,

PENGEMBANGAN *BACKEND* SISTEM PENDUKUNG KEPUTUSAN CERDAS UNTUK PEMILIHAN MAKANAN DAN MINUMAN RESTORAN SPESIFIK INDONESIA (PRECIFOOD)

MUHAMMAD ILHAM HAKIM SUHERMAN

PROGRAM SARJANA ILMU KOMPUTER
SEKOLAH SAINS DATA, MATEMATIKA, DAN INFORMATIKA
INSTITUT PERTANIAN BOGOR
BOGOR
2025

PERNYATAAN MENGENAI LAPORAN AKHIR DAN SUMBER INFORMASI SERTA PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa laporan akhir dengan judul "Pengembangan *Backend* Sistem Pendukung Keputusan Cerdas untuk Pemilihan Makanan dan Minuman Restoran Spesifik Indonesia (PreciFood)" adalah karya saya dengan arahan dari dosen pembimbing dan belum diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir laporan akhir ini.

Dengan ini saya melimpahkan hak cipta dari karya tulis saya kepada Institut Pertanian Bogor.

Bogor, April 2025

Muhammad Ilham Hakim Suherman
G6401211056

ABSTRAK

MUHAMMAD ILHAM HAKIM SUHERMAN. Pengembangan *Backend* Sistem Pendukung Keputusan Cerdas untuk Pemilihan Makanan dan Minuman Restoran Spesifik Indonesia (PreciFood). Dibimbing oleh KARLISA PRIANDANA dan KUDANG BORO SEMINAR.

PreciFood merupakan aplikasi berbasis website yang digunakan sebagai sistem rekomendasi pemilihan menu berbasis *Genetic Algorithm* (GA) berdasarkan kebutuhan gizi konsumen spesifik. Model GA telah dikembangkan pada penelitian sebelumnya namun masih dijalankan secara lokal dan belum terintegrasi dengan antarmuka pengguna maupun sistem manajemen *database*. Fokus utama dalam penelitian ini adalah pengembangan *backend* dengan menghasilkan REST-API dan *database* untuk mengintegrasikan antara *frontend* dan GA yang dikembangkan, juga menunjang kebutuhan dari aplikasi. Dua tipe *backend*, yaitu *backend app service* untuk menunjang logika bisnis/utama (inti) dari aplikasi dan *backend model* untuk menjalankan model sistem rekomendasi pemilihan menu berbasis GA. Metode pengembangan perangkat lunak berupa *Prototyping* dilakukan melalui tiga iterasi untuk membagi pengembangan enam fitur dari aplikasi, di antaranya *user*, autentikasi, menu, notifikasi, rekomendasi, dan pemesanan (*order*). Arsitektur pengembangan *software multi-tier* diterapkan dalam pengembangannya dengan membagi sistem ke dalam lima lapisan dengan fungsinya masing-masing, di antaranya *presentation, application, model, data,* dan *storage.* 38 API pada *backend app service* dan satu API pada *backend model* berhasil dikembangkan untuk menunjang kebutuhan dari aplikasi. Penelitian berhasil tercapai dengan keberhasilan *testing* pada seluruh *endpoint* API melalui *blackbox testing* sesuai dengan *test case* yang diberikan, dan integrasi *frontend* dengan model GA berhasil dilakukan.

Kata Kunci : *Backend*, *multi-tier architecture*, precifood, REST-API, sistem rekomendasi

ABSTRACT

MUHAMMAD ILHAM HAKIM SUHERMAN. Development of the Backend for an Intelligent Decision Support System for the Selection of Food and Beverages in Specific Indonesian Restaurants (PreciFood). Supervised by KARLISA PRIANDANA dan KUDANG BORO SEMINAR.

PreciFood is a web-based application serving as a menu-selection recommendation system using a Genetic Algorithm (GA) tailored to the nutritional requirements of specific consumers. The GA model existed from prior research but ran only locally and lacked integration with a user interface or database management system. This study focuses on backend development by delivering a REST-API and database to integrate the frontend with the GA model and to support application needs. Two backend types were created, an app service for core business logic and a model service to execute the GA-based recommendation engine. The Prototyping software-development method was applied across three iterations to implement six features: user, authentication, menu, notifications, recommendations, and ordering. A multi-tier architecture was adopted, dividing the system into five layers, presentation, application, model, data, and storage, each with distinct responsibilities. 38 APIs in the app service and one API in the model service were developed. All endpoints passed black-box testing against defined test cases, and frontend–GA integration was successfully achieved.

Keywords: Backend, multi-tier architecture, precifood, REST-API, recommendation system

PENGEMBANGAN *BACKEND* SISTEM PENDUKUNG KEPUTUSAN CERDAS UNTUK PEMILIHAN MAKANAN DAN MINUMAN RESTORAN SPESIFIK INDONESIA (PRECIFOOD)

MUHAMMAD ILHAM HAKIM SUHERMAN

Skripsi
sebagai salah satu syarat untuk memperoleh gelar
Sarjana pada
Program Studi Ilmu Komputer

PROGRAM SARJANA ILMU KOMPUTER
SEKOLAH SAINS DATA, MATEMATIKA, DAN INFORMATIKA
INSTITUT PERTANIAN BOGOR
BOGOR
2025

Tim Penguji pada Ujian Skripsi:

1 Dr. Hendra Rahmawan, S.Kom., M.T.

Judul Skripsi : Pengembangan *Backend* Sistem Pendukung Keputusan Cerdas untuk Pemilihan Makanan dan Minuman Restoran Spesifik Indonesia (PreciFood)
Nama : Muhammad Ilham Hakim Suherman
NIM : G6401211056

Disetujui oleh

Pembimbing 1:
Dr. Karlisa Priandana, S.T., M.Eng.

Digitally signed by:
Karlisa Priandana
Date: 9 Mei 2025 08.38.46 WIB
Verify at dsign.ipb.ac.id

Pembimbing 2:
Prof. Dr. Ir. Kudang Boro Seminar, M.Sc.

Digitally signed by:
Kudang Boro Seminar
Date: 8 Mei 2025 09.29.58 WIB
Verify at dsign.ipb.ac.id

Diketahui oleh

Ketua Program Sarjana Ilmu Komputer:
Dr. Sony Hartono Wijaya, S.Kom., M.Kom.
19810809 200812 1 002

Tanggal Ujian:
16 April 2025

Tanggal Lulus:

PRAKATA

Puji dan syukur penulis panjatkan kepada Allah SWT atas segala karunia-Nya sehingga skripsi ini berhasil diselesaikan. Penyusunan dan penyelesaian skripsi sebagai tugas akhir ini tentunya tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan ucapan terima kasih kepada:

1. Ibu Santi Aryani, Bapak Alm. Teguh Suherman Saputra, Muhammad Ikhsan Kamil Suherman selaku ibu, ayah, dan kakak penulis, serta keluarga lainnya yang senantiasa memberikan dukungan, doa, motivasi, dan kasih sayang yang melimpah kepada penulis.
2. Ibu Dr. Karlisa Priandana, S.T., M.Eng., Bapak Prof. Dr. Ir. Kudang Boro Seminar, M.Sc. selaku dosen pembimbing pertama dan kedua yang senantiasa memberikan ilmu, kesempatan, bimbingan, arahan, dan masukan dalam penelitian ini sehingga dapat diselesaikan dengan baik.
3. Bapak Dr. Harry Imantho, M.Sc., Ibu Prof. Evy Damayanthi, MS., Bapak Dr. Bonang Waspadadi Ligar, dan Ibu Dr. Annisa Utami Seminar selaku dosen peneliti yang juga tergabung di dalam penelitian BIMA 2024, yang telah memberikan ilmu, arahan, dan masukan selama penelitian berlangsung.
4. Bapak Dr. Hendra Rahmawan, S.Kom., M.T. selaku dosen penguji yang telah memberikan saran dan masukan sehingga penelitian ini dapat menjadi lebih baik.
5. Seluruh dosen Ilmu Komputer IPB yang telah memberikan ilmu, pembelajaran, dan pengalamannya yang sangat bermanfaat selama masa studi dan menjadi inspirasi bagi penulis dalam menyelesaikan penelitian ini.
6. Rekan-rekan asisten peneliti yang telah bekerja sama dengan aktif dan baik.
7. Sahabat penulis yang senantiasa mendukung dan memberikan motivasi untuk menyelesaikan penelitian ini.
8. Teman-teman program S1 Ilmu Komputer IPB angkatan 58 yang senantiasa memberikan dukungan, kebersamaan, dan semangat, menjadikan perjalanan studi ini penuh makna.
9. Berbagai pihak yang telah memberikan bantuan, dukungan, dan kesempatan berharga, baik secara langsung maupun tidak langsung, dalam mewujudkan penelitian ini.

Semoga karya ilmiah ini menjadi manfaat bagi pihak yang membutuhkan dan bagi kemajuan ilmu pengetahuan.

Bogor, April 2025

Muhammad Ilham Hakim Suherman

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PENDAHULUAN

1.1 Latar Belakang

Pangan merupakan salah satu kebutuhan dasar atau primer yang harus dipenuhi oleh manusia. Hal ini dikarenakan pangan menjadi sumber energi dan nutrisi yang penting untuk kesehatan dan proses tumbuh kembang (Tapsell *et al.* 2016). Pangan tersusun atas berbagai kandungan nutrisi yang kompleks dan sangat bervariasi sehingga terdapat kemungkinan adanya interasi yang sinergis antara komponen penyusunnya (Galanakis 2021). Nutrisi dalam pangan terbagi menjadi dua, yaitu *macronutrient* dan *micronutrient* (EFSA 2017). *Macronutrient* merupakan komponen nutrisi utama yang membentuk total asupan kalori, dikonsumsi oleh manusia dalam jumlah yang besar sehingga menjadi sumber energi utama bagi tubuh manusia (Carrerio *et al*. 2016). Komponen dari *macronutrient* di antaranya ialah karbohidrat, protein, dan lemak (Savarino *et al.* 2021). *Micronutrient* merupakan komponen dalam nutrisi yang tidak memberikan kontribusi yang signifikan pada asupan kalori, tetapi tetap penting untuk kesehatan dan fungsi vital dalam organ tubuh, meskipun hanya dibutuhkan dalam jumlah yang kecil (Shergill-Bonner 2017). Komponen dari *micronutrient* di antaranya ialah zink, zat besi, vitamin, dan asam folat (Savarino *et al.* 2021). Kekurangan kedua komponen tersebut akan menyebabkan terjadinya masalah kesehatan dan mengganggu proses tumbuh kembang manusia sehingga dalam pemenuhannya harus dilakukan secara seimbang (Savarino *et al.* 2021).

Pemenuhan pangan berupa makanan dan minuman yang sehat dan bergizi perlu menjadi perhatian. Namun, cara pemilihan yang tepat belum banyak diketahui oleh masyarakat pada umumnya sehingga sering kali terabaikan. Hal ini dikarenakan kurangnya pengetahuan dan informasi yang dimiliki. Padahal, pemenuhan gizi yang tidak mencukupi akan menyebabkan berbagai implikasi pada aspek kesehatan sehingga dapat memicu berbagai macam kerawanan penyakit, khususnya penyakit tidak menular (PTM) (Tapsell *et al*. 2016). PTM atau yang biasa dikenal dengan *noncommunicable diseases* (NCDs) merupakan penyebab utama dari kematian juga tantangan kesehatan global pada abad 21 (WHO 2018).

PTM dapat disebabkan oleh berbagai faktor, salah satunya ialah perilaku masyarakat yang berisiko, seperti konsumsi dan ketersediaan makanan yang tidak sehat (Kemenkes 2017). Penyediaan konsumsi makanan yang sehat dan bergizi sering kali terabaikan. Hal ini disebabkan dengan adanya perubahan pola konsumsi seiring dengan perkembangan zaman dan pergeseran pola hidup. Tren makan dan memesan makanan dari luar, seperti restoran juga telah menjadi salah satu pilihan utama di tengah kesibukan dan kemudahan yang diberikan. Pemilihan menu makanan yang sehat dan bergizi oleh konsumen harus menjadi perhatian utama untuk mencegah terjadinya PTM. Oleh karena itu, pertimbangan dalam pemenuhan gizi yang sehat melalui pemilihan menu oleh pengunjung restoran juga harus menjadi perhatian khusus sehingga dapat mendorong pengunjung untuk melakukan pembelian menu yang sesuai dengan kebutuhannya melalui kemudahan akses makanan yang sehat dan bergizi (Kang *et al.* 2015).

Pemenuhan gizi yang sehat dan tercukupi bagi setiap individu akan berbeda karena karakteristik dan keunikannya atau disebut juga sebagai *personalized nutrition*. *Personalized nutrition* berakar pada konsep bahwa satu ukuran tidak

cocok untuk semua, berbanding terbalik dengan rekomendasi pemenuhan gizi yang selama ini dilakukan oleh masyarakat, yaitu "*one size fits all*" (Bush *et al.* 2020). Tujuan dari *personalized nutrition* ialah untuk memberikan rekomendasi dalam pemenuhan gizi yang lebih efektif dan tepat sasaran untuk meningkatkan kesehatan dan kesejahteraan individu serta mencegah atau mengelola berbagai penyakit terkait gizi (Boorsma *et al*. 2017).

Penelitian yang dilakukan oleh Seminar *et al.* pada tahun 2024 berhasil mengembangkan model *Genetic Algorithm* (GA) sebagai algoritma *decision support system* (DSS) atau sistem pendukung keputusan cerdas yang memberikan rekomendasi pilihan menu berdasarkan karakteristik individu tertentu (jenis kelamin, usia, berat badan, tinggi badan, dan riwayat penyakit) dan kandungan nutrisi dari setiap menu, di antaranya kalori protein, lemak, karbohidrat, natrium, kolestrol, *saturated fatty acid* (SFA), *mono-unsaturated fatty acid* (MUFA), dan *poly-unsaturated fatty acid* (PUFA) (Seminar *et al.* 2024). Pengembangan lanjutan dilakukan dengan menjadikan PreciFood sebagai sebuah aplikasi berbasis *website*, yang merupakan bagian dari Basis Informasi Penelitian dan Pengabdian kepada Masyarakat (BIMA) 2024 berdasarkan Surat Keputusan Nomor 0459/E5/PG.02.00/2024 dan Kontrak Nomor 027/E5/PG.02.00.PL/2024 dari Direktorat Penelitian dan Pengabdian kepada Masyarakat (DPPM), Kementerian Pendidikan Tinggi Sains dan Teknologi Republik Indonesia, diketuai oleh Prof. Dr. Ir. Kudang Boro Seminar, M. Sc, dan Restoran Karimata menjadi mitra di dalam pengembangan. Namun, terdapat beberapa kendala untuk mengembangkan PreciFood menjadi sebuah aplikasi, di antaranya:

a. Kebutuhan (*requirement*) menjadi sebuah aplikasi belum didefinisikan dengan detail karena merupakan pengembangan pertama menjadi sebuah aplikasi sehingga diperlukan perancangan modul dan alur kerja.
b. Ketiadaan manajemen basis data (*database*) yang terstruktur untuk menyimpan dan mengelola data.
c. Model GA yang masih dijalankan secara lokal sehingga belum terintegrasi dengan aplikasi.

Pengembangan menjadi sebuah aplikasi dibagi menjadi dua tipe pengembangan, yaitu pengembangan antar muka (*interface*/*frontend*) dan *backend,* di mana pengembangan *backend* yang menjadi fokus dalam penelitian ini. Pengembangan *backend* dari PreciFood menggunakan metode *Prototyping* dengan sifatnya yang iteratif sehingga cocok untuk pengembangan aplikasi di mana kebutuhannya (*requirement*) masih general dan belum didefinisikan dengan detail. Dua tipe *backend*, yaitu *backend app service* dan *backend model* dikembangkan. *Backend app service* digunakan untuk mengelola logika bisnis (utama) dari PreciFood yang membagi pengembangan ke dalam enam modul (fitur) fungsionalitas di antaranya *user*, autentikasi, menu, notifikasi, rekomendasi, dan pemesanan (*order*). *Backend model* digunakan untuk menjalankan komputasi dari model GA. Pengembangan *backend* menghasilkan *Representational State Transfer-Application Programming Interface* (REST-API) sebagai bentuk metode komunikasi antar sistem yang akan digunakan untuk memenuhi kebutuhan dari aplikasi.

1.2 Perumusan Masalah

Berdasarkan kendala yang telah diuraikan pada latar belakang, rumusan masalah pada penelitian ini ialah bagaimana layanan *backend* berupa API dan

database dari PreciFood dapat dikembangkan sehingga integrasi antara sistem rekomendasi berbasis GA dan *frontend* dapat dilakukan, serta fitur dalam penggunaan PreciFood berupa *user,* autentikasi, menu, notifikasi, rekomendasi, dan pemesanan dapat dibangun.

1.3 Tujuan

Tujuan dari penelitian ini ialah membangun layanan *backend* dari PreciFood meliputi pengembangan *database* dan REST-API yang digunakan untuk mengintegrasikan antara model GA yang telah dikembangkan pada penelitian sebelumnya, dengan *frontend* berbasis *website*, dan pengembangan fitur/modul di antaranya *user*, autentikasi, menu, notifikasi, rekomendasi, dan pemesanan.

1.4 Manfaat

Manfaat dari penelitian ini adalah tersedianya layanan *backend* PreciFood berupa manajemen *database* dan REST-API untuk mendukung pengoperasian aplikasi sehingga memungkinkan komunikasi antar sistem, serta menghadirkan pengelolaan fitur-fitur seperti *user*, autentikasi, menu, notifikasi, rekomendasi, dan pemesanan. Hal ini dilakukan agar PreciFood dapat digunakan sebagai sebuah aplikasi sistem pendukung keputusan cerdas bagi konsumen di Restoran Karimata dalam melakukan pemilihan menu makanan dan minuman.

1.5 Ruang Lingkup

Ruang lingkup dari penelitian ini ialah sebagai berikut:

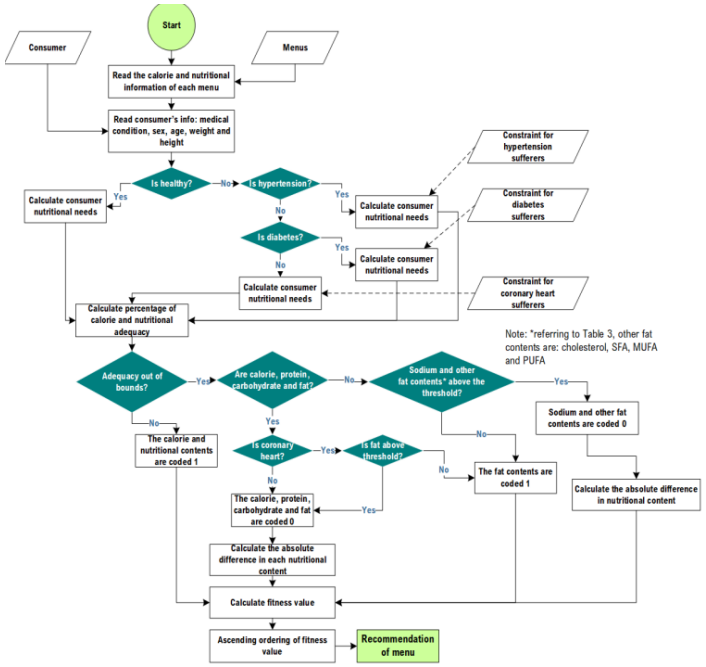
a. Subjek penelitian ini dilakukan pada restoran spesifik, yaitu Restoran Karimata yang berlokasi di Sentul, Bogor, Jawa Barat. Menu yang akan ditampilkan, dianalisis, dan menjadi basis rekomendasikan dalam PreciFood merupakan menu yang disajikan oleh restoran ini.
b. Fungsionalitas dari layanan *backend* yang dibangun dalam penelitian ini disesuaikan pada kebutuhan berbagai *stakeholder*, mulai dari Restoran Karimata, pengunjung restoran, dan peneliti yang tergabung di dalam pengembangan.
c. Penelitian ini akan mengintegrasikan layanan *backend* dengan GA yang dikembangkan oleh Seminar *et al.* (2024) dan *frontend* yang dikembangkan oleh Ismy Fana Fillah.
d. Model GA menerima dua jenis input data yang bersumber dari konsumen untuk menghitung kebutuhan gizi dan input dari menu berupa nutrisinya.
e. Perhitungan kebutuhan kalori dan nutrisi mempertimbangkan jenis kelamin, usia, tinggi badan, berat badan, dan riwayat kesehatan berupa penyakit tidak menular (PTM). Riwayat penyakit yang diakomodasikan pada model, di antaranya sehat, jantung koroner, diabetes, dan hipertensi. Pemilihannya dibatasi hanya pada satu kondisi medis yang dominan (Seminar *et al.* 2024).
f. Menu yang disimpan, dianalisis, dan ditampilkan sebanyak 82 macam menu, di antaranya makanan pokok 1 jenis, lauk pauk 33 jenis, sayuran 18 jenis, minuman 26 jenis, dan cemilan (*snack*) 4 jenis. Namun, pemberian rekomendasi satu set menu hanya berisikan 4 kategori menu, yaitu makanan pokok, lauk pauk, sayuran, dan minuman.

TINJAUAN PUSTAKA

2.1 *Personalized Nutrition*

Personalized nutrition (PN) merupakan konsep yang berakar dari "*one size not fit all*". Artinya setiap individu memiliki perbedaan dalam memenuhi kebutuhan nutrisinya berdasarkan biokimia, metabolisme, genetika, dan mikroba yang dimilikinya. Hal ini menyebabkan penyerapan nutrisi dan status nutrisi yang dimiliki dan dibutuhkan oleh setiap individu berbeda-beda, berbanding terbalik dengan rekomendasi pemenuhan gizi yang selama ini dilakukan oleh masyarakat, yaitu "*one size fits all*" (Bush *et al.* 2020). Tujuan dari *personalized nutrition* ialah untuk memberikan rekomendasi dalam pemenuhan gizi yang lebih efektif dan tepat sasaran untuk meningkatkan kesehatan dan kesejahteraan individu serta mencegah atau mengelola berbagai penyakit terkait gizi (Boorsma *et al.* 2017). Selain itu, *personalized nutrition* juga dikategorikan ke dalam bidang yang memanfaatkan individualitas manusia untuk mengembangkan strategi nutrisi yang mencegah, mengelola, dan mengobati penyakit serta mengoptimalkan kesehatan (Bush *et al.* 2020).

2.2 Sistem Rekomendasi Pemilihan Menu Berbasis *Genetic Algorithm*



Gambar 1 Alur kerja sistem rekomendasi berbasis *Genetic Algorithm* (Seminar *et al.* 2024)

Sistem rekomendasi pemilihan menu di suatu restoran telah dikembangkan oleh Seminar *et al.* (2024) dengan mengimplementasikan GA. Pemberian rekomendasi menu didasarkan pada karakteristik dari konsumen di antaranya jenis kelamin, usia, tinggi badan, berat badan, dan riwayat penyakit tidak menular (*non communicable diseases*) yang dideritanya. Proses dalam tahapan GA dimulai dengan inisialisasi populasi awal berupa kombinasi menu secara acak. Setiap individu pada populasi ini direpresentasikan sebagai satu solusi potensial yang terdiri atas sejumlah menu. Kemudian, dilakukan evaluasi menggunakan fungsi

objektif yang menghitung kesesuaian antara kebutuhan nutrisi konsumen dengan kandungan nutrisi dalam menu. Setelah evaluasi, dilakukan seleksi individu terbaik berdasarkan nilai kecocokan tersebut, dilanjutkan dengan proses *crossover* untuk menghasilkan keturunan baru dengan menggabungkan sebagian atribut dari dua individu terpilih. Mutasi juga diterapkan secara acak untuk menjaga keberagaman solusi. Proses ini berulang selama beberapa generasi hingga konvergen pada solusi terbaik, yaitu kombinasi menu yang paling sesuai dengan karakteristik dan kebutuhan gizi konsumen (Seminar *et al.* 2025).

2.3 Arsitektur *Software Multi-Tier*

Arsitektur *multi-tier* atau *N-tier* merupakan salah satu pendekatan dari *software architectural pattern* yang banyak diimplementasikan dalam pengembangan perangkat lunak modern. Arsitektur ini memisahkan antara berbagai fungsi dalam aplikasi ke dalam beberapa lapisan dengan tujuan untuk meningkatkan efisiensi dan skalabilitas. Berdasarkan penelitian yang dilakukan oleh Uzair dan Naz (2023), kelebihan dari arsitektur ini terletak pada modularitas yang memudahkan distribusi beban kerja ke berbagai *server*. Hal ini menjadikan pengembangan sistem lebih fleksibel. Dalam konteks implementasi *Artificial Intellegence* (AI), arsitektur ini memberikan keuntungan dengan memungkinkan integrasi komponen AI sebagai lapisan tersendiri sehingga model AI dapat dijalankan secara independen melalui *Application Programming Interface* (API). Hal ini memudahkan pembaruan dan pengelolaan model tanpa mengganggu operasi lapisan lainnya.

