Otomatisasi Verifikasi Kode Flutter pada Server untuk Pembelajaran Aplikasi Mobile

SKRIPSI

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

Oleh:

KEVIN NATANAEL WIJAYA NIM. 2041720091

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

HALAMAN PENGESAHAN

OTOMATISASI VERIFIKASI KODE FLUTTER PADA SERVER UNTUK PEMBELAJARAN APLIKASI MOBILE

Disusun oleh:

Kevin Natanael Wijaya NIM. 2041720091

Skripsi ini telah disidangkan pada hari **Rabu, 17 Juli 2024.**

Disetujui oleh:

Pembimbing Utama	: Yan Watequlis Syaifudin, ST., M.MT., Ph.D.	
	NIP. 198101052005011005	
Pembimbing Pendamping	: Pramana Yoga Saputra, S.Kom., MMT.	
	NIP. 198805042015041001	
Penguji Utama	: Dr. Rakhmat Arianto, S.ST., M.Kom.	
	NIP. 198701082019031004	
Penguji Pendamping	: Rudy Ariyanto, ST., M.Cs.	
	NIP. 197111101999031002	

Mengetahui

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

PERNYATAAN

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

Malang, 17 Juli 2024

Kevin Natanael Wijaya.

ABSTRAK

Natanael W., Kevin. “Otomatisasi Verifikasi Kode Flutter pada Server untuk Pembelajaran Aplikasi Mobile”.
Pembimbing: (1) Yan Watequlis Syaifudin, ST., M.MT., Ph.D. (2) Pramana Yoga Saputra, S.Kom., MMT.

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

Skripsi ini berjudul "Otomatisasi Verifikasi Kode Flutter pada Server untuk Pembelajaran Aplikasi Mobile" dan ditulis oleh Kevin Natanael Wijaya. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem otomatisasi verifikasi kode Flutter pada server untuk meningkatkan efisiensi dalam pembelajaran aplikasi mobile berbasis Flutter.

Sistem ini dirancang untuk membantu dosen dalam melakukan verifikasi kode secara otomatis dan membantu mahasiswa dalam mempelajari pengembangan aplikasi mobile dengan Flutter. Sistem ini menggunakan teknologi Real-Time untuk memantau dan mengevaluasi hasil pekerjaan mahasiswa. Analisis kebutuhan fungsional dan non-fungsional dilakukan untuk menentukan fitur-fitur yang diperlukan dalam sistem. Hasil penelitian menunjukkan bahwa sistem otomatisasi verifikasi kode dapat meningkatkan efisiensi dalam pembelajaran aplikasi mobile, serta meningkatkan pengalaman pengguna.

Kata kunci: Otomatisasi verifikasi kode, Flutter, Server, Pembelajaran Aplikasi Mobile.

ABSTRACT

Natanael W., Kevin. *"Automated Flutter Code Verification on Server for Mobile Application Learning".*
Supervisors: Yan Watequlis Syaifudin, ST., M.MT., Ph.D., Co-Supervisor: Pramana Yoga Saputra, S.Kom., MMT.

Thesis, Informatics Engineering Study Program, Information Technology Department, Politeknik Negeri Malang, 2024.

This thesis is titled "Automated Flutter Code Verification on Server for Mobile Application Learning" and is written by Kevin Natanael Wijaya. The study aims to design and implement an automated system for verifying Flutter code on a server to enhance efficiency in learning Flutter-based mobile applications. The system is designed to assist lecturers in automatically verifying code and help students learn mobile application development using Flutter.

The system uses Real-Time technology to monitor and evaluate students' work results. Functional and non-functional requirements analysis was conducted to determine the necessary features of the system. The research results indicate that the automated code verification system can improve efficiency in mobile application learning and enhance user experience.

Keywords: *Automated code verification, Flutter, Server, Mobile Application Learning.*

KATA PENGANTAR

Semua puji dan syukur penulis panjatkan kehadirat Allah SWT/Tuhan YME atas segala rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "OTOMATISASI VERIFIKASI KODE FLUTTER PADA SERVER UNTUK PEMBELAJARAN APLIKASI MOBILE". Skripsi ini penulis susun sebagai persyaratan untuk menyelesaikan studi program Diploma IV Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang.

Penulis menyadari bahwa 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 Andire Asmara, ST, MT., selaku Ketua Jurusan Teknologi Informasi Politeknik Negeri Malang.
2. Ibu Dr. Ely Setyo Astuti, S.T., M.T., selaku Ketua Program Studi DIV Teknik Informatika Politeknik Negeri Malang.
3. Bapak Yan Watequlis Syaifudin, ST., M.MT., Ph.D., selaku Dosen Pembimbinng Utama dalam proses penelitian Tugas Akhir kali ini.
4. Bapak Pramana Yoga Saputra, S.Kom., MMT., selaku Dosen Pembimbinng Pendamping dalam proses penelitian Tugas Akhir kali ini.
5. Kedua orang tua saya, yang telah memotivasi, mendoakan, dan terus memberikan semangat kepada saya untuk menjalani kegiatan ini dengan bersungguh-sungguh untuk membangun diri saya.
6. Teman-teman terdekat yang selalu ada untuk membantu, mendengarkan dan memberikan dorongan untuk saya sebagai penulis untuk menyelesaikan skripsi saya.
7. Dan semua pihak yang telah membantu dan mendukung lancarnya pembuatan Laporan Akhir dari Penelitian Tugas Akhir ini 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, 17 Juli 2024

Kevin Natanael Wijaya

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I. PENDAHULUAN

1.1. Latar Belakang

Beberapa tahun terakhir, perkembangan teknologi mengalami pertumbuhan yang sangat pesat, terutama dalam pemrograman mobile, dan dan Flutter menjadi salah satu framework paling populer untuk pengembangan aplikasi seluler. Dikembangkan oleh Google, Flutter menggunakan bahasa pemrograman Dart dan memungkinkan pengembang dengan cepat membuat aplikasi yang indah dan responsif. Oleh karena itu, banyak perusahaan dan pengembang independen yang tertarik menggunakan Flutter sebagai pilihan pertama mereka untuk pengembangan aplikasi seluler Salah satu alasan utama mengapa Flutter semakin populer adalah kemampuannya untuk mempercepat proses pengembangan. Dengan satu basis kode, pengembang dapat menghindari kebutuhan untuk menulis kode terpisah untuk setiap sistem operasi, seperti Android dan iOS, yang sebelumnya memerlukan tim pengembang yang berbeda. Hal ini tidak hanya menghemat waktu, tetapi juga biaya pengembangan (Stošović, Stefanović, Bogdanović, & Vukotić, 2022). Dukungan yang kuat dari komunitas dan Google juga berkontribusi pada adopsi Flutter. Banyak sumber daya, tutorial, dan paket tambahan tersedia, memudahkan pengembang untuk memulai dan menyelesaikan proyek mereka (Cheon & Chavez, 2021). Dengan semua keunggulan ini, Flutter tidak hanya memenuhi tuntutan pasar yang terus meningkat, tetapi juga memberikan solusi efisien untuk tantangan yang dihadapi oleh pengembang aplikasi saat ini.

Oleh karena itu, pelatihan dan pembelajaran Flutter menjadi kebutuhan mendesak untuk memenuhi permintaan pasar yang terus meningkat. Namun, meskipun kebutuhan akan pengetahuan Flutter semakin meningkat, tidak ada solusi otomatis untuk memeriksa kode Flutter. Dan juga adanya wabah penyakit covid-19 telah membawa perubahan yang mendesak pada berbagai sektor. Perkembangan virus dengan cepat menyebar luas di seluruh dunia. Setiap hari data di dunia mengabarkan bertambahnya cakupan dan dampak covid-19. Indonesia pun masuk dalam keadaan darurat nasional. Angka kematian akibat Corona terus meningkat sejak diumumkan pertama kali ada masyarakat yang positif terkena virus covid-19 pada awal Maret 2020. Hal tersebut mempengaruhi

perubahan-perubahan dan pembaharuan kebijakan untuk diterapkan. Kebijakan baru juga terjadi pada dunia pendidikan merubah pembelajaran yang harus datang ke kelas atau suatu gedung, dalam hal ini kampus, menjadi cukup di rumah saja. Anjuran pemerintah untuk stay at home dan physical and social distancing harus diikuti dengan perubahan modus belajar tatap muka menjadi online (Khasanah, Pramudibyanto, & Widuroyekti, 2020).

Aplikasi mobile programming semakin populer dan banyak digunakan oleh masyarakat. Namun, verifikasi kode pada server untuk pembelajaran aplikasi mobile masih dilakukan secara manual. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan Otomatisasi Verifikasi Kode Aplikasi Mobile Berbasis Flutter pada Server untuk Pembelajaran Aplikasi Mobile. Fokus utama dari skripsi ini adalah pengecekan didalam server dari hasil upload mahasiswa yang sudah mengerjakan test yang diberikan didalam aplikasi mobile agar mempermudah dosen dalam melakukan penilaian kinerja mahasiswa (Aziz, Andreswari, & Gumilang, 2020).

Penelitian ini bertujuan untuk mengatasi tantangan dalam proses verifikasi kode Flutter dengan mengembangkan sistem otomatisasi verifikasi kode pada server. Dengan cara ini, kami bertujuan untuk mempercepat pengecekan kode yang sudah tersimpan didalam server, khususnya dalam konteks pembelajaran. Sistem yang diusulkan diharapkan dapat memberikan hasil peninjauan kode yang cepat, akurat, dan andal, sehingga memungkinkan pengembang untuk fokus mengembangkan fungsionalitas aplikasinya tanpa membebani proses peninjauan kode secara manual.

1.2 Rumusan Masalah

1. Apakah tantangan utama yang dihadapi pengembang dalam melakukan verifikasi kode pada proyek pengembangan pembelajaran aplikasi mobile berbasis Flutter?
2. Bagaimana proses verifikasi kode dapat diotomatisasi secara efektif pada server untuk meningkatkan efisiensi pengembangan aplikasi mobile berbasis Flutter?

3. Apakah sistem otomatisasi verifikasi kode pada server mampu menghasilkan hasil verifikasi yang akurat dan dapat diandalkan untuk memastikan kualitas dan ketepatan kode pada aplikasi mobile berbasis Flutter?

1.3 Batasan Masalah

1. Penelitian ini akan memfokuskan pada aplikasi mobile yang dikembangkan untuk keperluan pembelajaran mahasiswa. Oleh karena itu, batasan lingkup melibatkan aplikasi pembelajaran dan fitur-fitur yang relevan dengan konteks pendidikan Mahasiswa.
2. Penelitian ini akan membatasi diri pada otomatisasi pengecekan bahasa pemrograman Flutter, sehingga sistem otomatisasi verifikasi kode akan difokuskan pada proyek-proyek pengembangan aplikasi mobile yang menggunakan Flutter sebagai framework utama.

1.4 Tujuan

Tujuan dari dilakukannya skripsi dengan judul "**Otomatisasi Verifikasi Kode Flutter pada Server untuk Pembelajaran Aplikasi Mobile**", adalah sebagai berikut:

1. Menganalisis dan mengidentifikasi tantangan utama yang dihadapi oleh pengembang dalam melakukan verifikasi kode pada proyek pengembangan aplikasi mobile berbasis Flutter.
2. Merancang dan mengimplementasikan sistem otomatisasi verifikasi kode yang efektif di server untuk meningkatkan efisiensi proses pengembangan aplikasi mobile menggunakan Flutter.
3. Mengevaluasi kinerja sistem otomatisasi verifikasi kode pada server dalam menghasilkan hasil verifikasi yang akurat dan dapat diandalkan untuk memastikan kualitas dan ketepatan kode pada aplikasi mobile berbasis Flutter.
4. Memberikan rekomendasi dan solusi yang dapat membantu pengembang mengatasi tantangan dalam verifikasi kode dan meningkatkan praktik pengembangan aplikasi mobile berbasis Flutter secara keseluruhan.

1.5 Manfaat

1. Peningkatan Efisiensi Pengembangan: Dengan adanya sistem otomatisasi verifikasi kode, pengembang dapat menghemat waktu dan upaya yang sebelumnya dihabiskan untuk proses verifikasi manual. Hal ini akan membantu meningkatkan efisiensi dalam pengembangan aplikasi mobile berbasis Flutter.
2. Peningkatan Keamanan Aplikasi Mobile: Verifikasi otomatis kode dapat membantu mengidentifikasi dan mengatasi potensi kerentanan keamanan pada aplikasi mobile. Ini akan memberikan kepercayaan tambahan kepada pengguna aplikasi, terutama dalam konteks aplikasi pembelajaran di mana privasi dan keamanan informasi pengguna menjadi sangat penting.
3. Meningkatkan Pengalaman Pengguna: Dengan memastikan kualitas dan keamanan kode, pengalaman pengguna aplikasi mobile, khususnya dalam konteks pembelajaran mahasiswa, dapat ditingkatkan. Pengguna dapat mengakses informasi yang dibutuhkan dengan lebih aman dan nyaman tanpa khawatir tentang potensi masalah teknis atau keamanan.

BAB II. LANDASAN TEORI

2.1 Studi Literatur

Studi literatur menjadi salah satu acuan penulis dalam melakukan penelitian sehingga penulis dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Berikut yaitu studi literatur berupa jurnal terkait dengan penelitian yang dilakukan oleh penulis ada beberapa penetitian yang dilakukan (Syaifudin, et al., 2022) ini membahas mengenai implementasi aplikasi mobile menggunakan IDE seperti Android Studio atau VS Code. Didalam sistem ini sudah terdapat guide book yang berisikan konsep mengenai Flutter dan panduan terstruktur mengenai penulisan kode sesuai dengan konsep. Untuk sistem prektik didalamnya, mahasiswa harus menyelesaikan tugas sesuai dengan yang diberikan dengan memasukkan source code. Source code tersebut yang nantinya akan dilakukan verifikasi secara otomatis didalam aplikasi.

Selanjutnya Penelitian yang dilakukan oleh (Syaifudin, Funabiki, & Kuribayashi, Implementation and Performance Evaluation of Unit Testing for Student's Answer Validation in Android Programming Learning Assistant System, 2021) Pada sistem Android Programming Learning Assistance System (APLAS) menggunakan metode test-driven development (TTD) untuk memverifikasi jawaban tugas siswa, termasuk kode sumber dan paket proyek Android, yaitu unit test. APLAS memiliki sisi klien dan sisi server. Pada sisi klien, akses server melalui browser dan jalankan tugas di PC siswa melalui Android Studio.Saat ini, terdapat PC dan server cloud di sisi server.Sistem Menyediakan aplikasi web, program verifikasi, dan sistem basis data.

Selanjutnya Penelitian yang dilakukan oleh (Syaifudin, et al., Web application implementation of Android programming learning assistance system and its evaluations, 2020), didalam penelitian yang dilakukan Mahasiswa dapat mengkases antarmuka web yang berguna untuk mendapatkan materi pembelajaran, dan mengerjakan tugas yang telah diberikan, dimana validator dapat dilakukan secara otomatis, dengan haril evaluasi yang dapat dipastikan efektivitas dan keakuratan proposal ini.

