TUGAS AKHIR - EF234801

PENGEMBANGAN BACKEND MODUL POINT OF SALE TOKO BAHAN BAKU HALAL PADA APLIKASI TERATAI DENGAN MENGGUNAKAN CLEAN ARCHITECTURE (STUDI KASUS: ITS MART)

NAILY KHAIRIYA

NRP 5025201244

Dosen Pembimbing I

Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

NIP 198508262015042002

Dosen Pembimbing II

Siska Arifiani, S.Kom., M.Kom.

NIP 1990202012034

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

TUGAS AKHIR – EF234801

PENGEMBANGAN BACKEND MODUL POINT OF SALE TOKO BAHAN BAKU HALAL PADA APLIKASI TERATAI DENGAN MENGGUNAKAN CLEAN ARCHITECTURE (STUDI KASUS: ITS MART)

NAILY KHAIRIYA

NRP 5025201244

Dosen Pembimbing I

Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

NIP 198508262015042002

Dosen Pembimbing II

Siska Arifiani, S.Kom., M.Kom.

NIP 1990202012034

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – EF234801

BACKEND DEVELOPMENT OF POINT OF SALE MODULE FOR HALAL RAW MATERIAL STORE ON TERATAI APPLICATION USING CLEAN ARCHITECTURE (CASE STUDY: ITS MART)

NAILY KHAIRIYA

NRP 5025201244

Advisor I

Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

NIP 198508262015042002

Advisor II

Siska Arifiani, S.Kom., M.Kom.

NIP 1990202012034

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2024

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN BACKEND MODUL POINT OF SALE TOKO BAHAN BAKU HALAL PADA APLIKASI TERATAI DENGAN MENGGUNAKAN CLEAN ARCHITECTURE (STUDI KASUS: ITS MART)

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **NAILY KHAIRIYA**
NRP. 5025201244

Disetujui oleh Tim Penguji Tugas Akhir:

1. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. Pembimbing
2. Siska Arifiani, S.Kom., M.Kom. Ko-pembimbing
3. Dr. Kelly Rossa Sungkono, S.Kom., M.Kom. Penguji
4. Bagus Jati Santoso, S.Kom., Ph.D. Penguji

SURABAYA
Juni, 2024

(Halaman ini sengaja dikosongkan)

APPROVAL SHEET

BACKEND DEVELOPMENT OF POINT OF SALE MODULE FOR HALAL RAW MATERIAL STORE ON TERATAI APPLICATION USING CLEAN ARCHITECTURE (CASE STUDY: ITS MART)

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a bachelor's degree at
Undergraduate Study Program of Informatics
Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By: **NAILY KHAIRIYA**
NRP. 5025201244

Approved by Final Project Examiner Team:

1. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. Advisor
2. Siska Arifiani, S.Kom., M.Kom. Co-Advisor
3. Dr. Kelly Rossa Sungkono, S.Kom., M.Kom. Examiner
4. Bagus Jati Santoso, S.Kom., Ph.D. Examiner

SURABAYA
June, 2024

(Halaman ini sengaja dikosongkan)

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	: Naily Khairiya / 5025201244
Departemen	: Teknik Informatika
Dosen Pembimbing / NIP	: 1. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. / 198508262015042002
	2. Siska Arifiani, S.Kom., M.Kom. / 1990202012034

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Pengembangan *Backend* Modul *Point of Sale* Toko Bahan Baku Halal pada Aplikasi Teratai dengan Menggunakan *Clean Architecture* (Studi Kasus: ITS Mart)" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 30 Juni 2024

Mahasiswa

(Naily Khairiya)

NRP. 5025201244

Mengetahui,

Dosen Pembimbing I	Dosen Pembimbing II
	
(Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc)	(Siska Arifiani, S.Kom., M.Kom.)
NIP. 198508262015042002	NIP. 1990202012034

(Halaman ini sengaja dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP	: Naily Khairiya / 5025201244
Department	: Informatics
Advisor / NIP	: 1. Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. / 198508262015042002
	2. Siska Arifiani, S.Kom., M.Kom. / 1990202012034

Hereby declare that Final Project with the title of "Backend Development of Point of Sale Module for Halal Raw Material Store on Teratai Application Using Clean Architecture (Case Study: ITS Mart)" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

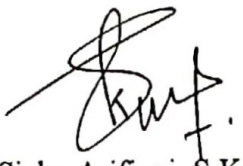
Surabaya, 30 June 2024

Student

(Naily Khairiya)

NRP. 5025201244

Acknowledge,

Advisor I	Advisor II
	
(Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc)	(Siska Arifiani, S.Kom., M.Kom.)
NIP. 198508262015042002	NIP. 1990202012034

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN BACKEND MODUL POINT OF SALE TOKO BAHAN BAKU HALAL PADA APLIKASI TERATAI DENGAN MENGGUNAKAN CLEAN ARCHITECTURE (STUDI KASUS: ITS MART)

Nama Mahasiswa / NRP : Naily Khairiya / 5025201244
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing : Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

Abstrak

Indonesia merupakan negara dengan populasi Muslim terbesar di dunia. Namun, Indonesia belum berada di peringkat sepuluh besar dalam hal produksi makanan halal, sehingga menimbulkan kekhawatiran akan integritas kehalalan produk makanan. Tantangan yang ada saat ini yaitu *traceability* dan aksesibilitas informasi yang terbatas dalam rantai pasokan makanan halal. Para pelaku bisnis, termasuk Toko Bahan Baku Halal, memegang peran yang sangat penting dalam menjaga kehalalan produk. Sebagai Toko Bahan Baku Halal pertama di tingkat perguruan tinggi, ITS Mart masih belum memiliki adanya sistem *point of sale* yang dapat menyimpan informasi terkait kehalalan produk dalam rangkaian pasokannya. Oleh karena itu, pada tugas akhir ini dikembangkan *back-end* modul *point of sale* toko bahan baku halal pada aplikasi Teratai untuk ITS Mart.

Proses pengembangan dilakukan dengan menggunakan metode *Systems Development Life Cycle* (SDLC) dengan model *waterfall*. Modul dikembangkan menggunakan *Clean Architecture* sebagai arsitekturnya sehingga sistem dapat beradaptasi dengan mudah apabila ada perubahan. Sistem telah melalui pengujian fungsional positif dan negatif untuk memastikan kemampuannya dalam mengelola proses penjualan produk bahan baku halal. Selain itu, pengujian non-fungsional seperti performa dan keamanan juga telah dilakukan.

Tugas akhir ini menunjukkan keberhasilan implementasi sistem *backend* POS untuk toko bahan baku halal di ITS Mart, yang dibuktikan dari hasil pengujian fungsional dan non-fungsional. Sistem yang terdiri dari 51 fungsional telah diuji oleh penulis serta dua responden lainnya dan 100% implementasi telah sesuai dengan skenario yang ditetapkan. Pengujian non-fungsional menunjukkan sistem dapat diakses melalui berbagai perangkat dan *browser* serta kontrol akses yang sesuai dengan peran pengguna. Berkaitan dengan performa, sistem mampu menangani 15 transaksi per detik dengan waktu respons rata-rata 53 ms. Pemisahan sistem POS dari sistem TERATAI meningkatkan performa dengan latensi rata-rata 235,5 ms dibandingkan 282,22 ms pada TERATAI.

Dengan demikian, implementasi ini tidak hanya berhasil memenuhi kebutuhan ITS Mart, tetapi juga berpotensi meningkatkan kepercayaan konsumen terhadap produk halal, serta memberikan referensi bagi toko bahan baku halal lainnya dalam pengembangan sistem POS yang terintegrasi dan transparan.

Kata kunci: *Clean Architecture*, Halal, ITS Mart, *Point of Sale*, Rantai Pasok Makanan, Sistem *Traceability*, Toko Bahan Baku.

(Halaman ini sengaja dikosongkan)

BACKEND DEVELOPMENT OF POINT OF SALE MODULE FOR HALAL RAW MATERIAL STORE ON TERATAI APPLICATION USING CLEAN ARCHITECTURE (CASE STUDY: ITS MART)

Student Name / NRP : Naily Khairiya / 5025201244
Department : Informatics FTEIC – ITS
Advisor : Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

Abstract

Indonesia, as the country with the largest Muslim population in the world, faces significant challenges in ensuring the integrity of halal food products, despite not yet being ranked among the top ten in halal food production globally. Current challenges include traceability and limited accessibility to information within the halal food supply chain. Business entities, including raw material store, play a crucial role in maintaining the halal integrity of products. As the first halal raw material store at the university level, ITS Mart currently lacks a point of sale system capable of storing information related to the halal status of its product supply chain. Therefore, this thesis aims to develop a back-end point of sale module for a halal raw material store within the Teratai application at ITS Mart.

The development process follows the Systems Development Life Cycle (SDLC) using the waterfall model. The module is implemented using Clean Architecture to facilitate easy adaptation to future changes. Functional testing, encompassing both positive and negative scenarios, ensures the system's capability to manage the sale of halal ingredient products. Non-functional testing, including performance and security assessments, has also been conducted.

This final project demonstrates the successful implementation of the backend POS system for the halal raw material store at ITS Mart, as evidenced by the results of functional and non-functional testing. The system, which consists of 51 functionalities, has been tested by the author and two other respondents, with 100% of the implementations aligning with the predetermined scenarios. Non-functional testing shows that the system can be accessed through various devices and browsers, and access control is appropriate for the user roles. In terms of performance, the system is capable of handling 15 transactions per second with an average response time of 53 ms. The separation of the POS system from the TERATAI system has improved performance, with an average latency of 235.5 ms compared to 282.22 ms on the TERATAI system.

Therefore, this implementation not only successfully meets the business needs of ITS Mart but also has the potential to increase consumer confidence in halal products and provide a reference for other halal raw material stores in developing an integrated and transparent POS system.

Keywords: Clean Architecture, Food Supply Chain, Halal, ITS Mart, Point of Sale, Raw Material Store, Traceability System.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul:

"PENGEMBANGAN BACKEND MODUL POINT OF SALE TOKO BAHAN BAKU HALAL PADA APLIKASI TERATAI DENGAN MENGGUNAKAN CLEAN ARCHITECTURE (STUDI KASUS: ITS MART)"

Begitu besar harapan penulis terhadap terselesainya buku Tugas Akhir ini. Semoga dengan adanya buku Tugas Akhir ini dapat memberikan wawasan dan kontribusi nyata terhadap perkembangan ilmu teknologi pada masa kini maupun di masa mendatang. Dalam proses penyusunan Tugas Akhir ini, penulis banyak mendapat bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, dengan segala kerendahan hati, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT. dan Nabi Muhammad SAW. atas kebaikan dan nikmat yang diberikan kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Kedua orang tua penulis, Ibu Evi Wigati dan almarhum Bapak Sugiman, yang senantiasa menjadi alasan bagi penulis untuk terus berjuang. Terima kasih atas segala cinta kasih dan doa yang telah tercurahkan di sepanjang hidup penulis. Tiada lain kepada kalianlah Tugas Akhir ini dipersembahkan.
3. Ibu Ir. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. selaku pembimbing Tugas Akhir yang telah memberikan bimbingan, arahan, serta motivasi kepada penulis selama penyusunan Tugas Akhir ini.
4. Ibu Siska Arifiani, S.Kom., M.Kom. selaku pembimbing Tugas Akhir yang telah memberikan nasihat dan dorongan semangat.
5. Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku kepala Departemen Teknik Informatika ITS.
6. Bapak dan Ibu Dosen yang telah memberikan ilmunya dan telah banyak membantu penulis selama berkuliah di Informatika ITS.
7. Tim Teratai yang telah memberikan dukungan yang sangat membantu sehingga penelitian ini dapat diselesaikan.
8. Ibu Ifa Faulina selaku narasumber yang membantu dalam pengumpulan kebutuhan tugas akhir ini.
9. Diah Ika Wati kakak penulis yang mendukung dan memberikan inspirasi.
10. Aisha Syifa Alinarrahman adik penulis yang selalu menyemangati.
11. Teman-teman KEJORA, BSO-K, dan FLFM yang senantiasa mendengarkan keluh kesah penulis.
12. Teman-teman angkatan 2020 yang telah membantu penulis baik secara akademik maupun non akademik selama berkuliah di Departemen Teknik Informatika ITS.
13. Seluruh pihak lainnya yang tidak dapat disebutkan di sini, yang telah banyak membantu penulis dalam penyusunan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna. Oleh karena itu, penulis memohon maaf atas segala kesalahan yang terdapat dalam Tugas Akhir ini. Penulis sangat menghargai kritik dan saran yang membangun dari berbagai pihak sebagai bahan perbaikan di masa mendatang. Akhir kata, penulis berharap semoga Tugas Akhir ini bisa bermanfaat bagi pengembangan ilmu pengetahuan.

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR SINGKATAN

API	: *Application Programming Interface*
BPJPH	: Badan Penyelenggara Jaminan Produk Halal
BSON	: *Binary* JSON
CSS	: *Cascading Style Sheets*
GUI	: *Graphical User Interface*
HTML	: *HyperText Markup Language*
HTTP	: *Hypertext Transfer Protocol*
JSON	: *JavaScript Object Notation*
LPPOM	: Lembaga Pengkajian Pangan, Obat-obatan & Kosmetika
MUI	: Majelis Ulama Indonesia
MVC	: *Model View Controller*
NPM	: *Node Package Manager*
POS	: *Point of Sale*
RPH	: Rumah Potong Hewan
SDLC	: *Software Development Life Cycle*
SH	: Sertifikat Halal
Tobaku	: Toko Bahan Baku
UMKM	: Usaha Mikro Kecil dan Menengah
VPS	: *Virtual Private Server*

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar belakang

Indonesia merupakan negara dengan jumlah populasi Muslim terbesar di dunia, di mana mayoritas penduduknya memeluk agama Islam sebagai ajaran utamanya. Seiring berjalannya waktu, jumlah umat Muslim terus meningkat dengan pesat. Menurut laporan Global Islamic Economy(GIE) tahun 2019-2020, jumlah populasi Muslim di dunia mencapai 1,8 miliar orang. Laporan tersebut memperkirakan bahwa Muslim menghabiskan US$2,2 triliun pada tahun 2018 di sektor makanan, farmasi, dan gaya hidup yang dipengaruhi oleh kebutuhan konsumsi etis berdasarkan keyakinan Islam. Pengeluaran ini mencerminkan pertumbuhan sebesar 5,2% secara tahunan dan diproyeksikan mencapai US$3,2 triliun pada tahun 2024.

Meskipun demikian, menurut laporan GIE tahun 2019-2020, Indonesia tidak berhasil mencapai peringkat sepuluh teratas dalam sektor makanan halal. Tingginya permintaan terhadap produk halal menimbulkan kekhawatiran terkait kehalalan produk makanan. "Halal" berarti diperbolehkan dan sah, sementara "haram" menunjukkan sesuatu yang dilarang. Suatu makanan dapat dikatakan halal apabila memenuhi persyaratan, seperti bukan termasuk ke dalam bahan yang haram, tidak terkontaminasi dengan bahan haram atau memabukkan, bebas dari zat beracun dan darah, serta disembelih dengan metode sesuai dengan syariat Islam (Khattak et al., 2011). Pentingnya aspek kehalalan ini menekankan perlunya manajemen integritas makanan yang efektif.

Mengelola integritas makanan adalah tanggung jawab semua pihak dalam rantai pasokan dan harus sepenuhnya terintegrasi serta mematuhi suatu jaminan sistem atau proses (Tan et al., 2017). Pada penelitian sebelumnya, Vanany et al. (2020) telah menyoroti keterbatasan informasi dan ketertelusuran dalam konteks penerapan teknologi untuk makanan halal dalam rantai pasokan. Meskipun konsumen dapat memverifikasi keaslian sertifikasi halal yang valid, namun data terkait produsen, perusahaan bahan baku halal, atau pelaku bisnis yang terlibat dalam suatu produk masih belum dapat diakses. Sejalan dengan konsep tersebut, Toko Bahan Baku (Tobaku) Halal tidak hanya berperan sebagai saluran distribusi, tetapi juga menjadi elemen kunci dalam menjaga kehalalan makanan sepanjang perjalanan dari produsen hingga konsumen.

Di sisi lain, ITS Mart menciptakan sejarah sebagai Tobaku halal pertama yang berbasis perguruan tinggi. Sebagai pionir Tobaku halal, belum ada sistem yang menangani penelusuran kehalalan suatu produk melalui Tobaku halal. Sistem tersebut akan berkaitan erat dengan *Point of Sale* (POS) Tobaku halal, karena pada modul tersebut, data terkait Tobaku halal akan diintegrasikan dan diproses untuk memastikan keberlanjutan kehalalan produk.

Pada tugas akhir ini, dikembangkan modul *Point of Sale* pada Toko Bahan Baku Halal ITS Mart pada aplikasi Teratai. Aplikasi ini dikembangkan dengan *Clean Architecture* dalam penulisan kode yang bertujuan untuk mempermudah perawatan dan pengelolaan aplikasi di masa mendatang. Diharapkan dengan adanya modul *Point of Sale* yang terintegrasi dengan aplikasi Teratai dapat memungkinkan pelacakan jual beli produk halal dari produsen ke konsumen melalui Tobaku halal serta memperkuat transparansi alur pasokan produk.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut.

1. Bagaimana penerapan *Clean Architecture* dalam pengembangan modul *Point of Sale* pada aplikasi Teratai?
2. Bagaimana rancangan dan implementasi modul *Point of Sale* untuk menangani proses penjualan produk bahan baku halal di ITS Mart?
3. Bagaimana integrasi modul *Point of Sale* dengan aplikasi Teratai?
4. Bagaimana pengujian modul *Point of Sale* aplikasi Teratai?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Aplikasi yang dibangun adalah aplikasi berbasis *web*.
2. Aplikasi dibangun dengan menggunakan kerangka kerja Express.js.
3. Uji coba modul *Point of Sale* dilakukan pada lingkungan *development*.
4. Transaksi jual beli melibatkan pemindaian QR *code* dan *input batch* produk untuk mengidentifikasi daftar barang yang akan dibeli.
5. Produk yang dikelola di dalam sistem adalah daging.
6. Sistem POS Teratai terbatas untuk pemasok dan pembeli yang telah terdaftar secara resmi di dalam aplikasi Teratai.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut.

1. Menerapkan *Clean Architecture* dalam pengembangan modul *Point of Sale* pada aplikasi Teratai.
2. Merancang dan mengimplementasikan modul *Point of Sale* untuk menangani proses penjualan produk bahan baku halal di ITS Mart.
3. Mengintegrasikan modul *Point of Sale* dengan aplikasi Teratai.
4. Menguji modul *Point of Sale* aplikasi Teratai.

1.5 Manfaat

Adapun manfaat dari penyusunan tugas akhir ini adalah sebagai berikut:

1. membantu ITS Mart dalam mengembangkan sistem *point of sale* yang sesuai dengan alur bisnis yang berlaku,
2. memberikan kejelasan dan keyakinan kepada masyarakat terkait asal-usul dari kehalalan produk halal,
3. menjadi pedoman bagi pelaku bisnis, khususnya Tobaku halal, dalam mengembangkan sistem *Point of Sale* pada sistem *halal traceability*.

BAB II
TINJAUAN PUSTAKA

Bagian ini mencakup penelitian terdahulu, aplikasi yang berkaitan dengan modul yang dikembangkan, serta penjelasan teori yang berkaitan dengan pengembangan perangkat lunak. Penjelasan ini dimaksudkan untuk memberikan gambaran tentang sistem yang dikembangkan dan berperan sebagai pendukung dalam proses pengembangan perangkat lunak.

2.1 Penelitian Terkait

Dalam tugas akhir ini, terdapat beberapa penelitian dan aplikasi terkait yang dijadikan sebagai referensi untuk pengerjaan tugas akhir. Tugas akhir ini sangat berhubungan erat dengan *traceability* atau penelusuran produk halal. Tabel 2.1 merupakan beberapa penelitian terdahulu terkait dengan *traceability*.

Tabel 2.1 Penelitian Terdahulu yang Berkaitan dengan Traceability

No	Penulis dan Judul Penelitian	Fokus Penelitian	Produk Halal	Rantai Pasok	Hasil
1	Vanany, I., Rakhmawati, N. A., Sukoso, S., & Soon, J. M. Indonesian Halal Food Integrity: Blockchain Platform.	Pengembangan kerangka konseptual dan arsitektur *blockchain*, terutama dengan menggunakan *Hyperledger Fabric* untuk meningkatkan integritas makanan halal di Indonesia.	Makanan Halal	Pemasok, Produsen, Distributor, dan Pengecer.	Kerangka konseptual teknologi *blockchain* di antara pelaku bisnis (usaha mikro kecil, menengah, dan besar), pengawas halal, industri audit halal, dan otoritas halal pemerintah.
2	Vanany, I., Maftuhah, D. I., Soeprijanto, A., Sukoso, & Zulhafizh, M. Modelling Halal Internal Traceability in Open Source ERP System for Chicken Meat Processing Company.	Pengembangan sistem jejak halal internal dalam sistem Odoo ERP *open source* yang digunakan untuk memastikan kehalalan bahan baku.	Ayam	Internal(administrator, *purchasing*, *industry*, manufaktur) dan eksternal (*supplier*)	Sistem halal internal *traceability* dalam Odoo ERP dengan modul katalog vendor, *purchasing*, katalog produk, *manufacturing*, dan *inventory*.
3	Akbar, A., Rakhmawati, N. A., & Vanany, I. Halal Blockchain Application for a Chicken Slaughtering Factory.	Sistem berbasis *blockchain* untuk mendukung jejak halal di *industry* pemotongan ayam.	Ayam	Pemasok(termasuk Rumah Potong Hewan), Pembeli, dan Logistik.	Sistem *blockchain* yang dirancang berdasarkan implementasi makanan halal di Indonesia dan memenuhi kebutuhan pelacakan rantai pasok makanan halal dengan

No	Penulis dan Judul Penelitian	Fokus Penelitian	Produk Halal	Rantai Pasok	Hasil
					memanfaatkan sertifikat halal dan *batch* produk.
4	Tan, A., Gligor, D., & Ngah, A. Applying Blockchain for Halal food traceability.	*Traceability* untuk rantai pasokan makanan halal di Malaysia agar sesuai dengan persyaratan Halal.	Daging	Rumah Potong Hewan, Logistik, Pabrik, *warehouse*	Kerangka *traceability* dengan menggunakan *blockchain* dengan tiga rantai pasok berbeda sesuai dengan kehidupan nyata dan memanfaatkan QR *Code* serta *batch* produk untuk penelusuran.
5	Safitri, A., Fahmi, M. Z., & Gunawan, S. Kajian Penelusuran Produk Halal Kornet Daging Sapi.	Penelusuran kehalalan kornet daging sapi di Indonesia dengan dengan metode studi literatur untuk menganalisis *traceability*, peran sertifikasi halal BPJPH, dan kesesuaian produk dengan standar MUI.	Delapan produk kornet daging sapi.	Rumah Potong Hewan dan Industri Pengolahan.	Kornet sapi produk Indonesia telah bersertifikasi halal MUI, sementara kornet sapi impor memiliki variasi, beberapa tanpa sertifikasi halal dari negara pembuat dan beberapa dengan sertifikasi namun belum diakui oleh MUI.

Pada modul *point of sale*, terdapat beberapa aplikasi yang sebelumnya sudah ada. Sebagai contoh, aplikasi WarungKu yang merupakan aplikasi *point of sale* dan *marketing* yang diciptakan oleh tim Kuliah Kerja Nyata Pengabdian kepada Masyarakat (Abmas) dari Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS). Aplikasi tersebut memiliki tiga fitur utama, yaitu stok, transaksi, dan riwayat (Dyantoro, 2023). Perbandingan fitur *point of sale* secara lengkap pada aplikasi Teratai dan pada aplikasi yang sudah ada sebelumnya ditunjukkan pada Tabel 2.2.

Tabel 2.2 Perbandingan Fitur Aplikasi Point of Sale

Fitur	Kasir Pintar (Pro)	Qasir	Warung Ku	POS Teratai
Register akun perusahaan	✓	✓	✓	✓
Register akun karyawan	✓	✓		✓
Login akun sesuai peran	✓	✓		✓
Manajemen produk	✓	✓		✓
Manajemen variasi harga	✓			✓
Manajemen diskon	✓			✓
Manajemen transaksi	✓	✓	✓	✓
Menambahkan transaksi *preorder*				✓
Setruk pembelian	✓		✓	✓
Laporan penjualan	✓	✓	✓	✓
Manajemen karyawan	✓	✓		✓

Fitur	Kasir Pintar (Pro)	Qasir	Warung Ku	POS Teratai
Manajemen *shift* kasir	✓			✓
Manajemen konsumen	✓			✓
Manajemen pemasok	✓			✓
Kemampuan memindai QR *code* produk				✓
Integrasi dengan e-*wallet*	✓	✓		✓
Traceability produk halal				✓
Integrasi dengan sertifikat BPJPH				✓

Tabel 2.2 menunjukkan bahwa modul POS Teratai memiliki sejumlah keunggulan dibandingkan dengan aplikasi lain seperti Kasir Pintar (Pro), Qasir, dan WarungKu. Aplikasi ini menawarkan fitur-fitur unggulan seperti manajemen produk, variasi harga, transaksi *preorder* maupun non-*preorder*, laporan, *shift* kasir, dan konsumen. Selain itu, Teratai juga mendukung kemampuan memindai QR *code* produk untuk *traceability* produk halal dan integrasi dengan sertifikat BPJPH. Fitur-fitur ini membuat modul POS yang dikembangkan menjadi solusi yang lebih unggul dalam menjaga kehalalan suatu produk berdasarkan sertifikat yang diberikan oleh BPJPH.

2.2 Dasar Teori

Bagian dasar teori menjelaskan dasar teori yang digunakan untuk mendukung penulisan tugas akhir ini.

2.2.1 Bahan Baku Halal

Bahan baku merupakan bahan yang digunakan dalam produksi utama atau bahan penolong pembuatan suatu produk. Bahan baku diperjualbelikan karena bahan baku merupakan faktor produksi. Dalam beberapa kasus, bahan baku dapat dibagi menjadi dua kategori: langsung dan tidak langsung. Bahan baku langsung merupakan bahan yang secara langsung digunakan oleh perusahaan dalam pembuatan produk jadi, seperti penggunaan kayu untuk membuat meja. Sementara itu, bahan baku tidak langsung merujuk pada bahan yang digunakan secara komprehensif namun bukan merupakan bagian dari produk akhir.

Bahan baku dapat juga diklasifikasikan berdasarkan sifat bagaimana barang tersebut diekstraksi. Jenis-jenis tersebut antara lain:

a. **Bahan baku yang ditambang** yang diambil dari dalam bumi, seperti bijih, batu, logam, minyak, dan batu bara.
b. **Bahan baku nabati** berasal dari pohon atau tumbuhan, antara lain buah-buahan, kacang-kacangan, bunga, sayuran, damar, kayu, kapas, dan lateks.
c. **Bahan baku hewani** diekstraksi dari hewan seperti susu, daging, bulu, kulit, dan wol.

Bahan baku dalam makanan dapat berupa barang yang berdiri sendiri seperti daging, susu, buah-buahan, dan sayuran. Bahan baku tersebut juga dapat merujuk pada bahan-bahan yang dimasukkan ke dalam suatu makanan atau resep. Misalnya, susu merupakan bahan baku yang digunakan dalam produksi keju dan *yogurt* (Banton, 2023). Dalam tugas akhir ini, bahan baku yang diteliti adalah bahan baku yang berbasis daging.

Produk yang bersertifikat halal merupakan hasil dari bahan-bahan yang diproses menggunakan peralatan dan prosedur yang memenuhi standar halal. Seluruh komponen yang terlibat dalam proses sertifikasi halal mencakup seluruh bahan yang digunakan atau yang memiliki kontak langsung dengan produk, guna memastikan bahwa produk tersebut memenuhi kriteria halal. Jenis bahan yang termasuk dalam proses ini mencakup bahan baku, bahan tambahan, bahan penolong, bahan pengemas primer, serta bahan pelumas yang berinteraksi langsung dengan produk atau fasilitas yang terhubung dengannya. Bahan tertentu harus didukung dengan sertifikat halal dari lembaga sertifikat halal yang diakui, misalnya: bahan hewani yang wajib melalui proses penyembelihan yang sesuai dengan syariah (Nadha, 2022).

Dapat disimpulkan bahwa, bahan baku halal merupakan materi yang memenuhi kriteria kehalalan dalam Islam dan digunakan dalam pembuatan produk yang sesuai dengan prinsip syariah. Bahan-bahan tersebut juga harus diproses dengan alat dan prosedur yang memenuhi standar halal.

2.2.2 Toko Bahan Baku Halal

Toko bahan baku halal merupakan tempat di mana beragam jenis bahan baku yang dijual telah melalui serangkaian verifikasi dan penjaminan kualitas untuk memastikan bahan baku yang dijual memenuhi prinsip-prinsip kehalalan. Toko bahan baku halal bukan hanya tempat transaksi komersial, melainkan juga bagian dari ekosistem ekonomi yang mendukung kebutuhan konsumen yang mengutamakan nilai-nilai kehalalan.

Gambar 2.1 Peresmian ITS Mart (Indahts, 2023).

Di Indonesia terdapat beberapa toko bahan baku halal. Pada tahun 2023, Institut Teknologi Sepuluh Nopember (ITS) kembali meresmikan tempat yang bertujuan untuk mendorong pertumbuhan ekonomi dan memenuhi kebutuhan masyarakat dengan membuka ITS Mart (Gambar 2.1). Toko ini merupakan salah satu inisiatif bisnis ITS yang berlokasi di kawasan kampus, yang berada di jalan Arief Rahman Hakim, Surabaya. ITS Mart secara resmi diresmikan melalui acara *grand opening* pada hari Jumat, 14 Juli 2023 (Indahts, 2023).

Dalam sambutannya, Tri Joko Wahyu Adi, ST MT PhD, Direktur Kerjasama dan Pengelolaan Usaha (DKPU) di Institut Teknologi Sepuluh Nopember (ITS), mengatakan bahwa ITS Mart merupakan suatu wadah kegiatan ekonomi yang bergerak pada sektor ritel. Seperti bisnis ritel pada umumnya, ITS Mart menyajikan berbagai produk kebutuhan sehari-hari, seperti makanan dan minuman, alat tulis kantor (ATK), serta peralatan rumah tangga (Indahts, 2023).

ITS Mart memiliki misi yaitu menjadikan ITS Mart menjadi toko bahan baku halal berbasis perguruan tinggi pertama di Indonesia. Sebagai upaya untuk mencapai misi tersebut, ITS Mart memerlukan keberanian untuk menjual bahan baku yang memiliki dasar produk daging dan tidak sekadar makanan atau minuman yang sudah jelas kehalalannya. Namun, saat ini untuk produk daging masih dalam tahap persiapan.

Pembeli yang ingin membeli produk di ITS Mart akan memiliki opsi untuk pembayaran. ITS Mart mengadopsi teknologi dalam sistem pembayaran, sehingga pembeli dapat melakukan pembayaran secara tunai maupun non-tunai. ITS Mart akan dibuka setiap hari mulai pukul 07.00 pagi hingga 22.00 malam (Indahts, 2023).

2.2.3 Halal Traceability

Dasar memilih makanan halal telah dinyatakan secara eksplisit dalam beberapa ayat Al-Quran yang menghimbau umat Islam untuk hanya memilih makanan yang diperbolehkan. Salah satu ayat tersebut artinya, “Makanlah apa yang telah Allah anugerahkan kepadamu sebagai rezeki yang halal lagi baik, dan bertakwalah kepada Allah yang hanya kepada-Nya kamu beriman.” (Al-Maidah 5:88).

Berdasarkan perintah di dalam Al-Qur'an tersebut, penting untuk memastikan kehalalan suatu produk. Saat ini, untuk melihat kehalalan suatu produk dapat diperiksa melalui *website* https://halalmui.org/. *Website* tersebut merupakan *website* dari LPPOM MUI yang merupakan Lembaga Pemeriksa Halal pertama di Indonesia. Suatu produk yang belum disertifikasi dapat disertifikasi dengan mendaftar sertifikasi halal melalui https://ptsp.halal.go.id/. Produk kemudian akan melalui serangkaian proses pengecekan kehalalan produk. Apabila produk memenuhi syarat-syarat halal maka akan diberikan sertifikat halal oleh Badan Penyelenggara Jaminan Produk Halal (BPJPH). Badan tersebut merupakan unit eselon I termuda di bawah Kementerian Agama (Kemenag) Republik Indonesia.

Dalam memastikan kehalalan suatu produk, tidak hanya penting untuk memeriksa kehalalannya. Sebagai contoh, untuk produk daging sapi, titik kritis halal daging sapi dimulai dari proses penyembelihan yang harus memenuhi syariat Islam dan memiliki SH BPJPH/MUI (Safitri et al., 2022). Titik kritis merupakan suatu tahapan proses produksi, baik itu untuk makanan, minuman, obat, kosmetik, barang konsumen, dan lainnya, yang memiliki potensi di mana suatu produk dapat menjadi tidak halal. Produk daging sapi yang keluar dari rumah

potong hewan kemudian didistribusikan ataupun diolah dengan bahan lain sebelum pindah ke tangan konsumen merupakan tahapan yang panjang. Tahapan yang panjang tersebut belum dapat dilihat hanya dengan nomor sertifikat halal. Oleh karena ini, perlu sistem *traceability* mengidentifikasi dan melacak setiap langkah dengan cermat.

Kemampuan *traceability* dapat meningkatkan integritas proses pelabelan halal karena mendorong transparansi informasi di seluruh rantai pasokan. *Traceability* merupakan kemampuan untuk membedakan, mengidentifikasi, dan mengikuti pergerakan suatu pangan melalui seluruh tahap produksi, pemrosesan, dan distribusi. (Zhang & Bhatt, 2014).

Sistem *traceability* merupakan alat komunikasi yang membuat informasi tersedia di sepanjang rantai pasokan. Sistem akan mengumpulkan, mendokumentasikan, memelihara, dan menyediakan informasi mengenai keseluruhan proses pada rantai pasokan. *Traceability* akan menjamin kehalalan produk pada rantai pasok. Sistem *traceability* yang efektif melibatkan semua pihak dalam bisnis halal, termasuk produsen, pemasok, organisasi Islam, pemerintah, lembaga penelitian, dan konsumen. Dengan melibatkan semua pihak dalam rantai pasok, sistem ketertelusuran dapat menjadi lebih efektif dalam memastikan keamanan dan kualitas produk halal (Mohd Saifudin et al., 2017).

2.2.4 Teratai

Teratai merupakan kependekan dari "Telusur dan Pantau Halal Indonesia," yang merujuk pada sebuah aplikasi yang dirancang untuk melacak kehalalan produk daging dari Rumah Potong Hewan (RPH) hingga sampai ke tangan konsumen akhir. Dalam kerangka ini, pelaku bisnis yang terlibat melibatkan berbagai entitas, termasuk Rumah Potong Hewan (RPH), Industri Bahan Baku Halal, Toko Bahan Baku (Tobaku) Halal, dan Usaha Mikro Kecil dan Menengah (UMKM). Semua pihak yang terlibat dalam rangkaian ini diberikan dukungan dan bimbingan oleh pembina yang memastikan kesesuaian dengan standar kehalalan yang berlaku.

Setiap pelaku bisnis yang menghasilkan produk baru akan diberikan QR *code*. QR *code* tersebut menyimpan data penelusuran untuk suatu produk. Produk tersebut ditambahkan ke sistem dengan menggunakan id pada sertifikat halal yang diterbitkan oleh BPJPH. Hal ini digunakan untuk menjamin kehalalan produk yang akan ditelusuri. Setiap kemasan produk pada produksi baru tidak memungkinkan untuk sering diperbarui. Oleh karena itu, untuk membedakan produk pada setiap produksinya maka ditambahkan *batch. Batch* berupa memiliki tipe data *string* yang bisa mencangkup angka, huruf dan karakter lainnya.

Pada studi kasus toko bahan baku halal, QR *code* tidak akan dibuat karena barang hanya melewati toko tersebut dan tidak menciptakan produk baru. Meskipun demikian, data terkait tobaku halal tetap tercatat di dalam sistem ketika pembeli melakukan pembelian produk dari tobaku halal. Setiap pembeli akan menunjukkan QR *code* keanggotaan pada sistem, kemudian penjual akan memindai QR *code* tersebut. Data produk yang terjual akan tersimpan pada transaksi penjual dan juga tersimpan sebagai bahan pada pembeli. Dengan proses ini, informasi mengenai setiap produk, tersimpan dengan baik dan dapat diakses untuk ditampilkan serta ditelusuri.

2.2.5 Point of Sale

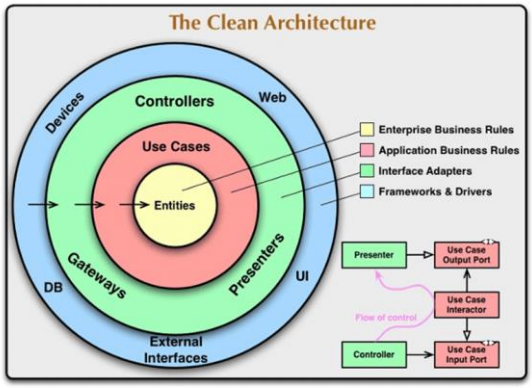
Sale atau penjualan merupakan hubungan antara dua pihak atau lebih yang melibatkan komunikasi, persuasi, dan membangun hubungan untuk mendorong transaksi dan memungkinkan pertukaran barang, jasa, atau ide. Memahami keinginan pembeli, memberikan solusi kepada pembeli, dan mendapatkan kepercayaan pembeli merupakan bagian penting dari proses tersebut. Penjualan dapat dikatakan selesai apabila pembeli menyetujui persyaratan dan memberikan kompensasi yang diperlukan serta kepemilikan barang atau jasa dialihkan dari penjual ke pembeli (Chandra, 2023).

Saat ini, banyak toko yang beroperasi di Indonesia yang menerapkan strategi penjualan produk kepada pembeli. Upaya ini melibatkan penggunaan rak modern, mewajibkan karyawan untuk mengenakan seragam, dan menerapkan sistem informasi penjualan yang dikenal sebagai *Point of Sales* (POS). Meskipun POS sering kali diidentifikasi dengan fungsi kasir, tetapi sistem ini bisa menjadi lebih dari itu. POS dapat mencakup analisis penjualan, manajemen inventaris, dan pelaporan.

2.2.6 Clean Architecture

Clean architecture, yang digagas oleh Robert C. Martin, merupakan suatu konsep arsitektur perangkat lunak yang bertujuan untuk mempermudah manajemen, modifikasi, dan pengujian perangkat lunak. Dalam bukunya berjudul "*Clean Architecture: A Craftsman's Guide to Software Structure and Design*," Martin menyatakan bahwa konsep ini mengambil inspirasi dari beberapa arsitektur perangkat lunak sebelumnya, termasuk *hexagonal architecture*, DCI (*data-context-interaction*), dan BCE (*boundary-control-entity*). Semua arsitektur tersebut memiliki tujuan utama, yaitu pemisahan konsep atau *separation of concern*, dengan membagi blok-blok kode sesuai dengan tujuan atau fitur masing-masing, untuk mengurangi ketergantungan antar-blok kode (Martin, 2018).

Pemisahan tersebut dilakukan dengan mengorganisir perangkat lunak ke dalam beberapa lapisan, termasuk minimal satu lapisan untuk logika atau aturan bisnis, dan lapisan lainnya untuk antarmuka pengguna dan sistem. *Clean architecture* bertujuan untuk menciptakan perangkat lunak yang tidak bergantung pada kerangka kerja, jenis antarmuka pengguna, atau basis data tertentu, serta lebih mudah diuji. Martin mengintegrasikan prinsip-prinsip ini ke dalam satu diagram, seperti yang terlihat Gambar 2.2.



Gambar 2.2 Ilustrasi Clean Architecture (Martin, 2018).

Dalam diagram tersebut, lapisan yang lebih dalam memiliki tingkatan yang lebih tinggi dalam perangkat lunak. Lingkaran luar merepresentasikan mekanisme, sementara lingkaran dalam mencakup kebijakan atau aturan. Prinsip yang ditetapkan dalam *clean architecture* adalah bahwa ketergantungan antar kode hanya dapat mengalir dari lingkaran luar ke lingkaran dalam. Tidak ada informasi yang seharusnya diketahui oleh lingkaran dalam terkait dengan apa yang ada di lingkaran luar, termasuk fungsi, kelas, variabel, atau entitas lainnya dari perangkat lunak. *Clean Architecture* terdiri dari lapisan *entities, use cases, interface adapters,* serta *Frameworks* dan *Drivers* (Martin, 2018).