2.4 *Backend*

Backend merupakan sebuah istilah yang digunakan dalam komponen sistem yang bertanggung jawab untuk menangani logika bisnis, *database*, hingga integrasi *server* dalam suatu aplikasi, yang tidak terlihat langsung oleh pengguna (terjadi pada *background* aplikasi) (Máriás dan Molnár 2020). Dalam pengembangan *software*, *backend* dan *frontend* memiliki peran yang berbeda namun tetap harus terintegrasi dengan baik. Hal ini disebabkan karena *backend* menyediakan *interface* seperti melalui penerapan API yang akan digunakan pada *frontend*, agar nantinya proses komunikasi, permintaan data, dan pengaksesan data dapat berjalan.

2.5 *Application Programming Interface* (API)

Application Programming Interface (API) merupakan sebuah teknologi yang berisikan sekumpulan instruksi/protokol untuk memfasilitasi pertukaran informasi atau data antara dua atau lebih aplikasi perangkat lunak (Hanafi *et al.* 2017). API juga berfungsi sebagai jembatan yang memungkinkan sistem atau aplikasi yang berbeda untuk saling berinteraksi tanpa harus mengetahui detail implementasi internal masing-masing. Misalkan, protokol ini digunakan sebagai protokol komunikasi untuk pertukaran data antara sistem *frontend* dan juga *backend* sehingganya keduanya dapat saling terintegrasi.

2.6 *Representational State Transfer* (REST)

Representational State Transfer (REST) merupakan salah satu dari gaya arsitektur modern perangkat lunak (software) yang mengatur bagaimana komunikasi antar sistem agar dapat saling terhubung. Arsitektur ini juga dapat diterapkan dalam membangun suatu *Application Programming Interface* (API).

REST menggunakan metode atau protokol HTTP untuk melakukan komunikasi antara *server* dengan klien. REST bersifat *stateless* yang artinya tidak ada status atau informasi yang disimpan di dalam *server* (Costa *et al.* 2014*).* Dalam komunikasi antara klien dan *server*, protokol HTTP digunakan untuk mengirim permintaan melalui berbagai metode, seperti GET, POST, PUT, PATCH, dan DELETE. Selain itu, JavaScript Object Notation (JSON) adalah format pertukaran data yang umum digunakan dalam REST API, yang menyajikan data dalam struktur *key-value* sederhana dan dapat dikelola oleh berbagai bahasa pemrograman.

2.7 *Cloud Computing*

Cloud computing atau komputasi awan, merupakan istilah sederhana yang berarti menyimpan dan mengakses data serta program melalui internet, bukan pada *hard drive* komputer pengguna (Rashid dan Chaturvedi 2019). Melalui penerapan *cloud computing*, pengguna dapat mengakses dan menggunakan berbagai aplikasi dari berbagai *device* melalui internet. Penyedia layanan komputasi *cloud* disebut juga sebagai Cloud Service Providers (CSPs), salah satunya ialah Google dengan layanan Google Cloud Platform (GCP) yang dimilikinya. Model layanan dari komputasi *cloud* terbagi menjadi empat, di antaranya *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), *Software as a Service* (SaaS), dan *Recovery as a Service* (RaaS) (Rashid dan Chaturvedi 2019).

2.8 *Serverless*

Serverless merupakan layanan komputasi *cloud* yang memungkinkan pengguna menjalankan aplikasi berbasis *event-driven* dengan tagihan yang terperinci, tanpa harus menangani infrastruktur *server* atau operasionalnya secara langsung (van Eyk *et al.* 2018). Pengelolaan *server* komputasi dari layanan ini akan sepenuhnya diserahkan kepada pihak layanan *cloud* yang digunakan, seperti Cloud Run yang merupakan layanan *serverless* dari Google Cloud Platform (GCP). Menurut van Eyk *et al.* (2018) terdapat beberapa manfaat dari penggunaan layanan *serverless*, di antaranya:

a. *Imporved resource management* yang berarti pengelolaan sumber daya menjadi lebih efisien karena pada *serverless*, aplikasi dipecah menjadi bagian-bagian kecil/spesifik (*fine-grained*) sehingga penyedia *cloud* dapat mencocokkan kebutuhan aplikasi dengan sumber daya yang tepat secara otomatis. Hal ini berbeda dengan model tradisional yang mengandalkan sumber daya besar dan umum seperti VM, yang sering kali tidak cocok dan menyebabkan pemborosan atau kekurangan sumber daya.
b. *More insight and control* yang berarti memberikan kontrol dan *insight* lebih besar terhadap perilaku aplikasi karena tanggung jawab untuk pemantauan, penyediaan, dan pengelolaan sumber daya kini berada pada penyedia *cloud*. Hal ini memungkinkan penyedia layanan untuk mengumpulkan data yang lebih detail, melakukan *profiling* yang akurat, dan membuat keputusan operasional seperti *autoscaling* dengan lebih baik.
c. *Granular scaling* yang berarti memiliki kemampuan untuk melakukan *scaling* secara spesifik pada bagian-bagian kecil dari aplikasi (fungsi atau layanan individual) sehingga sumber daya dapat disesuaikan hanya pada komponen yang membutuhkan skalabilitas, tanpa harus *scaling* seluruh aplikasi seperti pada pendekatan tradisional, yang tidak hanya kurang efisien tetapi juga sering kali tidak mampu mengatasi *bottleneck* dengan efektif.

METODE

3.1 Peralatan Penelitian

Penelitian ini dilakukan dengan menggunakan perangkat keras dan lunak dengan rincian spesifikasi sebagai berikut:

a. Perangkat keras (*hardware*) untuk pengembangan layanan *backend*:

1) Central Processing Unit (CPU) Ryzen 5 5600H
2) RAM 16 GB
3) SSD 512 GB

b. Perangkat lunak (*software*) untuk pengembangan layanan *backend*:

1) *Operating System* (OS) Windows 11 64-bit
2) *Integrated Development Environment* (IDE) WebStorm dan VSCode
3) Bahasa pemrograman TypeScript dengan *runtime environment* NodeJs dan *framework* ExpressJs untuk membangun *backend app service*.
4) Bahasa pemrograman Python dengan *framework* Quart untuk membangun *backend model*.
5) *Relational Database Management System* (RDBMS) PostgreSQL
6) API *Documentation and Testing Tool* Postman
7) Desain UML *Tool* draw.io

c. Lingkungan *deployment*

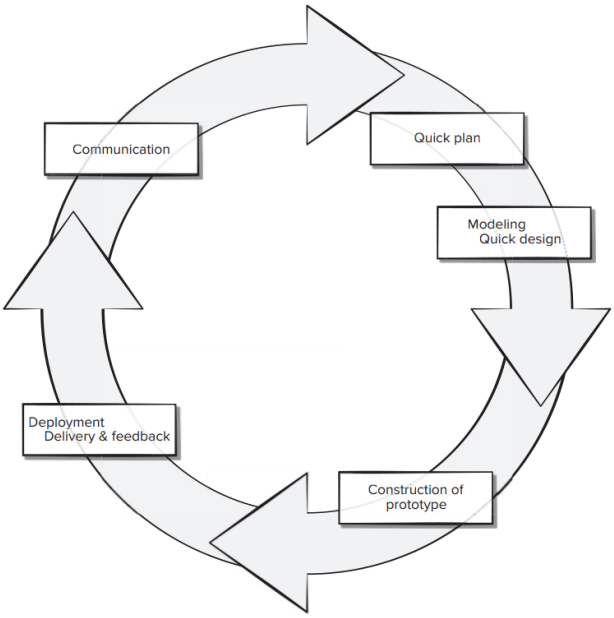
1) Kontenerisasi dengan Docker.
2) *Deployment backend app service, backend model,* dan *frontend* pada Google Cloud Platform (GCP) melalui layanan *serverless* yaitu Cloud Run.
3) *Object Storage* dengan Google Cloud Storage.
4) *Database hosting* dengan *provider* Neon.

3.2 Tahapan Penelitian

Metode *Prototyping* digunakan dalam pengembangan *backend* PreciFood. *Prototyping* merupakan metode dalam pengembangan aplikasi yang dapat memahami kebutuhan pengguna dengan lebih baik karena memiliki fleksibilitas terhadap *requirement* aplikasi yang masih bersifat general dan belum detail (Pressman dan Maxim 2020). Metode ini terbagi ke dalam lima tahapan, yaitu komunikasi (*communication*), perencanaan cepat (*quick plan*), pemodelan perancangan cepat (*modeling quick design*), pengembangan prototipe (*construction of prototype*), dan penyebaran dan umpan balik (*deployment delivery and feedback*), seperti pada Gambar 2.

Aplikasi PreciFood merupakan pengembangan pertama dalam penelitian ini sehingga *requirementnya* masih belum didefinisikan dengan detail. Penerapan metode *Prototyping* dapat lebih menyesuaikan kebutuhan dari *stakeholder* dan integrasi dari sistem. Pada penelitian ini, pengembangan dari *backend* PreciFood dilakukan dalam tiga iterasi. Iterasi pertama dilakukan untuk membangun *database* dan modul/fitur *user*, menu, dan notifikasi pada *backend app service*. Iterasi kedua

dilakukan untuk membangun *backend model* yang akan menjalankan komputasi sistem rekomendasi dari menu berbasis model GA. Iterasi ketiga dilakukan untuk membangun fitur rekomendasi dan pemesanan (*order*) pada *backend app service*. ‘



Gambar 2 Metode *Prototyping* (Pressman dan Maxim 2020)

Komunikasi (*Communication*)

Penelitian ini diawali dengan komunikasi antara pengembang dan juga *stakeholder* yang tergabung. Tujuannya ialah untuk mempelajari dan transfer ilmu terkait penelitian sebelumnya yang menghasilkan model GA sebagai sistem rekomendasi pemilihan menu oleh Seminar *et al.* (2024) dan kebutuhan dari *stakeholder*. *Stakeholder* dalam pengembangan merupakan para peneliti dan asisten yang tergabung di dalam Basis Informasi Penelitian dan Pengabdian kepada Masyarakat (BIMA) 2024, yang diketuai oleh Prof. Dr. Ir. Kudang Boro Seminar, M.Sc. Penelitian ini merupakan penelitian gabungan dari berbagai disiplin ilmu di Institut Pertanian Bogor dan Universitas Gunadarma, mulai dari Ilmu Komputer, Teknik Mesin dan Biosistem, Gizi, dan Sains Komunikasi dan Pengembangan Masyarakat (SKPM). Selain itu, terdapat *stakeholder* yang menjadi mitra dalam pengembangan PreciFood, yaitu Restoran Karimata yang berlokasi di Sentul, Bogor, Jawa Barat.

Perencanaan Cepat (*Quick Plan*)

Perencanaan cepat akan dilakukan melalui analisis kebutuhan fungsionalitas sistem dari hasil komunikasi. Analisis pengguna dari aplikasi selanjutnya dilakukan untuk mendefinisikan aktor dalam penggunaan PreciFood. *User requirement* dan *story* kemudian dibentuk untuk menggambarkan kebutuhan dari penggunaan PreciFood oleh masing-masing aktor. Berdasarkan *user requirement* dan *story* yang telah dianalisis, *Unified Modeling Language* (UML) diagram berupa *use case* akan dibangun untuk merepresentasikan fitur-fitur yang akan dikembangkan dari user *requirement* yang telah ditetapkan sehingga menjadi batasan dalam pengembangan. Pengembangan use case bertujuan untuk menerjemahkan user requirement yang

telah didapatkan melalui tahapan komunikasi menjadi fitur-fitur yang akan dikembangkan dalam PreciFood. Selain itu, pengembangan use case juga ditujukan agar terdapat batasan-batasan di dalam pengembangan aplikasi. Use case juga digunakan untuk menentukan hubungan antara sistem dengan setiap aktor yang menjadi pengguna di dalam PreciFood. Pada tahapan perencanaan juga dilakukan perancangan terkait bagaimana integrasi antara model GA ke dalam aplikasi PreciFood dapat diimplementasikan.

Pemodelan Perancangan Cepat (*Modeling Quick Design*)

Pemodelan perancangan cepat dilakukan untuk merancang gambaran konsep sistem *backend* yang akan dikembangkan dari PreciFood. Pada tahapan ini dilakukan perancangan seperti *class diagram*, *database design*, *sequence diagram*, *activity diagram*, hingga arsitektur sistem. Berbagai kebutuhan akan dirancang dan diterjemahkan dari yang sederhana ke dalam implementasi teknis pengembangan sistem. Hasil dari perancangan ini yang menjadi acuan di dalam pengembangan *prototype*.

Pengembangan Prototipe (*Construction of Prototype*)

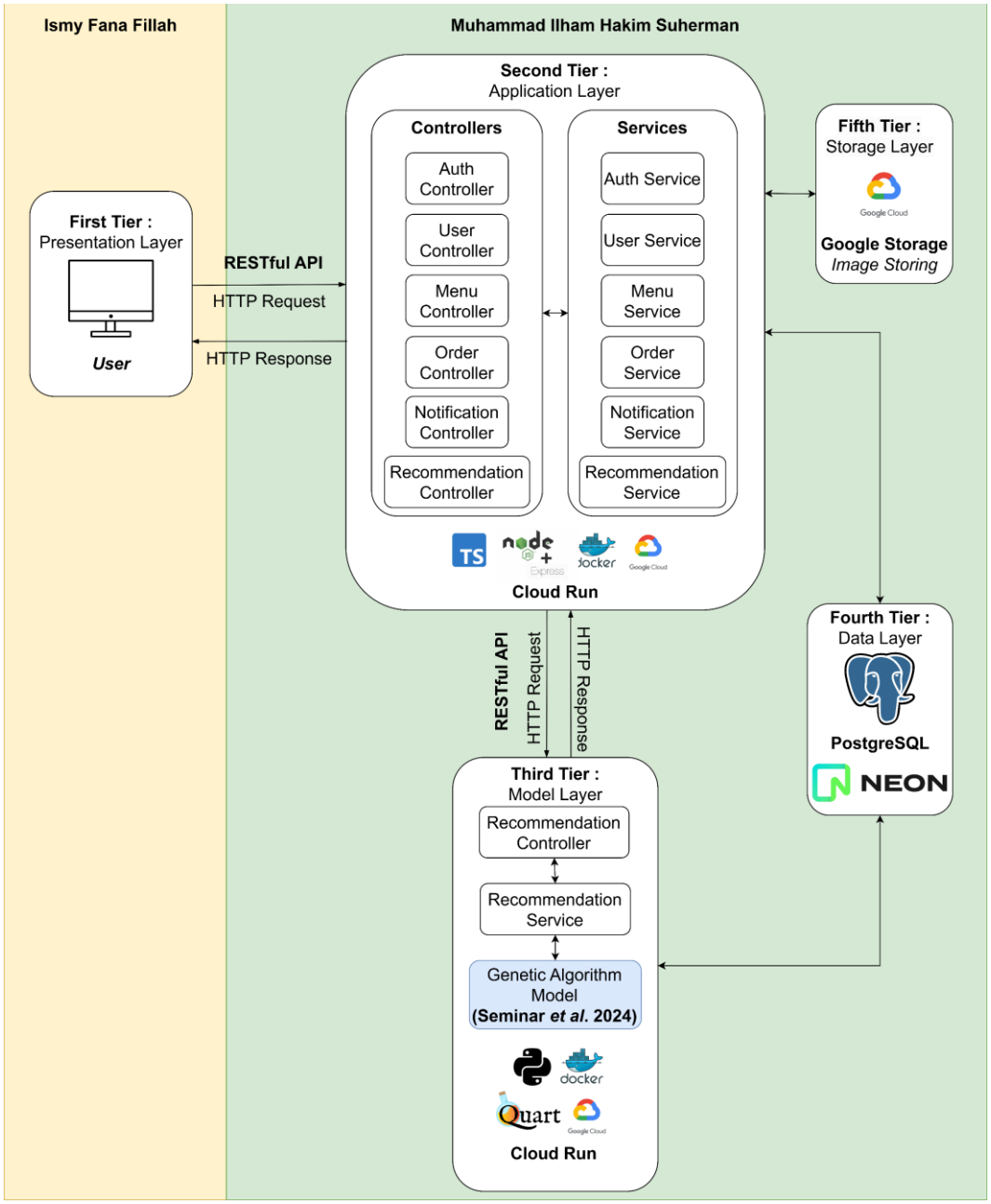
Pengembangan prototipe melalui implementasi kode program sesuai dengan desain dan kebutuhan yang telah ditetapkan di tahapan sebelumnya. Pada penelitian ini dikembangkan dua layanan *backend* yaitu *backend app service* dan *backend model*. *Backend app service* dibangun dengan bahasa pemrograman TypeScript dengan *framework* NodeJs dan ExpressJs. *Backend model* dibangun dengan bahasa pemrograman Python dengan *framework* Quart. Selain itu, pada tahapan ini juga dilakukan pengembangan *database* yang bertipe *Relational Database Management System* (RDBMS) dengan menggunakan PostgreSQL untuk memanajemen *structured data*. Pengembangan *object storage* untuk menyimpan *file* seperti gambar juga dibangun dengan menggunakan Google Cloud Storage.

Penyebaran dan Umpan Balik (*Deployment Delivery and Feedback*)

Setelah kode program dari kedua *backend* berhasil dikonstruksi, pengujian dengan *blackbox testing* dilakukan untuk menguji fungsionalitas dan integrasinya. *Integration testing* akan dilakukan dengan tujuan untuk menilai kinerja komponen individual secara keseluruhan dan mengidentifikasi masalah pada antar modul beserta fungsionalitasnya (Tan *et al.* 2022). Pengujian dilakukan terhadap *endpoint* API yang telah dikembangkan dengan skenario uji positif dan negatif. Pengujian dilakukan secara manual dengan menggunakan Postman. Selain itu, dilakukan evaluasi dari hasil *prototype* yang dikembangkan kepada *stakeholder*. Setelah itu, *deployment* dari hasil implementasi kode yang telah dibangun dan diuji dilakukan pada layanan *serverless* dari Google Cloud Platform (GCP) yaitu Cloud Run. Metode *deployment* dilakukan dengan melakukan kontenerisasi pada aplikasi atau kode program *backend* yang dibangun melalui Docker sehingga aplikasi dapat dijalankan dengan baik dan konsisten di berbagai lingkungan yang berbeda. Dockerfile terlebih dahulu dibuat agar nantinya dapat dijadikan sebagai intruksi atau perintah untuk *build image* dan *container* dari aplikasi.

HASIL DAN PEMBAHASAN

4.1 Arsitektur Sistem



Gambar 3 Arsitektur sistem

PreciFood mengimplementasikan *software architectural pattern* dengan menggunakan *multi-tier architecture* di dalam pengembangannya. Arsitektur ini membagi sistem menjadi beberapa lapisan (*layer*). Pengembangan PreciFood membagi sistem menjadi lima lapisan (*five-tier architecture*) yang tertera pada Gambar 3. Lapisan pertama yaitu *presentation layer* yang merupakan antarmuka (*interface*) atau *frontend* dari PreciFood berbasis *website application* yang dikembangkan oleh Ismy Fana Fillah. Lapisan kedua yaitu *application layer* yang berfungsi untuk memproses logika bisnis utama dari aplikasi, disebut juga sebagai

backend app service yang dibangun dengan bahasa pemrograman TypeScript dengan *runtime environtment* NodeJs dan *framework* ExpressJs. Lapisan ketiga yaitu *model layer* yang berfungsi untuk menjalankan komputasi dari model GA yang telah dikembangkan untuk menghasilkan (*generate*) rekomendasi menu, disebut juga sebagai *backend model* yang dikembangkan dengan menggunakan bahasa pemrograman Python dengan *framework* Quart. *Deployment* pada lapisan pertama, kedua, dan ketiga dilakukan pada layanan *serverless* berupa Cloud Run dengan melakukan kontenerisasi menggunakan Docker. Lapisan keempat yaitu *data layer* yang berfungsi untuk menyimpan dan mengelola *structured data* dengan menggunakan *Relational Database Management System* (RDBMS) yaitu PostgreSQL yang di*hosting* dengan menggunakan *provider* Neon. Terakhir lapisan kelima yaitu *storage layer* yang berfungsi untuk menyimpan *unstructured data* berupa *Binary Large Object* (BLOB) seperti gambar dengan menggunakan Google Cloud Storage. Komunikasi dilakukan melalui pengiriman *request* ke *endpoint* API dengan protokol HTTP yang berarsitektur RESTful API untuk mengakses sumber daya (*resource*) dari masing-masing *layer*.

4.2 Iterasi 1

4.2.1 Komunikasi (*Communication*)

Komunikasi terhadap *stakeholder* dilakukan untuk mempelajari kebutuhan (*requirement*) dari pengembangan aplikasi PreciFood dan mempelajari penelitian yang telah dicapai sebelumnya yaitu terkait pengembangan model sistem rekomendasi menu berbasis GA yang berhasil dikembangkan oleh Seminar *et al.* (2024). Model ini memberikan rekomendasi menu dengan mempertimbangkan kebutuhan nutrisi dari konsumen berdasarkan variabilitasnya dengan kandungan gizi (*macronutrient* dan *micronutrient*) dari menu yang telah dianalisis. Variabilitas dari individu ini diukur berdasarkan usia, jenis kelamin, tinggi badan, berat badan, dan riwayat penyakit yang dimilikinya, di antaranya diabetes, hipertensi, jantung koroner, dan sehat (tidak memiliki riwayat penyakit apapun). Kandungan gizi dari menu yang dianalisis untuk menjalankan algoritma ini di antaranya ialah kalori, protein, lemak, karbohidrat, natrium, kolesterol, *saturated fatty acid* (SFA), *mono-unsaturated fatty acid* (MUFA), *poly-unsaturated fatty acid* (PUFA). Kandungan gizi dari setiap menu dianalisis oleh peneliti dari Departemen Gizi melalui tahapan praproses berdasarkan Tabel Komposisi Pangan Indonesia (TKPI), Nutrisurvey, dan uji laboratorium.

Berdasarkan hasil komunikasi dengan *stakeholder* diketahui bahwa sistem rekomendasi pemilihan menu berbasis model GA yang dikembangkan pada penelitian sebelumnya ingin dibangun menjadi sebuah aplikasi Sistem Pendukung Keputusan (SPK) Cerdas dalam pemilihan menu di suatu restoran yang diberi nama PreciFood, yang mana pada penelitian ini Restoran Karimata dipilih menjadi mitra pengembangan. Namun, penelitian dan pengembangan lanjutan masih diperlukan karena terdapat kendala, di antaranya:

a. Model dari sistem rekomendasi masih dijalankan secara lokal.
b. Keterbatasan akses penggunaan sistem rekomendasi. Hal ini disebabkan karena ketiadaan *database* sehingga terjadi keterbatasan dalam melakukan manajemen data (*read*, *write*, *update*, dan *delete*). Selain itu, input data yang dibutuhkan oleh model masih dilakukan secara manual melalui *file Comma*

Separated Values (CSV). Hal ini juga menjadi tantangan ketika sistem rekomendasi diintegrasikan menjadi sebuah aplikasi.

c. Pengembangan PreciFood menjadi sebuah aplikasi pertama kali dilakukan pada penelitian ini. Oleh karena itu, perlu dilakukan analisis untuk menentukan proses atau logika bisnis dari aplikasi. Keterbatasan ini juga mengakibatkan ketiadaannya fitur-fitur pendukung yang menyebabkan lingkup penggunaan sistem rekomendasi menjadi terbatas.

Komunikasi juga dilakukan kepada Restoran Karimata sebagai mitra dari pengembangan PreciFood. Menu yang disajikan oleh restoran ini akan dianalisis kandungan nutrisinya oleh tim peneliti dari Departemen Ilmu Gizi IPB. Menu dari Restoran Karimata ini akan dimasukkan ke dalam *database* yang juga digunakan sebagai input data menu pada model GA. Selain itu, komunikasi dengan mitra bertujuan untuk mengetahui *requirement* yang diperlukan dan bagaimana sumber data atau informasi menu dari restoran dapat diakses dan ditampilkan.

4.2.2 Perencanaan Cepat (*Quick Planning*)

Perencanaan cepat selanjutnya dilakukan antara pengembang. Tahapan ini bertujuan untuk menerjemahkan kebutuhan-kebutuhan dari *stakeholder* menjadi suatu *system requirement*. Berdasarkan hasil komunikasi, disepakati dalam pengembangan ini setidaknya terdapat tiga aktor pengguna yang akan terlibat di dalam pengoperasian aplikasi, di antaranya admin, restoran, dan konsumen dengan definisi masing-masing aktor seperti pada Tabel 1.

Tabel 1 Pendefinisian setiap aktor pengguna PreciFood

No.	Jenis Aktor	Deskripsi
1	Peneliti (Admin)	Peneliti IPB dalam bidang ilmu gizi yang memiliki peran untuk menganalisis kandungan nutrisi dari menu yang disajikan restoran mitra.
2	Restoran	Restoran yang menjadi mitra PreciFood di mana menu yang disajikannya akan dianalisis, ditampilkan, dan direkomendasikan kepada konsumen. Pada penelitian ini, Restoran Karimata menjadi mitra pertama pengembangan PreciFood.
3	Konsumen	Pengguna dari aplikasi PreciFood yang dapat mengakses fitur-fitur konsumen dan melakukan *generate* rekomendasi menu dari restoran yang dipilihnya berdasarkan kebutuhan nutrisinya.

