TUGAS AKHIR - EF234801

RANCANG BANGUN SISTEM PENJUALAN UNTUK TOKO BAJU XYZ BERBASIS WEB MENGGUNAKAN NEXT.JS DAN GO

Adrian Karuna Soetikno

NRP 5025211019

Dosen Pembimbing

Ir. Siti Rochimah MT.,Ph.D
NIP 196810021994032001
Bintang Nuralamsyah, S.Kom, M.Kom
NIP 1999202311029

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

TUGAS AKHIR – EF234801

RANCANG BANGUN SISTEM PENJUALAN UNTUK TOKO BAJU XYZ BERBASIS WEB MENGGUNAKAN NEXT.JS DAN GO

Adrian Karuna Soetikno

NRP 5025211019

Dosen Pembimbing

Ir. Siti Rochimah MT.,Ph.D

NIP 196810021994032001

Bintang Nuralamsyah, S.Kom, M.Kom

NIP 1999202311029

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – EF234801

DESIGN AND DEVELOPMENT OF A WEB-BASED SALES SYSTEM FOR XYZ CLOTHING STORE USING NEXT.JS AND GO

Adrian Karuna Soetikno

NRP 5025211019

Main Advisor

Ir. Siti Rochimah MT.,Ph.D
NIP 196810021994032001
Co-Advisor
Bintang Nuralamsyah, S.Kom, M.Kom
NIP 1999202311029

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2025

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM PENJUALAN UNTUK TOKO BAJU XYZ BERBASIS WEB MENGGUNAKAN NEXT.JS DAN GO

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 : **ADRIAN KARUNA SOETIKNO**
NRP. 502511019

Disetujui oleh Tim Penguji Tugas Akhir:

1.	Ir. Siti Rochimah MT.,Ph.D	Pembimbing
2.	Bintang Nuralamsyah, S.Kom, M.Kom	Ko-pembimbing
3.	Dr. Sarwosri, S.Kom. , M.T.	Penguji
4.	Rizky Januar Akbar, S.Kom., M.Eng	Penguji

SURABAYA
Mei, 2025

(Halaman ini sengaja dikosongkan)

APPROVAL SHEET

DESIGN AND DEVELOPMENT OF A WEB-BASED SALES SYSTEM FOR XYZ CLOTHING STORE USING NEXT.JS AND GO

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Computer Science at
Undergraduate Study Program of Infomatics
Department of Infomatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By: **ADRIAN KARUNA SOETIKNO**
NRP. 5025211019

Approved by Final Project Examiner Team:

1.	Ir. Siti Rochimah MT.,Ph.D	Advisor
2.	Bintang Nuralamsyah, S.Kom, M.Kom	Co-Advisor
3.	Dr. Sarwosri, S.Kom. , M.T.	Examiner
4.	Rizky Januar Akbar, S.Kom., M.Eng	Examiner

SURABAYA
Mei, 2025

(Halaman ini sengaja dikosongkan)

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	: Adrian Karuna Soetikno / 5025211019
Departemen	: Teknik Informatika
Dosen Pembimbing I / NIP	: Ir. Siti Rochimah MT.,Ph.D / 196810021994032001
Dosen Pembimbing II / NIP	: Bintang Nuralamsyah, S.Kom, M.Kom / 1999202311029

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Rancang Bangun Sistem Penjualan Untuk Toko Baju XYZ Berbasis Web Menggunakan Next.Js Dan Go" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.
Bilamana di kemudian hari dietemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima saksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 23 Mei 2025

Mengetahui

Dosen Pembimbing I	Dosen Pembimbing II	Mahasiswa
(Ir. Siti Rochimah MT.,Ph.D)	(Bintang Nuralamsyah, S.Kom, M.Kom)	(Adrian Karuna Soetikno)
NIP. 196810021994032001	NIP. 1999202311029	NRP. 5025211019

(Halaman ini sengaja dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP	: Adrian Karuna Soetikno / 5025211019
Department	: Informatics Engineering
Advisor / NIP	: Ir. Siti Rochimah MT.,Ph.D / 196810021994032001
Co-Advisor / NIP	: Bintang Nuralamsyah, S.Kom, M.Kom / 1999202311029

Hereby declare that Final Project with the title of "Design And Development Of A Web-Based Sales System For XYZ Clothing Store Using Next.Js And Go" 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, 23 Mei 2025

Acknowledge

Advisor	Co-Advisor	Student
(Ir. Siti Rochimah MT., Ph.D)	(Bintang Nuralamsyah, S.Kom, M.Kom)	(Adrian Karuna Soetikno)
NIP. 196810021994032001	NIP. 1999202311029	NRP. 5025211019

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN SISTEM PENJUALAN UNTUK TOKO BAJU XYZ BERBASIS WEB MENGGUNAKAN NEXT.JS DAN GO

Nama Mahasiswa / NRP : Adrian Karuna Soetikno / 5025211019
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing I : Ir. Siti Rochimah MT.,Ph.D
Dosen Pembinbing II : Bintang Nuralamsyah, S.Kom, M.Kom

Abstrak

Dalam era digital, pengelolaan data supplier, stok, transaksi, dan penjualan menjadi faktor krusial dalam operasional bisnis ritel. Toko XYZ menghadapi tantangan dalam mengelola stok barang, transaksi penjualan, pencatatan akses, serta pengembalian barang secara efisien. Penelitian ini telah berhasil merancang dan membangun Sistem Penjualan berbasis web yang dapat mengotomatisasi dan mengintegrasikan proses bisnis utama toko. Aplikasi telah dikembangkan menggunakan Next.js sebagai framework *Frontend* dan Go sebagai *Backend* dengan basis data relasional *PostgreSQL* untuk menyimpan data transaksi dan stok barang. Sistem ini menyediakan fitur utama seperti manajemen stok barang, pencatatan transaksi kasir, log akses pengguna, dan pengelolaan retur barang. Hasil implementasi menunjukkan bahwa aplikasi ini memungkinkan aksesibilitas yang lebih fleksibel dan efisien bagi pengguna. Berdasarkan hasil pengujian yang telah dilakukan, sistem ini terbukti dapat memberikan performa yang memuaskan dengan mendapatkan nilai performa LCP dibawah 2.5 detik pada mayoritas skenario, jauh dibawah 200ms untuk uji INP, dan jauh dibawah 0.2 untuk uji CLS. Selain itu kemudahan penggunaan sistem juga terbukti dengan nilai pengujian SUS yang dilakukan oleh karyawan toko secara langsung pada angka 81.8 yang menunjukkan bahwa sistem telah berjalan dengan sangat baik dalam menjalankan operasional bisnis toko. Sehingga aplikasi ini telah menjadi solusi yang membantu toko mengelola bisnisnya secara lebih sistematis dan terstruktur.

Kata kunci: Sistem Penjualan, Manajemen Stok, Transaksi, Next.Js, Go

(Halaman ini sengaja dikosongkan)

DEVELOPMENT OF A WEB BASED SALES SYSTEM APPLICATION FOR XYZ CLOTHING SHOP WITH NEXT.JS AND GO

Student Name / NRP : Adrian Karuna Soetikno / 5025211019
Department : Informatics Engineering FTEIC - ITS
Advisor : Ir. Siti Rochimah MT.,Ph.D
Co-Advisor : Bintang Nuralamsyah, S.Kom, M.Kom

Abstract

In the digital era, managing supplier data, inventory, transactions, and sales has become a crucial factor in retail business operations. XYZ Store faces challenges in efficiently managing product inventory, sales transactions, access recording, and product returns. This research has successfully designed and built a web-based Sales System that can automate and integrate the store's main business processes. The application has been developed using Next.js as the Frontend framework and Go as the Backend with a PostgreSQL relational database to store transaction and inventory data. This system provides main features such as product inventory management, cashier transaction recording, user access logs, and product return management. Implementation results show that this application enables more flexible and efficient accessibility for users. Based on the testing results that have been conducted, this system has proven to deliver satisfactory performance by achieving LCP performance values below 2.5 seconds in most scenarios, far below 200ms for INP testing, and well below 0.2 for CLS testing. Additionally, the system's ease of use has been proven with SUS testing values conducted directly by store employees at 81.8, indicating that the system has performed very well in running the store's business operations. Therefore, this application has become a solution that helps the store manage its business more systematically and structurally.

Keywords: Sales System, Inventory Management, Transactions, Next.js, Go.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul "Pengembangan Sistem Penjualan Berbasis Web untuk Toko Pakaian XYZ Menggunakan Next.js dan Go".

Buku ini disusun sebagai salah satu syarat untuk menyelesaikan studi pada Program Sarjana Informatika pada Institut Teknologi Sepuluh Nopember. Latar belakang dari dibuatnya Skripsi ini adalah adanya keperluan digitalisasi manajemen operasional pada Toko XYZ yang proses pengembangannya dipercayakan kepada penulis. Dengan selesainya sistem yang dibuat, toko XYZ dapat mengelola stok, penjualan, dan laporan pendapatannya secara lebih efisien dan terintegrasi.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari dukungan dan bantuan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Ibu Siti Rochimah dan Pak Bintang Nuralamsyah, selaku pembimbing, yang telah memberikan bimbingan, arahan, dan masukan selama proses penyusunan buku skripsi ini.
2. Bapak Ary Mazharuddin Shiddiqi dan Bapak Hadzig Fabroyir, selaku Dosen Wali dan Sekretaris Departemen yang telah membantu penulis dalam menjalani dan menyelesaikan problematika perkuliahan.
3. Seluruh dosen, tenaga pendidik, ibu dan bapak yang bekerja di Teknik Informatika ITS yang telah mendidik, membimbing, dan memberikan fasilitas selama masa perkuliahan penulis di Institut Teknologi Sepuluh Nopember.
4. Orang tua dan keluarga tercinta, atas doa, dukungan moral, dan semangat yang tiada henti.
5. Saudara-saudara di "Kos Sultan" , "Kos Ai" dan "Kos Yuni" yang telah menemani saya dan memberi dukungan dan masukan dikala penulis merasa bingung dan jenuh.
6. Charlin yang setia memberikan dukungan dan juga tantangan dalam fase pengembangan sistem hingga pembuatan buku ini.
7. Pihak Toko, yang telah bersedia menjadi objek penelitian dan memberikan data serta dukungan dalam pelaksanaan studi ini.

Tidak ada hal yang sempurna di dunia ini, oleh karena itu penulis sangat terbuka dan menerima kritik dan masukan yang membangun demi perbaikan sistem maupun pengetahuan penulis baik itu sekarang ataupun di masa yang akan datang

Akhir kata, penulis berharap skripsi ini dapat memberikan manfaat, baik bagi dunia akademik maupun praktis, khususnya dalam pengembangan sistem informasi pada sektor usaha mikro dan menengah.

