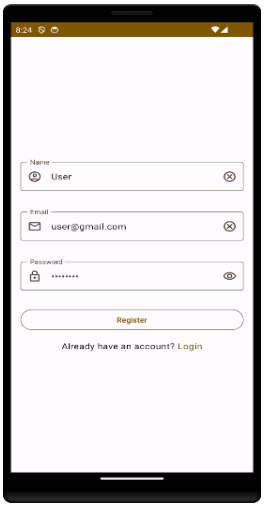
BAB IV
HASIL PENELITIAN

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

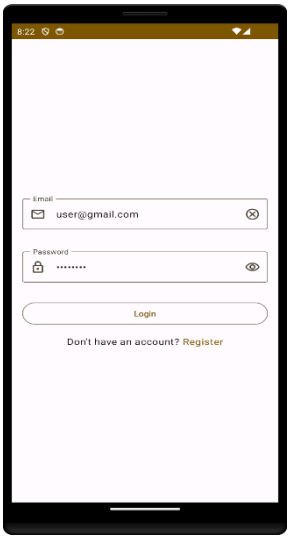
Aplikasi rekomendasi yang ada pada penelitian merupakan aplikasi rekomendasi yang membantu dalam menentukan pakaian. Fitur yang ada dan digunakan dalam aplikasi yaitu pengguna memasukan pakaian-pakaian yang dimiliki, kemudian aplikasi akan menentukan pemilihan pakaian dari atasan maupun bawahan untuk 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.

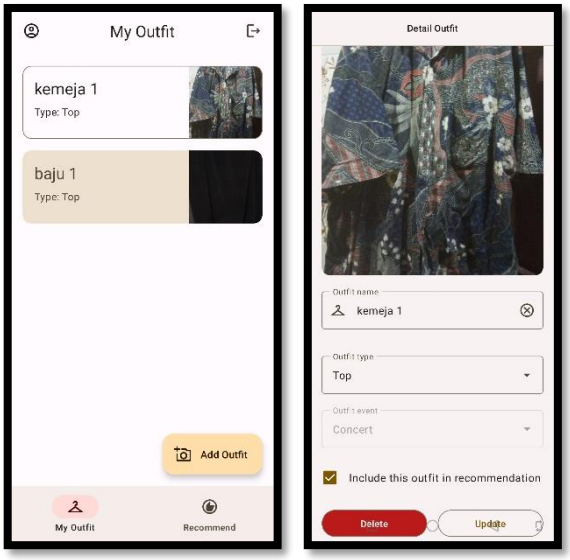


Gambar 4.2 Tampilan Login

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.

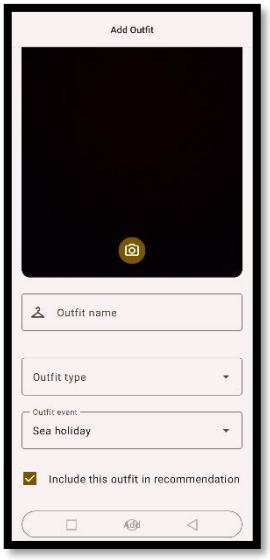
Setelah pengguna melakukan *login* maka akan langsung tampil tampilan etalase pakaian dengan menu *My Outfit.* Tampilan daftar pakaian ini berisi pakaian-pakaian dari lemari pribadi pengguna yang telah dimasukan kedalam aplikasi. Kumpulan baju ini akan digunakan seperti lemari pakaian yang nantinya digunakan 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.



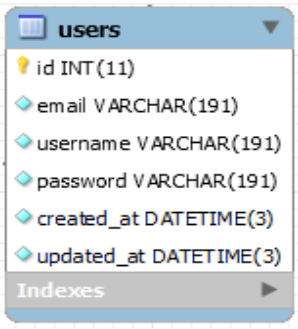
Gambar 4.4 Tampilan Add Outfit

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

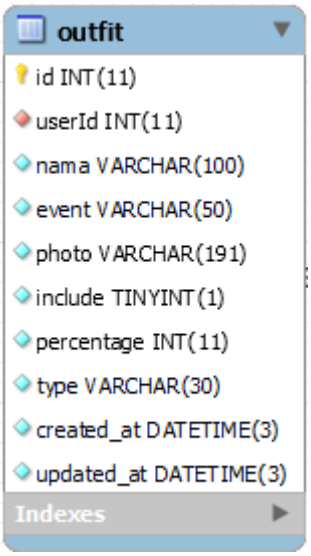
Basis data merupakan hal yang sangat penting dalam melakukan pengembangan aplikasi android. basis data perlu dilakukan karena pada aplikasi perlu merencanakan struktur dan organisasi dari basis data yang akan digunakan. basis data yang terstruktur dapat memudahkan pengembangan aplikasi karena menyediakan 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*

