mengambil daftar semua kandidat. Fungsi akan membuat array baru untuk menyimpan kandidat yang terdaftar dan mengembalikan array kandidat yang terdaftar.

- getAllVoterHistories

```
function getAllVoterHistories() public view returns
(VoterHistory[] memory) {
    uint256 totalHistories = 0;

    // Count total histories
    for (uint256 i = 0; i < voterIds.length; i++) {
        totalHistories +=
voterHistories[voterIds[i]].length + 1; // +1 for initial
state
    }

    VoterHistory[] memory allHistories = new
VoterHistory[](totalHistories);
    uint256 index = 0;

    // Gather all histories
    for (uint256 i = 0; i < voterIds.length; i++) {
        uint256 voterId = voterIds[i];

        // Add initial state
        allHistories[index] = VoterHistory({
            id: voters[voterId].id,
            name: voters[voterId].name,
            email: voters[voterId].email,
            hasVoted: voters[voterId].hasVoted,
            passwordHash: voters[voterId].passwordHash,
            timestamp: voters[voterId].lastUpdated
        });
        index++;
```

```
        // Add historical states
        VoterHistory[] memory histories =
voterHistories[voterId];
        for (uint256 j = 0; j < histories.length; j++) {
            allHistories[index] = histories[j];
            index++;
        }
    }

    return allHistories;
}
```

Fungsi diatas digunakan untuk mengambil semua riwayat pemilih. Fungsi akan menghitung jumlah total riwayat pemilih, mengumpulkan semua riwayat ke dalam array baru, dan mengembalikan array riwayat pemilih.

- getAllCandidateHistories

```
function getAllCandidateHistories() public view returns
(CandidateHistory[] memory) {
    uint256 totalHistories = 0;

    // Count total histories
    for (uint256 i = 0; i < candidateIds.length; i++) {
        totalHistories +=
candidateHistories[candidateIds[i]].length + 1; // +1 for
initial state
    }

    CandidateHistory[] memory allHistories = new
CandidateHistory[](totalHistories);
    uint256 index = 0;

    // Gather all histories
    for (uint256 i = 0; i < candidateIds.length; i++) {
        uint256 candidateId = candidateIds[i];

        // Add initial state
        allHistories[index] = CandidateHistory({
```

```
            id: candidates[candidateId].id,
            name: candidates[candidateId].name,
            visi: candidates[candidateId].visi,
            misi: candidates[candidateId].misi,
            voteCount: candidates[candidateId].voteCount,
            timestamp: candidates[candidateId].lastUpdated
        });
        index++;

        // Add historical states
        CandidateHistory[] memory histories =
candidateHistories[candidateId];
        for (uint256 j = 0; j < histories.length; j++) {
            allHistories[index] = histories[j];
            index++;
        }
    }

    return allHistories;
}
```

Fungsi diatas digunakan untuk mengambil semua riwayat kandidat. Fungsi akan menghitung jumlah total riwayat kandidat, mengumpulkan semua riwayat ke dalam array baru, dan mengembalikan array riwayat kandidat.

- getAllVoteCountHistories

```
function getAllVoteCountHistories() public view returns
(VoteCountHistory[] memory) {
    uint256 totalHistories = 0;

    // Count total histories
    for (uint256 i = 0; i < candidateIds.length; i++) {
        totalHistories +=
voteCountHistories[candidateIds[i]].length;
    }
```

```
    VoteCountHistory[] memory allHistories = new
VoteCountHistory[](totalHistories);
    uint256 index = 0;

    // Gather all histories
    for (uint256 i = 0; i < candidateIds.length; i++) {
        uint256 candidateId = candidateIds[i];
        VoteCountHistory[] memory histories =
voteCountHistories[candidateId];
        for (uint256 j = 0; j < histories.length; j++) {
            allHistories[index] = histories[j];
            index++;
        }
    }

    return allHistories;
}
```

Fungsi diatas digunakan untuk mengambil semua riwayat jumlah suara. Fungsi ini akan menghitung jumlah total riwayat jumlah suara, mengumpulkan semua riwayat ke dalam array baru, dan mengembalikan array riwayat jumlah suara.

Setelah melakukan pembuatan smart contract, akan dilakukan deployment. Pada penelitian ini deployment smart contact menggunakan RPC hardhat yang melakukan pengembangan ke jaringan local yang telah berjalan pada port yang ditentukan. Berikut adalah kode yang digunakan untuk pengembangan.

```
async function main() {
    const Election = await
hre.ethers.getContractFactory("Election");
    const election_ = await Election.deploy();

    await election_.deployed();

    console.log(`Contract Address: ${election_.address}`);
}
```

