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.

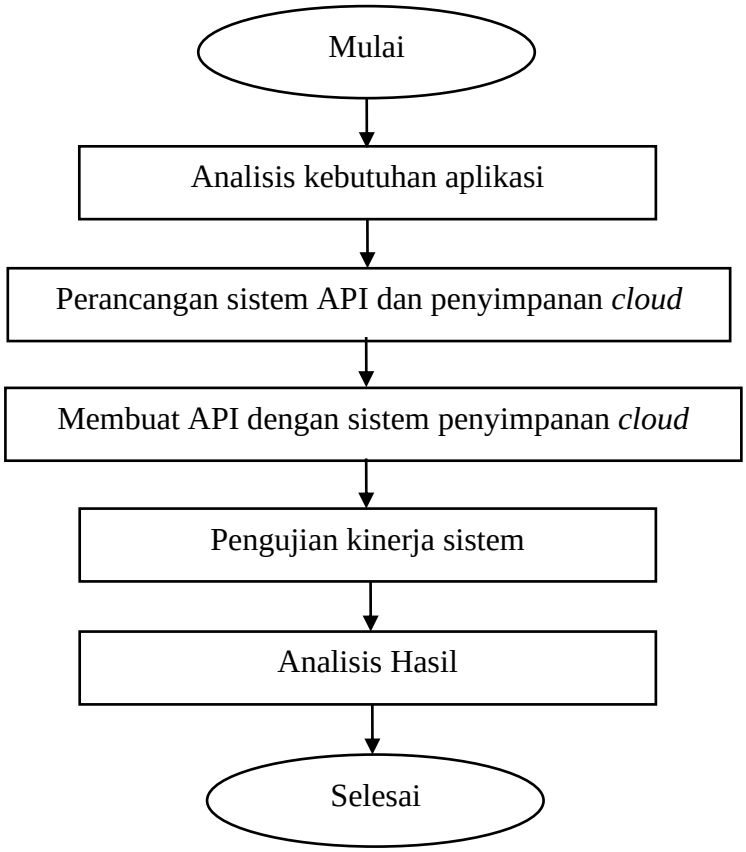
6. Pengujian Hasil Penelitian

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.



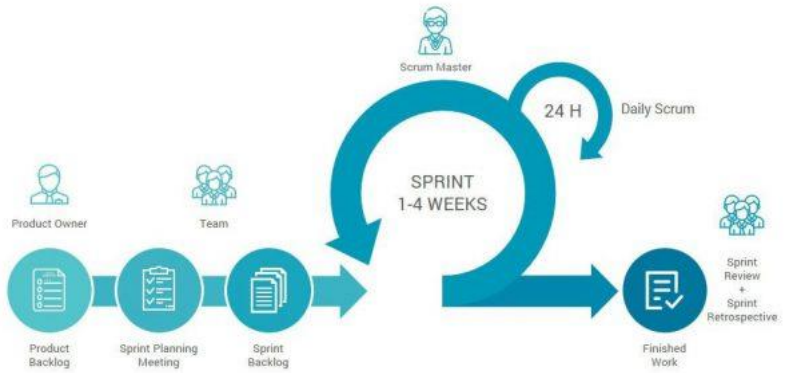
Gambar 3.1 Diagram Alir Penelitian

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