2.2.6.1 Entities

Entities merupakan sebagai bagian paling dalam dari *Clean Architecture,* berfungsi sebagai tempat untuk aturan-aturan bisnis yang krusial. Sebuah entitas dapat berupa objek yang memiliki sejumlah metode, atau juga dapat berbentuk kumpulan struktur data dan fungsi yang terkait dengan aturan bisnis. Dalam *clean architecture, entities* menjadi pusat dari sistem, yang berarti bahwa *entities* tidak tergantung pada komponen eksternal dan tetap tidak terpengaruh oleh perubahan teknologi atau perangkat lunak. Dengan mempertahankan kemandiriannya, *entities* menjamin keutuhan dan konsistensi data serta logika bisnis dalam aplikasi(Martin, 2018).

2.2.6.1 Use Cases

Lapisan *use cases* mengandung aturan bisnis yang khusus untuk aplikasi. Lapisan ini mengemas dan menerapkan semua *use cases* dari sistem. *Use cases* ini mengatur alur data ke entitas dan dari entitas, dan mengarahkan entitas tersebut untuk menggunakan aturan bisnis untuk mencapai tujuan *use case*. Perubahan pada lapisan *usecase* seharusnya tidak dapat mempengaruhi entitas. Lapisan ini juga tidak terpengaruh oleh perubahan pada elemen eksternal seperti database, antarmuka pengguna, atau kerangka kerja lainnya (Martin, 2018).

2.2.6.2 Interface Adapters

Lapisan *interface adapters* adalah seperangkat adapter yang mengonversi data dari format yang paling sesuai untuk *usecase* dan entitas ke format yang paling sesuai untuk elemen eksternal seperti basis data atau *web*. Misalnya, lapisan ini akan sepenuhnya mengandung arsitektur MVC dari GUI. *View* dan *controller* semuanya termasuk dalam lapisan *interface adapters*. Model-model kemungkinan hanya berupa struktur data yang dilewatkan dari *controller* ke *usecase*, dan kemudian kembali dari *usecase* ke *view*. Di dalam lapisan ini juga terdapat adapter lain yang diperlukan untuk mengonversi data dari suatu bentuk eksternal, seperti layanan eksternal, ke bentuk internal yang digunakan oleh *usecase* dan entitas (Martin, 2018).

2.2.6.3 Frameworks dan Drivers

Lapisan terluar dari model pada Gambar 2.2 umumnya terdiri dari kerangka kerja dan alat-alat seperti database dan kerangka kerja. Secara umum, pada lapisan ini, hanya sedikit kode yang dikembangkan selain dari fungsi-fungsi yang memfasilitasi komunikasi dengan lapisan yang lebih dalam. Contohnya termasuk mengelola koneksi ke basis data agar data dapat diakses oleh lapisan di dalamnya, menggabungkan layanan eksternal atau pihak ketiga, dan mengaitkan antarmuka pengguna dengan API aplikasi (Martin, 2018).

Pengembangan modul POS toko bahan baku halal ITS Mart menggunakan *clean architecture*. ITS Mart baru saja resmi dibuka pada tanggal 14 Juli 2023, sementara produk jenis daging segar baru mulai tersedia pada Juni 2024. Karena pembukaan yang baru saja dilakukan, memungkinkan adanya perubahan dalam proses bisnis di masa depan. *Clean architecture* dipilih karena *clean architecture* memungkinkan perubahan dan penambahan fitur di masa depan menjadi lebih mudah dilakukan. *Clean architecture* memisahkan logika bisnis dari mekanisme implementasi, sehingga pemeliharaan sistem menjadi lebih efisien dan risikonya lebih sedikit. Dengan pemisahan yang jelas antara lapisan-lapisan perangkat lunak, setiap komponen terenkapsulasi dengan baik. Pemisahan tersebut mengurangi ketergantungan antar komponen dan memudahkan dalam pengujian serta perbaikan. Selain itu, *clean architecture* mendukung skalabilitas sistem, yang sangat penting bagi ITS Mart dalam menghadapi pertumbuhan dan perkembangan kebutuhan bisnis di masa depan. Struktur yang fleksibel dan modular ini memungkinkan tim pengembang untuk menambahkan fitur baru tanpa mengganggu komponen yang sudah ada.

2.2.7 JavaScript

JavaScript merupakan bahasa pemrograman yang banyak digunakan di dunia. Program yang ditulis dalam JavaScript adalah *single-thread*. Hal ini berarti, program JavaScript tidak dapat membuat *thread* tambahan untuk menjalankan tugas yang berjalan lama seperti operasi IO. Untuk mengatasi keterbatasan tersebut, pengembang menggunakan API JavaScript yang menerima parameter *callback.* Fungsi API tersebut memulai tugas, seperti permintaan HTTP dan mengembalikan ke *callee* (fungsi yang dipanggil). Ketika tugas selesai, mesin JavaScript mengeksekusi fungsi *callback* (secara asinkron) untuk memberi tahu program dan meneruskan program data yang dihasilkan oleh tugas tersebut. *Callback* digunakan untuk menangani tugas dengan cara yang tidak memblokir. Namun, *callback* asinkron memberi pengembang dua tantangan.

Pertama, mekanisme penanganan kesalahan *try*/*catch* JavaScript tidak cukup untuk penanganan kesalahan yang tepat dalam konteks asinkron. Oleh karena itu, komunitas JavaScript mengandalkan protokol *error-first*, sebuah idiom pemrograman informal yang tidak diterapkan atau diperiksa oleh *runtime*. Kedua, *callback* JavaScript sering kali bersifat *nested*, sehingga sulit untuk ditangani (*callback hell*). Namun, ekstensi bahasa terbaru yang disebut *promise* memberikan alternatif untuk *callback* asinkron. Namun penerapan *promise* berjalan lambat karena *promise* memfaktorkan ulang kode yang ada (Gallaba et al., 2017).

JavaScript merupakan bahasa yang dinamis. Bahasa pemrograman yang dinamis menggambarkan kelas bahasa pemrograman yang memiliki sejumlah karakteristik *runtime* umum yang tersedia dalam bahasa statis hanya selama kompilasi apabila ada. Meskipun perilaku ini dapat diserupai pada hampir semua bahasa dengan kompleksitas yang memadai, perilaku tersebut merupakan karakteristik yang tidak terpisahkan dan tertanam dalam bahasa yang dinamis. Bahasa yang dinamis memiliki satu atau lebih ciri-ciri sebagai berikut.

a. **Pengetikan dinamis**. Dalam bahasa statis seperti C, C++ dan Java, tipe variabel dan parameter harus ditetapkan secara eksplisit sebelum kompilasi. Namun, pada sebagian besar bahasa dinamis, variabel tidak perlu dideklarasikan sebelum digunakan. Selain itu, tipenya tidak ditentukan hingga waktu proses ketika informasi tipe benar-benar diperlukan.

b. **Interpretasi**. Dengan bahasa statis, kode dikompilasi menjadi representasi biner atau bentuk perantara lainnya sebelum dieksekusi. Dengan bahasa dinamis, kode sumber dibaca saat *runtime*, diterjemahkan ke dalam representasi perantara atau kode mesin secara dinamis dan kemudian dieksekusi. Dari sudut pandang pengguna akhir, semua fase ini terjadi secara otomatis.
c. **Modifikasi waktu proses**. Saat menggunakan bahasa pemrograman statis, struktur kode tidak dapat diubah pada saat *runtime* terlepas dari beberapa ekstensibilitas terbatas. Namun, dengan bahasa yang dinamis, hierarki kelas dan aspek struktural dan perilaku lainnya dari program dapat dimodifikasi pada saat *runtime*. Misalnya, fungsi dan variabel baru dapat ditambahkan ke kelas dan objek dengan cepat (Mikkonen & Taivalsaari, 2007).

Saat ini, JavaScript dapat digunakan untuk pengembangan sisi klien dan sisi server. JavaScript sisi klien mengacu pada cara kerja JavaScript pada *browser*. Semua *browser web* utama dilengkapi dengan mesin JavaScript bawaan dari *browser* tersebut. Pengembang aplikasi *web* menulis kode JavaScript dengan fungsi berbeda yang terkait dengan berbagai *event*, seperti klik *mouse* atau kursor *mouse*. Fungsi-fungsi tersebut kemudian akan membuat perubahan pada HTML dan CSS.

Sedangkan, JavaScript pada sisi server mengacu pada penggunaan bahasa pengkodean dalam logika server *back-end* dimana mesin JavaScript berada secara langsung di server. Fungsi sisi server JavaScript memungkinkan akses ke basis data, mengeksekusi operasi logika yang beragam, dan merespons terhadap *event-event* yang dipicu oleh sistem operasi server. Keuntungan utama dari skrip sisi server adalah pengembang dapat mengkustomisasi respons situs *web* berdasarkan kebutuhan, hak akses, dan permintaan informasi dari situs *web*.

Penggunaan JavaScript memiliki sejumlah manfaat, seperti:

a. ***Overhead* yang lebih sedikit dibandingkan bahasa lain**. JavaScript adalah bahasa ringan yang tidak memerlukan instalasi perangkat lunak atau infrastruktur perangkat keras yang berat untuk dijalankan.
b. **JavaScript adalah bahasa yang cepat, dan dapat digunakan untuk membangun aplikasi berkinerja tinggi**. Oleh karena itu, JavaScript dapat menangani perhitungan dan operasi yang rumit. Kemampuannya untuk mengeksekusi kode di sisi klien juga membuatnya lebih cepat dibandingkan bahasa sisi server, sehingga mengurangi waktu yang diperlukan untuk transfer data.
c. **JavaScript adalah salah satu bahasa pemrograman paling populer di dunia**, dengan jutaan pengembang menggunakannya untuk membangun aplikasi *web* dan aplikasi seluler. Popularitasnya telah menghasilkan ekosistem perpustakaan, kerangka kerja, dan alat yang luas yang memudahkan pengembang untuk membuat aplikasi yang kuat dengan cepat dan efisien.
d. **JavaScript dapat digunakan untuk tujuan pengembangan *front-end* dan *back-end*.** Untuk pengembangan *back-end*, Node.js sering digunakan, sedangkan untuk pengembangan *front-end*, berbagai perpustakaan seperti AngularJS dan ReactJS dapat digunakan. Dengan memanfaatkan Express.js untuk Node.js, mengimplementasikan database dokumen seperti MongoDB, dan menggunakan JavaScript untuk *front end*, dimungkinkan untuk mengembangkan keseluruhan aplikasi *web* dari depan ke belakang hanya menggunakan JavaScript (Neville, 2023).

2.2.8 Node.js

Node.js merupakan sebuah Lingkungan *Runtime* JavaScript yang memiliki kinerja cepat dan dapat diandalkan untuk jumlah data yang besar dan aplikasi dengan beban sistem yang berat karena pendekatan *event driven*, *non-blocking*, dan *asynchronous*-nya. Fitur utama dari Node.js adalah penggunaannya terhadap I/O yang berbasis *event-driven* dan *non-blocking* dengan pendekatan *asynchronous* untuk tetap ringan dan efisien dalam menangani permintaan bersamaan. Dengan menggunakan Node.js, kompleksitas aplikasi *real-time* dapat dibangun yang dapat berskala hingga jutaan koneksi klien (Jadhav & Gonsalves, 2020).

Node.js saat ini mengalami tren peningkatan karena banyak digunakan oleh pengembang *web*, terutama karena keserbagunaannya dalam menangani permintaan asinkron dan mampu melayani lebih banyak klien dibandingkan kerangka kerja lainnya. Terlebih lagi, npm (*Node Package Manager*), manajer paket default untuk lingkungan *runtime* JavaScript Node.js, sejauh ini merupakan manajer paket terbesar, dan sudah ada lebih dari 1 juta paket.

Node.js dapat menyatukan pengembangan *stack* yang memungkinkan *developer* perangkat lunak untuk bekerja baik di sisi *client* suatu aplikasi serta di sisi server menggunakan bahasa pemrograman yang sama, JavaScript. Node.js memungkinkan pengembangan aplikasi *web* secara bersamaan berdasarkan JavaScript sisi server yang menggunakan I/O asinkron dengan model pemrograman berbasis *event-driven*. Dalam ekosistem Node.js, modul adalah *file* JavaScript yang berisi fungsi tertentu. Modul terbagi menjadi modul bawaan dan pihak ketiga yang dapat di-*install* dalam bentuk paket dari repositori publik menggunakan manajer paket npm (Ntantogian et al., 2021) .

Pada saat tulisan ini ditulis, versi stabil terbaru dari Node.js adalah 20.10.0. Terdapat banyak kerangka kerja yang dibangun di atas Node.js untuk memperluas fungsionalitas dan mendorong kemampuan bahasa tersebut lebih jauh untuk menghemat waktu dan sumber daya. Misalnya, Express.js, Koa.js, Hapi.js, Meteor.js, Next.js, dan Nuxt.js.

2.2.9 Express.js

Express adalah kerangka *web* yang populer dan kuat untuk Node.js. Kerangka ini dirancang untuk menyederhanakan proses pembuatan aplikasi *web* dan API. Express mengikuti pendekatan minimalis, menyediakan serangkaian fitur yang kuat sekaligus menjaga kerangka inti tetap ringan dan fleksibel. Salah satu kekuatan utama Express terletak pada kesederhanaan dan kemudahan penggunaannya. Express menawarkan API yang lugas dan intuitif yang memungkinkan pengembang mengatur rute dengan cepat, menangani permintaan dan respons, serta mengelola fungsi *middleware*. Kesederhanaan ini menjadikan Express pilihan yang sangat baik bagi pemula dan pengembang berpengalaman. Express menggunakan pendekatan modular, yang memungkinkan pengembang memilih dan mengintegrasikan *middleware* tambahan sesuai kebutuhan. Fleksibilitas ini memudahkan pengembang untuk menyesuaikan dan memperluas fungsionalitas aplikasi dengan mudah. Dengan banyaknya koleksi *middleware* yang dikembangkan komunitas, Express menyediakan ekosistem alat dan ekstensi yang kaya untuk meningkatkan pengembangan aplikasi. Express menganut sifat asinkron Node.js yang memungkinkan pengembang menangani permintaan secara efisien tanpa memblokir *event loop*. Arsitektur *non-blocking* ini memastikan skalabilitas dan daya tanggap yang tinggi, menjadikan Express sangat cocok untuk aplikasi yang memerlukan pembaruan *real-time* atau menangani permintaan bersamaan dalam jumlah besar (Nguyen, 2023).

Manfaat dari penggunaan Express antara lain:

1. **Skalabilitas yang cepat.** Dengan menggunakan Express, aplikasi dapat diskalakan dengan cepat. Dengan dukungan Node.js dan penambahan sumber daya tambahan, aplikasi dapat diskalakan dengan berbagai cara.
2. **Didukung oleh mesin Google v8**. Express didukung oleh mesin Google V8 yang dapat membuat kinerja lebih tinggi tanpa kesalahan dalam proses pengembangan.
3. **Mendukung *Caching***. Express mendukung fitur *caching* sehingga tidak perlu mengeksekusi ulang kode berulang kali. Selain itu, *caching* akan membantu halaman *web* memuat lebih cepat dari sebelumnya (CM, 2019).

2.2.10 MongoDB

MongoDB adalah sebuah basis data NoSQL sumber terbuka yang dikembangkan dalam bahasa pemrograman C++. MongoDB menggunakan model penyimpanan dokumen dengan koleksi yang mengelompokkan dokumen berdasarkan struktur. Dokumen di MongoDB dapat berupa berbagai tipe data dasar, seperti tanggal, array, angka, string, atau sub-dokumen. Format penyimpanan menggunakan BSON (Binary JSON) dengan batasan ukuran dokumen 16MB. Untuk meningkatkan kinerja saat bekerja dengan dokumen, MongoDB menggunakan pengindeksan yang mirip dengan basis data relasional. Setiap dokumen diidentifikasi oleh bidang _id dan di atas bidang tersebut dibuat indeks unik. Selain indeks otomatis yang dibuat pada bidang _id, indeks tambahan dapat dibuat oleh administrator basis data (Abramova & Bernardino, 2013).

Berkat model dokumen yang digunakan dalam MongoDB, informasi dapat disematkan dalam satu dokumen tunggal tanpa perlu bergantung pada operasi join yang mahal seperti pada basis data relasional tradisional. Hal ini membuat kueri jauh lebih cepat dan mengembalikan semua informasi yang diperlukan dalam satu kali panggilan ke basis data. Jika diperlukan, left outer join bisa dilakukan dengan tahap agregasi $*lookup*, yang menawarkan kinerja serupa dengan RDBMS. Dalam hal kinerja penulisan, MongoDB menawarkan fungsionalitas untuk menyisipkan dan memperbarui beberapa catatan sekaligus dengan *insertMany*() dan *updateMany*(). Kedua fungsi ini memberikan peningkatan kinerja yang signifikan dibandingkan dengan penulisan batch pada basis data tradisional(MongoDB, 2024).

Sebagai contoh, detail transaksi dalam sistem POS tobaku halal dapat mencakup informasi seperti idTransaksi, tanggal dan waktu transaksi, daftar produk yang dibeli, dan harga masing-masing produk. Dengan MongoDB, semua informasi ini dapat disimpan dalam satu dokumen tunggal, sehingga ketika ada kueri untuk mengambil data transaksi, semua detail tersebut dapat diperoleh dengan cepat dalam satu panggilan ke basis data. Jika ada kebutuhan untuk memperbarui banyak transaksi sekaligus, fungsi *updateMany*() dapat digunakan untuk meningkatkan efisiensi proses tersebut.

MongoDB memiliki beberapa mesin penyimpanan yang unik dalam satu implementasi. Hal tersebut berguna untuk memindahkan data antar teknologi penyimpanan. Hal ini dilakukan dengan menggunakan replikasi lokal. MongoDB mencapai kontrol konkurensi dan kompresi alami untuk efisiensi penyimpanan dan kinerja terbaik melalui mesin penyimpanan bawaan, yaitu WiredTiger. Kelebihan MongoDB terletak pada kemampuannya untuk menggabungkan

mesin *in-memory* untuk operasi dengan latensi sangat rendah dan mesin berbasis *disk* untuk keberlanjutan eksistensial (Jose & Abraham, 2017).

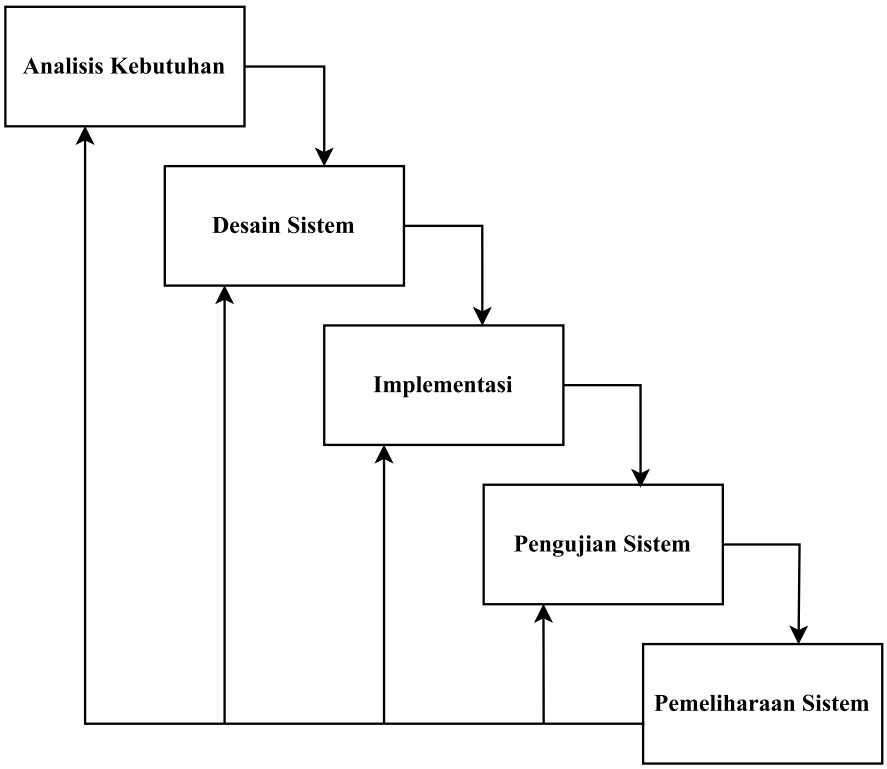
Kemampuan *auto-sharding* MongoDB memungkinkan penambahan node server replika untuk meningkatkan kinerja sistem. Kecepatan database ini ditingkatkan dengan adanya indeks pada atribut utama dan atribut sekunder, bahkan dalam sub-dokumen. Perbandingan berbagai koleksi dilakukan dengan menggunakan teknologi seperti *aggregation framework*, *MapReduce*, dan sistem *Hadoop*. Secara keseluruhan, MongoDB menyediakan solusi basis data yang cepat dan fleksibel dengan fitur-fitur canggih untuk manajemen data yang efisien (Jose & Abraham, 2017).

Database yang digunakan untuk menyimpan data POS adalah MongoDB. Model dokumen MongoDB memungkinkan hampir semua struktur data untuk dimodelkan dan dimanipulasi dengan mudah. Format data BSON MongoDB, yang terinspirasi dari JSON, memungkinkan data POS memiliki objek dalam satu koleksi dengan set bidang yang berbeda-beda (MongoDB, 2024a). Sebagai contoh, saat ini, modul POS di ITS Mart hanya menangani varian minimum pembelian. Di kemudian hari, modul ini dapat dikembangkan untuk menangani varian lainnya. Misalnya, penanganan produk dengan harga *bundling* produk. MongoDB mendukung pembuatan skema eksplisit dan validasi data. Fleksibilitas ini sangat bermanfaat saat menangani data yang sering mengalami perubahan dalam persyaratan.

MongoDB mendukung operasi kueri pada data geospasial, yang memudahkan manajemen data geospasial(MongoDB, 2024b). Fitur ini akan sangat berguna untuk sistem POS tobaku halal apabila dikembangkan untuk fitur pengiriman barang atau pelacakan lokasi. Dengan kemampuan ini, modul POS dapat mengelola dan memproses data lokasi secara efisien(MongoDB, 2024).

BAB III
METODOLOGI

Bab ini membahas metode yang digunakan dalam tugas akhir, yaitu metode *waterfall*. Metode tersebut terdiri dari 5 tahap yang dimulai dengan analisis kebutuhan aplikasi untuk memahami kebutuhan pengguna, diikuti dengan desain sistem untuk merencanakan implementasi aplikasi. Tahap selanjutnya adalah implementasi, di mana aplikasi dibangun sesuai dengan desain yang telah dibuat sebelumnya, serta pengujian untuk memastikan semua fitur berfungsi dengan baik sesuai dengan analisis sebelumnya. Tahap terakhir yaitu *maintenance* yang perlu dilakukan untuk memastikan keberlanjutan dan kualitas aplikasi. Semua tahapan pengerjaan tugas akhir dengan metodologi *waterfall* dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Pengerjaan Tugas Akhir

Gambar 3.1 menggambarkan bagaimana proses pengerjaan tugas akhir perancangan *Backend* modul *point of sale* Teratai. Penggunaan metode *waterfall* pada perancangan tugas akhir diperlukan perancangan yang matang pada setiap tahapan, karena metode ini merupakan metode yang berurutan. Tahapan-tahapan pada metode tersebut akan dijelaskan lebih lengkap pada subbab berikutnya.

3.1 Analisis Kebutuhan

Untuk memperoleh informasi yang diperlukan dalam mengembangkan *backend* modul POS Tobaku halal, dilakukan analisis dan pencatatan hal-hal yang akan mendukung proses

pengembangan modul tersebut. Pada tahap ini, digunakan metode wawancara untuk mendapatkan informasi terkait proses bisnis *point of sale* pada ITS Mart secara konvensional serta referensi dari aplikasi Kasir Pintar. Setelah wawancara selesai, hasil wawancara tersebut didokumentasikan. Transkrip wawancara kemudian disusun dalam bentuk tabel yang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Dokumentasi Wawancara Kebutuhan

ID	Pertanyaan	Jawaban
Q1	Bagaimana alur kerja konvensional proses penjualan bahan baku halal di ITS Mart saat ini?	Proses penjualan bahan baku halal di ITS Mart saat ini dimulai dengan penerimaan stok bahan baku dari pemasok. Stok tersebut dimasukkan ke dalam *freezer*. Transaksi penjualan dilakukan melalui sistem POS (*Point of Sale*) yang ada, di mana staf memasukkan data penjualan dan mencetak setruk pembayaran. Setelah pembayaran selesai, barang diserahkan kepada pembeli, dan stok di sistem POS diperbarui. Namun, karena bahan baku halal masih baru di ITS, kami masih belum tahu bagaimana pengelolaan barang yang seharusnya dikembalikan ataupun kapan barang busuk. Oleh karena itu, kami ingin menerapkan sistem *preorder* untuk mengurangi risiko kerugian.
Q2	Apa saja fitur utama yang dibutuhkan dalam modul POS?	Fitur utama yang diperlukan kurang lebih seperti pada aplikasi Kasir Pintar. Karena, sebelumnya kami sudah menggunakan Kasir Pintar. Fitur utama yang dibutuhkan antara lain fitur manajemen produk dan fitur pemrosesan transaksi penjualan.
Q3	Bagaimana proses pencatatan dan pengelolaan stok barang dilakukan saat ini?	Pada Kasir Pintar bisa dilakukan penambahan stok barang. Pengurangan stok juga otomatis ketika terdapat transaksi. Namun, belum ada pencatatan untuk produk retur, busuk, atau pun kemungkinan lain yang mungkin terjadi.
Q4	Apakah ada kendala yang sering dihadapi dalam penggunaan sistem POS yang sekarang?	Beberapa fitur belum tersedia di Kasir Pintar. Seperti fitur *preorder* dan riwayat perubahan detail produk.
Q5	Apa saja metode pembayaran yang biasa digunakan pembeli di ITS Mart?	Metode pembayaran di ITS Mart bisa menggunakan QRIS ataupun *Cash*. Karena, ITS Mart bisa melayani pembeli dari eksternal ITS juga.
Q6	Bagaimana cara mengetahui penjualan harian, mingguan, atau bulanan?	Pada Kasir Pintar terdapat laporan penjualan dan bisa diunduh dalam bentuk *excel*.

ID	Pertanyaan	Jawaban
Q7	Apakah ITS Mart memerlukan fitur khusus untuk diskon ataupun pajak?	Karena produk daging tidak ada pajak, maka tidak diperlukan fitur manajemen pajak. Namun, untuk diskon masih memungkinkan digunakan.
Q8	Bagaimana ITS Mart mengelola pengembalian barang saat ini?	Karena Tobaku halal ITS Mart baru saja dibuka, ITS Mart belum dapat mengelola pengembalian barang.
Q9	Apa saja laporan yang dibutuhkan dari sistem POS ?	Laporan bisa disesuaikan dengan Kasir Pintar, seperti laporan penjualan, transaksi, maupun laporan keuntungan ataupun kerugian.
Q10	Apakah daging yang dijual di ITS Mart tersedia dalam bentuk kemasan atau dijual berdasarkan berat?	Saat ini, produk daging yang dijual dalam bentuk kemasan semua.

Tabel 3.1 adalah tabel yang berisi transkrip wawancara. Tabel ini mencakup ID pertanyaan wawancara, pertanyaan, dan jawaban dari narasumber yang digunakan untuk mengumpulkan kebutuhan dalam pengembangan *backend* modul POS Tobaku halal. Sebagai contoh, pada ID pertanyaan Q5 terdapat pertanyaan mengenai metode pembayaran yang biasa digunakan di ITS Mart. Jawaban yang diperoleh menunjukkan bahwa metode pembayaran di ITS Mart dapat menggunakan QRIS atau tunai.

Setelah proses transkrip wawancara selesai, hasilnya yang tercatat dalam Tabel 3.1 dianalisis untuk mengidentifikasi daftar kebutuhan yang terkait dengan penelitian ini. Analisis dilakukan dengan memeriksa pernyataan dan tanggapan yang ada dalam transkrip wawancara secara cermat, dengan tujuan untuk mengungkap kebutuhan pengguna secara mendalam. Proses ini melibatkan pencarian kata kunci yang terkait dengan modul POS. Hasil dari analisis ini kemudian disusun menjadi daftar kebutuhan pengguna yang sistematis.

3.1.1 Kebutuhan Pengguna

Kebutuhan pengguna akan menjadi dasar dalam pengembangan *backend* modul POS Tobaku halal. Kebutuhan tersebut akan memastikan bahwa fitur dan fungsionalitas yang dikembangkan sesuai dengan apa yang dibutuhkan oleh pengguna. Hasil penentuan kebutuhan melalui analisis transkrip wawancara ditampilkan dalam Tabel 3.2. Tabel ini mencakup ID Pertanyaan, kalimat yang dianalisis, dan kebutuhan relevan yang diperoleh dari proses analisis tersebut.

Tabel 3.2 Hasil Analisis Transkrip Wawancara

ID Pertanyaan	Kalimat	Kebutuhan Pengguna
Q1	“Namun, karena bahan baku halal masih baru di ITS, kami masih belum tahu bagaimana pengelolaan barang yang seharusnya dikembalikan ataupun kapan barang busuk. Oleh karena itu, kami ingin menerapkan **sistem *preorder*** untuk	Mendukung implementasi sistem *preorder*.

ID Pertanyaan	Kalimat	Kebutuhan Pengguna
	mengurangi risiko kerugian.”	
Q2	“Fitur utama yang dibutuhkan antara lain fitur **manajemen produk** dan fitur pemrosesan transaksi.	Mengelola katalog produk.
Q2	“Fitur utama yang dibutuhkan antara lain fitur manajemen produk dan **fitur pemrosesan transaksi penjualan.”**	Mampu mengelola dan memproses transaksi penjualan.
Q3	“Namun, belum ada **pencatatan** untuk produk retur, busuk, atau pun kemungkinan lain yang mungkin terjadi.”	Mencatat dan mengelola riwayat perubahan detail produk.
Q5	“Metode pembayaran di ITS Mart bisa menggunakan **QRIS** ataupun *Cash*.”	Terintegrasi dengan API pembayaran secara non-tunai.
Q6	“Pada Kasir Pintar terdapat **laporan** penjualan dan bisa diunduh dalam bentuk ***excel***.”	Mengunduh laporan dalam bentuk *excel*.
Q7	“Namun, untuk **diskon** masih memungkinkan digunakan.”	Mengelola diskon termasuk menambah, mengedit, ataupun menghapus diskon.
Q9	“Laporan bisa disesuaikan dengan Kasir Pintar, seperti **laporan penjualan, transaksi, maupun laporan keuntungan ataupun kerugian**.”	Menyediakan laporan-laporan seperti laporan penjualan, transaksi, maupun laporan keuntungan/kerugian.
Q10	“Saat ini, produk daging yang dijual dalam bentuk kemasan semua.”	Menyediakan data jumlah stok daging kemasan yang tersedia.

Tabel 3.2 merupakan hasil analisis transkrip wawancara berupa daftar kebutuhan pengguna untuk pengembangan *backend* modul POS Tobaku halal. Kebutuhan pengguna didapat dengan melakukan pencarian kata kunci pada hasil wawancara untuk mendapatkan informasi dari kalimat hasil wawancara.

Selain menganalisis transkrip wawancara, observasi terhadap aplikasi Kasir Pintar juga dianggap penting. Aplikasi ini dipilih karena ITS Mart sudah terbiasa menggunakan aplikasi tersebut sebagai aplikasi *point of sale*. Hal tersebut bertujuan untuk memudahkan pengguna untuk lebih cepat dalam beradaptasi dengan sistem yang dikembangkan. Dari observasi yang dilakukan, beberapa kebutuhan diidentifikasi dengan memperhatikan kelemahan dari aplikasi yang sudah ada. Dengan mengenali kekurangan tersebut, ditemukan daftar kebutuhan yang bertujuan untuk memberikan solusi.

Hasil penyesuaian dan pembaharuan yang diperoleh dari referensi fungsionalitas aplikasi Kasir Pintar digunakan sebagai dasar untuk menambahkan kebutuhan pada POS tobaku halal. Berikut adalah beberapa kebutuhan tambahan untuk pengembangan modul *backend* POS Tobaku halal.

1. Mengelola informasi karyawan ITS Mart.
2. Mengelola informasi pembeli.
3. Mengelola informasi pemasok.

4. Mengelola variasi harga.

Berdasarkan pengumpulan kebutuhan di atas, dapat diringkas kebutuhan pengguna seperti pada Tabel 3.3.

Tabel 3.3 Daftar Kebutuhan Pengguna

ID	Kebutuhan Pengguna
KP-01	Mendukung implementasi sistem *preorder*.
KP-02	Mengelola katalog produk.
KP-03	Mampu mengelola dan memproses transaksi penjualan.
KP-04	Mencatat dan mengelola riwayat perubahan detail produk.
KP-05	Terintegrasi dengan API pembayaran secara non-tunai.
KP-06	Mengunduh laporan dalam bentuk *excel*.
KP-07	Mengelola diskon termasuk menambah, mengedit, ataupun menghapus diskon.
KP-08	Menyediakan laporan-laporan seperti laporan penjualan, transaksi, maupun laporan keuntungan/kerugian.
KP-09	Mengelola variasi harga produk.
KP-10	Mengelola informasi karyawan ITS Mart.
KP-11	Mengelola informasi pembeli.
KP-12	Mengelola informasi pemasok.

Dari daftar kebutuhan pengguna yang didapatkan seperti pada Tabel 3.3, selanjutnya dapat dijadikan acuan dalam membuat kebutuhan sistem dengan memahami aktor yang terlibat dan membuat *user story* dalam penggunaan POS Tobaku halal.

3.1.2 User Story

Sebagai upaya untuk lebih memahami kebutuhan sistem POS Tobaku halal, maka dibuat *user story*. Pengguna dalam sistem ini adalah admin dan karyawan dari Tobaku halal, pembeli, dan masyarakat umum. Sebagai contoh, untuk memenuhi kebutuhan awal KP-01, pembeli harus mengirimkan detail *preorder* terlebih dahulu. Proses *preorder* juga akan melibatkan kebutuhan integrasi dengan sistem pembayaran, pembuatan transaksi, dan penyelesaian transaksi. Hasil tinjauan ini kemudian disusun menjadi *user story* yang dijelaskan secara rinci dalam deskripsi *user story* di Tabel 3.4. Tabel tersebut memuat daftar kebutuhan, aktor yang terlibat, dan *user story*.

Tabel 3.4 User Story

ID	Kebutuhan	Aktor	*User Story*
US-01	Melakukan *login*	Pembeli	Sebagai seorang pembeli, saya ingin bisa melakukan *login* ke sistem agar saya dapat mengakses dan mengelola akun saya serta melakukan pemesanan.
US-02	Mengirimkan detail *preorder*	Pembeli	Sebagai seorang pembeli, saya ingin mengirimkan detail *preorder* agar saya dapat memesan produk yang diinginkan sebelum stok tersedia.
US-03	Melakukan pembayaran secara non-tunai	Pembeli	Sebagai seorang pembeli, saya ingin dapat melakukan pembayaran secara non-tunai agar transaksi saya lebih mudah dan aman.

ID	Kebutuhan	Aktor	*User Story*
US-04	Melihat status dan detail order	Pembeli	Sebagai seorang pembeli, saya ingin bisa melihat status dan detail order saya agar saya mengetahui perkembangan pesanan saya.
US-05	Melihat daftar *preorder*	Karyawan	Sebagai seorang karyawan, saya ingin melihat daftar *preorder* agar saya dapat memantau dan mengelola pesanan yang masuk.
US-06	Mengubah status *preorder*	Karyawan	Sebagai seorang karyawan, saya ingin bisa mengubah status *preorder* agar saya dapat memperbarui informasi pesanan sesuai dengan perkembangan.
US-07	Menyelesaikan transaksi dengan menambahkan *batch* produk	Karyawan	Sebagai seorang karyawan, saya ingin menyelesaikan transaksi dengan menambahkan *batch* produk sehingga dapat dihubungkan ke TERATAI.
US-08	Menambahkan produk baru untuk dijual dan mengelola produk	Karyawan	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
US-09	Membuat transaksi non-*preorder*	Karyawan	Sebagai seorang karyawan, saya ingin dapat membuat transaksi non-*preorder* dan memberikan detail pembeli agar saya bisa melayani pembeli yang melakukan pembeli langsung.
US-10	Mencatat perubahan produk	Karyawan	Sebagai seorang karyawan, saya ingin mencatat perubahan produk agar informasi produk selalu tercatat dan dapat dilihat.
US-11	Melihat laporan – laporan	Admin	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
US-12	Mengunduh laporan berupa excel	Admin	Sebagai seorang admin, saya ingin mengunduh laporan dalam format *excel* agar saya bisa menganalisis data lebih lanjut.
US-13	Menambahkan dan mengelola diskon	Karyawan	Sebagai seorang karyawan, saya ingin menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
US-14	Menambahkan dan mengelola variasi harga pada produk	Karyawan	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.
US-15	Mengelola informasi karyawan ITS Mart.	Admin	Sebagai seorang admin, saya ingin mengelola karyawan baru ke dalam sistem agar karyawan tersebut bisa mulai bekerja

ID	Kebutuhan	Aktor	*User Story*
			dan mengakses sistem.
US-16	Mengubah status karyawan	Admin	Sebagai seorang admin, saya ingin mengubah status karyawan agar dapat memperbarui informasi kepegawaian sesuai dengan kondisi terkini.
US-17	Mengelola informasi pembeli	Karyawan	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
US-18	Mengelola informasi pemasok	Admin	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
US-19	Membuka kasir	Karyawan	Sebagai seorang karyawan, saya ingin membuka kasir dengan menambahkan uang tunai di kasir sehingga saya dapat mencatat keuangan awal kasir.
US-20	Menutup kasir	Karyawan	Sebagai seorang karyawan, saya ingin menutup kasir di akhir *shift* dengan menghitung uang tunai sehingga saya dapat memastikan akurasi laporan keuangan.
US-21	Melihat riwayat kasir	Admin	Sebagai seorang admin, saya ingin melihat riwayat kasir sehingga dapat memantau kinerja karyawan.

3.1.3 Kebutuhan Sistem

Dari hasil wawancara dengan pengguna, kebutuhan pengguna berhasil diidentifikasi. Selanjutnya, dilakukan pencatatan ulang terhadap kebutuhan fungsional dan non-fungsional, yang disesuaikan dengan *user story*.

3.1.3.1 Kebutuhan Fungsional

Kebutuhan fungsional mendeskripsikan fitur yang harus tersedia pada sistem yang dikembangkan. Kebutuhan fungsional untuk modul POS tobaku halal dijelaskan dalam Tabel 3.5. Tabel ini mencantumkan daftar kebutuhan fungsional yang dihasilkan dari peninjauan ulang terhadap kebutuhan pengguna, termasuk ID kebutuhan fungsional, kebutuhan fungsional itu sendiri, dan prioritasnya.

