IMPLEMENTASI PENGGUNAAN API DENGAN *CLOUD STORAGE* UNTUK APLIKASI REKOMENDASI PAKAIAN

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T)

Disusun Oleh:

Rhein Kharnafis Dhirgham

3332200093

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2024

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis skripsi berikut

Judul	:	Implementasi API dengan *Cloud Storage* untuk Aplikasi Rekomendasi Pakaian
Nama Mahasiswa	:	Rhein Kharnafis Dhirgham
NPM	:	3332200093
Fakultas/Jurusan	:	Teknik/Teknik Elektro

menyatakan dengan sesungguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggungjawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini.

Cilegon, 20 Oktober 2024

Rhein Kharnafis Dhirgham
NPM. 3332200093

LEMBAR PENGESAHAN

Dengan ini ditetapkan bahwa skripsi berikut:

Judul	:	Implementasi API dengan *Cloud Storage* untuk Aplikasi Rekomendasi Pakaian
Nama Mahasiswa	:	Rhein Kharnafis Dhirgham
NPM	:	3332200093
Fakultas/Jurusan	:	Teknik/Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 28 November 2024 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa Cilegon dan dinyatakan LULUS/TIDAK LULUS

Dewan Penguji

		Tanda Tangan
Pembimbing I	: Dr. Irma Saraswati, S.Si., MT.	.....
Penguji I	: H. Alief Maulana, S.T., M.T.	.....
Penguji II	: Fadil Muhammad, S.T., M.T.	.....

Mengetahui,
Ketua Jurusan

Dr. Eng. Rocky Alfanz, S.T., M.Sc.
NIP.198103228201012100 1

PRAKATA

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat rahmat dan karunia-Nya penulis dapat menyelesaikan laporan akhir skripsi. Saya menyadari bahwa secara tidak langsung telah banyak pihak yang membantu saya dalam penyusunan laporan skripsi ini. Tanpa keterlibatan mereka, tentunya sulit bagi saya untuk menyelesaikan skripsi ini dengan baik. Oleh karena itu, pada kesempatan kali ini saya ingin mengucapkan banyak rasa terima kasih banyak kepada:

1. Dr. Eng. Rocky Alfanz, S.T., M.Sc., selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Sultan Ageng Tirtayasa.
2. M. Hartono, S.T., M.T. selaku Dosen Pembimbing Akademik.
3. Dr. Irma Saraswati, MT, selaku Dosen Pembimbing MBKM yang telah membimbing saya selama kegiatan berlangsung, dan penelitian skripsi ini.
4. Kedua orang tua tercinta serta seluruh keluarga yang telah memberikan segalanya, nasihat, semangat, kasih sayang, doa dalam setiap kegiatan yang saya lakukan.
5. Yuda Adi Pratama, selaku mentor yang telah rela membimbing saya selama program MBKM berlangsung.

Akhir kata, saya berharap semoga Tuhan yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Selain itu, besar harapan saya agar laporan Skripsi ini dapat bermanfaat untuk banyak orang.

Cilegon, 20 Oktober 2024

Rhein Kharnafis Dhirgham

ABSTRAK

Rhein Kharnafis Dhirgham

Teknik Elektro

Implementasi Penggunaan API dengan *Cloud Storage* untuk Aplikasi Rekomendasi Pakaian

Banyaknya penggunaan media sosial membuat muncul tren dalam waktu singkat dan mendorong seseorang untuk selalu mengikuti tren, salah satunya tren fashion. Pemilihan pakaian yang tepat sesuai tren meningkatkan rasa kepercayaan diri seseorang. Oleh karena itu, perlu adanya pemahaman terkait referensi pakaian, dengan adanya aplikasi rekomendasi berpakaian pada penelitian ini dapat menjadi inovasi yang relevan dan berguna dalam bidang *fashion* untuk memberikan rekomendasi sesuai referensi pengguna. Peningkatan kinerja aplikasi rekomendasi yang optimal menggunakan layanan *cloud computing* GCP. Layanan lain penting pada aplikasi adalah API yang digunakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Pada penelitian ini sudah dilakukan perancangan dan implementasi API sesuai kebutuhan aplikasi dengan membangun fungsionalitas melalui integrasi penggunaan API dengan sistem *cloud storage,* menganalisis performa waktu, keberhasilan pengoprasian, dan pengujian beban. Hasil pengujian fungsionalitas dilakukan dengan metode *black box* yang mendapatkan hasil yang baik. Penerapan integrasi antara sistem *cloud* dengan layanan API dibuat meringankan beban penyimpanan karena dilakukan pada penyimpanan *cloud*. Hasil pengujian performa dengan Jmeter mendapatkan hasil waktu respon yang baik dengan rata-rata dibawah 1 detik, namun terdapat API yang mendapatkan perolehan waktu lambat yang disebabkan oleh proses *hashing* yang lambat untuk keamanan dan penanganan *file*.

Kata Kunci: Pakaian, Sistem Rekomendasi, API, Cloud Storage, GCP

ABSTRACT

Rhein Kharnafis Dhirgham

Electrical Engineering

Implementation of API Usage with Cloud Storage for Clothing Recommendation Application

The widespread use of social media makes trends appear quickly and encourages people always to follow trends, one of which is fashion trends. Choosing the right clothes according to trends increases one's self-confidence. Therefore, it is necessary to have an understanding of clothing references, with the dressing recommendation application in this research can be a relevant and useful innovation in the fashion field to provide recommendations according to user references. Improving the optimal performance of recommendation applications using GCP cloud computing services. Another important service in the application is the API which is used as a link between an application and other applications. In this research, API design and implementation have been carried out according to application needs by building functionality through the integration of API usage with cloud storage systems, analyzing time performance, operation success, and load testing. The results of functionality testing are carried out using the black box method which gets good results. The application of integration between cloud systems and API services is made to lighten the storage load because it is carried out on cloud storage. The results of performance testing with Jmeter get good response time results with an average of under 1 second, but some APIs get slow time gains caused by slow hashing processes for security and file handling.

Keywords: Fashion, Recommendation System, API, Cloud Storage, GCP

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Penggunaan media sosial pada saat ini sangat mudah diakses yang dapat memungkinkan masyarakat dengan mudah terhubung dan berinteraksi menggunakan platform media sosial. Penggunaan *platform* media sosial diperkirakan akan terus meningkat. Jumlah pengguna media sosial aktif untuk platform media sosial di Indonesia sebanyak 167 juta orang, jumlah pengguna tersebut setara dengan 60,4% dari populasi masyarakat Indonesia [1]. Berdasarkan banyaknya jumlah masyarakat yang menggunakan platform media sosial serta perkembangan alur globalisasi di zaman sekarang membuat munculnya tren dalam waktu singkat dan mendorong seseorang untuk selalu mengikuti tren. Hal tersebut membuat resah masyarakat karena banyak orang menggangap rendah orang yang tidak mengikuti trend [2].

Fenomena tren *fashion* adalah salah satu bidang yang paling mudah dikonsumsi oleh masyarakat dan yang cepat berganti karena masyarakat dan orang yang aktif pada industri *fashion* ingin tampil terdepan dan mengadopsi tren paling cepat [3]. Pemilihan pakaian yang tepat dalam berbusana sesuai tren terkini dapat membuat seseorang lebih baik pada suasana hatinya dan juga membuat orang lebih baik dalam bersosialisasi karena dapat meningkatkan rasa percaya diri [4]. Pernyataan pemilihan pakaian dapat meningkatkan rasa percaya diri seseorang ini didukung oleh hasil survei yang menunjukan 91,9 % sepakat bahwa pemilihan pakaian dalam berbusana dapat meningkatkan rasa kepercayaan diri seseorang [5].

Peran *fashion* pada saat ini bukan hanya sekedar sebuah kebutuhan, melainkan telah menjadi gaya hidup seperti identitas pribadi ataupun keinginan memenuhi citarasa bagi orang, fashion itu sendiri juga merupakan fenomena yang terus berkembang karena pengaruh seperti norma sosial, seni, ikon gaya dan ikon kota tempat tinggal. Fashion yang digunakan sering berubah, dengan cara yang cepat, perubahan mendadak dan ketidakpastian yang meningkat [6], [7]. Oleh karena itu, untuk mengatasi dalam mengurangi rasa tidak percaya diri dan rasa

cemas pada seseorang terkait pakaian yang digunakannya, perlu adanya pemahaman lebih terkait referensi dalam pemilihan pakaian, dengan adanya aplikasi rekomendasi pakaian yang dapat menjadi inovasi yang relevan dan berguna dalam bidang *fashion*. Sistem rekomendasi pada umumnya ditujukan untuk individu yang kurang pengalaman atau kompetensi dalam mengevaluasi item pada kasus tertentu [8]. Aplikasi dengan sistem rekomendasi merupakan sebuah sistem yang dapat membantu memberikan keputusan dan memberikan rekomendasi sesuai dengan *preference* pengguna. Salah satu cara untuk mendapatkan hasil baik dari sistem rekomendasi menggunakan proses *machine learning*, karena dengan *machine learning* dapat memungkinkan aplikasi untuk memahami preferensi pengguna berdasarkan data untuk membuat model statistik yang digunakan pada sistem [9].

Aplikasi dalam bentuk android digunakan untuk mengemas sistem rekomendasi yang digunakan dalam penelitian ini, digunakannya aplikasi android agar supaya memberikan hasil yang baik dan optimal serta masyarakat dapat dengan mudah mengaksesnya. Aplikasi android digunakan karena saat ini banyak orang yang dalam aktivitas kesehariannya menggunakan perangkat seluler secara pribadi maupun professional yang membuat kehidupan sehari-hari lebih mudah [10][11]. Pengguna *smartphone* diperkirakan pada tahun 2024 hingga mencapai 6,935 miliar pengguna [12]. Sebanyak 71,43% diantaranya menggunakan sistem operasi android untuk pengguna diseluruh dunia [13].

Solusi dari masalah yang ada dalam penelitian ini perlu dibantu dengan kinerja aplikasi yang optimal. Mobilitas yang tinggi dengan perangkat yang terbatas akan membuat besarnya kebutuhan data. Oleh karena itu, salah satu upaya dalam meningkatkan kinerja aplikasi dengan cara menggunakan layanan *cloud computing*. Layanan *cloud computing* merupakan model layanan berbasisi internet yang memungkinkan akses dan penggunanan sumber daya dapat secara fleksibel dilakukan [14]. Alasan penggunaan layanan *cloud computing* dapat mengoptimalkan aplikasi yang dibutuhkan, karena dalam *cloud computing* sumber daya komputasi dan penyimpanan data disediakan layanan melalui internet. Model ini memberikan banyak maanfaat seperti skalabilitas, aksesibilitas, dan efisiensi yang tingi dalam pengelolaan dan pemanfaatan sumber daya komputasi

[15]. Layanan yang dipilih untuk mengoptimalkan kinerja aplikasi adalah GCP (Goole *Cloud Platform*) karena cukup terkenal dan merupakan kumpulan layanan *serverless computing*. Google *cloud platform* memiliki manfaat yang baik seperti kecepatan dalam fasilitas penerapan dan dioperasikan dalam waktu singkat [16].

Ekosistem pada aplikasi terdapat peran yang penting dan sangat dibutuhkan yaitu API (*Application Programming Interface*) merupakan kunci penting dalam integrasi dan interaksi antara berbagai aplikasi. API merupakan pengelollan yang terdiri dari *interface*, fungsi, kelas, dan struktur yang digunakan dalam membuat sebuah perangkat lunak. API dapat digambarkan sebagai antara aplikasi dengan aplikasi yang lain, API dapat memungkinkan programmer dalam memanfaatkan fungsionalitas sistem [6], [7].

API sebagai *Backend Development* yang bertanggung jawab atas sisi *server* dan basis data. Pengembangan aplikasi penelitian ini menggunakan teknologi *cloud computing* seperti *app engine* untuk membangun aplikasi dan *cloud storage* untuk penyimpanan data yang bersifat virtual dapat diimplementasikan pada *mobile* berbasis Android. Oleh karena ini *back-end* dilakukan perancangan REST API (*Representatif State Transition Application Programming Interface*), RESTAPI adalah standar arsitektur komunikasi yang digunakan untuk mengembangkan layanan berbasis aplikasi, dan API adalah koneksi yang dapat membuat aplikasi mampu berinteraksi dan bertukar data [17]. Oleh kerena itu pada aplikasi rekomendasi pakaian dari sisi *backend* dikembangkan dengan menggunakan REST API yang memfasilitasi transfer dan permintaan data malalui HTTP, yang dikenal dengan kemudahannya dalam paradigma *cloud* [18]. Penggunaan google *cloud storage* untuk penyimpanan yang menyediakan penyimpanan objek yang sangat tahan lama dan memungkinkan pengambilan data global [19].

1.2. Rumusan Masalah

Berdasarkan penjelasan yang telah dipaparkan pada bagian latar belakang penelitian ini, maka dapat diketahui bahwa rumusan masalah dalam penelitian kali ini ialah sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan API yang akan digunakan oleh aplikasi rekomendasi pakaian?
2. Bagaimana fungsionalitas aplikasi melalui integrasi dengan memanfaatkan penggunaan API dan *cloud storage*?
3. Bagaimana performa keberhasilan pengoprasian dan respon sistem dalam *load condition* yang dihasilkan atas penerapan *cloud storage* dan implementasi API dalam penelitian ini?

1.3. Tujuan Penelitian

Adapun tujuan dilaksanakannya penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan API yang akan digunakan oleh aplikasi.
2. Membangun fungsionalitas aplikasi melalui integrasi dengan memanfaatkan penggunaan API dan sistem *cloud storage.*
4. Menganalisis performa keberhasilan pengoprasian dan respon sistem dalam *load condition* yang dihasilkan atas penerapan *cloud storage* dan implementasi API dalam penelitian ini.

1.4. Manfaat Penelitian

Beberapa manfaat yang dapat dirasakan dari hasil penelitian ini diantaranya yaitu:

1. Hasil penelitian ini dapat dijadikan sebagai pertimbangan dalam penggunaan sistem *cloud storage* untuk mengoptimalkan infrastruktur aplikasi dan efisiensi penggunaan sumber daya komputasi.
2. Hasil penelitian ini dapat memberikan kerangka kerja yang fleksibel dalam membangun dan meningkatkan fungsionalitas aplikasi melalui integerasi dengan memanfaatkan penggunaan API.

3. Hasil penelitian ini dapat memberikan gambaran dan pemahaman mengenai besarnya kontribusi dari sistem *cloud* dan API yang dibuat terhadap kinerja aplikasi rekomendasi pemilihan busana.

1.5. Batasan Masalah

Agar penelitian dapat memperoleh wawasan secara mendalam, maka terdapat beberapa batasan yang perlu diperhatikan selama proses penelitian berlangsung. Adapun beberapa batasan yang terdapat dalam penelitian kali ini adalah sebagai berikut:

1. Rancangan arsitektur *cloud* yang digunakan dalam penelitian ini hanya diimplementasikan pada satu layanan penyedia *cloud storage,* yakni Google *Cloud Platform* (GCP).
2. Fokus pembahasan penelitian ini hanya terbatas pada pengembangan *backend* dan sisi implementasi API dengan sistem *cloud storage* saja.
3. Metode yang digunakan dalam pengembangan API hanya menggunakan metode RESTful API.
4. Layanan komputasi serta manajemen basis data yang digunakan dalam penelitian ini hanya berfokus pada layanan Google *Cloud Storage*.

1.6. Sistematika Penulisan

Sistematika laporan dapat terlihat baik apabila penyusunan pada setiap bab dilakukan secara terstruktur serta mudah untuk dipahami. Penulisan penelitian skripsi ini, terbagi menjadi 5 bab yang saling berkaitan antara satu dengan lainnya. Adapun isi dari setiap bab dalam laporan ini adalah sebagai berikut:

BAB I: PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang implementasi API dengan sistem cloud pada aplikasi rekomendasi *outfit*, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika penulisan.

BAB II: TINJAUAN PUSTAKA

Bab ini berisikan dasar teori yang berkaitan dengan pembahasan pada laporan skripsi hasil program studi independen mengenai *cloud computing*, Google *Cloud*

Platform, APP engine, Cloud Storage, Backend Development, API, Rest API, Node JS.

BAB III: METODOLOGI PENELITIAN

Bab ini berisikan diagram alir penelitian, penjelasan mengenai metode penelitian yang digunakan, alur penelitian, perancangan sistem *backend*, implementasi, pengujian API. Komponen penelitian yang dikenakan, studi literatur, perancangan, dan implementasi.

BAB IV ANALISIS

Bab ini berisi mengenai pembahasan hasil penelitian yang telah dilaksanakan dan membahas analisis data - data yang diperoleh sesuai dengan batasan dan parameter yang digunakan.

BAB V PENUTUP

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan serta saran untuk penelitian atau kegiatan mendatang.

BAB II
TINJAUAN PUSTAKA

2.1 *Cloud Computing*

Sistem teknologi *cloud computing* atau bisa disebut juga dengan komputasi awan saat ini sering digunakan dalam melakukan pengembangan aplikasi. *cloud computing* dalam era teknologi yang pesat dan sudah banyak dikenal dan dilakukan perkembangan yang sangat cepat. Karena secara *virtual* menjadikan Perusahaan besar di berbagai bidang kini sudah mulai fokus dan tidak asing lagi dalam penggunaan sistem *cloud computing*. Penggunaan *cloud computing*, pengguna diperlukan untuk melakukan penyewaan berberapa layanan penyedia komponen kerja dan server penyimpanan data hingga data *center* dalam melakukan pengembangan aplikasi.

Cloud computing memberikan banyak layanan dalam bentuk bisnis melalui Internet. penggunaan *platform cloud* dalam melakukan proses *cloud computing* dapat membuat tidak perlu melakukan kegiatan seperti tanggung jawab untuk melakukan pemeliharaan layanan, hanya perlu fokus pada masalah teknis yang lebih penting. Layanan *cloud computing* dapat membuat infrastruktur yang lebih efisien dan mampu membuat layanan yang dapat di atur sesuai kebutuhan pengguna. *Cloud computing* menurut Google merupakan sebuah *central* data yang ditempatkan pada luar komputer atau secara *virtual* dari aplikasi pengguna. Google memberikan fitur kontribusinya dalam sistem *cloud computing* yang memiliki fungsi utama agar dapat memanfaatkan sumber daya komputer dalam jumlah yang sangat besar dan terdiri dari banyaknya komputer tersebar di seluruh dunia, dapat digunakan untuk mempercepat operasi yang dilakukan oleh aplikasi modern [20].

Terdapat beberapa sifat yang dimiliki dalam sistem *cloud computing*, sifat tersebut diantaranya adalah sesuai permintaan dari penggunan layanan, elastis yang berarti dapat diatur sesuai kebutuhan, terukur yang membuat *developer* dapat mengetahui rancangan berkelanjutannya dan sepenuhnya dikelola oleh penyedia atau *provider* yang memudahkan dalam menggunakan layanan *cloud*. Layanan yang bersifat sesuai permintaan menjelaskan bahwa penggunaan layanan

cloud dapat bebas memilih dan memanfaatkan layanan-layanan yang telah diberikan oleh penyedia layanan *cloud*. Karakteristik elastis dan terukur menjelaskan mengenai bagaimana fleksibilitas dan kemampuan ruang yang dapat diminta pengguna ketika menggunakan satuan kecepatan data tertentu, sementara karakter layanan terkahir yaitu untuk dapat dikendalikan sepenuhnya oleh penyedia layanan, yang dapat memfasilitasi kemampuan pengguna untuk meringankan dalam memproses dan berbagi data dan dokumen dengan perangkat lain [21].

Cloud computing terdapat tiga jenis dasar penyusunnya layanan dari *cloud computing*, yaitu diantaranya *Software as a Service* (SaaS), *Platform as a Service* (PaaS), dan *Infrastructure as a Service* (IaaS). *Software as a Service* (SaaS) merupakan beberapa layanan *cloud computing* yang pertama kali dikenal oleh pengguna. Jenis SaaS ini merupakan inovasi selanjutnya dari konsep dari ASP (*Application Service Provider*). SaaS dapat mempermudah penggunanya dalam berlangganan, oleh karena itu pengguna SaaS tanpa harus mengeluarkan dana dalam pengembangan internal atau membeli lisensi. *Platform as a Service* (PaaS) adalah layanan penyediaan modul-modul yang secara langsung siap pakai dalam membangun sebuah aplikasi yang hanya mampu beroperasi di atas basis aplikasi. Layanan PaaS pada sistem *cloud* memberikan tempat untuk membuat dan mendistribusikan aplikasi tanpa harus mengetahui banyaknya *processor* atau memori yang diperlukan aplikasi. *Infrastructure as a Service* (IaaS) memiliki tingkat di bawah PaaS. IaaS adalah layanan yang berikan fitur sewa sumber daya teknologi informasi dasar seperti sistem operasi, daya komputasi, media penyimpanan, memori, kapasitas jaringan, dan layanan lainnya yang dapat digunakan oleh pengguna dalam menjalankan aplikasi [22].

2.2 *GCP (Google Cloud Platform)*

GCP merupakan layanan *cloud computing* yang di sediakan oleh Google, dan itu adalah kumpulan administrasi komputasi awan. GCP memberikan landasan tahap manajemen dan kondisi pendaftaran tanpa *server*. Google memperkenalkan *App Engine* pada bulan April 2008, yang berfungsi sebagai tahapan untuk membuat dan memfasilitasi aplikasi dalam *server* yang diawasi

Google. Ini terutama berhubungan dengan administrasi komputasi terdistribusi. GCP merupakan unit Google yang menggabungkan GCP kerangka *cloud* terbuka, hanya menggunakan *G Suite*, industri adaptasi Android dan Chrome OS, plus aplikasi antarmuka pemrograman (API) yang ditujukan untuk *Artificial Inteligent* (AI) dan melakukan administrasi pemetaan [23].