Berdasarkan hasil komunikasi diketahui juga bahwa setidaknya sistem rekomendasi berbasis model GA memerlukan data input yang berasal dari konsumen berupa variabilitas yang dimilikinya dan menu serta kandungan nutrisi yang terkandung di dalamnya. Oleh karena itu, rencana pengembangan *backend* dilakukan secara bertahap dan *incremental*, dimulai dari fitur/modul yang paling mendasar terlebih dahulu, dimulai dari pengembangan modul/fitur *user*, autentikasi, menu, dan notifikasi.

Modul *user* akan dikembangkan pertama kali untuk menunjang kebutuhan manajemen akun dari pengguna, seperti registrasi hingga manajemen profil pengguna. Pada modul ini, registrasi terhadap akun konsumen akan dikembangkan sehingga variabilitasnya didapatkan dan disimpan pada *database*. Selanjutnya, modul autentikasi akan dibangun untuk menunjang kebutuhan pengguna agar dapat mengakses modul-modul lainnya yang akan dikembangkan. Selain itu, modul autentikasi akan menunjang kebutuhan autentikasi dan otorisasi dari pengguna, salah satunya melalui penerapan *login*. Modul menu selanjutnya akan dikembangkan bagi pengguna agar dapat mengakses dan memanajemen menu yang tersedia di restoran, yaitu Restoran Karimata. Pada modul ini manajemen data menu dan nutrisi juga akan diatur sehingga dapat menjadi input bagi model GA. Terakhir, modul notifikasi akan dikembangkan sebagai petunjuk setiap adanya penambahan menu di suatu restoran. *User requirement* dari modul atau fitur yang akan dikembangkan didefinisikan pada Tabel 2.

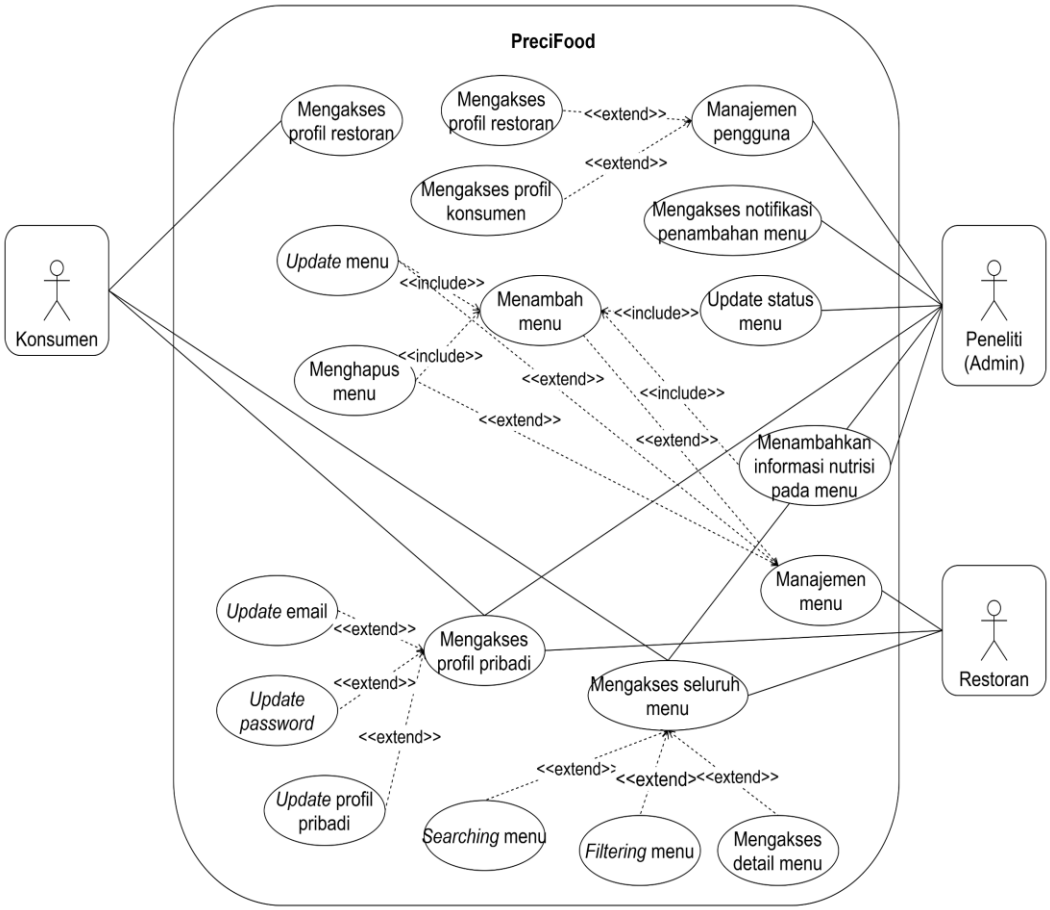
Tabel 2 *User requirement* iterasi pertama

Aktor	***Task***	***User Story***
Konsumen	Melakukan registrasi	Saya ingin mendaftar sebagai pengguna (konsumen) pada aplikasi PreciFood dan memasukkan berbagai informasi dan variabilitas yang saya miliki.
	Melakukan *log in*	Sebagai konsumen, saya ingin mengakses berbagai fitur yang disediakan dalam aplikasi.
	Mengakses profil pribadi	Sebagai konsumen, saya ingin mengakses profil pribadi.
	Memperbarui profil pribadi	Sebagai konsumen, saya ingin memperbarui profil yang saya miliki apabila terdapat perubahan.
	Mengakses restoran yang menjadi mitra.	Sebagai konsumen, saya ingin mengakses restoran yang menjadi mitra dalam aplikasi.
	Mengakses menu yang tersedia pada suatu restoran	Sebagai konsumen, saya ingin mengakses menu yang disajikan pada suatu restoran.
	Memperbarui *email*	Sebagai konsumen, saya ingin memperbarui *email* yang saya gunakan untuk keperluan autentikasi.
	Memperbarui *password*	Sebagai konsumen, saya ingin memperbarui *password* yang saya gunakan untuk keperluan autentikasi.

Tabel 2 *User requirement* iterasi pertama (*lanjutan*)

Restoran	Melakukan registrasi.	Saya ingin mendaftar sebagai restoran mitra PreciFood.
	Melakukan *login*	Sebagai restoran, saya ingin mengakses berbagai fitur yang disediakan dalam aplikasi.
	Melakukan manajemen terhadap menu yang disajikannya.	Sebagai restoran, saya ingin mendaftarkan menu-menu yang saya sajikan ke dalam aplikasi dan melakukan manajemen terhadap menu tersebut.
	Memperbarui informasi/profil dari restoran.	Sebagai restoran, saya ingin melakukan manajemen menu yang saya sediakan.
	Memperbarui *email*	Sebagai restoran, saya ingin memperbarui *email* yang saya gunakan untuk keperluan autentikasi.
	Memperbarui *password*	Sebagai restoran, saya ingin memperbarui *password* yang saya gunakan untuk keperluan autentikasi.
Peneliti (Admin)	Melakukan *login*	Sebagai admin, saya ingin mengakses berbagai fitur yang disediakan dalam aplikasi.
	Melakukan manajemen terhadap nutrisi dari menu yang disajikan.	Sebagai admin, saya ingin melakukan manajemen terhadap nutrisi dari menu yang disajikan.
	Memperbarui status *approval* terkait menu-menu baru yang dimiliki oleh restoran.	Sebagai admin, saya ingin menentukan *approval* dari suatu menu.
	Mengakses notifikasi terkait penambahan menu baru.	Sebagai admin, saya ingin mengetahui adanya penambahan menu baru yang dilakukan restoran.
	Melakukan akses terhadap pengguna aplikasi.	Sebagai admin, saya ingin mengakses profil pengguna baik konsumen dan restoran.
	Memperbarui *email*	Sebagai admin, saya ingin memperbarui *email* yang saya gunakan untuk keperluan autentikasi.
	Memperbarui *password*	Sebagai admin, saya ingin memperbarui *password* yang saya gunakan untuk keperluan autentikasi.

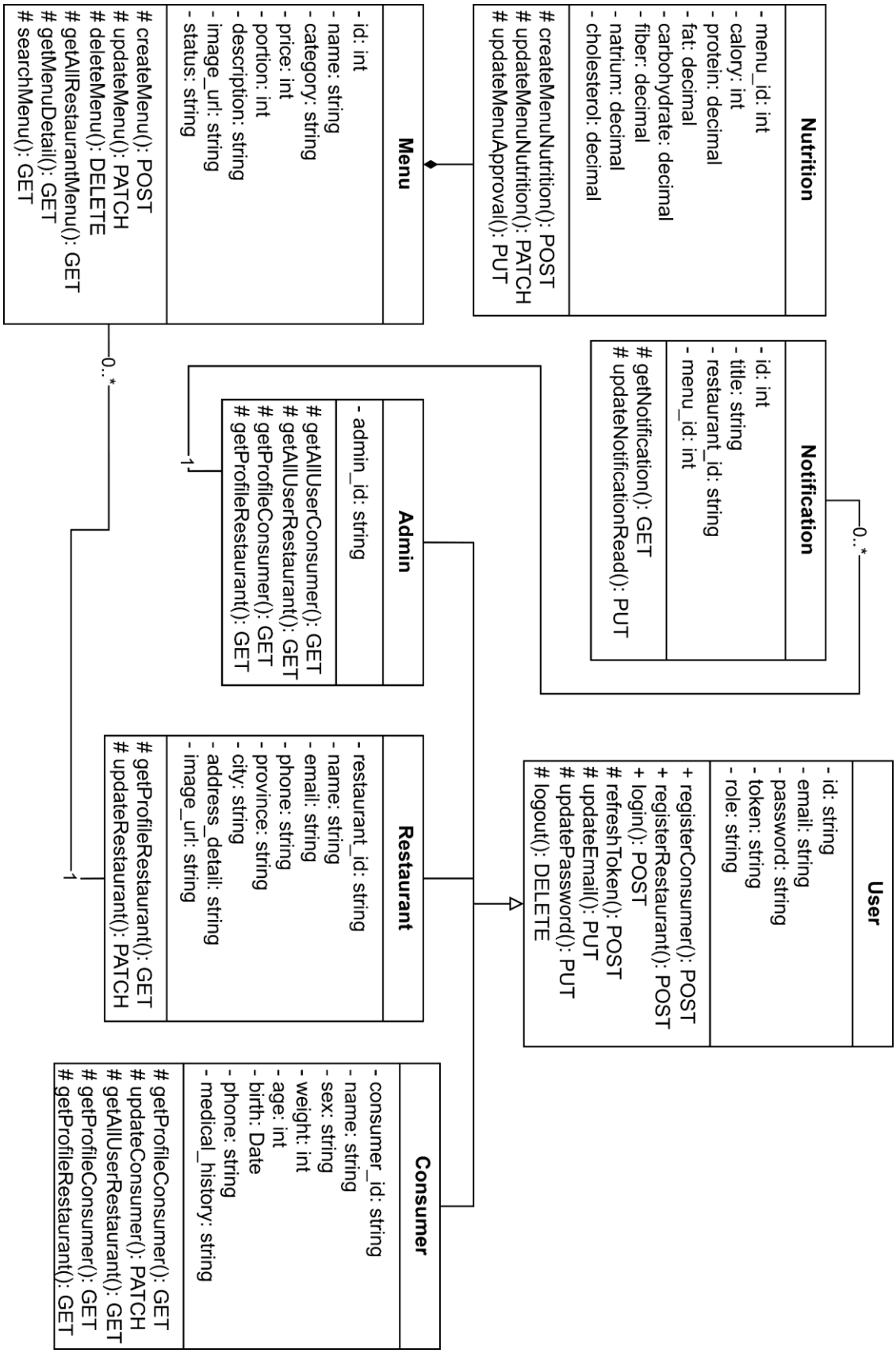
Use case selanjutnya dikembangkan berdasarkan *user requirement* yang telah dianalisis. Pada iterasi pertama ini, *use case* dari setiap aktor berhasil dikembangkan dan dapat terlihat pada Gambar 4. *Use case* pada iterasi pertama ini merupakan representasi dari modul atau fitur *user*, autentikasi, menu, dan notifikasi dengan asumsi bahwa pengguna telah berhasil melakukan registrasi dan *login*.



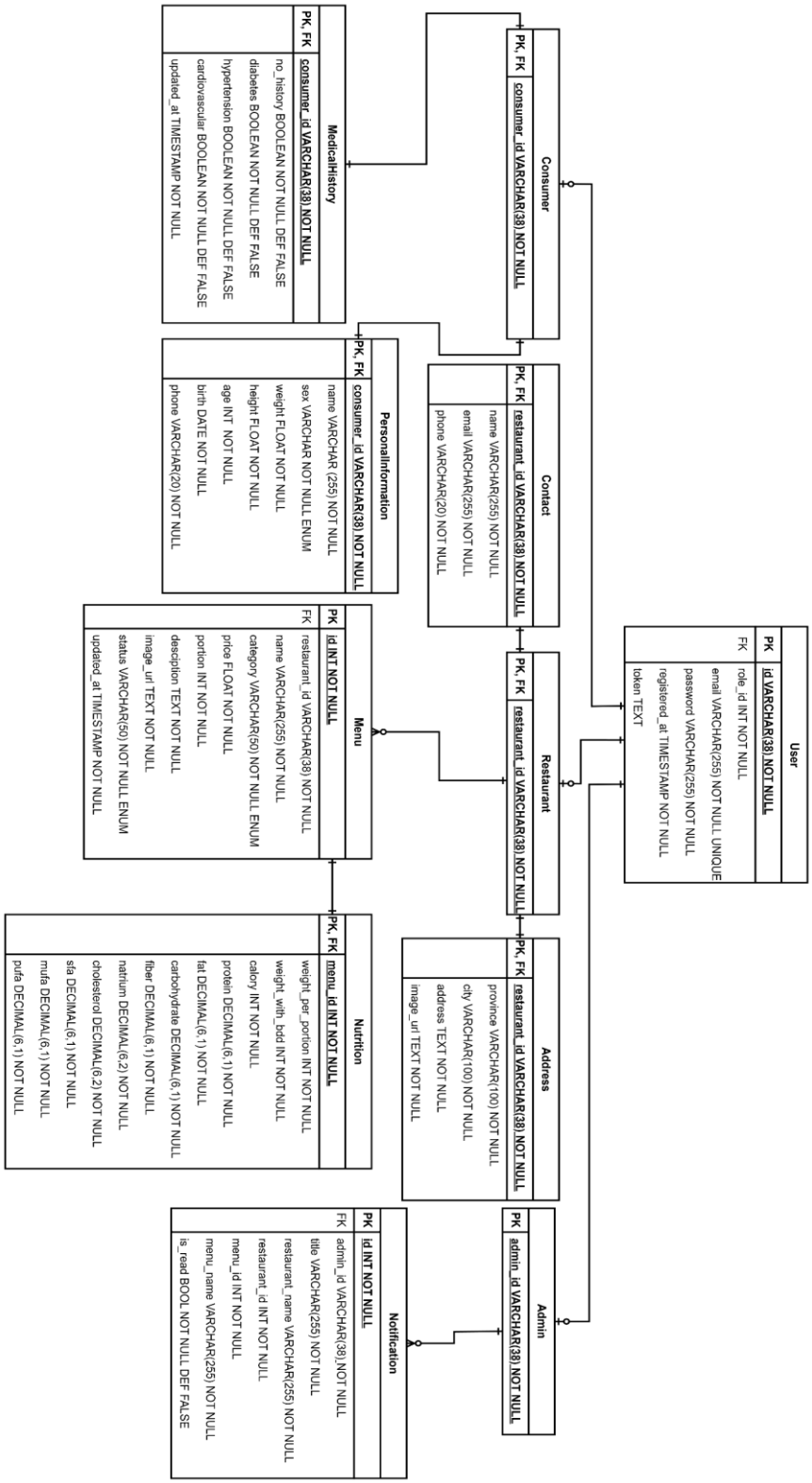
Gambar 4 *Use case diagram* iterasi pertama

4.2.3 Pemodelan Perancangan Cepat (*Modeling Quick Design*)

Class diagram didesain berdasarkan *use case* dan modul/fitur yang akan dikembangkan. Pengembangan *class diagram* disusun berdasarkan atribut yang menggambarkan data yang digunakan pada setiap modul, relasi antar *class* (modul), dan metode yang ada di dalamnya untuk menunjang proses bisnis dari PreciFood. Atribut dan metode yang didesain pada *diagram* tersebut akan menjadi acuan untuk pengembangan *endpoint* API. Pada tahapan ini, *class diagram* dari empat modul (fitur) dikembangkan terlebih dahulu, di antaranya fitur *user*, autentikasi, menu, dan notifikasi. Pada Gambar 5, *class* User memiliki turunan berupa Consumer, Restaurant, dan Admin. *Class* Restaurant memiliki hubungan dengan *class* Menu, yaitu 1 untuk Restaurant dan 0..* untuk Menu. Artinya, setiap restoran dapat memiliki nol atau lebih menu sedangkan setiap menu hanya dimiliki oleh satu restoran.



Gambar 5 *Class diagram* iterasi pertama



Gambar 6 Desain basis data (*database*) iterasi pertama

Perancangan basis data (*database*) dari PreciFood dikembangkan dengan tipe *relational database model* dengan menggunakan *Relational Database Management System* (RDBMS) PostgreSQL. Pada pengembangan ini *Object Relational Mapping* (ORM) Prisma digunakan untuk melakukan *modelling* dan *query database*. Pada iterasi pertama, dihasilkan 11 tabel dengan atribut dan relasinya masing-masing yang dapat dilihat pada Gambar 6 untuk menunjang fitur/modul *user*, autentikasi, menu, dan notifikasi. Desain *database* yang dirancang tidak membatasi pengguna dari restoran hanya terbatas pada Restoran Karimata saja. Desain ini memperbolehkan adanya penambahan restoran lainnya. Oleh karena itu, hubungan antara Tabel User dan Tabel Restaurant ialah *one to zero or many*.

Tabel Menu berisikan atribut di antaranya nama menu, kategori, harga, porsi, deskripsi, gambar, dan statusnya. Kategori dari menu diklasifikasikan ke dalam lima jenis, di antaranya makanan pokok, lauk pauk, sayuran, minuman, dan cemilan (*snack*). Untuk porsi, Restoran Karimata merupakan restoran keluarga sehingga porsi dari menu yang disajikan bervariasi mulai dari individu hingga bersama (*sharing*). Gambar dari menu yang disimpan pada *database* merupakan *image_url* yang didapatkan dari penyimpanan melalui Google Cloud Storage. Selain itu, terdapat hubungan atau relasi seperti pada tabel Menu yang memiliki relasi *one to one* dengan tabel Nutrition, artinya setiap menu memiliki satu informasi kandungan nutrisi berupa *macronutrient* dan *micronutrient*. Tabel Restaurant dan Menu memiliki relasi *one to zero or many* yang artinya setiap restoran dapat memiliki nol atau lebih menu. Pengembangan *database* PreciFood didesain dengan tidak membatasi adanya penambahan restoran dan menu baru.

Desain API kemudian dikembangkan dengan mengikuti arsitektur REST-API. Metode yang digunakan dalam pengembangan API ini di antaranya ialah POST, GET, PUT, PATCH, dan DELETE. Desain API dirancang untuk memenuhi prinsip-prinsip REST, yaitu *stateless*, *resource-based*, dan menggunakan representasi yang konsisten (misalnya JSON) untuk pertukaran data, sehingga memudahkan klien dalam melakukan operasi CRUD (*Create*, *Read*, *Update*, dan *Delete*) pada *resource* yang disediakan. Untuk mengakses resource yang ditujukan, pengguna harus mengirimkan *request* ke *endpoint* spesifik dengan URI yang menggambarkan *resource* tersebut. Metode dari API dirancang dengan acuan pada Tabel 3.

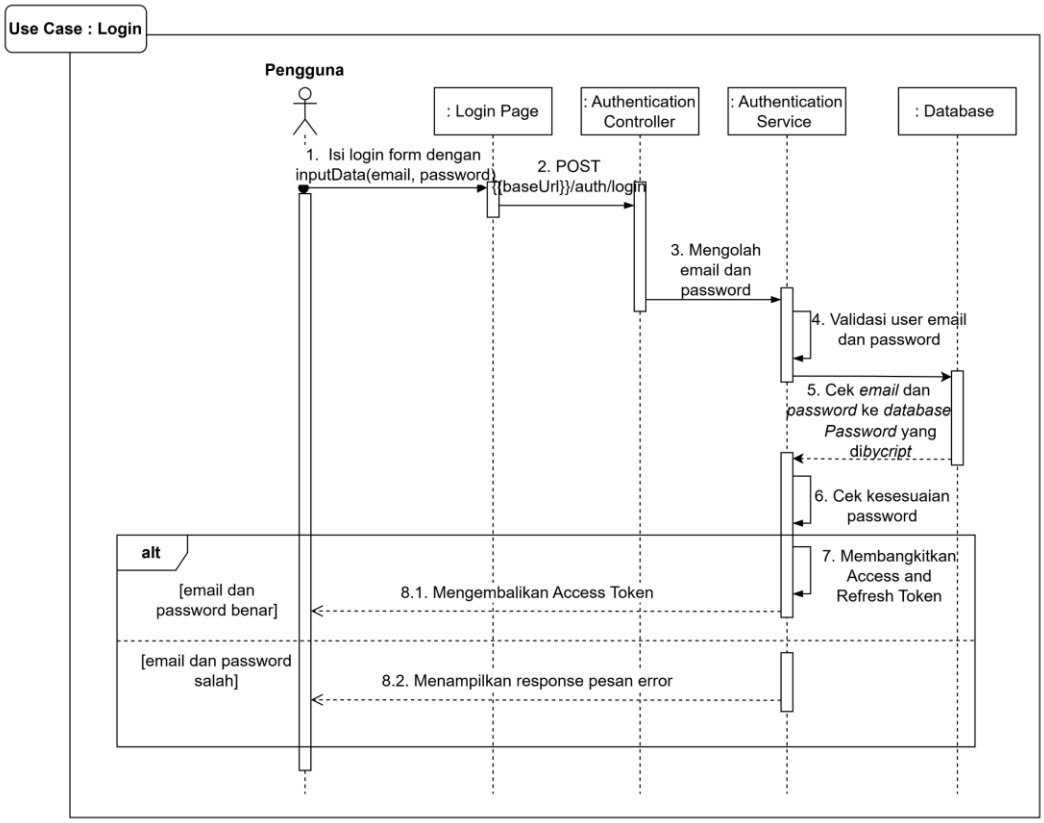
Tabel 3 Acuan penggunaan metode dalam perancangan API

Metode	Keterangan
POST	Membuat *resource* baru. Digunakan untuk menambahkan data baru ke *server* atau *database* (*Create*).
GET	Mengambil *resource* atau daftar *resource*. Digunakan untuk membaca data tanpa mengubah apapun (*Read*).
PUT	Memperbarui seluruh *resource* yang ada. Mengirim representasi penuh dari *resource* untuk menggantikan yang lama (*Fully-Update*).