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN v
PERNYATAAN ORISINALITAS ix
ABSTRAK xiii
KATA PENGANTAR xvii
DAFTAR ISI xix
DAFTAR GAMBAR xxvii
DAFTAR TABEL xxix
DAFTAR KODE SUMBER xxxi
BAB I PENDAHULUAN xxxii
1.1 Latar belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Manfaat 2
BAB II TINJAUAN PUSTAKA 5
2.1 Profil Toko XYZ 5
2.1.1. Karakteristik Usaha 5
2.1.2. Sistem Operasional Saat Ini 5
2.2 Hasil Penelitian Terdahulu 6
2.3 Dasar Teori 7
2.2.1 Point Of Sales (POS) 7
2.2.2 Next.Js 8
2.2.3 Go 9
2.2.4 RESTful API 9
2.2.5 Ukuran Performa Situs 10
BAB III METODOLOGI 13
3.1 Metode yang digunakan 13
3.1.1 Studi Literatur 13
3.1.2 Analisis Kebutuhan 13
3.1.3 Pengembangan Sistem Web 28
3.1.4 Perancangan Desain Antarmuka 28

3.1.5 Perancangan Desain Arsitektur 28
3.1.6 Bahan dan peralatan yang digunakan 34
3.3 Implementasi 34
3.3.1 Implementasi Bagian Antarmuka (Frontend) Sistem 34
3.3.2 Implementasi Bagian Backend 41
BAB IV HASIL DAN PEMBAHASAN 61
4.1 Functional Testing 61
4.1.1. Pengujian Melakukan Masuk/Login 61
4.1.2. Pengujian Membuat Data Karyawan 61
4.1.3. Pengujian Membuat Supplier 62
4.1.4. Pengujian Membuat Stok 63
4.1.5. Pengujian Menyetujui Pending Stok 65
4.1.6. Pengujian Membuat Penjualan 65
4.1.7. Pengujian Melakukan Retur 66
4.2 Usability Testing 68
4.3 Uji Performa Sistem 72
4.3.1 Pengujian LCP (Largest Contentful Paint) 72
4.3.2 Pengujian INP (Interaction to Next Paint) 73
4.3.3 Pengujian CLS (Cumulative Layout Shift) 74
4.4 Uji Keamanan Sistem 74
4.4.1 Uji Coba SQL Injection 75
4.4.2 Uji Coba DDOS dan Load Testing 75
4.5 Analisis Implementasi Arsitektur terhadap Performa Sistem 79
4.5.1 Frontent 79
4.5.2 Backend 80
BAB V KESIMPULAN DAN SARAN 81
5.1 Kesimpulan 81
5.2 Saran 81
DAFTAR PUSTAKA 83
LAMPIRAN-LAMPIRAN ATAU APPENDIK 85
BIODATA KLIEN 95
BIODATA PENULIS 97

DAFTAR GAMBAR

Gambar 2.1 Rentang pendapatan omset bulanan toko 5
Gambar 2.2 Rentang nilai performa tiap parameter Core Web Vitals 11
Gambar 3.1 Metodologi dan alur pengembangan yang digunakan 13
Gambar 3.2 Diagram Kasus Penggunaan Akun Karyawan Stok 20
Gambar 3.3 Diagram Kasus Penggunaan Akun Karyawan Kasir 21
Gambar 3.4 Diagram Kasus Penggunaan Akun Owner/Pemilik 21
Gambar 3.5 Alur Implementasi Aplikasi Web 28
Gambar 3.6 Desain Arsitektur Sistem 29
Gambar 3.7 Halaman Masuk aplikasi 35
Gambar 3.8 Halaman Cabang 35
Gambar 3.9 Halaman Supplier 36
Gambar 3.10 Halaman Karyawan 36
Gambar 3.11 Halaman Pending Stok 37
Gambar 3.12 Halaman Pengeluaran 37
Gambar 3.13 Halaman Riwayat Akses 38
Gambar 3.14 Halaman Riwayat Transaksi 38
Gambar 3.15 Halaman Stok 39
Gambar 3.16 Halaman Final Stok 39
Gambar 3.17 Halaman Return Barang 40
Gambar 3.18 Halaman Riwayat Return 40
Gambar 3.19 Halaman Transaksi 41
Gambar 4.1 Rentang nilai evaluasi SUS 70
Gambar 4.2 Grafik Varians jawaban tiap pertanyaan SUS 71
Gambar 4.2 Hasil pengujian terhadap banyak endpoint selama 10 menit 76
Gambar 4.3 Grafik response time dan throughput dari pengujian endpoint 77
Gambar 4.4 Hasil pengujian terhadap endpoint login selama 10 menit 77
Gambar 4.5 Grafik Response dan Throughput terhadap pengujian endpoint login selama 10 menit 78

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu 6
Tabel 2.2 Penjelasan Metode/Protokol RESTful API 9
Tabel 3.1 Tabel Kebutuhan Fungsionalitas 14
Tabel 3.2 Tabel Batasan Fungsional tiap modul 18
Tabel 3.3 Spesifikasi Penggunaan Fungsional Login 22
Tabel 3.4 Spesifikasi Penggunaan Fungsional Registrasi Akun/Data Karyawa 22
Tabel 3.5 Spesifikasi Penggunaan Fungsional Pembuatan Data Supplier 23
Tabel 3.6 Spesifikasi Penggunaan Fungsional Pembuatan Data Cabang 23
Tabel 3.7 Spesifikasi Penggunaan Fungsional Melihat Log Akses 24
Tabel 3.8 Spesifikasi Penggunaan Fungsional Pembuatan Data Produk Baru 24
Tabel 3.9 Spesifikasi Penggunaan Fungsional Penyetujuan Pending Produk 25
Tabel 3.10 Spesifikasi Penggunaan Fungsional Pembuatan Data Transaksi 25
Tabel 3.11 Spesifikasi Penggunaan Fungsional Return Konsumen 26
Tabel 3.12 Spesifikasi Penggunaan Fungsional Return Supplier 27
Tabel 3.13 Spesifikasi Penggunaan Fungsional Pembuatan Data pengeluaran 27
Tabel 3.14 Tabel Endpoint API 30
Tabel 3.15 Penjelasan Implementasi untuk endpoint Masuk Akun 41
Tabel 3.16 Penjelasan implementasi pengambilan data cabang 43
Tabel 3.17 Penjelasan implementasi pembuatan data Supplier 45
Tabel 3.18 Penjelasan implementasi pengambilan data karyawan 46
Tabel 3.19 Penjelasan implementasi pembuatan data pengeluaran 47
Tabel 3.20 Penjelasan implementasi middleware log akses 48
Tabel 3.21 Penjelasan implementasi Service pengambilan data karyawan 50
Tabel 3.22 Penjelasan implementasi pengambilan Riwayat Transaksi 52
Tabel 3.23 Penjelasan implementasi pembuatan stok 54
Tabel 3.24 Penjelasan implementasi penyetujuan data stok 55
Tabel 3.25 Penjelasan implementasi penyetujuan data stok 57
Tabel 3.26 Penjelasan implementasi penyetujuan data stok 58
Tabel 3.27 Penjelasan implementasi penyetujuan data stok 60
Tabel 4.1 Skenario pengujian masuk akun 61
Tabel 4.2 Skenario pengujian pembuatan data karyawan 62
Tabel 4.3 Skenario pengujian pembuatan data supplier 62
Tabel 4.4 Skenario pengujian pembuatan data stok barang 63

Tabel 4.5 Skenario pengujian penyetujuan pending stok 65
Tabel 4.6 Skenario pengujian pembuatan data transaksi/penjualan 66
Tabel 4.7 Skenario pengujian pembuatan retur barang 67
Tabel 4.8 Tabel pertanyaan kuisioner SUS 68
Tabel 4.9 Hasil evaluasi sus terhadap 36 responden acak 69
Tabel 4.10 Hasil Evaluasi SUS terhadap 4 karyawan toko 71
Tabel 4.11 Tabel rentang nilai parameter yang diuji 72
Tabel 4.12 Tabel hasil pengujian LCP 72
Tabel 4.13 Tabel hasil pengujian INP 73
Tabel 4.14 Tabel hasil pengujian CLS 74

DAFTAR KODE SUMBER

Kode Sumber 3.1 Implementasi untuk endpoint Masuk Akun 41
Kode Sumber 3.2 Implementasi untuk pengambilan data cabang 42
Kode Sumber 3.3 Implementasi untuk pembuatan data Supplier 45
Kode Sumber 3.4 Implementasi pengambilan data karyawan 46
Kode Sumber 3.5 Implementasi pembuatan data pengeluaran 47
Kode Sumber 3.6 Implementasi middleware log akses 48
Kode Sumber 3.7 Implementasi Service pembuatan data transaksi 50
Kode Sumber 3.8 Implementasi pengambilan Riwayat Transaksi 51
Kode Sumber 3.9 Implementasi pembuatan stok 54
Kode Sumber 3.10 Implementasi penyetujuan data stok 55
Kode Sumber 3.11 Implementasi penyetujuan data stok 57
Kode Sumber 3.12 Implementasi penyetujuan data stok 58
Kode Sumber 3.13 Implementasi penyetujuan data stok 60

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 2.1 Rentang pendapatan omset bulanan toko 5
Gambar 2.2 Rentang nilai performa tiap parameter Core Web Vitals 11
Gambar 3.1 Metodologi dan alur pengembangan yang digunakan 13
Gambar 3.2 Diagram Kasus Penggunaan Akun Karyawan Stok 20
Gambar 3.3 Diagram Kasus Penggunaan Akun Karyawan Kasir 21
Gambar 3.4 Diagram Kasus Penggunaan Akun Owner/Pemilik 21
Gambar 3.5 Alur Implementasi Aplikasi Web 28
Gambar 3.6 Desain Arsitektur Sistem 29
Gambar 3.7 Halaman Masuk aplikasi 35
Gambar 3.8 Halaman Cabang 35
Gambar 3.9 Halaman Supplier 36
Gambar 3.10 Halaman Karyawan 36
Gambar 3.11 Halaman Pending Stok 37
Gambar 3.12 Halaman Pengeluaran 37
Gambar 3.13 Halaman Riwayat Akses 38
Gambar 3.14 Halaman Riwayat Transaksi 38
Gambar 3.15 Halaman Stok 39
Gambar 3.16 Halaman Final Stok 39
Gambar 3.17 Halaman Return Barang 40
Gambar 3.18 Halaman Riwayat Return 40
Gambar 3.19 Halaman Transaksi 41
Gambar 4.1 Rentang nilai evaluasi SUS 70
Gambar 4.2 Grafik Varians jawaban tiap pertanyaan SUS 71
Gambar 4.2 Hasil pengujian terhadap banyak endpoint selama 10 menit 76
Gambar 4.3 Grafik response time dan throughput dari pengujian endpoint 77
Gambar 4.4 Hasil pengujian terhadap endpoint login selama 10 menit 77
Gambar 4.5 Grafik Response dan Throughput terhadap pengujian endpoint login selama 10 menit 78

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu 6
Tabel 2.2 Penjelasan Metode/Protokol RESTful API 9
Tabel 3.1 Tabel Kebutuhan Fungsionalitas 14
Tabel 3.2 Tabel Batasan Fungsional tiap modul 18
Tabel 3.3 Spesifikasi Penggunaan Fungsional Login 22
Tabel 3.4 Spesifikasi Penggunaan Fungsional Registrasi Akun/Data Karyawa 22
Tabel 3.5 Spesifikasi Penggunaan Fungsional Pembuatan Data Supplier 23
Tabel 3.6 Spesifikasi Penggunaan Fungsional Pembuatan Data Cabang 23
Tabel 3.7 Spesifikasi Penggunaan Fungsional Melihat Log Akses 24
Tabel 3.8 Spesifikasi Penggunaan Fungsional Pembuatan Data Produk Baru 24
Tabel 3.9 Spesifikasi Penggunaan Fungsional Penyetujuan Pending Produk 25
Tabel 3.10 Spesifikasi Penggunaan Fungsional Pembuatan Data Transaksi 25
Tabel 3.11 Spesifikasi Penggunaan Fungsional Return Konsumen 26
Tabel 3.12 Spesifikasi Penggunaan Fungsional Return Supplier 27
Tabel 3.13 Spesifikasi Penggunaan Fungsional Pembuatan Data pengeluaran 27
Tabel 3.14 Tabel Endpoint API 30
Tabel 3.15 Penjelasan Implementasi untuk endpoint Masuk Akun 41
Tabel 3.16 Penjelasan implementasi pengambilan data cabang 43
Tabel 3.17 Penjelasan implementasi pembuatan data Supplier 45
Tabel 3.18 Penjelasan implementasi pengambilan data karyawan 46
Tabel 3.19 Penjelasan implementasi pembuatan data pengeluaran 47
Tabel 3.20 Penjelasan implementasi middleware log akses 48
Tabel 3.21 Penjelasan implementasi Service pengambilan data karyawan 50
Tabel 3.22 Penjelasan implementasi pengambilan Riwayat Transaksi 52
Tabel 3.23 Penjelasan implementasi pembuatan stok 54
Tabel 3.24 Penjelasan implementasi penyetujuan data stok 55
Tabel 3.25 Penjelasan implementasi penyetujuan data stok 57
Tabel 3.26 Penjelasan implementasi penyetujuan data stok 58
Tabel 3.27 Penjelasan implementasi penyetujuan data stok 60
Tabel 4.1 Skenario pengujian masuk akun 61
Tabel 4.2 Skenario pengujian pembuatan data karyawan 62
Tabel 4.3 Skenario pengujian pembuatan data supplier 62
Tabel 4.4 Skenario pengujian pembuatan data stok barang 63