Layanan produk yang GCP mampu digunakan untuk melakukan perancangan dalam membangun infrastruktur server ialah *Compute Engine, Cloud SQL, Cloud Storage, dan Container Registry*. Produk Google *cloud* memiliki fitur dan teknik yang dapat diterapkan untuk membangun infrastruktur *server* dengan ketersediaan tinggi seperti fitur *autohealing, multiple zones, load balancing, autoscaling, automatic updating, dan failover*. Infrastruktur *server* mampu dibuat dengan menggunakan *service product* yang ada pada teknologi GCP, sehingga mampu menghasilkan sistem yang andal dan memiliki ketersediaan tinggi [24].

2.3 *App Engine*

Google *App Engine* adalah salah satu jenis layanan yang dijelaskan pada pembahasan google *cloud platform* yaitu *Platform as a Service* (PaaS). Google *App Engine* dapat membuat pengguna mampu menjalankan aplikasi web pada infrastruktur Google. Menggunakan *App Engine* tidak perlu lagi sebuah *server*, aplikasi tersebut sudah dapat bisa digunakan oleh pengguna sebagai bagian dari *server* [25]. Sebagian besar dari program modern saat ini dapat dengan mudah dilakukan dengan ditulis oleh pemprogram yang ahli dalam bahasa pemprograman seperti java dan python. Komputasi tanpa menggunakan *server* dapat dijelaskan sebagai model komputasi awan yang menyediakan berbagai layanan *backend* dengan sesuai permintaan. Penyedia layanan *cloud* saat ini cenderung memberikan penyediaan, mendistribusikan, dan menagih pengguna dari sumber daya dan penyimpanan yang mereka gunakan. Oleh karena itu, pengguna tidak perlu melakukan pembayaran dengan tarif yang tetap untuk banyaknya *server* dan *bandwidth* yang digunakan atau dipesan, hanya mencakup *server* tempat *database* dan *file* dari aplikasi pengguna. Meskipun proses ini disebut *serverless computing*, *server* berbentuk fisik masih menjadi bagian dari

persamaan, namun dengan perangkat lunak seperti ini, pengguna tidak perlu khawatir terkait pemeliharaan dan penyediaan *server* [20].

Google App Engine mengelola sepenuhnya untuk mengembangkan dan *hosting* aplikasi *web* dalam skala besar. Pengguna dapat menentukan penggunaan dan rancangan bahasa, pustaka, dan kerangka kerja untuk melakukan pengembangkan aplikasi, kemudian *App Engine* dapat menangani penyediaan *server* dan pengaturan penggunaan *instance* aplikasi berdasarkan permintaan. *App engine* adalah PaaS untuk membangun aplikasi yang skalabel. *App engine* merupakan lingkungan standar dengan lingkungan terbatas dan dukungan untuk bahasa seperti Python, Go, node.js dan yang lainnya merupakan lingkungan yang memiliki keunggulan fleksibel di mana pengguna dapat memiliki lebih banyak kebebasan seperti menjalankan *runtime* menggunakan docker, waktu tunggu permintaan respons yang lebih lama, kemampuan yang dapat mengunduh dependensi atau perangkat lunak khusus, dan SSH ke dalam mesin *virtual* [20].

Membangun aplikasi *Serverless* dengan *backend* atau API dapat mendukung beberapa bahasa pengembangan tanpa perlu khawatir tentang dukungan infrastruktur. Saat pengguna memiliki aplikasi yang memerlukan komunikasi dengan beberapa layanan seperti aplikasi atau API, *App Engine* merupakan solusi yang sesuai untuk aplikasi *serverless* dengan *backend* dan API tanpa perlu khawatir dukungan infrastruktur. Hubungan antara layanan tersebut memungkinkan aplikasi diperlakukan sebagai entitas yang dikelola [20].

2.4 *Cloud SQL*

Basis data SQL dikenal memiliki sifat yang disingkat ACID (*Atomic, Consistent, Isolation dan Durability*). Sifat *Atomic* pada data SQL apabila ada sebuah transaksi yang memiliki dua atau lebih bagian data informasi, semua bagiannya harus disimpan atau semua komponennya tidak disimpan. Tidak ada sebagian saja komponen yang disimpan atau tidak disimpan. *Consistent* adalah data yang disimpan tidak boleh melanggar integritas basis data. Inkonsisten data dapat mengalami gangguan dan dibatalkan untuk memastikan basis data berada dalam kondisi sama seperti sebelum ada perubahan. *Isolation* adalah sebuah transaksi yang tidak dipengaruhi oleh transaksi lain saat sedang berjalan. Hal ini

dapat bertujuan untuk mencegah terjadinya pertukaran yang gagal antara data dan transaksi. *Durability* adalah apabila transaksi sudah disimpan di basis data secara permanen, untuk seterusnya transaksi tersebut ada di basis data meskipun terjadi kegagalan sistem [26].

Penyedia layanan *cloud* GCP mempunyai layanan yang disebut Google *Cloud* SQL yang merupakan *database* MySQL yang disediakan oleh Google *Cloud*. Layanan yang ditawarkan GCP ini memiliki semua kemampuan dan fungsi dari MySQL, dengan beberapa fitur tambahan yaitu dilakukan secara *cloud*. Google *Cloud* SQL dapat mudah digunakan, dan tidak memerlukan instalasi perangkat lunak yang sulit atau pemeliharaan, oleh karena itu sangat ideal pengembangan aplikasi kecil hingga menengah [25].

2.5 *Cloud Storage*

Cloud Storage atau penyimpanan awan merupakan suatu layanan penyimpanan *file* berbasis internet. Sistem penyimpanan *cloud* melakukan proses penyimpanan *file* yang dapat langsung dan diakses mana saja dan kapan saja selama penggunanya terhubung internet yang dapat mengakses langsung ke *Cloud Storage*. Konsep dari *Cloud Storage* seperti konsep *file server* yang dimiliki kantor perusahaan, namun yang membedakan adalah infrastrukturnya oleh media penyimpanan langsung dikelola oleh *provider Cloud* dan pemanfaatannya dapat dijadikan layanan penyimpanan *file* yang dapat diakses menggunakan internet. Sebelum awal kemunculan *Cloud Computing* seperti saat ini, layanan dari *Cloud Storage* lebih dikenal dengan istilah *virtual drive* yaitu hanya sebagai penyimpanan secara virtual, namun saat ini dengan adanya istilah *Cloud Computing* lebih dikenal dengan sebutan *Cloud Storage*. Pengguna *Cloud Storage* tidak memerlukan membawa atau merawat media penyimpanan seperti *hard drive* untuk *file* yang telah tersimpan di dalam penyimpanan *cloud* [27].

Salah satu layanan dari *cloud storage* yang baik untuk digunakan adalah Google *Cloud Storage*. Google *Cloud Storage* merupakan layanan *cloud* yang termasuk dari banyaknya layanan yang ditawarkan oleh penyedia layanan *cloud* yaitu GCP. Penyimpanan *cloud* ini dapat membantu pengembang skala bawah hingga atas seperti di perusahaan untuk melakukan menyimpan dan mengakses

data di Google *Cloud*. Fitur ini dapat membantu para *developer* agar dapat mengakses langsung Google *scalable storage* dan infrastuktur jaringan yang sama baiknya dengan metode autentifikasi dan mekanisme dengan data *sharing* yang kuat [25].

2.6 *Backend Development*

Backend merupakan bagian dari suatu proses yang terjadi dibalik sebuah sebuah *website* atau aplikasi. Bahasa pemprograman untuk pengembangan atau pembuatan dari *backend development* secara umum digunakan diantaranya adalah javascript, PHP, Ruby, Python, dan banyak lainnya. *Backend* merupakan sebuah mesin yang dapat melakukan pekerjaan di balik layar, segala sesuatu yang pengguna akhir atau interaksi langsung tidak lihat dalam prosesnya, namun dengan itu dapat memberikan pada apa yang terjadi. Pengembangan *backend* berfokus pada *database*, *scripting*, dan arsitektur sesuai dengan kebutuhan aplikasi yang dapat memudahkan terciptanya fitur penggunaan aplikasi. Kode yang ditulis oleh pengembang dalam membangun *backend* dari aplikasi akan membantu transfer informasi basis data terhadap *browser* [28].

Backend merupakan fungsi atau bagian yang bertanggung jawab untuk melakukan proses data, menjalankan logika bisnis, dan berkomunikasi dengan basis data. *Backend* berfungsi untuk penghubung antara pengguna dengan penyimpanan data menggunakan antarmuka pengguna atau API. Proses membuatan *backend* dalam menggunakan *serverless platform* seperti Google *Cloud Platform* memungkinkan untuk melakukan *run code* tanpa harus mengelola infrastruktur server.

2.7 *API*

API adalah antarmuka yang sering digunakan dalam mengakses suatu aplikasi atau layanan dari program yang dibuat. API dapat membuat pengembang untuk menggunakan fitur atau fungsi yang sudah ada dari aplikasi lain, sehingga pengembang tidak perlu melakukan membuat ulang dari awal. Penggunan dalam konteks *website*, API merupakan pemanggilan fungsi melalui *Hyper Text Transfer Protocol* (HTTP) yang akan menerima sebuah respon dalam format berbentuk

Extensible Markup Language (XML) atau *JavaScript Object Notation* (JSON) [29].

Tujuan penggunaan dari API dalam pengembangan aplikasi berfungsi agar dapat saling berbagi data antar aplikasi yang berbeda, selain dapat saling berbagi data antar aplikasi API memiliki agar mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah *function* yang terpisah sehingga pengembang tidak perlu lagi merancang fitur yang sama. API yang bekerja pada tingkat sistem operasi dapat membantu aplikasi untuk bisa berkomunikasi dengan lapisan dasar dan satu sama lainnya dapat mengikuti serangkaian protokol dengan spesifikasi yang telah disesuaikan [29].

2.8 *REST API*

REST API adalah seperangkat prinsip arsitektur yang dapat melakukan transfer data melalui antarmuka yang terstandarisasi seperti HTTP. REST dapat berfungsi seperti layaknya aplikasi *web*. Penggunaan REST API *Client* mampu mengirimkan sebuah permintaan kepada server melalui protokol HTTP yang kemudian *server* memberikan kembali respon kepada klien.

REST dibuat dan dikembangkan oleh Roy Fielding yang merupakan salah satu pendiri dari Apache HTTP *Server Project*. Dalam arsitektur REST *server* mampu menyediakan dan mengakses sumber daya, data REST klien mengakses dan menunjukan sumber daya tersebut untuk penggunaan yang lain. Setiap sumber daya diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML [29].

2.9 *NodeJS*

Node.js merupakan *platform* yang dibuat untuk sebuah *webserver*. Node.js ini diulis dalam bentuk bahasa pemprograman javascript yang berbasis pada *event driven*. Berbeda dengan bahasa javascript yang dijalankan di klien, Node.js ini dapat berjalan sebagai aplikasi *server*. Node.js mempunyai kinerja dari *memory* yang terbilang efisien dan lebih baik ketika bekerja dalam kondisi beban yang berat. Pengguna yang menggunakan platform tidak perlu khawatir dengan

terjadinya kondisi *deadlock* pada saat menggunkannya karena tidak ada *lock* dalam Node.js. Aplikasi Nodejs ini terdiri dari V8. Mesin dari javascrpit yang dikebangkan oleh google dan beberapa komponen modul bawaan yang terhubung dengan Nodejs.org [30].

Pada platform Node.js terdapat operasi atas yang bersifat *synchronous* dan *asynchronous,* namun Node.js ini memiliki ke unggulannya yang terdapat pada *asynchronousnya* atau bisa juga disebut dengan *non-blocking* I/O yaitu dapat melakukan permintaan bersamaan dalam proses tunggal. Node.js menggunakan pendekatan *event-driven* berbasis *infinite event loop* dalam satu *thread* untuk mengurangi jumlah memori yang digunakan.

2.10 *Prisma ORM*

Object Relational Mapping (ORM) adalah teknik yang bertujuan untuk menghubungkan perbedaan antara paradigma pengembangan aplikasi berorientasi objek dengan pengembangan aplikasi berorientasi dengan paradigma basis data. Prisma merupakan *Object-Relational Mapper* (ORM) yang dirancang dan dibuat untuk dapat mengoptimalkan koneksi antar basis data dengan basis data lainnya [31]. Fitur yang didapatkan dari prisma yaitu untuk menyelaraskan model basis data proyek dengan membuat integrasi data yang ada pada basis data dengan API tidak terlalu rumit. Skema basis data dengan Prisma ditunjukan pada Gambar 2.1.

```
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model User {
  id         Int         @id @unique @default(autoincrement())
  email      String      @unique
  username   String      @unique
  password   String
  created_at DateTime    @default(now())
  updated_at DateTime    @updatedAt
  AuthUsers  AuthUsers[]
  Outfit Outfit[]

  @@map("users")
}
```

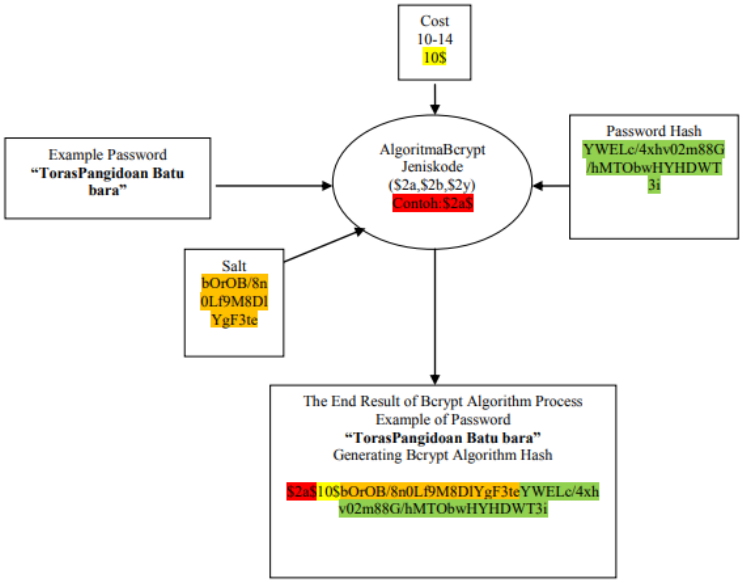
Gambar 2.1 Contoh Skema Tabel Basis Data pada Prisma

Pada Gambar 2.1 merupakan contoh skema pada penggunakan Prisma. Prisma menyediakan skema data yang dapat dibaca dan diubah menjadi definisi yang memudahkan bagi pengembang. Prisma menghasilkan kueri yang menghemat waku karena tanpa perlu menuliskan SQL secara manual dan prisma dapat menangani skema migrasi ketika tidak sesuai dan perlu diubah dalam *database*. Untuk menghubungkan basis data dengan aplikasi, prisma dipasang pada sisi *Backend* yang umumnya dalam menampilkan data para pengembang perlu melakukan penulisan SQL terlebih dahulu, namun dengan Prisma membuat prosesnya lebih mudah karena dapat memungkinkan untuk menunjukan data dalam kode berorientasi objek.

2.11 *Bcrypt*

Bcrypt adalah sebuah fungsi untuk melakukan *hash* yang dikembangkan oleh Niels Provos dan David Mazières berdasarkan enkripsi *Blowfish*. Nama *Bcrypt* terdiri dari B untuk *Blowfish* dan *Crypt*, nama fungsi *hash* yang digunakan dalam sistem kata sandi di UNIX [32]. Bcrypt merupakan sebuah fungsi untuk melakukan *hashing* suatu kata sandi dengan jumlah ilustrasi yang dilakukan lebih banyak untuk membuatnya menjadi lambat dan bertahan lama. Fungsi *hash* yang dilakukan bcrypt ini mampu bertahan terhadap serangan *brute force* dan meningkatkan daya komputasi dengan menggabungkan proses *salt* untuk melindungi dari serangan *rainbow table attack* [33].

Bcrypt ini mengenkripsi 192-bit *hash* dengan menggunakan 128-bit *salt*. Pada prosesnya bcrypt memiliki tahapan inisialisasi dengan secara *default* menggunakan 128-bit pada proses *hashing*. Proses awal bcrypt akan melakukan penurunan kunci, sekumpulan subkunci akan diturunkan dari sebuah kunci utama. Ketika kata sandi yang dimasukkan cukup pendek pada proses awal ini akan dibuat menjadi panjang yang dapat dikatakan melakukan penguatan kunci. Lalu selanjutnya dilakukan enkripsi dengan 192-bit *hash* menggunakan kunci yang telah dibuat dengan dilakukan sebanyak 64 kali. Skema proses *hashing* dengan bcrypt dapat dilihat pada Gambar 2.2.



Gambar 2.2 Proses Keamanan Kata Sandi dengan Bcrypt [33]

Gambar 2.2 merupakan langkah proses melakukan *hashing* pada suatu kata sandi yang dimasukkan, diawali dengan memilih jumlah *cost* yang akan di lakukan proses *hash*. *Default cost* untuk melakukan *hash* sebesar 10. Setelah dilakukan pemilihan nilai *cost* proses *salt* akan melakukan proses acaknya yang akan menghasilkan kode unik pada Gambar 2.2 hasil kode *salt* di *highlight* dengan warna jingga. Kode unik yang melewati proses *hashing* pada Gambar 2.2 diberikan *highlight* dengan warna hijau. Proses *hashing* ini akan menggabungkan kode untuk *salt* dengan kata kunci yang telah dilakukan proses hashing. Hasil akhir dari kode unik yang telah dilakukan proses *hashing* menghasilkan kombinasi karakter yang beragam dan membuat tinggi tingkat keamanan.

2.12 *Jmeter*

Jmeter merupakan alat perangkat lunak untuk melakukan pengujian dari apache berbasis *desktop*, yang ditulis dalam bahasa pemprogramn java. Meskipun JMeter dibuat menggunakan bahasa Java, namun alat ini dapat digunakan untuk menguji aplikasi yang ditulis dengan berbagai bahasa pemprograman. Alat pengujian ini dapat digunakan untuk melakukan pengujian halaman web, aplikasi, dan sumber statis atau dinamis seperti *database*, java object, FTP *server*, dan lain-lain. Jmeter dapat melakukan pengujian beban *load testing* dengan menghasilkan

beberapa parameter seperti dari parameter *quality of service* yaitu metrik yang dapat diukur dengan JMeter adalah TPS (*Transaction Per Second*) atau *throughput*. *Throughput* ini merupakan kecepatan yang di ukur sebagai jumlah permintaan respon dalam waktu tertentu. Latensi rata-rata yang diukur selama pengujian bebas dalam waktu responnya. *Error rate* yang merupakan presentasi permintaan yang gagal dibandingkan dengan total permintaan [34].

JMeter juga dapat mensimulasikan beban *server* yang tinggi dengan membuat beberapa pengguna virtual secara bersamaan di *server web*. Pengujian dapat dilakukan dengan atau tanpa skrip di aplikasi JMeter. Pengujian tanpa skrip dapat dilakukan dengan membuat *Test Plan* yang berisi satu atau lebih *thread*, yang masing-masing bisa terdiri dari beberapa *thread*. Sebuah *thread* adalah simulasi keadaan pengguna yang mengakses langsung aplikasi. Penggunaan Jmeter digunakan untuk melakukan pengujian performa, karena ringan dan mudah dalam penginstalan [35].

2.13 *Kajian Pustaka*

Pada kajian pustaka berisikan rangkuman dari penelitian terdahulu yang berkaitan dengan penelitian yang saat ini dilakukan yaitu mengenai implementasi pembuatan API dengan sistem *cloud storage* untuk aplikasi. Berikut merupakan beberapa penelitian yang menjadi landasan untuk penelitian saat ini.

Penelitian mengenai rancang bangun REST API untuk aplikasi menggunakan perancangan *System Development Life Cycle* (SDLC), dengan melakukan analisis, perancangan, implementasi dan pengujian. hasil dari penelitian ini menjelaskan API dapat membuat proses pengembangan aplikasi menjadi lebih cepat dengan cara membuat sebuah *function* yang terpisah sehingga *developer* perlu merancang fitur yang serupa. Penelitian ini menggunakan sistem *database* berupa ORM (*Object Relation Mapping*), sehingga proses data dapat berhubungan antar tabel [36].

Penelitian mengenai perancangan dan implementasi dari sistem informasi manajemen ini menggunakan metode identifikasi masalah dan melakukan perancangan RESTful API dengan menggunakan bahasa pemprograman java dan basisdata MySQL. Pengujian yang dilakukan pada penelitian ini dilakukan

menggunakan *software* Postman dan melakukan implementasi RESTful API yang di rancang. hasil dari penelitan menunjukan bahwa setiap metode HTTP yang digunakan mampu berjalan dan mengasilkan status berhasil [37].

Penelitian ini membahas terkait penerapan penyimpanan berbasis *cloud* untuk perusahaan hasil dari penelitian ini dengan menerapkan sistem penyimpanan *cloud* dapat memberikan respon positif bagi pengguna terkait efektifitas dan produktifitas pengiriman dan penyimpanan data dibandingkan dengan pertukaran data secara manual dengan media *hardware* [38].