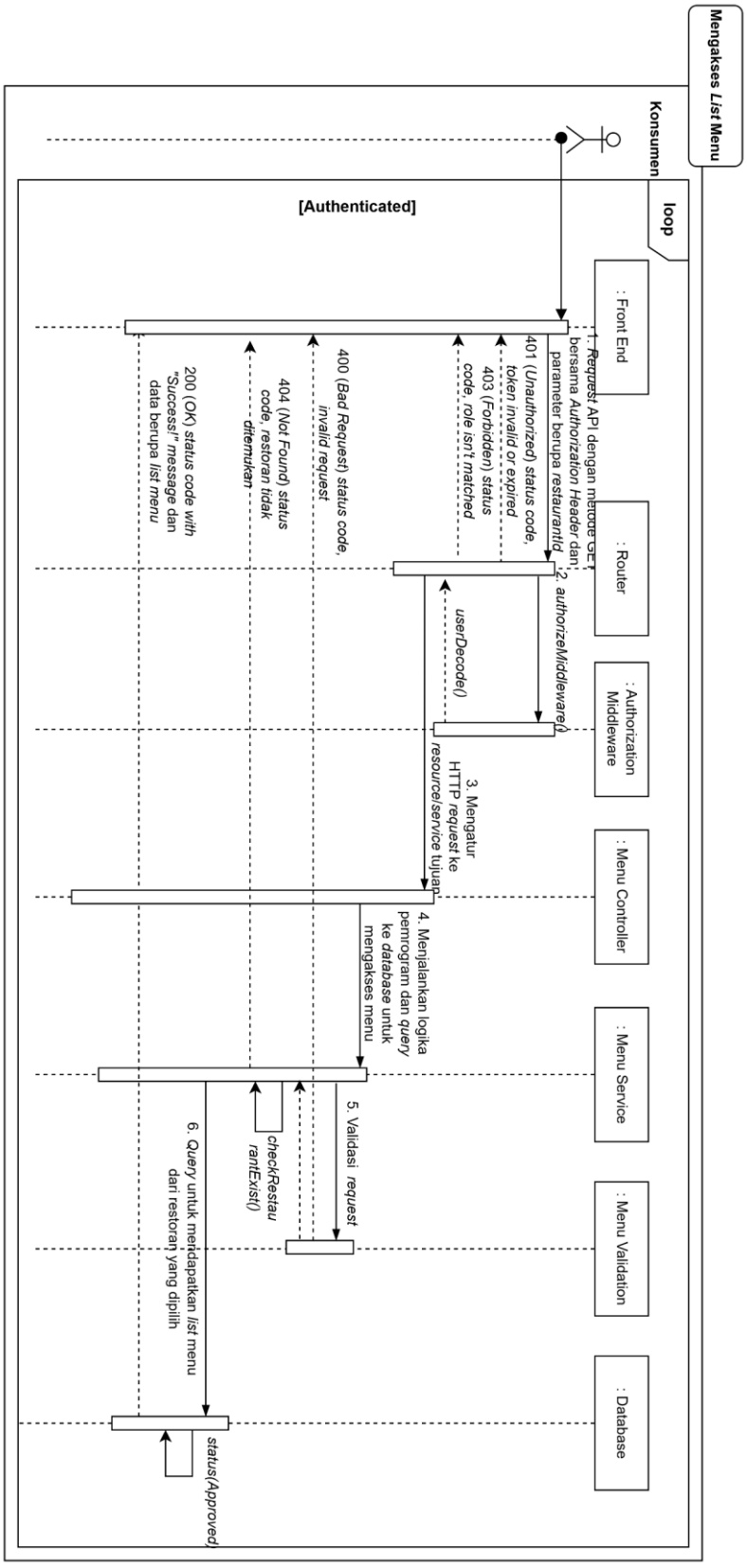
Tabel 3 Acuan penggunaan metode dalam perancangan API (*lanjutan*)

PATCH	Memperbarui sebagian *resource*. Mengirim hanya *field* yang diubah (*Partially-Update*).
DELETE	Menghapus *resource*. Digunakan untuk menghapus data dari *database* (*Delete*).

Sequence diagram kemudian dikembangkan untuk menentukan urutan interaksi antar objek dan aliran data dari setiap *request* pada *endpoint* API yang ada pada *backend app service*. Terdapat dua tipe API, yaitu Publik dan Privat. API yang bersifat Privat artinya harus dilakukan autentikasi terlebih dahulu dengan melalui *login* yang akan didapatkan *access token*. Selanjutnya akan dilakukan otorisasi berdasarkan *access token* yang didapatkan saat *login* yang kemudian akan dikirimkan setiap pengaksesan *endpoint* API yang bertipe Privat. Autentikasi merupakan proses verifikasi identitas pengguna dengan memeriksa kredensial, seperti melalui *email* dan *password.* Tujuannya untuk memastikan bahwa yang melakukan request benar-benar adalah pihak yang terdaftar. Otorisasi merupakan proses pengecekan dan pemberian hak akses (*permissions*) terhadap *resource* atau aksi tertentu berdasarkan peran (*role*) yang dimiliki oleh entitas setelah autentikasi berhasil. Proses *login* dapat dilihat pada Gambar 7.



Gambar 7 *Sequence diagram* untuk *login* pengguna



Gambar 8 *Sequence diagram* pengaksesan *list* menu di suatu restoran

Pada Gambar 8, *sequence diagram* dari pengaksesan menu oleh konsumen dimulai dengan mengirimkan *request* ke *endpoint* API dengan metode GET. *Request* tersebut berisikan *header* yang selanjutnya akan masuk ke dalam *router.* Pada *router, access token* yang dimiliki akan diverifikasi terlebih dahulu untuk menentukan hak akses yang dimiliki dan mengecek apakah JSON Web Token (JWT) yang dimilikinya masih valid. Apabila berhasil, *request* akan diarahkan menuju *controller* hingga *resource service* yang ditujukan. Apabila *request* ini berhasil, *status code* 200 (OK) akan diberikan berserta *list* menu yang tersedia di restoran yang dipilih, yaitu Restoran Karimata.

4.2.4 Pengembangan Prototipe (*Construction of Prototype*)

Tahapan konstruksi pada iterasi pertama ini dimulai pada pengembangan fitur *user*, autentikasi, menu, dan notifikasi pada *backend app service*. *Backend app service* dikembangkan dengan tujuan sebagai *backend* yang akan menangani logika utama (inti) dari aplikasi PreciFood.

Pengembangan *backend* ini mengimplementasikan desain dari *use case, class diagram,* dan basis data (*database*) yang telah dirancang pada tahapan sebelumnya. Pengembangan *backend app service* dilakukan dengan menggunakan bahasa pemrograman TypeScript dengan *framework* ExpressJs dan NodeJs. Basis data (*database*) dibangun dengan menggunakan *Relational Database Management System* (RDBMS) dengan menggunakan PostgreSQL. Prisma digunakan sebagai *Object Relational Mapping* (ORM) untuk melakukan *query* ke *database*.

Modul *user* dikembangkan untuk menunjang manajemen dan akses akun pengguna, seperti registrasi hingga manajemen profil. Registrasi ditujukan kepada konsumen dan restoran yang menjadi mitra. Tahapan registrasi pada konsumen akan menyimpan data berupa variabilitas yang diperlukan untuk menjalankan model GA, di antaranya jenis kelamin, usia, tinggi badan, berat badan, dan riwayat penyakit. Selain itu, data lainnya juga diperlukan dalam registrasi seperti nama, *email*, hingga *password* yang dapat dilihat pada Tabel 4. *Selain* itu, fitur registrasi juga ditujukan bagi pendaftaran restoran yang menjadi mitra dalam pengembangan dengan data yang dibutuhkan pada Tabel 5. Modul *user* juga digunakan untuk melakukan berbagai operasi lainnya yang menunjang kebutuhan pengguna lainnya, seperti *update* profil hingga pengaksesan restoran bagi konsumen dan admin. *Endpoint* API yang dikembangkan dalam modul/fitur *user* dapat dilihat pada Lampiran 3.

Tabel 4 Data konsumen yang diisikan saat registrasi

No	Jenis Data	Tipe Data	Keterangan
1	Nama	*String*	
2	Jenis kelamin	*String*	
3	Riwayat penyakit	*String*	Terbagi menjadi empat, yaitu jantung koroner, diabetes, hipertensi, dan sehat (tidak memiliki riwayat penyakit apapun). Konsumen harus memilih salah satu penyakit paling dominan yang dideritanya.

Tabel 4 Data konsumen yang diisikan saat registrasi (*lanjutan*)

4	Tanggal lahir	*Date*
5	Tinggi badan	*Integer*
6	Berat badan	*Integer*
7	Email	*String*
8	Nomor telepon	*String*
9	*Password*	*String*
10	Konfirmasi *password*	*String*

Tabel 5 Data restoran mitra

No	Jenis Data	Tipe Data
1	Nama restoran	*String*
2	*Email*	*String*
3	Nomor telepon	*String*
4	Provinsi	*String*
5	Kota	*String*
6	Alamat lengkap	*String*
7	Gambar restoran	*BLOB*
8	*Password*	*String*
9	Konfirmasi *password*	*String*

Modul autentikasi selanjutnya dikembangkan agar pengguna dapat melakukan autentikasi dan otorisasi sehingga modul lainnya dapat diakses. Setiap pengaksesan, pengguna melakukan *login* di awal terlebih dahulu dengan menggunakan *email* dan *password* yang telah didaftarkannya pada tahapan registrasi. Pada Lampiran 4 merupakan *endpoint* API yang dihasilkan untuk menunjang kebutuhan modul ini.

Modul menu dikembangkan untuk kebutuhan akses dan manajemen menu bagi pengguna. Konsumen dapat mengakses suatu menu pada restoran yang dipilihnya. Restoran dapat mengakses dan melakukan manajemen terhadap menu yang disajikannya, seperti penambahan, pembaharuan, hingga penghapusan. Data dari menu yang didapatkan melalui restoran dapat dilihat pada Tabel 6. Admin dapat mengakses dan menambahakan informasi kandungan nutrisi dari setiap menu yang telah dianalisisnya. Data nutrisi dari menu yang diisikan oleh admin dapat dilihat pada Tabel 7. Hasil *endpoint* API yang dikembangkan pada modul ini dapat dilihat pada Lampiran 5.

Tabel 6 Data menu yang digunakan dalam pengembangan PreciFood

No	Jenis Data	Tipe Data	Keterangan
1	Nama menu	*String*	
2	Kategori	*String*	Kategori menu terbagi menjadi lima, yaitu makanan pokok, lauk pauk, sayuran, minuman, dan *snack*.
3	Harga	*String*	

Tabel 6 Data menu yang digunakan dalam pengembangan PreciFood (*lanjutan*)

4	Porsi	*String*
5	Deskripsi	*String*
6	Gambar menu	*BLOB*

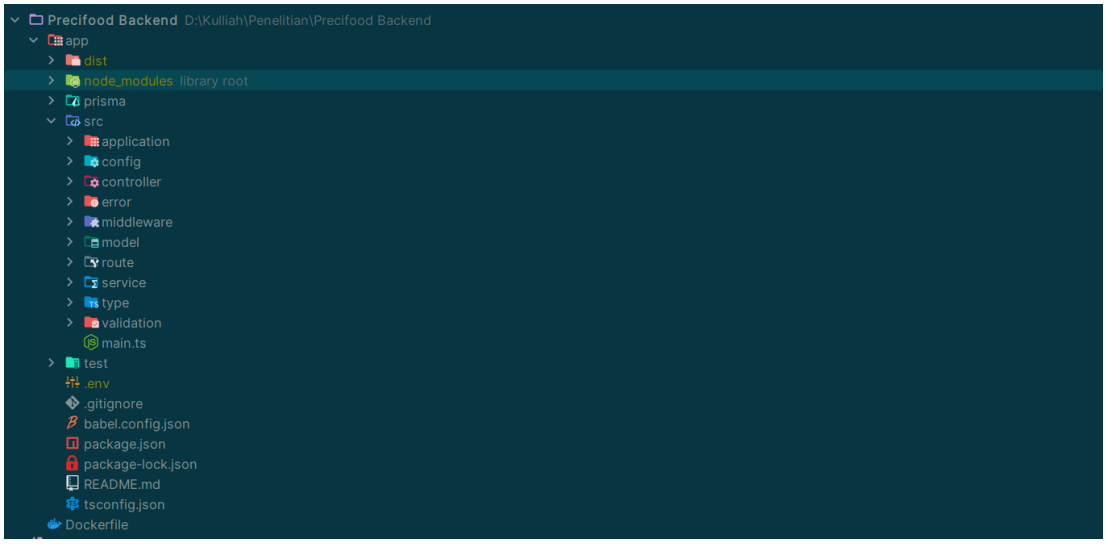
Tabel 7 Data kandungan gizi dari menu

No	Jenis Data	Tipe Data
1	Energi (kalori)	*Integer*
2	Protein	*Decimal*
3	Lemak	*Decimal*
4	Karbohidrat	*Decimal*
5	Lemak	*Decimal*
6	Serat	*Decimal*
7	Natrium	*Decimal*
8	Kolesterol	*Decimal*
9	SFA	*Decimal*
10	MUFA	*Decimal*
11	PUFA	*Decimal*

Modul notifikasi diperuntukkan bagi admin agar mendapatkan informasi terkait adanya penambahan menu pada restoran mitra. Penggunaan modul ini hanya dapat diakses bagi pengguna yang memiliki *role* sebagai admin. Setiap terdapat penambahan menu di restoran mitra maka notifikasi akan dibuat dan disimpan pada *database*. Hasil *endpoint* API yang dikembangkan pada modul ini dapat dilihat pada Lampiran 6.

Status dari *endpoint* yang telah dikembangkan terbagi menjadi dua, yaitu "Publik" dan "Privat". *Endpoint* dari API yang berstatus publik menandakan bahwa dalam pengaksesan *endpoint* tersebut, pengguna tidak perlu melakukan autentikasi terlebih dahulu. Sebaliknya, *endpoint* yang berstatus privat menandakan bahwa pengaksesan *endpoint* memerlukan autentikasi dan otorisasi dari JSON Web Token (JWT) yang dimiliki pengguna dengan mengirimkan *bearer token* berupa *access token* yang disematkan pada *header request*.

Struktur folder dari *backend app service* dirancang dengan memperhatikan modularitasnya agar memudahkan pengembagan. Struktur folder dari *backend* ini diawali dengan *root* yang terdiri dari folder *app* dan beberapa folder pendukung lainnya yang dapat dilihat pada Gambar 9. Logika utama dari *backend app service* terletak dalam folder /app/src yang terbagi ke dalam beberapa folder dan *file* dengan fungsi yang spesifik. Rincian mengenai struktur folder dan deskripsi fungsinya dapat dilihat pada Tabel 8 dan Tabel 9.



Gambar 9 Struktur *folder* dari *backend app service*

Tabel 8 Struktur folder dalam folder src *backend app service*

Folder	Keterangan
application	Konfigurasi utama dari *backend* seperti inisialisasi *server*, *middleware* global, dan pengaturan aplikasi lainnya.
config	Berisikan konfigurasi seperti pengaturan token JWT dan koneksi ke Google Cloud Storage untuk penyimpanan gambar.
controller	Folder yang berisi logika untuk menangani *request* dari *client* dan memanggil *service* yang sesuai.
error	Berisikan definisi *error custom class* yang digunakan untuk memberikan informasi *error* yang lebih jelas dan terstruktur.
middleware	Mengelola autentikasi dengan JWT Token, otorisasi, *error handling*, dan validasi *file input*.
model	Berisikan struktur data dari *request* dan *response* yang digunakan dalam komunikasi antara *client* dan *backend*.
route	Berisikan definisi *endpoint* API yang menghubungkan *request* ke *controller* yang sesuai. Setiap route yang bersifat privat memiliki middleware untuk autentikasi dan otorisasi sebelum diteruskan ke *controller*.
service	Berisikan logika utama dari *backend*, komunikasi dengan *database*, dan eksekusi *query* menggunakan Prisma ORM.
type	Berisikan definisi tipe data berupa peran pengguna (*role*) dan struktur data terkait autentikasi pengguna.
validation	Berisikan validasi input dari *request* yang diterima, memastikan data yang masuk sesuai dengan format yang diharapkan melalui penggunaan Zod.

Tabel 9 *File* di dalam folder service *backend app service*

File	Keterangan
auth-service.ts	Berisikan logika untuk menangani *log in*, *update email, password*, dan *access token* pengguna.
image-uploader-service.ts	Berisikan logika untuk menangani *image storing* ke Google Cloud Storage.
menu-service.ts	Berisikan logika untuk melakukan manajemen menu dan nutrisinya.
notification-service.ts	Berisikan logika untuk mengakses notifikasi dan memperbarui status baca.
user-service.ts	Berisikan logika untuk melakukan resgistrasi dan mengelola data pengguna.

Autentikasi pada sistem mengimplementasikan *token based authentication* dengan menggunakan JSON Web Token (JWT) dilakukan pada pengaksesan setiap *endpoint* API yang sifatnya privat. Sebelum mengakses PreciFood, pengguna akan diminta untuk *log in* terlebih dahulu dengan memasukkan *email* dan *password* yang telah didaftarkannya pada saat registrasi. *Password* yang disimpan dalam *database* dilakukan *hashing* terlebih dahulu dengan menggunakan algoritma *bycrpt* melalui *libary* yang disediakan oleh Node Package Management (NPM) untuk mengamankan kredensial pengguna PreciFood. Pada Gambar 10, dapat dilihat sebelum mengakses *controller*, setiap *request* harus diautentikasi terlebih dahulu melalui *middleware*.

```
privateRouter.get("/api/restaurants/menus",
authorizeMiddleware(Roles.Restaurant),
MenuController.getAllRestaurantMenu);
```

Gambar 10 Contoh *router* dari *endpoint* API yang bersifat privat

Otorisasi dikembangkan dengan menggunakan skema *Role Based Access Control* (RBAC) dengan menggabungkannya ke dalam implementasi JWT Token pada saat pengguna berhasil melakukan autentikasi. *Role* dalam pengembangan RBAC dibagi menjadi konsumen, restoran, dan admin. Otorisasi diterapkan saat pengguna mengakses *endpoint* dari API yang sifatnya privat. Saat pengguna melakukan *log in*, didapatkan *access token* yang akan digunakan setiap pengaksesan API berikutnya. Saat *request* ke *endpoint* API dikirimkan, *request* tersebut akan masuk ke dalam *router* untuk diarahkan ke *resource* yang sesuai. Selanjutnya, dilakukan pengecekan autentikasi dan otorisasi dilakukan secara bersamaan pada fungsi authorizeMiddleware. *Access token* yang ikut dikirimkan berisikan *payload* yang selanjutnya di*decode* untuk membaca jenis *role* yang dimiliki oleh pengguna. Jika autentikasi berhasil dan hasil verifikasi dari *role* telah sesuai dengan *role* yang ditetapkan pada *endpoint* maka *resource* yang menjadi tujuan dapat diakses oleh pengguna.

```
model Menu {

  id            Int        @id @default(autoincrement())
  name          String     @db.VarChar(255)
  category      String     @db.VarChar(50)
  price         Int        @db.Integer
  portion       Int        @db.Integer
  description   String     @db.Text
  image_url     String     @db.Text
  status        String     @default("Waiting") @db.VarChar(50)
  updated_at    DateTime   @updatedAt @db.Timestamp(0)

  restaurant_id String     @db.VarChar(38)
  restaurant    Restaurant @relation(fields: [restaurant_id], references: [restaurant_id], onDelete: Cascade)

  Nutrition     Nutrition?

  @@map("Menu")
}
```

Gambar 11 Pengembangan model skema *database* untuk tabel Menu

Inisialisasi skema *database* dilakukan dengan menggunakan ORM Prisma. Skema dari model *database* yang dikembangkan dibuat pada *file* app/prisma/schema.prisma. *File* ini berisikan deklarasi dari tabel, relasi dari setiap tabel, dan atribut beserta tipe datanya. Prisma juga digunakan untuk menangani *query* dalam melakukan operasi *read*, *write*, *update*, dan *delete* pada *database* melalui Prisma Client. Pada Gambar 11, dapat dilihat pembuatan model skema *database* pada tabel Menu. Kolom pertama merupakan inisialisasi dari nama atribut pada tabel. Kolom kedua berupa tipe data dari atribut untuk menentukan jenis nilai yang dapat disimpan. Kolom ketiga menyimpan atribut (*decorator*) seperti *@id* pada id yang menunjukkan relasi antar tabel (model). Terakhir akan dilakukan *mapping* dari skema model yang telah dibuat ke dalam *database* dengan menggunakan anotasi *@@map*. Perintah *migrate* kemudian dilakukan pada skema model yang telah dikembangkan.

Validasi diterapkan pada setiap *request* yang dikirimkan oleh pengguna dengan salah satu tujuannya ialah untuk memastikan konsistensi data yang disimpan pada *database*. Pertama, *request* JSON divalidasi dengan menggunakan Zod sebagai *library* untuk melakukan validasi pada setiap *request* yang dikirimkan. Pada Gambar 12 merupakan contoh validasi yang dilakukan pada *request* penambahan menu baru yang dilakukan oleh restoran. Validasi tersebut di antaranya nama menu yang harus lebih dari satu karakter dan maksimal 255 karakter, kategori menu yang dibatasi hanya berupa makanan pokok, lauk pauk, minuman, sayuran, atau cemilan (*snack*), harga dan porsi menu yang harus bernilai positif, hingga deskripsi yang harus diisikan.

```
export class MenuValidation {
    static readonly CREATEMENU: ZodType = z.object({
        restaurant_id: z.string().trim().min(38, "ID restoran tidak
        valid").max(38, "ID restoran tidak valid"),
        name: z.string().trim().min(1, "Nama menu minimal 1
          karakter").max(255, "Nama menu maksimal 255 karakter"),
        category: z.enum(["Makanan Pokok", "Lauk Pauk", "Minuman",
                  "Sayuran", "Snack"], {errorMap: () => ({ message:
                   "Kategori harus bernilai 'Makanan Pokok', 'Lauk
                    Pauk', 'Minuman', 'Sayuran', atau 'Snack'" })
```

Gambar 12 Potongan kode validasi dengan *library* Zod terhadap *request* penambahan menu

```
}),
        price: z.string().transform((val) => parseInt(val,
               10)).refine(val => !isNaN(val) && val > 0, {
               message: "Harga harus bernilai positif dan berupa angka"
        }),
        portion: z.string().transform((val) => parseInt(val,
                  10)).refine(val => !isNaN(val) &&
                 val > 0, {
                 message: "Porsi harus bernilai positif dan berupa
                           angka"
        }),
        description: z.string().trim().min(1, "Deskripsi tidak boleh
                     kosong"),
        image_url: z.string().trim().min(1, "URL gambar tidak boleh
                   kosong")
    });
. . .
}
```

Gambar 12 Potongan kode validasi dengan *library* Zod terhadap *request* penambahan menu (*lanjutan*)

Validasi juga dilakukan terhadap data BLOB (*Binary Large Object*) berupa *file* dari gambar menu dan profil restoran pada Gambar 13. Validasi dilakukan dengan menggunakan *library* Multer yang disediakan oleh NPM. Validasi ini memastikan bahwa *file* yang dikirim oleh pengguna merupakan gambar dengan format PNG, JPG, atau JPEG. Ukuran *file* juga dibatasi sebesar 5 MB. Selain itu, pembatasan jumlah *file* yang dikirimkan juga dibatasi dengan hanya mengirimkan satu *file* (*single*) pada setiap *request* yang dikirimkan.

```
export const multerMiddleware = multer({
    storage: multer.memoryStorage(),
    limits: {
        fileSize: 5 * 1024 * 1024,
    },
    fileFilter(_,      file:      Express.Multer.File,      callback:
multer.FileFilterCallback) {
        console.log("File    received    in    multer    middleware:",
file.originalname);
        const    allowedMimeTypes    =    ["image/jpeg",    "image/jpg",
"image/png"];

        if (allowedMimeTypes.includes(file.mimetype)) {
            callback(null, true);
        } else {
            callback(new ResponseError(400, "Hanya file dengan format
.jpg, .jpeg, dan
            .png yang diperbolehkan!"));
        }
    },
}).single("image");
```

Gambar 13 Validasi dengan Multer terhadap *image file* yang diunggah

Validasi dari *request* yang tidak valid akan mengembalikan *response* dengan *status code* 400 (*Bad Request*). Selain itu, terdapat *message error* yang telah disesuaikan agar pengguna mengetahui di mana *error* tersebut terjadi. Gambar 14 merupakan contoh *request* yang tidak valid di mana *email* yang dikirim oleh pengguna tidak sesuai dengan ketentuan format. *Response* yang dihasilkan akan berupa *status code* 400 (*Bad Request*) dengan pesan *error* yang tertera pada Gambar 15. Penerapan validasi ini bertujuan untuk menjaga konsistensi dari data yang disimpan dan mencegah beban (*load*) yang berlebihan apabila terjadi pengiriman *request* yang tidak sesuai.

```
{
    "name": "Joana",
    "email": "joanatest",
    "sex": "Perempuan",
    "birth": "1975-09-22",
    "phone": "085812340000",
    "height": 170,
    "weight": 60,
    "medical_history": "no_history",
    "password": "xty12345",
    "password_confirmation": "xty12345"
}
```

Gambar 14 Contoh *request* dengan validasi yang tidak sesuai

Gambar 15 Contoh *response* pada *request* dengan validasi yang gagal

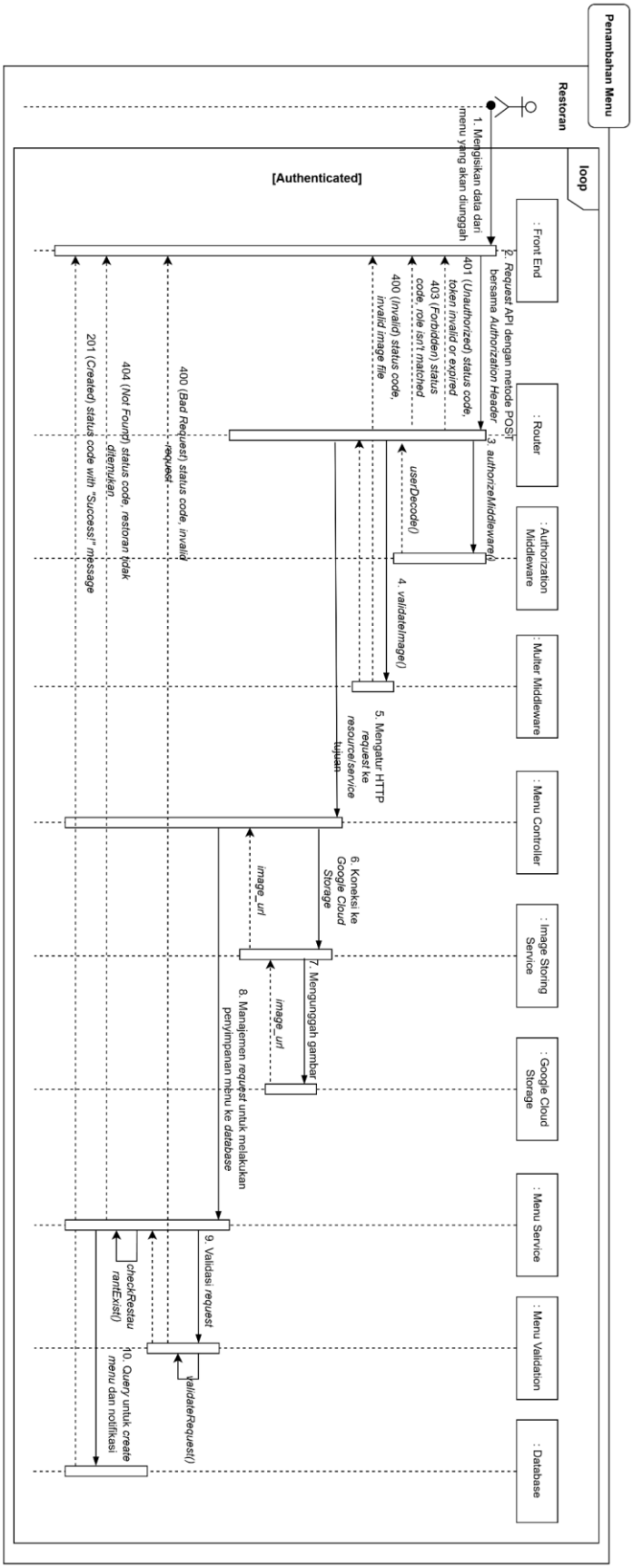
Pengaksesan aplikasi PreciFood dimulai dengan tahapan registrasi bagi konsumen yang belum memiliki akun. Konsumen akan melakukan registrasi dengan mengisikan data seperti pada Tabel 3. Pengiriman *request* untuk registrasi dilakukan melalui *endpoint* API /api/signup/consumer dengan metode POST. *Request* yang masuk selanjutnya akan dikelola oleh *router* terlebih dahulu untuk diarahkan pada *resource* yang sesuai. Data pada *request* selanjutnya akan divalidasi dengan *library* Zod. Apabila validasi berhasil maka *request* tersebut nantinya akan masuk ke dalam *service* dan akan disimpan pada *database*. *Request* dari *endpoint* API registrasi ini tidak memerlukan autentikasi dan otorisasi karena dapat diakses oleh seluruh pengguna baik yang belum memiliki akun. Oleh karena itu sifat dari *endpoint* API ini adalah publik.