Tabel 4.5 Skenario pengujian penyetujuan pending stok 65
Tabel 4.6 Skenario pengujian pembuatan data transaksi/penjualan 66
Tabel 4.7 Skenario pengujian pembuatan retur barang 67
Tabel 4.8 Tabel pertanyaan kuisioner SUS 68
Tabel 4.9 Hasil evaluasi sus terhadap 36 responden acak 69
Tabel 4.10 Hasil Evaluasi SUS terhadap 4 karyawan toko 71
Tabel 4.11 Tabel rentang nilai parameter yang diuji 72
Tabel 4.12 Tabel hasil pengujian LCP 72
Tabel 4.13 Tabel hasil pengujian INP 73
Tabel 4.14 Tabel hasil pengujian CLS 74

DAFTAR KODE SUMBER

Kode Sumber 3.1 Implementasi untuk endpoint Masuk Akun 41
Kode Sumber 3.2 Implementasi untuk pengambilan data cabang 42
Kode Sumber 3.3 Implementasi untuk pembuatan data Supplier 45
Kode Sumber 3.4 Implementasi pengambilan data karyawan 46
Kode Sumber 3.5 Implementasi pembuatan data pengeluaran..... 47
Kode Sumber 3.6 Implementasi middleware log akses..... 48
Kode Sumber 3.7 Implementasi Service pembuatan data transaksi..... 50
Kode Sumber 3.8 Implementasi pengambilan Riwayat Transaksi..... 51
Kode Sumber 3.9 Implementasi pembuatan stok 54
Kode Sumber 3.10 Implementasi penyetujuan data stok 55
Kode Sumber 3.11 Implementasi penyetujuan data stok 57
Kode Sumber 3.12 Implementasi penyetujuan data stok 58
Kode Sumber 3.13 Implementasi penyetujuan data stok 60

(Halaman ini sengaja dikosongkan)

BAB I PENDAHULUAN

1.1 Latar belakang

Dalam era digital, teknologi informasi telah menjadi elemen krusial dalam pengelolaan bisnis ritel. Sistem informasi modern awalnya dikembangkan sebagai evolusi dari sistem pencatatan manual tradisional dan bertujuan untuk menyatukan berbagai proses bisnis dalam satu aplikasi terintegrasi. Sistem berbasis web yang dirancang digunakan untuk meningkatkan efisiensi dan efektivitas operasional perusahaan dengan mengotomatisasi berbagai fungsi bisnis, namun penerapannya sering kali menghadapi tantangan seperti kompleksitas implementasi dan resistensi pengguna terhadap perubahan sistem (Coşkun et al., 2022).

Toko XYZ sebagai usaha ritel di bidang pakaian menghadapi tantangan dalam pengelolaan stok dan transaksi yang masih dilakukan secara manual menggunakan buku tulis. Metode konvensional dalam pencatatan sering kali menyebabkan ketidaksesuaian data stok, kesulitan dalam melacak riwayat transaksi, kurangnya transparansi dalam manajemen akses pengguna, dan mempersulit pemisahan transaksi berdasarkan cabang atau unit usaha. Kondisi ini menciptakan inefisiensi operasional dan potensi kerugian bisnis yang dapat dihindari melalui implementasi sistem informasi yang tepat.

Digitalisasi dalam manajemen bisnis memiliki manfaat signifikan dalam meningkatkan kecepatan, akurasi pengolahan data, serta mendukung produktivitas dan pengambilan keputusan berbasis data (Laudon et al., 2013). Selain aspek operasional, faktor perpajakan juga menjadi pertimbangan penting dalam pengembangan sistem ini. Dalam strategi pengelolaan bisnis ritel, pemisahan unit usaha dalam bentuk badan hukum berbeda, seperti CV atau cabang terpisah, dapat mengoptimalkan beban pajak dengan membagi pendapatan di beberapa entitas, sehingga mengurangi beban pajak keseluruhan dan meningkatkan efisiensi finansial perusahaan (Julia Kagan, 2024).

Toko XYZ memutuskan untuk mengembangkan sistem sendiri daripada menggunakan solusi yang tersedia di pasaran karena beberapa pertimbangan strategis. Pertama, sistem komersial umumnya memiliki biaya lisensi dan operasional yang cukup tinggi, termasuk biaya langganan cloud bulanan yang dapat membebani keuangan toko. Kedua, banyak solusi yang tersedia memiliki fitur yang kompleks dan tidak sepenuhnya sesuai dengan kebutuhan spesifik toko, sehingga diperlukan penyesuaian tambahan yang memakan biaya dan waktu. Ketiga, dengan mengembangkan sistem sendiri, toko dapat memiliki kendali penuh atas data dan sistem, memastikan keamanan informasi, serta memungkinkan fleksibilitas dalam menyesuaikan sistem sesuai dengan perkembangan bisnis di masa depan.

Berdasarkan permasalahan dan kebutuhan tersebut, penelitian ini bertujuan untuk merancang bangun aplikasi manajemen berbasis web bagi Toko XYZ guna mengotomatisasi dan mengintegrasikan proses bisnis utama. Sistem ini dikembangkan dengan Next.js sebagai frontend, Go sebagai backend, serta PostgreSQL guna memastikan performa tinggi dalam pengolahan data. Aplikasi dirancang agar dapat mengelola beberapa unit usaha secara terpisah untuk optimalisasi pajak serta fleksibilitas bisnis. Mengingat keinginan pemilik toko untuk menghindari biaya operasional bulanan, aplikasi akan di-deploy pada mini PC yang beroperasi secara kontinyu di lokasi toko. Dengan sistem ini, manajemen stok, transaksi penjualan, pencatatan akses, dan pengembalian barang dapat dilakukan lebih efisien, serta mendukung analisis data real-time, strategi perpajakan, dan transparansi operasional yang lebih baik.

1.2 Rumusan Masalah

Dalam Topik Tugas Akhir ini, rumusan masalah yang dapat diambil adalah sebagai berikut:

1. Bagaimana cara membuat Sistem penjualan berbasis Web untuk Toko XYZ?
2. Bagaimana sistem dapat memudahkan pengelolaan transaksi dan manajemen stok dibandingkan dengan metode konvensional yang saat ini digunakan?
3. Apakah sistem yang dikembangkan dapat memiliki tingkat kegunaan (*usability*) yang baik untuk dapat digunakan oleh karyawan-karyawan dengan mudah?
4. Bagaimana penerapan arsitektur berbasis Next.js dan Go dapat mendukung performa tinggi dalam sistem untuk bisnis ini?
5. Bagaimana cara melakukan pengujian dan evaluasi untuk performa dan keamanan yang diimplementasikan dari arsitektur yang digunakan?

1.3 Batasan Masalah

Topik Bahasan dari Rumusan Masalah pada poin diatas memiliki beberapa batasan, yang dijelaskan pada poin-poin berikut:

1. Implementasi Aplikasi ini pada sisi Frontend menggunakan Framework Next.Js dengan Bahasa Typescript.
2. Implementasi Aplikasi ini pada sisi Backend menggunakan Bahasa Pemrograman Go dengan basis data *PostgreSql.*
3. Pengujian dilakukan dengan menggunakan sistem yang telah di deploy pada VPS penulis dengan spesifikasi 2 core CPU dan 8 gb ram
4. Kedua Implementasi tersebut akan di *deploy* pada Server yang terletak Pada toko XYZ dan dapat diakses melalui halaman *web* toko

1.4 Tujuan

Berdasarkan Tumusan masalah yang telah ditetapkan pada poin sebelumnya, maka tujuan dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Merancang dan membangun sistem berbasis web untuk Toko XYZ yang dapat mengintegrasikan proses bisnis utama seperti manajemen stok, transaksi penjualan, pencatatan akses, dan pengelolaan retur barang guna meningkatkan efisiensi operasional toko.
2. Menganalisis dampak implementasi sistem dalam pengelolaan transaksi dan stok barang dibandingkan metode konvensional, dengan menilai aspek kemudahan pencatatan, akurasi data, serta kemudahan penggunaan sistem bagi pengguna.
3. Mengevaluasi implementasi sistem guna memastikan kemudahan penggunaan sistem pada pengguna akhir *(end-user).*
4. Mengevaluasi penerapan arsitektur berbasis Next.js dan Go dalam sistem untuk memastikan performa tinggi dan keandalan yang mendukung pertumbuhan bisnis ritel serta kebutuhan pengguna yang terus meningkat.
5. Mengevaluasi penerapan arsitektur berbasis Next.js dan Go dalam sistem untuk memastikan keamanan sistem terjaga dan tidak adanya kerentanan dalam sistem.

1.5 Manfaat

Hasil dari pengerjaan Tugas Akhir ini adalah sebuah sistem berbasis web yang mendukung pengelolaan stok, transaksi, pencatatan akses, serta pengelolaan multi-unit usaha atau cabang dalam satu platform. Aplikasi ini bermanfaat bagi pemilik dan karyawan toko dalam meningkatkan efisiensi operasional, meminimalkan kesalahan pencatatan, serta memberikan transparansi dalam pengelolaan bisnis. Selain itu, sistem ini juga membantu dalam optimalisasi perpajakan melalui pemisahan unit usaha yang terstruktur, sehingga dapat mengurangi beban

pajak secara legal dan efisien. Dengan arsitektur berbasis Next.js dan Go, aplikasi ini mampu memberikan performa tinggi dan skalabilitas yang baik, mendukung pertumbuhan bisnis ritel secara berkelanjutan.

BAB II TINJAUAN PUSTAKA

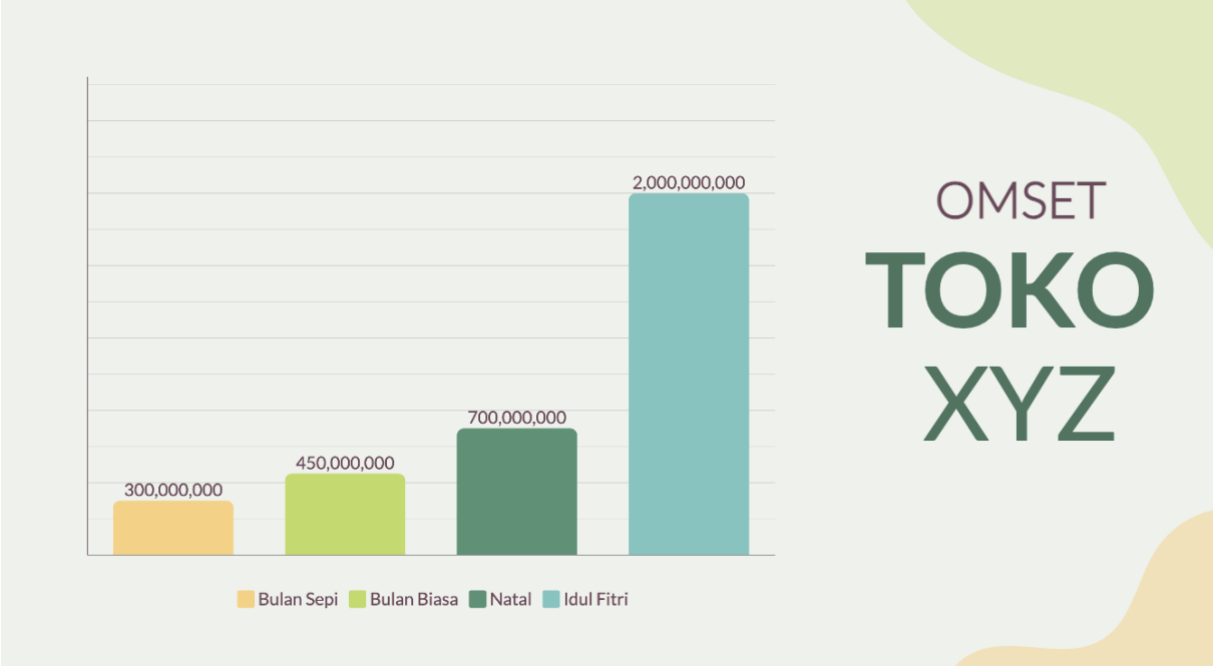
2.1 Profil Toko XYZ

Toko XYZ merupakan salah satu dari beberapa badan usaha yang dimiliki oleh seorang pengusaha dan keluarganya. Toko XYZ sendiri sudah berdiri selama beberapa tahun di wilayah lokal. Lebih tepatnya terletak pada sebuah kota di Kalimantan Timur. Toko XYZ sendiri memiliki beberapa cabang di Kota Tanjung Redeb, dengan total 3 cabang dan belum ditambah badan usaha lainnya yang dimiliki. Namun untuk penjelasan kali ini, akan difokuskan kepada toko XYZ, berikut merupakan penjelasan lebih mendalam dari badan usahanya.