Penelitian mengenai penerapan dan perancangan *backend* API dengan menggunakan *framework* express JS untuk sebuah aplikasi dengan menggunakan metode *waterfall* yaitu dengan melakukan analisa kebutuhan dari aplikasi, desain sistem, implementasi, pengujian, dan penerapan dari *back-end* yang telah dirancang. Penelitian ini menggunakan platform Nodejs dengan framework *Express* yang telah menyediakan fitur *routing*, *handling*, dan *response*. Pada penelitian ini juga memanfaatkan layanan GCP untuk sistem *cloud*. Hasil dari penelitian ini menjelaskan penerapan yang dilakukan membuat keseluruhan API berfungsi dengan baik dan didokumentasi untuk memudahkan dalam mengakses API [39].

Penelitian mengenai perancangan *back-end server* dengan menggunakan arsitektur REST dan platform Nodejs. pada penelitian ini menjelaskan peggunaan Nodejs memberikan keunggulan pada teknik *non-blocking* yang memungkinkan banyak *request* yang dapat diselesaikan secara parallel dengan kinerja dan fungsionalitas yang baik. Penelitian ini menjelaskan layanan REST API yang berjalan dengan Nodejs lebih sesuai untuk mengatasi respon langsung terhadap permintaan yang masuk tanpa melibatkan banyak perhitungan [40].

Penelitian ini melakukan perancangan dan implementasi RESTful API untuk aplikasi *mobile* pembelajaran flora dan fauna dengan menggunakan metode *waterfall* dengan Nodejs dan *framework* Express JS yang dilakukan *deployment* pada layanan *cloud computing Google Cloud Platform* (GCP) yang menghasilkan rancangan berhasil dibangun dan dengan menggunakan layanan dari GCP dapat mendukung performa RESTful API dari aplikasi EksFlorasi [41].

Oleh karena itu, dari hasil penelitian sebelumnya yang menjadi rujukan kajian pustaka untuk penelitian ini, maka pada penelitian ini melakukan implementasi pembuatan API dengan sistem *cloud storage* untuk komponen *backend* pada aplikasi rekomendasi pakaian dengan menggunakan metode analisis kebutuhan aplikasi, perancangan, implementasi, dan pengujian API yang telah dibuat, dengan begitu hasil dari API yang telah dibuat akan berfungsi dengan baik sebagai *backend* untuk aplikasi rekomendasi pakaian.

BAB III
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodologi penelitian ini memiliki beberapa tahapan proses untuk mencapai hasil yang diinginkan, diantaranya sebagai berikut.

1. Studi Literatur

Studi literatur dilakukan dengan upaya mengumpulkan data dengan cara menelusuri berbagai macam dokumen antara lain mengumpulkan informasi mengenai *cloud computing*, API, NodeJS, Google *cloud platform*, dan juga terkait dengan pengolahan data pada *website* yang berupa jurnal, buku, dan juga penelitian.

2. Perancangan

Perancangan adalah proses menggambarkan, merencanakan, dan membuat sketsa yang bertujuan untuk menjadi langkah awal dalam merancang suatu sistem. Selain itu, perancangan merupakan tahap pengembangan yang dilakukan setelah analisis selesai dilakukan, dengan fokus pada pembuatan rencana yang lebih terarah dan terstruktur. Oleh karena itu, dalam konteks pengembangan sistem, perancangan berperan penting sebagai dasar untuk memastikan setiap langkah selanjutnya dapat berjalan secara sistematis dan sesuai dengan kebutuhan yang telah diidentifikasi.

3. Perancangan Penyimpanan Data pada Aplikasi

Perancangan penyimpanan data dalam aplikasi merupakan proses dalam melakukan rancangan struktur data yang akan digunakan oleh aplikasi untuk menyimpan, mengelola, dan mengakses data. Skema penyimpanan mendefinisikan entitas atau tabel serta atribut atau kolom, dan hubungan-hubungan antara entitas yang lain. Pengembangan aplikasi ini perlu adanya perancangan penyimpanan data karena penyimpanan yang dirancang dengan baik dapat memastikan konsistensi, integritas, dan kinerja aplikasi. Proses perancangan penyimpanan data melibatkan analisis kebutuhan data, pemilikan tipe data, desain skema, dan implementasi rancangan.

4. Perancangan API

Perancangan API untuk aplikasi merupakan proses yang penting dalam pengembangan aplikasi yaitu dengan merencanakan, merancang, dan mengimplementasikan API yang memungkinkan pertukaran data dalam suatu aplikasi berjalan dengan baik, aman dan efisien. Proses perancangan API melibatkan keperluan seperti pemahaman kebutuhan untuk aplikasi rekomendasi pakaian yang dibuat, identifikasi fungsi atau fitur yang ada di aplikasi rekomendasi pakaian, mendesain struktur data, dan penanganan otoritas serta autentifikasi.

5. Implementasi

Implementasi dilakukan untuk mewujudkan perencanaan menjadi wujud sistem *backend* yang berhasil dan bekerja optimal sesuai dengan perancangan yang telah dibuat, sehingga dapat dilakukan ujicoba dan validasi fungsionalitas sistem yang telah dibuat.

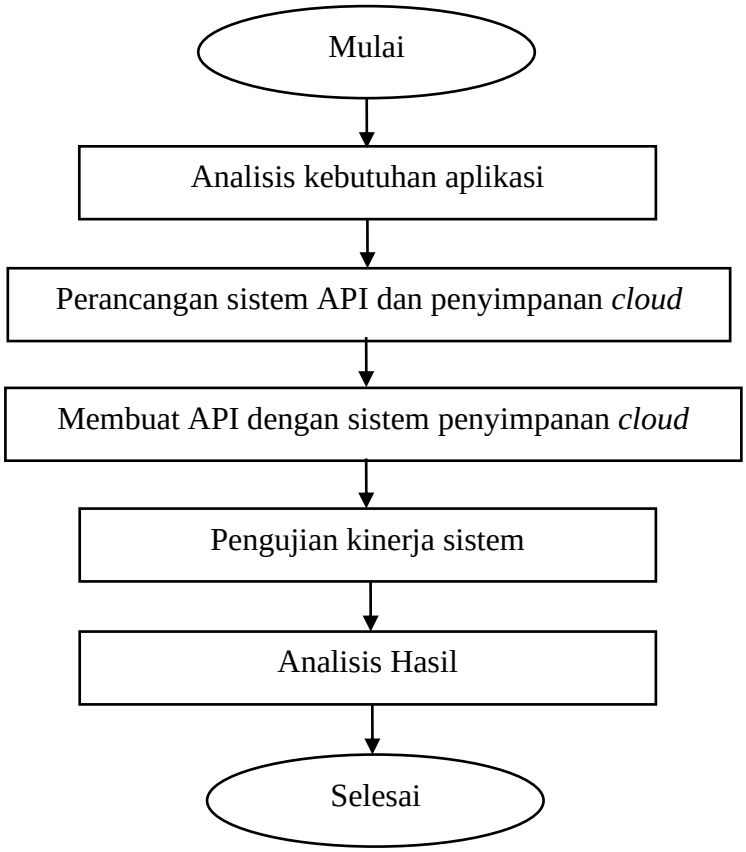
6. Pengujian Hasil Penelitian

Pengujian dari hasil penelitian merupakan tahapan akhir yang perlu dilakukan dengan tujuan mengetahui hasil performa penggunaan API untuk aplikasi yang dibuat. Tujuan pada tahapan ini untuk melakukan pengujian fungsional API, pengujian performa API dan pengujian respon sistem dalam *load condition* yang dihasilkan dari aplikasi rekomendasi pakaian. Pengujian yang didapatkan dari penelitian ini dapat menjadi bahan pertimbangan untuk mengoptimalkan kinerja aplikasi lebih baik pada penelitian-penelitian berikutnya.

3.2 Diagram Alir Penelitian

Diagram alir penelitian merupakan salah satu metode dalam memvisualisasikan pemahaman mengenai langkah-langkah yang akan dilakukan dalam proses penelitian pembuatan API dengan penyimpanan *cloud storage*. Diagram alir ini membantu menjelaskan hubungan antara satu kegiatan dengan kegiatan lainnya dalam penelitian yang akan memberikan pemahaman lebih baik mengenai proses yang dilakukan dalam penelitian. Diagram alir penelitian pada akan membantu menggambarkan urutan kegiatan yang akan dilakukan dalam

penelitian serta hubungan antara langkah-langkah penelitian yang dilakukan yang ditunjukkan pada Gambar 3.1.



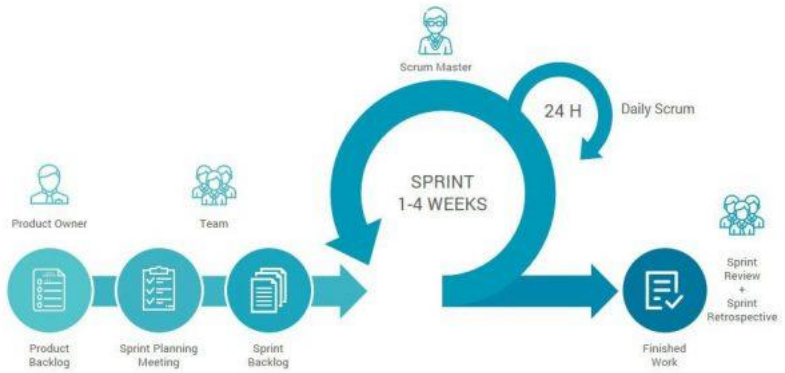
Gambar 3.1 Diagram Alir Penelitian

Gambar 3.1 merupakan diagram alir dari penelitian implementasi penggunaan API dengan *cloud storage* untuk aplikasi rekomendasi pakaian. Langkah awal dalam penelitian yaitu dimulai dengan menganalisis kebutuhan dari aplikasi rekomendasi. Selanjutnya melakukan perancangan dan implementasi API untuk aplikasi rekomendasi dengan penyimpanan menggunakan sistem *cloud storage*. Tahapan akhir dalam alur penelitian yaitu melakukan pengujian performa waktu respon dari implementasi API dan kemudian menganalisis hasilnya.

3.3 Metode *Agile Scrum*

Penelitian merupakan salah satu bagian dari kegiatan yang dilakukan di Bangkit *Academy* dengan melakukan pengerjaan *capstone* untuk proyek akhir dengan metode keseluruhan dengan *Agile Scrum. Agile scrum* merupakan suatu pendekatan teknik metodologi untuk melakukan pengembangan perangkat lunak.

Metode ini biasa digunakan pada proyek yang bersifat inkremental atau melakukan pengembangan sedikit demi sedikit dan melakukan praktik membuat dan menyempurnakan proyek yang memungkinkan tim untuk dapat responsif tehadap setiap perubahan kebutuhan pengerjaan proyek. Penjelasan mengenai metode *Agile Scrum* tersebut sesuai dengan kebutuhan pengerjaan yang dilakukan pada penelitian ini. Pengerjaan proyek dilakukan oleh tiga tim yang memiliki masing-masing pekerjaan yang berbeda. Gambaran mengenai metode *agile scrum* secara lebih jelasnya dapat dilihat melalui Gambar 3.2.



Gambar 3.2 Diagram Metode *Agile Scrum*

Gambar 3.2 menjelaskan mengenai poin-poin yang terdapat pada metode *agile scrum* dan menjadi bahan pertimbangan dalam metode pengembangan *agile scrum*. Beberapa langkah dalam menggunakan metode pengembangan *agile scrum* yang dijelaskan dalam gambar tersebut diantaranya adalah melakukan *product backlog, sprint planning meeting, sprint backlog,* dan *finished work*. Sedangkan peran yang terdapat dalam metode *agile scrum* diantaranya adalah product *owner, team,* serta *scrum master*.

3.4 Pembuatan API

Pembuatan API merupakan tahapan yang sangat penting untuk proses pembuatan aplikasi rekomendasi pakaian pada penelitian ini. Dalam penelitian ini untuk tahapan pembuatan API menggunakan bahasa program javascript karena javascript dapat membuat fleksibilitas yang tinggi. Selain fleksibilitas yang tinggi javascript memungkinkan adopsi model pengembangan yang terpusat pada

layanan *service-oriented*, dengan komponen aplikasi yang dikembangkan secara independen dan diintegrasikan melalui antarmuka yang didefinisikan dengan baik.

Pembuatan API dalam penelitian ini dengan menggunakan bahasa pemprograman javascript di sisi *server* maka *runtime* yang javascript diperlukan yaitu Node.JS. Node.JS sebagai *server side* dapat menciptakan solusi *end to end* yang seragam dan efisien dengan menggunakan model *non-blocking* I/O yang mudah. Node.JS mampu menangani banyak koneksi secara bersamaan dengan performa yang baik tanpa perlu mengorbankan kinerja, sehingga pembuatan API akan lebih efisien dan skalabel.

Penggunaan bahasa pemprograman javascript dengan Node.JS memiliki ketersediaan berbagai pustaka dan kerangka kerja dalam mempermudah dan mempercepat pembuatan API. Kerangka kerja yang dipilih dan digunakan dalam membuat API adalah Express.js yang merupakan salah satu *framework* yang baik dalam membangun aplikasi di Node.js. Kerangka kerja tersebut dapat membuat pengembangan lebih efisien dan fleksibel sesuai dengan kebutuhan aplikasi karena Express.js menyediakan fitur-fitur yang mempercepat seperti fitur *routing* yang baik dan *middleware* yang dapat disesuaikan, sehingga dapat memilih kendali penuh atas implementasi dari API.

Ekosistem dalam Node.js memiliki manajer paket yang digunakan untuk mengelola paket-paket perangkat lunak yaitu NPM (*Node package manager*). NPM ini dapat memberikan akses ke banyak paket modul yang digunakan untuk memperluas fungsionalitas aplikasi. Pada penelitian ini mengunakan modul yang membantu pemetaan objek relasional dengan Express.js. untuk menggunakannya perlu dilakukan pengunduhan dengan bantuan NPM. Setelah selesai mengunduh modul express.js, tahap yang perlu dilakukan yaitu memasukan modul tersebut ke dalam program dengan perintah “require()”. Perintah “require()” merupakan perintah yang digunakan untuk memuat modul ke *file* javascript yang sedang dieksekusi. Setelah berhasil menerapkan modul Express.js maka dalam pembuatan API dapat melakukan fungsi-fungsi yang ditawarkan oleh modul Express JS untuk membuat kerangka kerja yang baik. Kode penggunaan untuk express.js ditunjukkan pada Gambar 3.3.

```
const express = require('express')
const cors = require('cors');
const routes = require('./routes');
const morgan = require('morgan');
const app = express();
const port = process.env.PORT;
app.use(cors());
app.use(morgan("dev"))
app.use(express.json())
app.use(express.urlencoded({extended:true}))
app.get('/', async (req, res) => {
    res.status(200).json({
        message: 'Welcome to BeFitOutfit'
    })})
 routes(app)
app.listen(port,()=>{
  console.log(`Your application running in
http://localhost:${port}`)
})
```

Gambar 3.3 Penggunaan Express.JS

Gambar 3.3 merupakan *listing* program implementasi penggunaan *framework* Express JS. terdapat tiga buah fungsi Express yang akan digunakan dalam tahapan pembuatan API dalam penelitian ini yaitu “app.use()”, “app.get()”, dan “app.listen()”. Fungsi penggunaan perintah “app.use()” digunakan untuk menggunakan *middleware* terhadap aplikasi express, dengan perintah tersebut juga dapat mendaftarkan middleware untuk melakukan tugas dalam pengolahan data permintaan. Perintah “app.get()” memiliki fungsi yang digunakan untuk menentukan penanganan permintaan dengan metode *Get* pada *route* ketika metode GET tersebut diterima. Perintah “app.listen()” memiliki fungsi yang digunakan untuk memulai *server* Express dan mendengarkan pada *port* tertentu. Ketika *server* berhasil dijalankan maka akan terdapat *callback* yang diberikan seperti mencetak pesan ke *console* bahwa server berhasil berjalan.

Tahapan implementasi *framework* ExpressJS telah dilakukan dengan konfigurasi sesuai kebutuhan pembuatan API. Selanjutnya agar susunan file mudah dikelola perlu dilakukan pembuatan struktur *folder* yang baik dan jelas. Pembuatan standart struktur *folder* dalam pembuatan API sangat diperlukan dalam memudahkan pengelolahan *file* yang dibutuhkan. Pembuatan struktur *folder* yang baik dalam pelaksanaannya akan menjadikan struktur *folder* lebih

teratur dan konsisten dalam proyek, sehingga membuat kode mudah dipahami dan diorganisir. Struktur *folder* yang digunakan untuk penelitian ini ditunjukkan pada Gambar 3.4.

Gambar 3.4 Struktur Folder Program API

Gambar 3.4 menjelaskan struktur *folder* yang digunakan pada penelitian ini. Pada gambar menampilkan tiga buah *folder* penting dengan memiliki *file* yang dibuat dalam proses pembuatan API. Tiga buah *folder* tersebut adalah *Controllers*, *helpers*, dan *Modules*. *Folder controller* dibuat dengan tujuan menyimpan *file* yang bertanggung jawab dalam menangani logika aplikasi dengan permintaan HTTP masuk dan untuk mengatur penanganan rute API pada Express.JS, dengan pembentukan *folder* ini dapat dengan mudah memisahkan logika kebutuhan dari logika penanganan permintaan HTTP. *Folder helpers* merupakan direktori yang berisi *file* dengan fungsi yang membantu di berbagai bagian aplikasi karena berisi program yang sering dibutuhkan dalam pembuatan API. Salah satu contoh *file* yang ada pada *folder helpers* adalah Middleware.js yang dapat dengan mudah ditambahkan pada rute-rute API. *Folder modules* merupakan *folder* yang berisi *file* dengan modul berisi logika yang diperlukan aplikasi. Setiap file pada *folder modules* ini bertanggung jawab untuk satu entitas tertentu dalam API. Seperti contohnya membuat, mengambil, menghapus, atau merubah data sesuai keperluan fitur dalam aplikasi pada penelitian ini.

Pada penelitian ini dilakukan pembuatan API untuk sistem autentifikasi atau modul pengguna yang digunakan untuk bagi para pengguna, dan pembuatan API untuk sistem manajemen pakaian atau modul yang dibuat untuk segala kebutuhan aplikasi mengenai sistem yang ada pada aplikasi rekomendasi pakaian ini. Untuk bagian pembuatan yang dilakukan yaitu dalam proses autentifikasi pengguna ke dalam aplikasi API yang dibuat adalah sebagai berikut.

1. API Registrasi

Pembuatan API pertama dilakukan untuk proses autentifikasi akun pengguna agar memiliki fungsi membuat akun baru bagi pengguna sesuai kebutuhan aplikasi fitur untuk pengguna melakukan registrasi memerlukan data *username*, *email*, dan juga *password* yang merupakan salah satu keamanan yang diperlukan agar akun pengguna lebih personal. Pembuatan API registrasi ini memerlukan alur logika yang sesuai. Kebutuhan proses registrasi yang baik dalam penelitian ini alur logikanya adalah diawali dengan memvalidasi data masukkan, melakukan *hashing password*, pengecekan duplikasi dari *username* dan *email* yang dimasukkan, terakhir pembuatan pengguna baru kedalam *database*. Alur logika seperti ini membuat fitur registrasi akan baik dalam penggunaannya.

2. API *Login*

Fitur *login* membuat personalisasi pengalaman bagi pengguna karena, dengan *login* aplikasi dapat memberikan pengalaman yang personal dengan menyimpan data preferensi pengguna. API *login* merupakan sistem yang memiliki tanggung jawab untuk pengguna masuk ke dalam aplikasi dengan data yang valid atau melalui proses autentikasi yang merupakan pembuktian akun pengguna untuk *login* ke dalam aplikasi. Proses pembuktian identitas pengguna menggunakan kredensial *email* dan juga *password* yang valid. API *login* ini akan memproses otentikasi pengguna dan membiarkan akses ke dalam aplikasi ketika berhasil masuk.

3. API *Update User*

Aplikasi rekomendasi pakaian pada penelitian ini memiliki kebutuhan fitur untuk melakukan *update* atau merubah akun pengguna. Pentingnya fitur ini terhadap aplikasi khususnya pengguna agar pengguna dapat menjaga informasi pribadi agar tetap terkini serta akurat. API *update user* membuat *user* dapat

memiliki kemampuan untuk mengelola dan memperbarui informasi pribadi pengguna seperti *username* dan *email* yang digunakan pada saat *login*. Langkah program dalam pembuatan API *update user* ini diawali dengan memvalidasi data, kemudian melakukan pembaruan data akun pengguna.

Selain pembuatan API untuk proses autentifikasi. Dilakukan juga pembuatan API untuk proses manajemen pakaian. Manajemen pakaian pada penelitian ini dilakukan untuk melakukan proses memasukkan, menampilkan, merubah, serta menghapus data-data pakaian yang diperlukan pengguna dalam aplikasi. API yang dibuat untuk proses manajemen pakaian adalah sebagai berikut.