Tabel 3.5 Daftar Kebutuhan Fungsional

ID	Kebutuhan Fungsional	Prioritas
F-01	Sistem mampu menyediakan laporan laba dan rugi untuk periode tertentu.	Sedang
F-02	Sistem mampu menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Tinggi
F-03	Sistem mampu menyediakan laporan penjualan berdasarkan setiap produk yang dijual.	Tinggi

ID	Kebutuhan Fungsional	Prioritas
F-04	Sistem mampu menyediakan laporan pembeli yang dilakukan oleh perusahaan.	Tinggi
F-05	Sistem mampu menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu.	Tinggi
F-06	Sistem mampu menyediakan laporan *excel* laba dan rugi untuk periode tertentu.	Sedang
F-07	Sistem mampu menyediakan laporan *excel* yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Tinggi
F-08	Sistem mampu menyediakan laporan *excel* penjualan berdasarkan setiap produk yang dijual.	Tinggi
F-09	Sistem mampu menyediakan laporan *excel* pembelian yang dilakukan oleh perusahaan.	Tinggi
F-10	Sistem mampu menyediakan laporan *excel* penjualan yang mencakup total penjualan dalam periode tertentu.	Tinggi
F-11	Sistem mampu menyediakan daftar lengkap semua produk yang tersedia di sistem.	Tinggi
F-12	Sistem mampu memungkinkan pengguna untuk menambah data produk baru ke dalam sistem.	Tinggi
F-13	Sistem mampu memungkinkan pengguna untuk menambah jumlah stok produk yang ada di sistem.	Tinggi
F-14	Sistem mampu memungkinkan pengguna untuk mengedit informasi produk yang sudah ada di sistem.	Tinggi
F-15	Sistem mampu memungkinkan pengguna untuk menghapus produk dari sistem.	Tinggi
F-16	Sistem mampu menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.	Tinggi
F-17	Sistem mampu menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.	Sedang
F-18	Sistem mampu memungkinkan pengguna untuk menambah data varian baru dari suatu produk ke dalam sistem.	Sedang
F-19	Sistem mampu memungkinkan pengguna untuk mengedit informasi varian produk yang sudah ada di sistem.	Sedang
F-20	Sistem mampu memungkinkan pengguna untuk menghapus varian produk dari sistem.	Sedang
F-21	Sistem mampu menyediakan informasi detail mengenai setiap varian produk.	Sedang
F-22	Sistem mampu memungkinkan pengguna untuk menambah data riwayat baru terkait produk ke dalam sistem.	Sedang
F-23	Sistem mampu menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.	Sedang
F-24	Sistem mampu menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.	Sedang
F-25	Sistem mampu menyediakan daftar lengkap semua diskon yang tersedia di sistem.	Rendah
F-26	Sistem mampu memungkinkan pengguna untuk menambah data diskon baru ke dalam sistem.	Rendah
F-27	Sistem mampu menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.	Rendah

ID	Kebutuhan Fungsional	Prioritas
F-28	Sistem mampu memungkinkan pengguna untuk menghapus diskon dari sistem.	Rendah
F-29	Sistem mampu memungkinkan pengguna untuk mengedit informasi diskon yang sudah ada di sistem.	Rendah
F-30	Sistem mampu memungkinkan pengguna untuk menambah data pemasok baru ke dalam sistem.	Sedang
F-31	Sistem mampu menyediakan daftar lengkap semua pemasok yang tersedia di sistem.	Sedang
F-32	Sistem mampu menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.	Sedang
F-33	Sistem mampu memungkinkan pengguna untuk mengedit informasi pemasok yang sudah ada di sistem.	Sedang
F-34	Sistem mampu memungkinkan pengguna untuk menghapus pemasok dari sistem.	Sedang
F-35	Sistem mampu menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.	Tinggi
F-36	Sistem mampu menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.	Tinggi
F-37	Sistem mampu memungkinkan pengguna untuk membuat transaksi baru di dalam sistem.	Tinggi
F-38	Sistem mampu memungkinkan pengguna untuk mengubah status transaksi.	Tinggi
F-39	Sistem mampu memungkinkan pengguna untuk memperbarui *batch* produk dalam transaksi.	Tinggi
F-40	Sistem mampu menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.	Tinggi
F-41	Sistem mampu mendukung pembayaran melalui MidTrans.	Tinggi
F-42	Sistem mampu menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.	Sedang
F-43	Sistem mampu memungkinkan pengguna untuk menambah data pembeli baru ke dalam sistem.	Sedang
F-44	Sistem mampu menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.	Sedang
F-45	Sistem mampu memungkinkan pengguna untuk mengedit informasi pembeli yang sudah ada di sistem.	Sedang
F-46	Sistem mampu memungkinkan pengguna untuk menghapus pembeli dari sistem.	Sedang
F-47	Sistem mampu memungkinkan pengguna untuk membuka kasir dengan mencatat nilai awal keuangan saat memulai *shift* kasir.	Tinggi
F-48	Sistem mampu memungkinkan pengguna untuk menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.	Tinggi
F-49	Sistem mampu menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.	Tinggi
F-50	Sistem mampu menyediakan informasi detail mengenai setiap sesi kasir.	Tinggi
F-51	Sistem mampu menambahkan data *preorder*.	Tinggi

Tabel 3.5 menggambarkan kebutuhan fungsional yang diperoleh dari proses analisis kebutuhan dan dijabarkan secara rinci. Tabel ini memuat ID, penjelasan kebutuhan fungsional,

dan prioritas dari kebutuhan tersebut. Sebagai contoh, kebutuhan fungsional dengan ID F-32 menjelaskan bahwa sistem harus memungkinkan pengguna untuk dapat membuat transaksi baru, dengan prioritas tinggi yang menunjukkan bahwa fitur ini sangat penting untuk sistem POS Tobaku halal.

Setelah mendapatkan daftar kebutuhan fungsional sistem, langkah selanjutnya adalah memetakan kebutuhan fungsional tersebut dengan *user story*. Pemetaan ini bertujuan untuk mengevaluasi kesesuaian dan keterkaitan antara kebutuhan fungsional yang telah dibuat dengan *user story*. Untuk mencapai tujuan ini, digunakan *traceability matrix*. *Traceability matrix* ini ditampilkan dalam bentuk tabel seperti pada Tabel 3.6, dengan kolom utama yang berisi ID dan deskripsi kebutuhan fungsional serta *user story* yang relevan. Matriks ini memberikan gambaran yang jelas mengenai keterkaitan antara kebutuhan fungsional dengan kebutuhan awal yang telah ditetapkan.

Tabel 3.6 Traceability Matrix Kebutuhan Fungsional

ID Fungsionalitas	**Deskripsi Kebutuhan**	**ID Kebutuhan *User Story***	**Deskripsi *User Story***
F-01	Sistem mampu menyediakan laporan laba dan rugi untuk periode tertentu.	US-11	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
F-02	Sistem mampu menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.	US-11	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
F-03	Sistem mampu menyediakan laporan penjualan berdasarkan setiap produk yang dijual.	US-11	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
F-04	Sistem mampu menyediakan laporan pembelian yang dilakukan oleh perusahaan.	US-11	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
F-05	Sistem mampu menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu.	US-11	Sebagai seorang admin, saya ingin melihat laporan-laporan agar saya dapat memantau kinerja penjualan dan operasional perusahaan.
F-06	Sistem mampu menyediakan laporan excel laba dan rugi untuk periode tertentu.	US-12	Sebagai seorang admin, saya ingin mengunduh laporan dalam format excel agar saya bisa menganalisis data lebih lanjut.
F-07	Sistem mampu menyediakan laporan excel yang mendetail semua transaksi yang terjadi dalam periode tertentu.	US-12	Sebagai seorang admin, saya ingin mengunduh laporan dalam format excel agar saya bisa menganalisis data lebih lanjut.
F-08	Sistem mampu menyediakan laporan excel penjualan berdasarkan setiap produk yang dijual.	US-12	Sebagai seorang admin, saya ingin mengunduh laporan dalam format excel agar saya bisa menganalisis data lebih lanjut.

ID Fungsionalitas	Deskripsi Kebutuhan	ID Kebutuhan *User Story*	Deskripsi *User Story*
F-09	Sistem mampu menyediakan laporan excel pembelian yang dilakukan oleh perusahaan.	US-12	Sebagai seorang admin, saya ingin mengunduh laporan dalam format excel agar saya bisa menganalisis data lebih lanjut.
F-10	Sistem mampu menyediakan laporan excel penjualan yang mencakup total penjualan dalam periode tertentu.	US-12	Sebagai seorang admin, saya ingin mengunduh laporan dalam format excel agar saya bisa menganalisis data lebih lanjut.
F-11	Sistem mampu menyediakan daftar lengkap semua produk yang tersedia di sistem.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-12	Sistem mampu memungkinkan pengguna untuk menambah data produk baru ke dalam sistem.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-13	Sistem mampu memungkinkan pengguna untuk menambah jumlah stok produk yang ada di sistem.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-14	Sistem mampu memungkinkan pengguna untuk mengedit informasi produk yang sudah ada di sistem.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-15	Sistem mampu memungkinkan pengguna untuk menghapus produk dari sistem.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-16	Sistem mampu menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.	US-08	Sebagai seorang karyawan, saya ingin menambahkan produk baru ke dalam sistem dan dapat mengelolanya agar produk tersebut tersedia untuk dijual kepada pembeli.
F-17	Sistem mampu menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.	US-14	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.
F-18	Sistem mampu memungkinkan pengguna untuk menambah data varian baru dari suatu produk ke dalam sistem.	US-14	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.

ID Fungsionalitas	Deskripsi Kebutuhan	ID Kebutuhan *User Story*	Deskripsi *User Story*
F-19	Sistem mampu memungkinkan pengguna untuk mengedit informasi varian produk yang sudah ada di sistem.	US-14	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.
F-20	Sistem mampu memungkinkan pengguna untuk menghapus varian produk dari sistem.	US-14	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.
F-21	Sistem mampu menyediakan informasi detail mengenai setiap varian produk.	US-14	Sebagai seorang karyawan, saya ingin menambahkan dan mengelola variasi harga pada produk agar pembeli memiliki opsi harga sesuai dengan kebutuhan mereka.
F-22	Sistem mampu memungkinkan pengguna untuk menambah data riwayat baru terkait produk ke dalam sistem.	US-10	Sebagai seorang karyawan, saya ingin mencatat perubahan produk agar informasi produk selalu tercatat dan dapat dilihat.
F-23	Sistem mampu menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.	US-10	Sebagai seorang karyawan, saya ingin mencatat perubahan produk agar informasi produk selalu tercatat dan dapat dilihat.
F-24	Sistem mampu menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.	US-10	Sebagai seorang karyawan, saya ingin mencatat perubahan produk agar informasi produk selalu tercatat dan dapat dilihat.
F-25	Sistem mampu menyediakan daftar lengkap semua diskon yang tersedia di sistem.	US-13	Sebagai seorang karyawan, saya ingin menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
F-26	Sistem mampu memungkinkan pengguna untuk menambah data diskon baru ke dalam sistem.	US-13	Sebagai seorang karyawan, saya ingin menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
F-27	Sistem mampu menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.	US-13	Sebagai seorang karyawan, saya ingin menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
F-28	Sistem mampu	US-13	Sebagai seorang karyawan, saya ingin

ID Fungsionalitas	Deskripsi Kebutuhan	ID Kebutuhan *User Story*	Deskripsi *User Story*
	memungkinkan pengguna untuk menghapus diskon dari sistem.		menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
F-29	Sistem mampu memungkinkan pengguna untuk mengedit informasi diskon yang sudah ada di sistem.	US-13	Sebagai seorang karyawan, saya ingin menambahkan diskon dan mengelola diskon pada produk tertentu agar dapat menarik lebih banyak pembeli dan meningkatkan penjualan.
F-30	Sistem mampu memungkinkan pengguna untuk menambah data pemasok baru ke dalam sistem.	US-18	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
F-31	Sistem mampu menyediakan daftar lengkap semua pemasok yang tersedia di sistem.	US-18	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
F-32	Sistem mampu menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.	US-18	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
F-33	Sistem mampu memungkinkan pengguna untuk mengedit informasi pemasok yang sudah ada di sistem.	US-18	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
F-34	Sistem mampu memungkinkan pengguna untuk menghapus pemasok dari sistem.	US-18	Sebagai seorang admin, saya ingin menambah dan mengelola informasi pemasok agar data pemasok lengkap.
F-35	Sistem mampu menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.	US-05	Sebagai seorang karyawan, saya ingin melihat daftar *preorder* agar saya dapat memantau dan mengelola pesanan yang masuk.
F-36	Sistem mampu menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.	US-04	Sebagai seorang pembeli, saya ingin bisa melihat status order saya agar saya mengetahui perkembangan pesanan saya.
F-37	Sistem mampu memungkinkan pengguna untuk membuat transaksi baru di dalam sistem.	US-09	Sebagai seorang karyawan, saya ingin dapat membuat transaksi non-*preorder* dan memberikan detail pembeli agar saya bisa melayani pembeli yang melakukan pembeli langsung.
F-38	Sistem mampu memungkinkan pengguna	US-06	Sebagai seorang karyawan, saya ingin bisa mengubah status *preorder* agar

ID Fungsionalitas	Deskripsi Kebutuhan	ID Kebutuhan *User Story*	Deskripsi *User Story*
	untuk mengubah status transaksi.		saya dapat memperbarui informasi pesanan sesuai dengan perkembangan.
F-39	Sistem mampu memungkinkan pengguna untuk memperbarui *batch* produk dalam transaksi/	US-07	Sebagai seorang karyawan, saya ingin menyelesaikan transaksi dengan menambahkan *batch* produk sehingga dapat dihubungkan ke TERATAI.
F-40	Sistem mampu menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.	US-09	Sebagai seorang karyawan, saya ingin dapat membuat transaksi non-*preorder* dan memberikan detail pembelian agar saya bisa melayani pembeli yang melakukan pembelian langsung.
F-41	Sistem mampu mendukung pembayaran melalui MidTrans.	US-03	Sebagai seorang pembeli, saya ingin dapat melakukan pembayaran secara non-tunai agar transaksi saya lebih mudah dan aman
F-42	Sistem mampu menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.	US-17	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
F-43	Sistem mampu memungkinkan pengguna untuk menambah data pembeli baru ke dalam sistem.	US-17	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
F-44	Sistem mampu menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.	US-17	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
F-45	Sistem mampu memungkinkan pengguna untuk mengedit informasi pembeli yang sudah ada di sistem.	US-17	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
F-46	Sistem mampu memungkinkan pengguna untuk menghapus pembeli dari sistem.	US-17	Sebagai seorang karyawan, saya ingin menambah dan mengelola informasi pembeli agar data pembeli lengkap dan akurat untuk proses penjualan dan pemasaran.
F-47	Sistem mampu memungkinkan pengguna untuk membuka kasir dengan mencatat nilai awal keuangan saat memulai *shift* kasir.	US-19	Sebagai seorang karyawan, saya ingin membuka kasir dengan menambahkan uang tunai di kasir sehingga saya dapat mencatat keuangan awal kasir.
F-48	Sistem mampu	US-20	Sebagai seorang karyawan, saya ingin

ID Fungsionalitas	Deskripsi Kebutuhan	ID Kebutuhan *User Story*	Deskripsi *User Story*
	memungkinkan pengguna untuk menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.		menutup kasir di akhir *shift* dengan menghitung uang tunai sehingga saya dapat memastikan akurasi laporan keuangan.
F-49	Sistem mampu menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.	US-21	Sebagai seorang admin, saya ingin melihat riwayat kasir sehingga dapat memantau kinerja karyawan.
F-50	Sistem mampu menyediakan informasi detail mengenai setiap sesi kasir.	US-21	Sebagai seorang admin, saya ingin melihat riwayat kasir sehingga dapat memantau kinerja karyawan.
F-51	Sistem mampu menambahkan data *preorder*.	US-02	Sebagai seorang pembeli, saya ingin mengirimkan detail *preorder* agar saya dapat memesan produk yang diinginkan sebelum stok tersedia.

Tabel 3.6 menguraikan hubungan antara kebutuhan fungsionalitas dan *user story*. Dengan adanya *matriks traceability* menunjukkan kesesuaian antara kebutuhan fungsional yang dibuat dengan kebutuhan awal. Hal ini membantu mengurangi kesalahan yang terkait dengan pengembangan sistem.

3.1.3.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional diperoleh melalui peninjauan kebutuhan yang telah ditetapkan pada fase sebelumnya. Hasil tinjauan ini kemudian menjadi kebutuhan non-fungsional yang dapat dilihat di dalam Tabel 3.7.

Tabel 3.7 Daftar Kebutuhan Non-Fungsional

ID	Daftar Kebutuhan Non-Fungsional
NF-01	Sistem harus dapat diakses diberbagai perangkat yang terkoneksi internet.
NF-02	Sistem memastikan pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses mereka setelah melakukan login.
NF-03	Sistem harus dapat menangani 15 transaksi per detik dengan waktu respon tidak lebih dari 1 detik.

Tabel 3.7 berisi daftar kebutuhan non-fungsional yang mencakup atribut-atribut terkait dengan performa, ketersediaan, dan keamanan. Contoh dari ID KN02 adalah sistem memastikan bahwa pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses yang telah ditetapkan setelah mereka berhasil melakukan proses *login* ke dalam sistem. Hal ini bertujuan untuk meningkatkan keamanan dan menjaga privasi data dengan memastikan bahwa setiap pengguna hanya memiliki akses yang diperlukan sesuai dengan perannya dalam organisasi atau aplikasi tersebut.

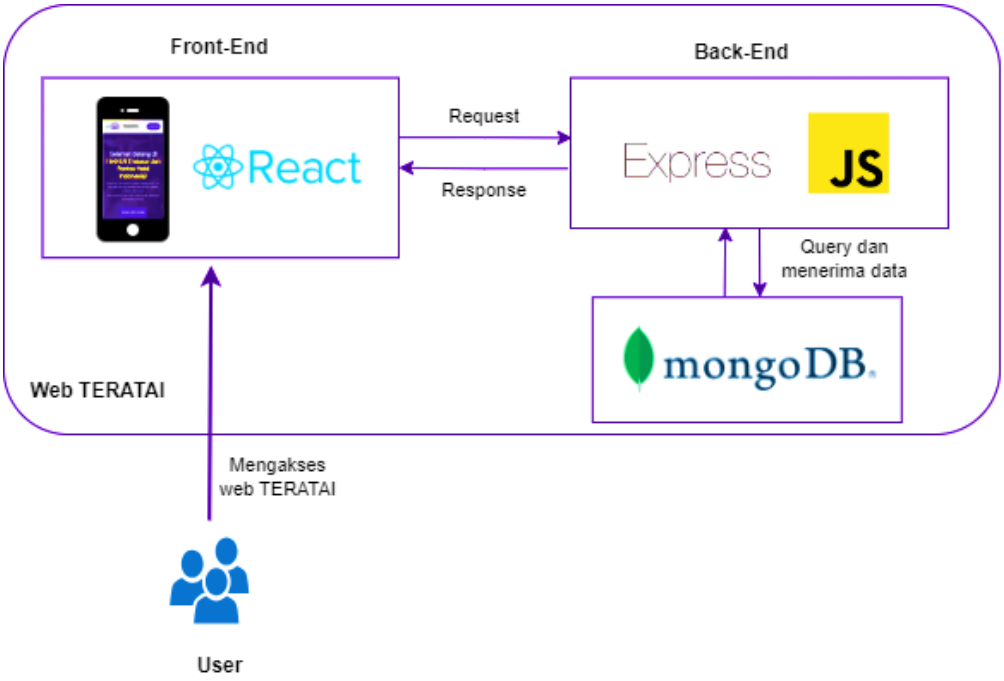
3.2 Desain Sistem

Desain sistem merupakan tahap setelah analisis kebutuhan. Pada tahap ini, dihasilkan

serangkaian artefak yang mendetail untuk memastikan solusi yang efektif dan sesuai dengan kebutuhan bisnis. Artefak tersebut terdiri dari arsitektur sistem, *usecase diagram*, penggunaan, alur proses bisnis, perancangan basis data, dan perancangan API *Endpoint* pada *backend* modul POS Tobaku halal.

3.2.1 Arsitektur Sistem POS

Pada tugas akhir ini, penulis bertanggung jawab dalam mengembangkan bagian *backend* sistem, yang mencakup pengolahan data, penyimpanan informasi, dan logika bisnis dari aplikasi. Sedangkan bagian *frontend*, yang mencakup antarmuka pengguna dan interaksi pengguna dengan sistem, dikerjakan oleh Amanda Salwa.



Gambar 3.2 Arsitektur Sistem POS

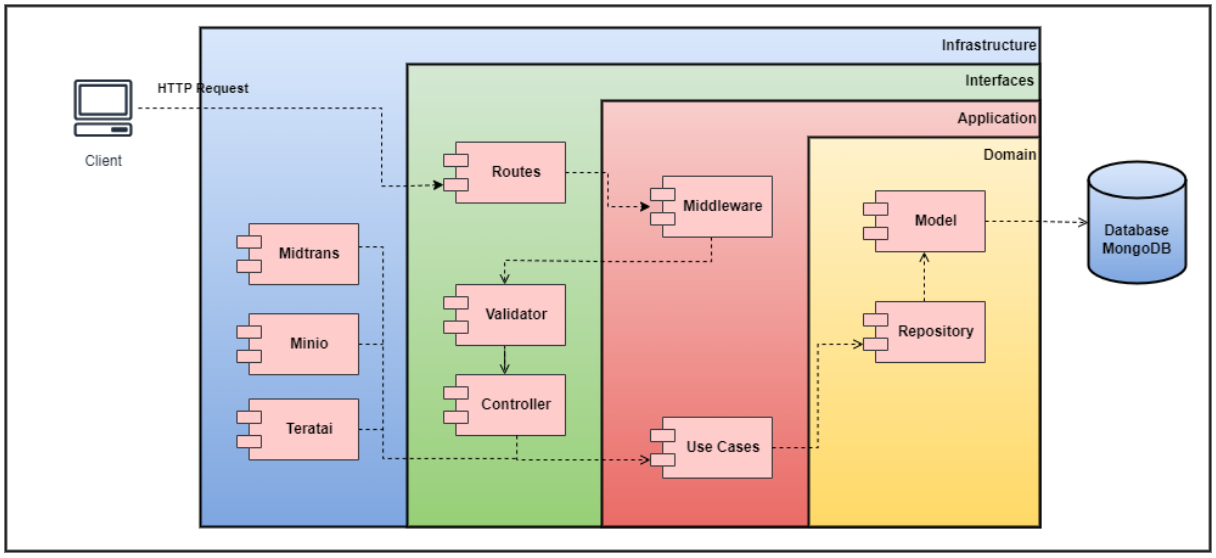
Gambar 3.2 menunjukkan arsitektur sistem web TERATAI yang terdiri dari dua komponen utama, yaitu *frontend* dan *backend*, serta interaksi antara keduanya dan pengguna. Pada bagian *frontend*, digunakan React untuk membangun antarmuka pengguna yang interaktif dan responsif. Pengguna mengakses web TERATAI melalui *browser* dan mengirimkan permintaan (*request*) dari antarmuka pengguna ke server. Bagian *backend* menggunakan Express, sebuah framework untuk Node.js, dan JavaScript untuk mengembangkan logika bisnis serta proses di server. MongoDB digunakan sebagai database NoSQL untuk menyimpan dan mengelola data. *Backend* menerima permintaan dari *frontend*, melakukan *query* ke MongoDB untuk mendapatkan atau menyimpan data, dan mengirimkan respons kembali ke *frontend*. Dengan demikian, alur kerja dimulai dari pengguna yang mengakses web, pengiriman permintaan dari antarmuka pengguna ke server, pemrosesan data di server, dan pengiriman respons kembali ke pengguna.

3.2.2 Arsitektur Sistem

Sistem yang sedang dikembangkan akan menggunakan *clean architecture*. *Clean*

architecture, yang diusulkan oleh Robert C. Martin, adalah sebuah konsep arsitektur perangkat lunak yang bertujuan untuk meningkatkan manajemen, modifikasi, dan pengujian perangkat lunak. *Clean architecture* mengusulkan pemisahan kode menjadi lapisan-lapisan yang terdefinisi dengan baik, dengan setiap lapisan memiliki tanggung jawabnya sendiri. Arsitektur pada *backend* modul *point of sale* toko bahan baku halal yang dirancang dapat dilihat pada Gambar 3.3.

Gambar 3.3 menggambarkan arsitektur *backend point of sale* Tobaku halal yang mengikuti prinsip-prinsip *Clean Architecture* dengan memisahkan tanggung jawab ke dalam berbagai lapisan untuk menjaga kebersihan kode, modularitas, dan keterpisahan antarkomponen. Pada awalnya, terdapat *Client* yang mengirimkan permintaan HTTP ke *server*. Permintaan ini akan diterima oleh lapisan *Interfaces* yang menangani rute (*Routes*) untuk mengarahkan permintaan tersebut ke *Controller* yang sesuai dalam lapisan Aplikasi.



Gambar 3.3 Arsitektur Backend Point of Sale Tobaku halal

Controller dapat memanfaatkan *Middleware* pada lapisan *Application* untuk proses tambahan seperti autentikasi serta *Validator* untuk memastikan data yang diterima valid. Setelah proses di *Middleware* dan *Validator*, *Controller* berinteraksi dengan *Use Cases* yang mengandung logika bisnis utama dari aplikasi. *Use Cases* ini kemudian berkomunikasi dengan *Repository* di lapisan *Domain* untuk melakukan operasi data seperti *Create*, *Read*, *Update*, *Delete* (CRUD). *Repository* bertanggung jawab mengakses *Model* yang merupakan representasi data dalam aplikasi dan berinteraksi dengan *Database* MongoDB. Lapisan Infrastruktur berisi layanan eksternal seperti Midtrans untuk integrasi pembayaran, Minio untuk penyimpanan objek, dan Teratai untuk integrasi *halal traceability.*

Arsitektur ini memastikan bahwa lapisan luar bergantung pada lapisan dalam tetapi tidak sebaliknya. Prinsip ini bertujuan untuk menjaga agar logika bisnis inti tetap terisolasi dari perubahan teknis yang mungkin terjadi pada lapisan luar. Dengan cara ini, setiap perubahan pada teknologi antarmuka pengguna, penyimpanan data, atau layanan eksternal lainnya tidak akan mempengaruhi logika bisnis utama. Isolasi ini memungkinkan pengembangan yang lebih fleksibel dan terpisah, di mana pengembang dapat mengubah atau memperbarui teknologi di lapisan luar tanpa perlu melakukan perubahan signifikan pada inti aplikasi. Dalam jangka

panjang, pendekatan ini mendukung pemeliharaan dan pengembangan berkelanjutan, mengurangi risiko gangguan sistem, dan meningkatkan kemampuan aplikasi untuk beradaptasi dengan kemajuan teknologi atau perubahan kebutuhan bisnis. Dengan demikian, arsitektur ini menciptakan fondasi yang kuat untuk sistem yang dapat terus berkembang dan bertahan dalam jangka waktu yang lama.

3.2.3 Diagram Usecase

Setelah menentukan daftar kebutuhan fungsional pada tahap sebelumnya, langkah berikutnya adalah menuliskan dan menggambarkan diagram *usecase*. Proses ini bertujuan untuk menciptakan diagram *usecase* dengan menggabungkan beberapa kebutuhan fungsional ke dalam satu kasus yang sama. Pengelompokkan kebutuhan fungsional dilakukan berdasarkan keterkaitan dan hubungan antara berbagai fungsionalitas.

Sebagai contoh, fungsionalitas dengan ID F-01 dan F-02 digabungkan ke dalam satu *usecase* yang sama, yaitu melihat laporan. Hal tersebut menggambarkan bagaimana pengguna akan berinteraksi dengan sistem untuk memenuhi tujuan tersebut. Seluruh kebutuhan fungsional dipetakan ke dalam *usecase* yang mencakup semua langkah dan aktivitas yang diperlukan untuk mencapai hasil akhir yang diinginkan. Tabel 3.8 menunjukkan klasifikasi *usecase*.

Tabel 3.8 Klasifikasi Usecase

ID	Klasifikasi *Usecase*	*Usecase*
A	Melihat Laporan	Menyediakan laporan laba dan rugi untuk periode tertentu.
		Menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.
		Menyediakan laporan penjualan berdasarkan setiap produk yang dijual.
		Menyediakan laporan pembelian yang dilakukan oleh perusahaan.
		Menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu.
		Menyediakan laporan *excel* laba dan rugi untuk periode tertentu.
		Menyediakan laporan *excel* yang mendetail semua transaksi yang terjadi dalam periode tertentu.
		Menyediakan laporan *excel* penjualan berdasarkan setiap produk yang dijual.
		Menyediakan laporan *excel* pembelian yang dilakukan oleh perusahaan.
		Menyediakan laporan *excel* penjualan yang mencakup total penjualan dalam periode tertentu.
B	Manajemen Produk	Menyediakan daftar lengkap semua produk yang tersedia di sistem.
		Menambah data produk baru ke dalam sistem.
		Menambah jumlah stok produk yang ada di sistem.
		Mengedit informasi produk yang sudah ada di sistem.
		Menghapus produk dari sistem.
		Menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.
B1	Manajemen Varian Produk	Menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.

ID	Klasifikasi *Usecase*	*Usecase*
		Menambah data varian baru dari suatu produk ke dalam sistem.
		Mengedit informasi varian produk yang sudah ada di sistem.
		Menghapus varian produk dari sistem.
		Menyediakan informasi detail mengenai setiap varian produk.
B2	Manajemen Riwayat Produk	Menambah data riwayat baru terkait produk ke dalam sistem.
		Menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.
		Menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.
		Menyediakan daftar lengkap semua diskon yang tersedia di sistem.
C	Manajemen Diskon	Menambah data diskon baru ke dalam sistem.
		Menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.
		Menghapus diskon dari sistem.
		Mengedit informasi diskon yang sudah ada di sistem.
D	Manajemen Pemasok	Menambah data pemasok baru ke dalam sistem.
		Menyediakan daftar lengkap semua pemasok yang tersedia di sistem.
		Menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.
		Mengedit informasi pemasok yang sudah ada di sistem.
		Menghapus pemasok dari sistem.
E	Manajemen Transaksi	Menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.
		Menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.
		Membuat transaksi baru di dalam sistem *preorder* dan non-*preorder*
		Mengubah status transaksi.
		Memperbarui batch produk dalam transaksi.
		Menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.
E1	Integrasi API Pembayaran	Mendukung pembayaran melalui MidTrans.
F	Manajemen Pembeli	Menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.
		Menambah data pembeli baru ke dalam sistem.
		Menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.
		Mengedit informasi pembeli yang sudah ada di sistem.
		Menghapus pembeli dari sistem.
G	Manajemen Kasir	Membuka kasir dengan mencatat nilai awal keuangan saat memulai *shift* kasir.
		Menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.
		Menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.

ID	Klasifikasi *Usecase*	*Usecase*
		Menyediakan informasi detail mengenai setiap sesi kasir.

Berdasarkan hasil klasifikasi use case yang terdapat pada Tabel 3.8, dapat dibuat diagram yang relevan. Hasil dari perancangan diagram *usecase* dapat dilihat pada Gambar 3.4. Pembuatan diagram ini menggunakan aplikasi Draw.io, yang memfasilitasi visualisasi berbagai komponen dan interaksi dalam aplikasi. Gambar 3.4 ini menunjukkan hasil perancangan diagram *usecase* secara umum, yang menggambarkan hubungan antara aktor (pengguna) dan berbagai fungsi atau fitur yang disediakan oleh aplikasi.

Gambar 3.4 merupakan diagram *usecase* dari aplikasi POS Tobaku halal yang menggambarkan interaksi antara aktor dan sistem. Warna yang berbeda pada *use case* menunjukkan arti yang berbeda. *Use case* berwarna kuning merupakan *use case* utama yang dikerjakan pada tugas akhir, seperti:

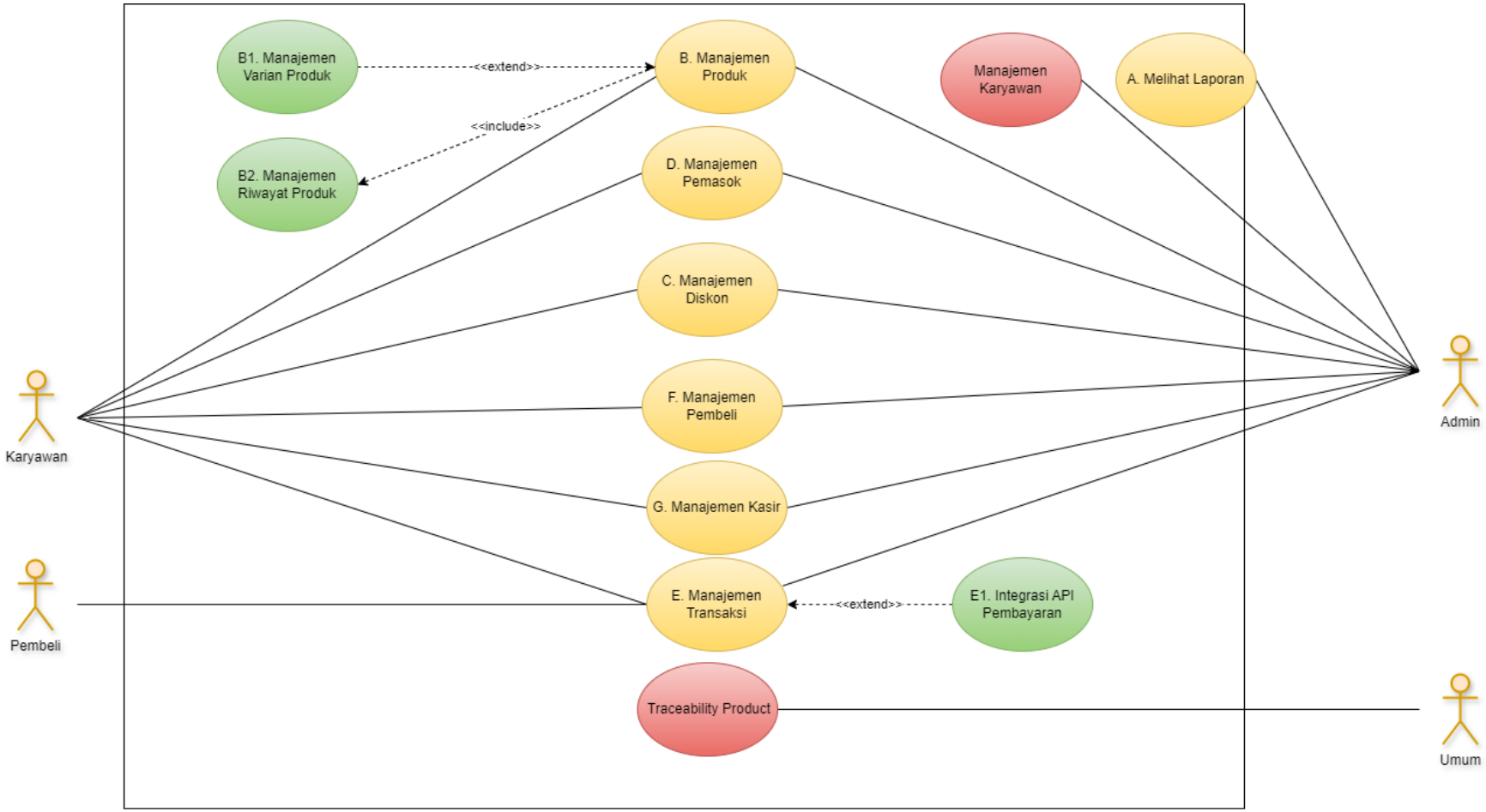
- B. Manajemen Produk
- D. Manajemen Pemasok
- C. Manajemen Diskon
- F. Manajemen Pembeli
- G. Manajemen Kasir
- E. Manajemen Transaksi

Use case berwarna hijau merupakan bagian dari use case utama yang diindikasikan dengan <<*include*>> atau <<*extend*>>. Dalam diagram *use case*, istilah <<include>> dan <<*extend*>> digunakan untuk menggambarkan hubungan antara *use case* yang berbeda. <<*include*>> menunjukkan bahwa suatu *use case* menyertakan perilaku dari use case lain sebagai bagian wajib dari prosesnya. Misalnya, B2. Manajemen Riwayat Produk di-*include* dalam B. Manajemen Produk, sehingga setiap kali B. Manajemen Produk dijalankan, B2. Manajemen Riwayat Produk juga akan dijalankan sebagai bagian dari proses tersebut. Sebaliknya, <<*extend*>> menunjukkan bahwa suatu *use case* mungkin memperluas perilaku dari *use case* lain dalam kondisi tertentu. Misalnya, B1. Manajemen Varian Produk adalah perpanjangan dari B. Manajemen Produk, yang berarti B1. Manajemen Varian Produk hanya dijalankan jika ada kebutuhan khusus untuk mengelola varian produk. Penggunaan <<*include*>> dan <<*extend*>> membantu dalam mengatur dan mengelompokkan fungsi yang terkait dalam sistem, memastikan bahwa perilaku umum dan tambahan dapat dikelola dengan cara yang terstruktur dan efisien.

Sementara itu, use case berwarna merah menunjukkan *use case* yang sudah dikerjakan sebelumnya pada pengembangan sistem TERATAI, yaitu:

- Manajemen Karyawan
- *Traceability Product*

Diagram ini membantu dalam memahami struktur dan interaksi yang terjadi dalam aplikasi POS Tobaku Halal, serta membedakan antara fitur yang dikembangkan dalam tugas akhir dan yang sudah ada sebelumnya.



Gambar 3.4 Diagram Usecase

Tabel 3.9 mendefinisikan *usecase* modul POS Tobaku halal. Tabel tersebut berisi nama kasus penggunaan, aktor yang terlibat, dan deskripsi masing-masing *usecase*.

Tabel 3.9 Deskripsi usecase modul POS Tobaku halal

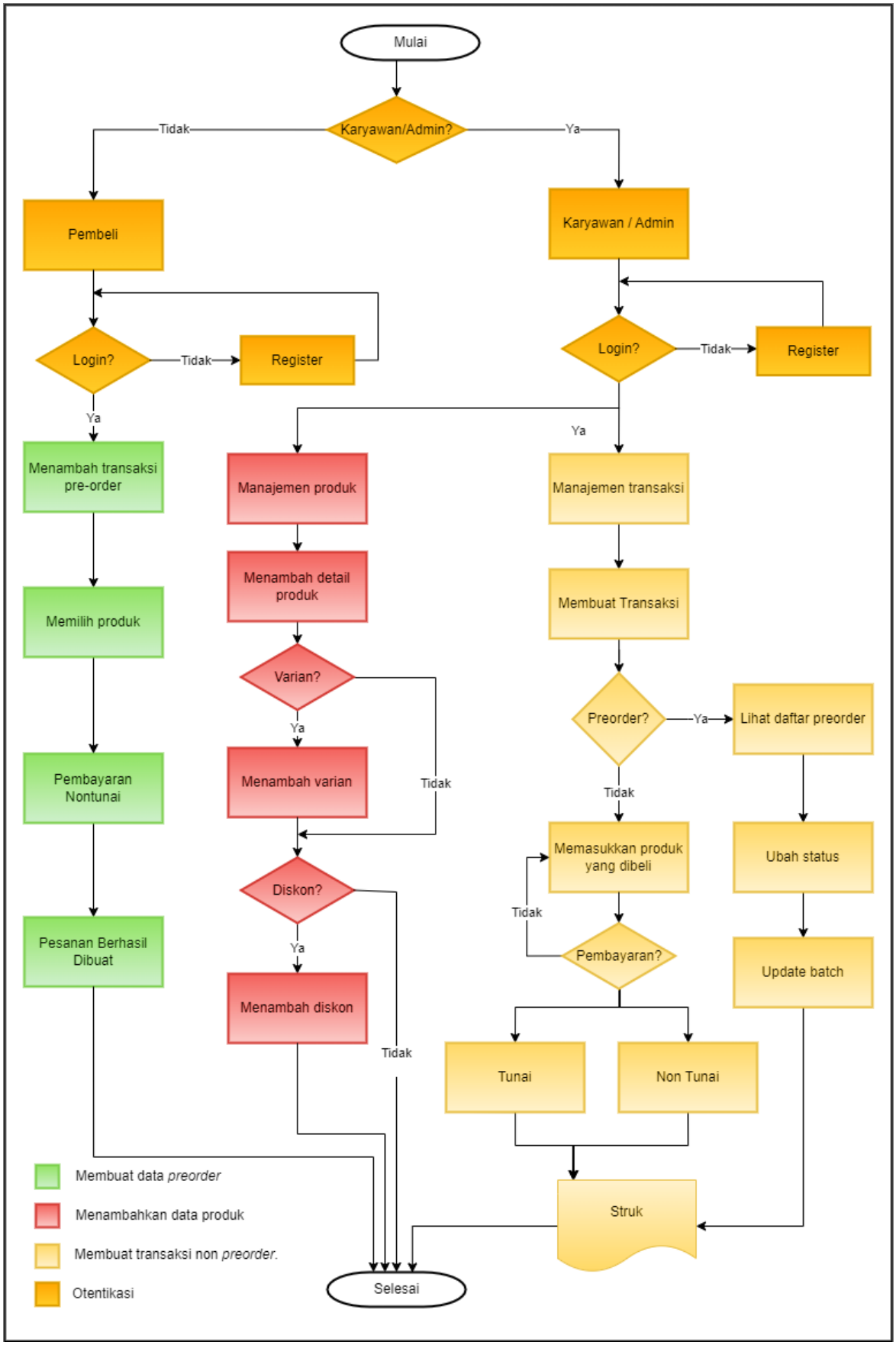
ID	Nama *usecase*	Aktor	Deskripsi
A	Melihat Laporan	Admin	Admin dapat melihat laporan penjualan, stok, dan performa toko secara keseluruhan.
B	Manajemen Produk	Admin, Karyawan	Admin dan karyawan dapat menambah, mengedit, dan menghapus produk yang dijual di toko.
B1	Manajemen Varian Produk	Admin, Karyawan	Admin dan karyawan dapat menambah, mengedit, dan menghapus varian produk seperti nama varian dan harganya.
B2	Manajemen Riwayat Produk	Admin, Karyawan	Admin dan karyawan dapat melihat dan mengelola riwayat perubahan produk yang ada di sistem.
C	Manajemen Diskon	Admin, Karyawan	Admin dan karyawan dapat membuat, mengedit, dan menghapus diskon yang berlaku di toko.
D	Manajemen Pemasok	Admin, Karyawan	Pengguna dapat menambah, mengedit, dan menghapus data pemasok serta melihat riwayat pembelian.
E	Manajemen Transaksi	Admin, Karyawan, Pembeli	Admin dan karyawan dapat membuat transaksi baru, mengubah status tertentu, dan memproses penjualan. Pembeli dapat membuat transaksi *preorder*.
E1	Integrasi API Pembayaran	Admin, Karyawan, Pembeli	Admin, karyawan, dan pembeli dapat memproses pembayaran digital seperti melalui QRIS.
F	Manajemen Pembeli	Admin, Karyawan	Admin dan karyawan dapat menambah, mengedit, dan menghapus data pembeli.
G	Manajemen Kasir	Admin, Karyawan	Karyawan dapat membuka dan menutup kasir. Admin mendapatkan riwayat kasir.