2.1.1. Karakteristik Usaha

Toko XYZ berfokus pada penjualan berbagai jenis pakaian untuk berbagai segmen konsumen. Dalam struktur bisnisnya, toko ini memiliki beberapa unit usaha yang dikelola secara terpisah untuk keperluan optimalisasi pajak dan fleksibilitas bisnis. Strategi pemisahan unit usaha ini merupakan pendekatan yang umum dilakukan oleh UKM untuk mengoptimalkan beban pajak dengan membagi pendapatan di beberapa entitas bisnis.

Toko XYZ memiliki total sekitar 25 karyawan, yang masing-masing terpecah pada masing-masing toko cabang. Toko utama yang terbesar memiliki karyawan minimal 15 orang, tergantung dari musim puncak yang mana toko dapat menambah tenaga karyawan tambahan untuk membantu menjalankan bisnis. Berdasarkan *gambar 2.1* dibawah, dapat kita lihat untuk pendapatan toko dikelompokkan berdasarkan bulan-bulan yang dialami. Seperti bulan normal, sepi, lalu ada bulan hari raya yang paling banyak memberikan omset seperti Idul Fitri dan Natal.



Gambar 2.1 *Rentang pendapatan omset bulanan toko*

2.1.2. Sistem Operasional Saat Ini

Untuk toko XYZ saat ini, sama sekali *tidak menggunakan sistem ataupun aplikasi manajemen*. Sehingga saat ini semua proses penjualan dicatat pada nota, dan salinan nota tersebut disimpan untuk diproses kembali oleh pemiliknya, baik itu menggunakan *excel* ataupun alat lain.

Sedangkan dalam badan usaha lain yang dimiliki yakni apotek, digunakan sebuah sistem

bernama Vmedis. Yang mana sistem ini memiliki fitur lengkap untuk mengelola sebuah apotek dan pihak klien saya membeli sistem tersebut untuk permanen (bukan bulanan). Namun, dalam penggunaanya setelah selama ini, dirasa performanya semakin menurun atau melambat. Dalam pengujian oleh penulis dari akun demo yang diberikan juga memang tiap halaman yang akan diakses memakan waktu lebih dari 5 bahkan mencapai puluhan detik untuk dapat diakses.

Oleh karena tidak adanya sistem yang digunakan pada saat ini, maka tidak ada yang dapat digunakan sebagai tolak ukur maupun standar yang harus diikuti. Penulis juga telah mendapat persetujuan dari pihak klien untuk kebebasan penggunaan *tools* dan *stacks*.

2.2 Hasil Penelitian Terdahulu

Dalam pengembangan sebuah sistem web, terdapat beberapa penelitian terdahulu yang relevan sebagai referensi dan dasar dalam pengembangan aplikasi ini. Studi-studi sebelumnya menunjukkan bahwa sistem dapat memberikan manfaat signifikan bagi bisnis, terutama dalam mengelola data secara terpusat, meningkatkan efisiensi operasional, serta mendukung strategi bisnis yang lebih optimal. Dalam *tabel 2.1* dibawah, telah dirangkumkan berbagai penelitian yang telah dilakukan dalam 5 tahun terakhir. Penelitian-penelitian pada tabel dibawah merupakan penelitian yang dirasa relevan dalam pembangunan sistem penjualan ini.

Tabel 2.1 *Penelitian Terdahulu*

No.	Penulis, Tahun	Pembahasan
1	*Web Based Online Inventory Information System*	
	Soegoto, E.S., Palalungan, A.F. (2020)	Penelitian ini membahas penggunaan sistem inventory online berbasis web untuk memaksimalkan kinerja manusia dalam proses kerja di perusahaan. Sistem informasi inventory online berbasis web merupakan inovasi dalam memudahkan pengelolaan barang, membantu membuat hal-hal rumit menjadi lebih mudah. Hasil penelitian menunjukkan bahwa sistem inventory online berbasis web dapat membantu perusahaan dalam menjalankan bisnisnya dengan lebih efisien. Perusahaan yang menggunakan sistem inventory online saling berkompetisi dalam mengembangkan sistem inventory yang ada agar dapat lebih mendukung pekerjaan perusahaan(Soegoto & Palalungan, 2020)
2	*Building high-performance web applications with NextJS*	
	Ekpobimi, H.O. (2024)	Penelitian ini mengeksplorasi strategi dan teknik utama untuk mengoptimalkan performa menggunakan NextJS, dengan menekankan kemampuannya dalam static site generation (SSG), server-side rendering (SSR), dan incremental static regeneration (ISR). NextJS menyederhanakan proses code splitting dan lazy loading, memastikan aplikasi memuat lebih cepat dan berkinerja lebih baik dengan hanya mengirimkan kode yang diperlukan kepada pengguna. Framework ini juga menyediakan komponen built-in untuk mengelola gambar secara efisien, mengurangi waktu loading dan meningkatkan performa visual (Harrison Oke Ekpobimi, 2024).
3	*Digital transformation of MSMEs in Indonesia: A systematic literature review*	
	Purnomo, Singgih., Nurmalitasari, Nurmalitasari.,	Penelitian ini mengeksplorasi keadaan transformasi digital di antara UMKM Indonesia dengan fokus pada pengembangan, adopsi digital, dan tantangan implementasi melalui Systematic

	Nurchim, Nurchim. (2024)	Literature Review publikasi dari Januari 2020 hingga Mei 2024. Hasil menunjukkan bahwa transformasi digital memberikan manfaat signifikan seperti peningkatan jangkauan pasar, efisiensi operasional, dan profitabilitas. Namun, proses ini menghadapi keterbatasan sumber daya, keterampilan teknis yang tidak memadai, dan isu keamanan data. Dukungan pemerintah dan kerjasama dengan penyedia teknologi menjadi esensial untuk mengatasi hambatan tersebut (Purnomo et al., 2024)
4	*Understanding Tax Deductions: Itemized vs. Standard Deduction*	
	Julia Kagan. (2024)	Dalam artikel *online* yang ditulis oleh Kagan pada situs *Investopedia* membahas strategi strukturisasi bisnis yang legal untuk optimasi perpajakan pada UKM. Beliau mengidentifikasi bahwa pemisahan unit usaha menjadi beberapa entitas bisnis yang berbeda (seperti CV) merupakan praktik yang sah dan umum dilakukan oleh pelaku usaha. Pendekatan ini memungkinkan bisnis untuk memanfaatkan berbagai insentif pajak dan tarif progresif yang tersedia untuk usaha kecil dan menengah. Misalnya, dengan memisahkan operasi retail menjadi beberapa CV, masing-masing entitas dapat memanfaatkan batas pendapatan yang lebih rendah untuk kualifikasi UKM, yang seringkali memiliki tarif pajak yang lebih menguntungkan. Kagan juga menekankan pentingnya dokumentasi yang tepat, pemisahan pembukuan yang jelas, dan pelaporan yang transparan untuk setiap entitas bisnis. Strategi ini harus diimplementasikan dengan konsultasi yang tepat dengan ahli pajak dan akuntan untuk memastikan kepatuhan penuh terhadap peraturan perpajakan yang berlaku (Julia Kagan, 2024).

2.3 Dasar Teori

Pada subbab ini akan dijelaskan berbagai teori dan konsep yang menjadi dasar dalam penyusunan Tugas Akhir ini. Teori-teori tersebut meliputi pemahaman terhadap domain permasalahan, yaitu sistem perencanaan sumber daya perusahaan serta teori-teori pendukung yang berkaitan dengan pengembangan perangkat lunak berbasis web. Selain itu, juga akan dibahas teknologi-teknologi yang digunakan dalam implementasi sistem, seperti framework serta arsitektur dan metode pengembangan perangkat lunak yang diterapkan.

2.2.1 *Point Of Sales (POS)*

Sistem Point of Sale (POS) adalah aplikasi yang digunakan untuk mengolah data transaksi di toko secara efisien. POS berfungsi sebagai sistem pencatatan transaksi penjualan atau yang sering disebut sebagai sistem kasir digital. Sistem ini dirancang untuk mendukung berbagai proses bisnis, seperti pembelian, penjualan, retur barang, dan pencatatan stok. Selain itu, POS juga meningkatkan keamanan transaksi dan keakuratan data, sehingga membantu pemilik usaha dalam pengelolaan inventaris dan pengambilan keputusan bisnis yang lebih efektif (Monica Pakpahan et al., 2022).

Dalam perkembangannya, POS tidak hanya berfungsi sebagai sistem kasir tetapi juga mendukung strategi bisnis melalui fitur promosi penjualan dan loyalitas pelanggan. Beberapa

sistem POS modern telah mengintegrasikan e-voucher sebagai bagian dari strategi pemasaran, memungkinkan pemilik usaha untuk memberikan diskon atau reward kepada pelanggan. Studi ini menekankan bahwa penerapan sistem POS yang efektif dapat meningkatkan daya saing bisnis, memudahkan proses transaksi, dan memberikan pengalaman pelanggan yang lebih baik (Monica Pakpahan et al., 2022).

1. **Fungsi Utama POS**
   POS berfungsi sebagai alat untuk memproses transaksi penjualan, yang mencakup pencatatan penjualan, penghitungan total, dan penerbitan struk. Menurut Laudon dan Laudon (2018), sistem POS yang baik harus mampu memberikan informasi yang akurat dan real-time mengenai penjualan, yang sangat penting untuk pengambilan keputusan bisnis. Dengan integrasi data penjualan, pemilik bisnis dapat menganalisis tren penjualan dan mengoptimalkan strategi pemasaran mereka.

2. **Integrasi dengan Sistem Lain**
   POS modern tidak hanya berfungsi sebagai kasir, tetapi juga terhubung dengan sistem lain dalam organisasi, seperti manajemen inventori dan akuntansi. Menurut O'Brien dan Marakas (2016), integrasi ini memungkinkan data dari berbagai sumber untuk dikumpulkan dan dianalisis, sehingga memberikan gambaran yang lebih jelas tentang kinerja bisnis. Misalnya, ketika suatu produk terjual, sistem POS secara otomatis memperbarui jumlah inventori, yang membantu dalam pengelolaan stok dan penghindaran kehabisan barang.