1. API *Add Outfit*

Fitur yang dibutuhkan pada aplikasi untuk melakukan *add outfit* ini ialah mengunggah foto pakaian sesuai keinginan pengguna, memasukkan nilai atribut pada gambar pakaian yang diungah seperi nama, *event* dan *type outfit* sesuai dengan gambar pakaian yang diunggah. API *Add Outfit* ini menggunakan google *cloud storage* untuk menyimpan gambar pakaian yang diunggah oleh pengguna. Ketika file gambar yang diunggah dan memiliki URL *public* di google *cloud storage*, maka URL tersebut akan digunakan untuk menyimpan gambar pakaian tersebut dalam *cloud storage*. Setelah proses validasi dan otentikasi selesai, pakaian baru akan ditambahkan ke dalam *database.*

2. API *Get Outfit*

API *get outfit* merupakan API yang bertanggung jawab untuk dapat melakukan proses dalam menampilkan informasi data pakaian yang telah diunggah. Proses yang perlu dilalui untuk dapat mengambil daftar pakaian pengguna dari *database* yaitu dengan memiliki akses *database* dengan token JWT. Data pengguna ID digunakan untuk mencari semua pakaian terkait dengan pengguna di *database*. Proses mengambil daftar pakaian dari *database* sesuai dengan pengguna terkait berhasil dilakukan.

3. API *Update Outfit*

Layanan API *update outfit* ini bertujuan untuk memenuhi kebutuhan aplikasi dalam fitur yang membuat pengguna mampu mengubah data pakaian yang dimiliki oleh pengguna dalam aplikasi. Proses yang dilalui untuk melakukan

pembuatan API ini diawali dengan mengambil Id *outfit* untuk menentukan pakaian mana yang akan diperbarui. API *update* yang dibuat mampu mengubah tiga buah data yang ada pada *detail* informasi pakaian pengguna. Data yang dapat diubah yaitu nama pakaian, tipe pakaian, dan nilai *include* untuk rekomendasi.

4. API *Delete Outfit*

API *delete outfit* diperlukan dengan membuat manajemen data lebih efisien bagi pengguna karena dapat menghapus pakaian yang sudah tidak relevan atau tidak digunakan lagi dan dapat meningkatkan pengalaman pengguna menggunakan aplikasi. Sebelum data dihapus data perlu dilakukan validasi data *outfit* ID. Setelah proses validasi berhasil, program API akan menghapus pakaian yang sesuai dengan ID yang diberikan dari *database.*

5. API *Get Rekomendasi*

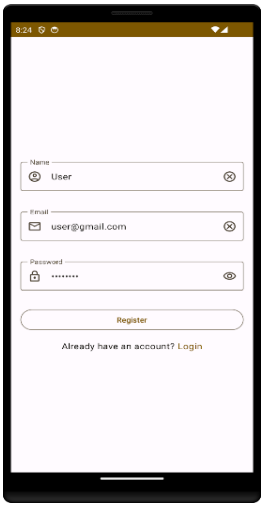
Pembuatan API rekomendasi untuk menampilkan hasil rekomendasi daftar pakaian sesuai acara yang diinginkan dengan daftar pakaian tipe atasan yaitu atasan dan bawahan sesuai dengan preferensi pengguna. Proses awal pada tahapan program API rekomendasi dimulai dengan memvalidasi parameter. Setelah proses validasi selesai API akan menyusun data rekomendasi dalam bentuk yang sesuai. Data rekomendasi akan memberikan rincian pakaian hasil rekomendasi seperti nama, tipe pakaian, jenis acara, gambar pakaian, dan status inklusi

BAB IV
HASIL PENELITIAN

Berdasarkan perancangan di Bab 3 diperoleh hasil Analisa mengenai implementasi penggunaan API dengan *Cloud Storage* untuk aplikasi rekomendasi pakaian. Hasil penelitian dan analisa tersebut di jelaskan pada Bab ini.

4.1 Analisis kebutuhan Aplikasi Rekomendasi

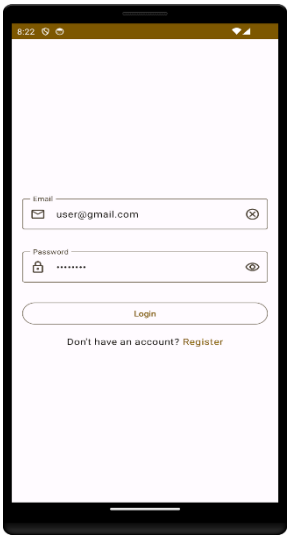
Aplikasi rekomendasi yang ada pada penelitian merupakan aplikasi rekomendasi yang membantu dalam menentukan pakaian. Fitur yang ada dan digunakan dalam aplikasi yaitu pengguna memasukan pakaian-pakaian yang dimiliki, kemudian aplikasi akan menentukan pemilihan pakaian dari atasan maupun bawahan untuk pengguna sesuai dengan kebutuhan acara tertentu dengan keinginan pengguna. Aplikasi rekomendasi ini diawali dengan tahapan registrasi seperti ditampilkan pada Gambar 4.1.



Gambar 4.1 Tampilan Registrasi

Pada tampilan diawali dengan tampilan *login* pengguna untuk masuk ke dalam aplikasi. Pada tampilan Gambar 4.1 terdapat pilihan untuk melakukan registrasi terlebih dahulu, hal ini dilakukan pada saat pengguna belum memiliki akun. Gambar 4.1 merupakan sebuah tampilan untuk registrasi akun pengguna untuk aplikasi rekomendasi penelitian ini. Tampilan *form* registrasi ini memiliki beberapa data yang diperlukan untuk pengguna memiliki akun yang dapat

digunakan untuk *login* ke dalam aplikasi. Data tersebut berupa *username*, *email* dan *password*. Data tersebut merupakan data kebutuhan yang digunakan dalam pembuatan API pada aplikasi. Data yang telah diisi oleh pengguna akan terkirim ke sistem yang telah di rancang. Tampilan menu *login* untuk aplikasi rekomendasi untuk penelitian ini ditunjukkan pada Gambar 4.2.

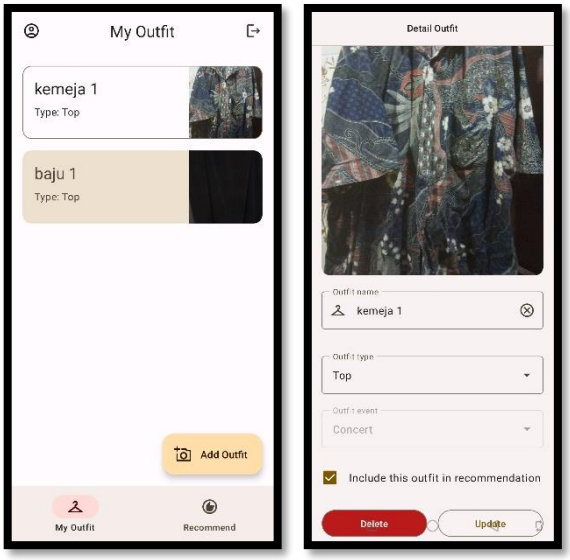


Gambar 4.2 Tampilan Login

Setelah registrasi akun berhasil dilakukan, pengguna akan memiliki akun yang digunakan untuk *login* dalam aplikasi. Gambar 4.2 merupakan tampilan untuk *login* pengguna masuk ke dalam aplikasi, dalam *login* tersebut menggunakan data yang sudah dimasukkan sebelumnya oleh pengguna *email* dan *password* pada *form* registrasi. *Login* menggunakan akun tersebut dapat memberikan akses ke fitur-fitur pada aplikasi dan memberikan pengalaman pengguna lebih personal. Data akun yang telah dibuat dan dirancang dalam *database* agar saling terhubung atau terintegrasi antara data akun registrasi pengguna dan data *login* pengguna, dengan menuliskan akun *email* dan *password* yang telah diregistrasikan.

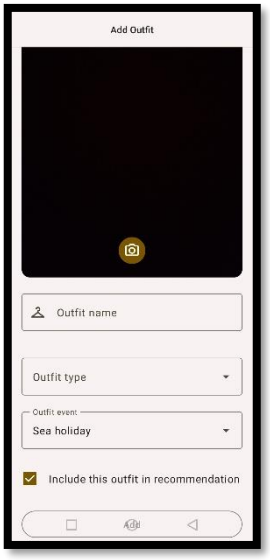
Setelah pengguna melakukan *login* maka akan langsung tampil tampilan etalase pakaian dengan menu *My Outfit.* Tampilan daftar pakaian ini berisi pakaian-pakaian dari lemari pribadi pengguna yang telah dimasukan kedalam aplikasi. Kumpulan baju ini akan digunakan seperti lemari pakaian yang nantinya digunakan untuk mendapatkan pakaian sesuai yang diberikan rekomendasi untuk acara yang ingin diikuti. Setiap pakaian yang dimasukan perlu menuliskan nama

pakaian dan tipe dari pakaian tersebut termasuk atasan atau bawahan. Sebagai pengguna, dalam menu *My Outfit* dapat melakukan *update* atau perubahan nama pakaian dan tipe dari pakaian yang telah dimasukan, pada *detail outfit* ini pengguna dapat menghapus pakaian dari menu daftar pakaian. Tampilan menu daftar pakaian dan *detail* pakaian dapat dilihat pada Gambar 4.3.



Gambar 4.3 Tampilan Etalase dan Detail Pakaian

Gambar 4.3 merupakan tampilan dari aplikasi setelah pengguna berhasil login untuk masuk ke dalam aplikasi, tampilan daftar pakaian dengan menu *My outfit* ditampilkan pada gambar sebelah kiri. Pada sisi sebelah kanan gambar menunjukan tampilan ketika memilih salah satu pakaian yang telah dimasukan pada *my outfit* yang menampilkan tampilan *detail outfit* dari pakakian yang telah di pilih. *Detail outfit* ini memiliki kebutuhan fitur API *delete* dan *update* dalam pakaian yang telah dimasukan. Pada Gambar 4.3 terdapat *bar* atau menu bertuliskan *add outfit* di bagian kanan bawah fitur tersebut akan menampilkan tampilan halaman *add outfit* yang ada pada Gambar 4.4.



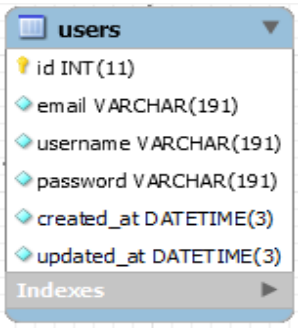
Gambar 4.4 Tampilan Add Outfit

Gambar 4.4 merupakan tampilan dalam melakukan *input* pakaian yang diinginkan berupa foto pakaian pengguna ke dalam aplikasi. Dalam menu *add outfit* ini terdapat pengisisan data berupa nama pakaian, dan tipe pakaian. Setelah pakaian dimasukkan maka data pakaian tersebut akan tersimpan ke dalam penyimpanan *cloud* yang telah dibuat dan telah diintergasikan antara API dengan penyimpanan *cloud*. Untuk memenuhi kebutuhan aplikasi rekomendasi pakaian pada penelitian ini dalam intergasi fitur dan data-data yang dibutuhkan oleh aplikasi, maka perlu pembuatan API dan penyimpanan *cloud* agar aplikasi berjalan dengan optimal.

4.2 Basis Data

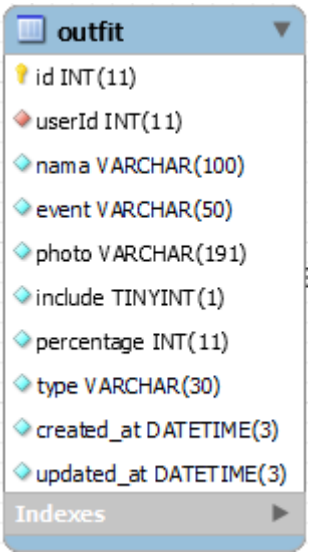
Basis data merupakan hal yang sangat penting dalam melakukan pengembangan aplikasi android. basis data perlu dilakukan karena pada aplikasi perlu merencanakan struktur dan organisasi dari basis data yang akan digunakan. basis data yang terstruktur dapat memudahkan pengembangan aplikasi karena menyediakan kerangka kerja yang jelas. Pada basis data akan dibatasi oleh data yang membantu menjaga integritas data dengan menangani data yang tidak valid atau tidak konsisten dalam basis data. Berdasarkan kebutuhan data yang digunakan pada aplikasi rekomendasi pakaian ini maka basis data pada penelitian ini dibuat empat buah tabel. Keempat buah tabel tersebut dapat memenuhi

kebutuhan penelitian ini antara lain tabel Roles, Auth_User, User, dan Outfit. Tabel User ditujukan pada Gambar 4.5.



Gambar 4.5 Data *User*

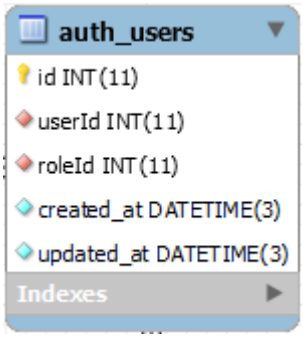
Pada Gambar 4.5 merupakan gambar data untuk *user* atau pengguna dalam basis data. Kolom tabel ini berisikan ID sebagai *primary key*. data *email*, *username*, *password* dengan tipe data *string*. Data *created_at* dan *update_at* dengan tipe data DATETIME. Tabel ini untuk menyimpan data identitas atau akun pengguna yang ditulis pada saat registrasi dan digunakan untuk masuk ke dalam aplikasi. Tabel data *outfit* yang dibuat ditunjukan pada Gambar 4.6



Gambar 4.6 Data *Outfit*

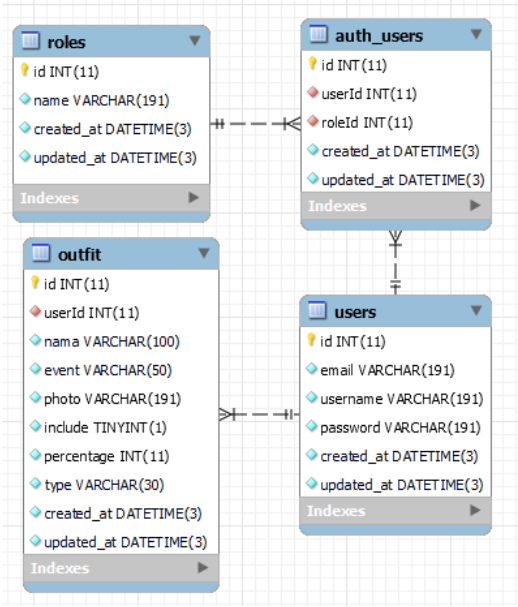
Pada Gambar 4.6 merupakan gambar data untuk *Outfit* atau pakaian pengguna dalam basis data. Kolom tabel ini berisikan id sebagai *primary key* yaitu kunci primer dari salah satu kolom yang unik dan mengidentifikasi setiap baris dalam tabel. *Primary key* penting untuk menjaga keunikan data agar entitas memiliki

identifikasi yang jelas. Kolom UserId merupakan data sebagai *foreign key* yaitu kolom gabungan dalam sebuah tabel yang menyimpan nilai pada kunci primer lain yaitu kolom Id pada tabel *Users*. *Foreign key* digunakan untuk membangun hubungan antara dua tabel, dan menghubungkan informasi antara tabel. Kolom data lainnya berisi mengenai *detail* pakaian yang diperlukan aplikasi seperti nama, *event* dan *type* dengan tipe data *string*. Data created_at dan update_at dengan tipe data DATETIME. Tabel ini digunakan untuk menyimpan data pakaian yang dimasukan dalam aplikasi. Data selanjutnya adalah data autentifikasi yang ditunjukan oleh table auth_users pada Gambar 4.7.



Gambar 4.7 Data Auth_Users

Pada Gambar 4.7 merupakan data untuk *autentifikasi user* atau otentikasi pengguna dalam basis data. Kolom tabel ini berisikan id sebagai *primary key* dengan tipe data *interger*. Dalam tabel ini terdapat kolom UserId dan RoleId merupakan data sebagai *foreign key* dengan tipe data *integer* yaitu kolom gabungan dalam sebuah tabel yang menyimpan nilai pada kunci primer lain yaitu kolom Id pada tabel Users dan juga *Role*. Tabel ini dibuat untuk proses otentikasi pengguna, oleh karena itu tabel berelasi dengan tabel *roles* dan juga *users*. Adapun struktur untuk memodelkan entitas dan hubungan antara tabel dan kolom untuk data *foreign key* dalam basis data yang digunakan aplikasi penelitian ini pada Gambar 4.8.



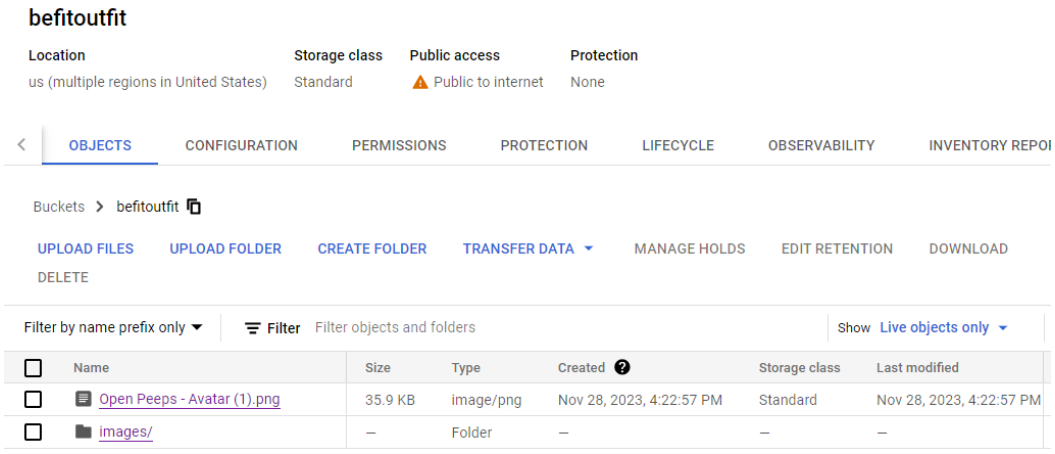
Gambar 4.8 Perancangan Basis Data Aplikasi

Gambar 4.8 merupakan skema basis data yang menjelaskan hubungan antara keempat tabel satu sama lain. Pada skema tersebut juga menjelaskan relasi antara kolom dari isi masing-masing tabel yang digunakan pada penelitian ini. Pemilihan jenis basis data dari perancangan yang digunakan merupakan hal penting dalam keseluruhan kinerja dalam pengembangan aplikasi. Perancangan yang dilakukan pada penelitian ini menggunakan basis data SQL karena SQL memiliki struktur yang terorganisir dengan model relasional memungkinkan data disimpan dalam tabel yang terorganisir dengan jelas.

4.3 Implementasi Sistem *Cloud*

Fitur aplikasi rekomendasi pakaian terdapat proses untuk memasukkan foto pakaian. Ketika menggunakan sistem penyimpanan dengan melibatkan perangkat pengguna, hal ini memberatkan perangkat android penyimpanan internal pengguna. Maka pada penelitian ini untuk meningkatkan skalabilitas dari penyimpanan yang diperlukan sistem *cloud storage*. Pada penelitian ini menggunakan layanan *cloud computing* dengan menggunakan *google cloud platform* menggunakan layanan google *cloud storage* dalam melakukan penyimpanan data yang dibutuhkan pada aplikasi seperti penyimpanan untuk foto

pakaian yang diunggah oleh pengguna. Tampilan *bucket* pada *cloud storage* yang digunakan ditunjukan pada Gambar 4.9.



Gambar 4.9 *Bucket Cloud Storage*

Gambar 4.9 merupakan tampilan *bucket* yang digunakan untuk penyimpanan *cloud* mengguna GCP dalam penelitian ini untuk aplikasi rekomendasi pakaian. Penyimpanan *cloud* dirancang mampu untuk menyediakan kapasitas penyimpanan yang tidak terbatas, dengan penyimpanan *cloud* ini dapat dengan mudah menambah atau mengurangi kapasitas penyimpanan sesuai kebutuhan aplikasi tanpa perlu mengelola infrasturuktur fisik karena dilakukan dengan penyimpanan *cloud*. Penyimpanan *cloud* mampu menawarkan ketersediaan tinggi untuk memastikan data tetap tersedia dalam situasi kegagalan penyimpanan perangkat pengguna. pada Gambar 4.5 terdapat file storge.js yang ada pada folder *helpers*. *File* tersebut merupakan program yang berisi mengenai penggunaan sistem penyimpanan google *cloud*.

```
const { Storage } = require('@google-cloud/storage')
const keyFilename = process.env.FILE_KEY_NAME
const bucketName = process.env.BUCKET_NAME
```

Listing code diatas menjelaskan mengenai proses awal program dalam sistem *cloud storage*. Tahapan proses untuk penyimpanan yang digunakan diawali dengan mengimpor modul konfigurasi "@google-cloud/storage" untuk berinteraksi langsung dengan layanan Google *cloud storage*. Modul yang diimpor merupakan klien resmi dari Google *Cloud Storage* untuk Node.js. Konfigurasi

proyek untuk nama *file* dan nama *bucket* Google *Cloud Storage* diberikan layanan dengan *file* kredensial untuk dapat mengakses penyimpanan *cloud* yang diambil dari *environment variables* menggunakan “process.env”. Kode untuk penggunaan sistem penyimpanan *cloud* dengan menggunakan GCP pada aplikasi rekomendasi pakaian untuk penelitian ini ditunjukkan seperti kode berikut.

```
const gcsname = `${Date.now()}-${req.file.originalname}`
            const file = bucket.file(`images/${gcsname}`)
            const stream = file.createWriteStream({
                metadata: {
                    contentType: req.file.mimetype
                }
            })
            stream.on('error', (err) => {
                req.file.cloudStorageError = err
                next(err)
            })
            stream.on('finish', () => {
                req.file.cloudStorageObject = gcsname
                req.file.cloudStoragePublicUrl =
publicUrl(gcsname)
                next()
            })
```

Listing program tersebut adalah metode untuk menggunggah gambar yang dimasukkan kedalam aplikasi oleh pengguna. Program tersebut bertanggung jawab untuk menggungah ke penyimpanan *bucket* yang ada pada *cloud storage*. Dalam *code* terdapat fungsi “const gcsname” yang berfungsi untuk memberikan rincian yang terdiri dari *timestamp* yaitu menunjukan waktu unggah *file*, dan nama asli *file* yang diterima. Kemudian *file* tersebut diunggah ke *bucket cloud storage* menggunakan perintah “createWriteStream”. Setelah unggah berhasil dilakukan URL *public file* yang diunggah, disiapkan, dan dikembalikan sebagai respon. Dengan menggunakan proses ini maka *file* yang diterima dari permintaan HTTP diunggah secara asinkron ke Google *Cloud Storage* dengan menggunakan *stream*, sehingga menghindari memori yang terlalu besar ketika menangani *file* besar. Dalam program ini terdapat metode untuk *delete file* yang ada pada google *cloud storage*. Program ini bertanggung jawab untuk menghapus *file* dari *bucket cloud storage*. Cara metode ini bekerja ID *outfit* yang diterima dari permintaan HTTP digunakan untuk mencari gambar pakaian yang sesuai dalam basis data. Ketika

file tersebut ditemukan URL gambar diekstrak dari data dan *file* yang sesuai di hapus dari *bucket* Google *cloud storage*. Dengan menggunakan program ini, aplikasi dapat mengelola *file* secara efisien dengan menggunakan layanan penyimpanan *cloud*, dalam penelitian ini Google *Cloud Storage*. Penggunaan *cloud storage* ini membentuk dalam memastikan keandalan, skalabilitas, dan ketersediaan data, serta tidak perlu menghawatirkan tentang infrastruktur penyimpanan.