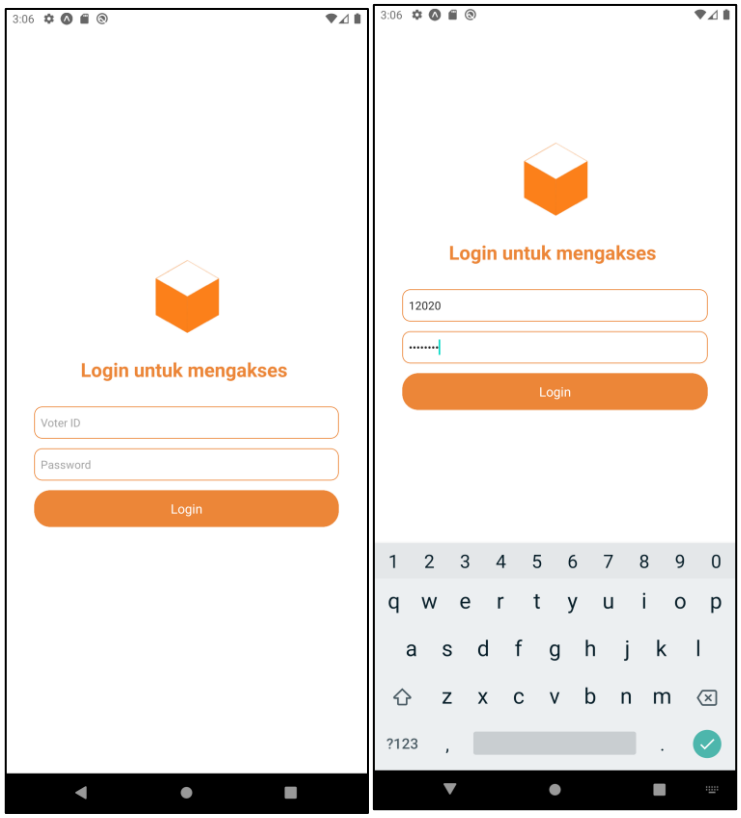
Pengembangan smart contract telah dilakukan. Langkah selanjutnya yaitu membuat RESTful API yang menghubungkan aplikasi mobile dengan kontrak pintar di blockchain Ethereum. Pada penelitian ini pengembangan API menggunakan Express.js. Membuat endpoint yang dapat mengakses smart contract sehingga dapat menghubungkan aplikasi dengan jaringan blockchain. Pada penelitian ini

5.4 Implementasi Antarmuka Pengguna

Dalam sebuah sistem e-voting tampilan pengguna (user interface) memainkan peran penting dalam memastikan kemudahan penggunaan bagi para pemilih. Sub bab ini akan menjelaskan bagaimana tampilan pengguna diimplementasikan dalam sistem.

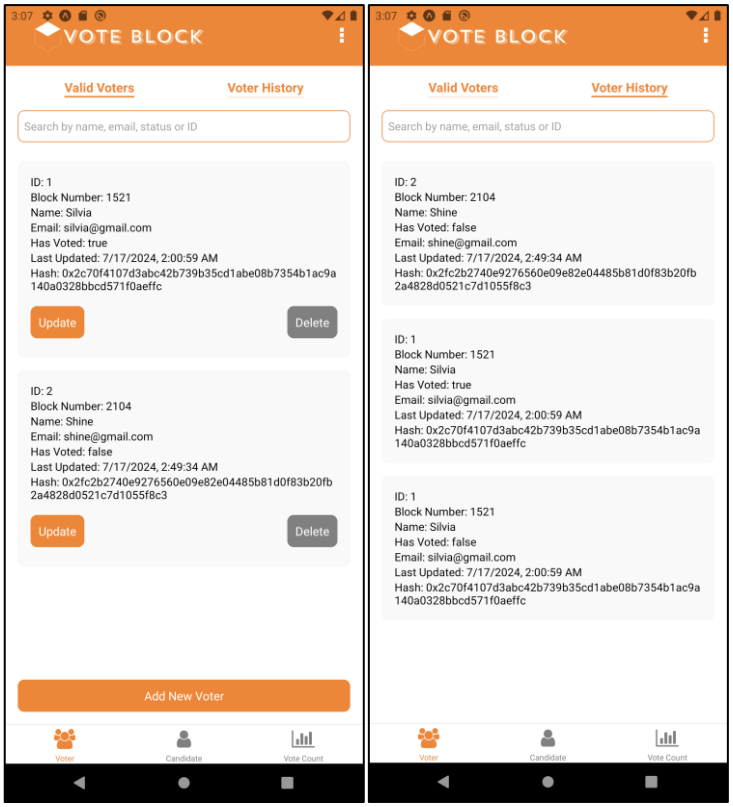
a. Tampilan Halaman Login

Halaman Login mempilkan bidang untuk memasukkan id dan password. Tampilan halaman login sama untuk Admin maupun User