3. **Perkembangan Teknologi dalam POS**
   Dengan kemajuan teknologi, sistem POS kini dapat diakses melalui berbagai platform, termasuk aplikasi web dan mobile. Menurut (Chaffey, 2015) penggunaan teknologi cloud dalam sistem POS memungkinkan akses data dari mana saja dan kapan saja, serta memudahkan kolaborasi antar tim. Dalam konteks pengembangan Sistem Penjualan menggunakan Next.js dan Golang, integrasi sistem POS dapat meningkatkan efisiensi operasional dan memberikan nilai tambah bagi pengguna.

2.2.2 *Next.Js*

Next.js adalah sebuah framework berbasis React yang dirancang untuk membangun aplikasi web dengan performa tinggi dan optimalisasi SEO yang lebih baik. Framework ini mendukung berbagai teknik rendering seperti *Server-Side Rendering* (SSR), *Static Site Generation* (SSG), dan *Client-Side Rendering* (CSR)**,** yang memungkinkan pengembang untuk memilih metode terbaik sesuai dengan kebutuhan aplikasi mereka (Harrison Oke Ekpobimi, 2024).

Keunggulan utama Next.js adalah kemampuannya dalam melakukan server-side rendering (SSR), yang memungkinkan konten halaman web diproses di server sebelum dikirim ke klien. Hal ini memberikan manfaat besar dalam hal kecepatan halaman dan optimalisasi mesin pencari, yang sangat penting dalam pengembangan web modern. Selain itu, Next.js juga dikenal dengan fitur *static site generation* (SSG), yang memungkinkan halaman web di-render pada saat build time, bukan saat permintaan. Halaman yang dihasilkan bersifat statis dan dapat disimpan di *content delivery network* (CDN) untuk distribusi yang lebih cepat (Susanto et al., 2023).Kerangka kerja Next.js mempermudah pengembangan aplikasi dengan menyediakan arsitektur fleksibel dan modular, yang memungkinkan pengembang untuk menggunakan komponen yang dapat dipakai kembali dan dengan mudah memperluas aplikasi mereka seiring waktu. Next.js digunakan dalam pengembangan bagian *frontend* aplikasi web dalam tugas akhir ini.

Namun, framework ini juga memiliki beberapa kelemahan. Penggunaan SSR dapat

menyebabkan *beban server yang lebih tinggi***,** terutama untuk aplikasi dengan jumlah pengguna yang besar. Selain itu, waktu interaktif halaman mungkin lebih lama karena pengguna harus menunggu eksekusi React sebelum dapat berinteraksi dengan halaman tersebut (Harrison Oke Ekpobimi, 2024).

2.2.3 *Go*

Golang, juga dikenal sebagai "Go," adalah bahasa pemrograman yang dikembangkan oleh Google yang berfokus pada skalabilitas, efisiensi, dan kecepatan. Goroutine, salah satu fitur utama Go yang unggul dalam pengembangan backend, memungkinkan eksekusi tugas secara bersamaan dengan konsumsi memori yang jauh lebih rendah dibandingkan dengan threading konvensional. Go sering digunakan untuk membuat API dan layanan backend yang skalabel karena ringan dan cepat. Karena kemampuan mereka untuk menangani permintaan HTTP yang cepat, framework seperti Gin-Gonic banyak digunakan dalam pengembangan API berbasis Go. Selain itu, menggunakan GORM sebagai ORM membuat komunikasi dengan basis data lebih mudah; namun, tanpa ORM, Go masih dapat mengolah data berbasis JSON dengan cepat. Selain itu, Go sebagai bahasa yang dikompilasi menawarkan performa setara dengan bahasa level rendah seperti C/C++, tetapi dengan keamanan memori dan kemudahan pengembangan yang lebih tinggi (Rosmala & Rasyidin, 2024).

Di sisi lain, meskipun Go dikenal karena performanya yang tinggi, ada beberapa tantangan dalam penggunaannya. Ekosistem pustaka Go belum sekomprehensif bahasa seperti JavaScript atau Python, sehingga pengembang terkadang harus membangun solusi mereka sendiri untuk fitur yang tidak tersedia. Selain itu, meskipun sistem manajemen dependensi Go telah berkembang, masih ada keterbatasan dibandingkan dengan bahasa yang lebih matang dalam aspek ini. Namun, dengan kombinasi kecepatan, efisiensi, dan dukungan bawaan untuk concurrency, Go tetap menjadi pilihan yang sangat baik untuk pengembangan backend yang membutuhkan kinerja tinggi dan skalabilitas (Rosmala & Rasyidin, 2024).

2.2.4 *RESTful API*

RESTful API (Application Programming Interface) adalah metode standar dalam pengembangan layanan web yang mengadopsi arsitektur *Representational State Transfer* (REST)**.** API ini memungkinkan sistem berkomunikasi melalui jaringan menggunakan protokol *Hypertext Transfer Protocol* (HTTP) dengan berbagai metode seperti *GET, POST, PUT, PATCH, dan DELETE* untuk mengelola sumber daya. Dengan pendekatan ini, aplikasi klien dapat berinteraksi secara efisien dengan server, mendukung berbagai platform, termasuk aplikasi berbasis web dan mobile (Fariz et al., 2022). Pada *tabel 2.2*, telah dirangkumkan penjelasan mengenai keempat metode HTTP yang ada dan digunakan dalam sistem ini.

***Tabel 2.2** Penjelasan Metode/Protokol RESTful API*

No.	Nama Metode	Deskripsi
1.	GET	Digunakan untuk mengambil data dari server tanpa mengubah sumber daya. Biasanya digunakan untuk membaca informasi seperti daftar pengguna, detail produk, atau data lainnya yang tersedia dalam sistem.
2.	POST	Digunakan untuk mengirim data baru **ke** server dan menambahkan sumber daya baru dalam sistem. Contohnya adalah saat pengguna mendaftarkan akun baru atau menambahkan entri baru dalam basis data.
3.	PUT	Digunakan untuk memperbarui seluruh data dari sumber daya yang sudah ada. Jika data yang diperbarui tidak ditemukan, metode ini dapat membuat sumber daya baru tergantung pada implementasi

		server.
4.	PATCH	Digunakan untuk memperbarui sebagian data dari sumber daya yang ada tanpa mengganti keseluruhan entitas. Contohnya adalah saat pengguna ingin mengubah hanya nama atau email tanpa memperbarui seluruh informasi akun.
5.	DELETE	Digunakan untuk menghapus sumber daya dari server. Misalnya, menghapus akun pengguna atau menghapus data produk dari basis data.

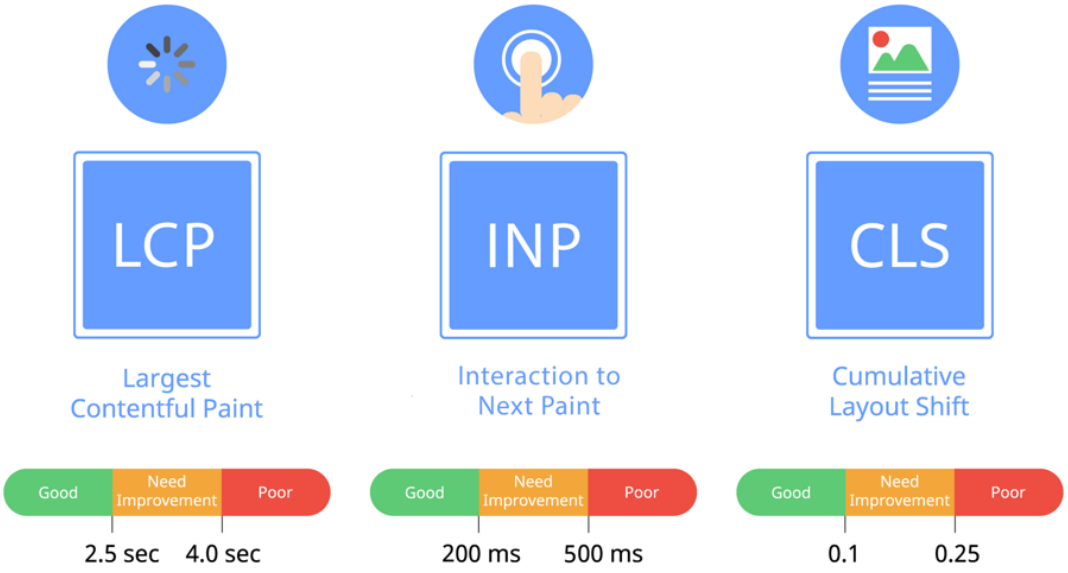
Salah satu keunggulan utama dari RESTful API adalah kemudahan dalam mengelola komunikasi antara klien dan server. API ini memanfaatkan *Uniform Resource Locator* (URL) untuk mengidentifikasi sumber daya serta menggunakan metode HTTP untuk menangani operasinya, sehingga integrasi dan pengembangan layanan menjadi lebih sederhana dan mudah dipahami. Selain itu, *RESTful* API bersifat *stateless*, yang berarti setiap permintaan dari klien harus mencakup semua informasi yang diperlukan oleh server untuk memprosesnya tanpa bergantung pada sesi sebelumnya. Namun hal ini juga dapat disebut sebagai kelemahan dari RESTful. Karena bersifat *stateless*, setiap permintaan harus mengirimkan ulang semua informasi yang diperlukan, yang dapat meningkatkan beban jaringan. Selain itu, *REST* kurang optimal dalam komunikasi real-time dibandingkan dengan *WebSockets*, serta tidak memiliki standar implementasi yang ketat, sehingga integrasi antar sistem dapat bervariasi Dengan karakteristik ini, *RESTful* API menjadi pendekatan utama dalam pengelolaan sumber daya dalam penelitian ini (Fariz et al., 2022).

2.2.5 *Ukuran Performa Situs*

Ukuran performa situs web kini lebih terukur berkat pengenalan metrik Core Web Vitals oleh Google, dengan tiga indikator utama yaitu Largest Contentful Paint (LCP), Interaction to Next Paint (INP), dan Cumulative Layout Shift (CLS). Metrik LCP mengukur kecepatan situs dalam menampilkan elemen konten terbesar, seperti gambar atau blok teks besar, yang secara langsung mempengaruhi persepsi kecepatan pengguna. Dalam penelitian oleh Pandya dkk., situs web universitas-universitas di Indonesia menunjukkan nilai LCP yang jauh lebih tinggi (rata-rata 6820ms) dibandingkan universitas global (rata-rata 891.8ms), yang mengindikasikan lambatnya waktu pemuatan konten utama .(Pandya et al., 2024)

Sementara itu, INP menggantikan metrik First Input Delay (FID) sebagai ukuran interaktivitas yang lebih akurat. INP mencerminkan waktu dari interaksi pengguna (seperti klik atau ketikan) hingga munculnya respons visual berikutnya di layar. Menurut Vepsäläinen dkk., perubahan ini dilakukan karena INP lebih baik dalam mengukur responsiveness situs secara keseluruhan dibanding FID yang hanya menangkap interaksi pertama. Performa yang baik dalam INP menunjukkan situs mampu menanggapi masukan pengguna dengan cepat, memberikan pengalaman yang mulus dan intuitif (Vepsäläinen et al., 2024).

Terakhir, CLS mengukur stabilitas tata letak halaman saat dimuat. Nilai CLS yang tinggi menandakan adanya pergeseran elemen halaman yang tidak terduga, yang bisa mengganggu pengalaman pengguna, terutama saat ingin mengklik elemen tertentu yang tiba-tiba berpindah posisi. Berdasarkan studi Pandya dkk., rata-rata nilai CLS situs universitas Indonesia mencapai 0.346, jauh di atas standar ideal ≤0.1 yang ditetapkan oleh Google, menunjukkan ketidakstabilan tampilan yang signifikan (Pandya et al., 2024). Oleh karena itu, optimalisasi pada aspek LCP, INP, dan CLS merupakan hal krusial untuk memastikan performa situs yang baik, baik dari sisi teknis maupun pengalaman pengguna. *Gambar 2.2* akan menunjukkan ilustrasi rentang nilai yang lebih jelas.