API yang telah dibuat, memerlukan *deployment* agar API dipublikasikan kedalam lingkungan produksi sehingga dapat diakses oleh pengguna aplikasi untuk digunakan. Google *cloud platform* memiliki salah satu layanan komputasi yang dapat melakukan membangun dan *deploy* aplikasi dengan mudah yaitu *App engine*. Google *app engine* merupakan layanan PaaS yaitu *platform as a Service* layanan yang menyediakan untuk menjalankan aplikasi, tanpa perlu mengelola infrastruktur fisik.

Proses *deployment* pada API diawali dengan melakukan persiapan program API yang telah dibuat pada penelitian ini. Program API di dokumentasikan pada Github. Dengan dilakukannya dokumentasi yang memudahkan untuk melakukan proses *clone* program API pada *cloud shell editor*. Proses *clone* dilakukan dengan menuliskan perintah ‘git clone’ dengan menambahkan URL sesuai dengan repositori yang ingin di-*clone*. Setelah kode berada pada *cloud shell editor* melakukan konfigurasi aplikasi dengan membuat *file* konfigurasi ‘app.yaml’ yang ditunjukkan pada Gambar 4.10.

app.yaml

```
1  runtime: nodejs18
2  service: default
3  vpc_access_connector:
4    name: projects/befitoutfit/locations/asia-southeast2/connectors/connect-to-sql
```

Gambar 4.10 File app.yaml

Gambar 4.10 merupakan konfigurasi aplikasi yang dilakukan pada *file* app.yaml. *File* konfigurasi berisi definisi suatu pengaturan dari aplikasi. Definisi pengaturan tersebut berisi seperi versi *runtime* yang digunakan pada penelitian ini

yaitu node.js18. service yang digunakan secara *default* dan mendefinisikan *VPC access connector*. Selanjutnya lakukan proses *deployment* yang dilakukan pada terminal *cloud shell* pada direktori projek yang sesuai dengan menuliskan "gcloud app deploy". Setelah selesai proses *deploy* dilakukan akan mendapatkan URL hasil *deployment*.

4.4 Pengujian Fungsionalitas API

Pengujian RESTful API dibuat untuk mengetahui bagaimana hasil dari fungsionalitas API yang dirancang dan diimplementasikan terhadap aplikasi rekomendasi pakaian. Pengujian fungsionalitas dilakukan dengan menggunakan metode *black box*. Pengujian metode *black box* adalah dengan mengamati hasil dari kondisi fitur yang ada pada aplikasi melalui data uji dan pengujian fungsional perangkat lunak seperti yang ditunjukan pada Tabel 4.1.

Tabel 4.1 Pengujian *Black box* modul Pengguna

Nama Pengujian	Kondisi	Hasil	Validasi
Aktivitas Tampilan Utama	1. Setiap item pada tampilan menu ditekan 2. Kedua menu tampilan awal ditekan	1. Tampilan yang ditekan mengarahkan ke aktivitas yang benar 2. Mengarahkan ke aktivitas yang benar seperti *register* dan *login*	1. Valid 2. Valid
Aktivitas Daftar	1. Data terisi Lengkap 2. Terdapat *field* kosong 3. Data yang dimasukkan sudah ada 4. Data *email* tidak sesuai format	1. Data tersimpan ke *database* 2. Tombol *register* tidak bisa di tekan 3. Muncul respon data sudah digunakan 4. Muncul respon data "*email* tidak valid"	1. Valid 2. Valid 3. Valid 4. Valid
Aktivitas Masuk	1. Data yang terisi benar 5. Salah satu dari data yang disi salah	1. Masuk ke aplikasi sukses 5. Muncul respon data masukkan salah	1. Valid 2. Valid
Pembaruan akun penguna	1. Pengguna memasukkan *username* baru 2. Pengguna memasukkan *email* baru	1. *Username* berhasil di perbarui 2. *Email* berhasil di perbarui	1. Valid 2. Valid

Pada Tabel 4.1 merupakan tabel yang berisikan hasil dari pengujian fungsional aplikasi dengan metode *black box* untuk modul pengguna. Pengujian pada modul pengguna dilakukan menggunakan metode *black box* untuk memastikan setiap fitur berfungsi sesuai spesifikasi yang diharapkan. Hasil pengujian menunjukkan bahwa semua skenario yang diuji dinyatakan valid, menandakan bahwa sistem bekerja dengan baik tanpa ditemukan cacat fungsional. Pengujian mencakup berbagai aspek, mulai dari navigasi pada tampilan utama, proses pendaftaran, autentikasi masuk, hingga pembaruan akun pengguna. Analisis mendalam terhadap hasil pengujian menunjukkan keunggulan pada implementasi validasi input, pengalaman pengguna, dan keakuratan sistem dalam merespons berbagai kondisi.

Pada pengujian pertama melakukan pada kondisi tampilan utama, pengujian memastikan bahwa setiap elemen menu berfungsi dengan baik, mengarahkan pengguna ke halaman yang sesuai, seperti *login* dan registrasi. Hasilnya menunjukkan navigasi yang intuitif dan bebas dari *error*, dengan begitu logika navigasi di terapkan dengan benar sehingga pengguna dapat mengakses fitur yang dibutuhkan dengan interaksi yang baik.

Pengujian yang kedua dilakukan untuk kondisi melakukan aktivitas pendaftaran atau registrasi akun melakukan pengujian dengan skenario berbeda, termasuk input data yang lengkap, *field* kosong, dan data yang sudah ada. Sistem berhasil memproses masukan data terhadap aplikasi dengan benar maka hasilnya sistem berhasil menyimpan data yang valid kedalam basis data, *field* kosong atau data yang belum diisi hasilnya sistem akan melakukan pencegahan proses daftar dengan menonaktifkan tombol registrasi dalam aplikasi, dan untuk kondisi data yang terindikasi data duplikasi data sistem akan melakukan tindakan penolakan dengan memberikan pesan kesalahan yang jelas seperti sistem akan memberikan pesan “Data sudah digunakan”. Validasi selanjutnya pada pengujian ini ketika kondisi data *email* pengguna yang dimasukkan tidak sesuai format *email*, maka akan muncul respon atau pesan kesalahan yang menjelaskan bahwa *email* yang dimasukkan tidak valid. Hal ini mencerminkan implementasi validasi masukan data pada sistem registrasi yang kuat.

Pada aktivitas *login* atau aktivitas dalam proses masuk kedalam aplikasi dengan akun pengguna yang sudah di registrasikan, pengujian pada proses ini mencakup skenario data yang benar dan salah. Sistem menunjukkan kinerja autentikasi yang akurat dan baik, dengan memberikan akses untuk data yang benar dan pesan kesalahan untuk data yang salah. Ketika data yang dimasukkan benar maka sistem akan merespon tampilan beranda awal yaitu mengizinkan pengguna masuk sesuai dengan akun yang dimasukkan, namun ketika data yang dimasukkan salah untuk kedua data salah dan salah satunya salah baik data email atau *password* pengguna, maka pengguna tidak dapat masuk kedalam aplikasi. Hasil respons ini mampu meningkatkan kenyamanan bagi pengguna. Dari sisi keamanan, dalam sistem *login* ini dilakukan enkripsi data *password* dengan sistem *hashing* melalui logika aplikasi dengan *bcrypt* untuk mencegah risiko kebocoran informasi sensitif dengan membuat data masukan tidak dapat dibaca walaupun mampu mengaskses *server*.

Pengujian selanjutnya yaitu melakukan pengujian fitur pembaruan akun pengguna, dengan fokus melakukan pada pembaruan data *username* pengguna dan *email* akun pengguna dalam upaya untuk melakukan keamanan serta memperbarui data akun untuk aplikasi. Hasil pengujian menunjukkan bahwa sistem berhasil memperbarui data baik *email* maupun *username* pengguna tanpa *error*, serta mencegah duplikasi data. Dalam memperbarui data akun dari sebelumnya proses masukkan data dengan karakter spesial, juga disarankan untuk memastikan stabilitas sistem.

Secara keseluruhan, pengujian fungsional yang dilakukan dengan metode *blackbox* menunjukkan hasil bahwa modul pengguna berfungsi sesuai kebutuhan fungsional dengan validasi masukkan yang ketat dan pengalaman pengguna yang positif. Meskipun demikian, pengujian lebih lanjut diperlukan dalam penelitian ini seperti pada aspek non-fungsional, seperti performa saat beban tinggi, hasil dari waktu respon yang didapatkan. Mengatasi peluang dari hasil pengujian yang diidentifikasi dan dianalisis, sistem yang dibuat berhasil mencapai tingkat keandalan dan keamanan yang lebih tinggi. Hasil pengujian untuk modul manajemen pakaian yang ditunjukan pada Tabel 4.2

Tabel 4.2 Pengujian *Black Box* Modul Manajemen Pakaian

Nama Pengujian	Kondisi	Hasil	Validasi
Aktivitas Tampilan Beranda	1. Lihat daftar pakaian	1. Respon memberikan daftar pakaian yang telah dimasukan	1. Valid
Menambahkan pakaian	1. menambahkan gambar dan data pakaian 2. terdapat data yang belum terisi	1. data berhasil disimpan 2. aktivitas tidak bisa dilanjutkan	1. Valid
Get Outfit	1. Melihat daftar pakaian yang dimasukkan 2. Ketika token jwt sudah tidak aktif	1. Menampilkan daftar pakaian berserta atribut data 2. Muncul respon *bad request*	1. Valid 2. Valid
Update Outfit	1. memasukkan nama pakaian baru 2. memasukkan tipe pakaian baru	1. nama pakaian berhasil di perbarui 2. tipe pakaian berhasil di perbarui	1. Valid 2. Valid
Menghapus pakaian	1. Memilih pakaian yang ingin di hapus	1. Berhasil menghapus pakaian dari *database*	1. Valid
Menampilkan rekomendasi pakaian	1. Memilih *event* untuk pakaian 2. Memilih *event* pakaian yang belum di masukkan	1. Menampilkan untuk seluruh tipe sesuai dengan *event* 2. Menampilkan hasil kosong pada menu	1. Valid 2. Valid

Pada Tabel 4.2 merupakan pengujian menggunakan metode *Black Box Testing* yang dilakukan pada sistem untuk manajemen pakaian dimulai dari pengujian untuk aktivitas beranda, melakukan *uploud* gambar pakaian kedalam sistem atau *database* dengan penyimpanan *cloud*, menampilkan daftar pakaian yang telah dimasukkan beserta dengan nilai data, melakukan perubahan data pakaian, melakukan hapus pakaian yang telah dimasukkan kedalam sistem aplikasi, dan aktivitas terakhir dari pengujian fungsional untuk modul manajemen pakaian adalah menampilkan hasil rekomendasi pakaian yang dihasilkan dari sistem pada aplikasi. menunjukkan bahwa fitur utama telah berfungsi dengan baik dan sesuai dengan ekspektasi dengan hasil yang valid.

Pada fitur tampilan beranda, sistem mampu menampilkan daftar pakaian yang telah dimasukkan oleh pengguna dengan akurat. Fitur ini sangat penting

karena menjadi gerbang awal bagi pengguna untuk melihat data yang telah pengguna *uploud* dan atur serta disimpan dalam aplikasi. Sistem beroperasi optimal ketika koneksi jaringan dalam kondisi stabil, tetapi waktu respons masih memerlukan optimasi lebih dalam prosesnya terutama dalam kondisi jaringan yang lambat. Selain itu, sistem juga telah menangani skenario kegagalan jaringan dengan cukup baik dengan melalui pesan *error* yang relevan, seperti "Koneksi tidak tersedia." penambahan mekanisme *caching* untuk menyimpan data sementara dapat meningkatkan pengalaman pengguna dalam kondisi jaringan yang kurang kurang baik dalam penggunan aplikasi.

Pengujian selanjutnya pada fitur menambahkan data pakaian kedalam aplikasi, sistem berhasil memastikan validasi *input* dengan baik, sehingga hanya data yang lengkap dan memenuhi kriteria yang dapat disimpan. Pengujian menunjukkan bahwa sistem menolak penyimpanan data jika ada atribut yang kosong atau tidak sesuai format yang ditentukan. Hasilnya sistem akan melakukan pencegahan proses untuk melakukan *uploud* gambar pakaian dengan menonaktifkan tombol *uploud* didalam aplikasi. Sebagai contoh, pengguna yang mencoba mengunggah gambar dengan ukuran *file* yang terlalu besar akan mendapatkan pesan *error* "Ukuran file terlalu besar." Demikian pula, jika format gambar tidak didukung, seperti selain JPG atau PNG, sistem memberikan pesan *error* "Format gambar tidak didukung." Validasi ini tidak hanya menjaga integritas data dalam *database*, tetapi juga melindungi sistem dari potensi gangguan dalam penggunaan aplikasi. Untuk meningkatkan kenyamanan, fitur ini dapat dikembangkan lebih lanjut dengan menyediakan pratinjau gambar sebelum data disimpan.

Pengujian selanjutnya melakukan aktivitas pada Fitur *Get Outfit* yaitu kondisi yang menampilkan daftar dari data pakaian yang telah di simpan sebelumnya seperti gambar pakaian, kebutuhan data seperti tipe pakaian, nama pakaian, dan hasil dari rekomendasi sesuai *event* yang sesuai, dalam proses dan hasilnya menunjukkan performa yang sangat baik dalam memastikan keamanan akses data melalui autentikasi berbasis token JWT. Sistem mampu memverifikasi validasi token dengan akurat dan memberikan respons yang sesuai berdasarkan kondisi token. Ketika token valid, sistem berhasil menampilkan daftar pakaian

beserta data atribut lengkapnya. Namun, ketika token telah melewati batas waktu atau kadaluwarsa, sistem memberikan pesan *error* yang informatif, seperti "Token kedaluwarsa," sehingga pengguna mengetahui alasan kegagalan. Demikian pula, untuk token yang tidak valid, sistem memberikan pesan *error* "Token tidak valid." Hal ini menunjukkan bahwa sistem telah dirancang dengan baik untuk mencegah akses yang tidak sah dari gangguan lain.

Pengujian selanjutnya untuk aktivtias dari Fitur *update outfit.* Mendapatkan hasil yang baik dan memuaskan selama pengujian. Sistem dapat memperbarui data seperti nama dan tipe pakaian dengan cepat dan akurat sesuai dengan yang diinginkan. Selain itu, validasi masukkan memastikan bahwa data yang dimasukkan sesuai dengan format, sehingga mencegah kesalahan seperti memasukkan angka pada kolom nama pakaian. Fitur ini sangat penting untuk memberikan fleksibilitas kepada pengguna dalam mengelola data pakaian mereka agar sesuai dan lebih terbaru untuk melakukan penggunaan dengan aplikasi.

Pengujian dalam aktivitas untuk fitur penghapusan pakaian juga telah diuji dan berfungsi dengan baik. Pengguna dapat dengan mudah memilih pakaian tertentu untuk dihapus, dan sistem berhasil menghapus data tersebut dari *database* penyimpanan *cloud*. Pada skenario khusus, seperti ketika pakaian yang ingin dihapus sedang digunakan dalam rekomendasi, sistem memberikan pesan *error* "Data tidak dapat dihapus karena sedang digunakan." Hal ini memastikan bahwa tidak ada data yang terhapus secara tidak sengaja atau menyebabkan inkonsistensi dalam sistem aplikasi yang dibuat.

Pada pengujian selanjutnya fitur menampilkan rekomendasi pakaian, sistem menunjukkan kemampuannya dalam mencocokkan data pakaian dengan kriteria *event* yang dipilih oleh kebutuhan pengguna. Sistem berhasil menampilkan rekomendasi yang relevan berdasarkan data yang tersedia. Jika tidak ada data yang cocok, sistem memberikan pesan informatif seperti "Tidak ada pakaian yang cocok untuk *event* ini." Pengujian juga menunjukkan bahwa algoritma rekomendasi bekerja dengan baik dalam menangani berbagai kondisi masukan data.

Secara keseluruhan, sistem menunjukkan performa yang baik dan konsistensi untuk penggunaan fitur yang ada pada aplikasi khususnya untuk

modul manajemen pakaian. Validasi *input*, autentikasi token JWT, dan pengelolaan data telah berjalan dengan optimal, mendukung keamanan serta keandalan sistem. Selain itu, sistem juga menunjukkan kemampuan yang baik dalam menangani skenario kegagalan yang terjadi dalam aplikasi, seperti contohnya dalam masukkan data yang tidak valid atau koneksi jaringan bermasalah. Dari hasil pengujian fungsionalitas pada aplikasi yang ada penelitian ini didapatkan hasil sesuai dengan ekspektasi yaitu semuat kondisi dan hasil yang didapatkan berjalan dengan baik dan divalidasi secara valid untuk keadaan fitur dan hasil yang di harapkan. juga penting untuk memastikan sistem tetap stabil dan responsif dalam berbagai kondisi operasional.

4.5 Analisis Performa Implementasi API

Analisis performa dilakukan pada penelitian ini merupakan hal yang penting dilakukan untuk melakukan pengukuran kinerja dari API yang telah di rancang dan diimplementasikan untuk aplikasi rekomendasi pakaian pada penelitian ini. Analisis performa untuk implementasi API pada aplikasi melakukan pengukuran kinerja dari API yang telah dibuat, untuk memastikan program yang diimplementasikan dapat beroperasi dengan baik dan sesuai kebutuhan dari aplikasi. Tujuan utama dari analisis performa pada penelitian ini adalah untuk mengidentifikasi, mengukur, dan memahami kinerja dari program dan sistem yang telah dibuat dengan cara yang terukur.

Proses pengukuran kinerja pada analisis performa dalam implementasi API yaitu dengan melakukan uji coba pada setiap layanan API yang telah dibuat. Uji coba dilakukan menggunakan aplikasi Jmeter. Dalam penelitian ini terdapat delapan program API yang diimplementasikan pada aplikasi. Delapan API tersebut akan dilakukan uji coba dengan menggunakan *Software* Jmeter yaitu dengan melakukan registrasi akun, *login* akun, *update* akun, menambahkan pakaian kedalam penyimpanan, menampilkan *detail* pakaian yang telah dimasukkan oleh pengguna, melakukan *update* data pakaian, menghapus pakaian yang telah dimasukkan penyimpanan, dan menampilkan hasil rekomendasi dari daftar pakaian sesuai dengan pilihan acara yang diinginkan oleh pengguna. Pengukuran kinerja ini melakukan pengumpulan data tentang waktu respon untuk

kecepatan pemprosesan dalam melakukan fitur aplikasi. Hasil uji coba API untuk melakukan pengukuran kinerja API untuk kecepatan dalam pemprosesan pada Tabel 4.3.

Tabel 4.3 Daftar Pengujian Respon

No.	Metode	API	Waktu Respon (detik)	Status
1	POST	*Registrasi*	4,38	Sukses
2	POST	*Login*	8,16	Sukses
3	PUT	*Update User*	0,33	Sukses
4	POST	*Add Outfit*	5.16	Sukses
5	GET	*Get outfit*	0,101	Sukses
6	PUT	*Update Outfit*	0,496	Sukses
7	DELETE	*Delete outfit*	0,585	Sukses
8	GET	*Get recomendation*	0.103	Sukses

Tabel 4.3 merupakan hasil dari uji coba yang dilakukan pada tiap layanan API yang digunakan untuk memenuhi kebutuhan aplikasi rekomendasi pakaian pada penelitian ini. Hasil uji coba diperoleh hasil waktu respon tiap API, semakin cepat dan baik API yang di uji coba maka layanan API yang dibuat dapat dikatakan baik untuk implementasi pada aplikasi untuk penelitian ini. Uji coba pertama yang dilakukan yaitu API registrasi, API registrasi ini menggunakan metode *post* yang digunakan untuk mengirimkan data dari *client* untuk diproses pada *server*. API registrasi pada uji coba didapatkan waktu respon sebesar 4,38 detik, sehingga waktu tersebut dapat dikatakan cukup memakan waktu. Waktu yang diperoleh cukup lama, yang disebabkan karena proses program yang memiliki banyak tahapan. Salah satunya *hashing password* untuk melakukan keamanan pada *password*. Proses *hash* yaitu mengubah *password* yang dimasukkan menjadi nilai *hash* menggunakan fungsi *hash* kriptografis. Pada proses *hashing* ini terdapat nilai *cost factor*, semakin banyak iterasi enkripsi yang dilakukan membuat proses *hashing* ini semakin lebih lambat dan membutuhkan waktu lebih banyak. Namun kelebihannya semakin besar nilai *cost factor* atau

semakin lambat proses *hash* dilakukan keamanan data akan semakin tinggi atau semakin baik.