Pengaksesan modul autentikasi selanjutnya dilakukan dengan melakukan *login* terlebih dahulu. Untuk melakukan *login*, pengguna harus memasukkan *email* dan *password* yang telah disimpan pada *database*. *Request* akan dikirimkan ke *endpoint* API /api/auth/login dengan metode POST. Apabila pengguna berhasil terautentikasi maka *access token* akan diberikan untuk

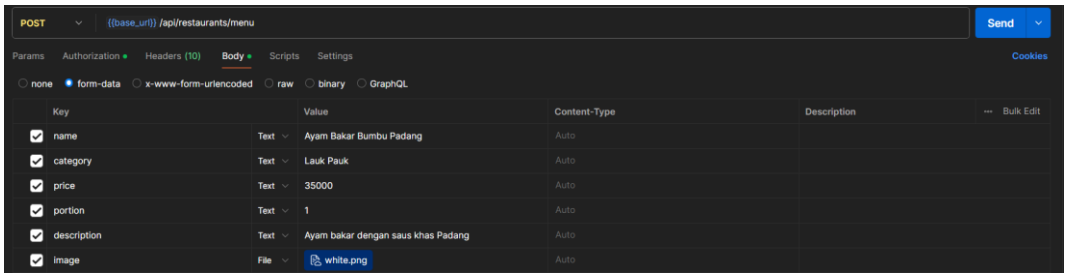
mengakses fitur/modul lainnya. *Endpoint* API untuk *login* bersifat publik sehingga pengguna dapat mengirimkannya tanpa perlu melakukan autentikasi terlebih dahulu.

Pengaksesan menu oleh konsumen dapat dilakukan apabila fitur menu berhasil dikembangkan. Fitur menu dimulai dengan melakukan pengembangan *endpoint* API POST /api/restaurants/menu untuk mengunggah menu yang disajikan oleh restoran. Untuk mengunggah satu menu, terdapat input yang perlu diunggah oleh restoran yang dapat dilihat pada Tabel 5. Selain itu, pada *request* yang dikirimkan juga tersimpan *header* berupa *access token* untuk melakukan validasi JSON Web Token untuk melakukan autentikasi dan otorisasi. *Terdapat input* berupa gambar dari menu yang diunggah. Gambar yang masuk akan divalidasi terlebih dahulu dengan menggunakan *library* Multer. Pada proses ini, penggunaan Google Cloud Storage juga diterapkan untuk penyimpanan objek (*object storage*) berupa gambar. Google Cloud Storage (GCS) digunakan sebagai *object storage* dalam lingkungan *serverless* karena menyediakan penyimpanan yang *scalable*, *durable*, dan dapat diakses secara global. Hal ini dikarenakan penggunaan Cloud Run sebagai layanan *deployment* yang merupakan layanan *serverless* menyebabkan data tidak dapat disimpan secara lokal. Penyimpanan lokal dalam jangka panjang seperti SSD tidak tersedia dalam layanan *serverless* sehingga penggunaan *cloud storage* seperti GCS harus diimplementasikan untuk melakukan penyimpanan gambar.

Setiap pengunggahan gambar yang berhasil, *return* berupa *image_url* akan didapatkan yang kemudian akan ikut disimpan ke dalam *database*. Selanjutnya, *request* akan masuk ke *controller* dan *service* untuk dilakukan manajemen penambahan menu. *Request* juga akan divalidasi saat masuk ke dalam *service* dengan menggunakan *library* Zod seperti Gambar 16. Selain itu, akan dilakukan pengecekan juga apakah *file* dari gambar menu dan profil restoran. Validasi dilakukan dengan menggunakan *library* Multer yang disediakan oleh NPM. Validasi ini memastikan bahwa *file* yang dikirim oleh pengguna merupakan gambar dengan format PNG, JPG, atau JPEG. Ukuran *file* juga dibatasi sebesar 5 MB. Selain itu, pembatasan jumlah *file* yang dikirimkan juga dilakukan dengan batasan satu *file* (*single*) setiap *request* yang dikirimkan. Apabila validasi dari *request* berhasil dan restoran tersebut tercatat dalam *database* maka *response* dengan *status code* 201 (*Created*) dan *message* berupa “Success!” akan didapatkan. Hal ini menunjukkan bahwa menu berhasil dibuat dan tersimpan pada *database*.



Gambar 16 *Sequence diagram* penambahan menu oleh restoran



Gambar 17 Struktur *request* pengunggahan menu oleh restoran

Setelah pengunggahan menu berhasil, menu tersebut akan tersimpan dalam *database* dan memiliki atribut status berupa "*Waiting*" secara *default*. Hal ini menandakan bahwa menu yang diunggah oleh restoran masih belum memiliki data terkait kandungan nutrisi berupa *macronutrient* dan *micronutrient*, juga belum mendapatkan persetujuan dari admin. Setiap pengunggahan menu baru oleh restoran, notifikasi akan didapatkan admin. Pada Gambar 18, dapat dilihat struktur JSON *response* dari *list* notifikasi penambahan menu yang didapatkan oleh admin. Fitur notifikasi ini tentunya akan memudahkan bagi admin untuk mengetahui adanya penambahan menu sehingga komunikasi lebih lanjut dengan pihak restoran dapat dilakukan untuk menganalisis kandungan nutrisi yang ada di dalam menu.


```
{
    "message": "Success!",
    "data": [
        {
            "id": 1,
            "title": "Menu baru ditambahkan!",
            "restaurant_name": "Restoran Karimata",
            "restaurant_id": "R-0192c857-35e6-7cc2-98da-46f0faf6f651",
            "menu_id": 83,
            "menu_name": " Ayam Goreng",
            "is_read": false
        },
        {
            "id": 2,
            "title": "Menu baru ditambahkan!",
            "restaurant_name": "Restoran Karimata Fix",
            "restaurant_id": "R-01944401-4391-7bb2-9c9b-2b7d5e983f5e",
            "menu_id": 84,
            "menu_name": "Ayam Bakar Bumbu Padang",
            "is_read": false
        }
    ]
}
```


Gambar 18 *Response* GET *list* notifikasi penambahan menu yang didapatkan admin

Pembuatan informasi kandungan nutrisi selanjutnya dapat dilakukan oleh admin dengan mengirimkan *request* ke *endpoint*. Setiap pengunggahan menu baru yang dilakukan oleh restoran, notifikasi akan dikirimkan kepada admin. Selanjutnya, akan dilakukan komunikasi antara restoran dan admin untuk

menganalisis kandungan nutrisi dari menu. Admin selanjutnya dapat memasukkan kandungan nutrisi dari setiap menu dengan mengirimkan *request* ke *endpoint* API POST Admin dapat memutuskan status dari menu tersebut, di antaranya "*Approved*", "*Waiting*", atau "*Rejected*". Pengisian nilai kandungan nutrisi dari suatu menu dilakukan dengan mengirimkan *request* dengan metode POST ke *endpoint* API /api/restaurants/:restaurantId/menus/:menuId/nutrition dengan *request body* pada Gambar 19.


```
{
    "weight_per_portion": 398,
    "weight_with_bdd": 252,
    "calory": 644,
    "protein": 88.3,
    "fat": 27.9,
    "carbohydrate": 4.6,
    "fiber": 0,
    "natrium": 880.82,
    "cholesterol": 419.2,
    "sfa": 20.9,
    "mufa": 19.8,
    "pufa": 8.4
}
```


Gambar 19 Contoh *request body* dari pengisian kandungan nutrisi suatu menu

Pengguna kemudian dapat mengakses *list* menu yang ada di suatu restoran. Bagi konsumen dan admin, akses terhadap *list* menu dapat dilakukan dengan pengiriman *request* dengan metode GET ke *endpoint* API /api/restaurants/:restaurantId/menus. Parameter berupa restaurantId digunakan untuk mengarahkan konsumen agar mengakses *list* menu dari restoran yang dipilihnya. Pada Gambar 20 didapatkan *list* menu dari Restoran Karimata. Untuk mengakses detail dari suatu menu parameter berupa menuId juga disematkan melalui pengiriman *request* dengan metode GET ke *endpoint* /api/restaurants/:restaurantId/menus/:menuId. Gambar 21 menunjukkan salah satu detail menu yang dipilih bagi pengguna dengan *role* konsumen.


```
{
    "message": "Success!",
    "data": [
        {
            "id": 3,
            "name": "Gurame Bakar Kecap",
            "status": "Approved",
            "price": 135000,
            "portion": 3,
            "category": "Lauk Pauk",
               "description": "Ikan gurame bakar yang dibalut kecap manis,
menghasilkan rasa gurih dan manis sempurna.",
            "image_url": "https://storage.googleapis.com/precifood-image/menu-
images/1731037163001_Gurame-Bakar-e1645604750351.png"
        },
        {
            "id": 4,
            "name": "Patin Goreng Kremes",
            "status": "Rejected",
```


Gambar 20 Potongan JSON *response* dari *list* menu Restoran Karimata yang diakses oleh konsumen

```
"price": 95000,
            "portion": 4,
            "category": "Lauk Pauk",
             "description": "Patin yang digoreng renyah dengan tambahan kremes
yang gurih dan kriuk.",
            "image_url": "https://storage.googleapis.com/precifood-image/menu-
images/1731037246781_patin-goreng-kremes-e1645603256995.png"
        },
        {
            "id": 5,
            "name": "Gurame Goreng Honje",
            "status": "Approved",
            "price": 140000,
            "portion": 3,
            "category": "Lauk Pauk",
              "description": "Gurame goreng dengan cita rasa unik dari bunga
kecombrang atau honje.",
            "image_url": "https://storage.googleapis.com/precifood-image/menu-
images/1731037298634_Gurame-goreng-honje-e1645603859988.png"
        }
 ...
]
```


Gambar 20 Potongan JSON *response* dari *list* menu Restoran Karimata yang diakses oleh konsumen (*lanjutan*)


```
{
    "message": "Success!",
    "data": {
        "id": 3,
        "name": "Gurame Bakar Kecap",
        "status": "Approved",
        "price": 135000,
        "portion": 3,
        "category": "Lauk Pauk",
        "description": "Ikan gurame bakar yang dibalut kecap manis, menghasilkan rasa gurih dan manis sempurna.",
        "image_url": "https://storage.googleapis.com/precifood-image/menu-images/1731037163001_Gurame-Bakar-e1645604750351.png",
        "nutrition": {
            "weight_per_portion": 398,
            "weight_with_bdd": 252,
            "calory": 644,
            "protein": 88.3,
            "fat": 27.9,
            "carbohydrate": 4.6,
            "fiber": 0,
            "natrium": 880.82,
            "cholesterol": 419.2,
            "sfa": 20.9,
            "mufa": 19.8,
            "pufa": 8.4
        }
    }
}
```

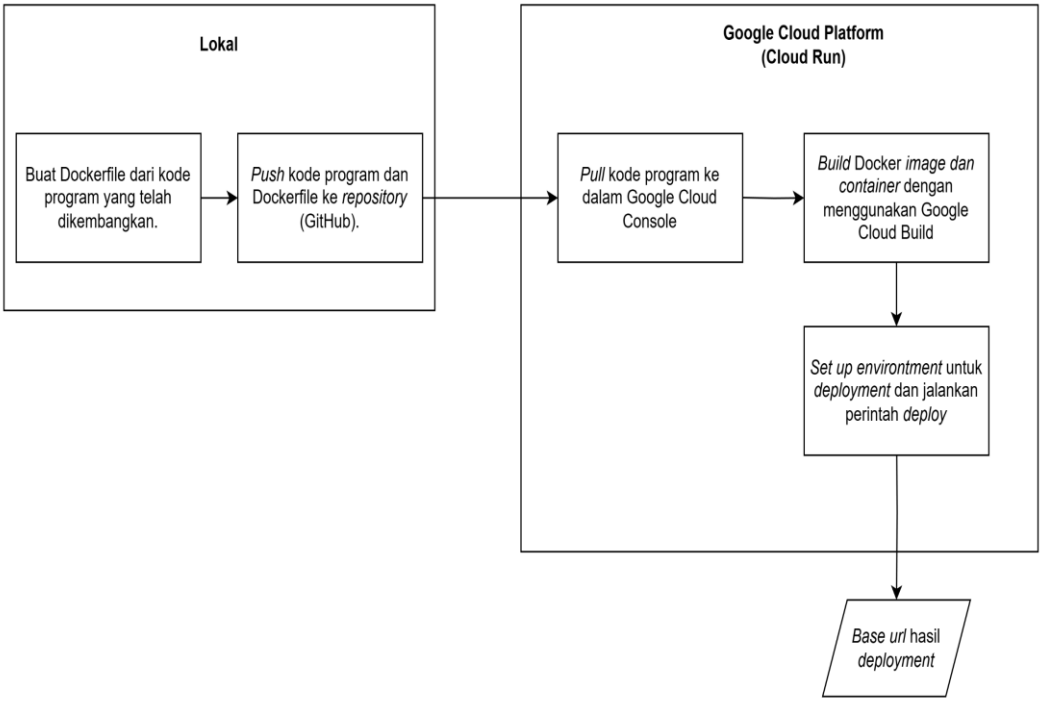

Gambar 21 JSON *response* dari detail menu

4.2.5 Penyebaran dan Umpan Balik (*Deployment Delivery and Feedback*)

Pengujian dengan metode *blackbox testing* dilakukan untuk memastikan fungsionalitas dari setiap *endpoint* API berjalan dengan benar dan sesuai. *Blackbox testing* dilakukan secara manual dengan menggunakan Postman. Dilakukan uji dengan mengimplementasikan dua skenario, yaitu skenario positif dan negatif. Skenario positif dilakukan melalui pengujian *test case* dengan input yang valid. Skenario negatif dilakukan melalui pengujian dengan *test case* input yang tidak valid. Apabila *request* yang dikirimkan valid maka data yang *response* yang dihasilkan juga valid. Apabila *request* yang dikirimkan tidak valid maka *response error* akan didapatkan. Hasilnya, didapatkan 29 *endpoint* API dengan hasil uji yang berhasil yang dapat dilihat pada Lampiran 10. Hasil

uji dikatakan berhasil apabila *endpoint* dari API dapat memberikan *response* yang sesuai dari *request* yang diberikan. Hasil dan skenario pengujian dapat dilihat pada Lampiran 1.

Deployment kemudian dilakukan setelah pengujian dengan *blackbox testing* selesai. *Deployment* pada tahapan ini dilakukan terhadap *backend app service* dan *database* hasil iterasi pertama dengan mekanisme yang dapat dilihat pada Gambar 22. *Deployment* dilakukan pada *database* terlebih dahulu. Dilakukan *hosting database* dengan menggunakan layanan dari Neo. Selanjtunya pada *backend app service*, *deployment* dilakukan melalui Google Cloud Platform (GCP) dengan layanan Cloud Run. Cloud Run merupakan layanan *serverless* yang memungkinkan dilakukannya *deployment* dari *image container* aplikasi atau kode program yang telah dikembangkan. Pembuatan Dockerfile diperlukan sesuai dengan *set up* dari *backend* yang dibuat. Selanjutnya, *code base* dari *backend* dan Dockerfile di*push* ke *repository* GitHub. Kemudian, *code base* dan Dockerfile dari *repository* GitHub akan di*pull* ke Google Cloud Console. Docker *image* dan *container* kemudian dibangun dengan menggunakan Cloud Build. Kemudian, dilakukan konfigurasi untuk melakukan *deployment* pada layanan *serverless*, seperti pemilihan *region deployment*, jumlah *virtual* CPU yang digunakan, hingga besaran memorinya yang berupa RAM (*Random Access Memory*). *Base url* yang dihasilkan melalui *deployment* Cloud Run secara otomatis menggunakan protokol HTTPS sehingga keamanan dari data yang dikirimkan antara pengguna dan *server* menjadi aman dan terlindungi. *Base url* akan didapatkan setelah *deployment* berhasil. Kemudian, *base url* dari *backend app service* ini yang akan digunakan untuk integrasi pada *frontend*.



Gambar 22 Tahapan *deployment* dengan layanan Cloud Run.

Feedback yang didapatkan pada iterasi pertama terdapat pada informasi yang didapatkan pengguna dari pengaksesan menu. Pada pengguna dengan

peran konsumen dan restoran, kandungan nutrisi dari menu yang dapat mereka akses berupa nutrisi yang sifatnya makro. Sebaliknya, pada pengguna admin kandungan nutrisi yang dapat diakses dari setiap menu merupakan kandungan *macronutrient* dan *micronutrient*.

4.3 Iterasi 2

4.3.1 Komunikasi (*Communication*)

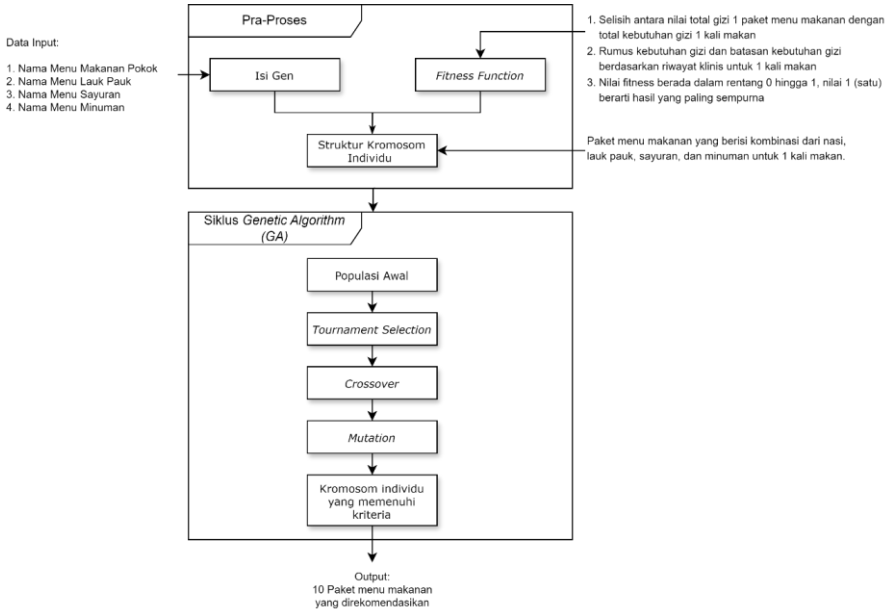
Iterasi kedua merupakan tahapan yang dilakukan untuk melakukan pengembangan *backend model*. *Backend model* dibangun agar sistem rekomendasi berupa model GA yang telah dikembangkan pada penelitian sebelumnya dapat dijalankan melalui *Application Programming Interface* (API) sehingga dapat diintegrasikan dengan komponen sistem lainnya. Berdasarkan hasil komunikasi dan diskusi dengan *stakeholder* yaitu para peneliti yang melakukan pengembangan model GA, terdapat beberapa perubahan yang harus dilakukan pada model GA yang dikembangkan. Pertama, model GA masih berkestensi Ipython Notebook (*.ipynb) sehingga harus dilakukan konversi ke format Python (.py). Kedua, terdapat parameter input yang harus diperhatikan untuk menjalankan model ini, yaitu input dari konsumen dan menu beserta nutrisi yang tersedia. Ketiga, pada model pembacaan parameter input tersebut masih melalui *file* dengan format CSV sehingga harus dilakukan perubahan *parsing* data dengan membacanya melalui *database*.

Tahapan ini juga dipelajari secara umum bagaimana proses *generate* rekomendasi menu dari model berbasis GA dilakukan. Tahapan dimulai dengan praproses, yang terdiri dari penyusunan gen kromosom dan perhitungan *fitness function*. Kromosom tersusun atas 13 gen. Empat gen pertama merupakan kombinasi dari kategori menu, di antaranya makanan pokok, lauk pauk, sayuran, dan minuman. Gen 5 hingga 9 merupakan representasi dari kandungan *macronutrient.* Gen 10 hingga 13 merupakan representasi dari kandungan *micronutrient*. Susunan gen dalam kromosom yang dikonstruksi untuk menjalankan model GA dapat dilihat pada Gambar 23. Perhitungan kebutuhan gizi dari individu spesifik juga akan dihitung pada model dan dijadikan fungsi *fitness*. Alur dari jalannya model ini mulai dari tahapan praproses hingga menghasilkan *output* dapat dilihat pada Gambar 24.

Gen 1	Gen 2	Gen 3	Gen 4	Gen 5	Gen 6	Gen 7	Gen 8	Gen 9	Gen 10	Gen 11	Gen 12	Gen 13
Makanan Pokok	**Lauk Pauk**	**Sayur**	**Minuman**	**Energi (kcal)**	**Protein (gram)**	**Lemak (gram)**	**Karbohidrat (gram)**	**Natrium (mg)**	**Kolesterol (mg)**	**MUFA (gram)**	**PUFA (gram)**	**SFA (gram)**
Nasi	Patin bakar kecap	Sup tahu + sayuran	Jus semangka	868	48.4	20.9	167.8	717.42	146.63	9.13	5.93	12.40

Gambar 23 Struktur kromosom yang dibentuk dan digunakan untuk perhitungan rekomendasi (Seminar *et al.* 2025)

\



Gambar 24 Alur model GA untuk rekomendasi paket menu makanan (Seminar *et al*. 2025)

Menu yang akan dianalisis dan direkomendasikan kepada konsumen dalam penelitian berasal dari Restoran Karimata yang menjadi mitra pengembangan. Terdapat 82 jenis menu yang telah dianalisis kandungan nutrisinya, baik *macronutrient* dan *micronutrient.* Atribut dari menu dan kandungan nutrisinya tersebut telah berhasil disimpan pada *database* di iterasi pertama. Pada Gambar 24, dihasilkan rekomendasi dari model GA yang merupakan konsumen dengan variabilitas laki-laki sehat, berusia 30 tahun, tinggi dan berat badan masing-masing 165 cm dan 70 kg.