3.2.4 Diagram Alir

Proses bisnis merupakan langkah selanjutnya setelah *usecase* ditetapkan. Proses ini membantu memperjelas dan merinci setiap langkah dari keseluruhan usecase yang telah dirancang. Setiap aktivitas dalam proses bisnis dirancang untuk memastikan bahwa semua langkah yang diperlukan untuk mencapai tujuan sistem terdefinisi dengan baik dan dapat diimplementasikan secara efisien. Proses bisnis utama dari modul POS Tobaku Halal dibagi menjadi tiga bagian, yaitu:

1. membuat data *preorder*,
2. menambahkan data produk,
3. dan membuat transaksi non *preorder*.

Proses bisnis utama pada modul POS Tobaku Halal digambarkan dalam Gambar 3.5. Gambar 3.5 ini adalah diagram alur yang menunjukkan detail langkah-langkah dalam proses bisnis dari modul POS Tobaku Halal, memberikan visualisasi yang jelas tentang bagaimana setiap bagian dari proses ini saling terkait dan berfungsi secara keseluruhan.



Gambar 3.5 Alur Bisnis POS Tobaku halal

Diagram alir pada Gambar 3.5 ini memberikan gambaran mengenai alur kerja sebuah sistem yang melibatkan pembeli dan pihak Tobaku halal. Pada bagian Pembeli, pengguna pertama kali diberikan pilihan untuk "*Login*" jika mereka sudah memiliki akun yang terdaftar, atau "Registrasi" jika mereka belum memiliki akun. Proses registrasi ini penting untuk mengidentifikasi dan mengotentikasi pengguna agar mereka bisa menggunakan layanan sistem. Setelah berhasil *login*, pembeli diarahkan untuk mengakses menu transaksi produk. Dalam menu ini, pembeli dapat melakukan *preorder* di Tobaku halal. Pembeli memiliki kesempatan untuk melihat detail produk yang mereka pilih. Setelah memilih produk pembeli dapat melanjutkan proses pembelian dengan pembayaran. Sistem *preorder* ini hanya menerima pembayaran non tunai. Kebijakan ini diterapkan untuk memastikan komitmen pembeli dalam melakukan pembelian.

Di sisi lain, Karyawan/Admin juga memiliki langkah awal yang mirip, dimulai dengan pilihan untuk "*Login*" atau "Registrasi". Proses ini memastikan bahwa hanya karyawan yang terotentikasi yang dapat mengakses fitur-fitur manajemen dalam sistem. Setelah berhasil *login*, karyawan/admin memiliki akses ke beberapa fungsi manajemen yang lebih kompleks dan beragam dibandingkan dengan pembeli. Fungsi pertama adalah Manajemen Produk, di mana admin dan karyawan dapat menambah produk baru atau mengubah detail produk yang sudah ada dalam sistem. Penambahan detail ini berasal dari koleksi bahan yang ada di Teratai. Dengan menambahkan detail, *traceability* suatu produk halal akan tetap terjaga.

Selain itu, sistem ini juga mencakup fitur manajemen kasir. Melalui fitur ini, karyawan atau admin memiliki wewenang untuk membuka dan menutup sesi kasir. Pada awal dan akhir setiap sesi, mereka dapat melakukan pencatatan keuangan. Proses ini memastikan bahwa semua transaksi yang terjadi selama jam operasional tercatat dengan baik.

Kemudian, terdapat fitur manajemen transaksi, di mana admin dan karyawan dapat melihat dan mengelola semua transaksi yang dilakukan oleh pembeli. Karyawan dan admin memiliki kewenangan untuk membatalkan transaksi jika diperlukan. Transaksi ini terbagi menjadi dua tipe pembelian, yaitu *preorder* dan bukan *preorder*. Transaksi bukan *preorder* berjalan seperti pembelian pada umumnya. Sedangkan untuk transaksi preorder, karyawan atau admin dapat melihat daftar *preorder*, memprosesnya, dan mengubah status sesuai dengan perkembangan yang ada. Sebagai tanda bahwa transaksi telah selesai, karyawan atau admin dapat menambahkan *batch* pada produk pembelian sehingga ketertelusuran (*traceability*) produk tetap terjaga. Diagram alir ini berakhir pada simpul "Selesai," yang menandakan bahwa semua alur kerja telah diselesaikan.

3.2.5 Desain Data

Setelah mendapatkan alur proses bisnis dari *usecase*, tahap selanjutnya adalah merancang entitas data yang akan digunakan. Proses ini dimulai dengan menganalisis dan meninjau kembali kebutuhan pengguna serta mengidentifikasi entitas dan atribut yang relevan. Contohnya, dalam modul POS Tobaku halal terdapat fungsionalitas menambah produk. Untuk memenuhi kebutuhan tersebut, diperlukan nama produk, ID Halal produk, gambar produk, dan detail lain dari masing-masing produk. Dari sini, dapat disimpulkan bahwa dalam perancangan modul POS Tobaku halal diperlukan entitas produk. Setelah mengidentifikasi entitas tersebut, langkah selanjutnya adalah menganalisis atribut apa saja yang dibutuhkan.

Hasil dari peninjauan dan analisis kebutuhan basis data yang akan digunakan dalam pengembangan backend modul POS Tobaku halal dicantumkan dalam Tabel 3.10. Tabel 3.10

ini menyajikan daftar entitas yang akan digunakan dalam pengembangan modul POS Tobaku halal yang berisikan nama entitas beserta penjelasan fungsinya.

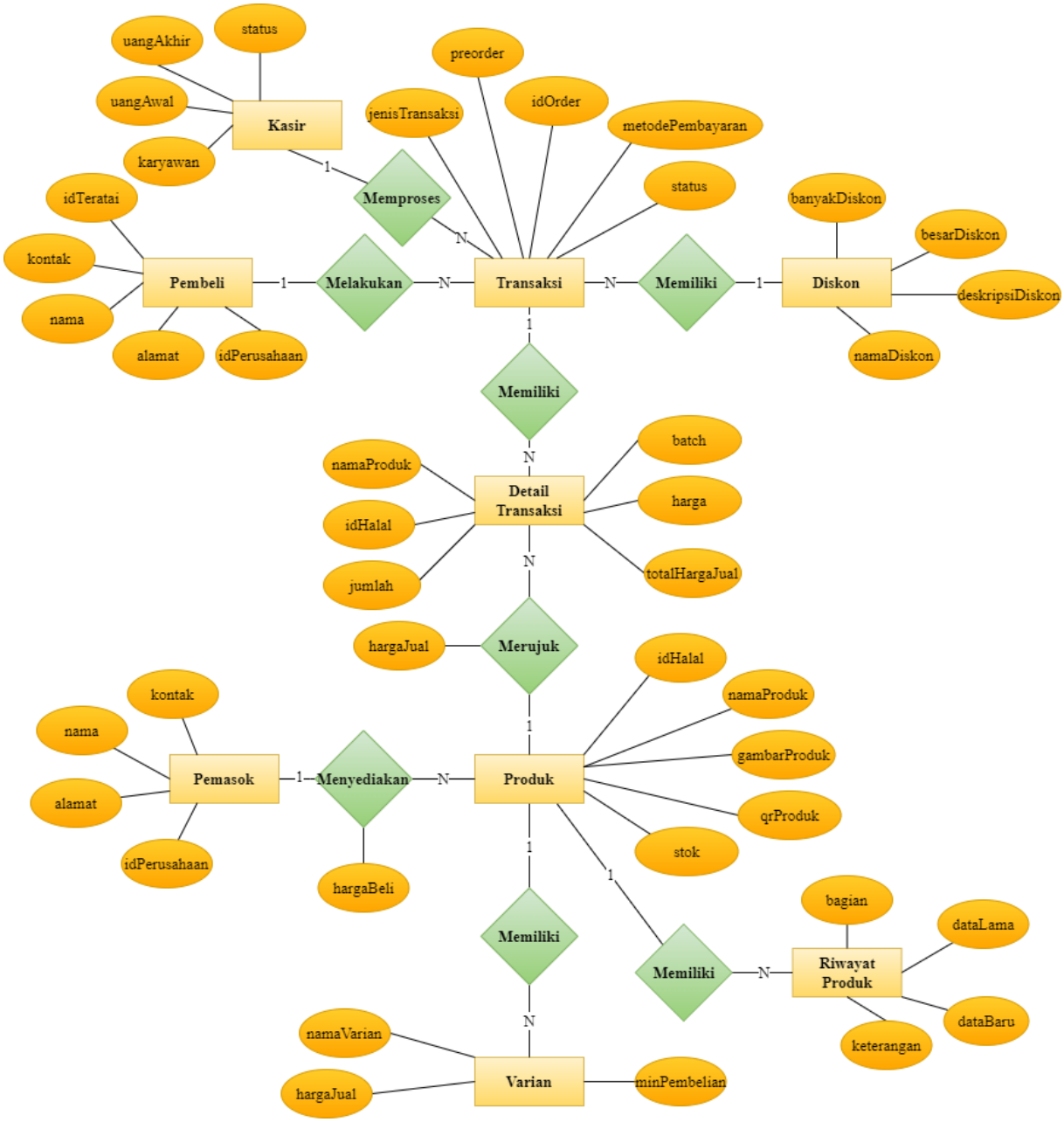
Tabel 3.10 Daftar Entitas Modul POS Tobaku halal

ID	Entitas	Deskripsi
E-01	Produk	Entitas ini mencakup semua informasi terkait produk yang tersedia dalam sistem.
E-02	Variasi Harga Produk	Entitas ini mencakup berbagai variasi harga dari produk yang ditawarkan.
E-03	Riwayat Produk	Entitas ini melacak perubahan atau pembaruan produk dalam sistem.
E-04	Pemasok	Entitas ini mencakup informasi tentang pemasok yang menyediakan produk.
E-05	Diskon	Entitas ini mencakup detail tentang diskon yang ditawarkan pada produk atau pada transaksi.
E-06	Pembeli	Entitas ini mencakup informasi tentang pembeli yang terdaftar dalam sistem.
E-07	Kasir	Entitas ini mencakup informasi tentang kasir yang bekerja dalam sistem.
E-08	Transaksi	Entitas ini mencakup semua informasi terkait transaksi yang dilakukan.
E-09	Detail Transaksi	Entitas ini mencakup rincian setiap item yang dibeli dalam suatu transaksi.

Tabel 3.10 merupakan daftar entitas yang diperoleh dari proses peninjauan kembali terhadap proses bisnis dan *usecase*. Dari proses ini, diidentifikasi entitas-entitas yang akan digunakan dalam pengembangan sistem. Setelah mengidentifikasi entitas-entitas yang akan digunakan dalam pengembangan *backend* modul POS toko bahan baku halal. Langkah selanjutnya dalam perancangan suatu *database* adalah membuat diagram yang digunakan untuk menunjukkan relasi antar objek atau entitas beserta atribut-atributnya. Pada tahap ini, konsep awal basis data akan dimodelkan dan dihubungkan antar data yang ada pada modul *point of sale* (POS) toko bahan baku halal.

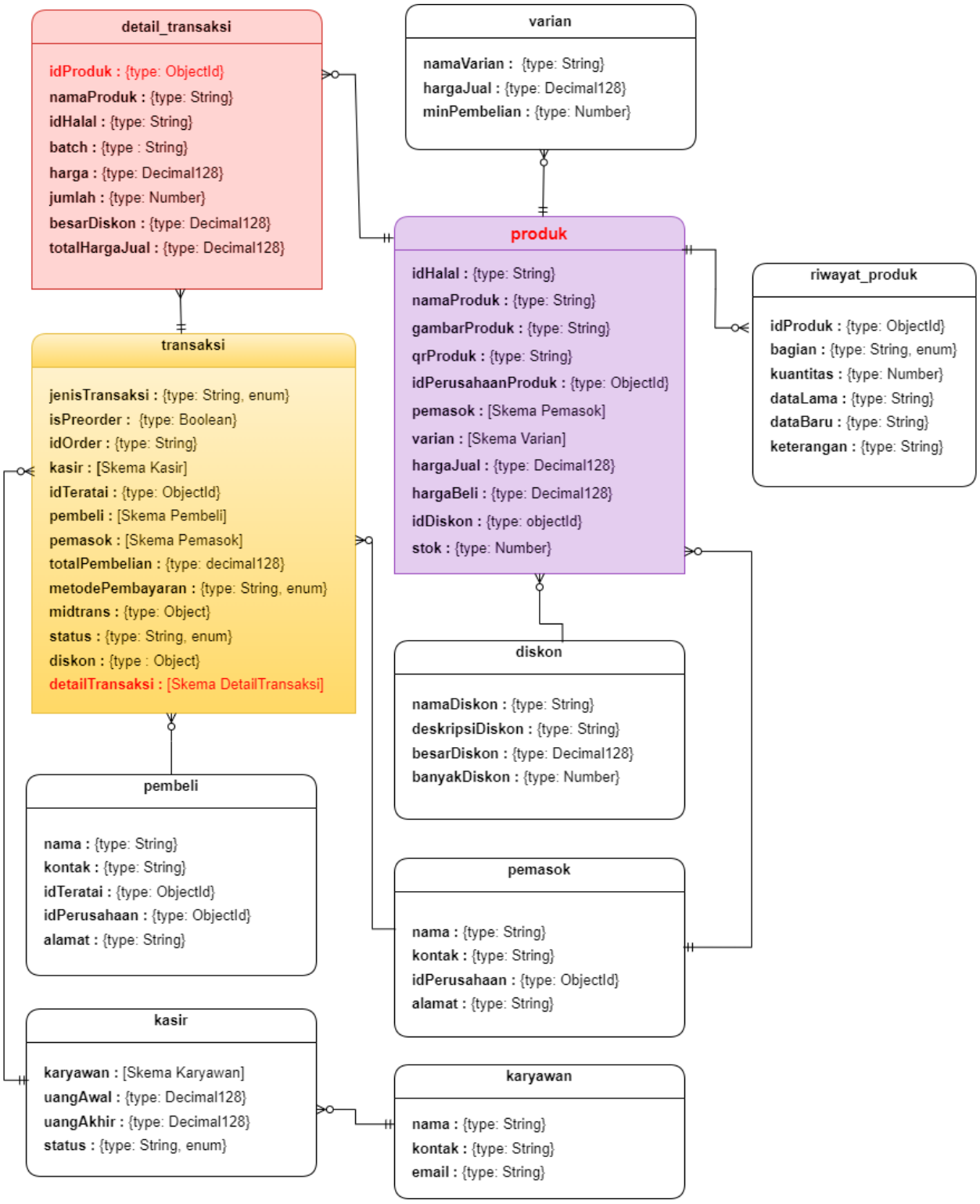
Gambar 3.6 merupakan diagram *Entity Relationship Diagram* (ERD) yang menggambarkan hubungan antar entitas dalam sistem POS toko bahan baku halal. Diagram ini terdiri dari beberapa entitas, yaitu Kasir, Pembeli, Transaksi, Diskon, Detail Transaksi, Produk, Pemasok, Varian, dan Riwayat Produk. Transaksi memiliki atribut seperti preorder, idOrder, jenisTransaksi, metodePembayaran, dan status. Transaksi memiliki hubungan dengan Detail Transaksi, yang mencakup atribut seperti namaProduk, idHalal, jumlah, harga, totalHargaJual, dan hargaJual, yang merujuk pada Produk. Produk memiliki atribut seperti idHalal, namaProduk, gambarProduk, qrProduk, dan stok, dan disuplai oleh Pemasok yang memiliki atribut seperti nama, kontak, alamat, dan idPerusahaan. Produk juga memiliki banyak Varian dengan atribut seperti namaVarian, hargaJual, dan idVarian, serta memiliki Riwayat Produk yang memiliki atribut untuk menyimpan data lama dan data baru. Setiap entitas dihubungkan dengan garis yang menunjukkan hubungan antara mereka, baik itu "memiliki", "merujuk", atau "menyediakan". Diagram ini membantu dalam memahami hubungan antar entitas dalam sistem penjualan yang kompleks.

Garis pada gambar Gambar 3.6 menunjukkan hubungan antara entitas. Diagram ini menggunakan garis-garis untuk menunjukkan jenis hubungan antara entitas, seperti relasi 1:N dan N:1. Contoh relasi 1 : N adalah antara entitas Transaksi dan Detail Transaksi, di mana satu transaksi dapat memiliki banyak detail transaksi. Sedangkan relasi N:1 terlihat antara entitas Produk dan Pemasok, dimana banyak produk dapat disuplai oleh satu pemasok. Garis-garis ini membantu memahami arah hubungan antar entitas, dengan notasi kardinalitas memberikan informasi tentang jumlah maksimum hubungan yang diizinkan. Dengan demikian, diagram ERD ini mendukung pemahaman yang lebih baik tentang sistem POS tobaku halal.



Gambar 3.6 Entity Relationship Model POS Tobaku halal

ERD (*Entity Relationship Diagram*) telah disusun untuk menggambarkan sistem POS toko bahan baku halal seperti pada Gambar 3.6. Dari ERD tersebut, kemudian dikembangan skema *database*.



Gambar 3.7 Skema Database POS Tobaku halal

Karena menggunakan MongoDB, struktur data akan berbentuk dokumen dengan koleksi-koleksi yang mewakili entitas seperti Kasir, Pembeli, Transaksi, Diskon, Detail Transaksi, Produk, Pemasok, Varian, dan Riwayat Produk seperti pada Gambar 3.7. Setiap entitas dalam ERD diimplementasikan sebagai koleksi yang terdiri dari dokumen-dokumen

JSON, yang memuat atribut-atribut dan referensi ke dokumen lain jika diperlukan. Dokumen-dokumen ini memuat atribut-atribut yang relevan, seperti nama, harga, dan deskripsi, serta referensi ke dokumen lain jika diperlukan, misalnya referensi dari koleksi Transaksi ke koleksi Diskon.

Diagram skema *database* pada Gambar 3.7 menggambarkan struktur data dan hubungan antar entitas dalam sistem POS tobaku halal. Skema ini mencakup beberapa entitas utama yaitu transaksi, detail transaksi, produk, varian, diskon, pemasok, pembeli, kasir, riwayat produk, dan karyawan. Sebagai contoh, koleksi transaksi menyimpan informasi mengenai transaksi, seperti jenis transaksi (penjualan atau pembelian), apakah transaksi adalah preorder atau bukan, ID Order, ID kasir, informasi pembeli, total pembelian, metode pembayaran, dan status transaksi. Skema transaksi memiliki relasi dengan beberapa koleksi lainnya.

Dalam MongoDB, relasi antar entitas dapat diimplementasikan menggunakan dua pendekatan utama: referensi (*reference*) dan *embed*. Referensi digunakan ketika ingin menghubungkan dokumen dari satu koleksi dengan dokumen di koleksi lain yang memungkinkan untuk menjaga data tetap normal dan menghindari redundansi. Contohnya, koleksi detail_transaksi dengan koleksi produk yang dihubungkan dengan idProduk. Sebaliknya, *embed* digunakan untuk menyimpan subdokumen dalam dokumen utama. Hal ini berguna ketika data yang terkait erat ingin disimpan bersama untuk memudahkan pengaksesan dan pengelolaan. Misalnya, detail transaksi bisa di-*embed* langsung dalam dokumen Transaksi, sehingga setiap transaksi menyimpan semua item detail yang terkait secara langsung.

Dengan pendekatan ini, skema database MongoDB dari sistem POS ini tidak hanya memungkinkan untuk melihat bagaimana data diatur dan dihubungkan, tetapi juga mendukung pengelolaan data yang efisien dan konsisten. Hal ini mempermudah proses pengembangan dan pemeliharaan sistem yang kompleks tersebut. Setelah tahap perancangan basis data pada POS tobaku halal telah ditetapkan selanjutnya adalah dilakukan API *endpoint* untuk POS tobaku halal.

3.2.6 API *Endpoint*

Untuk memastikan *frontend* POS tobaku halal dapat mengakses dan memanfaatkan data secara efektif, diperlukan perancangan API *Endpoint* yang tepat. API *Endpoint* ini tidak hanya memudahkan komunikasi antara aplikasi *frontend* dan *backend*, tetapi juga menyediakan struktur yang terstruktur dan efisien untuk pengambilan data. Hasil perancangan API *Endpoint* untuk *backend* POS tobaku halal dapat dilihat di dalam Tabel 3.11. Tabel 3.11 berisikan daftar rancangan API *Endpoint*. Pada tabel disertakan nama *endpoint*, *method* yang digunakan dan penggunaan dari *endpoint* dalam POS tobaku halal.

Tabel 3.11 API Endpoint POS Tobaku Halal

No	*Method*	*Endpoint*	Penggunaan
1	GET	/api/v1/tobaku/laporan/laba-rugi	Menyediakan laporan laba dan rugi untuk periode tertentu.
2	GET	/api/v1/tobaku/laporan/transaksi	Menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.
3	GET	/api/v1/tobaku/laporan/penjualan-per-produk	Menyediakan laporan penjualan berdasarkan setiap produk yang

No	*Method*	*Endpoint*	Penggunaan
			dijual.
4	GET	/api/v1/tobaku/laporan/pembelian	Menyediakan laporan pembelian yang dilakukan oleh perusahaan.
5	GET	/api/v1/tobaku/laporan/penjualan	Menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.
6	GET	/api/v1/tobaku/laporan/laba-rugi/excel	Menyediakan laporan *excel* laba dan rugi untuk periode tertentu.
7	GET	/api/v1/tobaku/laporan/transaksi/excel	Menyediakan laporan *excel* yang mendetail semua transaksi yang terjadi dalam periode tertentu.
8	GET	/api/v1/tobaku/laporan/penjualan-per-produk/excel	Menyediakan laporan *excel* penjualan berdasarkan setiap produk yang dijual.
9	GET	/api/v1/tobaku/laporan/pembelian/excel	Menyediakan laporan *excel* pembelian yang dilakukan oleh perusahaan.
10	GET	/api/v1/tobaku/laporan/penjualan/excel	Menyediakan laporan *excel* penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.
11	GET	/api/v1/tobaku/produk	Menyediakan daftar lengkap semua produk yang tersedia di sistem.
12	POST	/api/v1/tobaku/produk	Menambah data produk baru ke dalam sistem.
13	PATCH	/api/v1/tobaku/produk/tambah-stok/:id_produk	Menambah jumlah stok produk yang ada di sistem.
14	PATCH	/api/v1/tobaku/produk/:id_produk	Mengedit informasi produk yang sudah ada di sistem.
15	DELETE	/api/v1/tobaku/produk/:id_produk	Menghapus produk dari sistem.
16	GET	/api/v1/tobaku/produk/:id_produk	Menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.
17	GET	/api/v1/tobaku/:id_produk/varian	Menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.
18	POST	/api/v1/tobaku/:id_produk/varian	Menambah data varian baru dari suatu produk ke dalam sistem.
19	PATCH	/api/v1/tobaku/:id_produk/varian/:id_varian	Mengedit informasi varian produk yang sudah ada di sistem.
20	DELETE	/api/v1/tobaku/:id_produk/varian/:id_varian	Menghapus varian produk dari sistem.

No	*Method*	*Endpoint*	Penggunaan
21	GET	/api/v1/tobaku/:id_produk/varian/:id_varian	Menyediakan informasi detail mengenai setiap varian produk.
22	POST	/api/v1/tobaku/riwayat-produk	Menambah data riwayat baru terkait produk ke dalam sistem.
23	GET	/api/v1/tobaku/riwayat-produk	Menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.
24	GET	/api/v1/tobaku/riwayat-produk/:id_riwayat	Menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.
25	GET	/api/v1/tobaku/diskon	Menyediakan daftar lengkap semua diskon yang tersedia di sistem.
26	POST	/api/v1/tobaku/diskon	Menambah data diskon baru ke dalam sistem.
27	GET	/api/v1/tobaku/diskon/:id_diskon	Menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.
28	DELETE	/api/v1/tobaku/diskon/:id_diskon	Menghapus diskon dari sistem.
29	PATCH	/api/v1/tobaku/diskon/:id_diskon	Mengedit informasi diskon yang sudah ada di sistem.
30	POST	/api/v1/tobaku/pemasok	Menambah data pemasok baru ke dalam sistem.
31	GET	/api/v1/tobaku/pemasok	Menyediakan daftar lengkap semua pemasok yang tersedia di sistem.
32	GET	/api/v1/tobaku/pemasok/:id_pemasok	Menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.
33	PATCH	/api/v1/tobaku/pemasok/:id_pemasok	Mengedit informasi pemasok yang sudah ada di sistem.
34	DELETE	/api/v1/tobaku/pemasok/:id_pemasok	Menghapus pemasok dari sistem.
35	GET	/api/v1/tobaku/transaksi	Menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.
36	GET	/api/v1/tobaku/transaksi/:id_transaksi	Menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.
37	POST	/api/v1/tobaku/transaksi	Membuat transaksi baru di dalam sistem.
38	PATCH	/api/v1/tobaku/transaksi/update-status/:id_transaksi	Mengubah status transaksi.
39	PATCH	/api/v1/tobaku/transaksi/update-batch/:id_transaksi	Memperbarui *batch* produk dalam transaksi.

No	*Method*	*Endpoint*	Penggunaan
40	GET	/api/v1/tobaku/invoice/:id_transaksi	Menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.
41	GET	/api/v1/tobaku/pembeli	Menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.
42	POST	/api/v1/tobaku/pembeli	Menambah data pembeli baru ke dalam sistem.
43	GET	/api/v1/tobaku/pembeli/:id_pembeli	Menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.
44	PATCH	/api/v1/tobaku/pembeli/:id_pembeli	Mengedit informasi pembeli yang sudah ada di sistem.
45	DELETE	/api/v1/tobaku/pembeli/:id_pembeli	Menghapus pembeli dari sistem.
46	POST	/api/v1/tobaku/kasir/buka	Membuka kasir dengan mencatat nilai awal keuangan saat memulai shift kasir.
47	PATCH	/api/v1/tobaku/kasir/tutup/:idKasir	Menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.
48	GET	/api/v1/tobaku/kasir	Menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.
49	GET	/api/v1/tobaku/kasir/:idKasir	Menyediakan informasi detail mengenai setiap sesi kasir.
50	POST	/api/v1/tobaku/transaksi/preorder	Memungkinkan pengguna untuk menambah data *preorder*

Tabel 3.11 menyajikan daftar *endpoint* yang dapat digunakan dalam POS tobaku halal. Tabel tersebut menjelaskan masing-masing *endpoint* yang digunakan dalam sistem *backend* ini. Salah satu *endpoint* yang dijelaskan adalah endpoint /api/v1/tobaku/transaksi, yang digunakan untuk menambahkan data transaksi baru kedalam sistem. Hasil rancangan *endpoint* ini akan mempermudah proses penulisan kode program.

3.3 Implementasi

Tahap implementasi atau penulisan kode program adalah fase setelah desain sistem selesai, termasuk alur proses bisnis, rancangan basis data, dan rancangan API *Endpoint*. Pada fase ini, kode program ditulis dengan menggunakan kerangka kerja ExpressJS, yang merupakan kerangka kerja *web* aplikasi Node.js yang fleksibel dan minimalis. Pada bagian ini dijelaskan hasil implementasi aplikasi berdasarkan desain yang telah dibuat. Implementasi berupa kode sumber untuk membangun *backend*, termasuk penerapan *clean architecture* dalam struktur kode sumber proyek.

3.3.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak dari sistem yang digunakan sebagai lingkungan implementasi dalam mengembangkan Tugas Akhir ini ditunjukkan pada Tabel

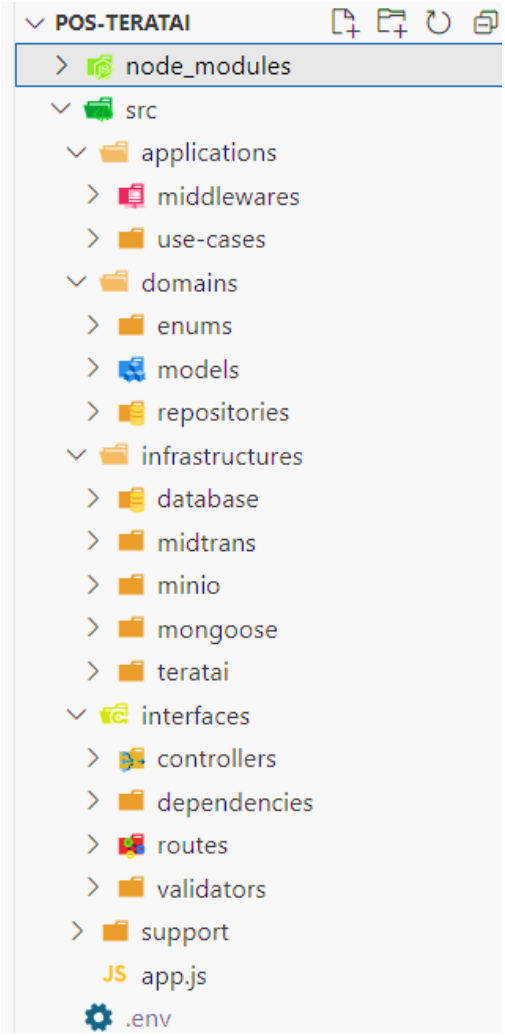
3.12.

Tabel 3.12 Lingkungan Implementasi

Jenis	Komponen	Spesifikasi
Perangkat Keras	Prosesor	Intel Core i5-1135G7 @ 2.4 GHz
	Memori	16 GB
Perangkat Lunak	Sistem Operasi	Windows 11 64-bit
	Perangkat Pengembang	Visual Studio Code, Git, Postman, MongoDB
	Bahasa Pemrograman	JavaScript
	Kerangka Kerja	ExpressJS

3.3.2 Implementasi Clean Architecture pada Struktur Proyek

Sistem POS tobaku halal dikembangkan dengan menerapkan prinsip *clean architecture*. Oleh karena itu, struktur direktori kode sumber diorganisasikan ke dalam beberapa folder yang mewakili lapisan-lapisan *clean architecture*, seperti yang ditunjukkan secara garis besar pada Gambar 3.8.



Gambar 3.8 Struktur Direktori

Gambar 3.8 menunjukkan folder yang melambangkan lapisan-lapisan dalam *clean architecture.* Berikut merupakan penjelasan fungsi dari tiap-tiap folder yang menyusun proyek POS Tobaku Halal.

1. **Folder src**
   Folder src (source) adalah tempat utama menyimpan semua kode sumber aplikasi. Folder ini adalah *direktori* yang berisi semua file dan folder yang dibutuhkan untuk menjalankan aplikasi, mengatur logika bisnis, interaksi dengan *database*, serta komponen-komponen lainnya. Di dalam folder ini terdapat *file* app.js, folder domain, folder infrastructure, folder interfaces, dan folder support.

2. **File app.js**
   App.js adalah *file* utama yang berfungsi sebagai *entry point* aplikasi. Pada *file* ini diinisialisasi konfigurasi aplikasi, *middleware*, rute-rute utama, serta pengaturan *server*. Pada dasarnya, *file* ini adalah tempat di mana aplikasi dijalankan.

3. **Folder Applications**
   Folder applications menyimpan logika bisnis aplikasi yang lebih tinggi. Berisi implementasi use-case dan middleware untuk memproses logika bisnis dan menangani permintaan dari lapisan interfaces. Di dalamnya terdapat sub folder sebagai berikut.
   a. ***middleware***: Menyimpan kode yang berjalan di antara permintaan HTTP dan *controller*. Middleware digunakan untuk mengecek otorisasi permintaan sebelum diteruskan ke use-case.
   b. ***usecases***: Menyimpan logika bisnis spesifik, atau kasus penggunaan, yang mendefinisikan bagaimana aplikasi merespons permintaan pengguna atau interaksi lainnya. Use case mengkoordinasikan berbagai operasi, mengimplementasikan aturan bisnis, dan memanggil repository untuk interaksi dengan database.

4. **Folder domains**
   Folder domains menyimpan entitas dan logika bisnis inti dari aplikasi. Folder ini dipecah menjadi beberapa sub-folder seperti enums, models, dan repositories.
   c. **enums**: Menyimpan enumerasi atau kumpulan konstanta yang digunakan di aplikasi.
   d. **models**: Menyimpan definisi model atau skema data yang digunakan dalam aplikasi, seperti skema *database*.
   e. **repositories**: Menyimpan kelas atau fungsi yang bertanggung jawab untuk berinteraksi dengan *database* atau penyimpanan data lainnya.

5. **Folder infrastructure**
   Folder infrastructures menyimpan kode yang berhubungan dengan infrastruktur aplikasi, seperti konfigurasi database, layanan pihak ketiga, dan lain-lain. Sub-folder di dalamnya mencakup.
   a. **database**: Kode yang berhubungan dengan konfigurasi dan koneksi database.
   b. **midtrans**: Kode untuk integrasi dengan layanan pembayaran Midtrans.
   c. **minio**: Kode untuk integrasi dengan layanan penyimpanan MinIO.
   d. **mongoose**: Kode yang terkait dengan penggunaan Mongoose, sebuah ODM untuk MongoDB.
   e. **teratai**: Kode untuk dapat terhubung dengan API Teratai.

6. **Folder interfaces**
   Folder interfaces menyimpan definisi antarmuka untuk interaksi eksternal aplikasi, seperti API. Di dalamnya terdapat subfolder sebagai berikut.
   a. **controllers**: Berisi logika pengontrol yang menangani permintaan HTTP dan merespon dengan data yang sesuai. Controllers juga akan memanggil *use case* yang sesuai.
   b. **dependencies**: Berisi pengaturan atau konfigurasi ketergantungan yang digunakan oleh antarmuka.
   c. **routes**: Menyimpan definisi rute atau *endpoint* untuk aplikasi.

7. **Folder support**
   Folder support menyimpan berbagai utilitas, *helper*, atau fungsi pendukung yang digunakan di berbagai bagian aplikasi.

8. **File .env**
   File .env adalah file konfigurasi yang umumnya digunakan dalam pengembangan perangkat lunak untuk menyimpan variabel lingkungan yang digunakan dalam aplikasi. File ini berisi pengaturan yang spesifik untuk lingkungan pengembangan atau produksi, seperti kunci API, pengaturan *database*, konfigurasi *server*, dan variabel-variabel lain yang diperlukan oleh aplikasi.

Setiap folder dan *file* memiliki peran dan fungsinya masing-masing yang saling melengkapi untuk membentuk aplikasi yang modular, terorganisir, dan mudah dikelola sesuai prinsip *clean architecture*. Dengan memahami struktur ini, pengembang dapat lebih efisien dalam mengembangkan, memelihara, dan mengembangkan fitur-fitur baru pada sistem.

3.3.3 Implementasi Kasus Penggunaan

Setelah proses desain sistem telah dibuat secara keseluruhan, langkah berikutnya adalah implementasi fungsionalitas. Pada tahap ini, desain yang telah disusun akan direalisasikan ke dalam bentuk kode program. Penulisan implementasi fungsionalitas ini menggunakan kerangka kerja Express JS. Implementasi kode sumber ini mencakup berbagai aspek penting dari aplikasi, mulai dari penanganan permintaan HTTP, pengelolaan logika bisnis, hingga interaksi dengan *database*. Berikut adalah beberapa contoh implementasi kode sumber untuk fungsionalitas pada kasus penggunaan menambahkan produk baru dan transaksi.

3.3.3.1 Implementasi Menambahkan Produk

Pada POS Tobaku Halal, produk berasal dari koleksi bahan yang tersedia dalam sistem Teratai. Setiap bahan disimpan dengan data identifikasi penjual dan pembeli, serta informasi seperti nama, ID Halal, dan *batch* produk. Namun, informasi terperinci seperti harga, varian, dan gambar produk belum ada pada koleksi tersebut. Untuk memastikan jejak halal tetap terjaga, penambahan produk dilakukan dengan memasukkan data baru yang terkait dengan ID Halal ke dalam koleksi produk di *database* POS Tobaku Halal. Dengan demikian, setiap transaksi baru yang terjadi dapat terhubung kembali ke sistem Teratai menggunakan ID Halal yang telah tercatat sebelumnya. Hal ini memungkinkan untuk menjaga jejak halal secara menyeluruh dari awal proses pembelian bahan hingga produk akhir yang dijual di POS Tobaku Halal. Kode Sumber 3.1 merupakan implementasi kode sumber untuk menambahkan produk baru pada POS Teratai.

```
async storeProduct({ idHalal, file, body, token, dependencies }) {
  try {
    // Langkah 1: Periksa apakah produk sudah ada di database
    const product = await this.repository.findOne({
                        idHalal: idHalal,
                        namaProduk: body.namaProduk
                    });
    if (product) {
      throw new CustomError(400, 'Produk sudah ada di sistem! Gunakan edit
      produk untuk mengedit');
    }
    // Langkah 2: Ambil data produk dari layanan Teratai dan upload gambar
    const [productTeratai, uploadImageResult] = await Promise.all([
      dependencies.terataiService.getProductByIdHalal(token, idHalal),
      file ? dependencies.minioUploader.uploadFile(file.originalname,
      file.buffer) : Promise.resolve(null)

    ]);
    // Ambil qr_code dan id_perusahaan dari hasil produk Teratai
    const { qr_code, id_perusahaan } = productTeratai;

    // Langkah 3: Buat data pemasok baru jika diperlukan
    const supplierData = await
    dependencies.supplierUsecase.createSupplierFromTeratai(
                            { idPerusahaan: id_perusahaan },
                            dependencies.terataiService, token
                        );
    // Persiapkan data produk untuk disimpan
    body = {
      ...body,
      namaProduk: body.namaProduk,
      qrProduk: qr_code,
      idPerusahaanProduk: id_perusahaan,
      gambarProduk: uploadImageResult,
      pemasok: supplierData
    };

    // Langkah 4: Simpan data produk ke dalam database
    const data = await this.repository.create(body);

    // Langkah 5: Persiapkan parameter untuk transaksi pembelian
    const parameter =
  dependencies.transactionUsecase.getPurchasedTransactionsParameter({
                        ...data._doc,
                        pemasok : supplierData
                    });

    // Langkah 6: Buat transaksi pembelian dan catatan riwayat produk baru
    const check = await Promise.allSettled([
      dependencies.transactionUsecase.create(parameter),
      dependencies.productHistoryUsecase.create({data})
    ]);
    console.log(check);

    // Mengembalikan data produk yang telah diproses
    return data;
  } catch (error) {
    // Tangkap dan lemparkan kembali kesalahan sebagai CustomError
    console.log(error);
    throw new CustomError(error.statusCode, error.message, error.data);
  }
}
```

Kode Sumber 3.1 Kode Sumber Kasus Penggunaan Menambahkan Produk Baru

Metode storeProduct pada Kode Sumber 3.1 adalah sebuah *async function* yang bertanggung jawab untuk menyimpan produk baru ke dalam sistem. Langkah-langkahnya dimulai dengan memeriksa apakah produk dengan idHalal dan namaProduk yang sama sudah ada di dalam *database*. Jika produk sudah ada, sebuah error kustom akan dilemparkan untuk memberitahu pengguna bahwa produk tersebut sudah terdaftar. Jika produk belum ada, langkah berikutnya adalah mengambil data produk dari layanan eksternal Teratai menggunakan idHalal, dan mengunggah gambar produk jika ada. Data dari produk Teratai, seperti qr_code dan id_perusahaan, akan diekstrak untuk digunakan dalam langkah selanjutnya.