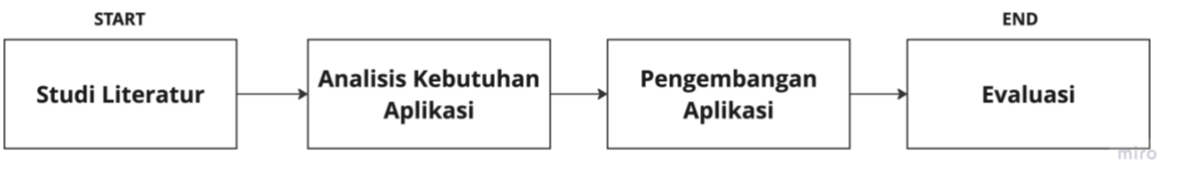
Gambar 2.2 *Rentang nilai performa tiap parameter Core Web Vitals*

(Halaman ini sengaja dikosongkan)

BAB III METODOLOGI

3.1 Metode yang digunakan

Pada Tugas Akhir ini, dibuatlah sebuah rancang bangun pembuatan Sistem Penjualan. Pada subbab ini, akan dijelaskan tentang tahapan-tahapan yang dilalui dalam pembuatan aplikasi tersebut. Gambar 3.1 akan lebih menggambarkan tentang perjalanan pembuatan sistem dalam garis besar. Yaitu dilakukannya studi literatur, pengumpulan kebutuhan dari aplikasi, Pengembangan, hingga evaluasi.



Gambar 3.1 *Metodologi dan alur pengembangan yang digunakan*

3.1.1 Studi Literatur

Pada tahap studi literatur, berbagai informasi yang relevan dengan pengembangan aplikasi web sistem penjualan dikumpulkan dan dipelajari. Informasi tersebut mencakup konsep sistem Point of Sales (POS), manajemen inventori, serta teknologi yang digunakan dalam pengembangan aplikasi, seperti Next.js, Go, dan RESTful API. Sumber informasi dapat berasal dari buku, jurnal ilmiah, internet, serta referensi akademik lainnya. Informasi yang diperoleh digunakan sebagai dasar dan panduan dalam proses pengembangan aplikasi tugas akhir. Hasil studi literatur ini akan dijelaskan secara lebih rinci pada Bab II.

3.1.2 Analisis Kebutuhan

Bagian ini menguraikan kebutuhan perangkat lunak untuk pengembangan aplikasi web sistem penjualan. Kebutuhan perangkat lunak dijelaskan melalui kebutuhan fungsional dan non fungsional (kualitas, keamanan, keandalan, dan sebagainya). Kebutuhan ini didapatkan dari pihak klien secara langsung dengan metode wawancara dan diskusi secara langsung. Baik itu pertemuan secara langsung ataupun melalui Zoom Meeting dan hasil dari wawancara tersebut kemudian diolah dan disusun menjadi sebuah Product Requirement Document (PRD), yang dicantumkan pada halaman lampiran.

3.1.2.1. Kebutuhan Non-Fungsional

Pada bagian kebutuhan non-fungsional ini, pihak klien tidak secara eksplisit menyampaikan standar khusus untuk aspek-aspek kualitas sistem yang diinginkan. Namun, berdasarkan best practices dalam pengembangan sistem informasi serta karakteristik umum usaha retail seperti Toko XYZ, peneliti menetapkan standar kebutuhan non-fungsional yang dianggap relevan dan sesuai dengan kondisi operasional serta ekspektasi pengguna sistem. Standar kebutuhan non-fungsional yang dirancang meliputi aspek performa, keamanan, dan usability, yang dijelaskan sebagai berikut:

1. **Performa**
   Meskipun klien tidak menetapkan target performa secara spesifik, sistem harus mampu mendukung kelancaran operasional toko, terutama pada transaksi real-time yang sensitif terhadap keterlambatan. Berdasarkan standar industri untuk aplikasi web bisnis, sistem ditetapkan mampu memberikan respons terhadap query normal dalam waktu maksimal 3 detik, dengan waktu pemuatan halaman tidak lebih dari 2.5 detik. Hal ini akan dijelaskan

lebih lanjut pada Bab IV. Selain itu, sistem harus mendukung minimal 15 pengguna yang aktif secara bersamaan, sesuai dengan jumlah karyawan yang diperkirakan akan mengakses sistem secara serentak di lingkungan operasional toko.

2. **Keamanan**
   Walaupun klien tidak memberikan spesifikasi keamanan secara detail, keamanan menjadi salah satu prioritas utama mengingat sistem akan mengelola data transaksi dan keuangan yang bersifat sensitif. Oleh karena itu, peneliti menetapkan standar keamanan dasar, antara lain:

   - Enkripsi password menggunakan algoritma hashing yang aman,
   - Penerapan metode login yang aman,
   - Ketahanan sistem terhadap potensi serangan, seperti SQL Injection atau Brute Force.

3. **Usability**
   Klien tidak memberikan spesifikasi khusus mengenai kemudahan penggunaan (usability). Namun, dengan mempertimbangkan latar belakang pengguna sistem yang sebagian besar adalah karyawan toko dengan tingkat literasi teknologi yang beragam, usability menjadi aspek penting. Sistem akan dirancang dengan antarmuka yang intuitif dan mudah dipahami, menggunakan navigasi yang konsisten, serta terminologi yang familiar dalam konteks retail. Standar usability yang ditetapkan mencakup:

   - Akses ke fitur utama dalam maksimal 3 klik,
   - Tampilan yang responsif dan dapat diakses melalui berbagai perangkat (desktop, tablet, dan mobile),
   - Pesan error yang jelas, informatif, dan mudah dipahami, untuk meminimalkan kurva pembelajaran pengguna.

3.1.2.2. Kebutuhan Fungsional

Kebutuhan fungsional perangkat lunak dalam tugas akhir ini didefinisikan berdasarkan layanan-layanan yang harus disediakan oleh aplikasi untuk mencapai tujuannya. Kebutuhan tersebut memberikan panduan yang jelas dan komprehensif mengenai fitur-fitur yang wajib dimiliki oleh sistem. Dimana pada *tabel 3.1* akan menjelaskan seluruh kebutuhan fungsional sistem yang telah dikumpulkan dan dirangkum dalam penjelasan singkat tentang deskripsi masing-masing fungsi.

Tabel 3.1 *Tabel Kebutuhan Fungsionalitas*

Bagian	Kode	Kebutuhan Fungsionalitas	Deskripsi
User	F01	Melakukan Masuk/*Login* kedalam akun	Pengguna dapat masuk dengan akun masing-masing
	F02	Melakukan pembuatan akun karyawan	Pengguna dapat membuat data dan akun untuk karyawan
	F04	Melakukan login akun	Pengguna dapat masuk dengan akun masing-masing
	F05	Melakukan pembaruan data karyawan	Pengguna dapat memperbarui data karyawan
	F06	Mengambil semua data karyawan	Pengguna dapat mengambil seluruh data karyawan yang ada dengan filter

			yang disediakan
	F06	Mengambil data karyawan tertentu	Pengguna dapat mengambil data karyawan tertentu saja
	F07	Menghapus data karyawan	Pengguna dapat menghapus data karyawan tertentu
	F08	Mengambil data pengguna saat ini	Pengguna dapat mengambil data akun yang sedang digunakan saat ini
	F09	Mengunduh data semua karyawan	Pengguna dapat mengunduh data seluruh karyawan dalam format excel
Kategori	F10	Melakukan pembuatan data kategori	Pengguna dapat membuat data kategori yang baru
	F11	Menghapus data kategori	Pengguna dapat menghapus data kategori yang ada
	F12	Mengambil semua data kategori	Pengguna dapat mengambil semua data kategori yang ada
Supplier	F13	Mengambil data awal pembuatan supplier	Pengguna dapat mendapatkan informasi merk, kategori, dan cabang yang diperlukan untuk membuat data supplier baru
	F14	Melakukan pembuatan data supplier	Pengguna dapat membuat data supplier baru
	F15	Menghapus data supplier	Pengguna dapat menghapus data supplier yang ada
	F16	Memperbarui data supplier	Pengguna dapat memperbarui data supplier yang ada
	F17	Memperbarui data stok merk supplier	Pengguna dapat memperbarui data dari merk dan jenis yang disuplai oleh supplier tersebut
	F18	Mengambil seluruh data supplier	Pengguna dapat mengambil seluruh data supplier yang ada dengan filter yang disediakan
	F19	Mengambil data supplier tertentu	Pengguna dapat mengambil data supplier tertentu
	F20	Mengunduh data supplier	Pengguna dapat mengunduh data supplier yang ada dengan format excel
Cabang	F21	Melakukan pembuatan data cabang	Pengguna dapat melakukan pembuatan data cabang baru

	F22	Melakukan pembaruan data cabang	Pengguna dapat melakukan pembuaruan data cabang yang ada
	F23	Mengambil data seluruh cabang	Pengguna dapat mengambil seluruh data cabang yang ada dengan filter yang disediakan
	F24	Mengambil data cabang tertentu	Pengguna dapat mengambil data detail dari cabang tertentu
	F25	Menghapus data cabang	Pengguna dapat menghapus data sebuah cabang
	F26	Mengunduh data cabang	Pengguna dapat mengunduh data cabang dalam format excel
Log Akses	F27	Mengambil seluruh data log akses	Pengguna dapat mengambil seluruh data log akses yang terjadi
Pengeluaran	F28	Melakukan pembuatan data pengeluaran	Pengguna dapat membuat data pengeluaran baru
	F29	Melakukan pengambilan data pengeluaran	Pengguna dapat mengambil data pengeluaran yang ada dengan filter yang disediakan
	F30	Melakukan pengambilan data pengeluaran tertentu	Pengguna dapat mengambil data pengeluaran tertentu dengan filter yang disediakan
	F31	Melakukan pembaruan data pengeluaran	Pengguna dapat memperbarui data pengeluaran yang sudah dibuat
	F32	Melakukan penghapusan data pengeluaran	Pengguna dapat menghapus data pengeluaran yang ada
	F33	Mengunduh data pengeluaran	Pengguna dapat mengunduh data pengeluaran yang ada dalam format excel
Produk	F34	Melakukan pengambilan informasi pembuatan stok produk baru	Pengguna dapat mengambil informasi merk, kategori, supplier, dan cabang untuk digunakan dalam membuat sebuah stok produk baru
	F35	Melakukan pembuatan data produk baru	Pengguna dapat membuat produk baru dengan informasi yang didapatkan sebelumnya
	F36	Melakukan pengambilan informasi pembuatan stok lama	Pengguna dapat mengambil informasi merk, kategori, supplier, dan cabang untuk digunakan dalam membuat sebuah stok produk lama
	F37	Melakukan pembuatan data produk lama	Pengguna dapat membuat produk lama dengan informasi yang didapatkan sebelumnya
	F38	Melakukan pengambilan data semua produk dengan status pending	Pengguna dapat mengambil data produk yang masih dalam status pending