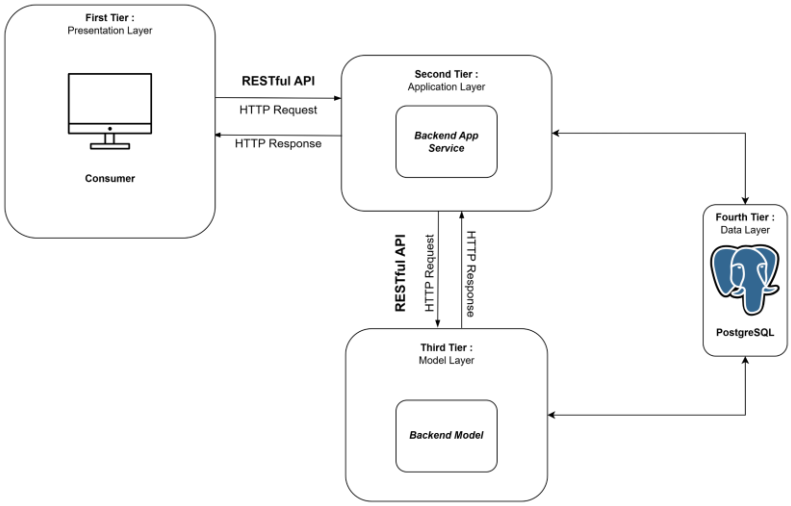
```
#L_sehat
# Gabungkan dua DataFrame berdasarkan kolom "Rekomendasi"
combined_df = pd.merge(food_df, nutritional_sums_df, on='Rekomendasi')

# Tampilkan DataFrame yang sudah digabungkan
combined_df
```

	Rekomendasi	makanan pokok	lauk pauk	sayur	minum	Energi (KKal)	Protein (g)	Lemak (g)	Karbohidrat (g)	Natrium (mg)	Kolesterol (mg)	MUFA (g)	PUFA (g)	SFA (g)	Fitness
0	Paket Menu 1	Nasi	Patin bakar kecap	Sup tahu + sayuran	Jus semangka	868	48.4	20.9	167.8	717.42	146.63	9.13	5.93	12.40	0.902321
1	Paket Menu 2	Nasi	Ayam goreng kremes	Sup sayuran	Jus mangga	846	22.8	34.9	158.3	756.66	109.66	10.87	5.63	19.70	0.849104
2	Paket Menu 3	Nasi	Patin bakar kecap	Brocoli bawang putih	Jus wortel	837	39.5	14.1	182.4	734.00	139.73	7.30	3.23	10.07	0.843461
3	Paket Menu 4	Nasi	Patin bakar dalam bambu	Terong sambal hijau	Jus stamina	990	52.0	38.7	157.2	717.52	82.50	1.29	2.40	9.07	0.828281
4	Paket Menu 5	Nasi	Dori crispy	Terong ikan asin	Es cendol	850	22.0	26.3	179.2	630.42	33.27	2.37	1.33	9.93	0.791956
5	Paket Menu 6	Nasi	Patin bakar kecap	Karedok	Jus munser	977	47.4	30.7	176.9	671.61	144.93	12.40	6.47	16.10	0.789451
6	Paket Menu 7	Nasi	Gurame goreng asam pedas	Cah kangkung terasi	Jus nanas	1007	48.4	25.4	185.7	748.85	131.60	8.97	6.60	24.03	0.761821
7	Paket Menu 8	Tanpa Nasi	Gurame goreng asam pedas	Sup tahu + sayuran	Jus glucoless	815	45.2	34.6	82.0	790.62	137.77	10.43	9.03	26.10	0.749247
8	Paket Menu 9	Nasi	Sup gurame asam kemangi	Sup sayuran	Es lemon	794	41.2	9.2	179.7	652.15	137.77	6.27	3.53	6.90	0.745711
9	Paket Menu 10	Nasi	Udang saus asam pedas	Sup tahu + sayuran	Jus stamina	769	33.1	23.7	156.4	625.54	75.90	3.23	4.33	8.00	0.744639

Gambar 25 Hasil rekomendasi *set* menu hasil komputasi model GA untuk pilihan menu di Restoran Karimata (Seminar *et al.* 2025)

4.3.2 Perencanaan Cepat (*Quick Planning*)

Perencanaan selanjutnya dilakukan untuk menentukan bagaimana model GA dapat diintegrasikan dan dijalankan untuk menghasilkan rekomendasi. Berdasarkan hasil diskusi, model GA akan dijadikan salah satu *service* pada *backend model*. Melalui arsitektur yang telah ditetapkan dalam pengembangan sistem yaitu *multi-tier*, *backend model* menjadi lapisan (*layer*) tersendiri yang akan menjalankan komputasi model GA untuk menghasilkan rekomendasi. Komunikasi dengan lapisan ini akan melalui pengiriman HTTP *request* dengan arsitektur REST API melalui *backend app service* ke *backend model*. Oleh karena itu, diperlukan adanya pengembangan *endpoint* API dari sisi *backend model* untuk menjalankan (*trigger*) komputasi model GA. Selain itu, *backend model* juga akan mengkonsumsi data input yang diperlukan untuk menjalankan model GA dari *database* yang sama dengan *backend app service* sehingga dihasilkan arsitektur integrasi sistem pada Gambar 26. Untuk memastikan bahwa *backend model* berhasil dibangun, pengembangan skema *database* untuk input model GA berupa data konsumen dan menu beserta nutrisinya harus dipastikan berhasil dibangun terlebih dahulu.



Gambar 26 Arsitektur integrasi sistem

Berdasarkan komunikasi dirumuskan input data yang dibutuhkan untuk menjalankan model GA yang dapat dilihat pada Tabel 10.

Tabel 10 Input data dari konsumen dan menu untuk menjalankan model GA

Sumber Data	**Jenis Data**	**Tipe Data**
Konsumen	Jenis kelamin	*String*
	Usia	*Integer*
	Tinggi badan	*Integer*
	Berat badan	*Integer*
	Riwayat penyakit	*String*

Tabel 10 Input data dari konsumen dan menu untuk menjalankan model GA (*lanjutan*)

Menu	Makanan pokok	*Integer*
	Lauk pauk	*Integer*
	Sayur	*Integer*
	Minuman	*Integer*
	Kalori (energi)	*Integer*
	Protein	*Decimal*
	Lemak	*Decimal*
	Karbohidrat	*Decimal*
	Natrium	*Decimal*
	Kolesterol	*Decimal*
	MUFA	*Decimal*
	PUFA	*Decimal*
	SFA	*Decimal*

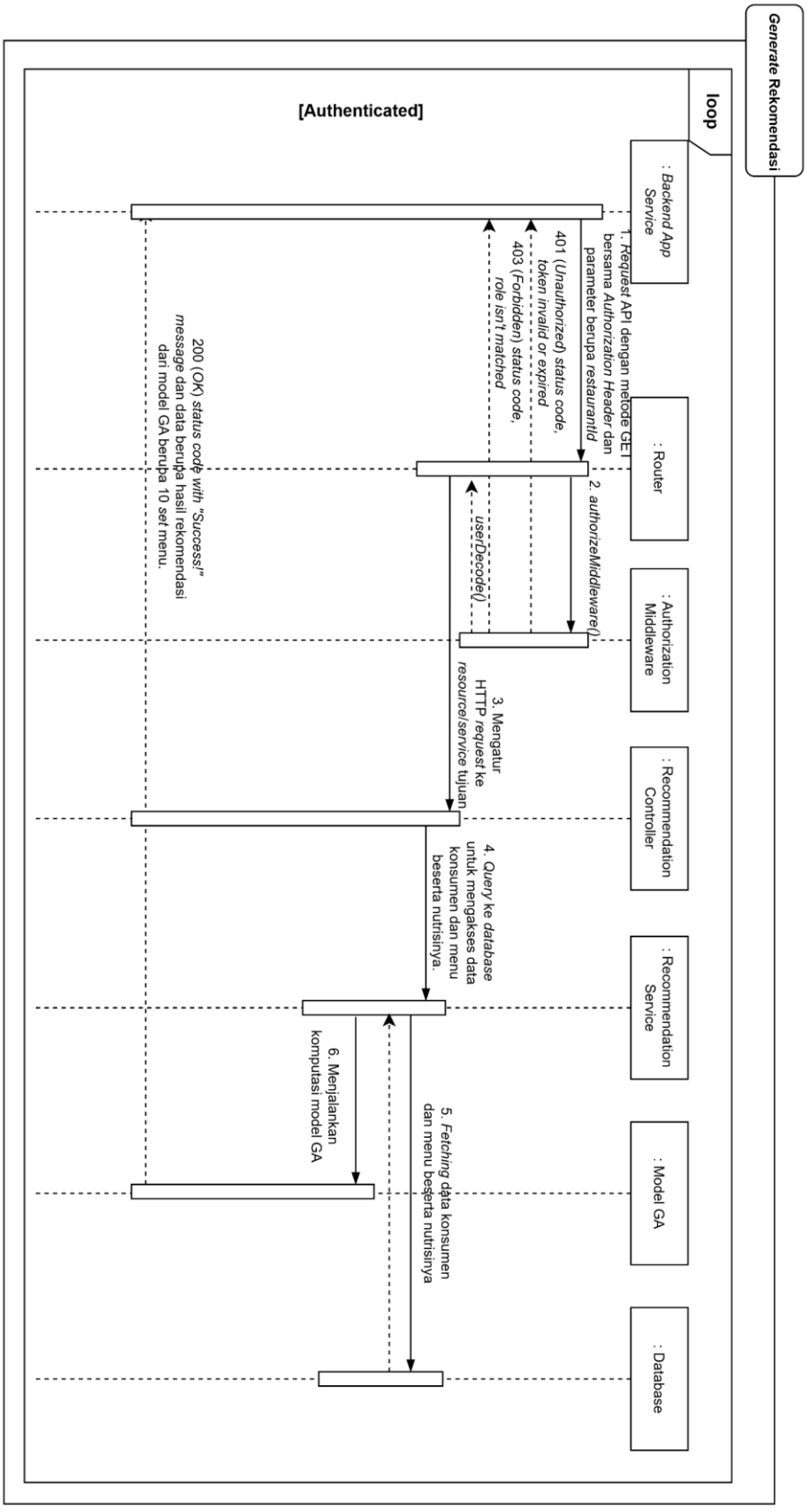
4.3.3 Pemodelan Perancangan Cepat (*Quick Modelling and Design*)

Response dari model sistem rekomendasi masih berupa *data frame* yang terlihat pada Gambar 25. Rekonstruksi *response* dari model GA tersebut selanjutnya akan dilakukan dengan mengubahnya menjadi JSON. Format JSON dari *response generate* rekomendasi dapat terlihat pada Gambar 27.

```
[
    {
        "rekomendasi": string,
        "makanan_pokok": int,
        "lauk_pauk": int,
        "sayuran": int,
        "minuman": int,
        "calory": int,
        "protein": float,
        "fat": float,
        "carbohydrate": float,
        "natrium": float,
        "cholesterol": float,
        "mufa": float,
        "pufa": float,
        "sfa": float,
        "fitness": float
    },

  ...
]
```

Gambar 27 Rancangan *response* rekomendasi dari model GA dengan format JSON

Sequence diagram dari proses *generate* rekomendasi dimulai dari pengiriman *request* ke *endpoint* API yang ada pada *backend model* dari *backend app service* yang dapat dilihat pada Gambar 28. Akan dilakukan verifikasi pada *access token* yang dimilikinya. Apabila berhasil maka *service* akan dijalankan untuk melakukan *query* dengan operasi *read* untuk mengambil data input yang akan dijalankan oleh model GA. Apabila *query* berhasil maka model GA akan dijalankan dan proses komputasi akan dimulai hingga rekomendasi *set* menu didapatkan. Rekomendasi *set* menu yang didapatkan akan dikirimkan kembali menjadi *response* berupa JSON dengan *status code* 200 jikalau berhasil.



Gambar 28 *Sequence diagram* proses *generate* rekomendasi pada *backend model*

4.3.4 Konstruksi Prototipe (*Construction of Prototype*)

Backend model dikembangkan dengan tujuan untuk menjalankan sistem rekomendasi pemilihan menu berupa model GA yang telah dikembangkan pada penelitian sebelumnya. Tahapan dimulai dengan mengubah GA yang awalnya masih berformat Ipython Notebook (*.ipynb) ke ekstensi Python (*.py). *Backend model* kemudian diinisialisasi dengan menggunakan bahasa Pemrograman Python dan *framework* Quart. Pada *backend* ini dikembangkan satu *Endpoint* API untuk menjalankan model pada Lampiran 7. Struktur folder dan *file* pada *backend model* yang dikembangkan dapat beserta detail keterangannya dapat dilihat pada Tabel 11 dan Tabel 12.

Tabel 11 Folder dan *file* utama yang dikembangkan pada *backend model*

Folder dan *File*	**Keterangan**
config	Digunakan untuk melakukan konfigurasi, misalnya pengaturan *database* dan JWT.
controller	Digunakan untuk melakukan *routing*, di mana setiap permintaan akan divalidasi JWT (via *middleware*) sebelum diteruskan ke service.
middleware	Digunakan untuk menangani *middleware* JWT, memastikan autentikasi sebelum permintaan diproses lebih lanjut.
service	Digunakan untuk melakukan *query* data (misalnya data konsumen dan menu), serta menjalankan model GA.
main.py	*File* utama yang berisi inisialisasi *server* dan menampung konfigurasi global aplikasi

Tabel 12 *File* yang berisikan logika utama dari folder *service*

File	**Keterangan**
recommendation_service.py	Berisikan logika untuk melakukan *query* ke *database* untuk mengambil data pengguna dan menu beserta kadungan nutrisinya
model_service.py	Berisikan model GA yang akan dijalankan untuk *generate* rekomendasi.

Alur *generate* rekomendasi menu dimulai dengan pengiriman *request* ke *endpoint* API /generate-recommendation pada *backend model* dari *backend app service* dengan metode POST. Penggunaan metode POST ialah dikarenakan menyimpan *payload* pada *request body*. *Request* yang dikirimkan berisikan *header* "*Authorization"* berupa JWToken yaitu *access token.* Selanjutnya, *request* akan masuk ke dalam *router* yang ada pada *file recommendation_controller.py* untuk menentukan *resource* yang akan diakses, dapat dilihat pada Gambar 29. Pada *router* juga dilakukan validasi terhadap autentikasi dari API yang dikirimkan. Hal ini dikarenakan untuk mengakses *endpoint* API diperlukan verifikasi terhadap JWT yang disematkan pada bagian *header request*. JWT akan di*decode* untuk mendapatkan *consumer_id* yang

tersimpan di dalam *payload* JWT. Selain itu, didapatkan juga *request body* yang dikirimkan berupa *restaurant_id* dari restoran yang dipilih oleh konsumen. Apabila tidak didapatkan *consumer_id* dan *restaurant_id* maka *response* berupa *error* dengan *status code* 400 (*Bad Request*) akan dikirimkan. Apabila validasi berhasil, selanjutnya *request* akan dikirimkan ke dalam *service*.

```
@recommendation_bp.route("/generate-recommendation", methods=["POST"])
@token_required
def get_menu():
    data = request.get_json()
    restaurant_id = data.get("restaurantId")
    consumer_id = request.consumer_id
    if not restaurant_id or not consumer_id:
        return jsonify({"error": "restaurantId and consumerId are
               required"}), 400
    recommendation = get_recommendation(restaurant_id, consumer_id)
    return (recommendation), 200
```

Gambar 29 Potongan kode program *router* dan *controller* pada *backend model*

Berdasarkan model GA yang dikembangkan, untuk menjalankan sistem rekomendasi dibutuhkan data konsumen. *File* berupa *recommendation_service.py* kemudian dibangun untuk melakukan *query* ke *database* untuk membaca data konsumen dan menu beserta kandungan nutrisinya, yang dapat dilihat pada Gambar 30. *Backend model* dan *app service* akan membaca *database* yang sama. Namun, pada *backend model* operasi yang dilakukan ke *database* hanya sebatas *read* saja. Setelah kedua data tersebut didapatkan maka *parsing data* ke dalam fungsi *set_data()* akan dilakukan. Selanjutnya, model GA akan dijalankan untuk menghasilkan rekomendasi dengan memanggil fungsi *generate_recommendations()*.

```
Def get_recommendation(restaurant_id, consumer_id):
    # Mendapatkan koneksi dari pool
    connection = get_db_connection()

    try:
        with connection.cursor(cursor_factory=RealDictCursor) as cursor:
            menu_query = """
                SELECT m.id AS menu_id, m.name AS menu_name, m.category,
                       n.weight_per_portion, n.calory, n.protein, n.fat,
                       n.carbohydrate, n.natrium, n.cholesterol, n.sfa,
                       n.mufa, n.pufa
                FROM "Menu" m
                LEFT JOIN "Nutrition" n ON m.id = n.menu_id
                WHERE m.restaurant_id = %s AND m.status = 'Approved'
            """
            cursor.execute(menu_query, (restaurant_id,))
            menu_data = cursor.fetchall()

            # Query data konsumen
            consumer_query = """
                SELECT c.consumer_id, pi.sex, pi.weight, pi.height, pi.age,
                       mh.no_history, mh.diabetes, mh.hypertension, mh.cardiovascular
                FROM "Consumer" c
                LEFT JOIN "PersonalInformation" pi ON c.consumer_id = pi.consumer_id
```

Gambar 30 Potongan kode program *query* untuk *read* data konsumen dan menu beserta kandungan nutrisinya

```
                LEFT JOIN "MedicalHistory" mh ON c.consumer_id = mh.consumer_id
                WHERE c.consumer_id = %s
            """
            cursor.execute(consumer_query, (consumer_id,))
            consumer_data = cursor.fetchone()

            ...

            # Generate rekomendasi
            set_data(df_menu, consumer_data)
            result = generate_recommendations()
            return result
}
```

Gambar 30 Potongan kode program *query* untuk *read* data konsumen dan menu beserta kandungan nutrisinya (*lanjutan*)

Hasil *generate* rekomendasi yang berhasil dijalankan akan mengirimkan *response* yang dapat dilihat pada Gambar 31. *Response* ini berupa *array* JSON yang berisikan 10 *set* menu yang direkomendasikan sesuai dengan variabilitas dari konsumen dan menu dari Restoran Karimata yang tersimpan di *database*.


```
[
    {
        "rekomendasi": "Paket Menu 1",
        "makanan_pokok": 1,
        "lauk_pauk": 17,
        "sayuran": 51,
        "minuman": 71,
        "calory": 1069,
        "protein": 20.4,
        "fat": 27.4,
        "carbohydrate": 182.2,
        "natrium": 698.58,
        "cholesterol": 205.2,
        "mufa": 3.5,
        "pufa": 1.4,
        "sfa": 21.3,
        "fitness": 0.78144600625
    },

  ...
]
```


Gambar 31 Potongan JSON *response* hasil rekomendasi dari *backend model*

4.3.5 Penyebaran dan Umpan Balik (*Deployment Delivery and Feedback*)

Blackbox testing selanjutnya dilakukan untuk menguji fungsionalitas dan integrasi dari *endpoint* pada *backend model*. Pengujian ini dilakukan melalui skenario uji positif dan negatif. Hasilnya, didapatkan bahwa pengujian terhadap satu *endpoint* yang ada pada *backend model* berhasil dilakukan yang dapat dilihat pada Lampiran 11.

Deployment pada *backend model* selanjutnya dilakukan dengan alur yang sama seperti Gambar 22 yaitu melalui kontenerisasi dengan Docker pada layanan *serverless* Cloud Run. *Feedback* yang didapatkan pada iterasi ini ialah *response* yang dihasilkan dari pemodelan GA harus diolah kembali sehingga informasi yang ditampilkan pada konsumen nantinya dapat lebih informatif.

4.4 Iterasi 3

4.4.1 Komunikasi (*Communication*)

Iterasi ketiga merupakan tahapan yang dilakukan fitur rekomendasi dan pemesanan pada *backend app service*. Pengembangan fitur rekomendasi pada *backend app service* dilakukan setelah *backend model* berhasil dibangun agar

integrasi dapat dilakukan. Model GA yang telah berhasil diintegrasikan akan menghasilkan *response* seperti Gambar 31 apabila *generate* rekomendasi berhasil berjalan. Berdasarkan hasil *response* yang dihasilkan dari *backend model*, didapatkan *fitness value, id* menu setiap kategori, agregat dari kandungan nutrisi setiap *set menu*. *Response* ini kemudian akan diolah kembali pada *backend app service* seperti *generate* informasi dari setiap menu yang direkomendasikan, menghitung total harga dari setiap *set* menu, hingga disimpan di dalam *database*.

Diskusi juga dilakukan untuk menentukan bagaimana pengembangan fitur pemesanan diterapkan. Berdasarkan hasil diskusi, pemesanan yang dilakukan oleh konsumen dibatasi terhadap *set* menu yang direkomendasikan saja. Untuk itu, konsumen harus melalukan *generate* rekomendasi terlebih dahulu untuk melakukan pemesanan. Selain itu, terdapat batasan yang diterapkan bahwa fitur pemesanan (*order*) tidak terhubung dengan restoran dan belum diterapkannya *payment gateway* dalam pengembangan ini.

4.4.2 Perencanaan Cepat (*Quick Planning*)

Pengembangan fitur rekomendasi dan pemesanan dilakukan pada iterasi ketiga. Perencanaan kemudian dilakukan untuk merancang bagaimana fitur rekomendasi dan pemesanan ini dapat dikembangkan. Berdasarkan hasil komunikasi dan *feedback* di iterasi kedua, fitur rekomendasi dikembangkan untuk melakukan *trigger* komputasi model GA yang ada pada *backend model*. Selain itu, fitur ini juga bertujuan untuk mengelola kembali hasil rekomendasi dari model GA sehingga pengaksesan dapat dilakukan. Fitur ini hanya dikembangkan pada pengguna yang memiliki peran sebagai konsumen agar dapat melakukan *generate* rekomendasi dan mengakses hasilnya.

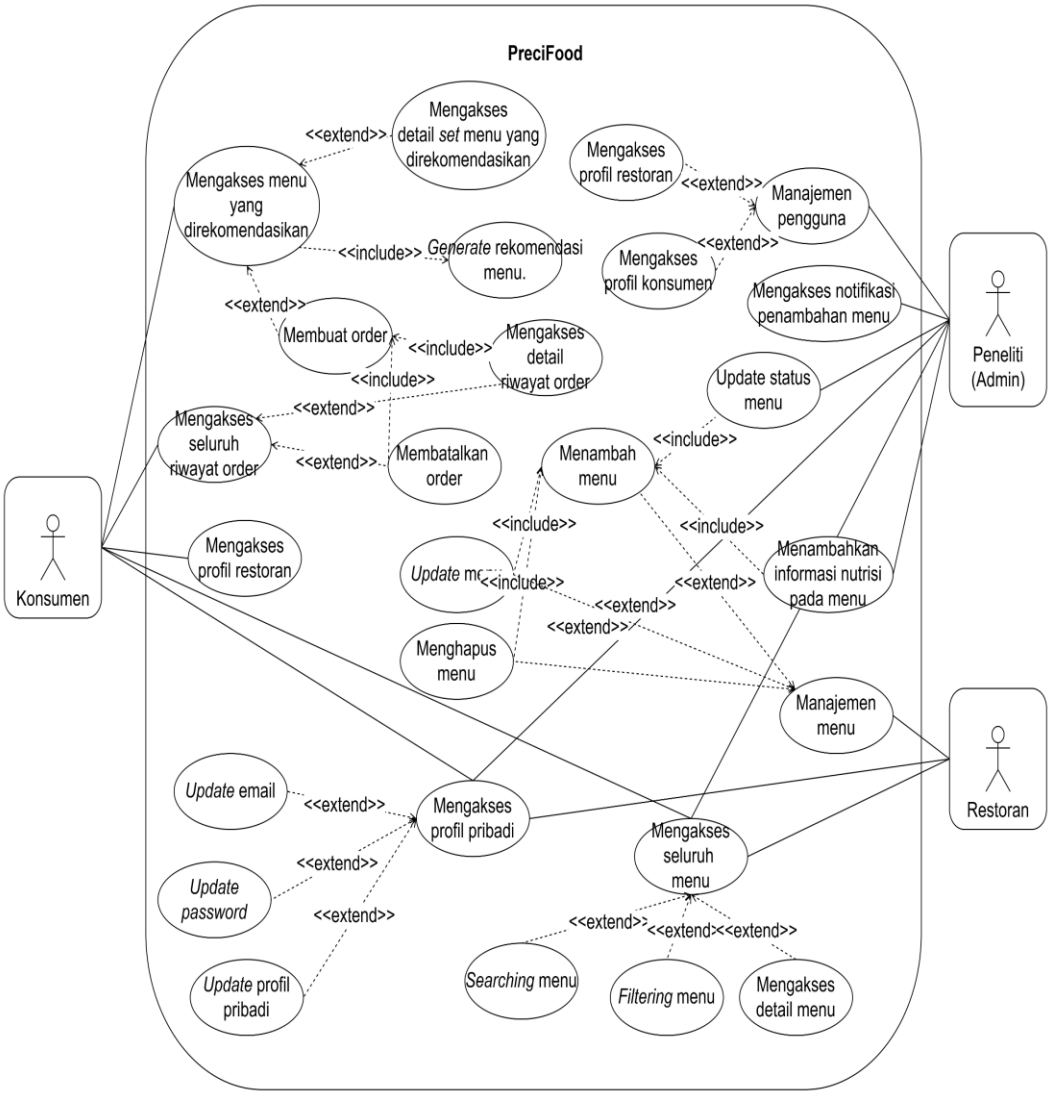
Modul pemesanan selanjutnya dapat diakses oleh konsumen dengan melakukan *order* terhadap salah satu *set* menu yang direkomendasikan setelah modul/fitur rekomendasi berhasil dikembangkan. Berdasarkan hasil diskusi, *user requirement* dari kedua modul atau fitur ini didefinisikan pada Tabel 13.