Langkah ketiga melibatkan pembuatan data pemasok baru jika diperlukan, berdasarkan id_perusahaan yang diperoleh dari produk Teratai. Setelah itu, data produk dipersiapkan untuk disimpan ke dalam database, termasuk informasi seperti qrProduk, idPerusahaanProduk, gambarProduk, dan data pemasok sehingga pada langkah 4 data produk dapat disimpan.

Selanjutnya, langkah kelima menyiapkan parameter untuk transaksi pembelian yang terkait dengan produk yang baru saja dibuat, termasuk data produk dan informasi pemasok. Langkah terakhir adalah membuat transaksi pembelian menggunakan parameter yang sudah disiapkan, serta mencatat riwayat pembuatan produk baru. Semua langkah ini dijalankan secara asynchronous dan diawasi menggunakan *Promise.allSettled* untuk menangani hasil dari dua operasi *async* secara bersamaan.

Jika ada kesalahan dalam proses tersebut, seperti kesalahan jaringan atau kesalahan dari layanan eksternal, *error* akan ditangkap dan dilemparkan kembali sebagai *CustomError* dengan kode status dan pesan yang sesuai untuk memberitahu pengguna tentang masalah yang terjadi. Hasil akhir dari metode ini adalah data produk yang berhasil disimpan ke dalam sistem setelah semua langkah diproses dengan sukses.

3.3.3.2 Implementasi Menambahkan Transaksi

Transaksi di POS Tobaku Halal dibagi menjadi dua jenis utama, yaitu transaksi penjualan dan transaksi pembelian. Pada sistem ini, transaksi pembelian secara otomatis dibuat ketika ada perubahan stok pada produk. Untuk transaksi penjualan, sistem mendukung dua jenis penjualan: *preorder* dan non-*preorder*. Penjualan *preorder* memungkinkan pelanggan untuk memesan produk sebelum tersedia, sedangkan penjualan non-*preorder* adalah transaksi biasa di mana produk langsung tersedia untuk dijual. Implementasi kedua jenis penjualan ini dilakukan dengan membedakan rute yang digunakan dalam sistem. Setelah transaksi ini berstatus “SELESAI”, sistem akan secara otomatis terhubung dengan Teratai untuk memastikan *traceability* produk. Kode Sumber 3.2 merupakan implementasi kasus penggunaan membuat transaksi baru.

```
1.  async storeTransaction({ token, body, dependencies }) {
2.      // Langkah 1: Tentukan status transaksi dan buat ID pesanan
3.      const status = this.determineStatus(body);
4.      const orderId = generateOrderID();
5.
6.      // Langkah 2: Dapatkan data pelanggan, validasi kasir, dan hitung total
    transaksi secara paralel
7.      const [customerData, cashierData, { totalTransaction,
    discountTransaction, detailTransaction }] = await Promise.all([
8.          this.getOrCreateCustomer(token, body.idTeratai, dependencies),
9.          this.validateCashier(body, dependencies),
10.         this.calculateTotalTransaction(body, dependencies)
11.     ]);
12.
```

```
    // Langkah 3: Proses pembayaran cashless jika metode pembayaran adalah 'CASHLESS'
    let midtransResponse = '';
    if (body.metodePembayaran === 'CASHLESS') {
        midtransResponse = await this.handleCashlessPayment(customerData, totalTransaction, orderId, dependencies);
    }

    // Langkah 4: Siapkan data transaksi untuk disimpan
    const transactionData = {
        ...body,
        idOrder: orderId,
        kasir: cashierData,
        pembeli: customerData,
        status: status,
        totalPembelian: totalTransaction,
        diskon: body.idDiskon ? discountTransaction : null,
        midtrans: midtransResponse || null,
        detailTransaksi: detailTransaction
    }

    // Langkah 5: Simpan data transaksi ke repository
    let transaction = await this.repository.create(transactionData);

    // Langkah 6: Proses transaksi yang sudah selesai
    await this.processCompletedTransaction(token, transaction, dependencies);

    // Mengembalikan data transaksi
    return transaction;
}
```

Kode Sumber 3.2 Kode Sumber Kasus Penggunaan Menambahkan Transaksi Baru

Metode storeTransaction pada Kode Sumber 3.2 adalah sebuah fungsi asinkron yang menangani penyimpanan data transaksi dalam sistem. Pertama, metode ini menentukan status transaksi dan menghasilkan ID pesanan unik. Kemudian, secara paralel, metode ini mendapatkan data pelanggan, memvalidasi data kasir, dan menghitung total transaksi. Jika metode pembayaran yang dipilih adalah "*CASHLESS*", metode ini akan memproses pembayaran melalui sistem Midtrans dan menyimpan hasilnya. Setelah itu, data transaksi disiapkan untuk disimpan dengan menggabungkan informasi pelanggan, kasir, status, total pembelian, diskon (jika ada), dan detail transaksi. Data yang telah disiapkan kemudian disimpan dalam repository. Langkah terakhir adalah memproses transaksi yang telah selesai, dan metode ini akan mengembalikan data transaksi yang berhasil disimpan.

Apabila transaksi sudah “SELESAI”, maka sistem akan memproses dan mengirimkan data bahan ke Teratai. Hal ini dilakukan untuk memastikan kelanjutan *traceability* produk, sehingga setiap bahan yang digunakan dalam transaksi dapat dilacak secara menyeluruh dari awal hingga akhir.

```
async processCompletedTransaction(token, transaction, dependencies) {
    // Langkah 1: Periksa apakah transaksi telah selesai
    if (transaction.status !== 'SELESAI') {
        throw new CustomError(400, 'Transaksi belum selesai', {});
    }

    // Langkah 2: Kirim data ke Teratai jika pembeli memiliki idPerusahaan
    if (transaction.pembeli.idPerusahaan) {
```

```
9.          const modifiedDetailTransaction =
    transaction.detailTransaksi.map(item => ({
10.             id_perusahaan_penjual: idTobaku,
11.             id_perusahaan_pembeli: transaction.pembeli.idPerusahaan,
12.             id_batch: item.batch,
13.             id_halal: item.idHalal
14.         }));
15.         await dependencies.terataiService.storeTransaction(token,
    modifiedDetailTransaction);
16.     }
17.
18.     // Langkah 3: Kurangi stok produk jika bukan preorder
19.     if (transaction.isPreorder == false) {
20.         await Promise.all(
21.             transaction.detailTransaksi.map(
22.                 async (data) => {
23.                     // Kurangi stok produk
24.                     await dependencies.productUsecase.decreaseStock({
25.                         idProduct: data.idProduk,
26.                         qty: data.jumlah
27.                     });
28.                 }
29.             )
30.         );
31.     }
32.     return;
33. }
```

Kode Sumber 3.3 Kode Sumber Untuk Memproses Transaksi yang Sudah Selesai

Metode processCompletedTransaction pada Kode Sumber 3.3 adalah fungsi asinkron yang menangani proses pasca penyelesaian transaksi. Pertama, fungsi ini memeriksa status transaksi, dan jika statusnya bukan "SELESAI", fungsi ini akan melemparkan kesalahan (*CustomError*) dengan pesan "Transaksi belum selesai". Selanjutnya, jika pembeli dalam transaksi memiliki idPerusahaan, fungsi ini memodifikasi detail transaksi untuk menyertakan informasi perusahaan penjual (idTobaku) dan perusahaan pembeli (id_perusahaan_pembeli). Data yang telah dimodifikasi kemudian dikirim ke layanan Teratai melalui metode storeTransaction. Data tersebut kemudian akan menjadi data pada koleksi bahan. Dari data bahan-bahan akan dapat dicari *traceability* halal suatu produk. Terakhir, jika transaksi bukan *preorder*, fungsi ini mengurangi stok produk yang terlibat dalam transaksi dengan memanggil metode decreaseStock dari productUsecase untuk setiap produk yang terlibat.

3.3.4 Implementasi Gerbang Pembayaran Midtrans

Implementasi Midtrans sebagai gerbang pembayaran untuk transaksi pada POS Tobaku Halal melibatkan beberapa tahapan penting. Dengan menggunakan Midtrans, proses dan verifikasi pembayaran dapat dikelola secara otomatis, sehingga lebih efisien dan mudah. Berdasarkan dokumentasi teknis yang tersedia, integrasi gerbang pembayaran Midtrans mencakup empat tahap utama yang harus diikuti.

3.3.4.1 Mendapatkan Token Transaksi Snap

Untuk memperoleh *token* transaksi Snap dari sisi *back-end*, beberapa komponen dalam HTTP *request* diperlukan. Komponen tersebut termasuk penggunaan *server key*, yang merupakan kode kunci yang dimiliki oleh pengguna terdaftar Midtrans untuk melakukan transaksi melalui gerbang pembayaran. Selain itu, terdapat transaction_details yang berisi informasi transaksi, seperti order_id yang adalah nomor identifikasi unik untuk setiap pesanan, dan gross_amount yang mencerminkan total nilai pesanan yang harus dibayar.

API Midtrans memverifikasi HTTP *request* menggunakan Basic Authorization dimana *username*-nya adalah *server key* pengguna. Implementasi HTTP *request* ke API Midtrans dari sisi *back-end* dijalankan melalui kelas MidtransService, yang terlihat dalam Kode Sumber 3.4.

```
1.  class MidtransService {
2.    constructor() {
3.        this.snap = new midtransClient.Snap({
4.            isProduction: config.isProduction,
5.            serverKey: config.serverKey,
6.            clientKey: config.clientKey
7.        });
8.    }
9.
10.   async createTransaction(parameter) {
11.      try {
12.          const transaction = await this.snap.createTransaction(parameter);
13.          return transaction;
14.      } catch (error) {
15.         throw new Error(`Error creating transaction: ${error.message}`);
16.      }
17.   }
18.    ....
19. }
20.
21. module.exports = MidtransService;
```

Kode Sumber 3.4 Implementasi Class MidtransService

Kode Sumber 3.4 mendefinisikan kelas MidtransService yang bertanggung jawab untuk berinteraksi dengan layanan Snap API dari Midtrans. Dalam konstruktor kelas, sebuah objek snap dari midtransClient.Snap dibuat dengan konfigurasi untuk lingkungan pengembangan yang sesuai serta menggunakan *serverKey* dan *clientKey* yang diperoleh dari sebuah *config file*. Metode createTransaction pada Kode Sumber 3.4 diimplementasikan untuk membuat transaksi baru dengan menggunakan Snap API, menangkap kesalahan yang mungkin terjadi selama prosesnya. Keseluruhan kelas MidtransService ini dapat diekspor untuk digunakan dalam modul lainnya melalui module.exports. Respons yang diterima ditunjukkan pada Kode Sumber 3.5. Atribut *token* pada objek tersebut disimpan pada data transaksi, yang kemudian akan diteruskan kembali ke *controller* sebagai bagian dari respons API pembuatan transaksi baru.

```
1.  {
2.    "token": "ca99be50-d46f-4742-82a7-426ba8031a42",
3.    "redirect_url":
    "https://app.sandbox.midtrans.com/snap/v4/redirection/ca99be50-d46f-4742-
    82a7-426ba8031a42"
4.  },
```

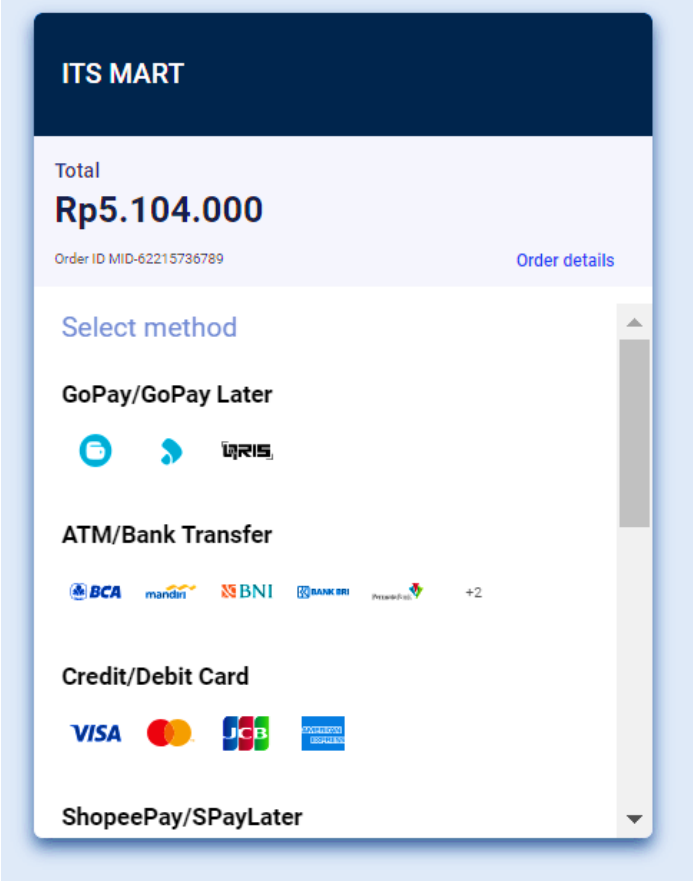
Kode Sumber 3.5 Objek Respons API Midtrans

Nilai kembalian yang diberikan oleh Midtrans API seperti pada Kode Sumber 3.5 adalah sebuah objek JSON yang memiliki dua properti utama. Properti pertama adalah "*token*" yang berisi sebuah *string* UUID yang unik untuk setiap transaksi. *Token* ini digunakan untuk mengidentifikasi secara spesifik transaksi yang telah dibuat di sistem Midtrans. Properti kedua adalah "*redirect_url"* yang berisi URL yang harus diarahkan oleh aplikasi pengguna ke halaman pembayaran Snap dari Midtrans. URL ini mengarahkan pengguna ke halaman yang sesuai

dengan token transaksi, di mana pengguna dapat menyelesaikan proses pembayaran dengan aman dan efisien. Dengan menyediakan *token* dan *redirect* URL, Midtrans memungkinkan aplikasi pengguna untuk mengintegrasikan proses pembayaran dengan baik, mengoptimalkan pengalaman pengguna dalam melakukan transaksi *online*.

3.3.4.2 Menampilkan Halaman Pembayaran Snap di Antarmuka Pengguna

Respons dari API Midtrans seperti yang terlihat dalam Kode Sumber 3.5 mengandung *redirect_url* yang merupakan tautan untuk mengarahkan pengguna langsung ke halaman pembayaran yang disediakan oleh Midtrans. Hal ini mempermudah dalam mengintegrasikan halaman pembayaran tanpa perlu melakukan penyesuaian tambahan pada aplikasi untuk menggunakan pustaka Snap.js. Jika tautan pada *redirect_url* diakses, halaman yang ditampilkan akan mirip dengan yang ditunjukkan dalam Gambar 3.9.



Gambar 3.9 Halaman Pembayaran Snap

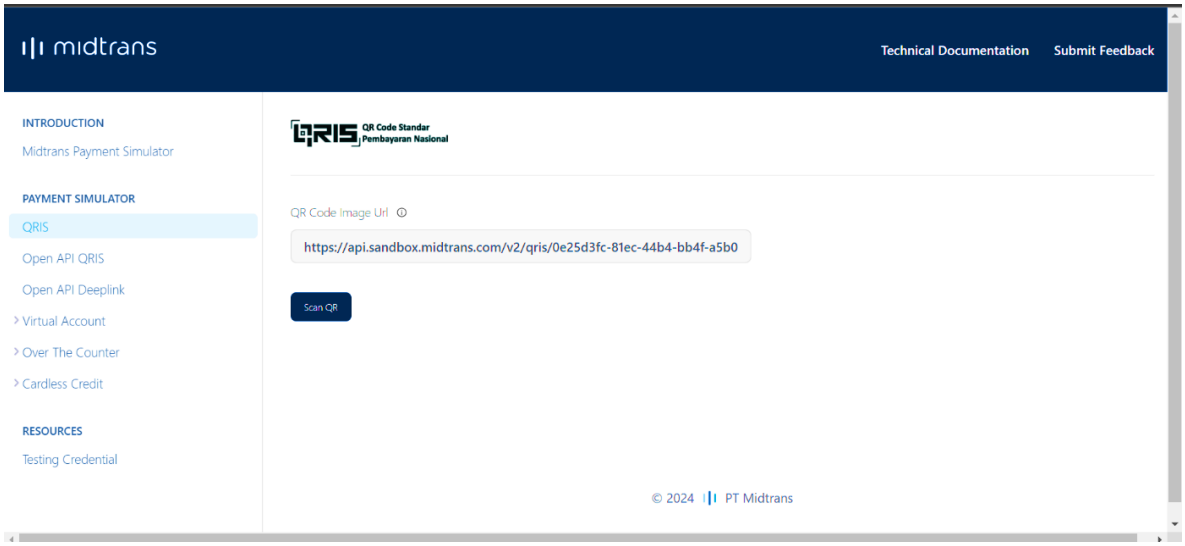
Gambar 3.9 menampilkan antarmuka halaman pembayaran Snap dari Midtrans yang digunakan oleh "ITS MART". Di bagian atas halaman, terlihat nama toko atau merchant "ITS MART". Di bawahnya, ditampilkan jumlah total yang harus dibayar oleh pengguna, serta nomor identifikasi pesanan unik Order ID. Bagian utama halaman menunjukkan berbagai metode pembayaran yang tersedia, termasuk GoPay/GoPay Later, QRIS, ATM/Bank Transfer, Credit/Debit, serta ShopeePay/SPayLater. Di sisi kanan halaman, terdapat scroll bar yang menunjukkan bahwa ada lebih banyak metode pembayaran atau informasi yang dapat diakses dengan menggulir ke bawah. Gambar 3.9 ini menunjukkan bagaimana Midtrans menyediakan halaman pembayaran yang *user-friendly* dan komprehensif, memudahkan pengguna dalam memilih metode pembayaran yang sesuai dengan preferensi pembeli.

3.3.4.3 Membuat Pembayaran Percobaan

Pembayaran dalam Tugas Akhir ini bersifat uji coba, yang berarti seluruh transaksi yang dilakukan tidak menggunakan uang nyata dan hanya meniru perilaku gerbang pembayaran dalam lingkungan produksi. Kredensial dan portal pembayaran yang digunakan untuk simulasi dalam lingkungan pengembangan ini sudah disediakan oleh Midtrans, sehingga transaksi dapat otomatis dianggap berhasil atau gagal oleh sistem Midtrans. Sebagai contoh, untuk metode pembayaran QRIS, Midtrans menyediakan portal simulator pembayaran di alamat https://simulator.sandbox.midtrans.com/qris/index. Setelah memilih QRIS sebagai metode pembayaran, seperti yang ditunjukkan pada Gambar 3.9, kode QR akan dibuat secara otomatis oleh sistem Midtrans. Pembeli dapat memindai kode QR tersebut, dan halaman akan menampilkan instruksi serta status pembayaran, seperti yang terlihat pada Gambar 3.10.

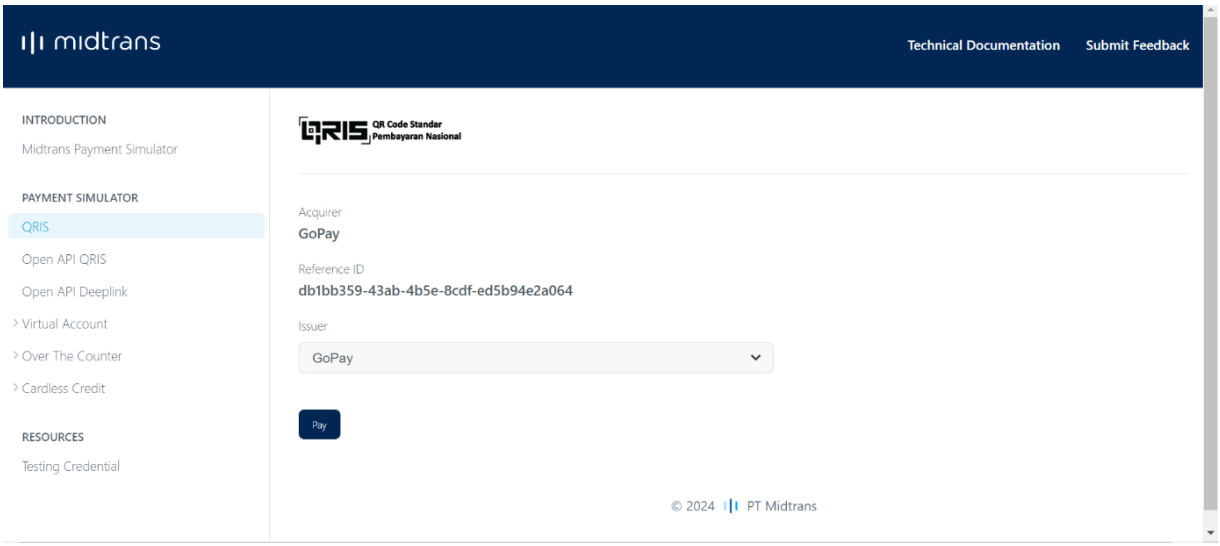
Gambar 3.10 Halaman Kode QRIS

Setelah mendapatkan kode QR yang menjadi tujuan pembayaran, kode tersebut dapat di salin alamat gambarnya untuk melakukan pembayaran simulasi menggunakan situs *web* simulator seperti yang telah disebutkan sebelumnya. Tampilan *web* simulator pembayaran ditunjukkan pada Gambar 3.11.



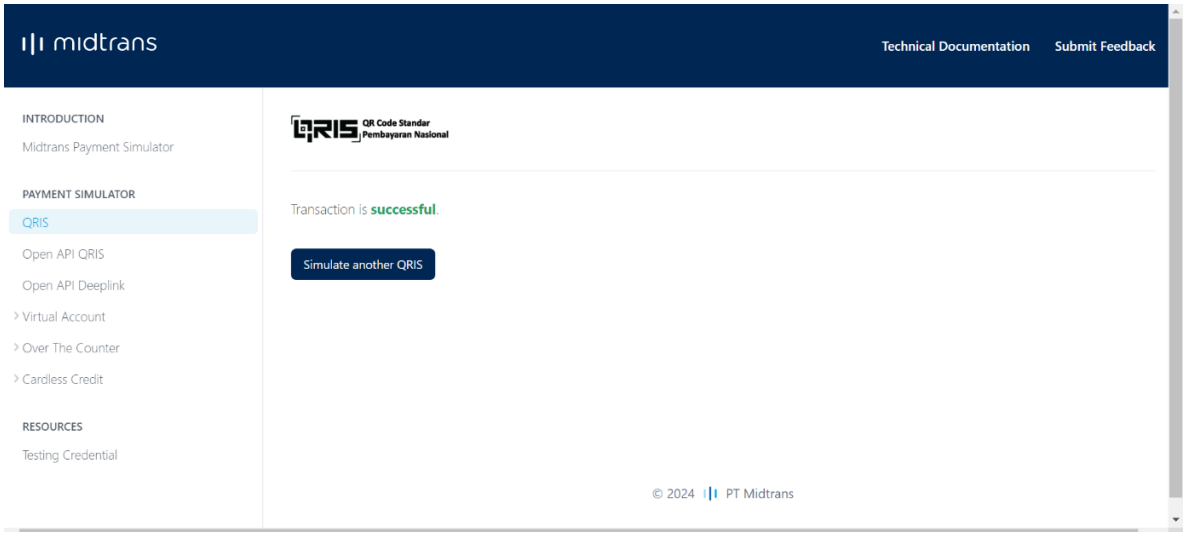
Gambar 3.11 Antarmuka Situs Web Simulator Pembayaran QRIS

Apabila tombol *Scan* QR pada web simulator di tekan, maka akan ditampilkan halaman untuk pembayaran seperti pada Gambar 3.12. Di bagian atas halaman, terlihat logo dan nama Midtrans. Terdapat beberapa informasi penting seperti penyedia layanan pembayaran "GoPay", ID referensi transaksi "db1bb359-43ab-4b5e-8cdf-ed5b94e2a064", dan penyedia dompet digital yang digunakan, yaitu "GoPay". Di bagian bawah halaman, terdapat tombol "Pay" yang dapat ditekan untuk melanjutkan proses pembayaran simulasi.



Gambar 3.12 Halaman Konfirmasi Pembayaran

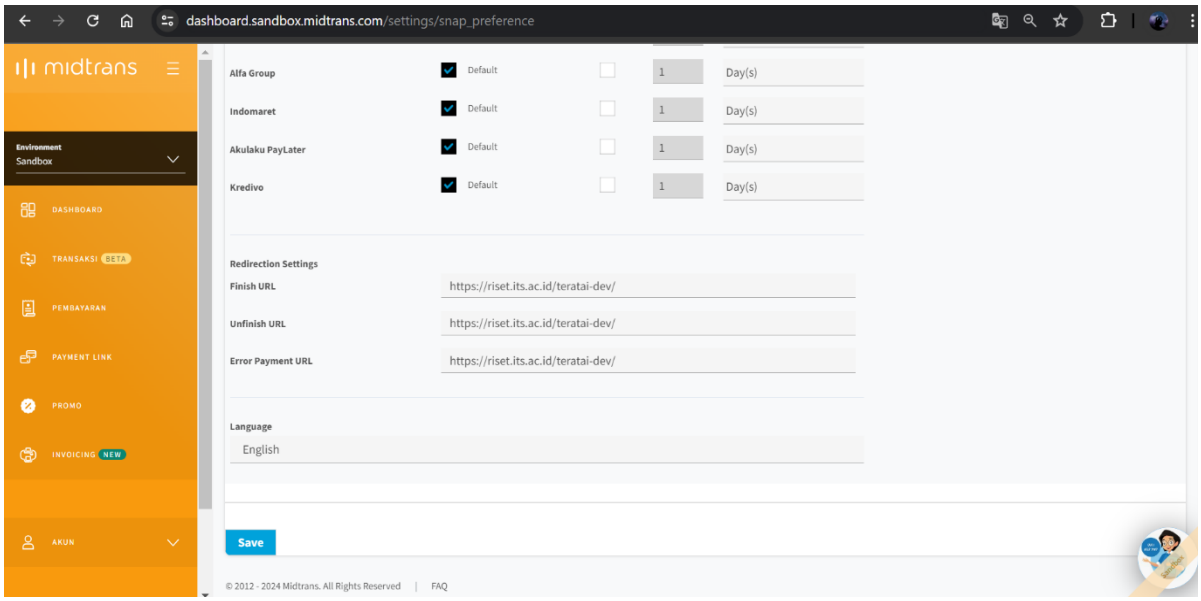
Setelah menekan tombol "Pay", sistem akan memproses transaksi dan kemudian menampilkan halaman status keberhasilan pembayaran QRIS untuk menunjukkan bahwa pembayaran telah berhasil dilakukan seperti pada Gambar 3.13.



Gambar 3.13 Halaman Status Keberhasilan Pembayaran dengan QRIS

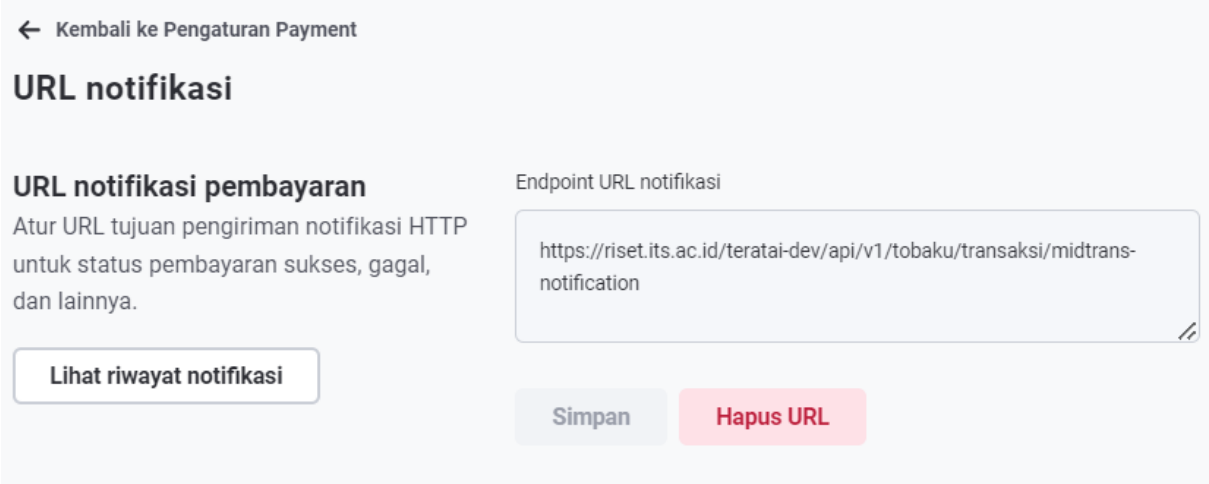
3.3.4.4 Menangani Notifikasi Status Pembayaran

Pada saat status transaksi berubah, seperti berhasil atau gagal, pembeli akan diarahkan ke halaman yang telah ditetapkan dalam *dashboard* pengguna Midtrans. Halaman ini ditunjukkan seperti pada Gambar 3.14 yang mencakup URL *Redirect* Selesai untuk transaksi yang berhasil, URL *Redirect* Belum Selesai untuk pengguna yang ingin kembali ke situs pemesanan tanpa menyelesaikan pembayaran, dan URL *Redirect* Kesalahan jika terjadi masalah selama proses pembayaran.



Gambar 3.14 Halaman Konfigurasi URL Pengalihan Midtrans

Selain pengalihan halaman (*redirect*) yang telah dikonfigurasi untuk menangani berbagai hasil dari proses pembayaran, Midtrans juga menyediakan mekanisme pemberitahuan pembayaran. Setelah sebuah transaksi diproses, Midtrans akan mengirimkan permintaan HTTP ke alamat yang telah ditetapkan oleh pengguna sebagai *Payment Notification* URL. Alamat ini dipilih dan diatur dalam *dashboard* konfigurasi Midtrans oleh pengguna untuk menerima pemberitahuan langsung dari Midtrans tentang status dan detail transaksi seperti pada Gambar 3.15.



Gambar 3.15 Halaman Konfigurasi URL notifikasi pemabayaran

HTTP *request* yang dikirim oleh Midtrans ke *Payment Notification* URL ini berbentuk objek JSON, yang mengandung informasi lengkap tentang transaksi yang telah berlangsung. Objek JSON ini berisi berbagai detail, seperti pada Kode Sumber 3.6. Pemberitahuan ini memungkinkan aplikasi atau sistem *backend* dari pengguna (seperti server web atau layanan lainnya) untuk secara *real-time* mengelola dan merespons status transaksi, misalnya untuk mengupdate database internal.

```
1. {
2.   "transaction_type": "on-us",
3.   "transaction_time": "2024-06-25 12:17:13",
4.   "transaction_status": "settlement",
5.   "transaction_id": "a9b4fe49-4445-472b-9efd-b9c83008b482",
6.   "status_message": "midtrans payment notification",
7.   "status_code": "200",
8.   "signature_key":
     "e5129c9a730945fb4a048913d8503dcc306df46b655bb7bacb6d1d8a5e5b3785f1a6865db6
     ce5292663fc243c9c7970ede56ce969ef9ec7e3d196f05148abcea",
9.   "settlement_time": "2024-06-25 12:17:30",
10.  "reference_id": "3aeee5db-5f1a-4a74-bb9d-25724c7cedd5",
11.  "payment_type": "qris",
12.  "order_id": "ORD-2406251217-636",
13.  "merchant_id": "G955735765",
14.  "issuer": "gopay",
15.  "gross_amount": "177300.00",
16.  "fraud_status": "accept",
17.  "expiry_time": "2024-06-25 12:32:13",
18.  "currency": "IDR",
19.  "acquirer": "gopay"
20. }
```

Kode Sumber 3.6 Objek Notifikasi Pembayaran Midtrans

Objek JSON notifikasi seperti pada Kode Sumber 3.6 dikirim melalui protokol HTTP ke alamat yang telah ditetapkan sebelumnya. Objek ini mencakup detail seperti waktu transaksi (transaction_time), status transaksi (transaction_status), ID unik transaksi (transaction_id), pesan status (status_message), kode status HTTP (status_code), waktu penyelesaian transaksi (settlement_time), ID referensi transaksi (reference_id), jenis pembayaran (payment_type), ID pesanan (order_id), ID merchant (merchant_id), penyedia layanan pembayaran (issuer), jumlah

pembayaran bruto (gross_amount), status kecurangan (fraud_status), waktu kedaluwarsa transaksi (expiry_time), mata uang (currency), dan penyedia layanan pembayaran (acquirer). Objek tersebut kemudian akan digunakan untuk melakukan update status transaksi pada sistem POS tobaku halal, seperti pada Kode Sumber 3.7.

```
async updateStatusMidtrans(transaction) {
    // Langkah 1: Mengambil data transaksi dari database berdasarkan orderId yang terkandung dalam objek transaksi dari midtrans.
    let transactionData = await this.repository.findOne({ idOrder: transaction.order_id });

    // Langkah 2: Jika transaksi tidak ditemukan dalam database, lemparkan error 404.
    if (!transactionData) {
        throw new CustomError(404, 'Transaksi tidak ditemukan', {});
    }

    // Langkah 3: Menggunakan switch statement untuk memproses berbagai status transaksi yang mungkin.
    switch (transaction.transaction_status) {
        case 'settlement':
            console.log('Pembayaran diselesaikan dari orderId', transaction.order_id);
            transactionData.status = transactionData.isPreorder === true ? "MENUNGGU ADMIN" : "SELESAI";
            await this.repository.update(transactionData._id, { status: transactionData.status });
            break;
        case 'deny':
        case 'cancel':
        case 'expire':
        case 'refund':
        case 'partial_refund':
            transactionData.status = "BATAL";
            await this.repository.update(transactionData._id, { status: transactionData.status });
            break;
        case 'pending':
            return;
        default:
            console.log('Status transaksi tidak dikenal:', transaction.transaction_status);
            return;
    }
    return;
}
```

Kode Sumber 3.7 Update Status Transaksi Berdasarkan Data Objek dari Midtrans

Metode updateStatusMidtrans pada Kode Sumber 3.7 adalah sebuah *async function* yang digunakan untuk memperbarui status transaksi berdasarkan data yang diterima dari Midtrans. Pertama, fungsi ini mencari data transaksi di *database* berdasarkan orderId yang ada dalam objek transaksi dari Midtrans. Jika transaksi tidak ditemukan, fungsi akan melemparkan error 404 menggunakan *CustomError*.

Setelah mendapatkan data transaksi dari *database*, langkah selanjutnya adalah menggunakan statement *switch* untuk memproses berbagai kemungkinan status transaksi yang dapat diterima dari Midtrans. Status-status seperti 'settlement', 'deny', 'cancel', 'expire', 'refund', dan 'partial_refund' dihandle dengan cara yang berbeda-beda:

1. Jika status transaksi adalah 'settlement', artinya pembayaran telah berhasil diselesaikan. Pada kasus ini, status di database diperbarui berdasarkan kondisi apakah transaksi ini merupakan preorder atau bukan.
2. Jika status adalah 'deny', 'cancel', 'expire', 'refund', atau 'partial_refund', transaksi dianggap dibatalkan dan statusnya diubah menjadi 'BATAL' di database.
3. Jika status adalah 'pending', tidak ada perubahan yang dilakukan pada data transaksi.
4. Untuk status lain yang tidak dikenali, fungsi akan mencatat bahwa status tersebut tidak dikenal dalam *console log* dan tidak melakukan perubahan apapun pada data transaksi di dalam database.

Fungsi ini menggunakan *async/await* untuk memastikan bahwa operasi-operasi database seperti pencarian dan pembaruan status dilakukan secara *asynchronous*, mengikuti aliran kode yang sesuai dengan status transaksi yang diterima dari Midtrans. Dengan demikian, notifikasi ini memainkan peran krusial dalam memastikan bahwa sistem pengguna dapat mengelola transaksi dengan baik.

3.4 Pengujian Sistem

Pengujian sistem yang akan dilakukan pada modul *point of sale* toko bahan baku halal yaitu dengan metode *black box* atau kotak hitam dengan skenario yang telah ditetapkan. *Black box testing* merupakan pendekatan yang mana menguji suatu sistem tanpa memiliki pengetahuan struktur internal program atau aplikasi. Pengujian ini akan menggunakan Postman sebagai alat untuk mengirimkan *request* dan akan dilakukan pengecekan pada *response* yang dikirimkan. Hasil pengujian akan dijabarkan dalam bentuk tabel. Contoh tabel hasil pengujian dapat dilihat pada Tabel 3.13.

Tabel 3.13 Contoh Tabel Pengujian

Menambah produk master	
ID	UAP-01
Skenario	Mengakses API *endpoint* untuk menambahkan diskon
Kondisi awal	Pengguna sudah memasukkan *token* sebagai administrator perusahaan
Hasil yang diharapkan	Sistem mengembalikan respons data diskon yang berhasil ditambahkan produk master dengan *status code* 201
Data uji coba	1. namaDiskon 2. besarDiskon
Langkah pengujian	2. Masukkan URL *endpoint* API untuk menambah diskon 3. Pilih metode POST 4. Masukkan data uji 4. Eksekusi pemanggilan API
Hasil yang diperoleh	Data diskon dengan *status code* 201
Hasil pengujian	Berhasil

Tabel 3.13 merupakan contoh dari tabel pengujian yang digunakan untuk menguji fungsionalitas API modul POS Tobaku halal. Dalam tabel tersebut, berbagai elemen penting pengujian dicatat, termasuk skenario pengujian, kondisi awal, hasil yang diharapkan, data uji coba, langkah pengujian, hasil yang diperoleh, dan hasil pengujian. Tabel ini menggambarkan langkah-langkah spesifik yang perlu dilakukan, seperti memasukkan URL *endpoint* API,

memilih metode HTTP yang sesuai, memasukkan data uji, dan mengeksekusi pemanggilan API. Selain itu, tabel ini juga mencatat hasil yang diperoleh dari pengujian untuk memastikan bahwa API berfungsi sesuai dengan spesifikasi yang diharapkan. Tabel pengujian seperti ini sangat penting dalam proses pengembangan perangkat lunak untuk memastikan bahwa setiap komponen sistem bekerja dengan benar dan memenuhi persyaratan yang telah ditetapkan.

Pada tugas akhir ini, pengujian fungsional terbagi menjadi dua jenis: pengujian positif dan pengujian negatif. Pengujian positif dilakukan untuk memastikan bahwa sistem berfungsi sesuai dengan yang diharapkan ketika diberikan input yang valid dan benar. Sedangkan pengujian negatif dilakukan untuk memverifikasi bahwa sistem dapat menangani input yang tidak valid atau kesalahan dengan baik, serta memberikan respons yang tepat dan aman. Pengujian ini penting untuk memastikan bahwa sistem tidak hanya berfungsi dengan baik dalam kondisi normal, tetapi juga mampu menangani situasi yang tidak terduga tanpa mengakibatkan kegagalan atau kerusakan pada sistem.

3.5 Pemeliharaan Sistem

Tahap pemeliharaan dilakukan untuk melakukan revisi-revisi yang terjadi pada fase pengembangan sebelumnya. Fase ini berfokus pada pemeliharaan dan perbaikan sistem perangkat lunak yang sudah dikembangkan. Tujuan utama dari fase pemeliharaan adalah untuk memastikan bahwa sistem perangkat lunak tetap berfungsi dengan baik, memenuhi kebutuhan pengguna, dan mengatasi masalah yang muncul setelah *deployment*.

Sebelum tahap pemeliharaan dimulai, sistem perangkat lunak perlu di-*deploy* terlebih dahulu. Dalam tugas akhir ini, *deployment* dilakukan pada VPS (*Virtual Private Server*) menggunakan Nginx sebagai *web server*. Langkah pertama adalah memastikan bahwa VPS telah disiapkan dengan sistem operasi yang sesuai, seperti Ubuntu, dan kemudian menginstal Nginx. Setelah itu, konfigurasi Nginx dilakukan dengan menyiapkan *file* konfigurasi di yang mencakup pengaturan *server* dan lokasi aplikasi. *File* konfigurasi ini kemudian disimboliskan ke direktori sites-enabled agar Nginx dapat mengenalinya.

Langkah berikutnya adalah memasang MinIO untuk menyimpan gambar dan berkas lainnya. MinIO adalah solusi *Object Storage* yang dapat diinstal di VPS, memungkinkan penyimpanan yang skalabel dengan akses melalui RESTful API. Proses instalasi melibatkan pengunduhan MinIO, konfigurasi lokasi penyimpanan data, dan opsionalnya, konfigurasi Nginx sebagai *reverse proxy* untuk memfasilitasi akses. Setelah terinstal, MinIO siap digunakan untuk menyimpan dan mengelola data secara efisien dalam lingkungan *server*.