2.2 Dart

Dart merupakan bahasa pemrograman yang dikembangkan oleh Google dan dirilis pada tahun 2011. Dart dirancang untuk memudahkan pengembangan aplikasi web dan mobile dengan fitur-fitur modern seperti asynchronous programming, garbage collection, dan type inference.

.Beberapa fitur didalam Bahasa pemrograman Dart antara lain Asynchronous programming karena Dart mendukung asynchronous programming dengan menggunakan Future dan async/await. Hal ini memungkinkan aplikasi untuk melakukan operasi I/O tanpa harus menunggu operasi tersebut selesai terlebih dahulu, Garbage collection didalam Dart yang secara otomatis menghapus objek yang tidak lagi digunakan oleh aplikasi. Hal ini memudahkan pengembang dalam mengelola memori aplikasi, Type inference didalam Dart mendukung memungkinkan pengembang untuk tidak harus menuliskan tipe data secara eksplisit dalam kode. Dart akan secara otomatis menentukan tipe data dari suatu variabel berdasarkan nilai yang diberikan (Hanif & Sinambela, 2020).

2.3 Flutter

Flutter, sebuah framework open-source yang dikelola oleh Google, dirancang untuk mempermudah pengembangan antarmuka pengguna (UI) pada aplikasi mobile, web, dan desktop dengan satu kode basis. Struktur dasar Flutter mencakup konsep dasar sebagai berikut: Widget adalah Elemen UI dalam Flutter disusun sebagai widget, dengan dua jenis utama, yaitu StatelessWidget (widget yang tidak berubah) dan StatefulWidget (widget yang dapat berubah). Material Design dan Cupertino didala m Flutter menyediakan widget yang sesuai dengan panduan desain Material Design dari Google dan gaya desain iOS dari Apple, memungkinkan pengembang menciptakan aplikasi yang konsisten dengan platform yang dituju. Hot Reload, sebagai fitur yang memungkinkan pengembang melihat perubahan langsung pada kode tanpa memerlukan restart aplikasi, membantu mempercepat proses pengembangan.

Kinerja pada Flutter menggunakan mesin rendering Skia sendiri untuk menggambar antarmuka pengguna, memberikan kinerja yang cepat dan responsif. Pengembangan lintas platform menjadi salah satu keunggulan Flutter, memungkinkan pengembang membuat aplikasi yang dapat berjalan di iOS,

Android, web, dan desktop dengan menggunakan kode yang sama. Dengan berbagai fitur dan kelebihan ini, Flutter menjadi pilihan populer di kalangan pengembang yang ingin menghasilkan aplikasi lintas platform dengan efisiensi tinggi (Flutter, 2023).

2.4 Flutter Testing (Unit Test & Widget Test)

Unit Test merupakan bentuk pengujian yang fokus pada verifikasi perilaku fungsi individu, metode, atau kelas. Fungsinya adalah untuk memastikan bahwa unit logika dapat berperilaku dengan benar dalam berbagai kondisi. Dalam konteks kerangka kerja Flutter, paket pengujian mencakup inti dari kerangka kerja untuk melakukan pengujian unit dan paket uji tambahan untuk utilitas pengujian widget (Syaifudin, et al., 2022).

Pengujian unit dilakukan untuk menguji fungsi secara umum tanpa terpengaruh oleh tindakan eksternal atau proses lain selama pengujian .Sebaliknya, Widget Test merupakan bentuk pengujian yang fokus pada komponen, memverifikasi interaksi yang baik antara UI dan widget, serta memastikan tata letak dan tindakan dapat dilakukan dengan benar. Meskipun lebih komprehensif dibandingkan dengan tes unit, lingkungan tes widget disederhanakan untuk mencocokkan implementasi yang lebih simpel dari sistem UI penuh (Hadjrianto, 2022).

2.5 Java

Java adalah sebuah bahasa pemrograman yang dikembangkan oleh Sun Microsystems (sekarang dikenal sebagai Oracle Corporation) pada tahun 1995. Java dirancang untuk memudahkan pengembangan aplikasi yang dapat berjalan di berbagai platform tanpa perlu diubah kode. Java menggunakan konsep "write once, run anywhere" yang memungkinkan aplikasi yang dibuat dengan Java dapat berjalan di berbagai sistem operasi dan perangkat keras tanpa perlu modifikasi (Kozak, 2023).

2.6 Spring Boot

Spring Boot adalah sebuah framework yang dikembangkan oleh SpringSource, sebuah divisi dari VMware. Spring Boot dirancang untuk memudahkan pengembangan aplikasi web berbasis Java dengan menggunakan

konfigurasi default dan minimalisasi kode. Spring Boot menggunakan konsep "opinionated" yang memungkinkan pengembang untuk fokus pada logika bisnis tanpa perlu memperhatikan detail teknis (Kozak, 2023).

2.7 Agile Method

Metode Agile adalah pendekatan dalam pengembangan perangkat lunak yang menekankan fleksibilitas, kolaborasi, dan iterasi berkelanjutan. Agile bertujuan untuk meningkatkan daya tanggap terhadap perubahan kebutuhan pelanggan dan meminimalkan risiko proyek. Prinsip utama dari Agile termasuk kolaborasi antara tim pengembang dan pelanggan, pengiriman perangkat lunak yang berfungsi secara iteratif, dan kemampuan untuk merespons perubahan dengan cepat. Beberapa kerangka kerja Agile yang populer termasuk Scrum, Kanban, dan Extreme Programming (XP). Agile memungkinkan untuk bekerja dalam siklus pendek yang disebut sprint, yang biasanya berlangsung antara satu hingga empat minggu, dengan tujuan menghasilkan produk yang dapat digunakan pada akhir setiap sprint (Stray, Hoda, Maria, & Kruchten, 2020).

2.8 White Box Testing

White Box Testing, juga dikenal sebagai clear box testing, glass box testing, atau structural testing, adalah metode pengujian perangkat lunak yang memeriksa struktur internal atau kode dari aplikasi yang diuji. Pengujian ini dilakukan dengan pengetahuan penuh tentang kode sumber, algoritma, dan struktur data yang digunakan dalam aplikasi. Tujuan utama dari White Box Testing adalah untuk memastikan bahwa semua jalur logika dalam kode telah diuji dan berfungsi dengan benar. Metode ini melibatkan pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian regresi. Pengujian ini sangat efektif dalam menemukan kesalahan yang mungkin tidak terdeteksi oleh metode pengujian lainnya karena fokusnya pada kode internal dan logika program (Kreinovich, Ceberio, & Kosheleva, 2020).

BAB III. METODOLOGI PENELITIAN

Metodologi penelitian menjelaskan metode yang digunakan didalam pengerjaan skripsi ini diantaranya:

1. Metode pengumpulan data
2. Metode pengolahan data
3. Desain system
4. Metode Pengujian
5. Metode Penelitian

3.1. Waktu dan Tempat Penelitian

Waktu yang digunakan oleh peneliti untuk melaksanakan penelitian ini dalam waktu kurang lebih 6 (enam) bulan, dan Tempat pelaksanaan penelitian ini dilakukan di Politeknik Negeri Malang di Jalan Soekarno Hatta No.9 Malang, Kota Malang, Jawa Timur, Indonesia.

3.2. Teknik Pengumpulan Data

Metode pengumpulan data yang akan digunakan dalam pengembangan aplikasi pembelajaran Flutter melibatkan proses pengumpulan, kajian, dan pemahaman informasi dari beragam sumber referensi. Sumber referensi ini akan menjadi dasar untuk membangun aplikasi otomatisasi verifikasi kode dalam pembelajaran mobile.

Materi pembelajaran, topik, dan proyek aplikasi Flutter ini akan disusun berdasarkan berbagai jurnal, artikel, dan penelitian yang disusun oleh Dosen Pengajar di Politeknik Negeri Malang yang memiliki keahlian dalam mengajar pembelajaran aplikasi mobile berbasis Flutter. Data ini akan diproses terlebih dahulu oleh admin sebelum diterapkan oleh mahasiswa yang mengakses platform pembelajaran aplikasi Flutter ini.

3.3 Teknik Pengolahan Data

Admin akan melakukan konfigurasi data didalam aplikasi mengenai topik yang akan diberikan didalam aplikasi. Konfigurasi tersebut berguna untuk mencegah terjadinya kesalahan data yang nanti akan diakses oleh mahasiswa . Lalu mahasiswa dapat mengunduh materi pembelajaran dan juga modul yang diberikan, selain itu mahasiswa juga dapat mengupload tugas ataupun project yang sudah

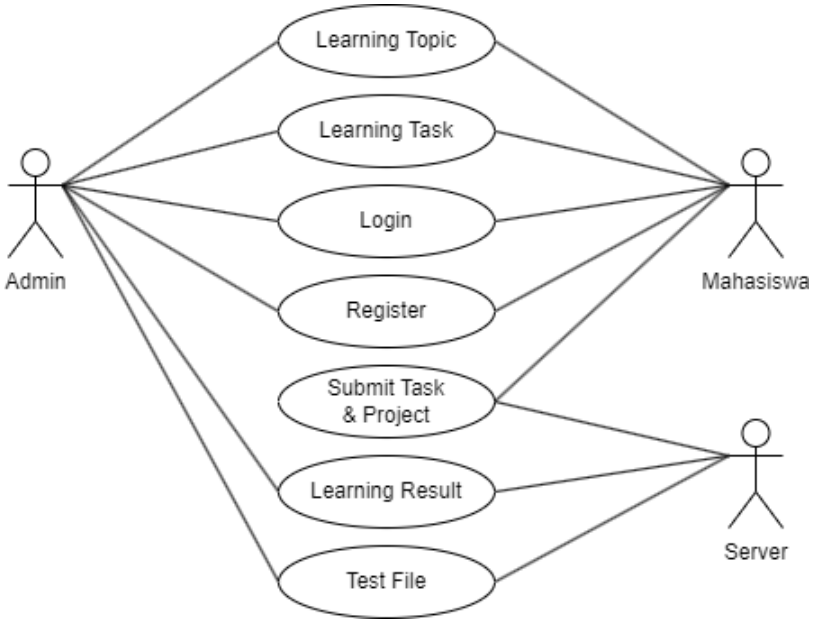
ditentukan oleh admin sesuai dengan topik pembelajaran yang diberikan. Selanjutnya mahasiswa dapat mengsubmit hasil dari pengerjaannya. Hasil pengerjaan dari tugas ataupun project nantinya akan dicek kembali dan divalidasi oleh admin.

3.4 Desain Sistem

Tabel dibawah ini merupakan deskripsi mengenai konsep dari aplikasi

Tabel 3. 1 Tabel Konsep Aplikasi

Judul	Otomatisasi Verifikasi Kode Flutter pada Server untuk Pembelajaran Aplikasi Mobile
Jenis Aplikasi	Aplikasi ini berguna untuk mempermudah dosen dalam melakukan pengecekan kode secara otomatis didalam sistem.
Dosen	Dosen menjadi lebih mudah dalam melakukan pengecekan kode yang sudah dikerjakan oleh mahasiswa
Mahasiswa	Mahasiswa bisa mendapatkan hasil dari pekerjaannya lebih cepat dan akurat
Materi	Materi yang akan diberikan didalam aplikasi ini adalah pembelajaran mobile programming berbasis Flutter mulai dari instalasi sampai pembuatan aplikasi
Aplikasi	Aplikasi ini dibuat menggunakan Flutter framework yang sudah mendukung berbagai platform seperti IOS dan android
Teknologi	Aplikasi ini menggunakan database Real-Time yang yang akan mengecek kode yang sudah diupload dengan cepat dan efektif

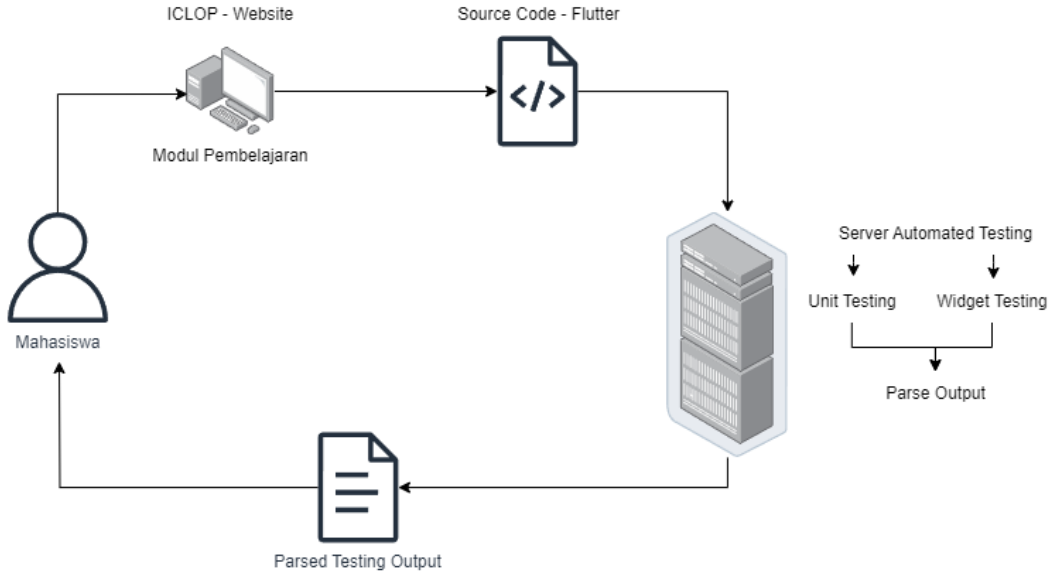


Gambar 3. 1 Use Case Diagram Aplikasi

Pada table dibawah ini akan dijelaskan mengenai deskripsi dari setiap actor yang terlibat didalam aplikasi

Aktor	Deskripsi
Admin	Admin sebagai orang yang bertanggung jawab untuk pengelolaan internal server dari aplikasi seperti mengatur data yang akan ditampilkan didalam aplikasi dan juga menilai hasil dari apa saja yang dikerjakan oleh mahasiswa
Mahasiswa	Sebagai pengguna aplikasi mahasiswa bertanggung jawab dalam mempelajari materi dan juga melakukan berbagai tugas maupun projek yang sudah tersedia didalam aplikasi
Server	Sebagai server berguna dalam mengolah data yang sudah diinputkan didalam database dan juga memverifikasi kode yang sudah dimasukkan

Deskripsi sistem secara umum dapat dilihat melalui diagram dibawah ini:



Gambar 3. 2 Diagram Siklus Aplikasi

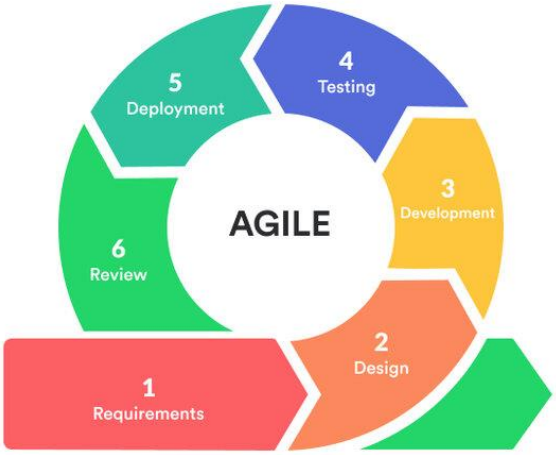
Dari diagram di atas, kita dapat memahami gambaran umum aplikasi ini. Pertama, mahasiswa akan mengerjakan source code aplikasi yang akan diuji sesuai dengan soal yang telah diberikan. Setelah itu, source code yang telah dikerjakan akan diunggah ke dalam situs web yang terhubung ke server. Di dalam server, dilakukan pengujian terhadap source code yang telah diunggah dengan dua jenis pengujian, yakni unit testing, yang menguji fungsi atau metode aplikasi agar dapat berfungsi sesuai harapan, dan widget testing, yang fokus pada pengujian tampilan dalam aplikasi terutama terhadap widget yang menyusun antarmuka pengguna (UI) pada aplikasi Flutter. Hasil pengujian yang telah dilakukan di dalam server akan menghasilkan output yang kemudian dipetakan untuk memudahkan pemahaman. Selanjutnya, dosen dan mahasiswa dapat menerima hasil output dari server dan melakukan pengecekan dengan lebih cepat dan efisien.