Tabel 13 *User requirement* iterasi ketiga

Aktor	***Task***	***User Story***
Konsumen	*Generate* rekomendasi *set* menu di suatu restoran.	Sebagai konsumen saya ingin mendapatkan rekomendasi dari menu yang tersedia di restoran berdasarkan variabilitas yang saya miliki.
	Cek status *generate* rekomendasi.	Sebagai konsumen saya ingin mengetahui status proses dari *generate* rekomendasi yang telah saya lakukan.
	Mengakses daftar (*list*) rekomendasi menu yang telah didapatkan.	Sebagai konsumen saya ingin dapat mengakses rekomendasi *set* menu yang telah direkomendasikan.
	Mengakses detail rekomendasi menu yang telah didapatkan.	Sebagai konsumen saya ingin dapat mengakses detail rekomendasi *set* menu yang telah direkomendasikan.

Tabel 12 *User requirement* iterasi ketiga (*lanjutan*)

Melakukan pemesanan menu.	Sebagai konsumen saya ingin melakukan pemesanan pada salah satu *set* menu yang direkomendasikan.
Mengakses riwayat pemesanan yang telah dilakukan.	Sebagai konsumen saya ingin mengakses riwayat pemesanan.
Mengakses riwayat detail pemesanan yang telah dilakukan.	Sebagai konsumen saya ingin mengakses detail riwayat pemesanan.
Mengonfirmasi pemesanan.	Sebagai konsumen saya ingin mengonfirmasi pemesanan yang telah saya lakukan.
Membatalkan pemesanan yang telah dilakukan.	Sebagai konsumen saya ingin membatalkan pemesanan.



Gambar 32 *Use case diagram* PreciFood iterasi ketiga

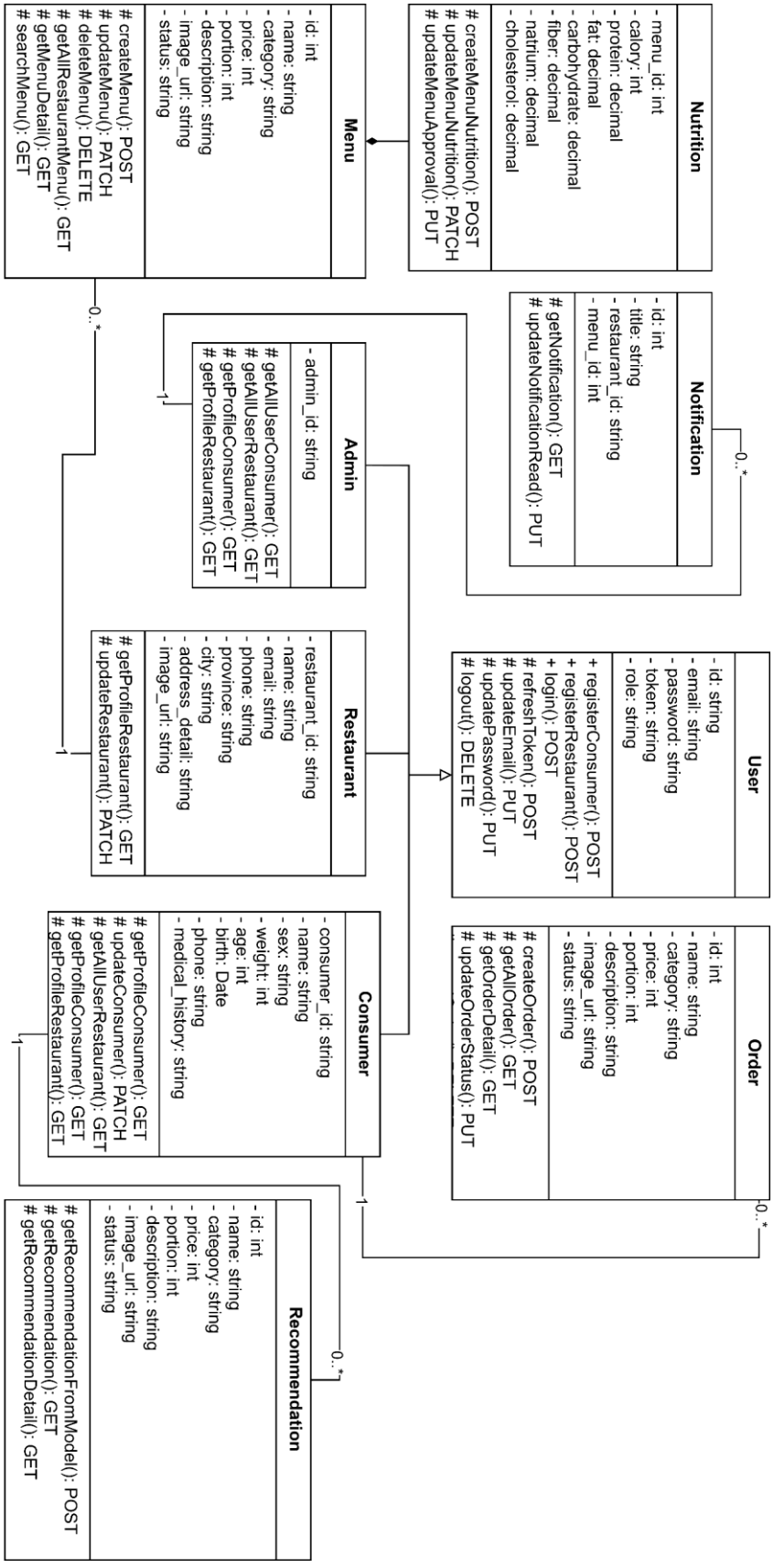
Use case diagram pada iterasi ketiga selanjutnya didefinisikan ulang dengan mengembangkan fitur rekomendasi dan pemesanan yang dapat dilihat pada Gambar 32. Konsumen memiliki *use case* seperti mengakses menu yang direkomendasikan pada suatu restoran, berdasarkan kondisi yang dimilikinya. Terdapat relasi *include* saat mengakses *set* menu yang direkomendasikan dengan proses *generate* rekomendasi, di mana konsumen harus terlebih dahulu melakukan *generate* rekomendasi sebelum dapat mengakses menu yang direkomendasikan. Selain itu, terdapat relasi *extend* antara mengakses menu yang direkomendasikan dengan membuat pesanan (*order*). Relasi *extend* ini menunjukkan bahwa pemesanan hanya dapat dilakukan setelah pengguna melihat menu yang direkomendasikan. Artinya, konsumen tidak bisa langsung membuat pesanan tanpa terlebih dahulu mengakses menu yang direkomendasikan. Selain itu, masih terdapat *use case* yang dapat dilakukan oleh konsumen, seperti mengakses riwayat pemesana (*order*), mengakses profil pribadi, mengakses profil restoran, hingga mengakses seluruh menu yang terdapat pada restoran.

4.4.3 Pemodelan Perancangan Cepat (*Quick Modelling Design*)

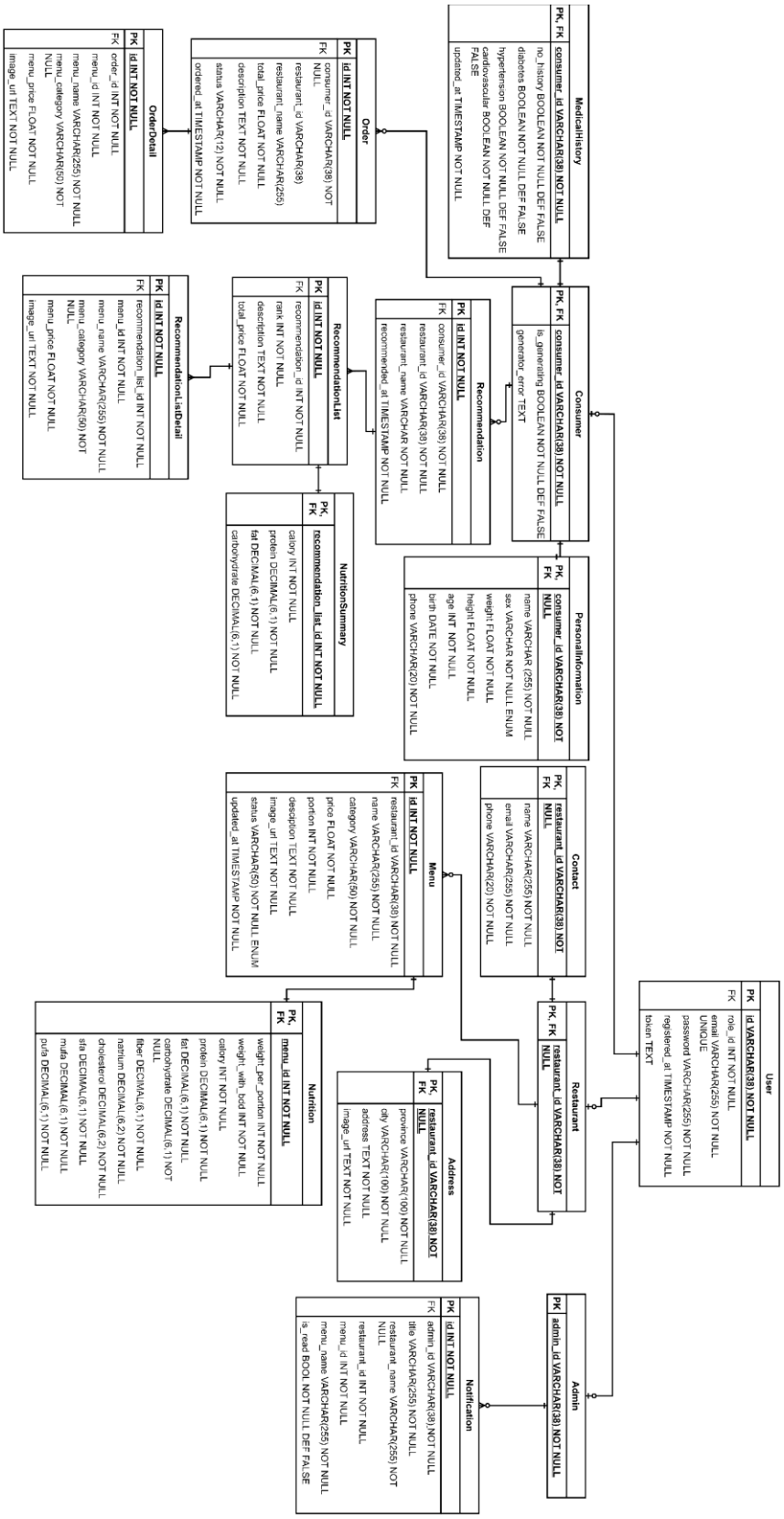
Terdapat penambahan *class* berupa modul/fitur yang dilakukan pada iterasi ketiga, yaitu fitur rekomendasi dan pemesanan. Pada Gambar 33, "Consumer" memiliki relasi *one to zero or many* dengan "Recommendation". Artinya, satu konsumen dapat memiliki nol (bagi konsumen baru) atau lebih rekomendasi menu. Setiap rekomendasi hanya dimiliki oleh satu konsumen saja. Hal ini juga menunjukkan bahwa setiap rekomendasi dapat berbeda-beda berdasarkan kondisi yang dimiliki oleh konsumen.

Penambahan tabel sebanyak enam tabel pada Gambar 34, di antaranya Order, OrderDetail, Recommendation, RecommendationListDetail, NutritionSummary, RecommendationListDetail. Hasil dari *generate* rekomendasi menu yang didapatkan oleh konsumen akan disimpan pada tabel Recommendation, RecommendationList, NutritionSummary, dan RecommendationListDetail. Selain itu, terdapat penambahan atribut berupa *is_generating* dan *generator_error* pada tabel Consumer. Kedua atribut ini ditujukan untuk melakukan *tracking* dari proses *generate* rekomendasi yang dilakukan oleh konsumen.

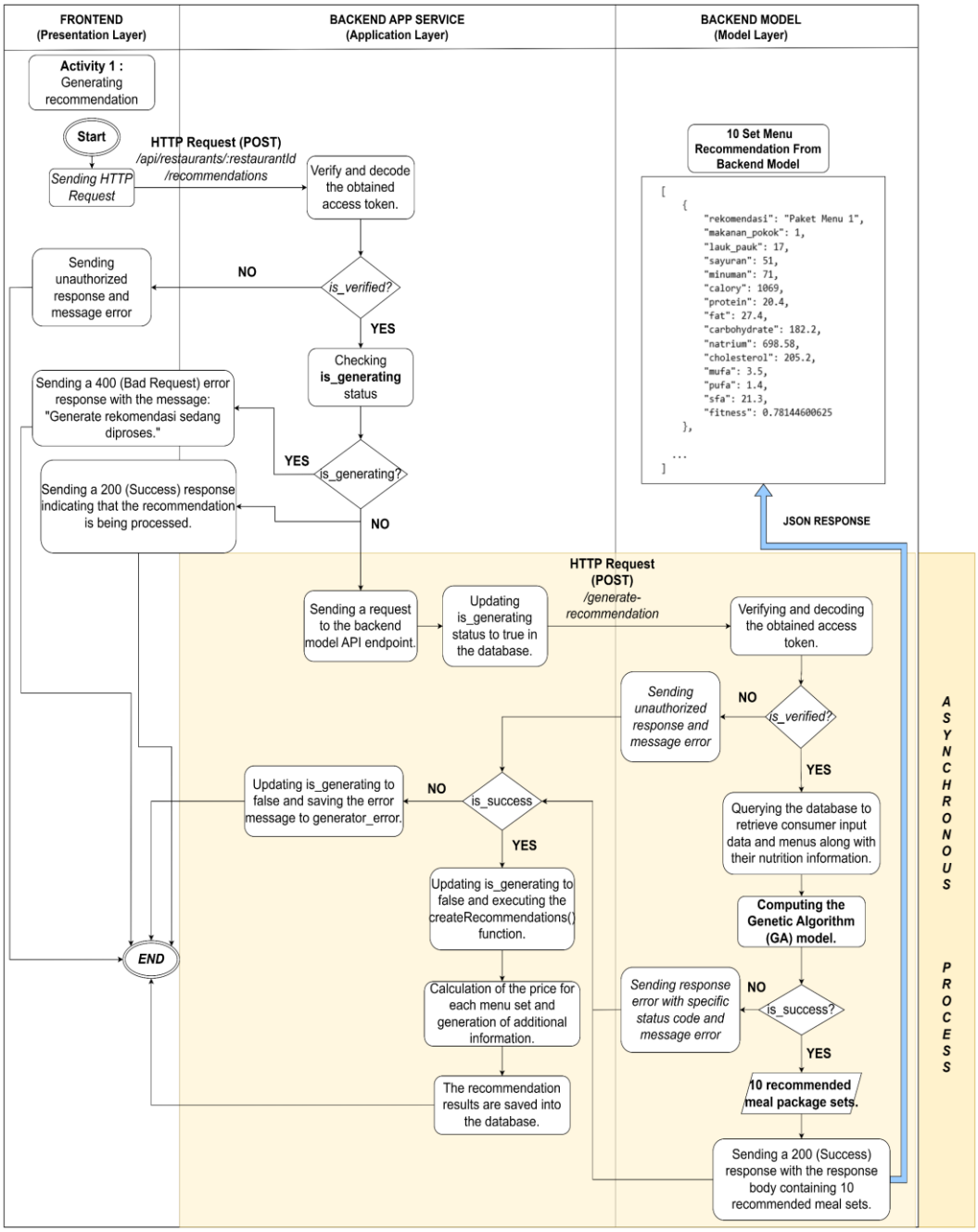
Untuk menggambarkan bagaimana proses *generate* rekomendasi dilakukan, *activity diagram* didesain pada Gambar 35. Proses *generate* rekomendasi dimulai melalui pengiriman *request* ke *endpoint* API yang ada di *backend app service* melalui *frontend* konsumen. Setiap *request* yang diberikan akan menyimpan *access token* pada *header* untuk nantinya diverifikas. Setelah verifikasi berhasil, pengecekan status proses *generate* rekomendasi akan dilakukan. Apabila proses *generate* rekomendasi masih dilakukan maka *response error* akan diberikan. Jika tidak maka *response success* dari *generate* rekomendasi akan diberikan yang menandakan bahwa *generate* rekomendasi mulai diproses secara *asynchronous.* Proses *generate* rekomendasi selanjutnya dilakukan secara *asynchronous* pada layanan *backend app service* dan *backend model*. Tujuannya ialah untuk menghindari terjadinya *timeout* pada *frontend* jika harus menunggu hasil rekomendasi dari kedua *backend*.



Gambar 33 *Class diagram* iterasi ketiga

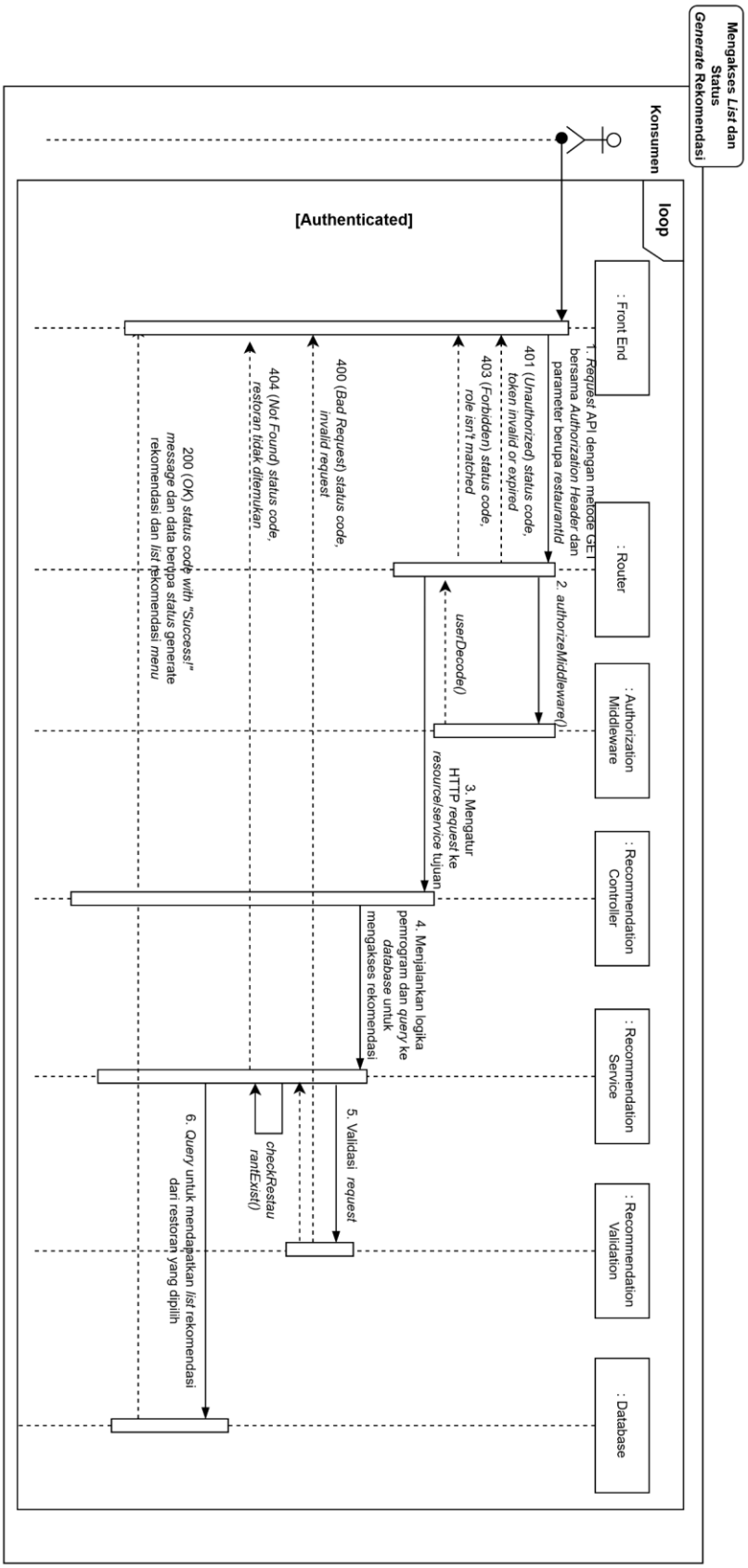


Gambar 34 *Database diagram* iterasi ketiga



Gambar 35 *Activity diagram* proses *generate* rekomendasi

Pada Gambar 36, dikembangkan *sequence diagram* untuk merepresentasikan alur mengakses hasil rekomendasi yang didapatkan dari *backend model* yang telah dikelola oleh *backend app service* dan disimpan pada *database*. *Response* tersebut berupa 10 *set* rekomendasi menu beserta status dari proses *generate* rekomendasi.



Gambar 35 *Sequence diagram* mengakses *list* rekomendasi

4.4.4 Konstruksi Prototipe (*Construction of Prototype*)

Pengembangan prototipe selanjutnya dilakukan pada fitur rekomendasi dan pemesanan di *backend app service*. Fitur rekomendasi merupakan fitur yang dapat digunakan oleh konsumen untuk melakukan *generate* rekomendasi, mengakses hasil rekomendasi, mengakses status proses dari *generate* rekomendasi, dan mengakses detail dari menu yang direkomendasikan. Pada Lampiran 8 dihasilkan *endpoint* API dari fitur rekomendasi.

Modul rekomendasi dikembangkan dengan tujuan agar konsumen dapat mengakses berbagai hal kebutuhan yang terdapat di dalamnya, seperti melakukan *generate* rekomendasi, mengecek status proses *generate*, hingga mengakses hasilnya. *Endpoint* API yang dikembangkan pada fitur ini dapat dilihat pada Lampiran 8.

Modul pemesanan dikembangkan dengan tujuan untuk melakukan manajemen terhadap pemesanan yang telah dilakukan oleh konsumen, seperti menambah pesanan dari *set* menu yang direkomendasikan. *Endpoint* API yang berasil dikembangkan dalam modul ini dapat dilihat pada Lampiran 9.

Pengembangan fitur rekomendasi pada *backend app service* dilakukan setelah pengembangan *backend model* berhasil dilakukan. Berdasarkan hasil pengembangan, rekomendasi dimulai dengan mengirimkan *request* ke *endpoint* API /api/restaurants/:restaurantId/recommendations dengan metode POST. Penggunaan metode POST didasarkan pada bahwa hasil *generate* nantinya akan disimpan ke *database* dan tidak ditampilkan secara langsung. *Request* tersebut di antarnya berisikan *header* yang menyimpan *key* 'Authorization' berupa *access token* dan parameter berupa id dari restoran yang dipilih. Selanjutnya, *request* akan masuk ke *router* untuk dilakukan validasi terkait JSON Web Token, yaitu *access token* yang menyimpan akses otorisasinya. Apabila validasi ini berhasil selanjutnya *request* tersebut akan diarahkan menuju recommendation-controller.ts. Untuk melihat bagaimana proses fitur rekomendasi berjalan dapat dilihat pada *activity diagram* Lampiran 2.