Selanjutnya, kode sistem POS tobaku halal yang dikembangkan dipindahkan ke VPS menggunakan Git, dan dependensi yang diperlukan di-*install* serta sistem POS tersebut dijalankan di latar belakang menggunakan PM2. Nginx kemudian dimulai ulang untuk menerapkan konfigurasi baru. Dengan demikian, *deployment* sistem POS tobaku halal telah berhasil dilakukan, dan sistem siap untuk masuk ke tahap pemeliharaan. Fase pemeliharaan akan memastikan sistem tetap memenuhi ekspektasi pengguna, serta siap mengatasi tantangan yang mungkin muncul di masa mendatang.

BAB IV
HASIL DAN PEMBAHASAN

Bab ini membahas tentang hasil pengujian perangkat lunak yang dikembangkan dalam Tugas Akhir ini.

4.1 Hasil Pengujian Sistem

Tahap ini merupakan tahap pengujian *backend* modul POS toko bahan baku halal. Pada tugas akhir ini diimplementasikan dua jenis pengujian yang berbeda untuk memastikan kualitas dan kinerja sistem yang dikembangkan: pengujian fungsional dan non fungsional.

Pengujian fungsionalitas sistem berfokus pada verifikasi fitur dan kemampuan sistem dalam menjalankan fungsi-fungsi seperti transaksi penjualan, manajemen produk, dan integrasi dengan sistem pembayaran. Sedangkan, pengujian non fungsionalitas sistem menilai aspek-aspek yang sudah ditentukan pada kebutuhan non fungsional.

4.4.1 Pengujian Fungsionalitas Sistem

Dalam melakukan pengujian fungsionalitas sistem modul POS toko bahan baku halal, diterapkan metode *black box* atau kotak hitam. Pendekatan ini memungkinkan untuk menguji sistem tanpa perlu memiliki pengetahuan mendalam tentang struktur internal program atau aplikasi yang diuji. Pengujian dilakukan dengan menggunakan Postman sebagai alat untuk mengirimkan *request* dan melakukan pengecekan terhadap respons yang diterima. Melalui metode ini, dapat dipastikan bahwa sistem dapat berfungsi sesuai dengan skenario yang telah ditetapkan dan respons sesuai dengan yang diharapkan.

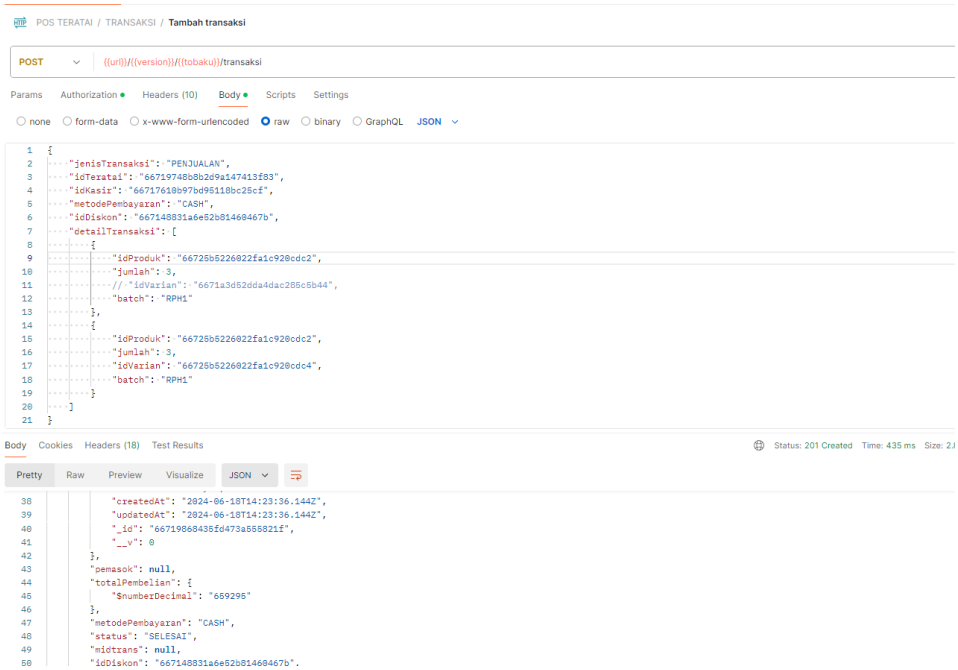
Pengujian fungsional dilakukan oleh tiga penguji. Satu penguji adalah penulis, sementara dua lainnya merupakan pihak lain yang bertindak sebagai *quality assurance*. Penguji tersebut merupakan mahasiswa Teknik Informatika ITS yang telah mengambil mata kuliah Pemrograman Web. Penguji memiliki latar belakang dalam pengembangan dan pengujian aplikasi *web* serta terlatih dalam teknik pengujian manual perangkat lunak menggunakan Postman. Tugas utama penulis adalah memastikan bahwa setiap fungsionalitas sistem berfungsi sesuai dengan spesifikasi yang telah ditetapkan. Di sisi lain, penguji lain bertanggung jawab untuk mengidentifikasi dan melaporkan *bug*, serta memastikan bahwa sistem bekerja dengan optimal dan memenuhi skenario yang diharapkan.

Pengujian ini juga terbagi menjadi dua jenis, yaitu pengujian positif dan pengujian negatif. Pengujian positif berfokus pada verifikasi bahwa sistem berfungsi sebagaimana mestinya ketika diberikan *input* yang valid dan sesuai dengan skenario penggunaan yang diharapkan. Misalnya, akan diuji apakah transaksi penjualan berhasil diproses, stok barang diperbarui dengan benar, dan integrasi dengan sistem pembayaran berjalan lancar ketika semua data yang dimasukkan valid. Apabila tidak ada data yang diuji, maka akan diuji apakah sistem memberikan *output* yang sesuai dengan skenario yang ditetapkan. Salah satu contoh pengujian positif adalah seperti pada Tabel 4.1 yang merupakan pengujian untuk menambahkan transaksi baru.

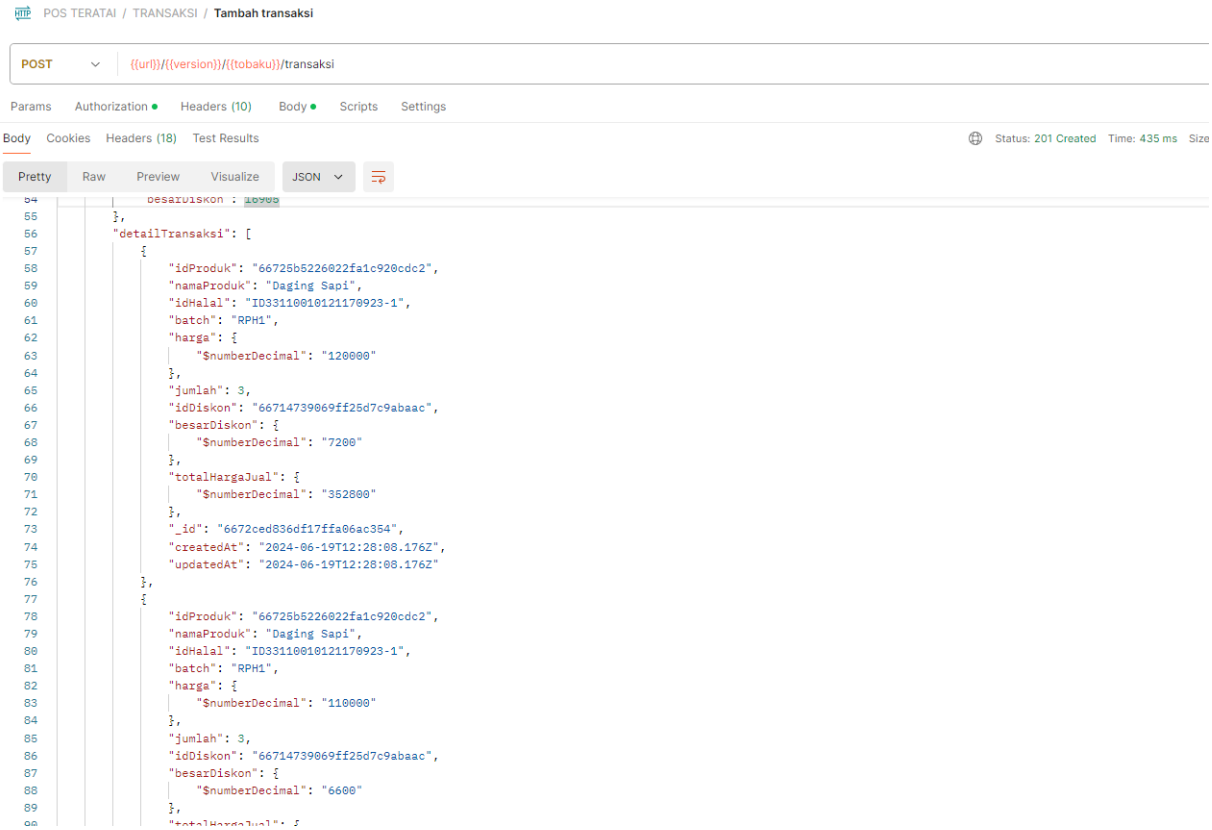
Tabel 4.1 Skenario Pengujian Positif Membuat Transaksi Baru

Membuat transaksi baru di dalam sistem	
ID	UFP-37
Skenario	Mengakses API *endpoint* untuk menambahkan transaksi
Kondisi awal	- Pengguna sudah memasukkan token untuk karyawan tobaku halal - Semua data yang diujikan valid
Hasil yang diharapkan	Sistem mengembalikan respons data transaksi baru di dalam sistem dengan *status code* 201
Data uji coba	1. jenisTransaksi 2. idTeratai 3. idKasir 4. metodePembayaran 5. idDiskon 6. detailTransaksi: {idProduk, jumlah, idVarian}
Langkah pengujian	1. Masukkan URL endpoint API untuk menambah transaksi baru 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API
Hasil yang diperoleh	Data transaksi baru di dalam sistem dengan *status code* 201
Hasil pengujian	Berhasil

Pengujian pada Tabel 4.1 bertujuan untuk memastikan bahwa sistem dapat membuat transaksi baru ketika menerima permintaan yang valid dari pengguna yang terotentikasi. Pengguna harus memasukkan *token* yang sah sebagai bukti otorisasi. Dengan metode POST, data transaksi dikirim sebagai *body request* ke *endpoint* API yang ditentukan. Jika semua data yang dikirimkan valid, sistem akan membuat transaksi baru dan mengembalikan data transaksi tersebut dengan *status code* 201, menandakan bahwa transaksi telah berhasil dibuat seperti pada Gambar 4.1 dan Gambar 4.2.



Gambar 4.1 Dokumentasi Pengujian Menambahkan Transaksi Baru 1



Gambar 4.2 Dokumentasi Pengujian Menambahkan Transaksi Baru 2

Tabel 4.2 merupakan daftar hasil pengujian positif yang telah dilakukan terhadap sistem *backend* modul POS toko bahan baku halal. Tabel ini berisikan ID pengujian, fungsional yang di uji, serta status pengujian. Dokumentasi dari setiap pengujian dapat dilihat pada lampiran.

Tabel 4.2 Hasil Uji Positif Fungsionalitas

ID	FUNGSIONAL	STATUS
UFP-01	Sistem mampu menyediakan laporan laba dan rugi untuk periode tertentu.	Berhasil
UFP-02	Sistem mampu menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Berhasil
UFP-03	Sistem mampu menyediakan laporan penjualan berdasarkan setiap produk yang dijual.	Berhasil
UFP-04	Sistem mampu menyediakan laporan pembelian yang dilakukan oleh perusahaan.	Berhasil
UFP-05	Sistem mampu menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.	Berhasil
UFP-06	Sistem mampu menyediakan laporan *excel* laba dan rugi untuk periode tertentu.	Berhasil
UFP-07	Sistem mampu menyediakan laporan *excel* yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Berhasil
UFP-08	Sistem mampu menyediakan laporan *excel* penjualan berdasarkan setiap produk yang dijual.	Berhasil

ID	FUNGSIONAL	STATUS
UFP-09	Sistem mampu menyediakan laporan *excel* pembelian yang dilakukan oleh perusahaan.	Berhasil
UFP-10	Sistem mampu menyediakan laporan *excel* penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.	Berhasil
UFP-11	Sistem mampu menyediakan daftar lengkap semua produk yang tersedia di sistem.	Berhasil
UFP-12	Sistem mampu memungkinkan pengguna untuk menambah data produk baru ke dalam sistem.	Berhasil
UFP-13	Sistem mampu memungkinkan pengguna untuk menambah jumlah stok produk yang ada di sistem.	Berhasil
UFP-14	Sistem mampu memungkinkan pengguna untuk mengedit informasi produk yang sudah ada di sistem.	Berhasil
UFP-15	Sistem mampu memungkinkan pengguna untuk menghapus produk dari sistem.	Berhasil
UFP-16	Sistem mampu menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.	Berhasil
UFP-17	Sistem mampu menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.	Berhasil
UFP-18	Sistem mampu memungkinkan pengguna untuk menambah data varian baru dari suatu produk ke dalam sistem.	Berhasil
UFP-19	Sistem mampu memungkinkan pengguna untuk mengedit informasi varian produk yang sudah ada di sistem.	Berhasil
UFP-20	Sistem mampu memungkinkan pengguna untuk menghapus varian produk dari sistem.	Berhasil
UFP-21	Sistem mampu menyediakan informasi detail mengenai setiap varian produk.	Berhasil
UFP-22	Sistem mampu memungkinkan pengguna untuk menambah data riwayat baru terkait produk ke dalam sistem.	Berhasil
UFP-23	Sistem mampu menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.	Berhasil
UFP-24	Sistem mampu menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.	Berhasil
UFP-25	Sistem mampu menyediakan daftar lengkap semua diskon yang tersedia di sistem.	Berhasil
UFP-26	Sistem mampu memungkinkan pengguna untuk menambah data diskon baru ke dalam sistem.	Berhasil
UFP-27	Sistem mampu menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.	Berhasil
UFP-28	Sistem mampu memungkinkan pengguna untuk menghapus diskon dari sistem.	Berhasil

ID	FUNGSIONAL	STATUS
UFP-29	Sistem mampu memungkinkan pengguna untuk mengedit informasi diskon yang sudah ada di sistem.	Berhasil
UFP-30	Sistem mampu memungkinkan pengguna untuk menambah data pemasok baru ke dalam sistem.	Berhasil
UFP-31	Sistem mampu menyediakan daftar lengkap semua pemasok yang tersedia di sistem.	Berhasil
UFP-32	Sistem mampu menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.	Berhasil
UFP-33	Sistem mampu memungkinkan pengguna untuk mengedit informasi pemasok yang sudah ada di sistem.	Berhasil
UFP-34	Sistem mampu memungkinkan pengguna untuk menghapus pemasok dari sistem.	Berhasil
UFP-35	Sistem mampu menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.	Berhasil
UFP-36	Sistem mampu menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.	Berhasil
UFP-37	Sistem mampu memungkinkan pengguna untuk membuat transaksi baru di dalam sistem.	Berhasil
UFP-38	Sistem mampu memungkinkan pengguna untuk mengubah status transaksi.	Berhasil
UFP-39	Sistem mampu memungkinkan pengguna untuk memperbarui *batch* produk dalam transaksi.	Berhasil
UFP-40	Sistem mampu menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.	Berhasil
UFP-41	Sistem mampu mendukung pembayaran melalui MidTrans.	Berhasil
UFP-42	Sistem mampu menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.	Berhasil
UFP-43	Sistem mampu memungkinkan pengguna untuk menambah data pembeli baru ke dalam sistem.	Berhasil
UFP-44	Sistem mampu menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.	Berhasil
UFP-45	Sistem mampu memungkinkan pengguna untuk mengedit informasi pembeli yang sudah ada di sistem.	Berhasil
UFP-46	Sistem mampu memungkinkan pengguna untuk menghapus pembeli dari sistem.	Berhasil
UFP-47	Sistem mampu memungkinkan pengguna untuk membuka kasir dengan mencatat nilai awal keuangan saat memulai *shift* kasir.	Berhasil
UFP-48	Sistem mampu memungkinkan pengguna untuk menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.	Berhasil
UFP-49	Sistem mampu menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.	Berhasil

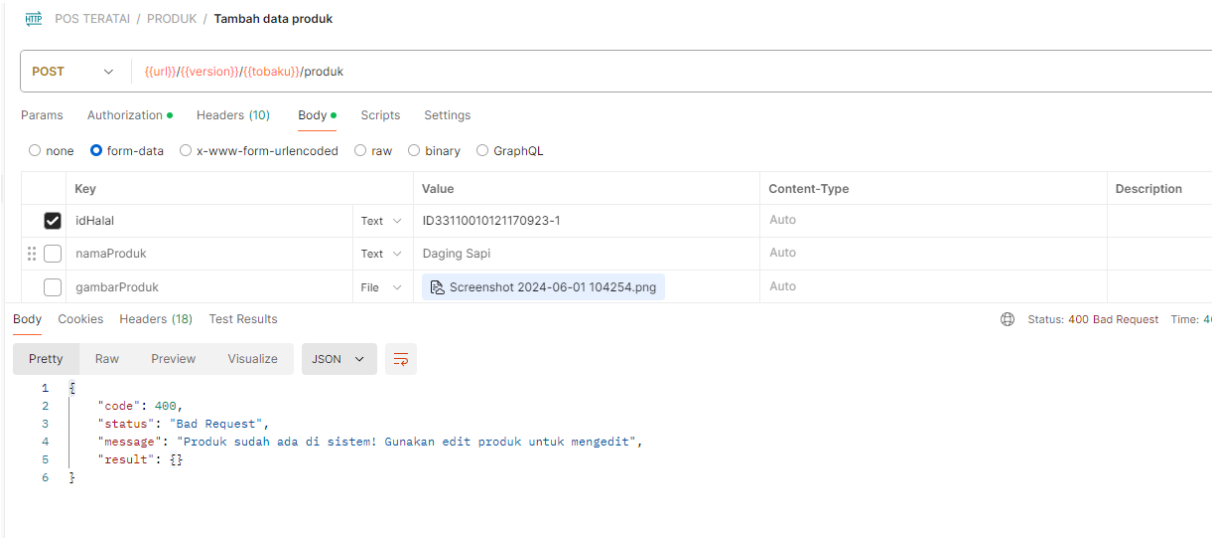
ID	FUNGSIONAL	STATUS
UFP-50	Sistem mampu menyediakan informasi detail mengenai setiap sesi kasir.	Berhasil
UFP-51	Sistem mampu memungkinkan pengguna untuk menambah data *preorder*	Berhasil

Pengujian negatif bertujuan untuk mengidentifikasi bagaimana sistem menangani input yang tidak valid atau kondisi yang tidak diharapkan. Contohnya, apabila memasukkan data yang salah atau tidak lengkap atau mencoba mengakses fitur tanpa otorisasi yang tepat. Tujuan dari pengujian negatif ini adalah untuk memastikan bahwa sistem dapat menangani kesalahan dengan baik, memberikan pesan *error* yang jelas, dan tetap menjaga integritas data serta keamanan sistem. Salah satu contoh pengujian negatif adalah seperti pada Tabel 4.3 yang merupakan pengujian untuk menambahkan produk baru.

Tabel 4.3 Skenario Pengujian Negatif Menambah Produk Baru

Menambah data produk baru ke dalam sistem	
ID	UFN-12
Skenario	Mengakses API *endpoint* untuk menambah data produk baru ke dalam sistem POS dengan idHalal sudah ada di sistem
Kondisi awal	Sudah memasukkan token sebagai administrator atau karyawan tobaku idHalal sudah ada
Hasil yang diharapkan	Sistem mengembalikan *error* 400 dengan pesan "Produk sudah ada di sistem! Gunakan edit produk untuk mengedit."
Data uji coba	1. idHalal 2. stok 3. hargaBeli 4. hargaJual
Langkah pengujian	1. Masukkan URL endpoint API untuk menambah produk 2. Pilih metode POST 3. Masukkan data uji 4. Eksekusi pemanggilan API
Hasil yang diperoleh	*Error* 400 dengan pesan "Produk sudah ada di sistem! Gunakan edit produk untuk mengedit."
Hasil pengujian	Berhasil

Pengujian pada Tabel 4.3 bertujuan untuk memastikan bahwa sistem dapat menangani upaya penambahan produk baru dengan idHalal yang sudah ada dalam *database* dengan cara yang benar. Setelah administrator atau karyawan toko memasukkan *token* otorisasi dan mencoba menambah produk baru dengan idHalal yang sudah terdaftar, sistem seharusnya memvalidasi bahwa produk tersebut sudah ada dan mengembalikan *error* 400 dengan pesan yang sesuai. Pengujian ini berhasil, menunjukkan bahwa sistem telah diimplementasikan dengan baik untuk mencegah duplikasi data produk dan memberikan panduan yang jelas kepada pengguna untuk menggunakan fitur edit produk jika ingin memperbarui informasi produk yang ada. Hasil pengujian pada Postman terlihat pada Gambar 4.3.



Gambar 4.3 Dokumentasi Pengujian Negatif Menambahkan Produk

Tabel 4.4 merupakan daftar hasil pengujian negatif yang telah dilakukan terhadap sistem *backend* modul POS toko bahan baku halal. Tabel ini berisikan ID pengujian, fungsional yang di uji, serta status pengujian. Dokumentasi dari setiap pengujian dapat dilihat pada lampiran.

Tabel 4.4 Hasil Uji Negatif Fungsionalitas

ID	FUNGSIONAL	STATUS
UFN-01	Sistem mampu menyediakan laporan laba dan rugi untuk periode tertentu.	Berhasil
UFN-02	Sistem mampu menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Berhasil
UFN-03	Sistem mampu menyediakan laporan penjualan berdasarkan setiap produk yang dijual.	Berhasil
UFN-04	Sistem mampu menyediakan laporan pembelian yang dilakukan oleh perusahaan.	Berhasil
UFN-05	Sistem mampu menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.	Berhasil
UFN-06	Sistem mampu menyediakan laporan *excel* laba dan rugi untuk periode tertentu.	Berhasil
UFN-07	Sistem mampu menyediakan laporan *excel* yang mendetail semua transaksi yang terjadi dalam periode tertentu.	Berhasil
UFN-08	Sistem mampu menyediakan laporan *excel* penjualan berdasarkan setiap produk yang dijual.	Berhasil
UFN-09	Sistem mampu menyediakan laporan *excel* pembelian yang dilakukan oleh perusahaan.	Berhasil
UFN-10	Sistem mampu menyediakan laporan *excel* penjualan yang mencakup total penjualan dalam periode tertentu, termasuk analisis penjualan.	Berhasil
UFN-11	Sistem mampu menyediakan daftar lengkap semua produk yang tersedia di sistem.	Berhasil

ID	FUNGSIONAL	STATUS
UFN-12	Sistem mampu memungkinkan pengguna untuk menambah data produk baru ke dalam sistem.	Berhasil
UFN-13	Sistem mampu memungkinkan pengguna untuk menambah jumlah stok produk yang ada di sistem.	Berhasil
UFN-14	Sistem mampu memungkinkan pengguna untuk mengedit informasi produk yang sudah ada di sistem.	Berhasil
UFN-15	Sistem mampu memungkinkan pengguna untuk menghapus produk dari sistem.	Berhasil
UFN-16	Sistem mampu menyediakan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia.	Berhasil
UFN-17	Sistem mampu menyediakan daftar lengkap semua varian dari suatu produk yang tersedia di sistem.	Berhasil
UFN-18	Sistem mampu memungkinkan pengguna untuk menambah data varian baru dari suatu produk ke dalam sistem.	Berhasil
UFN-19	Sistem mampu memungkinkan pengguna untuk mengedit informasi varian produk yang sudah ada di sistem.	Berhasil
UFN-20	Sistem mampu memungkinkan pengguna untuk menghapus varian produk dari sistem.	Berhasil
UFN-21	Sistem mampu menyediakan informasi detail mengenai setiap varian produk.	Berhasil
UFN-22	Sistem mampu memungkinkan pengguna untuk menambah data riwayat baru terkait produk ke dalam sistem.	Berhasil
UFN-23	Sistem mampu menyediakan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem.	Berhasil
UFN-24	Sistem mampu menyediakan informasi detail mengenai setiap riwayat produk, termasuk perubahan yang dilakukan, tanggal perubahan, dan keterangan perubahan.	Berhasil
UFN-25	Sistem mampu menyediakan daftar lengkap semua diskon yang tersedia di sistem.	Berhasil
UFN-26	Sistem mampu memungkinkan pengguna untuk menambah data diskon baru ke dalam sistem.	Berhasil
UFN-27	Sistem mampu menyediakan informasi detail mengenai setiap diskon, termasuk deskripsi dan persentase diskon.	Berhasil
UFN-28	Sistem mampu memungkinkan pengguna untuk menghapus diskon dari sistem.	Berhasil
UFN-29	Sistem mampu memungkinkan pengguna untuk mengedit informasi diskon yang sudah ada di sistem.	Berhasil
UFN-30	Sistem mampu memungkinkan pengguna untuk menambah data pemasok baru ke dalam sistem.	Berhasil
UFN-31	Sistem mampu menyediakan daftar lengkap semua pemasok yang tersedia di sistem.	Berhasil

ID	FUNGSIONAL	STATUS
UFN-32	Sistem mampu menyediakan informasi detail mengenai setiap pemasok, termasuk nama, kontak, dan alamat.	Berhasil
UFN-33	Sistem mampu memungkinkan pengguna untuk mengedit informasi pemasok yang sudah ada di sistem.	Berhasil
UFN-34	Sistem mampu memungkinkan pengguna untuk menghapus pemasok dari sistem.	Berhasil
UFN-35	Sistem mampu menyediakan daftar lengkap semua transaksi yang terjadi dalam sistem.	Berhasil
UFN-36	Sistem mampu menyediakan informasi detail mengenai setiap transaksi, termasuk item yang dibeli, harga, dan informasi pembeli.	Berhasil
UFN-37	Sistem mampu memungkinkan pengguna untuk membuat transaksi baru di dalam sistem.	Berhasil
UFN-38	Sistem mampu memungkinkan pengguna untuk mengubah status transaksi.	Berhasil
UFN-39	Sistem mampu memungkinkan pengguna untuk memperbarui *batch* produk dalam transaksi.	Berhasil
UFN-40	Sistem mampu menyediakan data transaksi yang diperlukan untuk pembuatan nota atau faktur.	Berhasil
UFN-41	Sistem mampu mendukung pembayaran melalui MidTrans.	Berhasil
UFN-42	Sistem mampu menyediakan daftar lengkap semua pembeli yang terdaftar di sistem.	Berhasil
UFN-43	Sistem mampu memungkinkan pengguna untuk menambah data pembeli baru ke dalam sistem.	Berhasil
UFN-44	Sistem mampu menyediakan informasi detail mengenai setiap pembeli, termasuk nama, kontak, dan alamat.	Berhasil
UFN-45	Sistem mampu memungkinkan pengguna untuk mengedit informasi pembeli yang sudah ada di sistem.	Berhasil
UFN-46	Sistem mampu memungkinkan pengguna untuk menghapus pembeli dari sistem.	Berhasil
UFN-47	Sistem mampu memungkinkan pengguna untuk membuka kasir dengan mencatat nilai awal keuangan saat memulai *shift* kasir.	Berhasil
UFN-48	Sistem mampu memungkinkan pengguna untuk menutup kasir dengan mencatat nilai akhir keuangan saat mengakhiri *shift* kasir.	Berhasil
UFN-49	Sistem mampu menyediakan daftar lengkap riwayat kasir, termasuk waktu pembukaan dan penutupan serta nilai awal dan akhir.	Berhasil
UFN-50	Sistem mampu menyediakan informasi detail mengenai setiap sesi kasir.	Berhasil
UFN-51	Sistem mampu memungkinkan pengguna untuk menambah data *preorder*	Berhasil

Semua hasil pengujian fungsional, baik 51 uji positif maupun 51 uji negatif yang

dilakukan penulis berhasil. Pengujian ini mencakup berbagai skenario untuk memastikan sistem berfungsi sesuai dengan spesifikasi. Pengujian positif memastikan fitur berjalan dengan benar ketika diberikan *input* yang valid, sedangkan pengujian negatif memastikan sistem dapat menangani *input* yang tidak valid dengan tepat. Hasilnya menunjukkan bahwa sistem beroperasi dengan baik dalam semua kondisi yang diuji. Dokumentasi lebih lanjut mengenai setiap pengujian dapat dilihat di lampiran.

Setelah dilakukan pengujian oleh penulis, dilakukan pengujian oleh *quality assurance.* Pengujian yang dilakukan oleh penguji pertama, Sidrotul Munawaroh, berfokus pada pengujian untuk pihak toko bahan baku halal yang akan menangani transaksi. Sidrotul memastikan bahwa setiap fitur dan fungsi yang berkaitan dengan proses penjualan dan manajemen transaksi di toko dapat berjalan dengan baik dan efisien. Sedangkan penguji kedua, Agnesfia Anggraeni, berfokus pada pengujian untuk pembeli. Agnesfia memastikan bahwa pengalaman pengguna dari sisi pembeli, termasuk proses pemesanan, pembayaran, dan pengecekan pesanan, berjalan dengan lancar dan tanpa hambatan.

Tabel 4.5 Hasil Pengujian Fungsional oleh Quality Assurance

No	Nama	Jumlah Fungsionalitas	Keberhasilan	Keterangan
1	Sidrotul	50	100%	Pihak ITS Mart
2	Agnesfia	10	100%	Pembeli

Pada Tabel 4.5, Sidrotul menguji 50 fungsionalitas yang berkaitan dengan fungsionalitas untuk pihak toko bahan baku halal dan berhasil menguji seluruhnya dengan tingkat keberhasilan 100%. Agnesfia menguji 10 fungsionalitas dengan sudut pandang pembeli dan juga berhasil menguji seluruhnya dengan tingkat keberhasilan 100%. Kedua penguji berhasil memverifikasi fungsionalitas sistem sesuai dengan peran dan tanggung jawab masing-masing, memastikan kualitas sistem sebelum diimplementasikan sepenuhnya.

4.4.2 Pengujian Non Fungsionalitas Sistem

Pengujian non-fungsional merupakan tahap penting dalam memastikan bahwa sistem tidak hanya berfungsi dengan baik secara teknis, tetapi juga memenuhi persyaratan kinerja, keamanan, dan kompatibilitas. Pengujian ini bertujuan untuk mengevaluasi aspek-aspek sistem yang tidak terkait langsung dengan fungsi spesifik, namun krusial untuk pengalaman pengguna dan keandalan sistem secara keseluruhan. Berikut ini adalah beberapa pengujian non fungsional yang telah dilakukan.

Tabel 4.6 Pengujian Non Fungsionalitas untuk Kompatibilitas Sistem

Sistem harus dapat diakses di berbagai perangkat yang terkoneksi internet	
ID	UNF-01
Skenario	Mengakses sistem menggunakan berbagai perangkat (laptop, *smartphone*) dan berbagai browser (Chrome, Firefox, Edge)
Kondisi awal	Sistem sudah di-*deploy* pada server yang dapat diakses melalui internet
Hasil yang diharapkan	Sistem dapat diakses dengan lancar dan data yang benar pada semua perangkat dan browser yang diuji
Langkah pengujian	1. Buka browser pada perangkat 2. Akses https://riset.its.ac.id/teratai-dev/api/v1/tobaku/produk

Sistem harus dapat diakses di berbagai perangkat yang terkoneksi internet	
Hasil yang diperoleh	Sistem dapat diakses di berbagai perangkat dan browser tanpa masalah dengan data yang sesuai
Hasil pengujian	Berhasil

Pada pengujian Tabel 4.6, sistem yang telah di-*deploy* berhasil menunjukkan kemampuan untuk diakses dengan lancar melalui berbagai perangkat yang terhubung ke internet, termasuk laptop dan *smartphone*. Hasil pengujian ini menegaskan bahwa sistem mampu beroperasi dengan baik dalam berbagai kondisi penggunaan, memenuhi harapan untuk akses yang lancar dan data yang konsisten di semua platform dan browser yang diuji.

Tabel 4.7 Pengujian Non Fungsionalitas untuk Keamanan Sistem

Sistem memastikan pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses mereka setelah melakukan login	
ID	UNF-02
Skenario	Melakukan *login* dengan akun yang memiliki hak akses berbeda dan mencoba mengakses fitur yang tidak sesuai dengan peran masing-masing
Kondisi awal	Pengguna dengan berbagai peran (administrator, karyawan, pelanggan) telah terdaftar dalam sistem
Hasil yang diharapkan	Sistem menolak akses ke fitur yang tidak diizinkan
Langkah pengujian	1. Memasukkan URL untuk melihat laporan (Hanya admin yang dapat melihat) 2. Pilih metode GET 3. Eksekusi pemanggilan API
Hasil yang diperoleh	Sistem mengembalikan *error* 403
Hasil pengujian	Berhasil

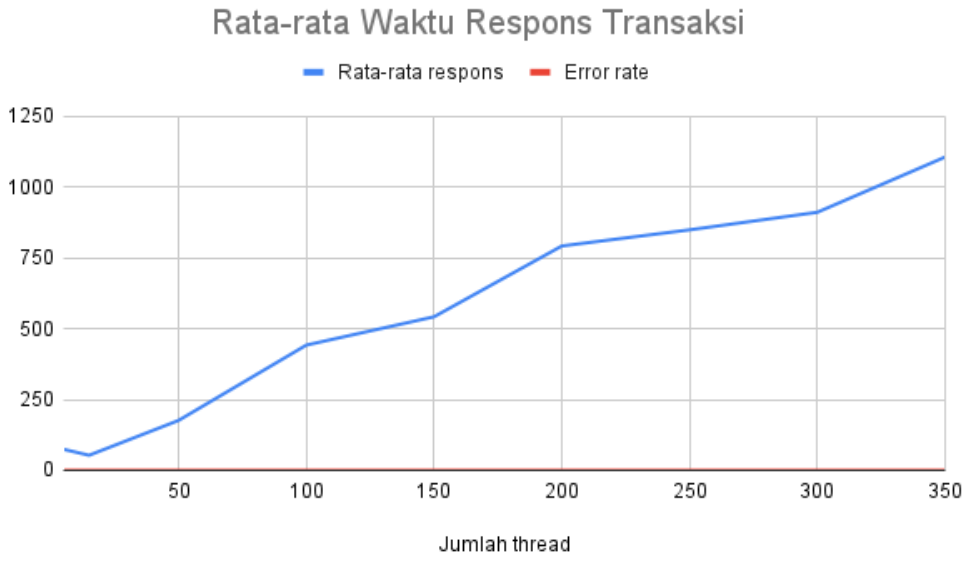
Pengujian pada Tabel 4.7 menguji kehandalan sistem dalam memastikan bahwa pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses mereka setelah melakukan *login*. Dalam kondisi pengujian, pengguna dengan berbagai peran seperti administrator, karyawan, dan pelanggan telah terdaftar dalam sistem. Langkah-langkah pengujian mencakup mencoba mengakses fitur yang tidak diizinkan dengan memasukkan URL tertentu yang hanya dapat diakses oleh admin, menggunakan metode GET, dan mengeksekusi pemanggilan API. Hasil yang diharapkan adalah apabila hak akses tidak sesuai maka sistem harus menolak akses dan mengembalikan *error* 403, yang menunjukkan bahwa pengujian berhasil memvalidasi kontrol akses yang tepat sesuai dengan peran pengguna.

Tabel 4.8 Pengujian Non Fungsionalitas untuk Performa Sistem

Sistem harus dapat menangani 15 transaksi per detik dengan waktu respon tidak lebih dari 1 detik	
ID	UNF-03

Sistem harus dapat menangani 15 transaksi per detik dengan waktu respon tidak lebih dari 1 detik	
Skenario	Mensimulasikan 15 transaksi per detik menggunakan alat uji beban seperti JMeter dan mencatat waktu respon.
Kondisi awal	Sistem dalam keadaan siap untuk menerima beban transaksi
Hasil yang diharapkan	Sistem dapat menangani beban tersebut dengan waktu respon tidak lebih dari 1 detik per transaksi
Langkah pengujian	1. Pilih alat uji (Misal JMeter) 2. Konfigurasi skrip uji 3. Pengaturan beban 4. Eksekusi pengujian
Hasil yang diperoleh	Sistem mampu menangani 15 transaksi per detik dengan waktu respon rata-rata di bawah 1 detik
Hasil pengujian	Berhasil

Pengujian pada Tabel 4.8 bertujuan untuk memastikan bahwa sistem mampu menangani beban transaksi dengan standar waktu respons yang telah ditentukan. Dalam kondisi awal pengujian, sistem sudah di-*deploy* pada *server* yang dapat diakses melalui internet. Tujuan dari pengujian ini adalah agar sistem dapat mengelola hingga 15 transaksi per detik dengan waktu respons tidak melebihi 1 detik per transaksi. Langkah-langkah pengujian meliputi pemilihan alat uji seperti JMeter, konfigurasi skrip uji, pengaturan beban yang dimaksudkan untuk mencapai beban maksimum yang diperlukan, dan akhirnya eksekusi pengujian. Hasil yang diperoleh menunjukkan bahwa sistem mampu menjalankan 15 transaksi per detik dengan waktu respons rata-rata di bawah 1 detik per transaksi yaitu 53 ms. Dengan demikian, pengujian ini dapat dikategorikan sebagai berhasil dalam memverifikasi kesiapan sistem untuk menangani beban transaksi yang tinggi. Dokumen lengkap mengenai pengujian ini dapat dilihat pada bagian lampiran.



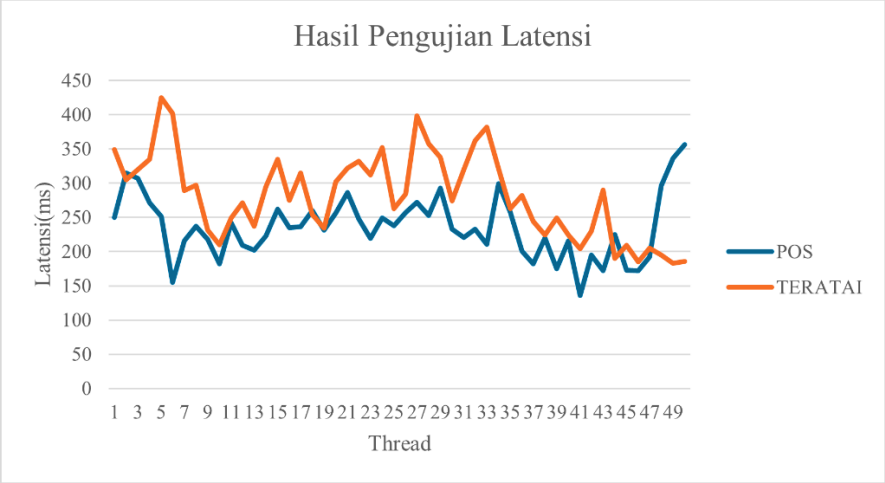
Gambar 4.4 Rata-rata waktu respons transaksi

Pada pengujian ini, dilakukan pengujian dengan jumlah *thread* atau pengguna secara bertahap hingga mencapai 350 *thread*. Grafik pada Gambar 4.4 menunjukkan rata-rata waktu respons transaksi yang diukur selama pengujian. Sumbu X menunjukkan jumlah *thread* atau pengguna yang sedang diuji, mulai dari 5 hingga 350 *thread*. Setiap titik pada sumbu ini menunjukkan tahap pengujian dengan jumlah *thread* tertentu. Sumbu Y menunjukkan rata-rata waktu respons dalam milidetik. Waktu respons adalah waktu rata-rata yang diperlukan oleh sistem untuk merespons permintaan dari pengguna.

Pada awal pengujian dengan jumlah *thread* yang relatif kecil(0 hingga 50 *thread*), rata-rata waktu respons berada di level yang rendah, sekitar 100 milidetik. Angka tersebut menunjukkan bahwa sistem dapat menangani beban kecil dengan cukup efisien. Ketika jumlah *thread* meningkat dari 50 hingga 150, rata-rata waktu respons menunjukkan peningkatan dengan rata-rata 386ms. Meskipun waktu respons meningkat, sistem masih dapat mengelola permintaan dengan cukup baik tanpa mengalami lonjakan besar dalam waktu respons. Tahap menengah dengan 150 hingga 200 *thread*, rata-rata waktu respons meningkat, mendekati angka 750 milidetik. Ini menunjukkan bahwa sistem mulai merasakan beban yang lebih berat, namun masih dalam batas toleransi. *Error rate* adalah metrik penting lainnya yang harus diperhatikan untuk memastikan bahwa peningkatan waktu respons bukan disebabkan oleh tingginya tingkat kegagalan dalam memproses permintaan. E*rror rate* pada pengujian ini adalah 0% pada setiap pengujian yang berarti tidak ada permintaan yang gagal.