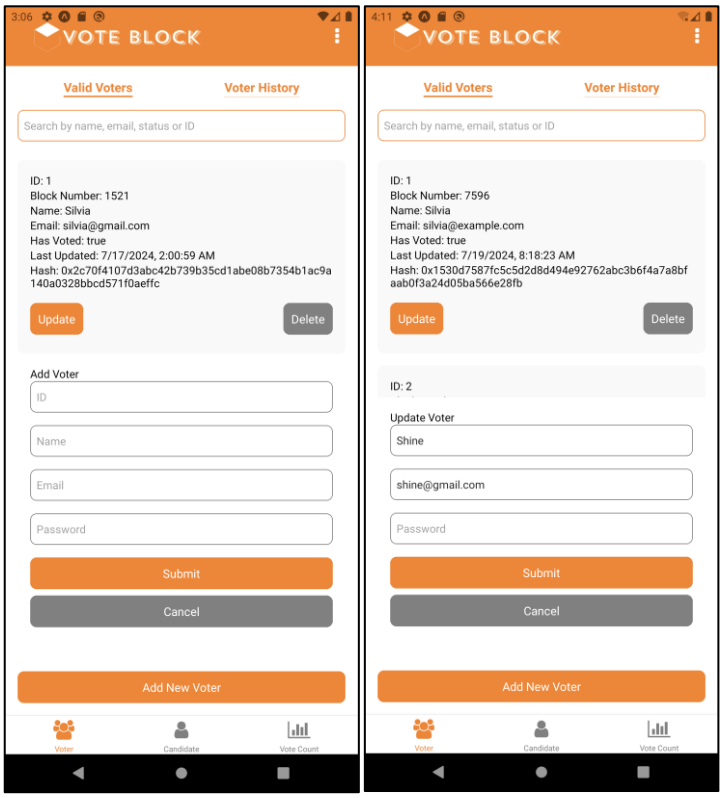


Gambar 5. 1 Implementasi Halaman Login

b. Tampilan Halaman Voters Admin



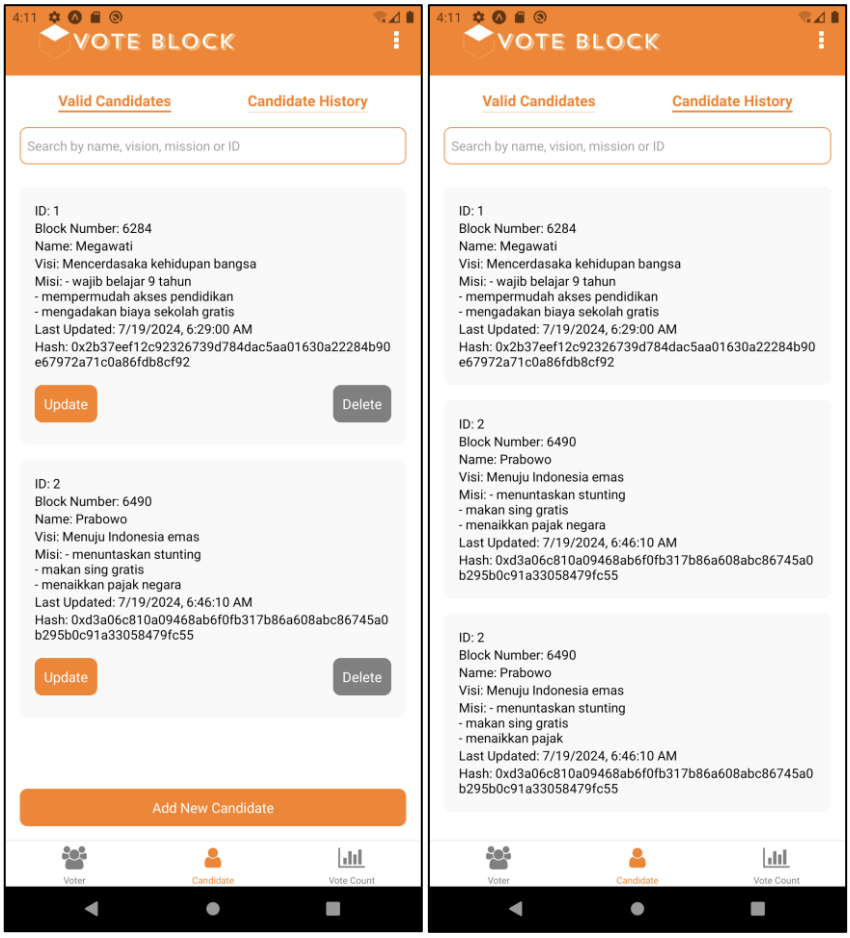
Gambar 5. 2 Implementasi Halaman Voters



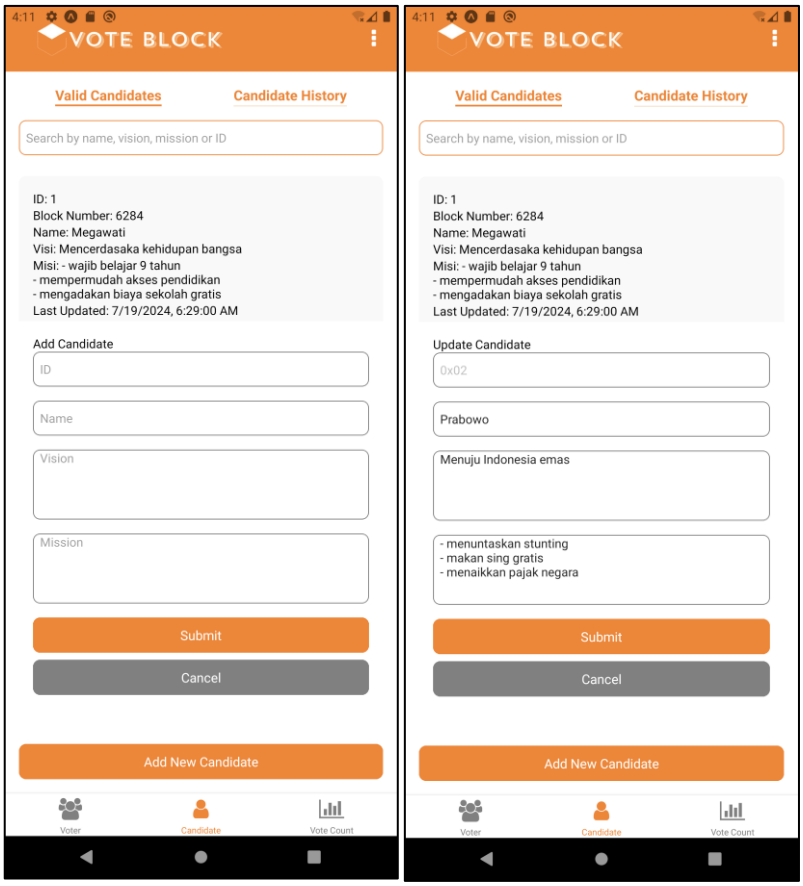
Gambar 5. 3 Implementasi Tambah dan Edit Data Pemilih

Halaman Voter menampilkan data voter serta beberapa tombol untuk mengolahan data. Tab Valid Voter dan Voter History digunakan untuk berpindah halaman dari tampilan yang hanya menampilkan data Pemilih yang valid ke data history seluruh data Pemilih baik yang valid maupun tidak dan sebaliknya.