Sebelum *request* mengakses *service* untuk mengirimkan *request* ke *endpoint* API pada *backend model*, dilakukan terlebih dahulu pengecekan pada *controller* apakah pengguna dengan id yang dikirimkan sedang melakukan proses *generate* rekomendasi atau tidak. Jika *is_generating* bernilai *true, response error* dengan *status code* 400 dan pesan "Rekomendasi sedang di*generate*". Apabila bernilai *false*, *response status* 200 dengan pesan "Generate rekomendasi berhasil! Silahkan untuk merefresh halaman rekomendasi menu setelah 5 menit" akan langsung dikirimkan. Proses *generate* rekomendasi dilakukan secara *asynchronous* agar proses dilakukan di *background* yang bertujuan untuk menghindari terjadi *timeout* pada *frontend*. Selanjutnya, *request* akan diteruskan ke *backend model* melalui *service*. Apabila terjadi kegagalan dalam melakukan *generate* rekomendasi, *error* akan disimpan di dalam *database* sehingga konsumen mengetahui bahwa proses *generate* rekomendasi yang dilakukannya mengalami kegagalan. Implementasi *controller* dan *service* dapat dilihat pada Gambar 37 dan Gambar 38.

```
export class RecommendationController {
    static async getRecommendationFromModel(req: UserRequest, res: Response,
      next: NextFunction) {
        const isGenerating = await RecommendationService
                            .checkGenerateStatus(req.user.id);
        if (isGenerating?.is_generating === true) {
            return next(new ResponseError(400, "Generate rekomendasi sedang
             dalam proses"));
        }

        try {
            const request: GenerateRecommendationRequest = {
                token: req.headers["authorization"] as string,
                consumer_id: String(req.user.id),
                restaurant_id: req.params.restaurantId
            };

            res.status(200).json({
                message: "Generate rekomendasi berhasil! Silahkan untuk
                          merefresh halaman
                          rekomendasi      menu      setelah      5      menit",
            });
            setImmediate(async () => {
                try {
                    await
RecommendationService.getRecommendationFromModel(request);
                } catch (error) {
                    console.error("Error in
RecommendationService.getRecommendation:", error);
                }
            });
        } catch (e) {
            await prismaClient.consumer.update({
                where: { consumer_id: String(req.user.id) },
                data: {
                    generator_error: String(e),
                    is_generating: false,
                }
            });
        }
    }
```

Gambar 37 Potongan kode *controller* getRecommendationFrom

```
static async getRecommendationFromModel(request: GenerateRecommendationRequest) {
    const recommendationRequest =
Validation.validate(RecommendationValidation.GENERATERECOMMENDATION, request);

    try {
        ...
        const response = await fetch(`${process.env.MODEL_URL}/generate-
recommendation`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': recommendationRequest.token,
            },
            body: JSON.stringify(payload),
        });

        if (response.status !== 200) {
```

Gambar 38 Potongan kode program *service* untuk menjalankan sistem rekomendasi dari *backend app service*

```
            const error = new Error('Terjadi kegagalan pada generate model. Silahkan
            generate ulang!');
            error.name = 'ModelGenerationError';

            await prismaClient.consumer.update({
                where: { consumer_id: recommendationRequest.consumer_id },
                data: {
                    generator_error: String(error),
                    is_generating: false,
                }
            });
            return;
        }

        const data = await response.json();
        await this.createRecommendations(data, recommendationRequest.restaurant_id,
recommendationRequest.consumer_id);

        await prismaClient.consumer.update({
            where: { consumer_id: recommendationRequest.consumer_id },
            data: { is_generating: false, generator_error: null }
        });

    } catch (error) {
        ...
    }
}
```

Gambar 38 Potongan kode program *service* untuk menjalankan sistem rekomendasi dari *backend app service* (*lanjutan*)

Apabila rekomendasi *set* menu berhasil didapatkan dari *backend model*, hasil rekomendasi pada Gambar 31 tersebut akan diolah kembali pada *backend app service* untuk disimpan di *database*. Pengolahan yang dilakukan ialah seperti menghitung agregat dari total harga setiap menu yang direkomendasikan hingga membuat deskripsinya. Setelah hasil rekomendasi berhasil diolah dan disimpan, atribut *is_generating* akan diperbaharui nilainya menjadi *false* dan *generator_error* menjadi *null*. Konsumen dapat mengirimkan *request* dengan metode GET ke *endpoint* API untuk melihat status *generate* rekomendasi yang dilakukannya dan mengakses hasil *set* menu yang direkomendasikan apabila berhasil. *Response* berupa JSON dapat dilihat pada Gambar 39 yang merupakan 10 *set* menu rekomendasi pada Restoran Karimata kepada konsumen beserta status dari proses *generate* rekomendasi, dengan detail *set* pada Gambar 40.

```
{
    "message": "Success!",
    "data": {
        "restaurant_id": "R-0192c857-35e6-7cc2-98da-46f0faf6f651",
        "restaurant_name": "Restoran Karimata",
        "recommended_at": 1733842345025,
        "status": {
            "is_generating": false,
            "generator_error": null
        },
        “recommendations”: [
            {
                “id”: 161,
                “rank”: 1,
                “description”: “Nasi, Gurame Goreng Rica-Rica, Brokoli Bawang Putih, Jus Strawberry”,
                “total_price”: 193000,
                “image_url”: [
                    {
                        “url”: “https://storage.googleapis.com/precifood-image/menu-images/....png”
                    },
                    {
                        “url”: “https://storage.googleapis.com/precifood-image/menu-images/...png”
```

Gambar 39 Potongan JSON *response* status dan *list* hasil *generate* rekomendasi

```
                },
                {
                    "url": "https://storage.googleapis.com/precifood-image/menu-...png"
                },
                {
                    "url": "https://storage.googleapis.com/precifood-image/menu-images/....png"
                }
            ]
        },
       ...
      ]
   }
}
```


Gambar 39 Potongan JSON *response* status dan *list* hasil *generate* rekomendasi (*lanjutan*)


```
{
    "message": "Success!",
    "data": {
        "total_price": 193000,
        "nutrition_summary": {
            "calory": 2061,
            "protein": 108.5,
            "fat": 198.4,
            "carbohydrate": 198.4
        },
        "recommendations": [
            {
                "menu_id": 1,
                "name": "Nasi",
                "category": "Makanan Pokok",
                "portion": 1,
                "price": 8000,
                "description": "Nasi putih yang disajikan hangat, cocok sebagai
                "image_url": "https://storage.googleapis.com/precifood-
            },
            {
                "menu_id": 9,
                "name": "Gurame Goreng Rica-Rica",
                "category": "Lauk Pauk",
                "portion": 3,
                "price": 135000,
                "description": "Ikan gurame goreng yang disajikan dengan sambal
                "image_url": "https://storage.googleapis.com/precifood-
            },
            {
                "menu_id": 44,
                "name": "Brokoli Bawang Putih",
                "category": "Sayuran",
                "portion": 3,
                "price": 25000,
                "description": "Brokoli yang ditumis dengan bawang putih,
                                memberikan rasa segar dan gurih.",
                "image_url": "https://storage.googleapis.com/precifood-
                              image/menu-images/....png"
            },
            {
                "menu_id": 70,
                "name": "Jus Strawberry",
                "category": "Minuman",
```


Gambar 40 Detail rekomendasi salah satu *set* menu

```
                "portion": 1,
                "price": 25000,
                "description": "Jus strawberry dengan rasa manis asam yang
segar, cocok sebagai minuman penutup.",
                "image_url":        "https://storage.googleapis.com/precifood-
image/menu-images/....png"
            }
        ]
    }
}
```


Gambar 40 Detail rekomendasi salah satu *set* menu (*lanjutan*)

Konsumen yang telah berhasil mendapatkan hasil *generate* rekomendasi menu, akan mendapatkan 10 *set* menu yang direkomendasikan. Setiap *set* menu tersebut berisikan id rekomendasi agar konsumen dapat mengakses *detail* dari *set* menu yang direkomendasikan, di antaranya ada total harga, *nutrition summary*, dan menu-menu yang direkomendasikan terbagi atas empat kategori, makanan pokok, lauk pauk, sayuran, dan minuman.

Setelah konsumen berhasil mendapatkan *list* berupa sepuluh set menu yang direkomendasikan, selanjutnya pemesanan dapat dilakukan dengan mengirimkan request dengan metode POST ke *endpoint* API /api/consumers/orders/:recommendationId. Parameter berupa recommendationId menunjukkan bahwa untuk melakukan pemesanan, konsumen hanya dibatasi dengan hanya memesan *set* menu yang direkomendasikan dari model GA berdasarkan variabilitas yang dimilikinya. Konsumen dapat melihat riwayat pemesanan yang telah dilakukannya dengan mengakses ke *endpoint* API /api/consumers/orders melalui metode GET. Hasil JSON dari riwayat pemesanan dapat dilihat pada Gambar 41. Selanjutnya, konsumen dapat melihat detail, mengonfirmasi pemesanan, hingga membatalkannya dengan mengirimkan *request* ke *endpoint* API yang sesuai.


```
{
  "message": "Success!",
  "data": [
    {
      "id": 5,
      "restaurant_name": "Restoran Karimata",
      "total_price": 148000,
      "ordered_at": 1730861038164,
      "status": "Unconfirmed",
      "description": "Nasi, Cumi Bakar Sambal Ijo, Terong Penyet, Es Kelapa
Jeruk Peras"
    },
    {
      "id": 6,
      "restaurant_name": "Restoran Karimata",
      "total_price": 83000,
      "ordered_at": 1730861038164,
      "status": "Unconfirmed",
      "description": "Nasi, Sup Ayam Asam Kemangi, Terong Sambal Hijau, Jus
Munser"
    },
```

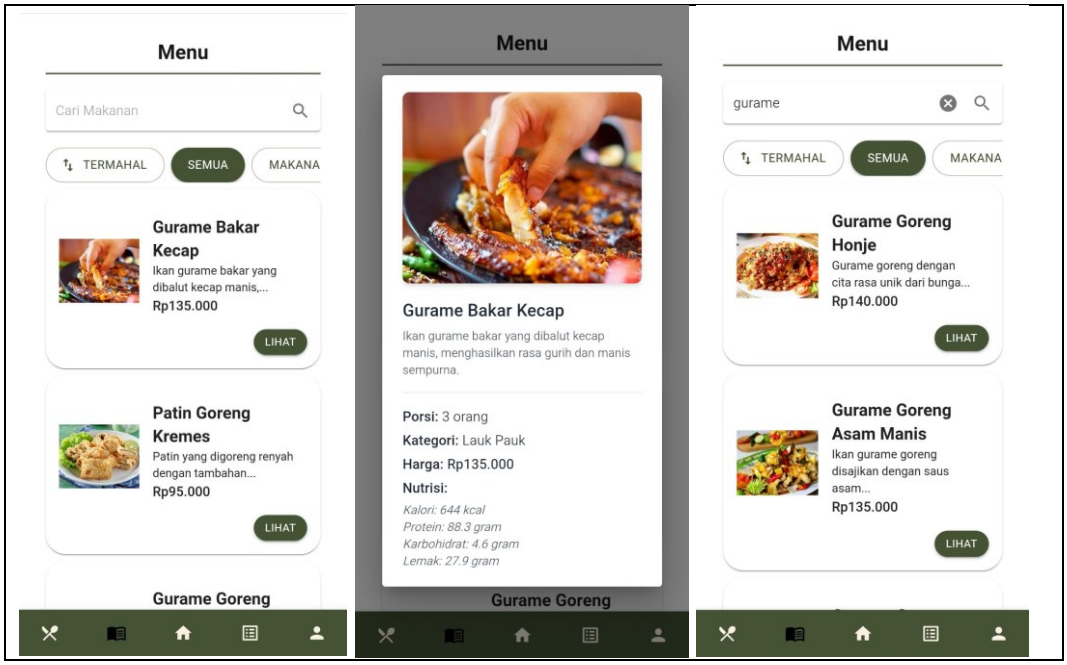

Gambar 41 Potongan JSON riwayat pemesanan (*order*)

```
  {
      "id": 4,
      "restaurant_name": "Restoran Karimata",
      "total_price": 113000,
      "ordered_at": 1730861038164,
      "status": "Confirmed",
      "description": "Nasi, Udang Goreng Tepung, Terong Penyet, Jus Pepaya"
    }
  ]
}
```

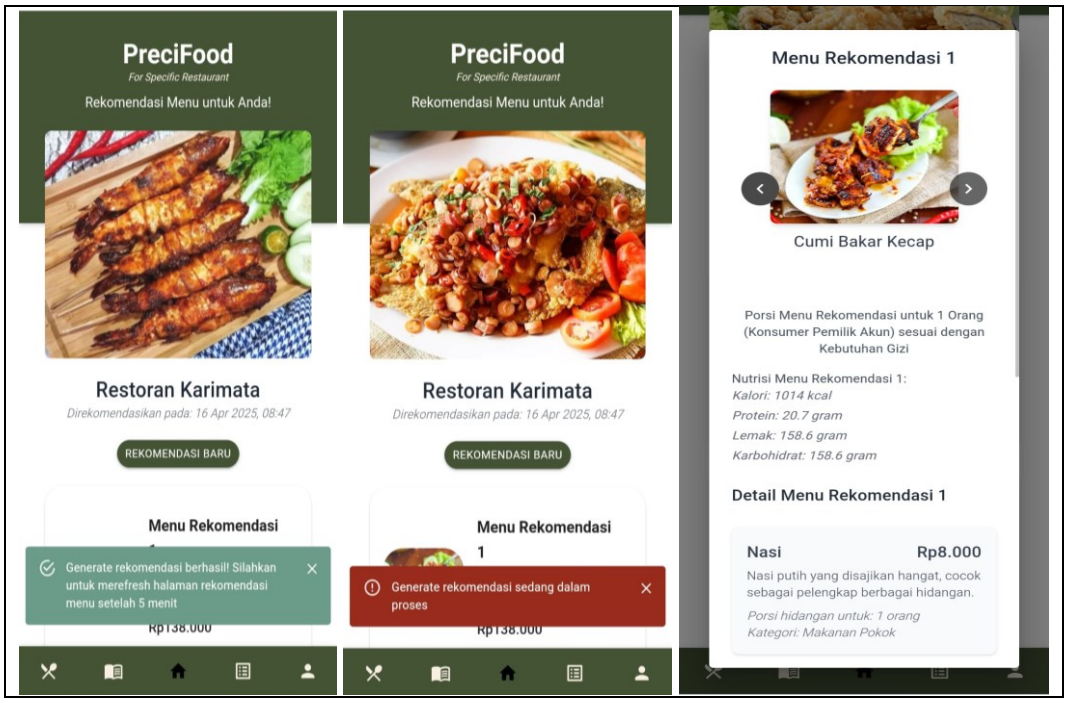

Gambar 41 Potongan JSON riwayat pemesanan (*order*) (*lanjutan*)

4.4.5 Penyebaran dan Umpan Balik (*Deployment Delivery and Feedback*)

Blackbox testing kemudian dilakukan pada iterasi ketiga untuk menguji seluruh *endpoint* API yang telah dibuat pada *backend app service* dan *backend model.* Sebanyak 39 *endpoint* API diuji dengan skenario *test case* positif dan negatif untuk menguji fungsionalitas dan integrasi dari masing-masing *endpoint* API. Hasilnya, 39 *endpoint* tersebut berhasil diuji dengan menghasilkan *response* atau *output* yang sesuai dari input atau *request* yang diterima, baik untuk *test case* positif dan negatif. Hal ini menunjukkan bahwa fungsionalitas dan integrasi dari masing-masing *endpoint* telah berjalan dengan lancar dan integrasi. *Test case* positif yang diujikan ialah dengan mengirimkan *request* dengan data yang valid, mulai dari tipe data dan atribut yang dikirimkan, *access token* yang valid, dan *role* yang sesuai. Pada pengujian dengan *test case* negatif ialah kebalikan dari *test case* positif. Hasil pengujian iterasi ketiga dapat dilihat pada Lampiran 12 , skenario *test case* negatif dapat diakses pada Lampiran 13.



Gambar 42 Hasil integrasi fitur menu berupa akses menu oleh konsumen pada *frontend* yang dikembangkan oleh Ismy Fana Fillah



Gambar 43 Hasil integrasi fitur rekomendasi berupa *generate* dan akses rekomendasi oleh konsumen pada *frontend* yang dikembangkan oleh Ismy Fana Fillah

Deployment pada *backend model* dilakukan terlebih dahulu seperti sebelumnya melalui kontenerisasi dengan Docker. *Base url* hasil *deployment backend model* ini selanjutnya akan digunakan pada *backend app service* untuk melakukan *fetch* ke *endpoint* API untuk *generate* rekomendasi pada *backend model*. Selanjutnya, dengan cara yang sama *deployment* dari *backend app service* juga dilakukan pada Cloud Run. Hasil *base url* dari *deployment backend app service* ini yang nantinya diintegrasikan terhadap *frontend* yang dikembangkan oleh Ismy Fana Fillah. Contoh hasil integrasi dengan frontend seperti fitur menu dan rekomendasi dapat dilihat pada Gambar 42 dan Gambar 43. Setelah berhasil terintegrasi, *frontend* yang dikembangkan juga dilakukan *deployment* pada Cloud Run. *Custom domain mapping* selanjutnya dilakukan pada *base url* yang dihasilkan dari *deployment frontend* sehingga PreciFood kini memiliki *domain url*nya sendiri. *Custom domain mapping* sendiri merupakan salah satu fitur yang disediakan pada layanan Cloud Run sehingga *domain mapping* dapat dilakukan dengan mudah. Hasilnya, integrasi antara *frontend*, *backend app service*, *backend model*, hingga *database* berhasil dilakukan dan aplikasi PreciFood telah diluncurkan (*go live*) kepada pengunjung (konsumen) Restoran Karimata pada tanggal 14 Desember 2024.

SIMPULAN DAN SARAN

5.1 Simpulan

Penelitian ini berhasil mengintegrasikan *frontend* yang dikembangkan oleh Ismy Fana Fillah dan model GA yang Seminar *et al.* (2024) melalui pengembangan dua *backend*, yaitu *backend app service* dan *backend model* yang menghasilkan API dengan arsitektur REST. *Backend app service* digunakan untuk menunjang logika bisnis/utama (inti) dari aplikasi dan *backend model* untuk menjalankan sistem rekomendasi pemilihan menu berbasis *Genetic Algorithm* (GA) yang telah dikembangkan pada penelitian sebelumnya. Pengembangan dilakukan dengan metode *Prototyping* melalui tiga kali iterasi. Arsitektur pengembangan *software multi-tier architecture* diterapkan dalam pengembangannya dengan membagi sistem ke dalam lima lapisan dengan fungsinya masing-masing, di antaranya *presentation layer, application layer, model layer, data layer,* dan *storage layer.* 38 API pada *backend app service* dan 1 API pada *backend model* berhasil dikembangkan untuk menunjang kebutuhan dari aplikasi. Pengujian dengan *blackbox testing* dilakukan untuk menguji fungsionalitas dan integrasi dari *endpoint* API yang telah dibangun dengan hasil semua *endpoint* bekerja sesuai dengan *test case* yang diberikan. PreciFood telah diluncurkan kepada pengunjung (konsumen) Restoran Karimata pada 14 Desember 2024.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, terdapat pengembangan yang masih dapat dilakukan baik dari segi fungsionalitas, non-fungsionalitas, hingga kebutuhan lainnya dari aplikasi.

a. Dari sisi fungsionalitas, analisis alur/proses bisnis dari PreciFood perlu dilakukan secara lebih detail dan menyeluruh pada pengembangan selanjutnya. Hal ini dikarenakan aplikasi PreciFood merupakan kali pertama dikembangkan di dalam penelitian ini sehingga masih terdapat kemungkinan adanya penambahan logika bisnis dari aplikasi. Misalkan, terkait proses analisis kandungan nutrisi dari menu yang disajikan restoran mitra dan admin masih dilakukan di luar aplikasi. Tentunya hal ini dapat diakomodir di dalam pengembangan ke depannya.
b. Dari sisi non-fungsionalitas, terdapat beberapa hal yang perlu ditingkatkan, yaitu terkait performa, skalabilitas, dan keamanan. Salah satunya dengan optimalisasi *query database* yang perlu dilakukan pada pengembangan selanjutnya untuk meningkatkan performa dari aplikasi. Dari aspek keamanan (*security*), penerapan API Key dapat dilakukan selain menggunakan JSON Web Token (JWT) untuk memberikan perlindungan ekstra terhadap API yang dikembangkan.
c. Penerapan *pipeline* CI/CD (*Continuous Integration* / *Continuous Deployment*) pada penelitian ini belum dilakukan secara maksimal. Terdapat beberapa *layer* yang perlu diintegrasikan sehingga pengembangan *pipeline* CI/CD yang matang dapat memudahkan proses pengembangan dan *deployment* ke depannya.
d. Penerapan *repository* yang lebih baik sehingga memberikan kemudahan bagi pengembangan lanjutan untuk melakukan *tracking* dan peningkatan (*update*) dari aplikasi yang dibangun.

e. *Testing* yang menyeluruh perlu dilakukan karena aplikasi PreciFood ditujukan bagi publik sehingga perlu dilakukan pengecekan secara menyeluruh selain dilakukannya *functional testing* melalui *blackbox testing*. *Testing* ini di antaranya *performance testing*, *stress testing,* hingga *penetration testing*.
f. Saat ini, penerapan *Role Based Access Control* (RBAC) masih terbatas pada level *endpoint* API sehingga sulit untuk mengelola hak akses secara rinci. Pengembangan RBAC yang lebih dinamis diperlukan agar akses dapat diatur hingga ke level modul dan fitur tertentu sehingga menjadi fleksibel dan lebih aman.

DAFTAR PUSTAKA

Boorsma A, van SE, de HI, Hogenelst K, van EM, Wopereis S, Rouhani-Rankouhi T, van dBT, Pasman W, van OB, Anthony JC. 2017. Systems biology of personalized nutrition. *Nutrition Reviews*. 75(8):579–599. doi:10.1093/nutrit/ nux029.

Bush CL, Blumberg JB, El-Sohemy A, Minich DM, Ordovás JM, Reed DG, Behm VAY. 2020. Toward the Definition of Personalized Nutrition: A Proposal by The American Nutrition Association. *J Am Coll Nutr*. 39(1):5–15. doi:10.1080/07315724.2019.1685332.

Carreiro AL, Dhillon J, Gordon S, Higgins KA, Jacobs AG, McArthur BM, Redan BW, Rivera RL, Schmidt LR, Mattes RD. 2016. The Macronutrients, Appetite, and Energy Intake. *Annu Rev Nutr*. 36(1): 73–103. doi: 10.1146/annurev-nutr-121415-112624.

Costa B, Pires PF, Delicato FC, Merson P. 2014. Evaluating a Representational State Transfer (REST) architecture: What is The Impact of REST in My Architecture?. 2014 IEEE/IFIP Conference on Software Architecture; 2014 April 7-11; Sydney, Australia. Sydney: hlm 105-114. https://ieeexplore.ieee.org/document/6827107.

[EFSA] European Food Safety Authority. 2017. Dietary Reference Values for Nutrients Summary Report. *EFSA Supporting Publication*. doi: 10.2903/sp.efsa.2017.e15121.

Galanakis CM. 2021. Functionality of Food Components and Emerging Technologies. *Foods*. 10(1):128. doi: 10.3390/foods10010128.

Hanafi A, Sukarsa IM, Wiranatha AAKAC. 2017. Pertukaran Data antar Database dengan Menggunakan Teknologi API. *Lontar Komputer*. 8(1):22–30. doi: 10.24843/LKJITI.2017.v08.i01.p03.

Kang J, Jun J, Arendt SW. 2015. Understanding customers' healthy food choices at casual dining restaurants: Using the Value–Attitude–Behavior model. *International Journal of Hospitality Management*. 48(1):12–21. [diunduh 2024 Okt 30]. doi:10.1016/j.ijhm.2015.04.005.

[Kemenkes] Kementerian Kesehatan Republik Indonesia. 2017. *Rencana Aksi Kegiatan Pengendalian Penyakit Tidak Menular Revisi I*. Jakarta: Kementrian Kesehatan RI.

Máriás Z, Molnár B. 2020. A Study on Formal Consistency Evaluation of Backend and Frontend Business Logic in a Modern Client-Server Application. 2020 Proceedings of the 11th International Conference on Applied Informatics;

2020 Januari 29-31; Eger, Hungaria. Eger: hlm 224-231. https://ceur-ws.org/.

Pressman RS, Maxim BR. 2020. *Software Engineering: A Practitioners Approach 9th ed*. New York (US): McGraw-Hill.

Rashid A, Chaturvedi A. 2019. Cloud Computing Characteristics and Services: A Brief Review. *IJCSE International Journal of Computer Sciences and Engineering*. 7(2): 421-426. doi: 10.26438/ijcse/v7i2.421426.

Savarino G, Corsello A, Corsello G. 2021. Macronutrient balance and micronutrient amounts through growth and development. *Italian Journal of Pediatrics*. 47(1). doi:10.1186/s13052-021-01061-0.

Seminar KB, Damayanthi E, Suyatma NE, Priandana K, Imantho H, Ligar BW, Seminar AU. 2024. An Intelligent And Precise System For Commercial Food Selection. *Jurnal Keteknikan Pertanian*. 12(1):139-152. doi: 10.19028/jtep.012.1.140-152.

Seminar KB, Damayanthi E, Priandana K, Imantho H, Ligar BW, Seminar AU, Krishnajaya AD, Aditya MR, Suherman MIH, Fillah IF. 2025. AI-based system for food and beverage selection towards precision nutrition in Indonesian restaurants. *Frontiers in Nutrition*. 12. doi: 10.3389/fnut.2025.1590523.

Shergill-Bonner R. 2017. Micronutrients. *Paediatrics and Child Health*. 27(8):357-362. doi: 10.1016/j.paed.2017.04.002.

Tan ZYJ, Hasa MM, Wong MY, Ramasamy RK. 2022. Implementation Approach of Unit and Integration Testing Method Based on Recent Advancements in Functional Software Testing. *Journal of System and Management Sciences*. 12(4):85-100. doi: 10.33168/JSMS.2022.0406.

Tapsell LC, Neale EP, Satija A, Hu FB. 2016. Foods, Nutrients, and Dietary Patterns: Interconnections and Implications for Dietary Guidelines. *Advances in Nutrition*. 7(3): 445–454. doi:10.3945/an.115.011718.

Uzair W, Naz S. 2023. Six-Tier Architecture for AI-Generated Software Development: A Large Language Models Approach. *Research Square Platform LLC*. doi: 10.21203/rs.3.rs-3086026/v1.

van Eyk E, Toader L., Talluri S, Versluis L, Uta A, Iosup A. 2018. Serverless is More: From PaaS to Present Cloud Computing. *IEEE Internet Computing*. 22(5), 8–17. doi:10.1109/mic.2018.053681358

[WHO] World Health Organization. 2018. *Noncommunicable Diseases Country Profiles 2018*. Geneva: WHO.