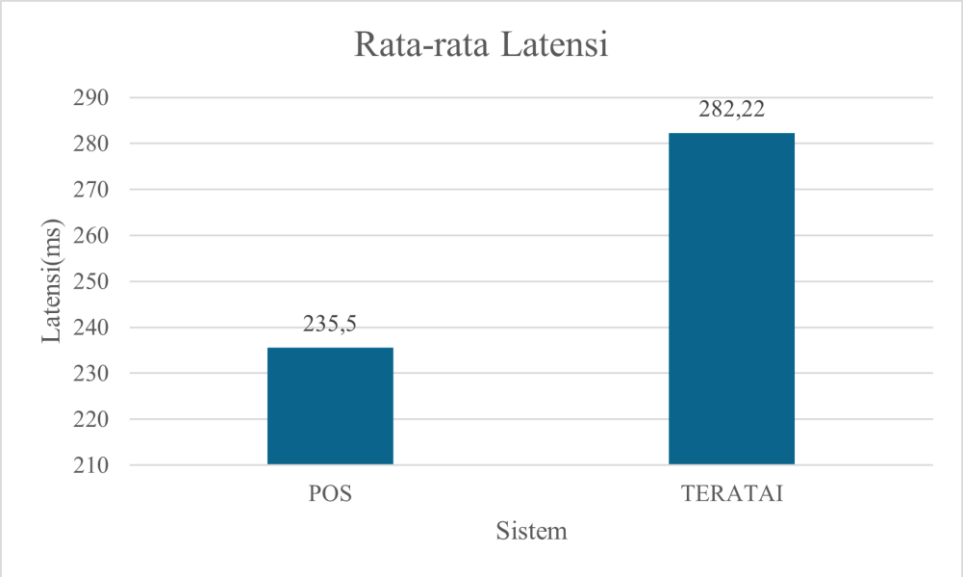
Selain pengujian non-fungsional yang dilakukan seperti pada Tabel 4.6, Tabel 4.7, dan Tabel 4.8, pengujian latensi juga dilakukan pada POS tobaku halal. Latensi adalah waktu yang dibutuhkan untuk data berpindah dari satu titik ke titik lain. Dalam konteks API, ini adalah waktu yang dibutuhkan untuk permintaan API berpindah dari klien ke server dan mengembalikan respons. Latensi API dapat diuji menggunakan JMeter. Dengan JMeter, dapat diukur waktu yang dibutuhkan untuk setiap permintaan API dari klien ke server dan kembali lagi, sehingga dapat mengidentifikasi dan menganalisis latensi untuk meningkatkan kinerja API.

Dalam skenario pengujian pada tugas akhir ini, JMeter akan digunakan untuk membandingkan bagaimana latensi pengambilan data produk yang sama apabila diambil langsung dari sistem Teratai dan diambil dari API POS yang terhubung dengan Teratai. Pengambilan dilakukan sebanyak 50 kali untuk mendapatkan latensi dan rata-ratanya akan dihitung. Pada Gambar 4.5 menunjukkan hasil pengujian latensi.



Gambar 4.5 Hasil Pengujian Latensi

Garis biru menunjukkan hasil pengujian dari pengambilan data melalui API POS, sedangkan garis oranye menunjukkan hasil pengujian dari pengambilan data langsung dari sistem Teratai. Hasil yang naik turun dapat disebabkan oleh berbagai faktor, seperti beban server, kondisi jaringan internet, dan tingkat konkurensi pengguna atau *thread* yang melakukan permintaan secara bersamaan. Berdasarkan hasil pada Gambar 4.5, dapat dihitung nilai rata-rata latensi apabila pengambilan data langsung dari Teratai maupun melalui POS seperti pada Gambar 4.6.



Gambar 4.6 Rata-rata Hasil Latensi

Grafik pada Gambar 4.6 menunjukkan perbandingan rata-rata latensi antara sistem POS dan TERATAI. Rata-rata latensi untuk POS adalah 235,5 milidetik, sedangkan untuk TERATAI adalah 282,22 milidetik. Ini menunjukkan bahwa POS memiliki waktu respon yang lebih cepat dibandingkan TERATAI, dengan selisih sekitar 46,72 milidetik. Dalam konteks ini, sistem dengan latensi yang lebih rendah (POS) lebih diunggulkan karena dapat memberikan respons yang lebih cepat terhadap permintaan yang diterima.

Sebagai tambahan, untuk memberikan gambaran yang lebih jelas, API dengan performa tinggi umumnya memiliki waktu respon rata-rata antara 0,1 hingga 1 detik. Pada latensi 2 detik, penundaan sudah mulai terasa. Jika latensi mencapai 5 detik, penundaan akan sangat signifikan sehingga pengguna mulai meninggalkan aplikasi atau situs *web* tersebut(Nikolov, 2024). Dengan rata-rata latensi di bawah 1 detik, baik POS maupun TERATAI masih dapat dianggap memiliki performa yang baik, meskipun POS memiliki keunggulan lebih dalam hal kecepatan respons.

4.2 Pembahasan

Pengujian fungsionalitas yang telah dilakukan bertujuan memastikan tingkah laku dan fungsionalitas seluruh API *endpoint* memenuhi seluruh kebutuhan fungsional aplikasi, serta memastikan bahwa setiap API *endpoint* berhasil menerima permintaan dan mengembalikan respons sesuai dengan fungsionalitas API *endpoint* tersebut. Pengujian dilakukan oleh tiga penguji, yaitu penulis sendiri untuk menguji semua fungsionalitas, Sidrotul Munawaroh sebagai sudut pandang toko bahan baku halal, dan Agnesfia Anggraeni sebagai sudut pandang pembeli. Dari total 51 fungsionalitas yang diujikan, menunjukkan bahwa 100% pemanggilan memenuhi

kebutuhan dengan mengembalikan respons yang sesuai kurang dari 1 detik dengan *status code* 200 maupun 201. *Status code* 200 (OK) menunjukkan bahwa permintaan telah berhasil diproses dan server mengembalikan data yang diminta. *Status code* 201 (*Created*) berarti bahwa permintaan telah berhasil diproses dan sumber daya baru telah dibuat sebagai hasil dari permintaan tersebut. *Status code* digunakan untuk permintaan POST, di mana klien mengirimkan data untuk membuat sumber daya baru, dan server berhasil membuat sumber daya tersebut. Keberhasilan ini menunjukkan bahwa sistem API yang dibangun telah dirancang dan diimplementasikan dengan baik, sehingga dapat digunakan untuk mendukung operasional aplikasi secara keseluruhan.

Selain pengujian fungsional, pengujian non fungsional juga telah dilakukan. Pengujian non fungsional ini mencakup beberapa aspek penting, termasuk aksesibilitas sistem di berbagai perangkat yang terkoneksi internet dengan hasil 100% dapat diakses melalui laptop dan smartphone dengan berbagai *browser*. Sistem juga menjamin bahwa pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses mereka setelah *login*. Pembeli 100% hanya dapat mengakses fitur yang sesuai dengan peran untuk melakukan pembelian dan karyawan 100% hanya dapat mengakses fitur yang yang sesuai untuk proses penjualan.

Toko bahan baku halal ITS Mart hanya menangani produk daging dengan satu kasir yang menangani transaksi langsung dan *preorder* daging. Dalam hal ini, sistem dirancang untuk menangani 15 transaksi bersama-sama per detik dengan rata-rata 53 ms, di mana waktu respons maksimum tidak melebihi 1 detik. Ini berarti bahwa sistem dapat dengan cepat memproses setiap transaksi dalam waktu kurang dari 0,2 detik. Jumlah ini cukup untuk toko bahan baku halal yang menangani penjualan hanya dengan 1 kasir dan *preorder* apabila daging tidak tersedia pada toko. Meskipun skenario operasional ini hanya melibatkan satu kasir, sistem ini dapat menangani hingga 300 *thread* secara bersamaan dengan waktu respons tetap di bawah 1 detik.

Pengembangan sistem POS dipisahkan dari sistem TERATAI untuk meningkatkan performa dan efisiensi operasional. Grafik pada Gambar 4.6 menunjukkan perbandingan rata-rata latensi antara sistem POS dan TERATAI. Rata-rata latensi untuk sistem POS adalah 235,5 milidetik, sedangkan untuk TERATAI adalah 282,22 milidetik. Selisih 46,72 milidetik mengindikasikan bahwa pemisahan sistem POS berhasil meningkatkan kecepatan respons. Latensi ideal untuk API adalah antara 0,1 hingga 1 detik, di mana latensi di bawah 1 detik menunjukkan performa yang baik. Dengan hasil ini, pemisahan sistem terbukti efektif dalam meningkatkan kinerja sistem POS.

Proses pengujian memastikan bahwa sistem tidak hanya berfungsi sesuai dengan kebutuhan fungsional, tetapi juga memenuhi kriteria performa dan keamanan yang telah ditetapkan. Dengan demikian, keseluruhan tahap pengembangan pada modul *backend* POS Tobaku Halal telah berjalan lancar dan sesuai dengan ekspektasi yang telah ditetapkan sebelumnya.

(Halaman ini sengaja dikosongkan)

BAB V
KESIMPULAN DAN SARAN

Bab ini merangkum hasil-hasil yang diperoleh dari tahapan analisis, perancangan, implementasi, dan pengujian dalam Tugas Akhir ini, serta memberikan saran-saran yang diharapkan bermanfaat untuk pengembangan sistem di masa mendatang.

5.1 Kesimpulan

Berdasarkan hasil pengembangan *backend* modul POS Tobaku Halal yang dilakukan pada penelitian ini didapatkan kesimpulan sebagai berikut.

1. Penerapan *Clean Architecture* dalam pengembangan modul *point of sale* (POS) pada aplikasi TERATAI berhasil dengan memisahkan setiap lapisan sesuai dengan tanggung jawabnya. Lapisan *domain* berisi serangkaian struktur dan fungsi yang berhubungan dengan pengelolaan data pada POS tobaku halal. Lapisan *application* menangani logika aplikasi POS serta aliran data antara domain dan antarmuka pengguna. Lapisan *interfaces* mengatur interaksi antara pengguna dan sistem POS melalui API *endpoints*. Lapisan *infrastructures* menghubungkan dengan sistem eksternal seperti basis data penjualan dan layanan pembayaran. Pendekatan ini memastikan struktur yang jelas dan pemeliharaan kode yang efisien.
2. Rancangan dan implementasi modul POS untuk menangani proses penjualan produk bahan baku halal di ITS Mart dilakukan dengan memperhatikan kebutuhan ITS Mart. Proses pengembangan dimulai dengan wawancara untuk pengumpulan kebutuhan, yang kemudian dianalisis untuk merumuskan kebutuhan fungsional dan non fungsional. Setelah analisis kebutuhan selesai, langkah berikutnya adalah desain *database* serta penentuan *endpoint* API untuk memfasilitasi pengembangan *frontend* oleh tim POS Tobaku Halal. Implementasi modul POS dilakukan menggunakan *framework* ExpressJS dan MongoDB sebagai *database* untuk penyimpanan data. Selain itu, MinIO digunakan untuk menyimpan *file*, seperti gambar produk, mendukung manajemen *file* dalam sistem. Modul POS mendukung 15 transaksi per detik dengan rata-rata waktu respons 53 ms, lebih baik dibandingkan standar API tinggi yang berada dalam 0,1 hingga 1 detik.
3. Integrasi modul POS dengan aplikasi Teratai berhasil dilakukan melalui penggunaan API yang tersedia di Teratai. Pada Teratai, terdapat koleksi bahan yang merupakan kumpulan data produk master yang dikenai transaksi yang menyimpan informasi mengenai penjual dan pembeli. Dengan adanya koleksi bahan, baik penjual maupun pembeli memiliki akses terhadap informasi terkait transaksi serta dapat menggunakan data bahan sebagai bahan dari produk master yang dimiliki. Untuk memungkinkan integrasi POS Tobaku Halal dan TERATAI, POS Tobaku Halal mengambil data ID Halal dari data bahan untuk dimasukkan pada koleksi produk. Ketika terjadi transaksi penjualan, data produk yang dijual akan dikirimkan ke TERATAI sebagai bahan baru agar proses *traceability* produk halal tetap dapat terlacak dengan baik.
4. Pengujian pada *backend* modul POS Tobaku Halal digunakan untuk memverifikasi kinerja sistem agar memenuhi semua kebutuhan pengguna. Proses pengujian mencakup pengujian fungsional yang meliputi pengujian positif dan negatif, serta pengujian non fungsional. Pengujian fungsional dilakukan oleh penulis dan dua *quality assurance* sebagai sudut pandang pihak toko bahan baku halal dan pembeli. Dari hasil pengujian 100% pengujian sesuai dengan skenario yang telah ditetapkan. Pengujian non fungsional yang dilakukan mencakup aksesibilitas sistem di berbagai perangkat dan *browser*, keamanan sistem terkait otentikasi dan otorisasi, serta performa sistem dalam menangani 15 transaksi dengan waktu respons kurang dari 1 detik. Hasil menunjukkan

sistem dapat diakses pada berbagai perangkat dan *browser*. Selain itu, apabila seorang pengguna mencoba mengakses sumber daya yang tidak diizinkan, sistem mengembalikan pesan kesalahan yang sesuai untuk mencegah akses tidak sah. Pesan kesalahan yang digunakan dalam situasi ini adalah *Error* 403 *Forbidden*. Berkaitan dengan performa sistem, hasil pengujian menunjukkan bahwa sistem POS memenuhi 100% dari kriteria performa untuk menangani 15 transaksi secara bersamaan, dengan waktu respons 53 ms. Sistem POS juga mampu menangani hingga 300 *thread* secara bersamaan dengan waktu respons kurang dari 1 detik. Rata-rata latensi dari sistem POS 46,72 milidetik lebih cepat dari sistem TERATAI yang menunjukkan bahwa performa dari sistem POS lebih baik daripada TERATAI. Dengan demikian, dapat disimpulkan bahwa *backend* modul POS Tobaku Halal telah sesuai dengan kebutuhan awal penggunaan yang telah ditetapkan.

5.2 Saran

Sebagai bahan perbaikan dikemudian hari, saran yang dapat diterapkan berdasarkan hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut.

1. **Pengembangan modul POS yang generik.**
   Untuk meningkatkan fleksibilitas dan daya saing, pertimbangkan untuk mengembangkan modul POS agar dapat diimplementasikan tidak hanya untuk ITS Mart, tetapi juga untuk toko-toko bahan baku halal lainnya serta Rumah Potong Hewan atau Industri Toko Bahan Baku Halal. Pengembangan tersebut dapat dilakukan dengan merancang modul agar dapat disesuaikan dengan berbagai jenis bisnis dan kebutuhan operasional yang berbeda.

2. **Penambahan fitur rekomendasi produk sesuai preferensi pelanggan dengan memanfaatkan teknologi *machine learning*.**
   Memanfaatkan teknologi *machine learning* untuk memberikan rekomendasi produk dapat meningkatkan pengalaman pelanggan. Integrasi model *machine learning* untuk menganalisis pola pembelian pelanggan, preferensi produk, dan tren pasar dapat membantu dalam menyajikan rekomendasi produk yang lebih personal dan relevan. Dengan memanfaatkan data historis transaksi dan perilaku pengguna, modul POS dapat menjadi lebih proaktif dalam menawarkan produk yang sesuai dengan kebutuhan dan minat pelanggan serta meningkatkan kepuasan pelanggan secara keseluruhan.

DAFTAR PUSTAKA

Abramova, V., & Bernardino, J. (2013). NoSQL databases: MongoDB vs Cassandra. *ACM International Conference Proceeding Series*, 14–22. https://doi.org/10.1145/2494444.2494447

Banton, C. (2023). *Raw Materials: Definition, Accounting, and Direct vs. Indirect*. https://www.investopedia.com/terms/r/rawmaterials.asp

Chandra, A. (2023). *What is Sales & Why it is Important for Your Business?* https://www.brandloom.com/what-is-sales-importance-for-your-business

CM, S. (2019). *What Are The Benefits Of Using Express.Js For Backend Development*. https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/

Dyantoro, S. (2023). *Mahasiswa ITS Ciptakan Aplikasi Point of Sale and Marketing Bernama WarungKu*. https://tekno.tempo.co/read/1800927/mahasiswa-its-ciptakan-aplikasi-point-of-sale-and-marketing-bernama-warungku

Gallaba, K., Hanam, Q., Mesbah, A., & Beschastnikh, I. (2017). Refactoring asynchrony in JavaScript. *Proceedings - 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017*, 353–363. https://doi.org/10.1109/ICSME.2017.83

Indahts. (2023). *Bantu Memajukan Perekonomian Masyarakat, ITS Mart Resmi Beroperasi*. https://www.its.ac.id/news/2023/07/14/bantu-memajukan-perekonomian-masyarakat-its-mart-resmi-beroperasi/

Jadhav, G., & Gonsalves, F. (2020). Role of Node.js in Modern Web Application Development. *International Research Journal of Engineering and Technology (IRJET)*, *7*(6), 6145–6150. www.irjet.net

Jose, B., & Abraham, S. (2017). Exploring the merits of nosql: A study based on mongodb. *2017 International Conference on Networks and Advances in Computational Technologies, NetACT 2017, July 2017*, 266–271. https://doi.org/10.1109/NETACT.2017.8076778

Khattak, J. Z. K., Mir, A., Anwar, Z., Wahedi, H. M., Abbas, G., Khattak, H. Z. K., & Ismatullah, H. (2011). Concept of Halal food and biotechnology. *Advance Journal of Food Science and Technology*, *3*(5), 385–389.

Martin, R. C. (2018). *Clean Architecture: A CRAFTSMAN'S GUIDE TO SOFTWARE STRUCTURE AND DESIGN*.

Mikkonen, T., & Taivalsaari, A. (2007). Using Javascript as a real programming language. *Network*, 17. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.7304&rep=rep1&type=pdf

Mohd Saifudin, A., Othman, S. N., & Elias, E. M. (2017). Exploring in Setting a Model for Islamic Supply Chain in Malaysia. *International Review of Management and Marketing*, *7*(1), 95–102.

MongoDB. (2024a). *Advantages of MongoDB*. https://www.mongodb.com/resources/compare/advantages-of-mongodb

MongoDB. (2024b). *MongoDB Manual - Geospatial Queries*. https://www.mongodb.com/docs/v4.2/geospatial-queries/

Nadha, C. (2022). *The Urgency of Listing Non-Critical Ingredients in the Halal Certification Process*. https://halalmui.org/en/urgency-list-of-non-critical-ingredients-in-process-halal-certification/

Neville, M. (2023). *The Advantages and Disadvantages of JavaScript*. https://softjourn.com/insights/the-advantages-and-disadvantages-of-javascript

Nguyen, T. (2023). *Express JS vs Django: A Comprehensive Comparison of Two Leading Web Frameworks*. https://www.frontendmag.com/insights/express-js-vs-django/

Nikolov, L. (2024). *What's the difference between API Latency and API Response Time?* https://blog.sentry.io/whats-the-difference-between-api-latency-and-api-response-time/#:~:text=Just to have a number,to abandon the application%2Fwebsite.

Ntantogian, C., Bountakas, P., & Antonaropoulos, D. (2021). Journal of Information Security and Applications NodeXP : NOde . js server-side JavaScript injection vulnerability DEtection and eXPloitation. *Journal of Information Security and Applications*, *58*(January), 102752. https://doi.org/10.1016/j.jisa.2021.102752

Safitri, A., Fahmi, M. Z., & Gunawan, S. (2022). Kajian Penelusuran Produk Halal Kornet Daging Sapi. *Halal Research Journal*, *2*(2), 64–76. https://doi.org/10.12962/j22759970.vi.194

Tan, K. H., Ali, M. H., Makhbul, Z. M., & Ismail, A. (2017). The impact of external integration on halal food integrity. *Supply Chain Management*, *22*(2), 186–199. https://doi.org/10.1108/SCM-05-2016-0171

Vanany, I., Rakhmawati, N. A., Sukoso, S., & Soon, J. M. (2020). Indonesian Halal Food Integrity: Blockchain Platform. *CENIM 2020 - Proceeding: International Conference on Computer Engineering, Network, and Intelligent Multimedia 2020*, 297–302. https://doi.org/10.1109/CENIM51130.2020.9297968

Zhang, J., & Bhatt, T. (2014). A Guidance Document on the Best Practices in Food Traceability. *Comprehensive Reviews in Food Science and Food Safety*, *13*(5), 1074–1103. https://doi.org/10.1111/1541-4337.12103

LAMPIRAN-LAMPIRAN

LAMPIRAN A. DOKUMENTASI PENGUJIAN FUNGSIONAL POSITIF

Lampiran 1. Dokumentasi Pengujian Fungsional Positif

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFP-01	Mengakses API endpoint untuk menyediakan laporan laba dan rugi untuk periode tertentu	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Sistem mengembalikan respon data laporan laba dan rugi untuk periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan laba dan rugi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data laporan laba dan rugi pada periode yang sesuai dengan status code 200	Berhasil
UFP-02	Mengakses API endpoint untuk menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Sistem mengembalikan respon data laporan semua transaksi untuk periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan transaksi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data laporan transaksi pada periode yang sesuai dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFP-03	Mengakses API endpoint untuk menyediakan laporan penjualan untuk setiap produk yang dijual	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Sistem mengembalikan respon data laporan penjualan untuk setiap produk yang dijual	-	1. Masukkan URL endpoint API untuk menyediakan laporan penjualan untuk setiap produk yang dijual 2. Pilih metode GET 4. Eksekusi pemanggilan API	Data laporan penjualan untuk setiap produk yang dijual dengan status code 200	Berhasil
UFP-04	Mengakses API endpoint untuk menyediakan laporan pembelian yang dilakukan oleh perusahaan	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Sistem mengembalikan respon data laporan pembelian yang dilakukan oleh perusahaan	-	1. Masukkan URL endpoint API untuk menyediakan laporan pembelian yang dilakukan oleh perusahaan 2. Pilih metode GET 4. Eksekusi pemanggilan API	Data laporan pembelian yang dilakukan oleh perusahaan dengan status code 200	Berhasil
UFP-05	Mengakses API endpoint untuk menyediakan laporan penjualan dalam periode tertentu	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Sistem mengembalikan respon data laporan penjualan yang mencakup total penjualan dalam periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter	Data laporan penjualan yang mencakup total penjualan dalam periode tertentu dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
					4. Eksekusi pemanggilan API		
UFP-06	Mengakses API endpoint untuk menyediakan laporan excel laba dan rugi untuk periode tertentu	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Excel berisi file laporan excel laba dan rugi untuk periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan laba dan rugi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API dengan send and download	Excel berhasil didownload dan berisi file laporan excel laba dan rugi untuk periode tertentu	Berhasil
UFP-07	Mengakses API endpoint untuk menyediakan laporan excel yang mendetail semua transaksi yang terjadi dalam periode tertentu	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Excel berisi file laporan excel yang mendetail semua transaksi yang terjadi dalam periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan excel yang mendetail semua transaksi yang terjadi dalam periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API dengan send and download	Excel berhasil didownload dan berisi file laporan excel yang mendetail semua transaksi yang terjadi dalam periode tertentu	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFP-08	Mengakses API endpoint untuk menyediakan laporan excel penjualan berdasarkan setiap produk yang dijual	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Excel berisi file laporan excel penjualan berdasarkan setiap produk yang dijual	-	1. Masukkan URL endpoint API untuk menyediakan laporan excel penjualan berdasarkan setiap produk yang dijual 2. Pilih metode GET 3. Eksekusi pemanggilan API dengan send and download	Excel berhasil didownload dan berisi file laporan excel penjualan berdasarkan setiap produk yang dijual	Berhasil
UFP-09	Mengakses API endpoint untuk menyediakan laporan excel pembelian yang dilakukan oleh perusahaan	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Excel berisi file laporan excel pembelian yang dilakukan oleh perusahaan	-	1. Masukkan URL endpoint API untuk menyediakan laporan excel pembelian yang dilakukan oleh perusahaan 2. Pilih metode GET 3. Eksekusi pemanggilan API dengan send and download	Excel berhasil didownload dan berisi file laporan excel pembelian yang dilakukan oleh perusahaan	Berhasil
UFP-10	Mengakses API endpoint untuk menyediakan laporan excel penjualan yang mencakup total	Pengguna sudah memasukkan token sebagai administrator Tobaku Halal	Excel berisi file laporan excel penjualan yang mencakup total penjualan dalam periode tertentu	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan excel penjualan yang mencakup total penjualan dalam	Excel berhasil didownload dan berisi file laporan penjualan yang mencakup total	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	penjualan dalam periode tertentu				periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API dengan send and download	penjualan dalam periode tertentu	
UFP-11	Mengakses API endpoint untuk menyediakan daftar lengkap semua produk yang tersedia di sistem	Pengguna sudah memasukkan token login	Sistem mengembalikan respon data daftar semua produk yang tersedia di sistem dan status code 200	-	1. Masukkan URL endpoint API untuk mendapatkan daftar semua produk yang tersedia di sistem dan status code 200 2. Pilih metode POST 3. Eksekusi pemanggilan API	Data daftar semua produk yang tersedia di sistem dan status code 200	Berhasil
UFP-12	Mengakses API endpoint untuk menambah data produk baru ke dalam sistem POS	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data produk yang berhasil ditambahkan produk master dengan status code 201	1. idHalal 2. gambarProduk 3. stok 4. idDiskon 5. array varian	1. Masukkan URL endpoint API untuk menambah produk 2. Pilih metode POST 3. Masukkan data uji 4. Eksekusi pemanggilan API	Data produk dengan status code 201	Berhasil
UFP-13	Mengakses API endpoint untuk edit produk yang ada di sistem	Pengguna sudah memasukkan token sebagai administrator	Sistem mengembalikan respon data produk dengan stok yang	1. kuantitas 2. keterangan	1. Masukkan URL endpoint API untuk menambah stok 2. Pilih metode POST	Data produk dengan stok yang sudah ditambah	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		perusahaan atau karyawan perusahaan	sudah ditambah dengan status code 200		3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	dengan status code 200	
UFP-14	Mengakses API endpoint untuk mengedit informasi produk yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data produk yang sudah diedit dengan status code 200	namaProduk	1. Masukkan URL endpoint API untuk menambah diskon 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data produk yang sudah diedit dengan *status code* 200	Berhasil
UFP-15	Mengakses API endpoint untuk menghapus produk dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data produk yang berhasil di hapus dengan status code 200	idProduk	1. Masukkan URL endpoint API untuk menghapus produk dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data yang berhasil dihapus dan *status code* 200	Berhasil
UFP-16	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap	Pengguna sudah memasukkan token sebagai administrator perusahaan atau	Sistem mengembalikan respon data detail mengenai setiap	idProduk	1. Masukkan URL endpoint API untuk melihat detail produk 2. Pilih metode POST 3. Masukkan data uji	Data detail mengenai setiap produk dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia	karyawan perusahaan	produk dengan *status code* 200		sebagai parameter 4. Eksekusi pemanggilan API		
UFP-17	Mengakses API endpoint untuk menampilkan daftar lengkap semua varian dari suatu produk yang tersedia di sistem	Pengguna sudah memasukkan token untuk login	Sistem mengembalikan respon data daftar lengkap semua varian dari suatu produk yang tersedia di sistem	idProduk	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua varian dari suatu produk yang tersedia di sistem 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data daftar lengkap semua varian dari suatu produk yang tersedia di sistem dan status 200	Berhasil
UFP-18	Mengakses API endpoint untuk menambah data varian baru dari suatu produk ke dalam sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data produk dengan variasi yang berhasil ditambahkan dengan status code 200	1. namaVarian 2. hargaJual 3. minPembelian	1. Masukkan URL endpoint API untuk menambah data varian baru dari suatu produk ke dalam sistem 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data produk dengan varian baru dari suatu produk ke dalam sistem dengan status code 200	Berhasil
UFP-19	Mengakses API endpoint untuk	Pengguna sudah memasukkan token sebagai	Sistem mengembalikan respon data produk	1. namaVarian 2. hargaJual 3. hargaBeli	1. Masukkan URL endpoint API untuk mengedit data varian	Data produk dengan varian yang berhasil di	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	mengedit data varian dari suatu produk	administrator perusahaan atau karyawan perusahaan	dengan variasi yang berhasil ditambahkan dengan status code 200		dari suatu produk 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	edit dari suatu produk dengan status code 200	
UFP-20	Mengakses API endpoint untuk menghapus data varian baru dari suatu produk ke dalam sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon status code 200	idProduk dan idVarian	1. Masukkan URL endpoint API untuk menambah data varian baru dari suatu produk ke dalam sistem 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API	Status code 200	Berhasil
UFP-21	Mengakses API endpoint untuk menampilkan detail varian dari suatu produk yang tersedia di sistem	Pengguna sudah memasukkan token untuk login	Sistem mengembalikan respon data varian dari suatu produk yang tersedia di sistem dengan status code 200	idProduk	1. Masukkan URL endpoint API untuk mendapatkan varian dari suatu produk yang tersedia di sistem 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data varian dari suatu produk yang tersedia di sistem dan status 200	Berhasil
UFP-22	Mengakses API endpoint untuk	Pengguna sudah memasukkan	Terdapat riwayat baru dari produk	1. kuantitas 2. keterangan	1. Masukkan URL endpoint API untuk	Terdapat riwayat baru dari produk	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	lakukan pengedit pada produk dan melihat riwatar riwayat baru	token sebagai administrator perusahaan atau karyawan perusahaan	setelah melakukan pengeditan		lakukan pengeditan pada produk 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API 5. Melihat riwayat baru produk dengan mengakses API untuk melihat riwayat produk dengan parameter idProduk	setelah melakukan pengeditan	
UFP-23	Mengakses API endpoint untuk menampilkan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem	Pengguna sudah memasukkan token untuk login sebagai administrator atau karyawan perusahaan	Sistem mengembalikan respon data daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Data daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem dan status code 200	Berhasil
UFP-24	Mengakses API endpoint untuk menampilkan informasi detail mengenai setiap riwayat produk	Pengguna sudah memasukkan token untuk login sebagai administrator atau karyawan perusahaan	Sistem mengembalikan respon data informasi detail mengenai setiap riwayat produk	idRiwayatProduk	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap riwayat produk 2. Pilih metode GET	Data informasi detail mengenai setiap riwayat produk dan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
					3. Eksekusi pemanggilan API		
UFP-25	Mengakses API endpoint untuk mendapatkan daftar lengkap semua diskon yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data daftar lengkap semua diskon yang tersedia di sistem dengan status code 200	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua diskon yang tersedia di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Data daftar lengkap semua diskon yang tersedia di sistem dengan status code 200	Berhasil
UFP-26	Mengakses API endpoint untuk menambahkan diskon	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data diskon yang berhasil ditambahkan produk master dengan status code 200	1. namaDiskon 2. deskripsiDiskon 3. besarDiskon 4. banyakDiskon	1. Masukkan URL endpoint API untuk menambah diskon 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data diskon dengan status code 201	Berhasil
UFP-27	Mengakses API endpoint untuk mendapatkan detail diskon yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data detail diskon yang tersedia di sistem dengan status code 200	idDiskon	1. Masukkan URL endpoint API untuk mendapatkan detail diskon yang tersedia di sistem 2. Pilih metode GET 3. Tambahkan data uji sebagai parameter	Data detail diskon yang tersedia di sistem dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
					4. Eksekusi pemanggilan API		
UFP-28	Mengakses API endpoint untuk menghapus diskon dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan response data diskon yang dihapus dengan status code 200	idDiskon	1. Masukkan URL endpoint API untuk menghapus diskon dari sistem 2. Pilih metode DELETE 3. Masukkan data uji coba sebagai parameter 3. Eksekusi pemanggilan API	Data diskon yamg dihapus dengan status code 200	Berhasil
UFP-29	Mengakses API endpoint untuk mengedit informasi diskon yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data diskon yang berhasil diperbarui produk master dengan status code 200	1. deskripsiDiskon 2. besarDiskon	1. Masukkan URL endpoint API untuk mengedit informasi diskon yang sudah ada di sistem 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data diskon yang berhasil diperbarui dengan status code 200	Berhasil
UFP-30	Mengakses API endpoint untuk menambahkan pemasok	Pengguna sudah memasukkan token sebagai administrator perusahaan atau	Sistem mengembalikan respon data pemasok yang berhasil ditambahkan dengan status code 201	1. nama 2. kontak 3. alamat	1. Masukkan URL endpoint API untuk menambah pemasok 2. Pilih metode POST 3. Masukkan data uji sebagai body	Data pemasok dengan status code 201	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		karyawan perusahaan			4. Eksekusi pemanggilan API		
UFP-31	Mengakses API endpoint untuk mendapatkan daftar lengkap semua pemasok yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data daftar lengkap semua pemasok yang tersedia di sistem dengan status code 200	-	1. Masukkan URL endpoint API untuk daftar lengkap semua pemasok yang tersedia di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Data daftar lengkap semua pemasok yang tersedia di sistem dengan status code 200	Berhasil
UFP-32	Mengakses API endpoint untuk mendapatkan detail pemasok yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data detail pemasok yang tersedia di sistem dengan status code 200	idPemasok	1. Masukkan URL endpoint API untuk daftar lengkap semua pemasok yang tersedia di sistem 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data detail pemasok yang tersedia di sistem dengan status code 200	Berhasil
UFP-33	Mengakses API endpoint untuk mengedit informasi pemasok yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data pemasok yang sudah diperbarui dengan status code 200	idPemasok	1. Masukkan URL endpoint API untuk mengedit informasi pemasok yang sudah ada di sistem 2. Pilih metode PATCH 3. Masukkan data uji	Data pemasok yang sudah diperbarui dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
					sebagai parameter 4. Eksekusi pemanggilan API		
UFP-34	Mengakses API endpoint untuk menghapus pemasok dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan data pemasok yang dihapus dengan status code 200	idPemasok	1. Masukkan URL endpoint API untuk menghapus pemasok dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data pemasok yang dihapus dengan status code 200	Berhasil
UFP-35	Mengakses API endpoint untuk mendapatkan daftar lengkap semua transaksi yang terjadi dalam sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data daftar lengkap semua transaksi yang terjadi dalam sistem dengan status code 200	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua transaksi yang terjadi dalam sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Data daftar lengkap semua transaksi yang terjadi dalam sistem dengan status code 200	Berhasil
UFP-36	Mengakses API endpoint untuk mendapatkan detail mengenai setiap transaksi	Pengguna sudah memasukkan token untuk login	Sistem mengembalikan respon data detail mengenai setiap transaksi dengan status code 200	idTransaksi	1. Masukkan URL endpoint API untuk mendapatkan detail mengenai setiap transaksi 2. Pilih metode GET	Data detail mengenai setiap transaksi dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
					3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API		
UFP-37	Mengakses API endpoint untuk menambahkan transaksi	Pengguna sudah memasukkan token untuk administrator atau karyawan perusahaan	Sistem mengembalikan respon data transaksi baru di dalam sistem dengan status code 201	1. jenisTransaksi 2. idTeratai 3. idKasir 4. metodePembayaran 5. idDiskon 6. detailTransaksi: {idProduk, jumlah, idVarian}	1. Masukkan URL endpoint API untuk menambah transaksi baru 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data transaksi baru di dalam sistem dengan status code 201	Berhasil
UFP-38	Mengakses API endpoint untuk mengubah status transaksi	Pengguna sudah memasukkan token sebagai karyawan atau administrator perusahaan	Sistem mengembalikan respon data transaksi yang berhasil diubah statusnya dengan status code 200	status	1. Masukkan URL endpoint API untuk mengubah status transaksi 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data transaksi yang berhasil diubah statusnya dengan status code 200	Berhasil
UFP-39	Mengakses API endpoint untuk memperbarui batch produk dalam transaksi	Pengguna sudah memasukkan token sebagai administrator perusahaan atau	Sistem mengembalikan respon data transaksi dengan status SELESAI dan batch	1. idKasir 2. detailTransaksi: {idDetailTransaksi, batch}	1. Masukkan URL endpoint API untuk memperbarui batch produk 2. Pilih metode PATCH	Data transaksi dengan status SELESAI dan batch yang	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		karyawan perusahaan	yang diperbarui dengan status code 200		3. Masukkan data uji 4. Eksekusi pemanggilan API	diperbarui dengan status code 200	
UFP-40	Mengakses API endpoint untuk membuat transaksi baru	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Pdf nota yang dapat didownload	idTransaksi	1. Masukkan URL endpoint API untuk membuat transaksi baru 2. Pilih metode POST 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Pdf nota yang dapat didownload	Berhasil
UFP-41	Mengakses API endpoint untuk membuat transaksi baru	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan data transaksi yang mencangkup data dari midtrans dengan status code 201		1. Masukkan URL endpoint API untuk membuat transaksi baru 2. Pilih metode POST 3. Masukkan data uji 4. Eksekusi pemanggilan API	Data transaksi yang mencangkup data dari midtrans dengan status code 201	Berhasil
UFP-42	Mengakses API endpoint untuk mendapatkan daftar lengkap semua pembeli yang terdaftar di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data daftar lengkap semua pembeli yang terdaftar di sistem dengan status code 200	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua pembeli yang terdaftar di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Data daftar lengkap semua pembeli yang terdaftar di sistem dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFP-43	Mengakses API endpoint untuk menambahkan data pembeli	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data pembeli dengan status code 201	1. nama 2. kontak 3. idTeratai 4. alamat	1. Masukkan URL endpoint API untuk menambah pembeli 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data pembeli dengan status code 201	Berhasil
UFP-44	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap pembeli	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data informasi detail mengenai setiap pembeli dengan status code 200	idpembeli	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap pembeli 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data informasi detail mengenai setiap pembeli dengan status code 200	Berhasil
UFP-45	Mengakses API endpoint untuk mengedit informasi pembeli yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data pembeli yang sudah diperbarui dengan status code 200	alamat	1. Masukkan URL endpoint API untuk mengedit data pembeli 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API	Data pembeli yang sudah diperbarui dengan status code 200	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFP-46	Mengakses API endpoint untuk menghapus pembeli dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan status code 200	idpembeli	1. Masukkan URL endpoint API untuk menghapus pembeli dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Status code 200	Berhasil
UFP-47	Mengakses API endpoint untuk membuka kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data kasir diawal dengan status code 201	1. uangAwal 2. idKaryawan	1. Masukkan URL endpoint API untuk membuka kasir 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data kasir diawal dengan status code 201	Berhasil
UFP-48	Mengakses API endpoint untuk menutup kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data kasir diakhir dengan status code 200	uangAkhir	1. Masukkan URL endpoint API untuk menutup kasir 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data kasir diakhir dengan status code 200	Berhasil
UFP-49	Mengakses API endpoint untuk	Pengguna sudah memasukkan	Sistem mengembalikan	-	1. Masukkan URL endpoint API untuk	Data daftar lengkap riwayat	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	mendapatkan daftar lengkap riwayat kasir	token sebagai administrator perusahaan atau karyawan perusahaan	respon data daftar lengkap riwayat kasir dengan status code 200		mendapatkan daftar lengkap riwayat kasir 2. Pilih metode GET 3. Eksekusi pemanggilan API	kasir dengan status code 200	
UFP-50	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap sesi kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan respon data informasi detail mengenai setiap sesi kasir dengan status code 200	idKasir	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap sesi kasir 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Data informasi detail mengenai setiap sesi kasir dengan status code 200	Berhasil
UFP-51	Mengakses API endpoint untuk menambahkan data preorder	Pengguna sudah memasukkan token untuk login	Sistem mengembalikan respon data informasi preorder dengan status code 201	1. jenisTransaksi 2. idTeratai 3. metodePembayaran 4. detailTransaksi: {idProduk, jumlah, idVarian}	1. Masukkan URL endpoint API untuk menambahkan data preorder 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Data informasi preorder dengan status code 201	Berhasil