Penelitian ini berguna untuk menjalankan otomatisasi yang berada didalam server secara terintegrasi dengan source code yang dilakukan verifikasi didalamnya dengan sumber dari modul dan juga task mahasiswa. Untuk memudahkan pengecekan dari hasil output testing dari aplikasi akan dilakukan mapping dari setiap hasil output yang sudah ada didalam server agar menjadi lebih efisien pengecekan dari dosen terhadap testing aplikasi dari mahasiswa. Dengan sistem tradisional yang selama ini dilakukan secara manual dalam melakukan pengecekan

aplikasi mobile yang memakan banyak waktu untuk melakukan pengecekan maka dengan otomatisasi akan menjadi lebih mudah, cepat, dan akurat.

3.5 Metode Penelitian

Metodologi Agile adalah pendekatan manajemen proyek dan pengembangan perangkat lunak yang bertujuan untuk meningkatkan daya tanggap terhadap perubahan kebutuhan pelanggan dan meminimalkan risiko proyek. Metode ini sangat cocok untuk proyek yang kompleks dan dinamis dimana perubahan sering terjadi. Agile menekankan kolaborasi, interaksi pelanggan, dan penyampaian produk yang berfungsi.



Gambar 3. 3 Gambar Metode Agile

Berikut adalah beberapa prinsip utama dari metodologi agile:

1. Individu dan interaksi melalui proses dan alat:
    - Partisipasi aktif antara pengembang, pemilik produk, dan pemangku kepentingan lainnya dianggap penting.
2. Perangkat lunak fungsional lebih dari sekedar dokumentasi lengkap:
    - Berfokus pada pengembangan produk yang berfungsi dan memberikan nilai langsung kepada pelanggan.
    - Dokumentasi masih penting, namun lebih ditekankan pada hasil fungsional.
3. Berkolaborasi dengan Pelanggan Di Luar Negosiasi Kontrak:
    - Bekerja secara langsung dengan pelanggan dan calon pelanggan untuk memahami dan mengatasi perubahan kebutuhan mereka.

4. Menanggapi perubahan lebih dari sekadar mengikuti sebuah rencana:
    - Agile menyadari bahwa perubahan kebutuhan pelanggan adalah hal yang wajar dan dapat diterima.
    - Tim harus mampu beradaptasi dengan perubahan tersebut tanpa mengorbankan tujuan akhir proyek.
5. Pendekatan berulang dan bertahap:
    - Proyek dibagi menjadi iterasi atau "sprint" yang biasanya berlangsung dari 1 hingga 4 minggu.
    - Setiap iterasi menghasilkan produk yang dapat digunakan dan dapat dikirimkan ke pelanggan.

Kerangka kerja agile yang umum digunakan mencakup Scrum, Kanban, dan Extreme Programming (XP). Meskipun masing-masing memiliki pendekatan dan aturan khusus, keduanya memiliki prinsip dasar yang sama untuk pengembangan perangkat lunak yang lebih responsif dan adaptif.

3.6 Metode Pengujian

Metode pengujian untuk sistem otomatisasi verifikasi kode Flutter akan menggunakan White Box Testing. Pengujian ini fokus pada pengujian struktur internal kode dan logika program. Penguji perlu memahami kode dan logika program untuk melakukan pengujian ini.

.

BAB IV ANALISIS DAN PERANCANGAN SISTEM

4.1 Pendahuluan

Bab ini akan menguraikan tentang analisis dan perancangan sistem pengujian otomatis Flutter untuk platform ICLOP (Intelligence Computer-Assisted Learning for Computer Programming). Pembahasan meliputi identifikasi kebutuhan sistem, desain arsitektur, serta implementasi dari setiap komponen sistem untuk memastikan integrasi dan operasional fitur yang diusulkan berjalan dengan baik.

4.2 Analisis Kebutuhan Sistem

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional dari sistem ini meliputi beberapa aspek penting. Pertama, mahasiswa harus dapat mengakses antarmuka untuk mengajukan kode Flutter mereka melalui formulir pengajuan kode yang memungkinkan pengunggahan kode dalam berbagai format yang sesuai. Kedua, sistem harus mampu menjalankan proses pengujian otomatis terhadap kode yang diajukan, yang meliputi validasi sintaks, pengujian unit, dan pengujian integrasi. Selanjutnya, hasil pengujian harus disimpan dalam basis data untuk referensi lebih lanjut, mencakup status pengujian, waktu eksekusi, dan log hasil pengujian. Terakhir, mahasiswa harus dapat melihat hasil pengujian melalui antarmuka pengguna dengan informasi yang ditampilkan secara jelas, termasuk kesalahan yang ditemukan.

4.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional dari sistem ini juga sangat penting untuk memastikan performa yang optimal. Pertama, sistem harus dapat beroperasi secara konsisten tanpa mengalami downtime yang signifikan, dan setiap pengujian harus menghasilkan hasil yang dapat diandalkan. Kedua, proses pengujian harus dilakukan dengan cepat untuk memberikan umpan balik yang segera, dan sistem harus mampu menangani beberapa pengujian secara bersamaan tanpa penurunan kinerja. Selain itu, sistem harus dirancang agar scalable, mampu menangani peningkatan jumlah pengguna dan volume pengajuan kode tanpa masalah kinerja.

4.3 Perancangan Sistem

4.3.1 Desain Arsitektur Sistem

Desain arsitektur sistem meliputi komponen-komponen utama dari sistem dan bagaimana mereka berinteraksi satu sama lain. Arsitektur sistem ini terdiri dari tiga lapisan utama: presentasi, logika bisnis, dan penyimpanan data. Lapisan presentasi bertanggung jawab untuk interaksi pengguna dengan sistem melalui antarmuka yang intuitif. Lapisan logika bisnis mengelola aturan dan logika dari proses verifikasi kode serta pengelolaan materi. Sementara itu, lapisan penyimpanan data bertanggung jawab untuk menyimpan dan mengelola data yang digunakan oleh sistem, termasuk hasil verifikasi dan informasi pengguna.

4.3.2 Desain Model Data

Model data untuk sistem otomatisasi verifikasi kode Flutter diimplementasikan dengan berbagai entitas dan hubungan yang diatur dalam beberapa tabel basis data. Berdasarkan kode yang diberikan, beberapa tabel utama dan hubungan yang terlibat dalam model data adalah sebagai berikut:

- Tabel Topik : Tabel ini menyimpan informasi mengenai berbagai topik yang tersedia dalam materi pembelajaran Flutter. Setiap entri dalam tabel ini mencakup detail seperti ID topik dan deskripsi.
- Tabel Detail Topik Flutter: Tabel ini menyimpan detail lebih lanjut tentang setiap topik Flutter, termasuk deskripsi atau nama file yang terkait dengan materi topik tersebut.
- Tabel Hasil Uji Flutter : Tabel ini menyimpan hasil uji yang dilakukan oleh mahasiswa, termasuk ID pengguna, ID Flutter, jumlah tes yang berhasil, jumlah tes yang gagal, dan skor yang diperoleh.
- Tabel Pengguna : Tabel ini menyimpan informasi tentang pengguna sistem, seperti mahasiswa, dosen, dan admin. Informasi ini termasuk ID pengguna, nama, dan peran pengguna.
- Tabel Tugas Flutter : Tabel ini menyimpan informasi mengenai tugas-tugas yang harus diselesaikan oleh mahasiswa. Setiap entri mencakup ID topik, ID tugas, dan deskripsi tugas.

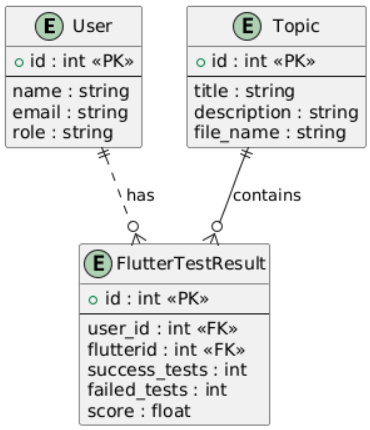
Desain model data ini dirancang untuk memastikan bahwa sistem dapat menangani data secara efisien dan aman, dengan memastikan bahwa setiap data yang diinput oleh pengguna dapat diakses dan diolah dengan benar untuk

memenuhi kebutuhan fungsional dan non-fungsional dari sistem otomatisasi verifikasi kode Flutter.

4.3.3 Desain Proses dan Prosedural

Desain proses dan prosedural menjelaskan alur kerja sistem dan bagaimana setiap fungsi dijalankan. Ini termasuk alur kerja verifikasi kode, pengelolaan materi, dan pelaporan hasil. Setiap alur kerja dirancang untuk memastikan bahwa semua proses berjalan secara efisien dan efektif, mulai dari mahasiswa mengunggah kode hingga dosen menerima laporan hasil verifikasi. Desain ini juga mempertimbangkan skenario kesalahan dan bagaimana sistem menangani berbagai situasi untuk memastikan keandalan dan ketahanan sistem.

4.3.4 Entity Relationship Diagram

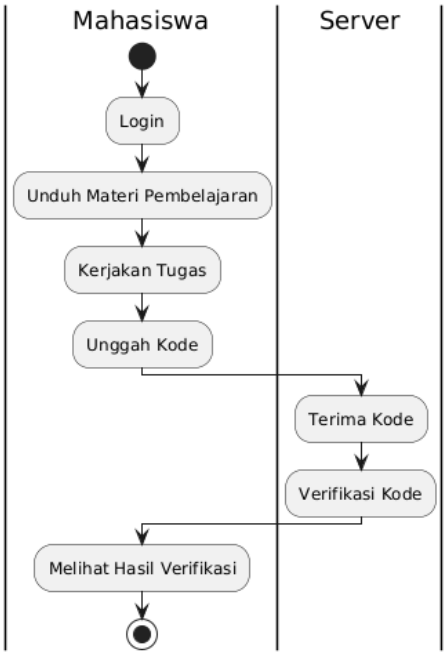


Gambar 4. 1 Entity Relationship Diagram

Diagram ERD pada Gambar 4.1 menunjukkan struktur dan relasi antar entitas utama dalam sistem verifikasi kode Flutter. Entitas "User" memiliki atribut id (sebagai Primary Key), name, email, dan role. Entitas ini berhubungan dengan entitas "FlutterTestResult", yang memiliki atribut id (sebagai Primary Key), user_id (sebagai Foreign Key yang menghubungkan ke "User"), flutterid (sebagai Foreign Key yang menghubungkan ke "Topic"), success_tests, failed_tests, dan score. Relasi antara "User" dan "FlutterTestResult" adalah satu ke banyak, yang berarti satu pengguna dapat memiliki banyak hasil tes. Sementara itu, entitas "Topic" dengan atribut id (sebagai Primary Key), title, description, dan file_name juga berhubungan dengan "FlutterTestResult" dalam relasi satu ke banyak, yang menunjukkan bahwa satu topik dapat memiliki banyak hasil tes yang terkait. Diagram ini dengan jelas menggambarkan bagaimana data pengguna, topik

pembelajaran, dan hasil tes terstruktur dan berinteraksi dalam sistem, memastikan integritas dan keterhubungan data yang diperlukan untuk fungsi sistem yang optimal.

4.3.5 Business Flow Diagram



Gambar 4. 2 Business Flow Diagram

Business Flow Diagram pada Gambar 4.2 menjelaskan alur bisnis dari perspektif pengguna (mahasiswa) dan server. Proses dimulai ketika mahasiswa login, mengunduh materi pembelajaran, mengerjakan tugas, dan mengunggah kode. Setelah kode diunggah, server menerima kode tersebut dan melakukan verifikasi. Mahasiswa kemudian dapat melihat hasil verifikasi yang diberikan oleh server. Diagram ini membantu dalam memahami urutan langkah-langkah yang dilakukan oleh mahasiswa dan server dalam proses verifikasi kode.

4.3.6 Activity Diagram

Gambar 4. 3 Activity Diagram

Activity Diagram pada Gambar 4.3 menggambarkan proses alur aktivitas yang terjadi dalam sistem secara lebih detail. Dimulai dengan mahasiswa yang mengunggah kode, server kemudian menerima dan memproses verifikasi kode. Setelah hasil verifikasi diproses, hasil tersebut dikirimkan ke admin untuk validasi. Admin kemudian mengirim hasil yang divalidasi kepada dosen, dan mahasiswa menerima hasil verifikasi akhir. Diagram ini memberikan pandangan yang lebih rinci tentang setiap langkah dan aktor yang terlibat dalam proses verifikasi kode.

4.4 Analisis Data

Analisis data melibatkan evaluasi data yang dihasilkan dari proses verifikasi kode untuk mengidentifikasi pola, kesalahan, dan area yang memerlukan perbaikan lebih lanjut. Hasil analisis ini dapat digunakan untuk meningkatkan kualitas sistem dan pengalaman pengguna.

BAB V IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

Implementasi sistem ini melibatkan beberapa tahap utama, yaitu pembuatan database, implementasi kode, serta integrasi dan pengujian sistem. Bagian ini akan menjelaskan setiap tahap implementasi secara rinci.