c. Tampilan Halaman Candidate Admin



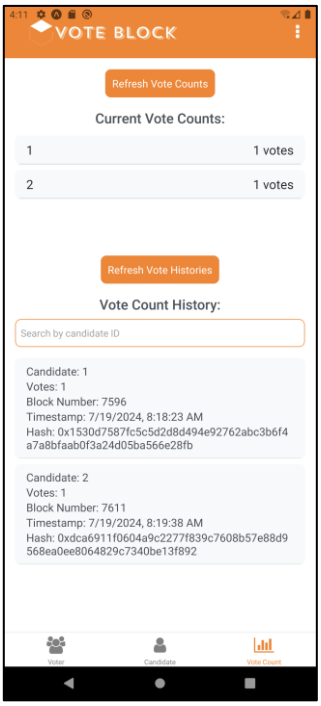
Gambar 5. 4 Implemetasi Halaman Candidates



Gambar 5. 5Implementasi Halaman Tambah dan Edit data Kandidat

Halaman Candidate menampilkan data kandidat serta beberapa tombol untuk mengolahan data. Tab Valid Candidate dan Candidate History digunakan untuk berpindah halaman dari tampilan yang hanya menampilkan data Pemilih yang valid ke data history seluruh data Pemilih baik yang valid maupun tidak dan sebaliknya.

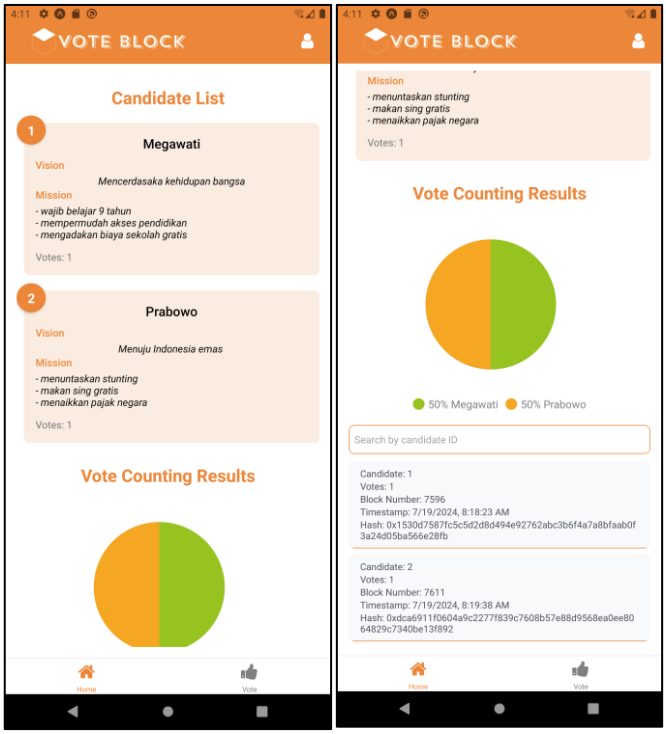
d. Tampilan Halaman Vote Count Admin



Gambar 5. 6 Implementasi Halaman Vote Count

Halaman Vote Count Admin berisi hasil pemilihan dan Vote history yang menyimpan setiap terdapat suara yang masuk. Halaman vote count dapat dilakukan refresh untuk melihat data terbaru

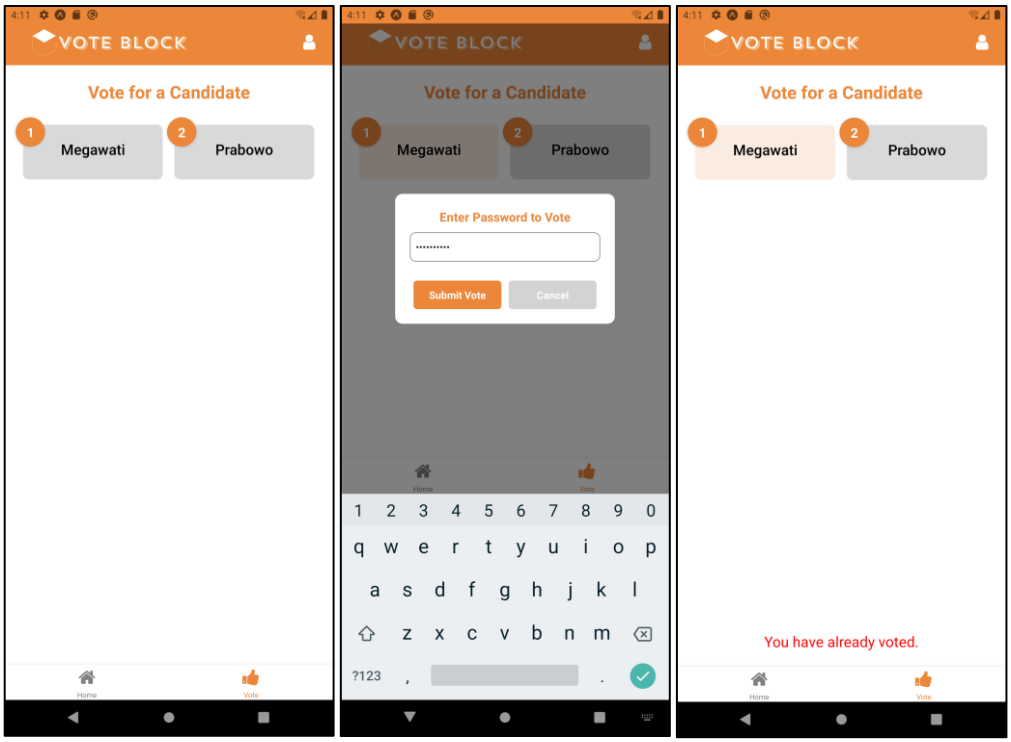
e. Tampilan Halaman Home User



Gambar 5. 7 Implementasi Halaman Home User

Halaman Home User menampilkan data Kandidat yang terdapat dalam pemilihan serta menampilkan hasil pemilihan suara terkini. Pada halaman home juga terdapat fitur search untuk mrncari data suara yang masuk melalui id kandidat.