	F39	Melakukan pengambilan data sebuah produk dengan status pending	Pengguna dapat mengambil data dari produk yang masih memiliki status pending
	F40	Melakukan penggantian status produk pending menjadi aktif	Pengguna dapat menyetujui restok produk dari produk yang memiliki status pending tersebut
	F41	Melakukan pembaruan data sebuah produk dengan status pending	Pengguna dapat memperbarui data dari produk dan stok yang dimasukkan dalam status pending tersebut
	F42	Melakukan penghapusan data produk dengan status pending	Pengguna dapat menolak/menghapus data produk yang memiliki status pending tersebut
	F43	Melakukan pengambilan data seluruh produk aktif	Pengguna dapat mengambil semua data produk yang telah memiliki status aktif atau tidak pending dengan filter yang disediakan
	F44	Melakukan pengambilan data detail sebuah produk aktif	Pengguna dapat mengambil data detail dari produk yang telah memiliki status aktif atau tidak pending
	F45	Melakukan pengambilan informasi filter final stok	Pengguna dapat mengambil informasi untuk digunakan mengfilter final stok produk
	F46	Melakukan pengambilan data final stok seluruh produk saat itu	Pengguna dapat mengambil data final stok produk yang berisikan stok, merk, supplier, dan perhitungan nilai jual produk tersebut
Transaksi/Penjualan	F47	Melakukan pengambilan informasi awal pembuatan transaksi penjualan	Pengguna dapat mengambil informasi produk yang dapat dijual/dimasukkan dalam sebuah transaksi
	F48	Melakukan pembuatan transaksi penjualan	Pengguna dapat membuat atau mencatat transaksi penjualan dari produk-produk yang telah dimasukkan
	F49	Melakukan pengambilan histori penjualan/transaksi	Pengguna dapat mengambil data histori penjualan atau transaksi dengan filter yang disediakan
	F50	Melakukan pengambilan data untuk mencetak nota	Pengguna dapat mengambil data transaksi yang akan digunakan untuk mencetak nota
	F51	Melakukan pengambilan data transaksi penjualan berdasarkan nomor nota	Pengguna dapat mengambil data transaksi detail berdasarkan nomor nota untuk perihal retur barang pelanggan

	F52	Melakukan pengambilan data restok berdasarkan nomor restok	Pengguna dapat mengambil data restok berdasarkan nomor restok untuk perihal retur barang supplier
Return	F53	Melakukan pengambilan histori return pelanggan	Pengguna dapat mengambil histori retur dari pelanggan yang telah terjadi
	F54	Melakukan pengambilan histori return supplier	Pengguna dapat mengambil histori retur dari supplier yang telah terjadi
	F55	Melakukan pembuatan data return pelanggan	Pengguna dapat membuat data retur pelanggan berdasarkan data yang disediakan
	F56	Melakukan pembuatan data return supplier	Pengguna dapat membuat data retur supplier berdasarkan data yang disediakan

Sedangkan dari kebutuhan kebutuhan diatas, terdapat personalisasi atau bisa disebut sebagai batasan-batasan yang digunakan dalam pengembangan sistem. Batasan-batasan ini merupakan permintaan khusus dari klien, yang nantinya akan mempengaruhi tiap desain modul sistem. Untuk mengetahui lebih dalam tentang batasan-batasan tiap modul, maka telah dikumpulkan seluruh batasan tersebut didalam *tabel 3.2* yang telah merangkum keseluruhan batasan tersebut.

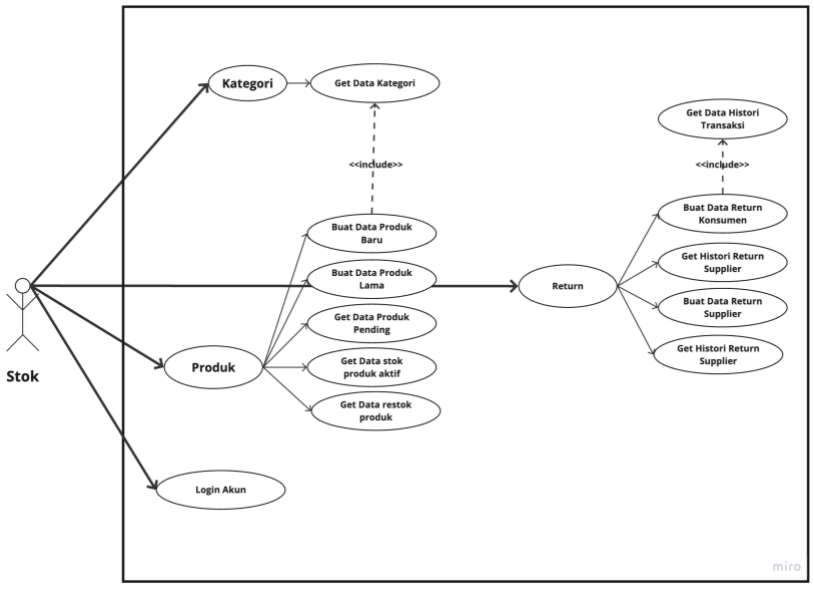
Tabel 3.2 *Tabel Batasan Fungsional tiap modul*

Modul	Batasan yang diinginkan
Pengeluaran/Finance	Pada modul ini, yang dapat mengakses hanya admin/pemilik
	Modul menyimpan jumlah pengeluaran, kategori, metode pembayaran/sumber pengeluaran, dan keterangan pengeluaran
	Modul dapat membantu mencatat dan menampilkan pengeluaran dari petty cash toko
Produk (Stok)	Dengan adanya modul ini, sistem diharuskan memiliki sebuah metode pencegahan dan validasi dalam pencatatan stok/inventori dalam Gudang
	Dari sisi sistem sendiri, telah dilakukan validasi ketika hendak memasukkan data supplier dan stok.

	Selain itu juga sistem hanya memberikan opsi dari pilihan ataupun kombinasi yang terdapat pada sistem
	Selain itu juga terdapat sistem pending stok, dimana pemilik secara manual dapat memastikan bahwa proses input yang dilakukan dicatat dengan baik.
	Pada modul ini, yang dapat mengakses adalah karyawan dengan jabatan stok dan pemilik
	Hanya pemilik yang dapat melakukan pengadaan atau memasukkan data supplier karena adanya rahasia-rahasia perusahaan
	Baik karyawan dan pemilik dapat melakukan input data stok produk berdasarkan data merk yang telah dimasukkan sebelumnya
	Setelah dilakukan pengadaan data stok, stok tersebut tidak langsung aktif/dapat digunakan dalam transaksi
	Stok pada poin sebelumnya, masuk pada fase *pending* yang mana perlu disetujui oleh admin/pemilik, agar status produk dapat “aktif”, sehingga dapat dimasukkan dalam transaksi
	Sistem diharuskan memiliki kemampuan untuk mengatur dan mencatat stok dari produk pada tiap cabang
	Sehingga pada halaman stok dan pencatatan stok masing-masing produk dapat dicantumkan cabang yang menyimpan stok tersebut.
	Penjualan juga mencatat hasil penjualan masing-masing cabang, sehingga pendapatan dapat dipisahkan dari masing-masing cabang
	Dalam satu produk, biasanya setiap ukuran memiliki kode barcode yang berbeda, namun pada sistem ini, pengguna ingin membuat barcodenya sendiri dan tidak mengikuti barcode asli produk.

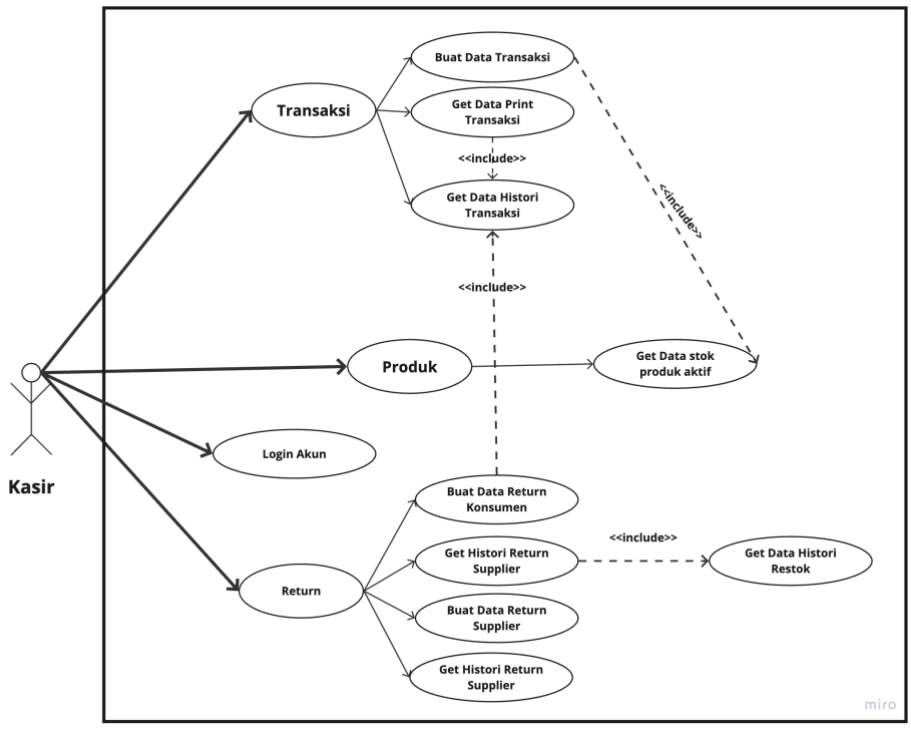
	Satu produk dapat disupplai oleh banyak supplier.
	Diskon dari supplier, apabila tidak diberikan detail dari kategori, maka supplier tersebut memiliki diskon yang sama pada setiap jenis/kategori yang ada.
Transaksi	Sistem berjalan layaknya sistem penjualan/POS seperti umumnya
	Penambahan produk harap dipermudah dengan memberikan sebuah tombol untuk menambah varian lain produk tsb, sehingga tidak perlu double scan
	Dapat diakses oleh karyawan kasir dan pemilik/admin

Dari uraian kebutuhan fungsional diatas akan disusun diagram kasus penggunaan (*use case diagram*) yang menggambarkan interaksi antara pengguna (aktor) dan sistem. Dimana pada pada sistem ini, pengguna akan dibedakan menjadi tiga jabatan, yaitu Karyawan Stok, Karyawan Kasir, dan Akun Admin/*Owner*. Dibawah ini adalah diagram kasus penggunaan untuk karyawan Stok, yang hanya bertugas dalam mengisikan data stok barang baik itu barang baru ataupun barang lama. Selain mengurus stok, karyawan stok juga dapat mengurus return barang baik dari pelanggan atau supplier seperti yang dijelaskan pada *Gambar 3.3.*



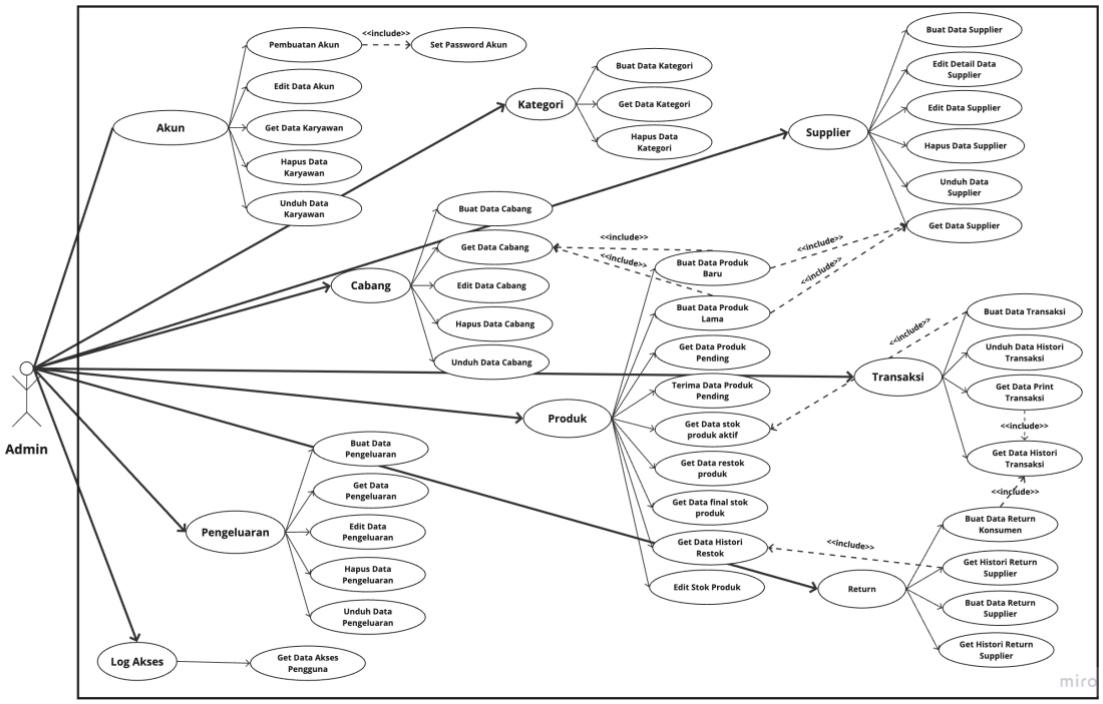
Gambar 3.2 *Diagram Kasus Penggunaan Akun Karyawan Stok*

Kemudian untuk karyawan Kasir, bertugas dalam mengisikan data transaksi yang berisikan produk-produk, selain itu juga kasir juga dapat mengecek stok produk yang ada dan memproses pengembalian atau retur barang baik itu dari pelanggan atau seperti yang dijelaskan pada *Gambar 3.4* yang menggambarkannya dengan diagram kasus penggunaan seperti yang dapat dilihat dibawah ini.