Uji coba yang kedua pada Tabel 4.1 merupakan uji coba untuk API *login* dengan metode POST. Hasil dari API *login* ini mendapatkan waktu respon sebesar 8,16 detik untuk pengguna masuk ke dalam aplikasi dengan fitur *login*. Waktu respon yang didapatkan cukup terbilang lambat karena pada layanan API *login* ini memiliki langkah yang kompleks seperti memvalidasi data masukan yang berupa email dan *password* pengguna, pencarian pengguna yang sesuai pada *database*, memverifikasi kesesuaian *password* yang dimasukkan dengan yang ada pada *database*, dan terakhir melakukan pembuatan token JWT yang digunakan untuk melakukan otoritas kepada pengguna untuk mengakses tindakan tertentu.

Uji coba yang ketiga adalah uji coba untuk layanan API *update user* yaitu fitur untuk pengguna dapat mengelola dan memperbarui informasi akun pribadi. API ini menggunakan metode PUT yaitu metode yang digunakan untuk mengirimkan data lalu agar diperbarui data yang pada *server*. Waktu respon yang didapatkan dari uji coba ini adalah 0,33 detik merupakan waktu yang terbilang cepat dan baik, dan dapat dikatakan memiliki performa yang baik untuk aplikasi.

Layanan API yang diuji coba selanjutnya adalah API untuk manajemen pakaian seperti *add outfit, get outfit, update outfit, delete outfit*, dan *get recomendation*. API *add outfit* ini menggunakan metode POST. Waktu respon yang didapatkan adalah 5,16 detik yang dipengaruhi oleh waktu penyimpanan seperti memasukan data *file*, tipe, dan acara yang dikirimkan ke *cloud storage*.

Layanan API yang diuji selanjutnya adalah API *get outfit* dengan menggunakan metode GET, metode yang digunakan untuk mengambil atau meminta data pada *database* dengan waktu respon 0,101 detik. API selanjutnya adalah *update outfit* dengan metode *put* dan waktu respon 0,496 detik. API *delete outfit* dengan metode *delete* yang digunakan untuk menghapus data file dengan waktu respon 0,585 detik. Waktu respon yang didapatkan dari pengujian rata-rata memiliki waktu yang terbilang cepat dengan waktu di bawah 1 detik. Hasil performa ini sangat baik dan bekerja dengan baik untuk aplikasi. API terakhir merupakan API *get* rekomendasi untuk menampilkan pakaian hasil dari rekomendasi yang sesuai dengan masukan acara bagi pengguna. API *get*

rekomendasi menggunakan metode POST dan mendapatkan waktu yang sangat cepat yaitu 0,103 detik dengan perolehan waktu tersebut performa dari fungsionalitas API dikatakan baik.

Hasil keseluruhan untuk API yang telah diimplementasikan pada penelitian ini rata-rata mendapakatkan hasil perolehan waktu respon yang baik. Walapun terdapat API yang mendapatkan perolehan waktu respon yang terbilang lambat, namun waktu tersebut dapat ditolerir dengan baik. Hasil waktu respon yang cukup lama tersebut dapat dikarenakan oleh beberapa faktor seperti proses *hashing* yang lambat untuk meningkatkan keamanan akun pengguna, penanganan *file* yang besar ataupun koneksi jaringan yang lambat yang menyebabkan keterlambatan antara klien dan *server*.

4.6 Analisis *Load Testing* API

Load Testing pada penelitian ini dilakukan untuk mengukur performa respon yang diberikan sistem dalam *load condition*. pengujian ini dilakukan bertujuan untuk menentukan respon keadaan API ketika beberapa pengguna mengakses aplikasi secara bersamaan. *Load testing* dibutuhkan agar dapat melakukan simulasi akses aplikasi secara simultan atau bersama-sama dan agar memeriksa sebuah sistem yang sedang dikembangkan dapat menangani masalah atau beban yang diujikan yang disesuaikan dengan berapa banyak pengguna. Pengujian dengan cara melakukan *load testing* ini lebih baik dibandingkan dengan harus melakukan langsung dengan mengundang banyak pengguna, sekaligus dalam mengakses sebuah aplikasi yang diuji.

Pengujian *load testing* yang dilakukan penelitian ini tiga nilai penting yang perlu diujikan untuk indikator pengujian yaitu waktu respon dari hasil pengujian yang merupakan ukuran lamanya waktu saat melakukan *request* ke *server* dan kembali merespon, indikator yang kedua adalah *throughput* nilai *throughput* adalah jumlah *request* pengguna yang berhasil diproses *server* dan berhasil di tujuan dalam satuan waktu, dan indikator yang ketiga *Error rate* nilai dari hasil perhitungan persentase atas proses yang tidak berhasil dalam proses melakukan *request* dan *response* berlangsung. Tahap pengujian *load testing* ini dilakukan dengan menggunakan JMeter. JMeter digunakan agar dapat melakukan simulasi

keadaan beban pada *server*, agar dapat menguji performa dan menganalisis performa API yang diujikan secara menyeluruh. Hasil pengujian beban untuk API dari rekomendasi pakaian di tunjukan pada Tabel 4.4.

Tabel 4.4 Hasil Pengujian *Load testing*

API	Sample *user*	Waktu Respon (ms)	Throughput	Error rate
Login	10	15.164	0,4/s	0%
	50	57.149	0,47/s	0%
	100	88.217	0,57/m	0%
Update User	10	220	3,5/s	0%
	50	198	16,2/s	0%
	100	242	31.5/s	0%
Add Outfit	10	1019	2.8/s	0%
	50	2478	9.4/s	0%
	100	2700	19.2/s	0%
Get Outfit	10	265	3.4/s	0%
	50	341	15.5/s	0%
	100	463	29.9/s	0%
Get Recommendation	10	185	3.5/s	0%
	50	187	16.5/s	0%
	100	280	31.9/s	0%
Update Outfit	10	223	3.4/s	0%
	50	242	15.9/s	0%
	100	380	29.4/s	0%

Pengujian *Load testing* yang ditunjukan Tabel 4.4 pada penelitian ini tidak melakukan pengujian untuk API registrasi dan *Delete Outfit*, hal tersebut dikarenakan pada API registrasi memiliki sistem penggunaan masukkan data yang unik sehingga tidak dapat dilakukan pengujian berulang dengan data yang sama. API *delete* juga tidak bisa dilakukan pengujian *load testing* karena sistem hapus *outfit* yang ada pada aplikasi dilakukan dengan cara satu per satu gambar yang dimasukan pengguna. Berdasarkan hasil dari pengujian *load testing* yang ditampilkan pada Tabel 4.4 didapatkan hasil performa dari enam *endpoint* API yang dibuat pada penelitian ini. Ketiga nilai indikator yang diujikan saling berkaitan dengan satu sama lain terhadap beban *request* dari beberapa jumlah *sample* pengguna yang diujikan. Pada indikator pertama yaitu waktu respon

didapatkan hasil yang menunjukan rata-rata dari hasil waktu respon dalam satuan detik waktu respon dalam satuan waktu *milisecond* (ms) berbading lurus dengan jumlah *sample* pengguna untuk beban *request*, semakin besar beban atau semakin banyak sample pengguna maka rata-rata waktu respon akan semakin meningkat.

Nilai indikator kedua dari pengujian ini adalah *Throughput* dalam pengujian ini menampilkan hasil penanganan pengiriman data dalam kondisi beban berat. Semakin tinggi nilai dari *Throughput*, maka akan semakin baik kinerja proses API karena tandanya data yang berhasil terkirim lebih baik. Pada indikator *throughput* dari hasil pengujian dalam penelitian ini menujukan semakin besar beban *request* atau banyaknya *sample* pengguna maka waktu proses meningkat hal ini mengakibatkan beban semakin meningkat menyebabkan performa API menurun. Pada pengujian untuk API login di dapatkan hasil waktu yang cukup lama dan hasil *throughput* yang cukup kecil, karena dalam logika yang dibuat pada API login memiliki proses *hashing* yang membuat waktu proses menjadi cukup lama. namun semua *request* masih bisa ditangani hingga sebanyak 100 pengguna yang ditunjukan pada *error rate* mendapatkan nilai sebesar 0% pada setiap proses.

Berdasarkan hasil *load testing* pada Tabel 4.4, sistem menunjukkan tingkat stabilitas yang sangat baik dengan *error rate* 0% untuk semua API yang diuji, meskipun beban pengguna bertambah hingga 100 pengguna. Ini menandakan bahwa sistem secara umum telah dirancang dengan baik untuk menangani beban kerja tanpa menghasilkan kesalahan. Namun, terdapat variasi yang signifikan dalam performa waktu respon dan *throughput* di antara API, yang menunjukkan bahwa beberapa aspek sistem masih memerlukan optimasi untuk meningkatkan efisiensi dan kecepatan.

API *Login* memiliki waktu respons yang meningkat secara signifikan dari 15.164 ms untuk 10 pengguna menjadi 88.217 ms untuk 100 pengguna, lebih dari lima kali lipat kenaikan. Peningkatan ini masih tergolong wajar untuk proses yang melibatkan autentikasi, tetapi *throughput* yang rendah, hanya mencapai 0.57 permintaan per detik pada 100 pengguna, menunjukkan adanya kemungkinan *bottleneck*. Masalah ini kemungkinan besar disebabkan oleh *overhead* pada proses enkripsi seperti *hashing password*, atau kompleksitas *query database* yang

digunakan untuk memverifikasi kredensial pengguna. Untuk meningkatkan performa API ini, beberapa langkah dapat diambil, seperti memanfaatkan *caching* untuk token autentikasi.

API *Update User* menunjukkan performa yang stabil dan efisien. Waktu *respons* untuk API ini tetap berada di kisaran 200-242 ms meskipun jumlah pengguna meningkat hingga 100. *Throughput* juga mengalami peningkatan yang signifikan, dari 3.5 permintaan per detik untuk 10 pengguna menjadi 31.5 permintaan per detik untuk 100 pengguna, menandakan bahwa API ini memiliki kemampuan skalabilitas yang baik. Hal serupa juga terjadi pada *Update Outfit* API, yang menunjukkan waktu respons yang hanya sedikit meningkat dari 223 ms menjadi 380 ms, dengan *throughput* yang mencapai hampir 30 permintaan per detik pada 100 pengguna. Kedua API ini telah dioptimalkan dengan baik.

API *Get Recommendation* menonjol sebagai salah satu API dengan performa terbaik dalam pengujian ini. Dengan waktu respons yang tetap di bawah 300 ms bahkan pada beban 100 pengguna, API ini membuktikan bahwa proses rekomendasi telah diimplementasikan secara efisien. *Throughput* juga meningkat secara linear, mencapai 31.9 permintaan per detik untuk 100 pengguna, menunjukkan bahwa API ini mampu menangani beban kerja yang besar tanpa penurunan kinerja. Kecepatan dan stabilitas ini mungkin dihasilkan oleh penggunaan *caching* yang efisien atau algoritma rekomendasi yang telah dirancang untuk berjalan dalam memori.

API *Get Outfit* juga menunjukkan performa yang baik dengan waktu respon yang tetap berada di bawah 500 ms bahkan pada 100 pengguna. *Throughput* API ini juga menunjukkan peningkatan yang konsisten, dari 3.4 permintaan per detik untuk 10 pengguna menjadi 29.9 permintaan per detik untuk 100 pengguna. Namun, dibandingkan dengan *Get Recommendation* API, waktu respons untuk *Get Outfit* API sedikit lebih tinggi, yang mungkin disebabkan oleh volume data yang lebih besar atau kebutuhan untuk mengakses data tambahan dari *database*. Langkah optimasi yang dapat dilakukan meliputi penggunaan *caching* atau pengurangan volume data yang diambil dengan menerapkan teknik seperti *pagination*.

API *Add Outfit* menunjukkan performa yang paling bermasalah dalam pengujian ini. Waktu respons untuk API ini sangat tinggi, mencapai 2700 ms pada 100 pengguna, menunjukkan adanya *bottleneck* yang signifikan. Meskipun *throughput* meningkat dari 2.8 permintaan per detik untuk 10 pengguna menjadi 19.2 permintaan per detik untuk 100 pengguna, waktu respons yang lambat membuat API ini kurang efisien dibandingkan API lain. Permasalahan ini kemungkinan besar disebabkan oleh proses berat seperti penyimpanan *file* atau interaksi dengan sistem I/O. Operasi tulis ke *database* yang kompleks atau pengunggahan *file* besar juga dapat menjadi penyebab waktu respons yang lambat.

Secara keseluruhan, hasil pengujian menunjukkan bahwa sistem memiliki stabilitas yang sangat baik dengan kemampuan untuk menangani peningkatan beban tanpa kesalahan. Namun, terdapat ruang untuk performa, terutama pada *Add Outfit* API, yang saat ini menjadi titik lemah dalam sistem. Selain itu, API lainnya seperti *Login* juga dapat ditingkatkan untuk memperbaiki *throughput* dan waktu respon agar lebih optimal. API seperti *Get Recommendation, Update User,* dan *Update Outfit* menunjukkan performa yang sangat baik, langkah-langkah optimasi yang tepat, sistem dapat menjadi lebih efisien dan mampu melayani kebutuhan pengguna dengan lebih baik.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan dan implementasi penggunaan API dengan sistem *cloud storage* pada penelitian ini, didapat sebagai berikut:

1. Rancangan yang dihasikan pada penelitian ini berdasarkan kebutuhan fitur dari aplikasi rekomendasi pakaian. Pembuatan basis data yang dibutuhkan dan juga penggunaan modul dan *package* untuk melakukan implementasi pada API. Implementasi dilakukan dengan melakukan pembuatan layanan API sesuai kebutuhan dari aplikasi untuk dapat melakukan fitur dengan baik.
2. Hasil dari fungsionalitas untuk aplikasi sesuai dengan yang diharapkan yaitu fitur dapat bekerja dengan baik. Penerapan integrasi antara sistem *cloud* dengan layanan API yang dibuat meringankan beban dalam penyimpanan karena menggunakan penyimpanan *cloud* dalam melakukan *input* data pengguna.
3. Hasil uji coba layanan API yang telah diimplementasikan ini rata-rata mendapatkan hasil waktu respon yang cepat di bawah 1 detik. Walaupun terdapat API yang mendapatkan perolehan waktu lambat lebih dari 5 detik. Hasil waktu respon tersebut disebabkan oleh faktor seperti proses *hashing* yang lambat untuk meningkatkan keamanan akun pengguna, dan penanganan *file*. Hasil dari *load testing* semakin besar beban *request* maka waktu proses semakin meningkat, meningkat beban menyebabkan performa API menurun dan untuk semua *request* dapat ditangani.

5.2 Saran

Adapun saran yang dapat diberikan dari hasil implementasi API ini antara lain sebagai berikut:

1. Meningkatkan spesifikasi sistem *cloud* yang digunakan untuk melakukan proses berjalannya fitur aplikasi.
2. Melakukan pengembangan API yang lebih optimal, dengan membuat logika bisnis lebih efisien.

DAFTAR PUSTAKA

[1] H. A, S. Ayu Ashari, R. R. Taufik Bau, dan S. Suhada, "Eksplorasi Iintensitas Penggunaan Sosial Media (Studi Deskriptif pada Mahasiswa Teknik Informatika UNG)," *Journal of Information Technology Education*, vol. 3, no. 2, 2023, [Daring]. Tersedia pada: http://ejurnal.ung.ac.id/index.php/inverted

[2] N. Arsita dan V. F. Sanjaya, "Pengaruh Gaya Hidup dan Trend Fashion Terhadap Keputusan Pembelian Online Produk Fashion pada Media Sosial Instagram," *Jurnal Ilmu Manajemen Saburai*, vol. 07, no. 02, 2021.

[3] N. Aninda dan Y. Y. Sunarya, "Siklus Tren Fashion di Media Sosial (Studi Kasus Tren Berkain di Instagram Remaja Nusantara)," *Jurnal Seni dan Reka Rancang*, vol. 6, no. 1, 2023, doi: 10.25105/jsrr.v6i1.16961.

[4] M. Mukhtar, "Tata Cara Berpakaian dapat Mempengaruhi Perkembangan Jiwa Anak," *Educandum*, vol. 8, no. 2, 2022, [Daring]. Tersedia pada: https://id.m.wikipedia.org

[5] R. Mudiawati, S. Martus, S. Nur, S. Nurhayati, dan I. Ridwan Yusup, "Penggunaan Outfit Terhadap Rasa Percaya Diri Mahasiswa Pendidikan Semester 7," *Jurnal Psikologi Islam Al-Qalb*, vol. 11, no. 2, 2020.

[6] P. Gazzola, E. Pavione, R. Pezzetti, dan D. Grechi, "Trends in the fashion industry. The perception of sustainability and circular economy: A gender/generation quantitative approach," *Sustainability (Switzerland)*, vol. 12, no. 7, hlm. 1–19, Apr 2020, doi: 10.3390/su12072809.

[7] Z. Al-Halah dan K. Grauman, "From Paris to Berlin: Discovering Fashion Style Influences Around the World," dalam *CVPR*, 2020. [Daring]. Tersedia pada: https://bit.ly/3dBAQ5W

[8] D. Siswanto, Zamzami, L. Nijal, S. Rajab, dan S. Ridar Wilis Rambe, "Aplikasi Rekomendasi Dalam Pemilihan Buku Siswa di Perpustakaan Menggunakan Metode Collaborative Filtering pada SMKN 2 Mandau Berbasis Web," *Jurnal Sistem Informasi*, vol. 4, no. 1, 2022.

[9] A. I. Putra dan R. R. Santika, "Implementasi Machine Learning dalam Penentuan Rekomendasi Musik dengan Metode Content-Based Filtering,"

(Edumatic) Jurnal Pendidikan Informatika, vol. 4, no. 1, 2020, doi: 10.29408/edumatic.v4i1.2162.

[10] J. Senanayake, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, dan L. Piras, "Android Source Code Vulnerability Detection: A Systematic Literature Review," *ACM Comput Surv*, vol. 55, no. 9, Jan 2023, doi: 10.1145/3556974.

[11] S. L. Bangare, S. Gupta, M. Dalal, dan A. Inamdar, "Using Node.Js to Build High Speed and Scalable Backend Database Server," *IJRAT*, 2016, [Daring]. Tersedia pada: www.ijrat.org

[12] P. Taylor, "Number of Smartphone Mobile Network Wubscriptions Worldwide," Statista. Diakses: 26 Maret 2024. [Daring]. Tersedia pada: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[13] "Mobile Operating System Market Share Worldwide," StatCounter. Diakses: 26 Februari 2024. [Daring]. Tersedia pada: https://gs.statcounter.com/os-market-share/mobile/worldwide

[14] A. Wijoyo, A. R. Silalahi, A. Raihan, P. Arrasyid, dan R. Diana, "Sistem Informasi Manajemen Berbasis Cloud," *Jurnal Teknologi, Bisnis dan Pendidikan (TEKNOBIS)*, vol. 1, no. 2, hlm. 1–15, 2023, [Daring]. Tersedia pada: https://jurnalmahasiswa.com/index.php/teknobis

[15] D. Nafis Alfarizi dan I. Heidiani Ikasari, "Tinjauan Literatur Terhadap Pemanfaatan Cloud Computing," *JURIHUM: Jurnal Inovasi dan Humaniora*, vol. 01, no. 01, hlm. 148–154, 2023, [Daring]. Tersedia pada: https://jurnalmahasiswa.com/index.php/jurihum

[16] I. Barokah dan Asriyanik, "Analisis Perbandingan Serverless Computing Pada Google Cloud Platform," *Jurnal Teknologi Informatika dan Komputer*, vol. 7, no. 2, hlm. 169–187, Sep 2021, doi: 10.37012/jtik.v7i2.662.

[17] Hasanuddin, H. Asgar, dan B. Hartono, "Rancang Bangun RESTAPI Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan," *JINTEKS (Jurnal Informatika Teknologi dan Sains)* , vol. 1, no. 4, hlm. 8–14, 2022.

[18] I. O. Suzanti, N. Fitriani, A. Jauhari, dan A. Khozaimi, *REST API Implementation on Android Based Monitoring Application*, vol. 1569, no. 2. IOP Publishing Ltd, 2020. doi: 10.1088/1742-6596/1569/2/022088.

[19] Praveen Borra, "A Survey of Google Cloud Platform (GCP): Features, Services, and Applications," *International Journal of Advanced Research in Science, Communication and Technology*, vol. 4, no. 3, hlm. 191–199, Jun 2024, doi: 10.48175/ijarsct-18922.

[20] I. Barokah dan A. Asriyanik, "Analisis Perbandingan Serverless Computing Pada Google Cloud Platform," *Jurnal Teknologi Informatika dan Komputer*, vol. 7, no. 2, hlm. 169–187, Sep 2021, doi: 10.37012/jtik.v7i2.662.