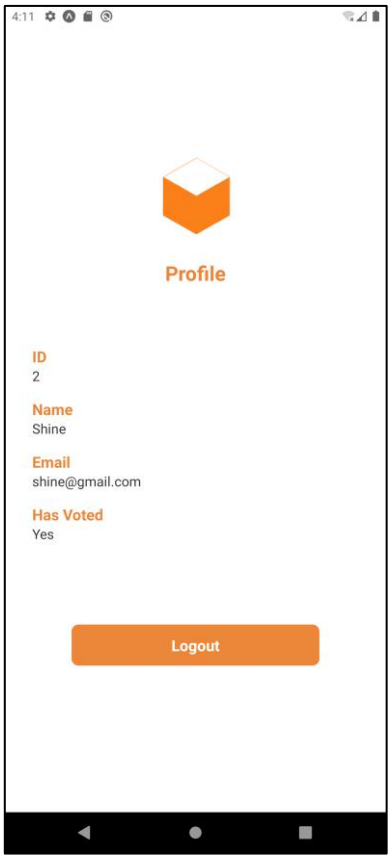
f. Tampilan Halaman Vote



Gambar 5. 8 Implementasi Halaman Vote User

Halaman Vote User digunakan untuk melakukan pemungutan suara. User dpaat memilih card dengan nama kandidat yang dipilih kemudian dapat memasukkan password untuk konfirmasi. Setelah melakukan vote maka halaman akan dibatasi dan tidak dapat melakukan pemilihan kembali

g. Tampilan Halaman Profile User



Gambar 5. 9 Implementasi Halaman Profile User

Halaman Profile menampilkan profil pengguna yang saat ini sedang aktif, Pada halaman profile juga terdapat tombol logout yang dapat digunakan untuk keluar dari state dan kembali ke halaman login

5.5 Pengujian Jaringan Blockchain

Pengujian pada jaringan blockchain dilakukan dengan menjalankan jaringan blockchain dan mengecek apakah jaringan berjalan dengan baik, serta memeriksa kemungkinan adanya akses lain kedalam blockchain. Berikut hasil dari pengujian jaringan blockchain.

Tabel 5. 1 Pengujian Jaringan

Test Case	Skenario Pengujian	*Input*	Hasil yang diharapkan	Hasil
Menguji keamanan jaringan	Menambahkan data ketika node validator mati	Mematikan node validator dan mencoba menambahkan data	Terjadi error dan data tidak dapat ditambahkan	Sesuai

	Mengakses jaringan blockchain menggunakan node yang tidak bergabung dalam jaringan blockchain	Mengganti private key menggunaka private key account yang tidak terdaftar pada jaringan	Terjari error dan tidak dapat menambahkan data	Sesuai
	Menambahkan data menggunakan node yang bukan node validator	Mengganti private key menggunakan private key node member	Terjadi error dan tidak dapat menambahkan data	Sesuai
Menguji transparasi jaringan	Mengakses data menggunakan node yang bukan node validator	Mengganti private key menggunakan private key node member	Data dapat ditampilkan	Sesuai
	Mengakses data menggunakan node member saat node validator dimatikan	Mematikan node validator dan mencoba mengakses data menggunakan private key node member	Data dapat ditampilkan	Sesuai

5.1 Pengujian Smart Contract

Pengujian pada smart contract dilakukan menggunakan hardhat test yang dijalankan dengan beberapa fungsi dengan beberapa scenario untuk membuktikan smart contract berjalan dengan baik. Berikut hasil dari pengujian smart contract.

Tabel 5. 2 Pengujian Smart Contract

No	Skenario Pengujian	*Input*	Hasil yang diharapkan	Hasil
1	Memverifikasi pemilik kontrak saat deployment	Deploy kontrak	Pemilik adalah address owner	Sesuai
2	Memverifikasi status pemilihan saat deployment	Deploy kontrak	Status pemilihan aktif	Sesuai
3	Menambahkan pemilih baru	addVoter(1, "Alice",	Pemilih dengan ID 1 ditambahkan	Sesuai

		"alice@example.com", "password")		
4	Menambahkan pemilih dengan ID yang sudah ada	addVoter(1, "Alice", "alice@example.com", "password"), addVoter(1, "Bob", "bob@example.com", "password")	Revert dengan pesan "Voter already registered"	Sesuai
5	Memperbarui pemilih yang sudah ada	addVoter(1, "Alice", "alice@example.com", "password"), updateVoter(1, "Alice Smith", "alice.smith@example.com", "newpassword")	Data pemilih diperbarui	Sesuai
6	Menghapus pemilih	addVoter(1, "Alice", "alice@example.com", "password"), deleteVoter(1)	Pemilih dihapus (isActive = false)	Sesuai
7	Menambahkan kandidat baru	addCandidate(1, "Bob", "Visi Bob", "Misi Bob")	Kandidat dengan ID 1 ditambahkan	Sesuai
8	Menambahkan kandidat dengan ID yang sudah ada	addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), addCandidate(1, "Charlie", "Visi Charlie", "Misi Charlie")	Revert dengan pesan "Candidate already exists"	Sesuai
9	Memperbarui kandidat yang sudah ada	addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), updateCandidate(	Data kandidat diperbarui	Sesuai