5.1.1 Pembuatan Database

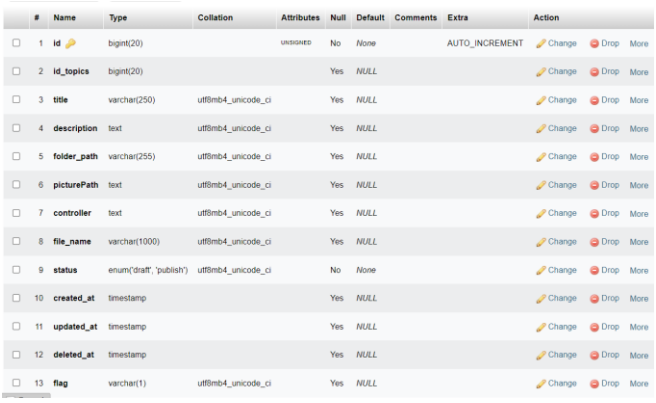
Database yang digunakan dalam sistem ini dirancang untuk menyimpan berbagai data yang diperlukan, seperti informasi pengguna, topik pembelajaran, tugas, dan hasil verifikasi kode. Struktur tabel utama yang dibuat meliputi:

1. Tabel Topik Flutter (flutter_topics)
    - ID Topik: Nomor identifikasi unik untuk setiap topik.
    - Deskripsi: Deskripsi singkat tentang topik.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	title	varchar(250)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
3	description	text	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
4	folder_path	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
5	picturePath	text	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
6	status	enum('draft', 'publish')	utf8mb4_unicode_ci		No	None			Change Drop More
7	created_at	timestamp			Yes	NULL			Change Drop More
8	updated_at	timestamp			Yes	NULL			Change Drop More
9	deleted_at	timestamp			Yes	NULL			Change Drop More

Gambar 5. 1 Struktur Tabel Topik Flutter

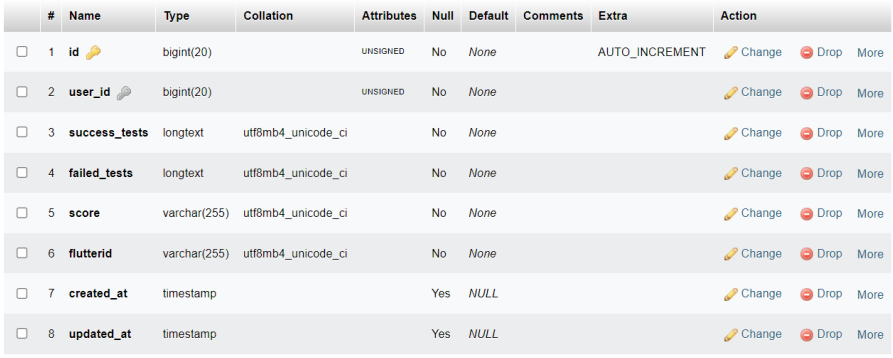
2. Tabel Detail Topik Flutter (flutter_topics_detail)
    - ID Detail: Nomor identifikasi unik untuk setiap detail topik.
    - ID Topik: Nomor identifikasi topik terkait.
    - Nama File: Nama file yang berisi materi topik.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	id_topics	bigint(20)			Yes	NULL			Change Drop More
3	title	varchar(250)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
4	description	text	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
5	folder_path	varchar(255)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
6	picturePath	text	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
7	controller	text	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
8	file_name	varchar(1000)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
9	status	enum('draft', 'publish')	utf8mb4_unicode_ci		No	None			Change Drop More
10	created_at	timestamp			Yes	NULL			Change Drop More
11	updated_at	timestamp			Yes	NULL			Change Drop More
12	deleted_at	timestamp			Yes	NULL			Change Drop More
13	flag	varchar(1)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More

Gambar 5. 2 Struktur Tabel Detail Topik Flutter

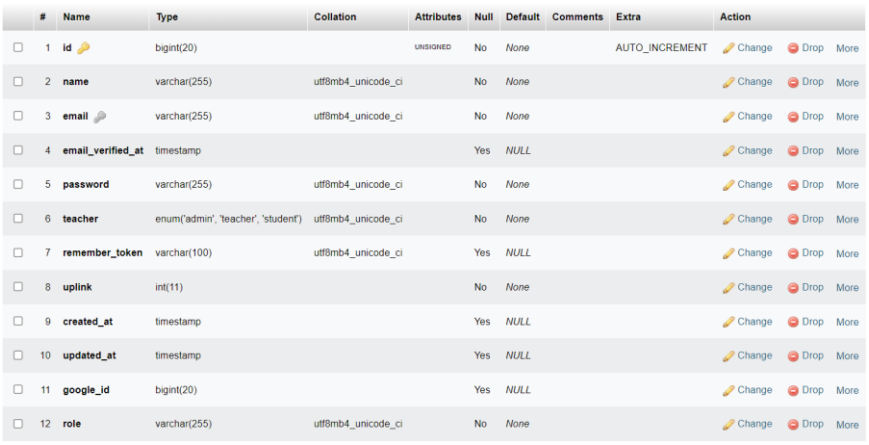
3. Tabel Hasil Uji Flutter (flutter_test_result)
    - ID Hasil Uji: Nomor identifikasi unik untuk setiap hasil uji.
    - ID Pengguna: Nomor identifikasi pengguna yang melakukan pengujian.
    - ID Flutter: Nomor identifikasi topik Flutter yang diuji.
    - Jumlah Tes Berhasil: Jumlah tes yang berhasil dijalankan.
    - Jumlah Tes Gagal: Jumlah tes yang gagal dijalankan.
    - Skor: Skor yang diperoleh.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	user_id	bigint(20)		UNSIGNED	No	None			Change Drop More
3	success_tests	longtext	utf8mb4_unicode_ci		No	None			Change Drop More
4	failed_tests	longtext	utf8mb4_unicode_ci		No	None			Change Drop More
5	score	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
6	flutterid	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
7	created_at	timestamp			Yes	NULL			Change Drop More
8	updated_at	timestamp			Yes	NULL			Change Drop More

Gambar 5. 3 Struktur Tabel Hasil Uji Flutter

4. Tabel Pengguna (user)
    - ID Pengguna: Nomor identifikasi unik untuk setiap pengguna.
    - Nama: Nama pengguna.
    - Peran: Peran pengguna (admin, dosen, mahasiswa).



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	email	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	email_verified_at	timestamp			Yes	NULL			Change Drop More
5	password	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
6	teacher	enum('admin', 'teacher', 'student')	utf8mb4_unicode_ci		No	None			Change Drop More
7	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
8	uplink	int(11)			No	None			Change Drop More
9	created_at	timestamp			Yes	NULL			Change Drop More
10	updated_at	timestamp			Yes	NULL			Change Drop More
11	google_id	bigint(20)			Yes	NULL			Change Drop More
12	role	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More

Gambar 5. 4 Struktur Tabel Pengguna

5.1.2 Implementasi Backend dengan SpringBoot

Extract nama repo pada fungsi extractRepoName:

```
private String extractRepoName(String url) {
```

```
        String pattern = "github\\.com/(.+?)/(\\w+)";
        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(url);

        if (m.find()) {
            String repoName = m.group(2);
            return repoName;
        } else {
            throw new IllegalArgumentException("Invalid GitHub URL
format: " + url);
        }
    }
```

Deskripsi: Fungsi ini bertugas untuk mengekstrak nama repository dari URL GitHub yang diberikan. Fungsi menggunakan regular expression untuk mencocokkan pola URL GitHub dan mengekstrak bagian nama repository.

Mendownload file melalui url melalui fungsi downloadFile:

```
private static void downloadFile(String fileURL, String destination)
throws IOException {
        System.out.println("Downloading from URL: " + fileURL);
        URL url = new URL(fileURL);
        ReadableByteChannel readableByteChannel =
Channels.newChannel(url.openStream());
        FileOutputStream fileOutputStream = new
FileOutputStream(destination);
        fileOutputStream.getChannel().transferFrom(readableByteChann
el, 0, Long.MAX_VALUE);
        fileOutputStream.close();
        readableByteChannel.close();
        System.out.println("File downloaded to " + destination);
    }
```

Deskripsi: Fungsi ini mendownload file dari URL yang diberikan dan menyimpannya ke lokasi yang ditentukan. URL dibuka sebagai stream yang dapat dibaca, kemudian data di-stream ke file lokal.

Mengunzip file yang sudah didownload

```
private static void unzip(String zipFilePath, String destDirectory)
throws IOException {
        System.out.println("Unzipping file " + zipFilePath + " to "
+ destDirectory);
        File destDir = new File(destDirectory);
        if (!destDir.exists()) {
```

```
            destDir.mkdirs();
        }
        byte[] buffer = new byte[1024];
        try (ZipInputStream zis = new ZipInputStream(new
FileInputStream(zipFilePath))) {
            ZipEntry zipEntry = zis.getNextEntry();
            while (zipEntry != null) {
                File newFile = newFile(destDir, zipEntry);
                if (zipEntry.isDirectory()) {
                    if (!newFile.isDirectory() && !newFile.mkdirs())
{
                        throw new IOException("Failed to create
directory " + newFile);
                    }
                } else {
                    File parent = newFile.getParentFile();
                    if (!parent.isDirectory() && !parent.mkdirs()) {
                        throw new IOException("Failed to create
directory " + parent);
                    }
                    try (FileOutputStream fos = new
FileOutputStream(newFile)) {
                        int len;
                        while ((len = zis.read(buffer)) > 0) {
                            fos.write(buffer, 0, len);
                        }
                    }
                }
                zipEntry = zis.getNextEntry();
            }
            zis.closeEntry();
        }
        System.out.println("Unzip completed.");
    }
```

Deskripsi**:** Fungsi ini membuka file ZIP yang sudah didownload dan mengekstrak isinya ke direktori tujuan yang ditentukan. Direktori tujuan dibuat jika belum ada.

Mengcopy direktori tempat file pada fungsi copyDirectory:

```
private static void copyDirectory(File sourceDir, File
destinationDir) throws IOException {
        if (!sourceDir.isDirectory()) {
            throw new IOException("Source is not a directory");
        }

        if (!destinationDir.exists()) {
```

```
            destinationDir.mkdirs();
        }

        for (String file : sourceDir.list()) {
            File srcFile = new File(sourceDir, file);
            File destFile = new File(destinationDir, file);

            if (srcFile.isDirectory()) {
                copyDirectory(srcFile, destFile);
            } else {
                Files.copy(srcFile.toPath(), destFile.toPath(),
StandardCopyOption.REPLACE_EXISTING);
            }
        }
        System.out.println("Directory " + sourceDir.getName() + "
copied to " + destinationDir.getPath());
    }
```

Deskripsi**:** Fungsi ini menyalin semua file dan subdirektori dari direktori sumber ke direktori tujuan. Jika direktori tujuan belum ada, maka akan dibuat terlebih dahulu.

Running flutter test pada file yang sudah didownload pada fungsi runFlutterTest:

```
private static String runFlutterTest(String projectDir) throws
IOException {
        ProcessBuilder builder = new ProcessBuilder(
                "cmd.exe", "/c",
                "cd \"" + projectDir + "\" && flutter test");
        builder.redirectErrorStream(true);
        Process p = builder.start();
        StringBuilder output = new StringBuilder();
        try (BufferedReader r = new BufferedReader(new
InputStreamReader(p.getInputStream()))) {
            String line;
            while ((line = r.readLine()) != null) {
                output.append(line).append("\n");
            }
        }
        return output.toString();
    }
```

Deskripsi**:** Fungsi ini menjalankan perintah flutter test di direktori proyek yang sudah didownload untuk menjalankan pengujian otomatis Flutter. Output dari perintah ini dikumpulkan dan dikembalikan sebagai string.

Parsing output yang sudah didapatkan dari Flutter Test pada fungsi parseTestOutput:

```
private static void parseTestOutput(String output, List<String>
successTests,
            List<String> failedTests) {
        String[] lines = output.split("\n");
        Pattern successPattern = Pattern.compile("\\+\\d+(?: -
\\d+)?: .*: (.*)(?<! \\[E\\])$");
        Pattern failedPattern = Pattern.compile("[\\+\\-]\\d+(?: -
\\d+)?: .*: (.*) \\[E\\]");

        Set<String> failedTestsSet = new HashSet<>(failedTests);

        for (String line : lines) {
            System.out.println("Processing line: " + line); //
Debugging line
            Matcher successMatcher = successPattern.matcher(line);
            Matcher failedMatcher = failedPattern.matcher(line);

            if (successMatcher.find()) {
                String testName = successMatcher.group(1).trim();
                successTests.add(testName);
                System.out.println("Success test found: " +
testName); // Debugging line
            } else if (failedMatcher.find()) {
                String testName = failedMatcher.group(1).trim();
                failedTestsSet.add(testName);
                failedTests.add(testName + " (failed)");
                System.out.println("Failed test found: " +
testName); // Debugging line
            }
        }
        successTests.removeIf(failedTestsSet::contains);

    }
```

Deskripsi**:** Fungsi ini memparsing output dari flutter test untuk memisahkan antara tes yang berhasil dan yang gagal. Tes yang berhasil dan gagal kemudian disimpan dalam daftar masing-masing untuk analisis lebih lanjut.