[21] L. Christiani, "Peluang dan Tantangan Penerapan Cloud Computing (Komputasi Awan) Sebagai Solusi Automasi Kerjasama Antar Perpustakaan," *ANUVA*, vol. 2, no. 1, hlm. 43–53, 2018.

[22] I. N. 'Abidah, M. A. Hamdani, dan Y. Amrozi, "Implementasi Sistem Basis Data Cloud Computing pada Sektor Pendidikan," *KELUWIH*, vol. 1, no. 2, hlm. 77–84, Agu 2020, doi: 10.24123/saintek.v1i2.2868.

[23] M. A. Kamal*, H. W. Raza, M. M. Alam, dan M. M. Su'ud*, "Highlight the Features of AWS, GCP and Microsoft Azure that Have an Impact when Choosing a Cloud Service Provider," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 5, hlm. 4124–4132, Jan 2020, doi: 10.35940/ijrte.D8573.018520.

[24] N. Ramsari dan A. Ginanjar, "Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform," *SENATIK*, vol. 7, no. 1, 2022, doi: 10.28989/senatik.v7i1.472.

[25] E. Riana, "Implementasi Cloud Computing Technology dan Dampaknya Terhadap Kelangsungan Bisnis Perusahaan Dengan Menggunakan Metode Agile dan Studi Literatur," *JURIKOM (Jurnal Riset Komputer)*, vol. 7, no. 3, hlm. 439, Jun 2020, doi: 10.30865/jurikom.v7i3.2192.

[26] W. N. Suliyanti, "Studi Literatur Basis Data Sql dan NoSQL," *Jurnal Kilat*, vol. 8, no. 1, 2019.

[27] M. Kholil dan S. Mu'min, "Pengembangan Private Cloud Storage sebagai Sentralisasi Data Universitas Nahdlatul Ulama Sidoarjo Berbasis Open Source Owncloud," *Jurnal Ilmu Komputer dan Desain Komunikasi Visual*, vol. 3, no. 1, 2018.

[28] I. Kurniawan, Humaira, dan F. Rozi, "Rest Api Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," *JItsi*, vol. 1, no. 4, hlm. 127–132, 2020, [Daring]. Tersedia pada: http://jurnal-itsi.org

[29] Hassanudin, H. Asgar, dan B. Hartono, "Rancang Bangun Rest Api Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan," *Jinteks*, vol. 4, no. 1, hlm. 8–14, 2022.

[30] R. Sulistyo, A. Erlansari, dan F. Farady Coastera, "Aplikasi Cloud SQL Berbasis Web," *Jurnal Rekursif*, vol. 5, no. 1, 2017, [Daring]. Tersedia pada: http://ejournal.unib.ac.id/index.php/rekursif/75

[31] A. Luiz Florencio Matias, D. Costa Rodrigues, G. da Silva Mendes, A. Maria de Souza Rocha, dan J. Roberto de Lima, "System for Counting Hours of Professional Contextualisation," *EnGaTac*, vol. 1, no. 1, 2024.

[32] M. D. Akbar dan A. Antoni, "Aplikasi Absensi Pegawai pada Dinas Komunikasi dan Informatika Kabupaten Deli Serdang dengan QR Code Menggunakan Algoritma Bcrypt," *Sudo Jurnal Teknik Informatika*, vol. 1, no. 1, hlm. 8–16, Mar 2022, doi: 10.56211/sudo.v1i1.2.

[33] T. P. Batubara, S. Efendi, dan E. B. Nababan, "Analysis Performance BCRYPT Algorithm to Improve Password Security from Brute Force," dalam *Journal of Physics: Conference Series*, IOP Publishing Ltd, 2021. doi: 10.1088/1742-6596/1811/1/012129.

[34] E. Gulo dan I. Ferdiansyah, "Pengujian Performa Aplikasi E-Commerce Meningkatkan Skalabilitas dan Responsivitas Menggunakan Jmeter," *Kohesi*, vol. 3, no. 12, 2024.

[35] N. Husufa dan I. Prihandi, "Optimizing JMeter on Performance Testing Using the Bulk Data Method," *Journal of Information Systems and Informatics*, vol. 3, no. 1, 2022, [Daring]. Tersedia pada: http://journal-isi.org/index.php/isi

[36] hasanuddin, H. Asgar, dan B. Hartono, “Rancang Bangun REST API Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan,” *JINTEKS*, vol. 4, no. 1, hlm. 8–14, 2022.

[37] I. Ayu Kanindiya Pradnya Paramitha, D. Made Wiharta, dan I. Made Arsa Suyadna, “Perancangan dan Implementasi RESTFUL API pada Sistem Informasi Manajemen Dosen Universitas Udayana,” *Spektrum*, vol. 9, no. 3, 2022.

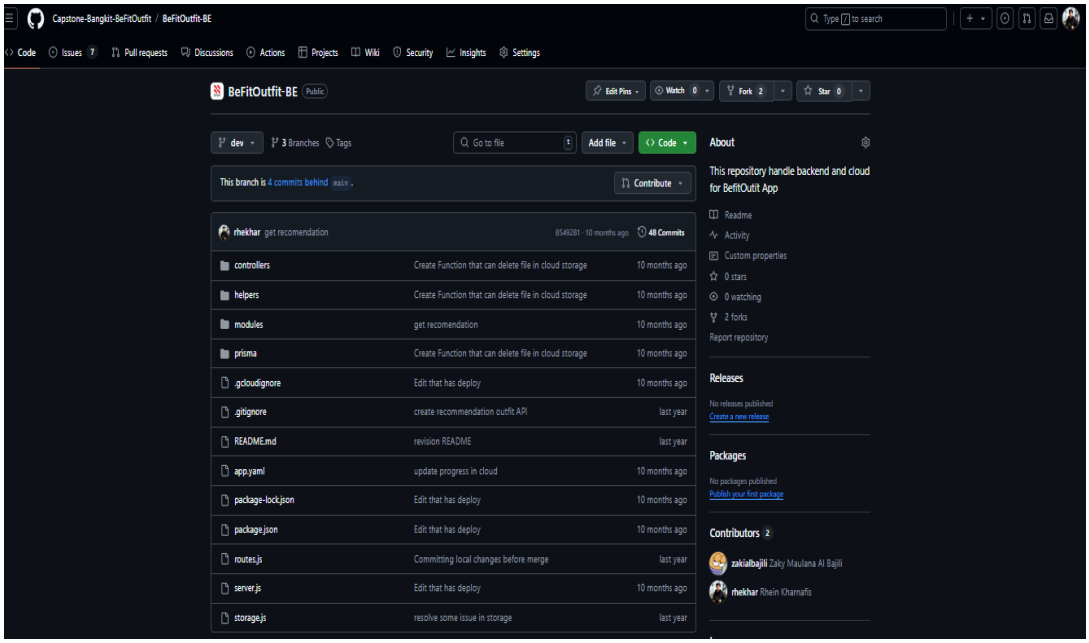
[38] Yannisto dan T. Wibowo, “Perancangan dan Implementasi Storage Berbasis Cloud pada Perusahaan PT Indonesia Weda Bay Industrial Park,” *UIB*, vol. 1, no. 1, 2020, [Daring]. Tersedia pada: http://journal.uib.ac.id/index.php/cbssit

[39] D. Hadi Bachtiar, P. Paniran, dan I. Made Budi Suksmadana, “Perancangan Back-end Api pada Aplikasi Mobile Fruityfit Menggunakan Framework Express JS,” *Mars*, vol. 2, no. 3, hlm. 107–117, 2024, doi: 10.61132/mars.v2i3.138.

[40] A. Mubariz *dkk.*, “Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang),” 2020.

[41] C. Chandra, F. Wijaya, J. A. Gunawan, J. R. Lee, dan A. Maulana, “Perancangan dan Implementasi RESTful API untuk Aplikasi Mobile Pembelajaran Flora dan Fauna pada Google Cloud Platform,” *Satesi*, vol. 4, no. 1, hlm. 58–69, 2024, doi: 10.54259/satesi.v4i1.2850.

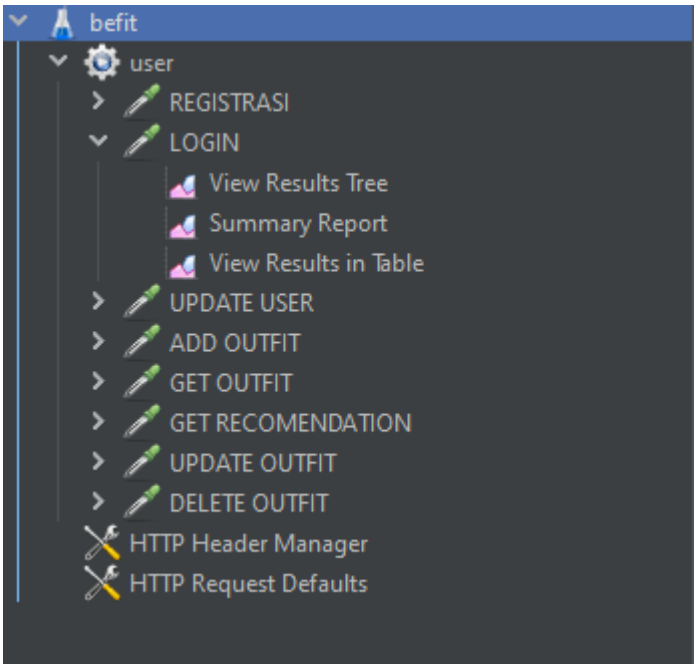
LAMPIRAN A DOKUMENTASI BACKEND APLIKASI



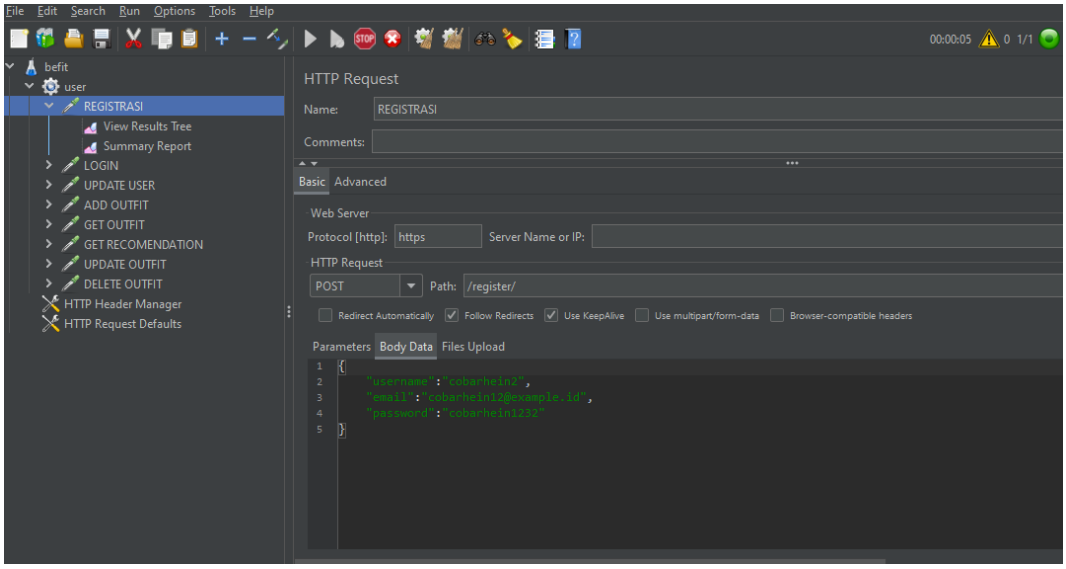
Gambar A. 1 Dokumentasi Back-end aplikasi

https://github.com/Capstone-Bangkit-BeFitOutfit/BeFitOutfit-BE.git

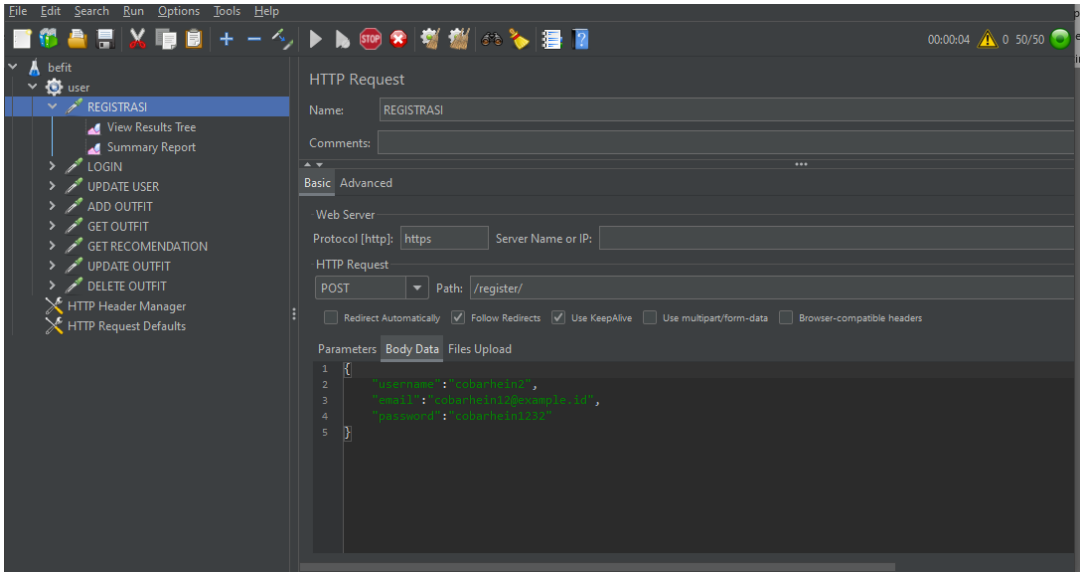
LAMPIRAN B PENGUJIAN DENGAN JMETER



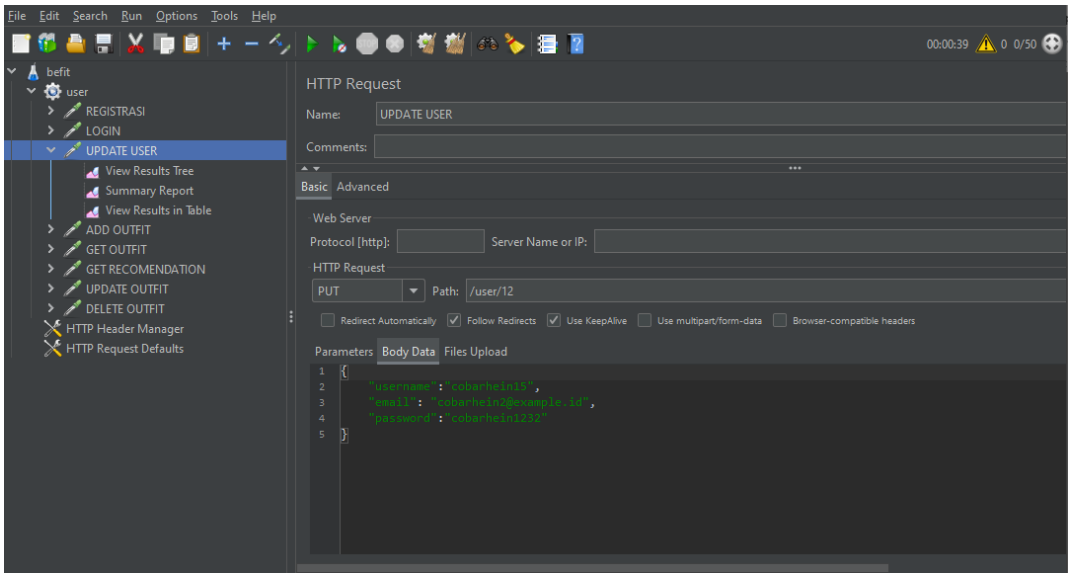
Gambar B. 1 Testing Plan API dengan Jmeter