		1, "Bob Smith", "Visi Bob Smith", "Misi Bob Smith")		
10	Menghapus kandidat	addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), deleteCandidate(1)	Kandidat dihapus (isActive = false)	Sesuai
11	Memilih kandidat oleh pemilih	addVoter(1, "Alice", "alice@example.com", "password"), addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), vote(1, 1, "password")	Pemilih berhasil memberikan suara	Sesuai
12	Memilih lebih dari sekali oleh pemilih	addVoter(1, "Alice", "alice@example.com", "password"), addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), vote(1, 1, "password"), vote(1, 1, "password")	Revert dengan pesan "Voter has already voted"	Sesuai
13	Memilih dengan password yang salah	addVoter(1, "Alice", "alice@example.com", "password"), addCandidate(1, "Bob", "Visi Bob", "Misi Bob"), vote(1, 1, "wrongpassword")	Revert dengan pesan "Invalid password"	Sesuai

5.6 Pengujian Fungsional Sistem

Pengujian fungsional sistem dilakukan dengan menjalankan setiap fitur sesuai dengan fungsi yang diharapkan serta memeriksa kelengkapan hasil yang ditampilkan. Skenario pengujian fungsionalitas sistem dibagi menjadi dua bagian, yaitu pengujian fungsional fitur untuk admin dan pengujian fungsional fitur untuk user (pemilih). Rincian pengujian ini ditunjukkan mulai dari tabel . hingga tabel . sebagai berikut:

Tabel 5. 3 Pengujian Fungsional Sistem Admin

No	Skenario Pengujian	Input	Hasil yang diharapkan	Hasil
1	Login sistem	Username benar dan Password benar	Berhasil login ke dalam sistem	Sesuai
		Username benar dan Password salah	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai
		Username salah dan Password benar	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai
		Username salah dan Password salah	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai
2	Mengelola data Pemilih	Menambahkan data Pemilih baru	Berhasil menambahkan data, halaman menampilkan Alert Success Voter added successfully	Sesuai
		Menambahkan data pemilih dengan mengosongkan salah satu bidang	Gagal menambahkan data. Halaman menampilkannValidation Error, all fields must be filled	Sesuai

		Menambahkan data Pemilih baru menggunakan id yang telah digunakan	Gagal menambahkan data, Halaman menampilkan keterangan Error, Voter ID already exists	Sesuai
		Melakukan update data Pemilih	Berhasil mengupdate data. Halaman menampilkan Alert Success Voter updated successfully	Sesuai
		Melakukan delete data Pemilih	Berhasil menghapus data. Halmana menampilkan alert Success voter deleted successfully	Sesuai
		Melihat History data Voter	Halaman menampilkan seluruh history data voter baik yang valid maupun tidak valid	Sesuai
3	Mengelola data Kandidat	Menambahkan data Kandidat baru	Berhasil menambahkan data, halaman menampilkan Alert Success Candidate added successfully	Sesuai
		Menambahkan data Kandidat dengan mengosongkan salah satu bidang	Gagal menambahkan data. Halaman menampilkannValidation Error, all fields must be filled	Sesuai
		Menambahkan data Kandidat baru menggunakan id yang telah digunakan	Gagal menambahkan data, Halaman menampilkan keterangan error	Sesuai
		Melakukan update data Kandidat	Berhasil mengupdate data. Halaman menampilkan Alert	Sesuai

			Success Candidate updated successfully	
		Melakukan delete data Kandidat	Berhasil menghapus data. Halmana menampilkan alert Success Candidate deleted successfully	Sesuai
		Melihat history data Kandidat	Halaman menampilkan seluruh history data candidate baik yang valid maupun tidak valid	Sesuai
4	Melihat hasil pemilihan	Mengakses halaman vote count dan melakukan refresh untuk melihat data terbaru	Halaman menampilkan hasil pemilihan terkini	Sesuai
5	Mehilat history pemilihan	Mengakses halaman vote count dan melakukan refresh untuk melihat data history pemilihan	Halaman menampilkan history data pemilihan	Sesuai

Tabel 5. 4 Pengujian Fungsional Sistem User

No	Skenario Pengujian	Input	Hasil yang diharapkan	Hasil
1	Login sistem	Username benar dan Password benar	Berhasil login ke dalam sistem	Sesuai
		Username benar dan Password salah	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai
		Username salah dan Password benar	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai

		Username salah dan Password salah	Login gagal, halaman menampilkan *Alert Login Error Invalid Credentials*	Sesuai
2	Melihat kandidat dalam pemilihan	Mengakses halaman home	Halaman Home menampilkan daftar Kandidat	Sesuai
3	Melakukan pemilihan	Memilih kandidat dan memasukkan password user untuk memilih	Berhasil melakukan pemilihan. Halaman akan dibatasi dan tidak daoat melakukan pemilihan kembali	Sesuai
		Memilih kandidat dan memasukkan password user yang salah	Gagal melakukan pemilihan. Halman menampilkan alert Error failed to vote.	Sesuai
4	Melihat hasil pemilihan	Mengakses halaman Home	Halaman Home menampilkan hasil pemilihan terkini	Sesuai
5	Melihat history pemilihan	Mengakses halaman Home	Halaman home menampilkan history pemilihan	Sesuai