LAMPIRAN B. DOKUMENTASI PENGUJIAN FUNGSIONAL NEGATIF

Lampiran 2. Dokumentasi Pengujian Fungsional Negatif

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFN-01	Mengakses API endpoint untuk menyediakan laporan laba dan rugi untuk periode tertentu	Pengguna sudah memasukkan token sebagai karyawan Tobaku Halal	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan laba dan rugi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	Berhasil
UFN-02	Mengakses API endpoint untuk menyediakan laporan yang mendetail semua transaksi yang terjadi dalam periode tertentu	Pengguna sudah memasukkan token sebagai karyawan Tobaku Halal	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan transaksi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	Berhasil
UFN-03	Mengakses API endpoint untuk menyediakan laporan penjualan untuk setiap produk yang dijual	Pengguna sudah memasukkan token sebagai karyawan Tobaku Halal	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	-	1. Masukkan URL endpoint API untuk menyediakan laporan penjualan untuk setiap produk yang dijual 2. Pilih metode GET 4. Eksekusi pemanggilan API	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	Berhasil
UFN-04	Mengakses API endpoint untuk	Pengguna sudah memasukkan	Sistem mengembalikan	-	1. Masukkan URL endpoint API untuk menyediakan	Sistem mengembalikan	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	menyediakan laporan pembelian yang dilakukan oleh perusahaan	token sebagai pembeli	error 403 dan pesan "Anda tidak memiliki akses admin"		laporan pembelian yang dilakukan oleh perusahaan 2. Pilih metode GET 4. Eksekusi pemanggilan API	error 403 dan pesan "Anda tidak memiliki akses admin"	
UFN-05	Mengakses API endpoint untuk menyediakan laporan penjualan dalam periode tertentu	Pengguna sudah memasukkan token sebagai pembeli	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan penjualan yang mencakup total penjualan dalam periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 403 dan pesan "Anda tidak memiliki akses admin"	Berhasil
UFN-06	Mengakses API endpoint untuk menyediakan laporan excel laba dan rugi untuk periode tertentu	Pengguna tidak memasukkan token apapun	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan laba dan rugi untuk periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API dengan send and download	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	Berhasil
UFN-07	Mengakses API endpoint untuk menyediakan laporan excel	Pengguna tidak memasukkan token apapun	Sistem mengembalikan error 403 dan	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan excel yang mendetail semua transaksi yang terjadi	Sistem mengembalikan error 403 dan	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	yang mendetail semua transaksi yang terjadi dalam periode tertentu		pesan "Token tidak valid"		dalam periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API dengan send and download	pesan "Token tidak valid"	
UFN-08	Mengakses API endpoint untuk menyediakan laporan excel penjualan berdasarkan setiap produk yang dijual	Pengguna tidak memasukkan token apapun	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	-	1. Masukkan URL endpoint API untuk menyediakan laporan excel penjualan berdasarkan setiap produk yang dijual 2. Pilih metode GET 3. Eksekusi pemanggilan API dengan send and download	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	Berhasil
UFN-09	Mengakses API endpoint untuk menyediakan laporan excel pembelian yang dilakukan oleh perusahaan	Pengguna sudah memasukkan token yang tidak valid	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	-	1. Masukkan URL endpoint API untuk menyediakan laporan excel pembelian yang dilakukan oleh perusahaan 2. Pilih metode GET 3. Eksekusi pemanggilan API dengan send and download	Sistem mengembalikan error 403 dan pesan "Token tidak valid"	Berhasil
UFN-10	Mengakses API endpoint untuk menyediakan laporan excel penjualan yang mencakup total	Belum ada transaksi yang dilakukan	Sistem mengembalikan error 404 dan pesan "Belum ada data transaksi"	1. startDate 2. endDate	1. Masukkan URL endpoint API untuk menyediakan laporan excel penjualan yang mencakup total penjualan dalam periode tertentu 2. Pilih metode GET 3. Masukkan data uji sebagai	Error 404 dan pesan "Belum ada data transaksi"	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	penjualan dalam periode tertentu				parameter 4. Eksekusi pemanggilan API dengan send and download		
UFN-11	Mengakses API endpoint untuk menyediakan daftar lengkap semua produk yang tersedia di sistem	Belum ada produk yang tesedia	Sistem mengembalikan error 404 dengan pesan "Data produk belum ada."	-	1. Masukkan URL endpoint API untuk mendapatkan daftar semua produk yang tersedia di sistem dan status code 200 2. Pilih metode POST 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data produk belum ada."	Berhasil
UFN-12	Mengakses API endpoint untuk menambah data produk baru ke dalam sistem POS dengan idHalal sudah ada di sistem	Sudah memasukkan token sebagai administrator atau karyawan tobaku idHalalsudah ada	Sistem mengembalikan error 400 dengan pesan "Produk sudah ada di sistem! Gunakan edit produk untuk mengedit."	1. idHalal 2. stok 3. hargaBeli 4. hargaJual	1. Masukkan URL endpoint API untuk menambah produk 2. Pilih metode POST 3. Masukkan data uji 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Produk sudah ada di sistem! Gunakan edit produk untuk mengedit."	Berhasil
UFN-13	Mengakses API endpoint untuk edit produk yang ada di sistem dan mencoba langsung mengubah stok	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan error 400 dengan pesan "Tidak boleh ubah stok secara langsung - Pakai kuantitas".	1. stok 2. keterangan	1. Masukkan URL endpoint API untuk menambah stok 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak boleh ubah stok secara langsung - Pakai kuantitas".	Berhasil
UFN-14	Mengakses API endpoint untuk	Pengguna sudah memasukkan	Sistem mengembalikan	idHalal	1. Masukkan URL endpoint API untuk menambah diskon	Error 400 dengan pesan "Tidak	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	mengedit informasi produk yang sudah ada di sistem	token sebagai administrator perusahaan atau karyawan perusahaan	error 400 dengan pesan "Tidak boleh ubah idHalal".		2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	boleh ubah idHalal".	
UFN-15	Mengakses API endpoint untuk menghapus produk dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Produk tidak tersedia	Sistem mengembalikan error 404 dengan pesan "Data produk tidak ada."	idProduk	1. Masukkan URL endpoint API untuk menghapus produk dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 404 dengan pesan "Data produk tidak ada."	Berhasil
UFN-16	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap produk, termasuk harga, ID Halal, dan jumlah stok yang tersedia	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Produk tidak tersedia	Sistem mengembalikan error 404 dengan pesan "Data produk tidak ada."	idProduk	1. Masukkan URL endpoint API untuk melihat detail produk 2. Pilih metode POST 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data produk tidak ada."	Berhasil
UFN-17	Mengakses API endpoint untuk menampilkan daftar lengkap semua varian	Pengguna sudah memasukkan token untuk login	Sistem mengembalikan error 404 dengan pesan "Data	idProduk	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua varian dari suatu produk yang tersedia di sistem	Error 404 dengan pesan "Data produk tidak ada."	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	dari suatu produk yang tersedia di sistem	Produk tidak memiliki varian	produk tidak ada."		2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API		
UFN-18	Mengakses API endpoint untuk menambah data varian baru dari suatu produk ke dalam sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. namaVarian	1. Masukkan URL endpoint API untuk menambah data varian baru dari suatu produk ke dalam sistem 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil
UFN-19	Mengakses API endpoint untuk mengedit data varian dari suatu produk	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak ada varian didalam produk itu	Sistem mengembalikan error 404 dengan pesan "Varian tidak ditemukan pada produk ini".	1. namaVarian 2. hargaJual 3. hargaBeli	1. Masukkan URL endpoint API untuk mengedit data varian dari suatu produk 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Varian tidak ditemukan pada produk ini".	Berhasil
UFN-20	Mengakses API endpoint untuk menghapus data varian baru dari	Pengguna sudah memasukkan token sebagai administrator perusahaan atau	Sistem mengembalikan error 404 dengan pesan "Varian		1. Masukkan URL endpoint API untuk menambah data varian baru dari suatu produk ke dalam sistem 2. Pilih metode PATCH	Error 404 dengan pesan "Varian tidak ditemukan pada produk ini".	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	suatu produk ke dalam sistem	karyawan perusahaan Tidak ada varian didalam produk itu	tidak ditemukan pada produk ini".		3. Masukkan data uji 4. Eksekusi pemanggilan API		
UFN-21	Mengakses API endpoint untuk menampilkan detail varian dari suatu produk yang tersedia di sistem	Pengguna sudah memasukkan token untuk login Tidak ada produk	Sistem mengembalikan error 404 dengan pesan "Data produk tidak ada".	1. idProduk 2. idVarian	1. Masukkan URL endpoint API untuk mendapatkan varian dari suatu produk yang tersedia di sistem 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data produk tidak ada".	Berhasil
UFN-22	Mengakses API endpoint untuk menambahkan riwayat	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Bagian produk tidak valid	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. kuantitas 2. keterangan	1. Masukkan URL endpoint API untuk menambah riwayat 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API 5. Melihat riwayat baru produk dengan mengakses API untuk melihat riwayat produk dengan parameter idProduk	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil
UFN-23	Mengakses API endpoint untuk menampilkan daftar lengkap semua riwayat perubahan dan	Pengguna sudah memasukkan token untuk login sebagai administrator atau karyawan	Sistem mengembalikan error 404 dengan pesan "Data riwayat produk belum ada".	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua riwayat perubahan dan aktivitas yang terkait dengan produk di sistem	Error 404 dengan pesan "Data riwayat produk belum ada".	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	aktivitas yang terkait dengan produk di sistem	perusahaan Tidak ada riwayat perubahan			2. Pilih metode GET 3. Eksekusi pemanggilan API		
UFN-24	Mengakses API endpoint untuk menampilkan informasi detail mengenai setiap riwayat produk	Pengguna sudah memasukkan token untuk login sebagai administrator atau karyawan perusahaan tidak id riwayat produk	Sistem mengembalikan error 404 dengan pesan "Data riwayat produk tidak ada".	idRiwayatProduk	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap riwayat produk 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data riwayat produk tidak ada".	Berhasil
UFN-25	Mengakses API endpoint untuk mendapatkan daftar lengkap semua diskon yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Belum ada data diskon	Sistem mengembalikan error 404 dengan pesan "Data diskon belum ada."	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua diskon yang tersedia di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data diskon belum ada."	Berhasil
UFN-26	Mengakses API endpoint untuk menambahkan diskon dengan	Pengguna sudah memasukkan token sebagai administrator perusahaan atau	Sistem mengembalikan error 400 dengan pesan "Tidak	1. namaDiskon 2. deskripsiDiskon	1. Masukkan URL endpoint API untuk menambah diskon 2. Pilih metode POST 3. Masukkan data uji sebagai	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	data uji coba tidak lengkap	karyawan perusahaan	memenuhi validasi"		body 4. Eksekusi pemanggilan API		
UFN-27	Mengakses API endpoint untuk mendapatkan detail diskon yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idDiskon tidak tersedia	Sistem mengembalikan error 404 dengan pesan "Data diskon tidak ada."	idDiskon	1. Masukkan URL endpoint API untuk mendapatkan detail diskon yang tersedia di sistem 2. Pilih metode GET 3. Tambahkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data diskon tidak ada."	Berhasil
UFN-28	Mengakses API endpoint untuk menghapus diskon dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idDiskon tidak tersedia	Sistem mengembalikan error 404 dengan pesan "Data diskon tidak ada."	idDiskon	1. Masukkan URL endpoint API untuk menghapus diskon dari sistem 2. Pilih metode DELETE 3. Masukkan data uji coba sebagai parameter 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data diskon tidak ada."	Berhasil
UFN-29	Mengakses API endpoint untuk mengedit informasi diskon yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan banyakDiskon	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. banyakDiskon	1. Masukkan URL endpoint API untuk mengedit informasi diskon yang sudah ada di sistem 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		berupa non integer					
UFN-30	Mengakses API endpoint untuk menambahkan pemasok dengan data yang tidak lengkap (tidak ada nama)	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. alamat 2. kontak	1. Masukkan URL endpoint API untuk menambah pemasok 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil
UFN-31	Mengakses API endpoint untuk mendapatkan daftar lengkap semua pemasok yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Belum ada data pemasok yang tersedia	Sistem mengembalikan error 404 dengan pesan "Data pemasok belum ada."	-	1. Masukkan URL endpoint API untuk daftar lengkap semua pemasok yang tersedia di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data pemasok belum ada."	Berhasil
UFN-32	Mengakses API endpoint untuk mendapatkan detail pemasok yang tersedia di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak ada	Sistem mengembalikan error 404 dengan pesan "Data pemasok tidak ada."	idPemasok	1. Masukkan URL endpoint API untuk daftar lengkap semua pemasok yang tersedia di sistem 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data pemasok tidak ada."	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		idPemasok yang dimaksud					
UFN-33	Mengakses API endpoint untuk mengedit informasi pemasok yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak ada idPemasok pada sistem	Sistem mengembalikan error 404 dengan pesan "Data pemasok tidak ada."	idPemasok	1. Masukkan URL endpoint API untuk mengedit informasi pemasok yang sudah ada di sistem 2. Pilih metode PATCH 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 404 dengan pesan "Data pemasok tidak ada."	Berhasil
UFN-34	Mengakses API endpoint untuk menghapus pemasok dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak ada idPemasok pada sistem	Sistem mengembalikan error 404 dengan pesan "Data pemasok tidak ada."	idPemasok	1. Masukkan URL endpoint API untuk menghapus pemasok dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	error 404 dengan pesan "Data pemasok tidak ada."	Berhasil
UFN-35	Mengakses API endpoint untuk mendapatkan daftar lengkap semua transaksi	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan	Sistem mengembalikan error 404 dengan pesan "Data transaksi belum ada."	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua transaksi yang terjadi dalam sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data transaksi belum ada."	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
	yang terjadi dalam sistem	perusahaan Belum ada data transaksi					
UFN-36	Mengakses API endpoint untuk mendapatkan detail mengenai setiap transaksi	Pengguna sudah memasukkan token untuk login Tidak ada data dengan idTransaksi yang dimaksud	Sistem mengembalikan error 404 dengan pesan "Data transaksi tidak ada."	idTransaksi	1. Masukkan URL endpoint API untuk mendapatkan detail mengenai setiap transaksi 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data transaksi tidak ada."	Berhasil
UFN-37	Mengakses API endpoint untuk menambahkan transaksi	Pengguna sudah memasukkan token untuk administrator atau karyawan perusahaan idKasir sudah tutup	Sistem mengembalikan error 400 dengan pesan "Sesi kasir sudah berakhir!"	1. jenisTransaksi 2. idTeratai 3. idKasir 4. metodePembayaran 5. idDiskon 6. detailTransaksi: {idProduk, jumlah, idVarian}	1. Masukkan URL endpoint API untuk menambah transaksi baru 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Sesi kasir sudah berakhir!"	Berhasil
UFN-38	Mengakses API endpoint untuk mengubah status transaksi	Pengguna sudah memasukkan token sebagai karyawan atau administrator perusahaan	Sistem mengembalikan error 400 dengan pesan "Status tidak diketahui atau tidak sesuai urutan".	status	1. Masukkan URL endpoint API untuk mengubah status transaksi 2. Pilih metode PATCH 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Status tidak diketahui atau tidak sesuai urutan".	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		Status tidak sesuai urutan					
UFN-39	Mengakses API endpoint untuk memperbarui batch produk dalam transaksi	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idTransaksi tidak ada di dalam sistem	Sistem mengembalikan error 404 dengan pesan "Data transaksi tidak ada."	1. idKasir 2. detailTransaksi: {idDetailTransaksi, batch}	1. Masukkan URL endpoint API untuk memperbarui batch produk 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data trasaksi tidak ada."	Berhasil
UFN-40	Mengakses API endpoint untuk membuat transaksi baru	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Transaksi belum SELESAI	Sistem mengembalikan error 400 dengan pesan "Transaksi belum selesai".	idTransaksi	1. Masukkan URL endpoint API untuk membuat transaksi baru 2. Pilih metode POST 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Transaksi belum selesai".	Berhasil
UFN-41	Mengakses API endpoint untuk membuat transaksi baru	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan	Sistem mengembalikan error 500 dengan pesan error dari midtrans	1. jenisTransaksi 2. idTeratai 3. metodePembayaran 4. detailTransaksi:	1. Masukkan URL endpoint API untuk membuat transaksi baru 2. Pilih metode POST 3. Masukkan data uji 4. Eksekusi pemanggilan API	Error 500 dengan pesan error dari midtrans	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		perusahaan Server key midtrans tidak valid Memilih jenisTransaksi Cashless		{idProduk, jumlah}			
UFN-42	Mengakses API endpoint untuk mendapatkan daftar lengkap semua pembeli yang terdaftar di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Belum ada daftar pembeli didalam sistem	Sistem mengembalikan error 404 dengan pesan "Data pembeli belum ada."	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap semua pembeli yang terdaftar di sistem 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data pembeli belum ada."	Berhasil
UFN-43	Mengakses API endpoint untuk menambahkan data pembeli dan data yang dimasukkan tidak mengandung id Teratai ataupun nama	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. kontak 2. alamat	1. Masukkan URL endpoint API untuk menambah pembeli 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil

ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFN-44	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap pembeli	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idPembeli tidak tersedia	Sistem mengembalikan error 404 dengan pesan "Data pembeli tidak ada."	idpembeli	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap pembeli 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data pembeli tidak ada."	Berhasil
UFN-45	Mengakses API endpoint untuk mengedit informasi pembeli yang sudah ada di sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idPembeli tidak tersedia pada sistem	Sistem mengembalikan error 404 dengan pesan "Data pembeli tidak ada."	alamat	1. Masukkan URL endpoint API untuk mengedit data pembeli 2. Pilih metode PATCH 3. Masukkan data uji 4. Eksekusi pemanggilan API	Sistem mengembalikan error 404 dengan pesan "Data pembeli tidak ada."	Berhasil
UFN-46	Mengakses API endpoint untuk menghapus pembeli dari sistem	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan idPembeli tidak tersedia pada sistem	Sistem mengembalikan error 404	idPembeli	1. Masukkan URL endpoint API untuk menghapus pembeli dari sistem 2. Pilih metode DELETE 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Sistem mengembalikan error 404	Berhasil

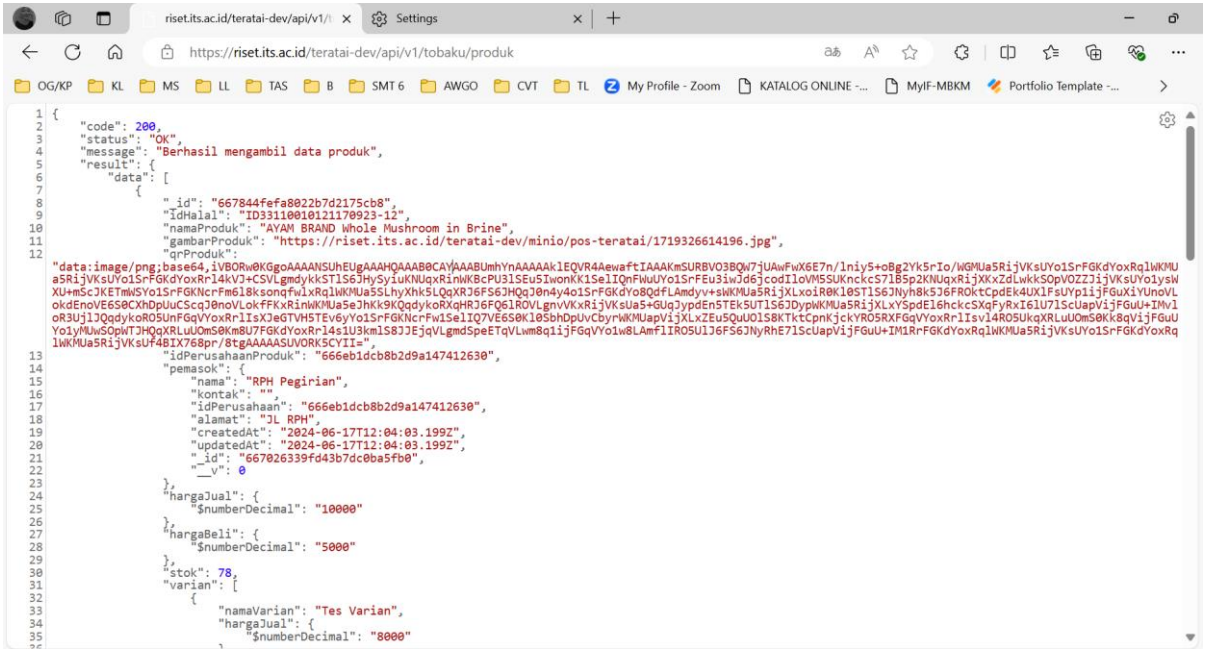
ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
UFN-47	Mengakses API endpoint untuk membuka kasir dengan tidak memasukkan data karyawan	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak memasukkan data karyawan	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	1. uangAwal	1. Masukkan URL endpoint API untuk membuka kasir 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil
UFN-48	Mengakses API endpoint untuk menutup kasir dengan tidak memasukkan uangAkhir pada kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak memasukkan uangAkhir	Sistem mengembalikan error 400 dengan pesan "Tidak memenuhi validasi"	-	1. Masukkan URL endpoint API untuk menutup kasir 2. Pilih metode PATCH 3. Eksekusi pemanggilan API	Error 400 dengan pesan "Tidak memenuhi validasi"	Berhasil
UFN-49	Mengakses API endpoint untuk mendapatkan daftar lengkap riwayat kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan	Sistem mengembalikan error 404 dengan pesan "Data riwayat kasir belum ada"	-	1. Masukkan URL endpoint API untuk mendapatkan daftar lengkap riwayat kasir 2. Pilih metode GET 3. Eksekusi pemanggilan API	Error 404 dengan pesan "Data riwayat kasir belum ada"	Berhasil

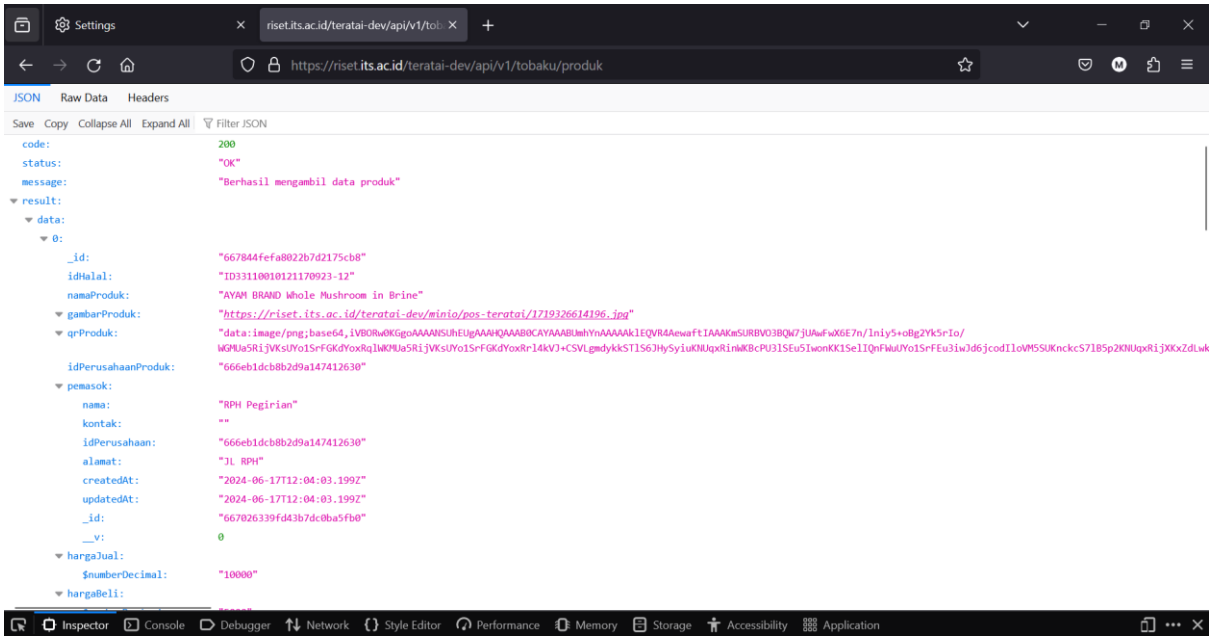
ID	Skenario	Kondisi Awal	Hasil yang Diharapkan	Data Uji Coba	Langkah Pengujian	Hasil yang DIperoleh	Hasil Pengujian
		Belum ada data kasir					
UFN-50	Mengakses API endpoint untuk mendapatkan informasi detail mengenai setiap sesi kasir	Pengguna sudah memasukkan token sebagai administrator perusahaan atau karyawan perusahaan Tidak ada idKasir pada sistem	Sistem mengembalikan error 404 dengan pesan "Data riwayat kasir tidak ada"	idKasir	1. Masukkan URL endpoint API untuk mendapatkan informasi detail mengenai setiap sesi kasir 2. Pilih metode GET 3. Masukkan data uji sebagai parameter 4. Eksekusi pemanggilan API	Error 404 dengan pesan "Data riwayat kasir tidak ada"	Berhasil
UFN-51	Mengakses API endpoint untuk menambahkan data preorder	Pengguna sudah memasukkan token untuk login Metode pembayaran cash	Sistem mengembalikan error 400 dengan pesan "Preorder tidak boleh menggunakan sistem pembayaran cash"	1. jenisTransaksi 2. idTeratai 3. metodePembayaran 4. detailTransaksi: {idProduk, jumlah}	1. Masukkan URL endpoint API untuk menambahkan data preorder 2. Pilih metode POST 3. Masukkan data uji sebagai body 4. Eksekusi pemanggilan API	Error 400 dengan pesan "Preorder tidak boleh menggunakan sistem pembayaran cash"	Berhasil

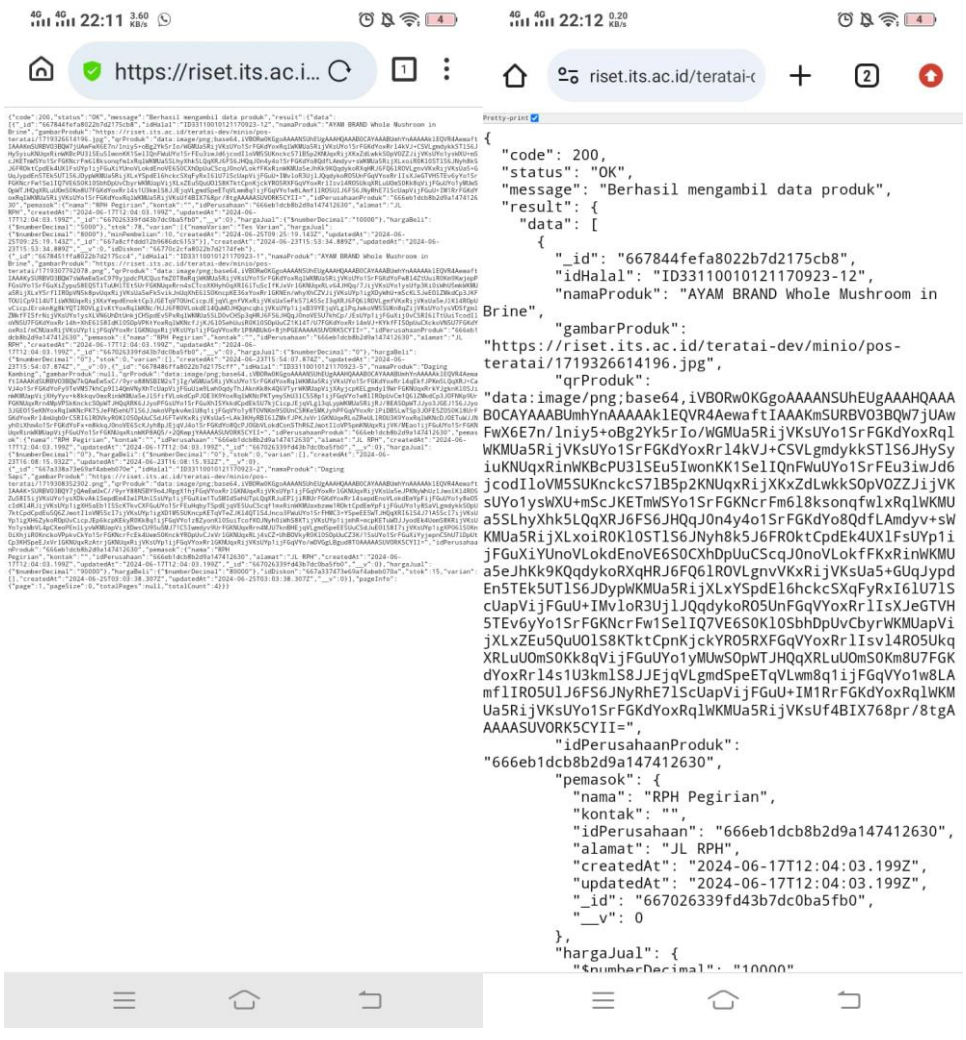
LAMPIRAN C. DOKUMENTASI PENGUJIAN NON-FUNGSIONAL

Lampiran 3. Tabel Pengujian Non Fungsional UNF-01

Sistem harus dapat diakses di berbagai perangkat yang terkoneksi internet	
ID	UNF-01
Skenario	Mengakses sistem menggunakan berbagai perangkat (desktop, laptop, tablet, smartphone) dan berbagai browser (Chrome, Firefox, Safari, Edge)
Kondisi awal	Sistem sudah di-deploy pada server yang dapat diakses melalui internet
Hasil yang diharapkan	Sistem dapat diakses dengan lancar dan tampil dengan benar pada semua perangkat dan browser yang diuji
Langkah pengujian	1. Buka browser pada perangkat 2. Akses https://riset.its.ac.id/teratai-dev/api/v1/tobaku/produk
Hasil yang diperoleh	Sistem dapat diakses di berbagai perangkat dan browser tanpa masalah tampilan
Hasil pengujian	Berhasil



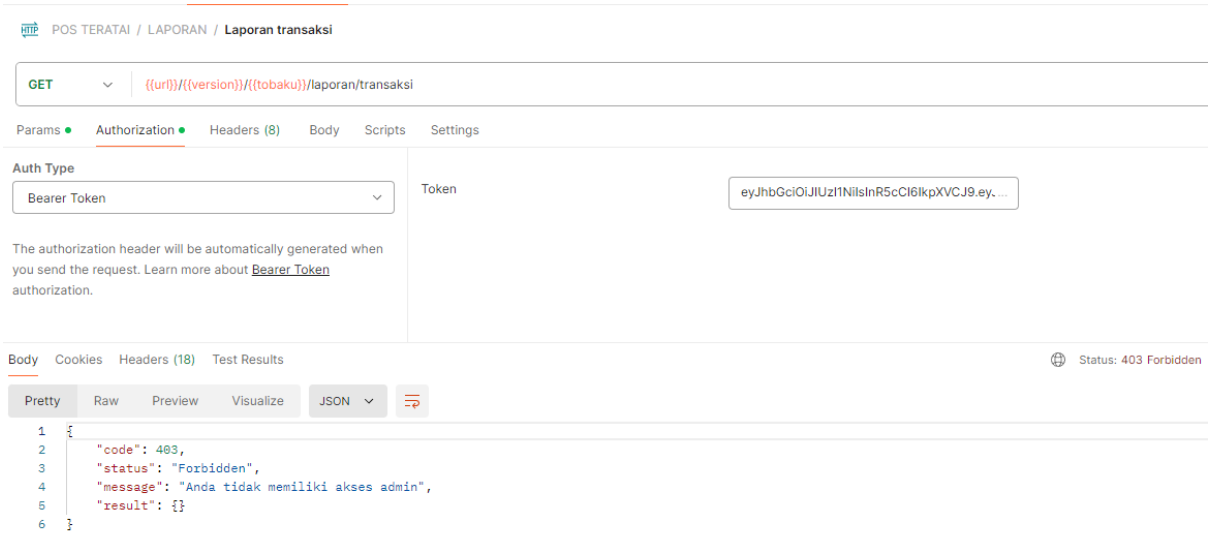




Lampiran 4. Dokumentasi Pengujian Non Fungsional UNF-01

Lampiran 5. Tabel Pengujian Non Fungsional UNF-02

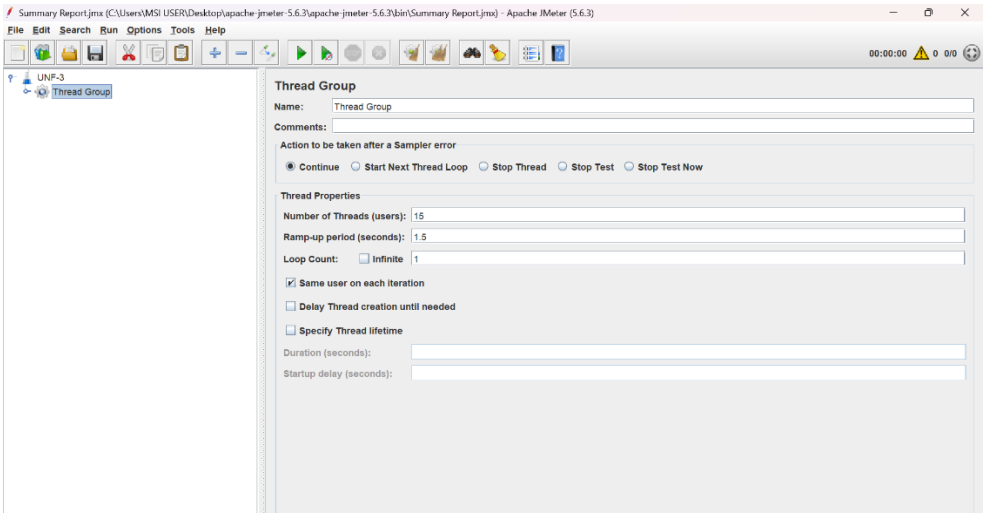
Sistem memastikan pengguna hanya dapat mengakses fitur yang sesuai dengan peran atau hak akses mereka setelah melakukan login	
ID	UNF-02
Skenario	Melakukan login dengan akun yang memiliki hak akses berbeda dan mencoba mengakses fitur yang tidak sesuai dengan peran masing-masing
Kondisi awal	Pengguna dengan berbagai peran (administrator, karyawan, pelanggan) telah terdaftar dalam sistem
Hasil yang diharapkan	Sistem menolak akses ke fitur yang tidak diizinkan
Langkah pengujian	1. Memasukkan URL untuk melihat laporan (Hanya admin yang dapat melihat) 2. Pilih metode GET 3. Eksekusi pemanggilan API
Hasil yang diperoleh	Sistem mengembalikan error 403
Hasil pengujian	Berhasil



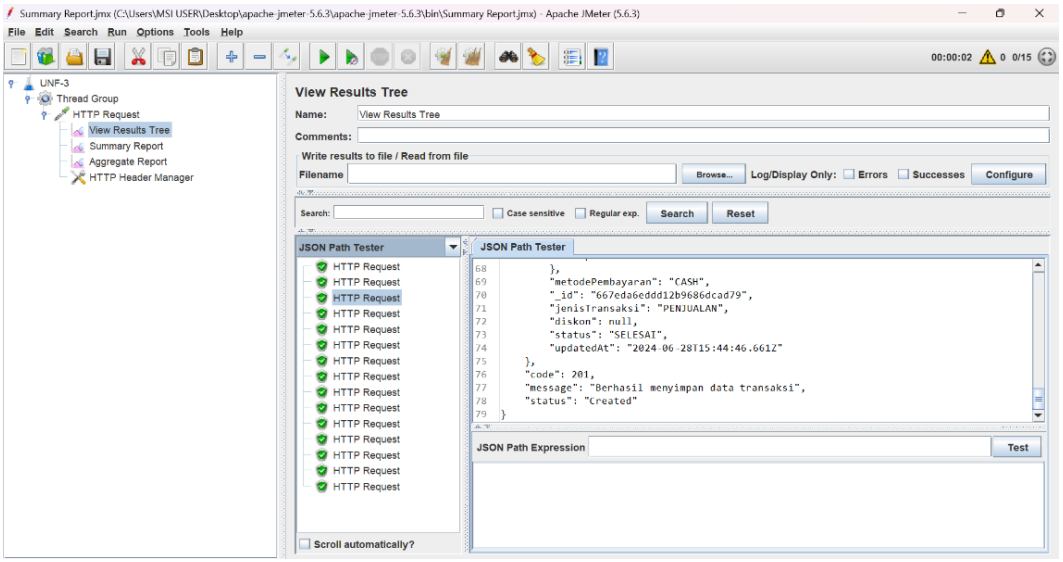
Lampiran 6. Dokumentasi Pengujian Non Fungsional UNF-02

Lampiran 7. Tabel Pengujian Non Fungsional UNF-03

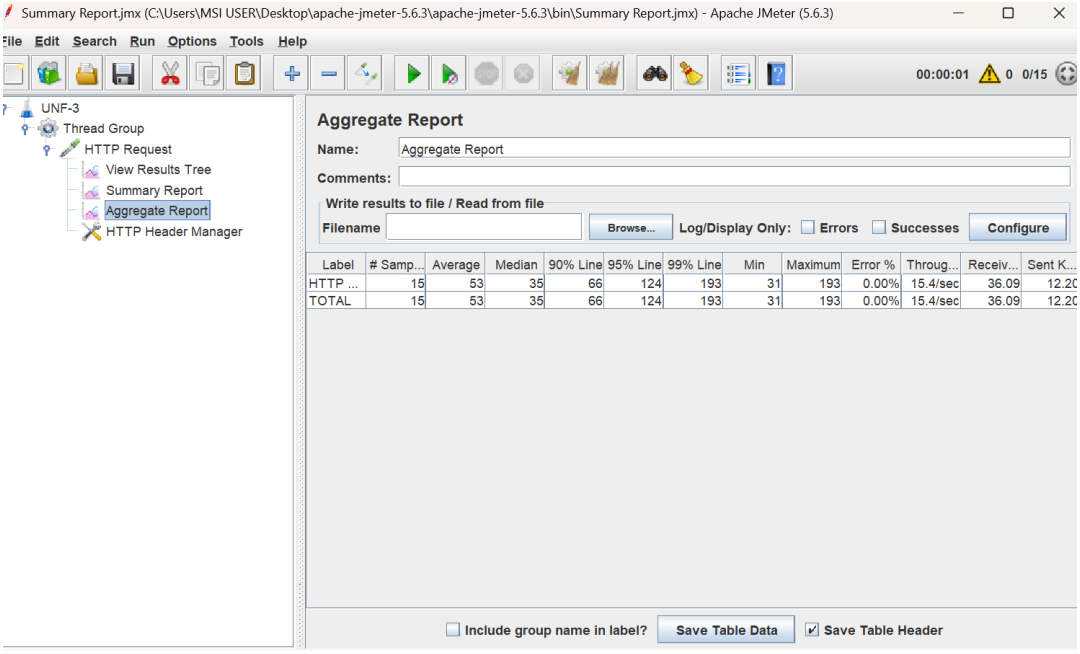
Sistem harus dapat menangani 15 transaksi per detik dengan waktu respon tidak lebih dari 1 detik	
ID	UNF-03
Skenario	Sistem dalam keadaan siap untuk menerima beban transaksi tinggi
Kondisi awal	Sistem sudah di-deploy pada server yang dapat diakses melalui internet
Hasil yang diharapkan	Sistem dapat menangani beban tersebut dengan waktu respon tidak lebih dari 1 detik per transaksi
Langkah pengujian	1. Pilih alat uji (Misal JMeter) 2. Konfigurasi skrip uji 3. Pengaturan beban 4. Eksekusi pengujian
Hasil yang diperoleh	Sistem mampu menangani 15 transaksi per detik dengan waktu respon rata-rata di bawah 1 detik
Hasil pengujian	Berhasil



Lampiran 8 Dokumentasi Pengujian Non Fungsional UNF-03



Lampiran 9 Dokumentasi Pengujian Non Fungsional UNF-03



Lampiran 10. Dokumentasi Pengujian Non Fungsional UNF-03

BIODATA PENULIS

Penulis dilahirkan di Temanggung, 05 Agustus 2001, merupakan anak kedua dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK ABA Murni 1 Kalisat, SDN 2 Mojotengah, SMPN 1 Parakan, dan SMA Semesta BBS Semarang. Setelah lulus dari SMA Semesta BBS Semarang, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2020 dan terdaftar dengan NRP 5025201244.

Selama menempuh pendidikan di perguruan tinggi, penulis aktif sebagai wakil ketua bidang Sosial Masyarakat Himpunan Mahasiswa Teknik Computer-Informatika (HTMC) 2023, sekretaris Schematics 2021 dan 2022, staf di Departemen Badan Semi Otonom dan Kreasi (BSO-K) BIMITS 2021, serta terlibat dalam beberapa kegiatan pelatihan di Koperasi Mahasiswa dr. Angka ITS. Selain itu, penulis juga ikut serta dalam pengembangan proyek *web Halal Traceability* (TERATAI) dan menjalani magang sebagai *Backend Developer* di PT Telkom Indonesia pada bulan September 2023 – Februari 2024.

Akhir kata, penulis mengucapkan rasa syukur yang sebesar-besarnya atas terselesaikannya penulisan Tugas Akhir ini. Untuk keperluan komunikasi dan diskusi lebih lanjut, penulis dapat dihubungi melalui email nailykhairiya@gmail.com.