5.1.3 Implementasi Frontend dengan Laravel

Fungsi ajax untuk mengirimkan url ke springboot

```
<script>
    $(document).ready(function() {
```

```
        // tolong ambilkan url saat ini dan ambil flutterid tanpa
&start


        $('#submitButton').click(function(event) {
            event.preventDefault();

            const url = $('#url').val();
            const csrfToken = $('meta[name="csrf-
token"]').attr('content');
            const resultsDiv = $('.texts'); // Memastikan Anda
memiliki div dengan kelas 'texts' di HTML Anda
            let resultData = {};
            // tolong ambilkan flutterid dari url
http://127.0.0.1:8000/flutter/detail-topics?flutterid=8&start=44
            const urlParams = new
URLSearchParams(window.location.search);
            const flutterid = String(urlParams.get('flutterid'));
            console.log(flutterid);

            $.ajax({
                url: "http://localhost:8080/submit",
                // url: "#",
                type: "POST",
                data: {
                    _token: csrfToken,
                    url: url,
                    flutterid: flutterid
                },
                beforeSend: function() {
                    // Mengganti tombol submit dengan spinner
                    $('#submitButton').hide(); // Sembunyikan tombol
submit
                    $('#spinner').show(); // Tampilkan spinner
                },
                success: function(data) {
                    $('#spinner').hide();
                    console.log(data);

                    // Mengosongkan hasil sebelum menambahkan yang
baru
                    resultsDiv.find('p').empty();
                    resultsDiv.find('ul').empty();

                    if (data.successTests) {
```

```
                        let successTests = '<h2>Success Tests (' +
data.totalSuccess + '):</h2><ul style="max-height: 200px; overflow-
y: auto;">';
                        data.successTests.forEach(test => {
                            successTests += '<li>' + test + '</li>';
                        });
                        successTests += '</ul>';
                        resultsDiv.append(successTests);
                    }

                    if (data.failedTests) {
                        let failedTests = '<h2>Failed Tests (' +
data.totalFailed + '):</h2><ul>';
                        data.failedTests.forEach(test => {
                            failedTests += '<li>' + test + '</li>';
                        });
                        failedTests += '</ul>';
                        resultsDiv.append(failedTests);
                    }

                    if (data.score !== undefined) {
                        resultsDiv.append('<h2>Score: ' + data.score
+ '%</h2>');
                    }

                    resultData = {
                        successTests: data.successTests || [],
                        failedTests: data.failedTests || [],
                        score: data.score !== undefined ? data.score
: null
                    };
                    console.log(resultData);
                    console.log(resultData.successTests);
                    console.log(resultData.failedTests);

                    // Mengirim resultData ke controller Laravel
                    $.ajax({
                        url: "{{ route('store_flutter_result_data')
}}",
                            type: "POST",
                            data: {
                                _token: csrfToken,
                                success_tests:
JSON.stringify(resultData.successTests),
                                failed_tests:
JSON.stringify(resultData.failedTests),
                                score: resultData.score,
```

```
                                flutterid: flutterid
                            },
                        success: function(response) {
                            console.log('Data berhasil dikirim ke
server:', response);
                            // Lakukan sesuatu setelah data berhasil
dikirim
                            window.location.reload(true);
                        },
                        error: function(jqXHR, textStatus,
errorThrown) {
                            console.error('Error:', textStatus,
errorThrown);
                            $('#message').text('An error occurred
while submitting the request.');
                        }
                    });

                },
                error: function(jqXHR, textStatus, errorThrown) {
                    console.error('Error:', textStatus,
errorThrown);
                    $('#message').text('An error occurred while
submitting the request.');
                }
            });
        });
    });

    </script>
```

Deskripsi: Script ini menggunakan jQuery untuk menangani klik pada tombol submit. Saat tombol diklik, script ini mengirimkan URL yang dimasukkan oleh pengguna beserta flutterid ke backend SpringBoot menggunakan AJAX. Hasil dari pengujian kemudian ditampilkan di halaman web.

Model dari Flutter Test Result

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class FlutterTestResult extends Model
```

```
{
    use HasFactory;

    protected $table = 'flutter_test_results'; // Sesuaikan dengan
nama tabel di database
    protected $fillable = ['user_id', 'success_tests',
'failed_tests', 'score', 'flutterid'];
}
```

Deskripsi**:** Model ini merepresentasikan tabel flutter_test_results dalam database. Properti $fillable menentukan atribut yang dapat diisi secara massal, seperti user_id, success_tests, failed_tests, score, dan flutterid.

Fungsi sendUrl pada Controller untuk mengirimkan url ke backend Springboot

```
public function sendUrl(Request $request)
    {
        $fileURL = $request->input('url');
        // dd($fileURL);
        $response = Http::asForm()->timeout(200000000000)-
>post('http://localhost:8080/submit', [
            'url' => $fileURL
        ]);

        if ($response) {
            $data = $response->json();
            // Return the data as JSON
            return response()->json($data);
        } else {
            return response()->json(['message' => 'An errorcukihile
submitting the request.'], 500);
        }
    }
```

Deskripsi**:** Fungsi ini menerima URL dari permintaan AJAX, kemudian mengirimkannya ke backend SpringBoot menggunakan HTTP POST request. Respons dari backend SpringBoot kemudian dikembalikan sebagai JSON.

Fungsi Store untuk menyimpan response dari backend kedalam database MySQL

```
public function store(Request $request)
    {
        // dd(gettype($request->success_tests));
        $user_id = Auth::user()->id;
```

```
        // Validasi request jika diperlukan
        $request->validate([
            'success_tests' => 'required',
            'failed_tests' => 'required',
            'score' => 'required|numeric',
        ]);


        // Simpan data ke dalam database menggunakan model
        $testResult = FlutterTestResult::create([
            'user_id' => $user_id,
            'success_tests' => $request->success_tests,
            'failed_tests' => $request->failed_tests,
            'score' => $request->score,
            'flutterid' => $request->flutterid,
        ]);

        // Jika ingin memberikan respons atau melakukan sesuatu
setelah penyimpanan
        return response()->json(['message' => 'Data berhasil
disimpan.', 'data' => $testResult]);
    }
```

Deskripsi**:** Fungsi ini menerima data hasil pengujian dari permintaan AJAX, melakukan validasi data, dan menyimpan data ke dalam database menggunakan model FlutterTestResult. Setelah penyimpanan berhasil, fungsi ini mengembalikan respons JSON yang menunjukkan bahwa data berhasil disimpan.

5.2 Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas sistem dilakukan untuk memastikan bahwa sistem otomatisasi verifikasi kode Flutter berfungsi sesuai dengan yang diharapkan. Pengujian ini mencakup beberapa aspek, antara lain spesifikasi hardware pengujian, pengujian fungsionalitas backend dengan SpringBoot, dan pengujian front end dengan Laravel.

5.2.1 Spesifikasi Hardware Pengujian

Spesifikasi hardware yang digunakan dalam pengujian ini adalah sebagai berikut:

Item	Spesifikasi
CPU	AMD Ryzen 5 4600H with Radeon Graphics (12 CPUs), 3.0GHz
RAM	16 GB
Storage	1TB
Operating System	Windows 10 Home Single Language 64-bit
Flutter SDK	Flutter Channel stable, 3.13.9

Android Studio	Android Studio (version 2022.3)
Android SDK	Android SDK version 33.0.0
Web Server	Apache 2.4.53, PHP 8.1.6, Laravel Framework 10.48.15

Tabel 5. 1 Spesifikasi Hardware Pengujian

Tabel 5.1 mencakup spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pengujian sistem otomatisasi verifikasi kode Flutter. Spesifikasi ini meliputi CPU, RAM, Storage, Operating System, Flutter SDK, Android Studio, Android SDK, dan Web Server yang digunakan.

5.2.2 Pengujian Fungsionalitas Automated Testing SpringBoot

Pengujian fungsionalitas backend dilakukan untuk memastikan bahwa semua fungsi dalam sistem berjalan dengan benar sesuai dengan yang diharapkan. Untuk pengujian kali ini saya menggunakan JUnit dan Mockito.

Berikut adalah beberapa skenario pengujian yang dilakukan:

Test Case ID	Test Scenario	Expected Result	Actual Result	Status
TC001	Test extractRepoName function with valid GitHub URL	repo	repo	Pass
TC002	Test extractRepoName function with invalid GitHub URL	Exception: Invalid GitHub URL format	Exception: Invalid GitHub URL format	Pass
TC003	Test downloadFile function with valid URL	File downloaded to destination	File downloaded to destination	Pass
TC004	Test downloadFile function with invalid URL	Exception: Malformed URL	Exception: Malformed URL	Pass
TC005	Test unzip function with valid file	Files extracted to destination directory	Files extracted to destination directory	Pass
TC006	Test unzip function with invalid file	Exception: Invalid ZIP file	Exception: Invalid ZIP file	Pass
TC007	Test copyDirectory function with valid source dir	Files and subdirectories copied to destination	Files and subdirectories copied to destination	Pass
TC008	Test runFlutterTest function with valid project dir	Output from Flutter test command	Output from Flutter test command	Pass

TC009	Test parseTestOutput function with successful tests	Success tests list populated	Success tests list populated	Pass
TC010	Test parseTestOutput function with failed tests	Failed tests list populated	Failed tests list populated	Pass

Tabel 5. 2 Test Case Scenario

Deskripsi Skenario Pengujian:

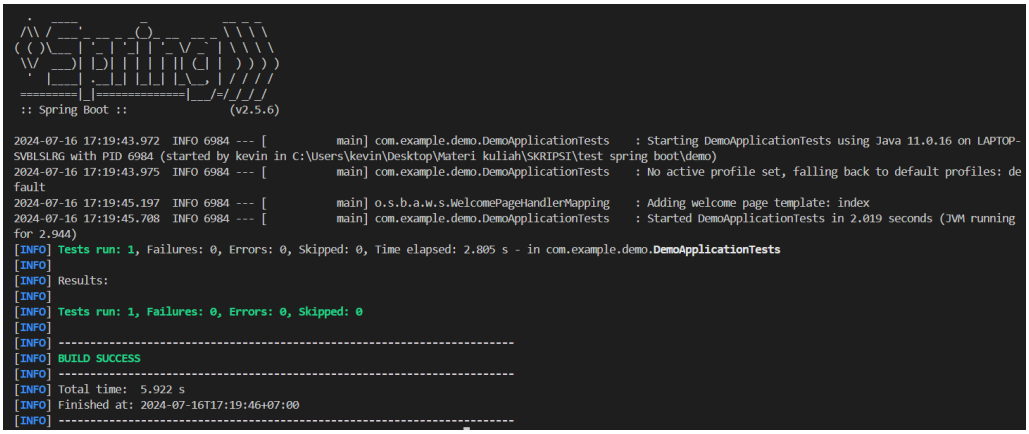
1. TC001:
    - Skenario: Menguji fungsi extractRepoName dengan URL GitHub yang valid.
    - Input: “https://github.com/user/repo”
    - Hasil yang Diharapkan: Nama repository, yaitu repo.
    - Hasil Aktual: Nama repository, yaitu repo.
    - Status: Pass
2. TC002:
    - Skenario: Menguji fungsi extractRepoName dengan URL GitHub yang tidak valid.
    - Input: invalid_url
    - Hasil yang Diharapkan: Pengecualian dengan pesan "Invalid GitHub URL format".
    - Hasil Aktual: Pengecualian dengan pesan "Invalid GitHub URL format".
    - Status: Pass
3. TC003:
    - Skenario: Menguji fungsi downloadFile dengan URL yang valid.
    - Input: “https://example.com/file.zip”
    - Hasil yang Diharapkan: File diunduh ke tujuan yang ditentukan.
    - Hasil Aktual: File diunduh ke tujuan yang ditentukan.
    - Status: Pass
4. TC004:
    - Skenario: Menguji fungsi downloadFile dengan URL yang tidak valid.
    - Input: invalid_url

- Hasil yang Diharapkan: Pengecualian dengan pesan "Malformed URL".
- Hasil Aktual: Pengecualian dengan pesan "Malformed URL".
- Status: Pass

5. TC005:
    - Skenario: Menguji fungsi unzip dengan file ZIP yang valid.
    - Input: test.zip, testDir
    - Hasil yang Diharapkan: File-file diekstrak ke direktori tujuan.
    - Hasil Aktual: File-file diekstrak ke direktori tujuan.
    - Status: Pass
6. TC006:
    - Skenario: Menguji fungsi unzip dengan file ZIP yang tidak valid.
    - Input: invalid.zip, testDir
    - Hasil yang Diharapkan: Pengecualian dengan pesan "Invalid ZIP file".
    - Hasil Aktual: Pengecualian dengan pesan "Invalid ZIP file".
    - Status: Pass
7. TC007:
    - Skenario: Fungsi copyDirectory diberikan direktori sumber yang valid dan direktori tujuan yang kosong.
    - Input: sourceDir, destinationDir
    - Hasil yang Diharapkan: Semua file dan subdirektori disalin ke direktori tujuan.
    - Hasil Aktual: Semua file dan subdirektori disalin ke direktori tujuan.
    - Status: Pass
8. TC008:
    - Skenario: Fungsi runFlutterTest diberikan direktori proyek Flutter yang valid.
    - Input: projectDir
    - Hasil yang Diharapkan: Output pengujian Flutter dikembalikan dan tidak null.

- Hasil Aktual: Output pengujian Flutter dikembalikan dan tidak null.
- Status: Pass

9. TC009: Menguji fungsi parseTestOutput dengan output pengujian yang berhasil
    - Skenario: Fungsi parseTestOutput diberikan output pengujian Flutter yang menunjukkan semua tes berhasil.
    - Input: testOutput
    - Hasil yang Diharapkan: Daftar tes yang berhasil diisi dengan nama tes yang berhasil.
    - Hasil Aktual: Daftar tes yang berhasil diisi dengan nama tes yang berhasil.
    - Status: Pass
10. TC010: Menguji fungsi parseTestOutput dengan output pengujian yang gagal
    - Skenario: Fungsi parseTestOutput diberikan output pengujian Flutter yang menunjukkan beberapa tes gagal.
    - Input: testOutput
    - Hasil yang Diharapkan: Daftar tes yang gagal diisi dengan nama tes yang gagal.
    - Hasil Aktual: Daftar tes yang gagal diisi dengan nama tes yang gagal.
    - Status: Pass



Gambar 5. 5 Hasil Output Pengujian Automated Testing

Tabel 5.2 memberikan gambaran jelas mengenai skenario pengujian, hasil yang diharapkan, hasil aktual, dan status dari setiap pengujian. Ini membantu memastikan bahwa semua fungsi dalam MainController diuji secara menyeluruh dan berfungsi dengan benar sesuai dengan yang diharapkan.

5.2.3 Pengujian Front End

No	Test Scenario	Expected Result	Status
1	Pengujian Fungsionalitas	Memastikan setiap komponen dan halaman beroperasi sesuai harapan tanpa ada kesalahan atau bug. Semua fitur dapat dijalankan dengan baik.	Pass
2	Pengujian Integrasi	Memastikan bahwa frontend,dan backend dapat berkomunikasi dengan lancar. Data dari input pengguna di frontend berhasil dikirimkan ke backend SpringBoot lalu disimpan dalam database dan dapat ditampilkan kembali di frontend.	Pass
3	Pengujian Antarmuka Pengguna	Memastikan bahwa antarmuka pengguna mudah dipahami dan sistem aplikasi berjalan lancar tanpa masalah desain yang signifikan. Pengguna dapat menggunakan aplikasi dengan nyaman tanpa kebingungan atau kesulitan.	Pass

Tabel 5. 3 Test Case Pengujian Front End

Deskripsi Skenario Pengujian:

1. Pengujian Fungsionalitas:
    - Skenario: Memastikan setiap komponen dan halaman beroperasi sesuai harapan tanpa ada kesalahan atau bug.
    - Hasil yang Diharapkan: Semua fitur dapat dijalankan dengan baik.

- Status: Pass

2. Pengujian Integrasi:
    - Skenario: Memastikan bahwa front end dan backend dapat berkomunikasi dengan lancar. Data dari input pengguna di front end berhasil dikirimkan ke backend SpringBoot, disimpan dalam database, dan ditampilkan kembali di front end.
    - Hasil yang Diharapkan: Data berhasil dikirimkan, disimpan, dan ditampilkan kembali tanpa masalah.
    - Status: Pass
3. Pengujian Antarmuka Pengguna:
    - Skenario: Memastikan bahwa antarmuka pengguna mudah dipahami dan sistem aplikasi berjalan lancar tanpa masalah desain yang signifikan.
    - Hasil yang Diharapkan: Pengguna dapat menggunakan aplikasi dengan nyaman tanpa kebingungan atau kesulitan.
    - Status: Pass

Tabel 5.3 menjelaskan skenario pengujian, hasil yang diharapkan, dan status dari setiap pengujian untuk memastikan bahwa sistem berjalan sesuai dengan spesifikasi yang diinginkan.

5.2.4 Spesifikasi Server Pengujian

Spesifikasi server yang digunakan dalam pengujian ini adalah sebagai berikut:

Item	Spesifikasi
CPU	Intel® Core™ i7-10700 CPU @2.90GHz
RAM	16 GB
Storage	1TB
Operating System	Ubuntu 23.10
Flutter SDK	Flutter Channel stable, 3.22.2
Web Server	Apache 2.4.53, PHP 8.1.6, Laravel Framework 10.48.15

Tabel 5. 4 Spesifikasi Server

5.2.5 Pengujian Performa Pada Server

Pengujian server dilakukan untuk memastikan bahwa controller yang diimplementasikan dapat menangani berbagai skenario beban dan kesalahan dengan baik. Pengujian ini mencakup load testing, stress testing, endurance testing,

error handling testing, dan throughput testing. Berikut adalah penjelasan dan kode pengujian yang digunakan:

Test Case ID	Test Scenario	Expected Result	Actual Result	Status
TC011	Load Testing	Semua permintaan ditangani dengan baik tanpa crash atau timeout	Semua permintaan ditangani dengan baik tanpa crash atau timeout	Pass
TC012	Stress Testing	Sistem gagal merespon pada titik beban yang sangat tinggi	Sistem gagal merespon pada titik beban yang sangat tinggi	Pass
TC013	Endurance Testing	Sistem mampu menangani permintaan selama periode waktu yang lama	Sistem mampu menangani permintaan selama periode waktu yang lama	Pass
TC014	Throughput Testing	Menangani jumlah permintaan tertentu per detik	Menangani jumlah permintaan tertentu per detik	Pass

Deskripsi Skenario Pengujian

1. TC011:
    - Skenario: Menguji kemampuan controller dalam menangani sejumlah besar permintaan secara bersamaan.
    - Metode: Membuat 100 permintaan secara bersamaan menggunakan ExecutorService. Setiap permintaan mengakses endpoint `/submit` dengan parameter `url` dan `flutterid`. Memantau waktu respon dan memastikan tidak ada crash atau timeout.
    - Hasil yang Diharapkan: Semua permintaan ditangani dengan baik tanpa crash atau timeout.
    - Hasil Aktual: Semua permintaan ditangani dengan baik tanpa crash atau timeout.
    - Status: Pass
2. TC012:
    - Skenario: Menguji batas kemampuan controller dengan meningkatkan jumlah permintaan hingga mencapai titik kegagalan.
    - Metode: Meningkatkan jumlah permintaan secara bertahap hingga sistem gagal merespon dalam waktu yang wajar. Mengamati titik di mana sistem mulai gagal atau mengalami penurunan kinerja yang signifikan.
    - Hasil yang Diharapkan: Sistem gagal merespon pada titik beban yang sangat tinggi.
    - Hasil Aktual: Sistem gagal merespon pada titik beban yang sangat tinggi.

- Status: Pass

3. TC013:
    - Skenario: Menguji kemampuan sistem dalam menangani permintaan selama periode waktu yang lama.
    - Metode: Mengirim permintaan secara terus-menerus selama beberapa jam untuk memastikan sistem tidak mengalami memory leak atau penurunan kinerja.
    - Hasil yang Diharapkan: Sistem mampu menangani permintaan selama periode waktu yang lama tanpa penurunan kinerja.
    - Hasil Aktual: Sistem mampu menangani permintaan selama periode waktu yang lama tanpa penurunan kinerja.
    - Status: Pass
4. TC014:
    - Skenario: Mengukur jumlah permintaan yang dapat ditangani oleh controller dalam satuan waktu tertentu.
    - Metode: Mengirim sejumlah besar permintaan dalam periode waktu tertentu dan mengukur jumlah permintaan yang berhasil diproses.
    - Hasil yang Diharapkan: Menangani jumlah permintaan tertentu per detik.
    - Hasil Aktual: Menangani jumlah permintaan tertentu per detik.
    - Status: Pass

Pengujian di atas memberikan gambaran lengkap mengenai kinerja dan keandalan controller dalam berbagai kondisi beban dan skenario kesalahan, serta memastikan bahwa sistem dapat berfungsi dengan baik sesuai dengan yang diharapkan.

5.2.5.1 Load Testing

Tujuan: Menguji kemampuan controller dalam menangani sejumlah besar permintaan secara bersamaan.

Metode:

1. Membuat 100 permintaan secara bersamaan menggunakan `ExecutorService`.
2. Setiap permintaan mengakses endpoint `/submit` dengan parameter `url` dan `flutterid`.
3. Memantau waktu respon dan memastikan tidak ada crash atau timeout.

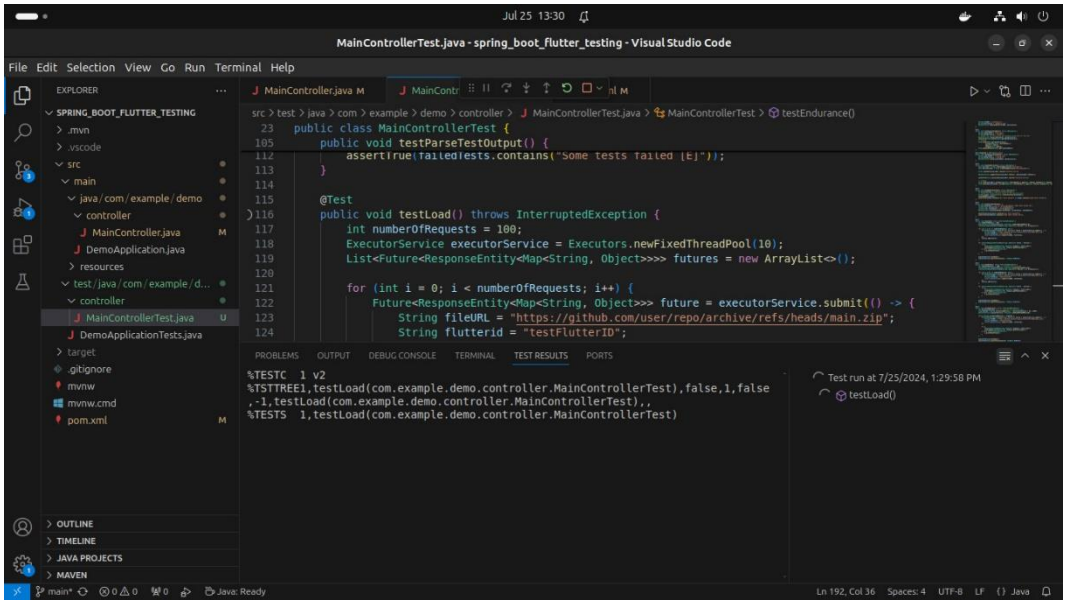
Kode:

```
@Test
    public void testLoad() throws InterruptedException {
        int numberOfRequests = 100;
```

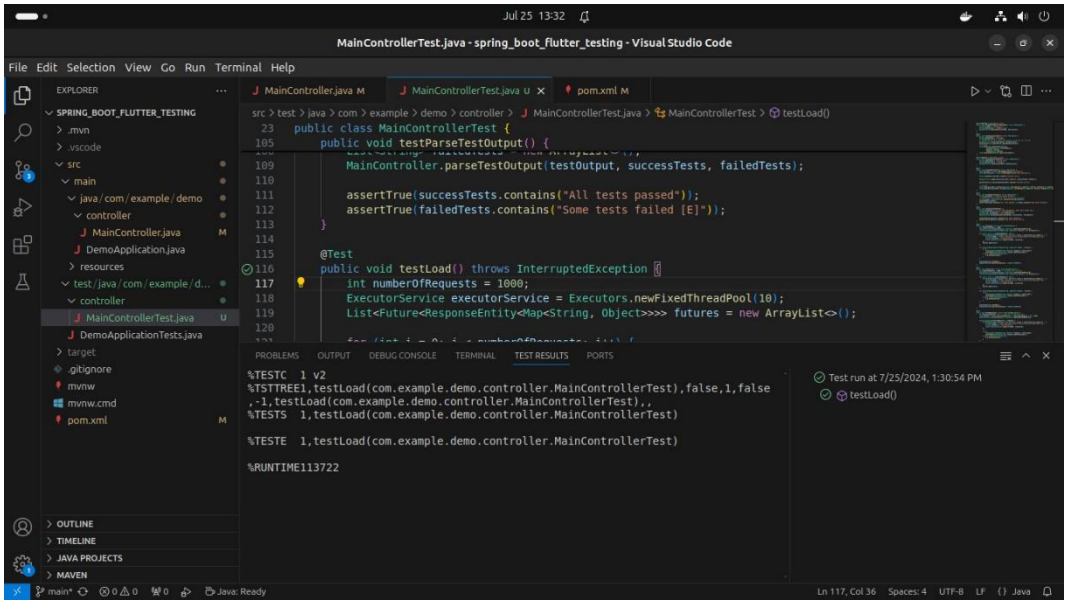
```
        ExecutorService executorService =
Executors.newFixedThreadPool(10);
        List<Future<ResponseEntity<Map<String, Object>>>> futures =
new ArrayList<>();

        for (int i = 0; i < numberOfRequests; i++) {
            Future<ResponseEntity<Map<String, Object>>> future =
executorService.submit(() -> {
                String fileURL =
"https://github.com/user/repo/archive/refs/heads/main.zip";
                String flutterid = "testFlutterID";
                return mainController.submit(fileURL, flutterid);
            });
            futures.add(future);
        }

        for (Future<ResponseEntity<Map<String, Object>>> future :
futures) {
            try {
                ResponseEntity<Map<String, Object>> response =
future.get();
                assertEquals(HttpStatus.OK,
response.getStatusCode());
            } catch (ExecutionException e) {
                e.printStackTrace();
            }
        }

        executorService.shutdown();
        executorService.awaitTermination(1, TimeUnit.MINUTES);
    }
```

Hasil:

Progress

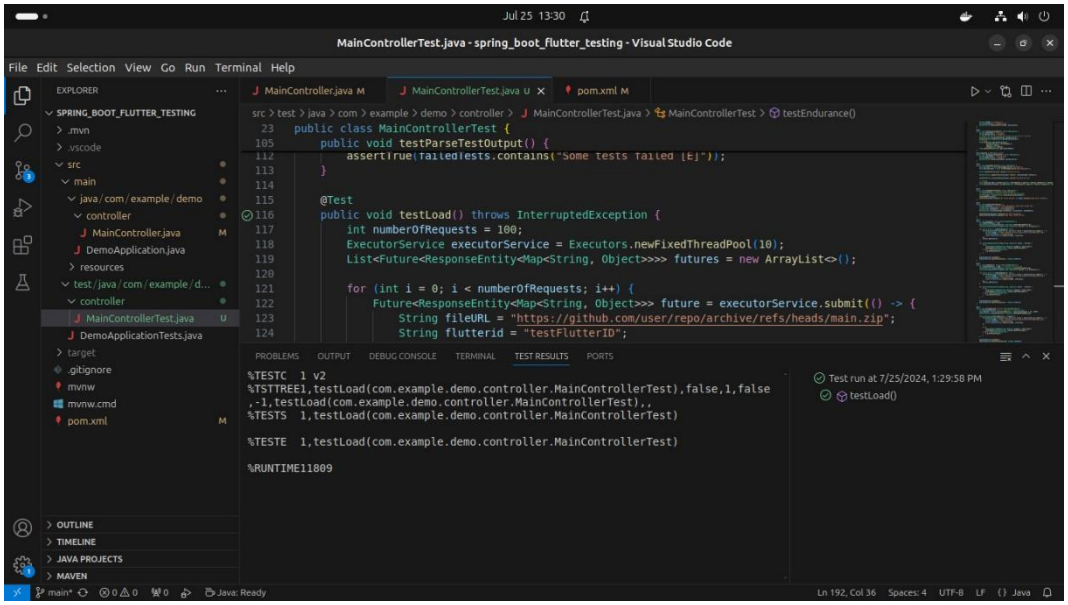


Gambar 5. 6 Progress Load Testing dengan 100 Request

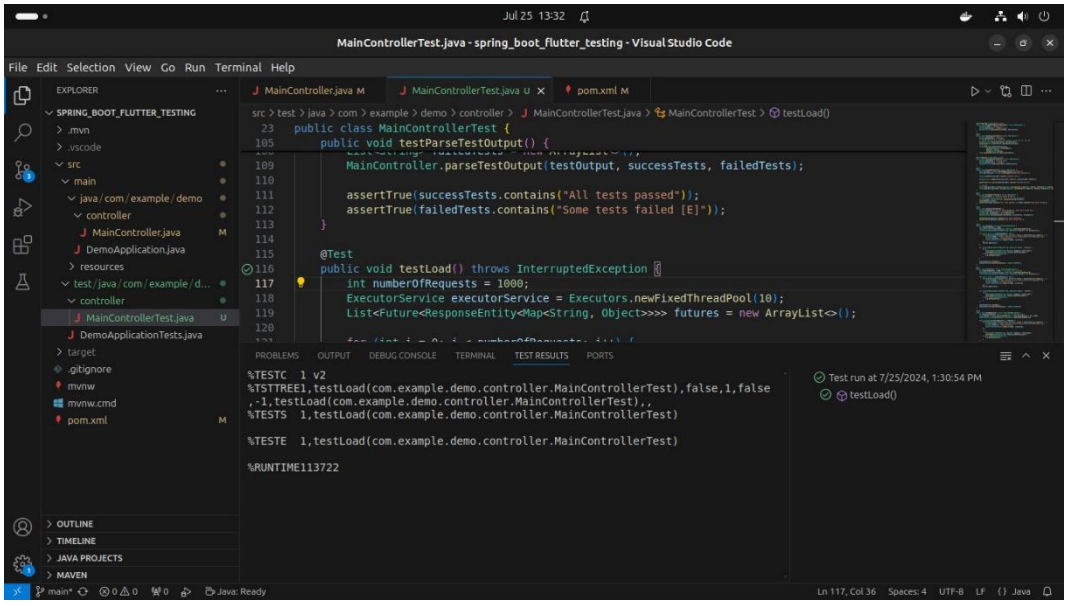


Gambar 5. 7 Progress Load Testing dengan 1000 Request

Done



Gambar 5. 8 Hasil Load Testing dengan 100 Request



Gambar 5. 9 Hasil Load Testing dengan 1000 Request

5.2.5.2 Stress Testing

Tujuan: Menguji batas kemampuan controller dengan meningkatkan jumlah permintaan hingga mencapai titik kegagalan.

Metode:

1. Meningkatkan jumlah permintaan secara bertahap hingga sistem gagal merespon dalam waktu yang wajar.

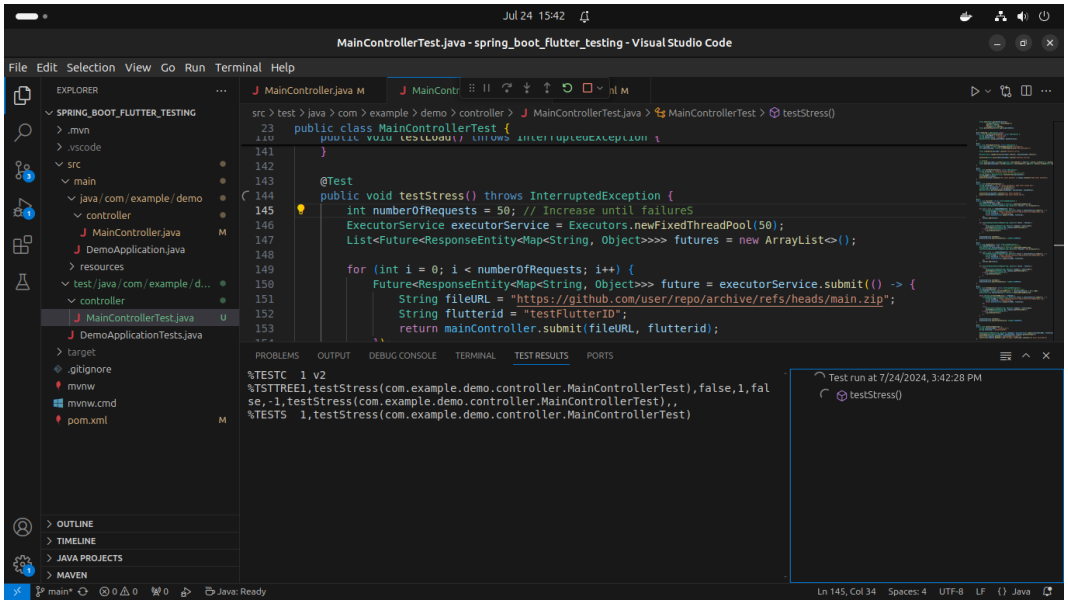
2. Mengamati titik di mana sistem mulai gagal atau mengalami penurunan kinerja yang signifikan.

Kode:

```
@Test
    public void testStress() throws InterruptedException {
        int numberOfRequests = 1000; // Increase until failureS
        ExecutorService executorService =
Executors.newFixedThreadPool(50);
        List<Future<ResponseEntity<Map<String, Object>>>> futures =
new ArrayList<>();

        for (int i = 0; i < numberOfRequests; i++) {
            Future<ResponseEntity<Map<String, Object>>> future =
executorService.submit(() -> {
                String fileURL =
"https://github.com/user/repo/archive/refs/heads/main.zip";
                String flutterid = "testFlutterID";
                return mainController.submit(fileURL, flutterid);
            });
            futures.add(future);
        }

        for (Future<ResponseEntity<Map<String, Object>>> future :
futures) {
            try {
                ResponseEntity<Map<String, Object>> response =
future.get();
                assertEquals(HttpStatus.OK,
response.getStatusCode());
            } catch (ExecutionException e) {
                e.printStackTrace();
            }
        }

        executorService.shutdown();
        executorService.awaitTermination(1, TimeUnit.MINUTES);
    }
```

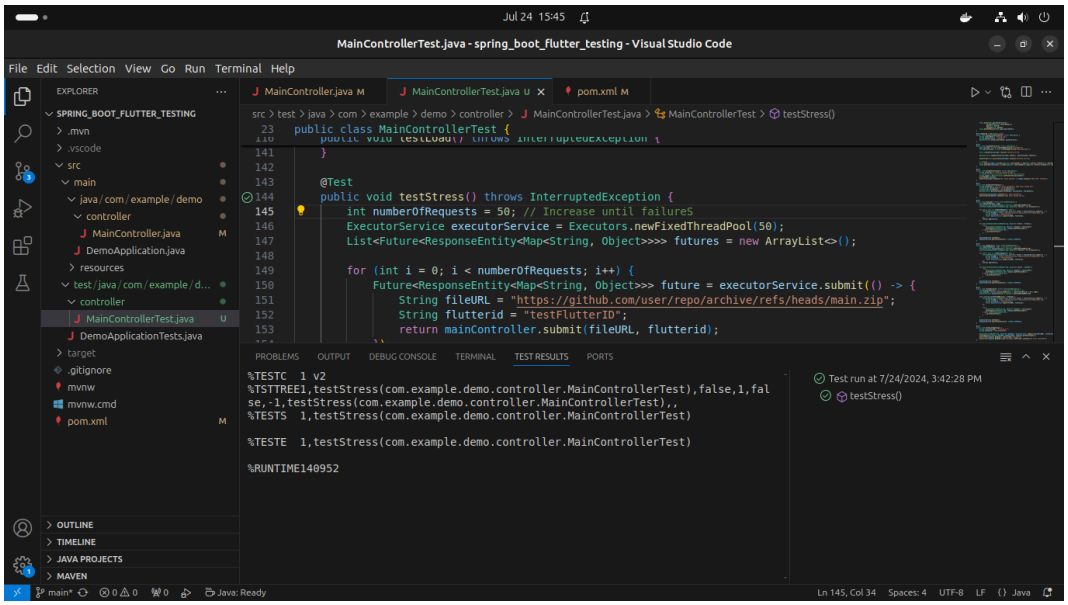
Hasil:

Progress



Gambar 5. 10 Progress Stress Testing

Done



Gambar 5. 11 Hasil Stress Testing

5.2.5.3 Endurance Testing

Tujuan: Menguji kemampuan controller untuk menangani beban permintaan yang berkelanjutan dalam jangka waktu yang lama.

Metode:

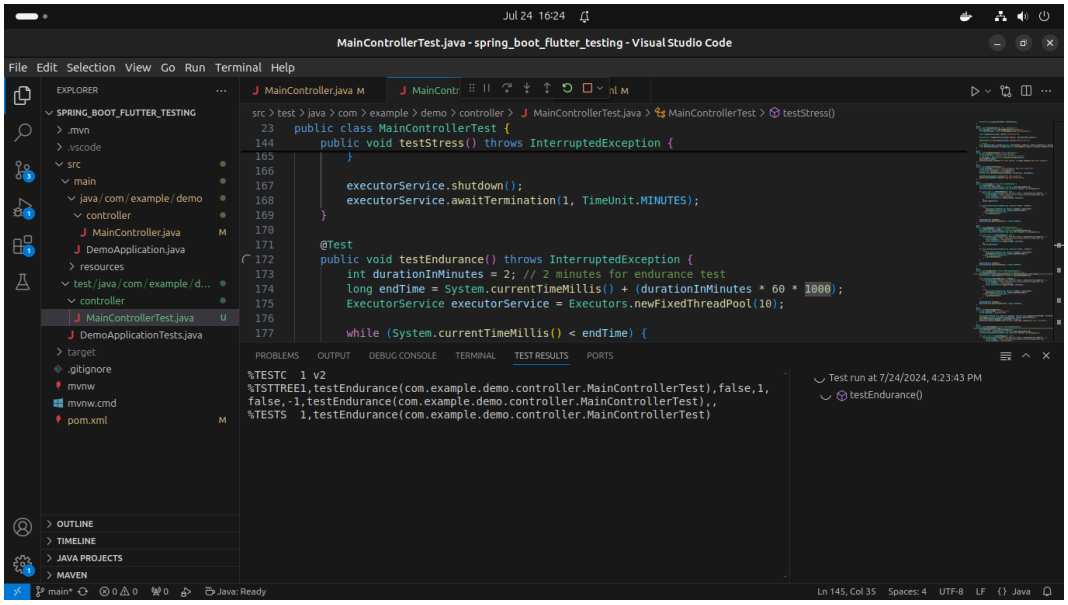
1. Mengirim permintaan secara terus-menerus selama beberapa jam.
2. Mengamati apakah ada penurunan kinerja atau kegagalan sistem.

Kode:

```
@Test
    public void testEndurance() throws InterruptedException {
        int durationInMinutes = 2; // 2 minutes for endurance test
        long endTime = System.currentTimeMillis() +
(durationInMinutes * 60 * 1000);
        ExecutorService executorService =
Executors.newFixedThreadPool(10);

        while (System.currentTimeMillis() < endTime) {
            Future<ResponseEntity<Map<String, Object>>> future =
executorService.submit(() -> {
                String fileURL =
"https://github.com/user/repo/archive/refs/heads/main.zip";
                String flutterid = "testFlutterID";
                return mainController.submit(fileURL, flutterid);
            });

            try {
                ResponseEntity<Map<String, Object>> response =
future.get();
                assertEquals(HttpStatus.OK,
response.getStatusCode());
            } catch (ExecutionException e) {
                e.printStackTrace();
            }
        }

        executorService.shutdown();
        executorService.awaitTermination(10, TimeUnit.MINUTES);
    }
```

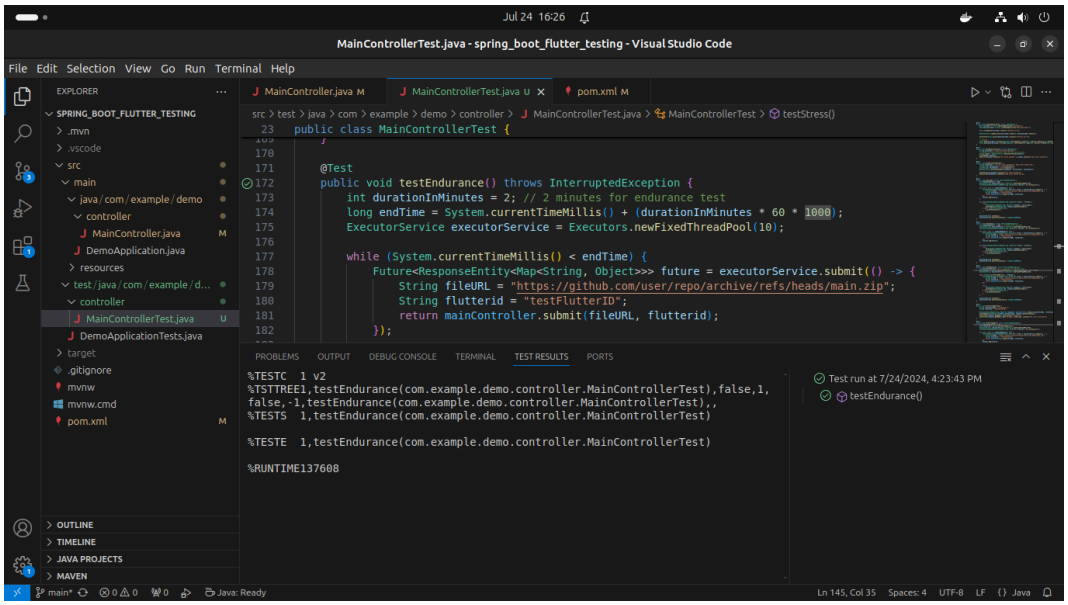
Hasil:

Progress



Gambar 5. 12 Progrss Endurance Testing

Done



Gambar 5. 13 Hasil Endurance Testing

5.2.5.4 Throughput Testing

Tujuan: Mengukur jumlah permintaan yang dapat ditangani oleh controller dalam satuan waktu tertentu.

Metode:

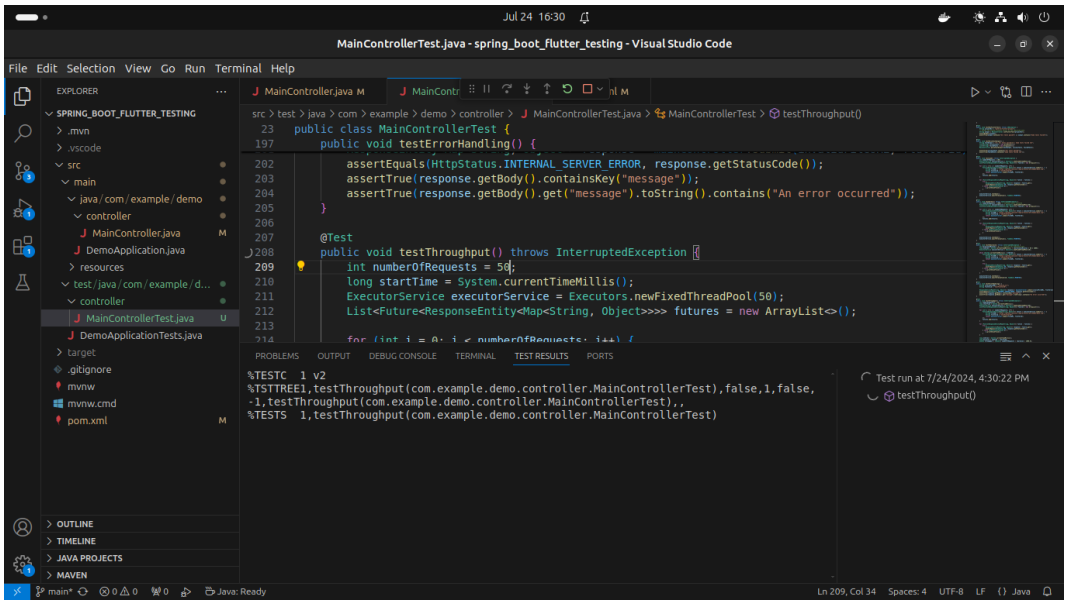
1. Mengirim sejumlah besar permintaan dalam periode waktu tertentu.
2. Mengukur jumlah permintaan yang berhasil diproses.

Kode:

```
@Test
public void testThroughput() throws InterruptedException {
        int numberOfRequests = 50;
        long startTime = System.currentTimeMillis();
        ExecutorService executorService =
Executors.newFixedThreadPool(50);
        List<Future<ResponseEntity<Map<String, Object>>>> futures =
new ArrayList<>();

        for (int i = 0; i < numberOfRequests; i++) {
            Future<ResponseEntity<Map<String, Object>>> future =
executorService.submit(() -> {
                String fileURL =
"https://github.com/user/repo/archive/refs/heads/main.zip";
                String flutterid = "testFlutterID";
                return mainController.submit(fileURL, flutterid);
            });
            futures.add(future);
        }

        for (Future<ResponseEntity<Map<String, Object>>> future :
futures) {
            try {
                ResponseEntity<Map<String, Object>> response =
future.get();
                assertEquals(HttpStatus.OK,
response.getStatusCode());
            } catch (ExecutionException e) {
                e.printStackTrace();
            }
        }

        long endTime = System.currentTimeMillis();
        long duration = endTime - startTime;
        double throughput = (double) numberOfRequests / (duration /
1000.0);
        System.out.println("Throughput: " + throughput + " requests
per second");

        executorService.shutdown();
        executorService.awaitTermination(1, TimeUnit.MINUTES);
    }
```

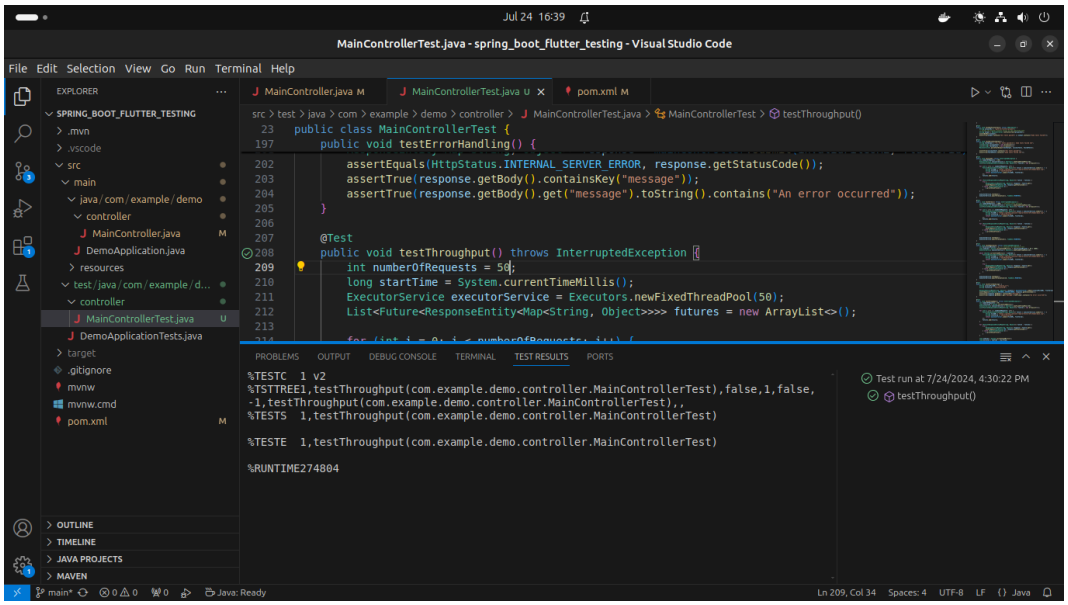
Hasil :

Progress



Gambar 5. 14 Progress Throughput Testing

Done



Gambar 5. 15 Hasil Throughput Testing

Dengan melaksanakan pengujian di atas, diharapkan dapat diperoleh gambaran lengkap mengenai kinerja dan keandalan controller dalam berbagai kondisi beban dan skenario kesalahan.

BAB VI HASIL DAN PEMBAHASAN

6.1 Hasil Penelitian

Penelitian ini bertujuan untuk mengembangkan sistem otomatisasi verifikasi kode Flutter pada server menggunakan JUnit dan Mockito guna meningkatkan efisiensi dalam pembelajaran aplikasi mobile berbasis Flutter. Berikut adalah hasil yang diperoleh dari penelitian ini:

1. Peningkatan Efisiensi Pembelajaran: Sistem otomatisasi verifikasi kode yang dikembangkan berhasil meningkatkan efisiensi proses pembelajaran. Dosen tidak perlu lagi memverifikasi kode secara manual, yang dapat memakan waktu lama. Sebaliknya, sistem otomatis dapat melakukan verifikasi dalam waktu singkat dan memberikan umpan balik yang cepat kepada mahasiswa.
2. Akurasi dan Ketepatan Verifikasi: Hasil pengujian menunjukkan bahwa sistem otomatisasi verifikasi kode memiliki akurasi yang tinggi dalam mendeteksi kesalahan dan memastikan ketepatan kode. Sistem ini mampu menjalankan berbagai jenis pengujian, termasuk unit testing dan widget testing, yang memastikan bahwa kode yang dihasilkan oleh mahasiswa memenuhi standar kualitas yang ditetapkan.
3. Penerimaan Pengguna: Evaluasi dari pengguna, baik dosen maupun mahasiswa, menunjukkan tingkat kepuasan yang tinggi terhadap sistem ini. Dosen merasa terbantu dengan adanya sistem ini, sementara mahasiswa merasa mendapatkan umpan balik yang cepat dan berguna untuk memperbaiki kode mereka.
4. Komparasi dengan Sistem Manual: Dibandingkan dengan metode verifikasi manual yang sebelumnya digunakan, sistem otomatis ini menunjukkan peningkatan yang signifikan dalam hal kecepatan dan konsistensi. Verifikasi manual sering kali terhambat oleh subjektivitas dan ketidakpastian, sementara sistem otomatis menyediakan hasil yang objektif dan konsisten.

6.2 Pembahasan

Dalam pembahasan ini, akan diuraikan mengenai perbandingan hasil penelitian dengan penelitian sebelumnya, kelebihan dan kekurangan solusi yang dikembangkan, serta analisis terhadap hasil pengujian oleh pengguna.

1. Perbandingan dengan Penelitian Sebelumnya**:**
    - Sistem APLAS (Android Programming Learning Assistant System): Penelitian sebelumnya oleh Syaifudin et al. (2021) mengembangkan sistem APLAS untuk verifikasi kode Android. Sistem tersebut menggunakan metode test-driven development (TDD) untuk verifikasi. Dibandingkan dengan APLAS, sistem verifikasi kode Flutter yang dikembangkan dalam penelitian ini memiliki keunggulan dalam hal integrasi dengan framework Flutter dan kemudahan penggunaan bagi pengembang aplikasi mobile.
    - Web Application Implementation for Android Programming Learning: Penelitian ini mengimplementasikan sistem berbasis web untuk pembelajaran pemrograman Android Java dan Kotlin. Hasil evaluasi menunjukkan bahwa sistem tersebut efektif dalam memberikan materi dan tugas, namun masih memerlukan validasi manual untuk beberapa aspek. Sistem verifikasi otomatis pada penelitian ini mengatasi kelemahan tersebut dengan menyediakan verifikasi kode secara menyeluruh dan otomatis.
2. Kelebihan Solusi yang Dikembangkan:
    - Efisiensi Waktu: Dengan otomatisasi verifikasi, waktu yang dibutuhkan untuk memeriksa kode berkurang secara signifikan. Hal ini memungkinkan dosen untuk fokus pada aspek pembelajaran lainnya.
    - Akurasi yang Tinggi: Sistem ini mampu mendeteksi kesalahan dengan akurasi tinggi, mengurangi kemungkinan kesalahan yang terlewatkan dalam verifikasi manual.
    - Umpan Balik Cepat: Mahasiswa mendapatkan umpan balik cepat, yang sangat penting dalam proses pembelajaran, memungkinkan mereka untuk segera memperbaiki dan meningkatkan kualitas kode mereka.
3. Kekurangan Solusi yang Dikembangkan:

- Keterbatasan dalam Pengujian Kompleks: Meskipun sistem ini efektif untuk pengujian dasar seperti unit test dan widget test, pengujian yang lebih kompleks seperti integrasi dengan API eksternal masih memerlukan validasi manual.
- Ketergantungan pada Infrastruktur: Implementasi sistem otomatisasi ini memerlukan infrastruktur server yang andal dan cepat. Gangguan pada server dapat menghambat proses verifikasi.

4. Analisis Hasil Pengujian oleh Pengguna:
   - Kepuasan Pengguna: Hasil survei menunjukkan bahwa mayoritas pengguna merasa puas dengan sistem ini. Dosen merasa terbantu dengan otomatisasi verifikasi, sementara mahasiswa merasa terbantu dengan umpan balik yang cepat dan jelas.
   - Peningkatan Keterampilan Mahasiswa: Mahasiswa yang menggunakan sistem ini menunjukkan peningkatan yang signifikan dalam keterampilan pemrograman mereka. Mereka lebih memahami kesalahan yang mereka buat dan dapat memperbaikinya dengan cepat.

6.3 Kesimpulan

Berdasarkan hasil dan pembahasan di atas, dapat disimpulkan bahwa sistem otomatisasi verifikasi kode Flutter yang dikembangkan berhasil meningkatkan efisiensi dan akurasi dalam proses pembelajaran aplikasi mobile berbasis Flutter. Meskipun terdapat beberapa kekurangan, kelebihan yang ditawarkan oleh sistem ini jauh lebih signifikan, memberikan manfaat besar bagi dosen dan mahasiswa dalam proses pembelajaran.

Bab VII Kesimpulan dan Saran

7.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan mengenai otomatisasi verifikasi kode Flutter pada server untuk pembelajaran aplikasi mobile, beberapa kesimpulan dapat diambil sesuai dengan tujuan penelitian yang telah dijabarkan pada bab pendahuluan.

Sistem otomatisasi verifikasi kode yang dikembangkan berhasil meningkatkan efisiensi proses pembelajaran. Dengan adanya verifikasi otomatis, dosen tidak lagi perlu memeriksa kode secara manual, yang sebelumnya memakan waktu dan tenaga. Sistem ini memungkinkan proses verifikasi berjalan lebih cepat dan memberikan umpan balik segera kepada mahasiswa.

Selain itu, sistem ini menunjukkan tingkat akurasi yang tinggi dalam mendeteksi kesalahan dan memastikan ketepatan kode. Penggunaan unit testing dan widget testing memastikan bahwa kode yang dihasilkan oleh mahasiswa memenuhi standar kualitas yang telah ditetapkan.

Dibandingkan dengan metode verifikasi manual, sistem otomatisasi ini menunjukkan peningkatan yang signifikan dalam hal kecepatan dan konsistensi. Verifikasi manual yang sering kali terhambat oleh subjektivitas dan ketidakpastian dapat diatasi dengan hasil yang objektif dan konsisten dari sistem otomatis.

7.2 Saran

Meskipun sistem otomatisasi verifikasi kode Flutter yang dikembangkan telah menunjukkan hasil yang positif, terdapat beberapa area yang masih dapat diperbaiki dan dikembangkan lebih lanjut untuk meningkatkan efektivitas dan efisiensi sistem:

1. Pengujian Kompleks: Saat ini, sistem ini efektif untuk pengujian dasar seperti unit test dan widget test. Namun, untuk pengujian yang lebih kompleks, seperti integrasi dengan API eksternal, masih memerlukan validasi manual. Disarankan untuk mengembangkan fitur tambahan yang dapat mengotomatisasi pengujian kompleks tersebut.
2. Peningkatan Infrastruktur: Implementasi sistem otomatisasi ini sangat bergantung pada infrastruktur server yang andal dan cepat. Disarankan untuk meningkatkan kapasitas dan keandalan server untuk menghindari gangguan yang dapat menghambat proses verifikasi.

3. Ekspansi Fitur Pembelajaran: Selain otomatisasi verifikasi, sistem ini dapat dikembangkan lebih lanjut dengan menambahkan fitur-fitur pembelajaran lainnya, seperti analisis kesalahan yang lebih mendalam dan rekomendasi perbaikan, sehingga dapat memberikan nilai tambah bagi mahasiswa dalam proses pembelajaran.
4. Penggunaan Teknologi Terbaru: Memanfaatkan teknologi terbaru dalam pengembangan sistem ini, seperti machine learning untuk prediksi kesalahan umum atau otomatisasi lebih lanjut dalam penilaian kode, dapat meningkatkan performa dan akurasi sistem.
5. User Experience (UX) Improvement: Meningkatkan antarmuka pengguna agar lebih intuitif dan mudah digunakan, sehingga baik dosen maupun mahasiswa dapat menggunakan sistem ini dengan lebih efektif dan efisien.

Dengan memperhatikan saran-saran tersebut, diharapkan sistem otomatisasi verifikasi kode Flutter pada server untuk pembelajaran aplikasi mobile ini dapat terus berkembang dan memberikan manfaat yang lebih besar bagi seluruh pengguna.

Daftar Pustaka

Aziz, D. A., Andreswari, R., & Gumilang, S. F. (2020). PERANCANGAN BISNIS DAN ARSITEKTUR APLIKASI PADA APLIKASI MOBILE MANAWA INVESTASI HEWAN TERNAK. *e-Proceeding of Engineering : Vol.7, No.2 Agustus 2020*, 7112.

Cheon, Y., & Chavez, C. (2021). Converting Android Native Apps to Flutter Cross-Platform Apps. *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*.

Flutter. (2023). Retrieved from Flutter Documentation: https://flutter.dev/

Hadjrianto, A. S. (2022). Pengembangan Topik Basic Application Untuk Pembelajaran Aplikasi Mobile Berbasis Flutter Pada Platform Intelligent Computer-Assisted Programming Learning Platform. *Politeknik Negeri Malang*.

Hanif, I. F., & Sinambela, G. M. (2020). PEMBUATAN APLIKASI E-TATIB BERBASIS ANDROID MENGGUNAKAN BAHASA PEMROGRAMAN DART. *Vol. 3 No. 2 (2020): Vol 3 No 2 (2020): Jurnal Teknologi dan Terapan Bisnis*, 23-29.

Khasanah, D. R., Pramudibyanto, H., & Widuroyekti, B. (2020). Pendidikan Dalam Masa Pandemi Covid-19. *Jurnal Sinestesia, Vol. 10, No. 1, April2020*, 41-42.

Kozak, M. (2023). Analysis of the Spring Boot and Spring Cloud in developing Java cloud applications. *Journal of Computer Sciences Institute*.

Kreinovich, V., Ceberio, M., & Kosheleva, O. (2020). White- and Black-Box Computing and Measurements Under Limited Resources: Cloud, High Performance, and Quantum Computing, and Two Case Studies - Robotic Boat and Hierarchical Covid Testing. *International Workshop on Information-Communication Technologies & Embedded Systems*.

Stošović, S., Stefanović, D., Bogdanović, M., & Vukotić, N. (2022). THE USE OF THE FLUTTER FRAMEWORK IN THE DEVELOPMENT PROCESS OF HYBRID MOBILE APPLICATIONS. *KNOWLEDGE - International Journal*.

Stray, V., Hoda, R., Maria, P., & Kruchten, P. (2020). Agile Processes in Software Engineering and Extreme Programming. *International Conference on Agile Software Development*.

Syaifudin, Y. W., Funabiki, N., & Kuribayashi, M. (2021). Implementation and Performance Evaluation of Unit Testing for Student's Answer Validation in Android Programming Learning Assistant System. *THE INSTITUTE OF ELECTRONICS, INFORMATION AND COMMUNICATION ENGINEERS*, 2-5.

Syaifudin, Y. W., Funabiki, N., Kuribayashi, M., Mentari, M., Saputra, P. Y., Yunhasnawa, Y., & Ulfa, F. (2020). Web application implementation of Android programming learning assistance system and its evaluations. *IOP Conference Series: Materials Science and Engineering, Volume 1073, The 2nd Annual Technology Applied Science and Engineering Conference (ATASEC 2020)*.

Syaifudin, Y. W., Hatjrianto, A. S., Funabiki, N., Liliana, D. Y., Kaswar, A. B., & Nurhasan, U. (2022). An Implementation of Automatic Dart Code Verification for Mobile Application Programming Learning Assistance System Using Flutter. *International Conference on Electrical and Information Technology(IEIT), Malang, Indonesia*, 322-326.