BAB VI. HASIL DAN PEMBAHASAN

6.1 Hasil Penelitian

Penelitian ini berfokus pada pengembangan dan pengujian sistem e-voting berbasis blockchain dengan menggunakan Go Ethereum dan berbagai komponen lainnya. Berikut adalah hasil yang diperoleh dari penelitian ini:

1. Pengujian Fungsionalitas Sistem
    - Sistem e-voting berhasil menjalankan semua fungsi utamanya, termasuk login, melihat kandidat, melakukan pemilihan, dan melihat hasil serta riwayat pemilihan. Semua skenario pengujian fungsional menunjukkan hasil yang sesuai dengan harapan.
    - Pengujian login menunjukkan bahwa sistem mampu membedakan antara kredensial yang valid dan tidak valid, serta memberikan umpan balik yang sesuai.
    - Pengujian pemilihan menunjukkan bahwa sistem berhasil memproses pemilihan dan mencegah pemilihan ganda oleh pengguna yang sama.
2. Pengujian Jaringan Blockchain
    - Ketika node validator dimatikan, upaya untuk menambahkan data ke blockchain gagal, yang menegaskan pentingnya node validator dalam proses penambahan data.
    - Node yang tidak terdaftar atau bukan validator tidak dapat mengakses jaringan atau menambahkan data, yang menunjukkan mekanisme keamanan yang efektif dalam jaringan blockchain ini.
    - Ketika node validator tidak aktif upaya untuk mendapatkan data dari blockchain masih dapat dilakukan melalui node member. Hal ini mebuktikan bahwa blockchain bersifat transparan karena masing-masing node memiliki salinan transaksi.
3. Pengujian Smart Contract
    - Pengujian smart contract menggunakan hardhat test menunjukkan bahwa smart contract berfungsi dengan baik dalam berbagai skenario pengujian, memastikan integritas dan validitas data pemilihan

6.2 Pembahasan

Berdasarkan hasil pengujian, beberapa kesimpulan dapat diambil mengenai keseluruhan sistem e-voting berbasis blockchain ini:

1. Transparasi dan Keamanan
    - Sistem e-voting yang dikembangkan menunjukkan keandalan dalam menjalankan fungsi-fungsinya dengan menyediakan transparansi. Seluruh node dalam jaringan memiliki salinan transaksi yang sama, sehingga setiap node dapat mengakses data transaksi tersebut. Hal ini memungkinkan semua pihak yang terlibat untuk memonitor transaksi secara independen,
    - Mekanisme keamanan yang diterapkan, seperti penggunaan node validator dan autentikasi node, berhasil mencegah akses tidak sah dan memastikan integritas data. Fitur-fitur ini memastikan bahwa hanya pihak yang berwenang yang dapat berpartisipasi dalam proses pemilihan, dan setiap transaksi yang tercatat di dalam jaringan adalah valid dan tidak dapat diubah. Dengan demikian, sistem ini melindungi data pemilihan dari potensi manipulasi dan serangan.
2. Efektivitas Blockchain

    Penggunaan teknologi blockchain dalam sistem e-voting ini memberikan tingkat keamanan dan transparansi yang tinggi. Smart contract yang digunakan memastikan bahwa setiap transaksi pemilihan diverifikasi dan disimpan dengan aman. Seluruh node dalam jaringan memiliki salinan dari semua transaksi, memastikan bahwa data pemilihan transparan dan dapat diakses oleh semua node.
3. Keterbatasan dan Tantangan
    - Meskipun sistem ini menunjukkan kinerja yang baik, ada beberapa tantangan yang perlu diatasi, seperti optimasi kecepatan eksekusi transaksi dan peningkatan skalabilitas sistem untuk menangani jumlah pemilih yang lebih besar di masa mendatang.
    - Penelitian lebih lanjut diperlukan untuk menguji sistem dalam skenario dunia nyata dengan jumlah pengguna yang lebih besar dan kondisi jaringan yang lebih bervariasi.

Dengan hasil dan pembahasan ini, penelitian ini memberikan kontribusi yang signifikan dalam mengembangkan sistem e-voting yang aman dan transparan menggunakan teknologi blockchain. Temuan ini juga membuka jalan untuk penelitian dan pengembangan lebih lanjut dalam penerapan blockchain untuk sistem e-voting dan aplikasi lainnya yang memerlukan tingkat keamanan dan transparansi yang tinggi

BAB VII. KESIMPULAN DAN SARAN

7.1 Kesimpulan

Hasil dari penelitian yang telah dilakukan oleh penulis pada implementasi aplikasi e-voting berbasis blockchain menggunakan Go Ethereum untuk mengatasi manipulasi data perolehan suara dan melakukan transparansi data pemilihan dengan tetap menjaga anonimitas pemilih, maka dapat diperoleh kesimpulan sebagai berikut:

- Sistem e-voting berbasis blockchain ini berhasil menjaga transparansi data pemilihan serta hasil pemungutan suara. Seluruh node dalam jaringan memiliki salinan dari semua transaksi, memungkinkan setiap node untuk memverifikasi dan mengakses data pemilihan secara real-time. Sistem ini mampu melaksanakan proses pemungutan dan penghitungan suara dengan efisien dan transparan, menampilkan hasil secara langsung. Smart contract berfungsi dengan baik dalam menyimpan data pemilihan, memastikan bahwa hasil pemilihan dapat diakses oleh semua pihak yang terlibat.
- Penggunaan teknologi blockchain dalam sistem e-voting ini juga berhasil menjaga keamanan data pemilihan dan akses. Node validator memastikan bahwa hanya data yang valid yang dapat ditambahkan ke dalam blockchain, mencegah manipulasi, dan menjaga integritas data. Pengujian fungsionalitas menunjukkan bahwa sistem dapat melakukan login, menampilkan kandidat, memproses pemilihan, dan mencegah pemilihan ganda dengan baik. Namun, penelitian ini juga mengidentifikasi tantangan seperti optimasi kecepatan transaksi dan peningkatan skalabilitas untuk jumlah pemilih yang lebih besar.