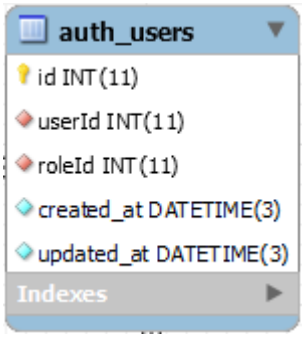
Pada Gambar 4.5 merupakan gambar data untuk *user* atau pengguna dalam basis data. Kolom tabel ini berisikan ID sebagai *primary key*. data *email*, *username*, *password* dengan tipe data *string*. Data *created_at* dan *update_at* dengan tipe data DATETIME. Tabel ini untuk menyimpan data identitas 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*

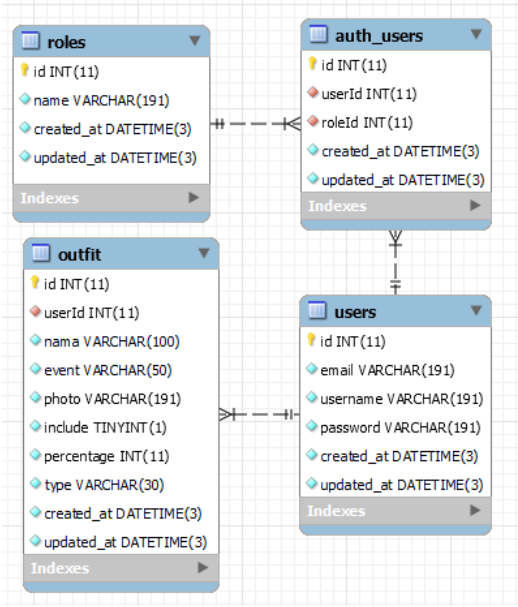
Pada Gambar 4.6 merupakan gambar data untuk *Outfit* atau pakaian pengguna dalam basis data. Kolom tabel ini berisikan id sebagai *primary key* yaitu kunci primer dari salah satu kolom yang unik dan mengidentifikasi setiap baris dalam tabel. *Primary key* penting untuk menjaga keunikan data agar entitas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



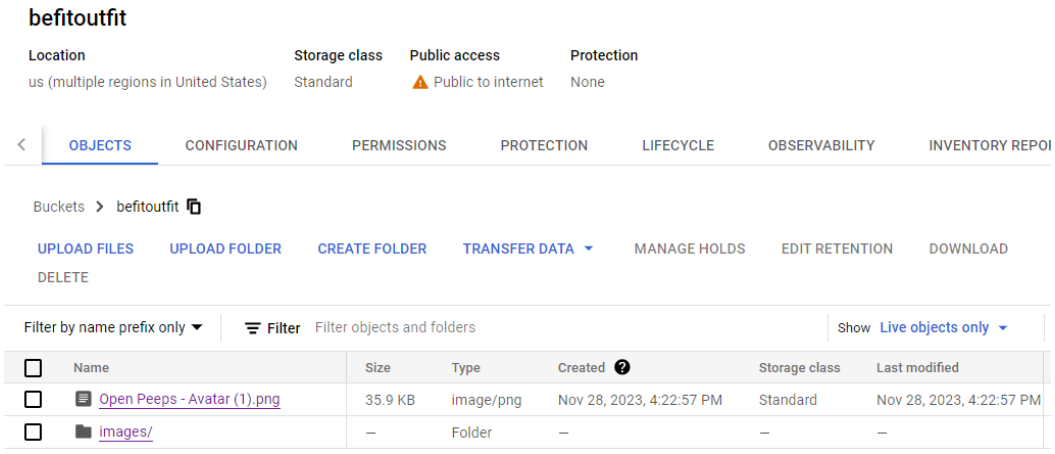
Gambar 4.8 Perancangan Basis Data Aplikasi

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.