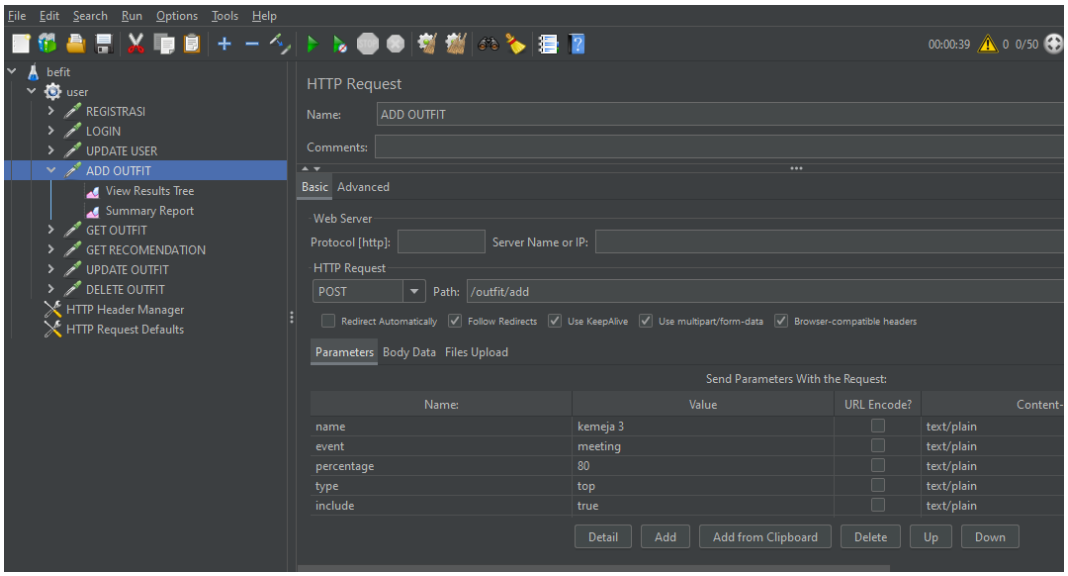
Gambar B. 2 Pengujian API registrasi



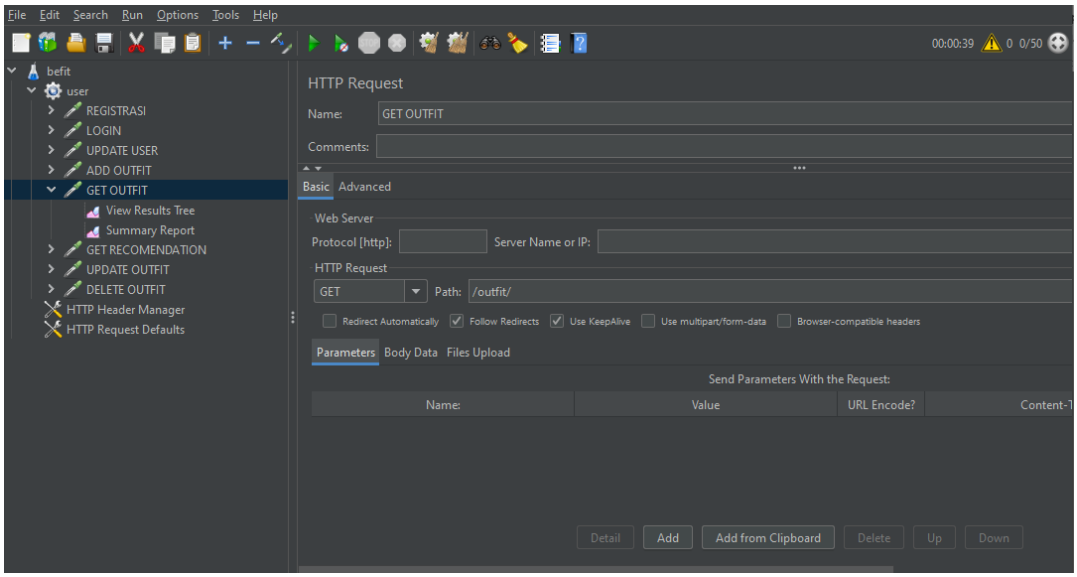
Gambar B. 3 Pengujian API login



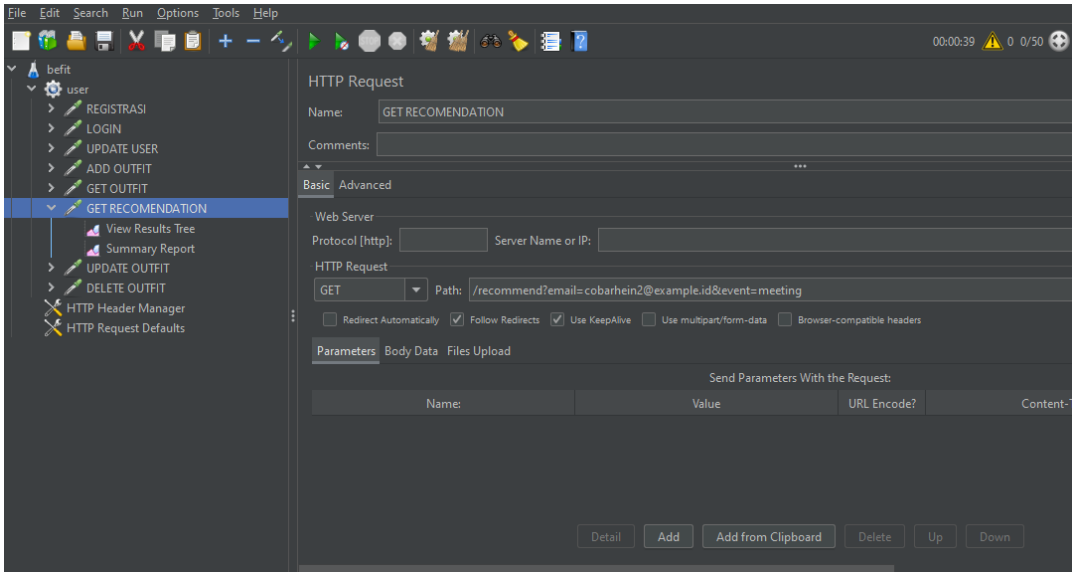
Gambar B. 4 Pengujian API Update User



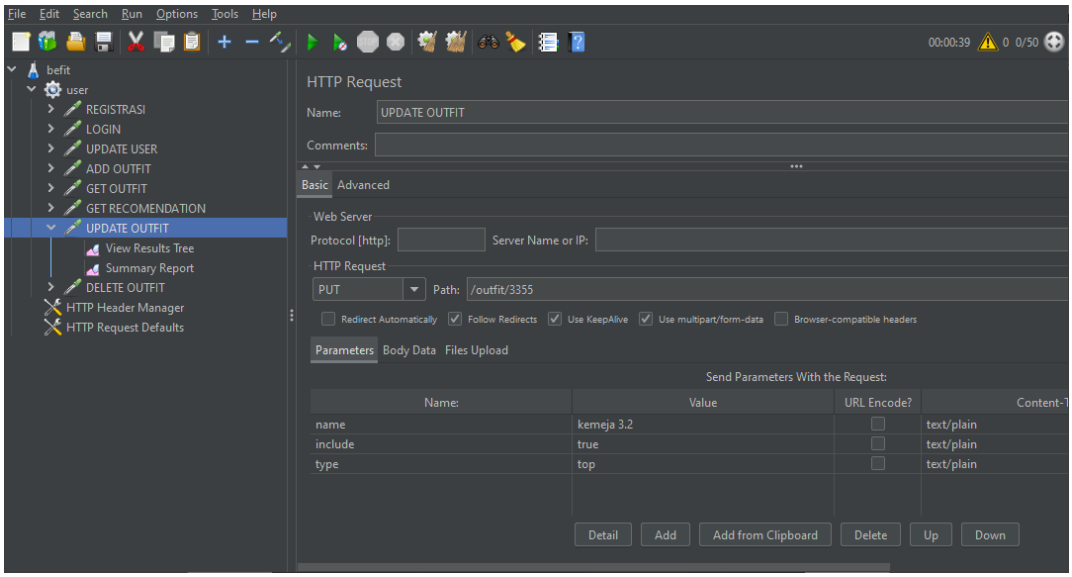
Gambar B. 5 Pengujian Update Outfit



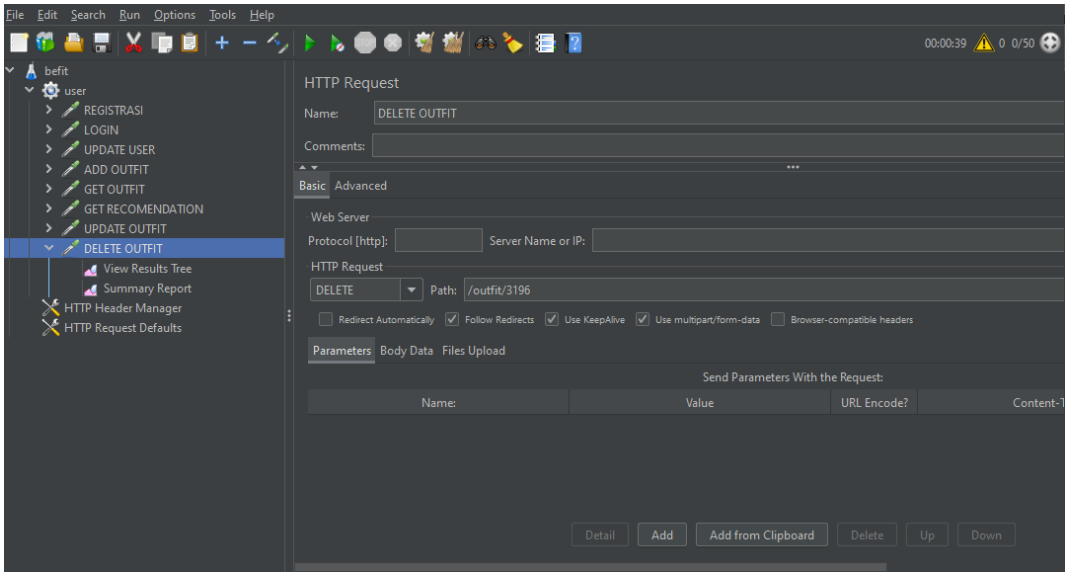
Gambar B. 6 Pengujian Get Outfit



Gambar B. 7 Pengujian Get Recommendation

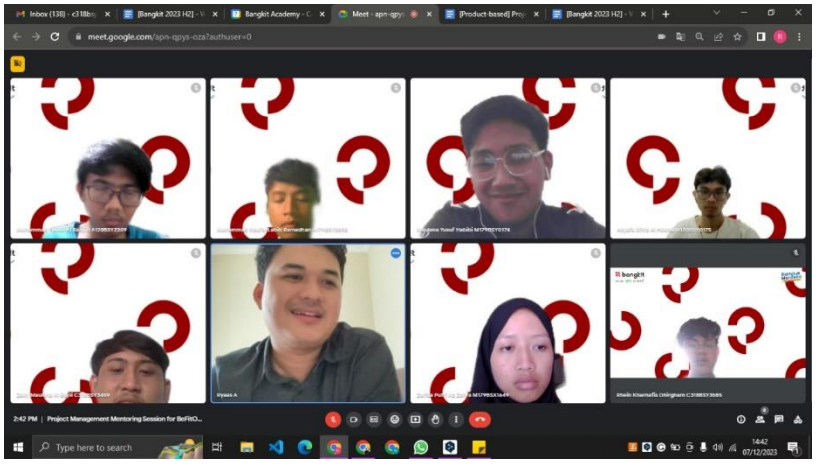


Gambar B. 8 Pengujian Update Outfit

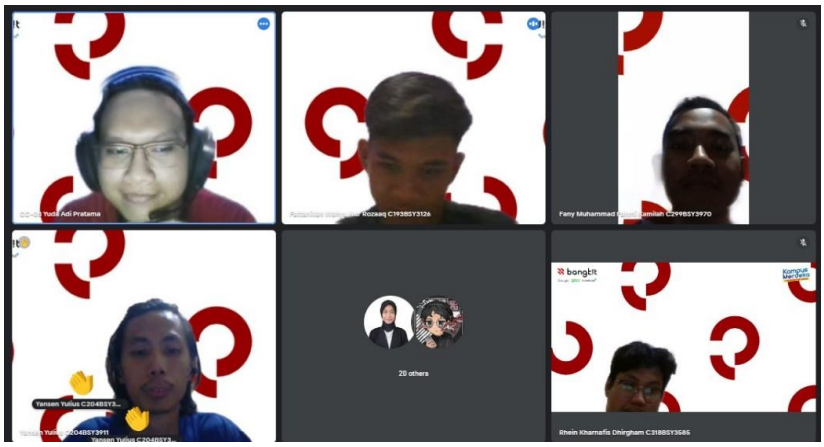


Gambar B. 9 Pengujian Delete Outfit

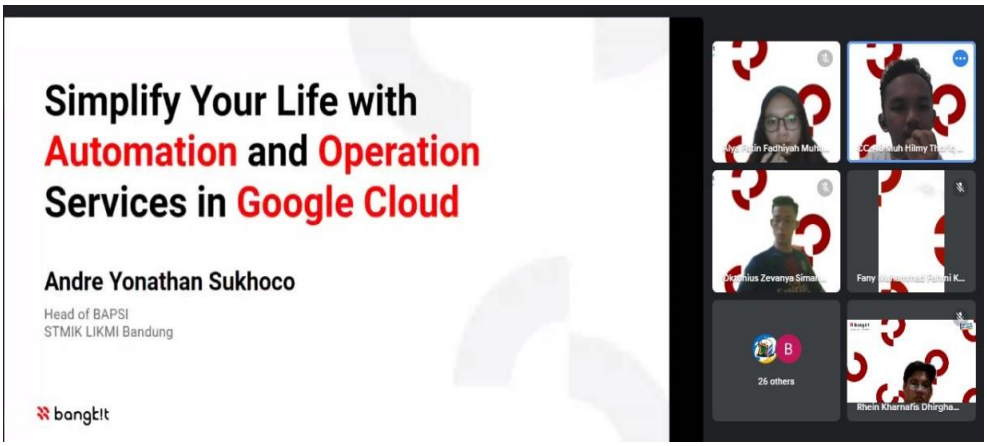
LAMPIRAN C DOKUMETASI KEGIATAN MBKM



Gambar C. 1 Mentoring Project Management untuk Project Capstone



Gambar C. 2 Mentoring mingguan



Gambar C. 3 Instructor-Led Training Cloud Computing

LAMPIRAN D SERTIFIKAT BANGKIT

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI

bangkit

Kampus Merdeka INDONESIA JAYA

BA23/GRAD/XXIV-01/C318BSY3585

Certificate of Completion

is proudly presented to

Rhein Kharnafis Dhirgham

for successfully completing **Bangkit, specializing in Cloud Computing.**

Bangkit is a Google-led academy designed to produce high-caliber technical talent for world-class Indonesian technology companies and startups.

January 19, 2024

Dora S.

Dora Songco

Product Marketing Manager
Google Indonesia

Google goto traveloka

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI

bangkit

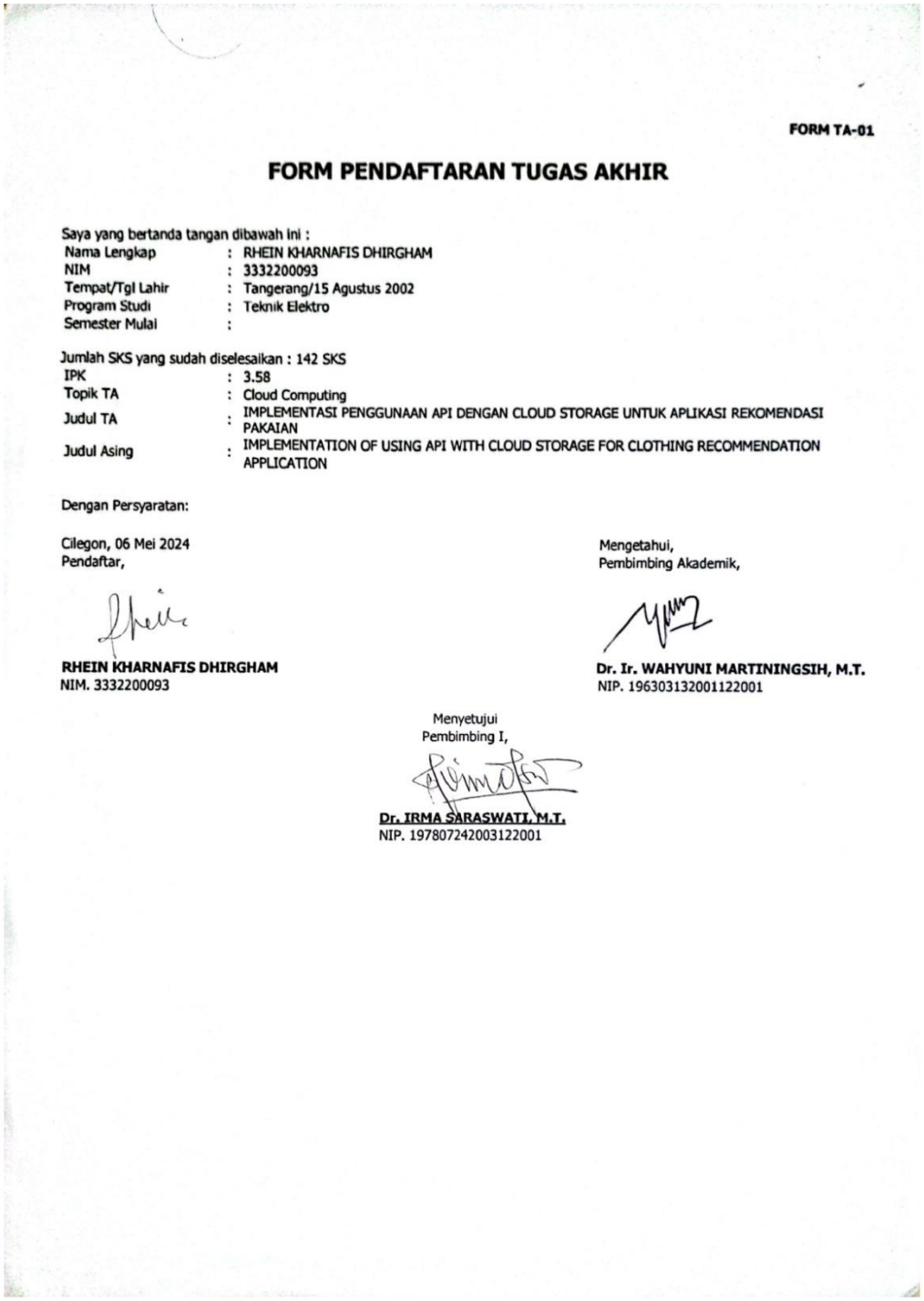
Kampus Merdeka INDONESIA JAYA

STUDENT LEARNING ACHIEVEMENT

Bangkit ID	: C318BSY3585	Bangkit Completion	: Full Graduate
Name	: Rhein Kharnafis Dhirgham	Learning Path	: Cloud Computing
University	: Universitas Sultan Ageng Tirtayasa	Capstone Status	: Finished

No	Courses/Specialization/Activities	Learning Outcomes	Hours	Score (0-100)	Score Description
1	JavaScript Basic	By the end of the course, the student will be able to develop programs with JavaScript using Node.js and Text Editors, namely Visual Studio Code.	45	90	The student is skilled in developing programs with JavaScript using Node.js and Text Editors, namely Visual Studio Code.
2	Web Programming Basic	By the end of the course, the student will be able to develop a simple website using programming code that conforms to global standards.	41	90	The student competently develops a simple website using programming code that conforms to global standards.
3	Intro to Back-End Development using Google Cloud	By the end of the course, the student will be able to build a simple RESTful APIs independently to support the functionality of an application.	45	90	The student is skilled in building simple RESTful APIs independently to support the functionality of an application.
4	Google Cloud Computing Foundations	By the end of the course, the student will be able to apply basic DevOps skills.	43	90.4	The student is adept at applying basic DevOps skills.
5	Google Cloud Engineer Learning Path	By the end of the course, the student will be able to comprehend and operate the essential technology of Google Cloud to become a Cloud Engineer.	117	90.4	The student is knowledgeable to grasp and operate the essential technology of Google Cloud to become a Cloud Engineer.
6	Google Cloud Skills Boost Quests	By the end of the course, the student will be able to apply basic DevOps and machine learning dataset skills.	77	90.4	The student is proficient in applying basic DevOps and machine learning dataset skills.
7	Google IT Support	By the end of the course, the student will be able to comprehend the basics of technology and modern network protocols, the overview of the cloud, practical applications, and network troubleshooting.	30	90.4	The student is adept at comprehending the basics of technology and modern network protocols, the overview of the cloud, practical applications, and network troubleshooting.
8	Becoming a Google Cloud Engineer	By the end of the course, the student will be able to comprehend the cloud concepts and terminology along with the various services on the Google Cloud Platform.	42	87.1	The student is proficient in understanding the cloud concepts and terminology along with the various services on the Google Cloud Platform.
9	Preparing for Associate Cloud Engineer Certification	By the end of the course, the student will be able to comprehend the exam area and learn the Google-recommended references needed to pursue the exam.	33	87.8	The student is adept at comprehending the exam area and studying the Google-recommended references needed to pursue the exam.
10	Capstone / Final Project	By the end of the course, the student will be able to begin stages of a final project, namely developing an application/solution which validates their product development skills and boosts the portofolio.	200	90.5	The student is competent to begin stages of a final project, namely developing an application/solution which validates their product development skills and boosts the portofolio.
11	Soft skill & Career Development	By the end of the course, the student will be able to comprehend Life Path, Growth Mindset, The Power of Feedback, Time Management, Critical Thinking, Problem Solving, Adaptability, Resilience, Project Management, Professional Communication, Networking, Digital Branding, and Interview Communication	249	88.5	The student thoroughly comprehends Life Path, Growth Mindset, The Power of Feedback, Time Management, Critical Thinking, Problem Solving, Adaptability, Resilience, Project Management, Professional Communication, Networking, Digital Branding, and Interview Communication

LAMPIRAN E FORM TA-01



FORM TA-01

FORM PENDAFTARAN TUGAS AKHIR

Saya yang bertanda tangan dibawah ini :

Nama Lengkap	:	RHEIN KHARNAFIS DHIRGHAM
NIM	:	3332200093
Tempat/Tgl Lahir	:	Tangerang/15 Agustus 2002
Program Studi	:	Teknik Elektro
Semester Mulai	:	

Jumlah SKS yang sudah diselesaikan : 142 SKS

IPK	:	3.58
Topik TA	:	Cloud Computing
Judul TA	:	IMPLEMENTASI PENGGUNAAN API DENGAN CLOUD STORAGE UNTUK APLIKASI REKOMENDASI PAKAIAN
Judul Asing	:	IMPLEMENTATION OF USING API WITH CLOUD STORAGE FOR CLOTHING RECOMMENDATION APPLICATION

Dengan Persyaratan:

Cilegon, 06 Mei 2024
Pendaftar,

RHEIN KHARNAFIS DHIRGHAM
NIM. 3332200093

Mengetahui,
Pembimbing Akademik,

Dr. Ir. WAHYUNI MARTININGSIH, M.T.
NIP. 196303132001122001

Menyetujui
Pembimbing I,

Dr. IRMA SARASWATI, M.T.
NIP. 197807242003122001

LAMPIRAN F FORM TA-02

FORM TA-02

FORM BIMBINGAN TUGAS AKHIR

Nama Mahasiswa : RHEIN KHARNAFIS DHIRGHAM
NIM : 3332200093
Program Studi : TEKNIK ELEKTRO - S1 Reguler
Semester : Genap Tahun Akademik 2023/2024
Pembimbing 1 : Dr. . IRMA SARASWATI, S.Si., M.T.

Judul Tugas Akhir:
IMPLEMENTASI PENGGUNAAN API DENGAN SISTEM CLOUD STORAGE UNTUK APLIKASI REKOMENDASI PAKAIAN

No	Tanggal	Topik Pembahasan	Paraf Pembimbing
1	14 September 2023	Melaporkan progres pembelajaran bulanan ke-1	
2	18 Oktober 2023	Melaporkan progres pembelajaran bulanan ke-2	
3	08 Desember 2023	Melaporkan progres pembelajaran bulanan ke-3	
4	02 Februari 2024	Bimbingan konversi dan pembelajaran di Mitra	
5	05 Maret 2024	Bimbingan penulisan laporan MBKM BAB 1-2	
6	15 Maret 2024	Bimbingan Penulisan laporan MBKM BAB 3	
7	21 Maret 2024	Bimbingan Penulisan bagian metodologi untuk pada bab 3	
8	23 April 2024	Bimbingan Penulisan BAB 4 pada sub bab 4.1 hingga 4.4	
9	6 Mei 2024	Bimbingan Penulisan BAB 4 pada sub bab 4.6 hingga 4.7	
10	10 Mei 2024	Bimbingan Penulisan BAB pada sub bab 4.8 hingga 4.9	
11	14 Mei 2024	Bimbingan Penulisan BAB 5	
12	6 Juni 2024	Melakukan pembahasan penulisan dari skrispi mengenai pengujian	
13	31 Juli 2024	Melakukan bimbingan penambahan pengujian API dengan Load Testing	
14	19 Agustus 2024	Melakukan laporan progres terkait penambahan pengujian	

15	3 September 2024	Melaporkan hasil pengujian API yang telah di tambahkan	
16	10 September 2024	Melaporkan pengujian API dengan penulisan pembahasan	
17	27 September 2024	Melakukan bimbingan penulisan kembali pada pengujian load testing	
18	17 Oktober 2024	Melakukan bimbingan penulisan skripsi terkait penambahan pengujian	

Cilegon, 31 Mei 2024
Mahasiswa,

RHEIN KHARNAFIS DHIRGHAM
NIM. 3332200093

Mengetahui,
Pembimbing Akademik,

Dr. Ir. WAHYUNI MARTININGSIH, M.T.
NIP. 196303132001122001

LAMPIRAN G FORM TA-03

FORM TA-03

FORM PENDAFTARAN SIDANG TA

Nama Mahasiswa	:	RHEIN KHARNAFIS DHIRGHAM
NIM	:	3332200093
Program Studi	:	Teknik Elektro
Semester Mulai	:	Tahun Akademik 2023/2024
Topik TA	:	Cloud Computing
Judul Tugas Akhir	:	

IMPLEMENTASI PENGGUNAAN API DENGAN SISTEM CLOUD STORAGE UNTUK APLIKASI REKOMENDASI PAKAIAN

Dengan ini mengajukan untuk pelaksanaan Sidang Ujian Tugas Akhir dengan menyampaikan persyaratan terlampir.

Cilegon, 04 Juni 2024
Mahasiswa,

RHEIN KHARNAFIS DHIRGHAM
NIM 3332200093

Mengetahui,
Pembimbing Akademik

Dr. Ir. WAHYUNI MARTININGSIH, M.T.
NIP 196303132001122001

Menyetujui,
Pembimbing 1

Dr. . IRMA SARASWATI, S.Si., M.T.
NIP 196303132001122001