7.2 Saran

Dari hasil penelitian aplikasi mobile untuk e-voting mengunakan blockchain ini masih terdapat beberapa kekurangan. Saran yang dapat diberikan untuk pengembangan penelitian ini kedepannya, yaitu

1. Peningkatan Skala: Uji dan tingkatkan sistem untuk menangani pemilihan yang lebih besar tanpa mengurangi kinerja.
2. Penggunaan Teknologi yang Lebih Cepat: Pertimbangkan penggunaan platform berbeda untuk menyimpan data tambahan agar tidak terlalu banyak

fungsi dan struct dalam smart contract, sehingga blockchain dapat memproses lebih cepat.

3. Pengembangan Fitur Tambahan: Tambahkan fitur seperti diagram yang dapat menampilkan data-data hasil pemilihan yang lebih lengkap dan peningkatan antarmuka pengguna untuk pengalaman yang lebih baik.

Dengan menerapkan saran-saran ini, diharapkan sistem e-voting dapat menjadi lebih optimal dan memenuhi kebutuhan pemilu yang lebih kompleks.

DAFTAR PUSTAKA

Ante, L. (2021). Smart contracts on the blockchain – A bibliometric analysis and review. *Telematics and Informatics*, *57*, 101519. https://doi.org/https://doi.org/10.1016/j.tele.2020.101519

Arianto, E., Umam, C., & Handoko, L. B. (2021). Purwarupa Sistem Pemilihan Umum Elektronik dengan Pemanfaatan Protokol Ethereum pada Teknologi Blockchain. *TRANSFORMATIKA*, *19*(1), 88–97.

Asad, N. Al, Elahi, M. T., Hasan, A. Al, & Yousuf, M. A. (2020). Permission-based blockchain with proof of authority for secured healthcare data sharing. *2020 2nd International Conference on Advanced Information and Communication Technology, ICAICT 2020*, 35–40. https://doi.org/10.1109/ICAICT51780.2020.9333488

Balqis, A. N., Ramadhana, L., Wirawan, R., & Isnainiyah, I. N. (2018). PERANCANGAN APLIKASI E-VOTING GRAB YOUR VOTE (GRAVOTE) BERBASIS ANDROID PADA LINGKUP PERGURUAN TINGGI. *Seminar Nasional Informatika, Sistem Informasi Dan Keamanan Siber (SEINASI-KESI)*, 132–138.

Hjalmarsson, F. P., Hreioarsson, G. K., Hamdaqa, M., & Hjalmtysson, G. (2018). Blockchain-Based E-Voting System. *IEEE International Conference on Cloud Computing, CLOUD*, *2018-July*, 983–986. https://doi.org/10.1109/CLOUD.2018.00151

Hu, S. D. K., Palit, H. N., & Handojo, A. (2019). Implementasi Blockchain: Studi Kasus e-Voting. *Jurnal Infra*, *7*(1), 183–189.

Jri Fadli, F., Jatmiko, S., & Lamsani, M. (2020). Electronic Voting Using Decentralized System Based on Ethereum Blockchain. *Jurnal Ilmiah KOMPUTASI*, *19*(1), 1412–9434. https://doi.org/10.32409/jikstik.19.1.2718

Karmanis. (2021). ELECTRONIC-VOTING (E-VOTING) DAN PEMILIHAN UMUM. *Jurnal MIMBAR ADMINISTRASI*, *18*(2), 11–24.

Khoury, D., Kfoury, E. F., Kassem, A., & Harb, H. (2018). Decentralized Voting Platform Based on Ethereum Blockchain. *2018 IEEE International*

Multidisciplinary Conference on Engineering Technology (IMCET), 1–6. https://doi.org/10.1109/IMCET.2018.8603050

Liu, Y., & Wang, Q. (2017). An E-voting Protocol Based on Blockchain. *IACR Cryptol. Eprint Arch*, 1043–1045.

Malkawi, M., Yassein, M. B., & Bataineh, A. (2021). Blockchain based voting system for Jordan parliament elections. *International Journal of Electrical and Computer Engineering*, *11*(5), 4325–4335. https://doi.org/10.11591/ijece.v11i5.pp4325-4335

Meth, M. (2019). *Blockchain in Libraries* (S. Imburgia, Ed.; Vol. 55). ALA Production Services. https://doi.org/10.5860/ltr.55n8

Saputra, B. W., & Nasution, B. J. (2021). TINDAK LANJUT TERHADAP PENERAPAN ELEKTRONIK VOTING DALAM PELAKSANAAN PEMILIHAN KEPALA DAERAH BERDASARKAN PERATURAN PERUNDANG-UNDANGAN. *Limbago: Journal of Constitutional Law*, *1*(2), 212–232.

Setia, T. E. H., & Susanto, A. (2019). Smart Contract Blockchain pada E-Voting. *JURNAL INFORMATIKA UPGRIS*, *5*(2), 188–191.

Shen, M., Tan, Z., Niyato, D., Liu, Y., Kang, J., Xiong, Z., Zhu, L., Wang, W., Xuemin, & Shen. (2023). Artificial Intelligence for Web 3.0: A Comprehensive Survey. *Journal of the ACM*, *37*(4), 111. http://arxiv.org/abs/2309.09972