Gambar 3.3 *Diagram Kasus Penggunaan Akun Karyawan Kasir*

Kemudian terakhir adalah *Owner*, yaitu pemilik usaha dan otoritas tertinggi dalam struktur pengguna. Akun ini diberikan hak akses penuh terhadap seluruh modul dan fitur yang tersedia di dalam sistem. Gambar 3.5 menggambarkan secara visual dan detail struktur otorisasi berlapis ini, yang menunjukkan bagaimana setiap level pengguna memiliki batasan akses yang berbeda hingga mencapai tingkat akses tertinggi pada level *Owner*.



Gambar 3.4 *Diagram Kasus Penggunaan Akun Owner/Pemilik*

Untuk memberikan gambaran yang lebih jelas mengenai bagaimana setiap fungsi dalam sistem bekerja, berikut disajikan sejumlah tabel yang merinci alur eksekusi dari fitur-fitur tersebut. Tabel-tabel ini akan memperlihatkan hubungan antar aktor pengguna, modul yang terlibat, serta proses input dan output yang terjadi di dalam sistem.

3.1.2.1. Kasus Penggunaan Melakukan Login

Dalam kasus penggunaan ini, pengguna melakukan proses untuk masuk ke dalam akun yang dimiliki dalam aplikasi. Seperti yang dijelaskan pada *Tabel 3.3* Proses login ini merupakan langkah awal yang penting agar sistem dapat mengidentifikasi dan mengatur hak akses pengguna berdasarkan peran yang dimilikinya. Tanpa melalui proses login ini, pengguna tidak akan dapat mengakses modul-modul utama dalam sistem yang bersifat terbatas atau sensitif.

Tabel 3.3 *Spesifikasi Penggunaan Fungsional Login*

Nama Use Case	**Melakukan Login**
Kode	**F01**
Deskripsi	Pengguna melakukan *Login* atau masuk kedalam akun pada aplikasi web
Aktor	Pengguna
Kondisi Awal	Web pengguna belum terautentikasi
Kondisi Akhir	Web pengguna telah terautentikasi
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman website sebelum login 3. Pengguna mengisikan data akun masing-masing 4. Pengguna memencet tombol *Login*	2. Sistem menampilkan form *Login* 5. Sistem mengautentikasi data pengguna, dan mengatur token pada *cookies*

3.1.2.2. Kasus Penggunaan Melakukan Registrasi Data/Akun Karyawan

Dalam kasus penggunaan ini, pengguna dengan peran sebagai owner melakukan proses untuk memasukkan data karyawan baru ke dalam sistem. Seperti yang diuraikan dalam Tabel 3.4, proses ini mencakup pencatatan informasi penting seperti nama, jabatan, kontak, serta pembuatan akun login yang nantinya akan digunakan oleh karyawan tersebut untuk mengakses sistem. Dengan demikian, data karyawan dan akun akses dapat terintegrasi dan langsung digunakan dalam proses operasional, termasuk kasus penggunaan login yang telah dijelaskan sebelumnya.

Tabel 3.4 *Spesifikasi Penggunaan Fungsional Registrasi Akun/Data Karyawa*

Nama Use Case	**Melakukan Registrasi Akun/Data Karyawan**
Kode	**F02**
Deskripsi	Pengguna membuat data karyawan baru
Aktor	Pengguna (Owner)
Kondisi Awal	Data Karyawan Belum Ditambahkan
Kondisi Akhir	Data Karyawan Telah Ditambahkan
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman karyawan 3. Pengguna mengisikan data karyawan	2. Sistem menampilkan formulir penambahan karyawan

yang diperlukan 4. Pengguna meng-klik tombol Tambah Kata Sandi 5. Pengguna mengisikan kata sandi untuk akun karyawan tersebut	6. Sistem menyimpan data karyawan dan akunnya

3.1.2.3. Kasus Penggunaan Melakukan Pembuatan Data Supplier

Dalam kasus penggunaan ini, pengguna dengan peran owner melakukan proses pemasukan data supplier ke dalam sistem. Seperti yang dijelaskan dalam *Tabel 3.5* data yang dimasukkan mencakup informasi umum seperti nama supplier, alamat, kontak, serta detail tambahan seperti merek atau brand produk yang disuplai oleh masing-masing supplier. Proses ini penting untuk memastikan keterhubungan antara data supplier dan data barang yang mereka suplai, sehingga dapat digunakan dalam proses pembelian dan pengelolaan stok secara lebih akurat.

Tabel 3.5 *Spesifikasi Penggunaan Fungsional Pembuatan Data Supplier*

Nama Use Case	**Melakukan Pembuatan Data Supplier**
Kode	**F14**
Deskripsi	Pengguna mengisikan data supplier yang berisikan supplier, merk yang disuplai, diskon per merk, dan sebagainya
Aktor	Pengguna (Owner)
Kondisi Awal	Data supplier belum tersimpan
Kondisi Akhir	Data supplier telah tersimpan
Alur Normal	
Pengguna	**Sistem**
1.Pengguna mengakses halaman supplier 4.Pengguna mengisikan Data Supplier 5.Pengguna mengisikan merk-merk yang disuplai oleh supplier tersebut 6.Pengguna mengisikan kategori-kategori dari tiap merk yang dijual oleh supplier tersebut 7.Pengguna mengisikan diskon pembelian dari merk dan kategori yang telah diisi sebelumnya 8. Pengguna meng-klik tombol submit/simpan	2. Sistem mengambil informasi yang diperlukan untuk pembuatan data supplier (seperti merk dan kategori yang ada saat ini) 3. Sistem menampilkan form penambahan data supplier 9.Sistem menyimpan data supplier beserta merk dan diskon yang telah dimasukkan

3.1.2.4. Kasus Penggunaan Melakukan Pembuatan Data Cabang

Dalam kasus penggunaan ini, pengguna dengan peran *owner* melakukan proses pemasukan data cabang ke dalam sistem. Data yang dimasukkan mencakup informasi umum seperti nama cabang, alamat, kontak, serta detail tambahan. Pada penjelasan *Tabel 3.6* dapat dilihat, bahwa proses ini penting untuk memastikan keterhubungan antara data produk dan cv atau cabang yang menjual produk tersebut.

Tabel 3.6 *Spesifikasi Penggunaan Fungsional Pembuatan Data Cabang*

Nama Use Case	**Melakukan Pembuatan Data Cabang**

Kode	F21
Deskripsi	Pengguna memasukkan data Cabang yang dimiliki kedalam sistem
Aktor	Pengguna (Owner)
Kondisi Awal	Data cabang belum dimasukkan
Kondisi Akhir	Data cabang telah dimasukkan
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman tambah cabang 3. Pengguna mengisikan data-data cabang yang diperlukan 4. Pengguna meng-klik Submit/simpan	2. Sistem menampilkan form penambahan data Cabang 5.Sistem menyimpan data cabang

3.1.2.5. Kasus Penggunaan Melihat Data Log Akses

Dalam kasus penggunaan ini, pengguna dengan peran sebagai *owner* memiliki hak untuk mengakses dan meninjau data log aktivitas dari seluruh akun pengguna sistem. Data log tersebut mencatat informasi penting seperti waktu akses, identitas pengguna, serta detail perubahan atau aksi yang dilakukan dalam sistem. Fitur ini berfungsi sebagai alat monitoring dan audit internal, yang membantu pemilik usaha memastikan keamanan dan akuntabilitas dalam penggunaan sistem.

Tabel 3.7 *Spesifikasi Penggunaan Fungsional Melihat Log Akses*

Nama Use Case	Melakukan Log Akses
Kode	**F27**
Deskripsi	Pengguna melihat data Riwayat akses *website*
Aktor	Pengguna (owner)
Kondisi Awal	Web pengguna
Kondisi Akhir	Halaman Log Akses
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman riwayat akses	2.Sistem menampilkan halaman web seperti biasa 3.Halaman Log akses menampilkan data akses beserta filter-filter yang tersedia

3.1.2.6. Kasus Penggunaan Melakukan Pembuatan Data Produk Baru

Dalam kasus penggunaan ini, pengguna bertugas untuk melakukan entri data produk baru ke dalam sistem. Data yang dimasukkan meliputi nama produk, kategori, harga, stok awal, dan informasi lain yang relevan. Pada *tabel 3.8* akan menjelaskan bagaimana proses ini memastikan produk tersedia dalam basis data sehingga dapat digunakan dalam proses penjualan dan manajemen stok.

Tabel 3.8 *Spesifikasi Penggunaan Fungsional Pembuatan Data Produk Baru*

Nama Use Case	Melakukan Pembuatan data produk
Kode	**F35**
Deskripsi	Pengguna mengisikan data produk kedalam sistem
Aktor	Pengguna

Kondisi Awal	Data produk belum dimasukkan
Kondisi Akhir	Data produk telah dimasukkan
Alur Normal	
Pengguna	**Sistem**
1.Pengguna mengakses halaman Tambah Produk 5. Pengguna mengisikan informasi data produk 6. Pengguna mengisikan jumlah masing-masing masing stok per ukuran 7. Pengguna mengklik simpan	2. Sistem mengambil informasi merk, kategori, dan supplier 3. Sistem mempersiapkan form pembuatan data produk 4. Sistem menampilkan form pembuatan data produk 8. Sistem menyimpan data produk tersebut kedalam pending stok

3.1.2.7. Kasus Penggunaan Melakukan penyetujuan pending produk

Dalam kasus penggunaan ini, pengguna akan melakukan validasi terhadap produk-produk yang sebelumnya telah dimasukkan dan berstatus "pending". Persetujuan ini diperlukan agar produk dapat dipublikasikan dan tersedia untuk dijual melalui sistem POS atau modul penjualan. Dengan demikian, kontrol kualitas dan validitas data produk dapat dijaga sebelum produk benar-benar masuk ke alur transaksi.

Tabel 3.9 *Spesifikasi Penggunaan Fungsional Penyetujuan Pending Produk*

Nama Use Case	**Melakukan Penyetujuan pending produk**
Kode	**F40**
Deskripsi	Pengguna menyetujui status produk menjadi aktif
Aktor	Pengguna
Kondisi Awal	Data produk belum aktif
Kondisi Akhir	Data produk telah aktif dan dapat dijual
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman pending stok 4. Pengguna memilih restok yang ingin disetujui 5. Pengguna memilih setuju	2. Sistem mengambil informasi stok yang pending 3. Sistem menampilkan histori-histori restok yang masih pending 7. Sistem merubah data produk tersebut menjadi aktif

3.1.2.8. Kasus Penggunaan Melakukan Pembuatan Data Transaksi/Penjualan

Dalam kasus penggunaan ini, pengguna yang bertugas di kasir akan mencatat produk-produk yang dibeli oleh pelanggan ke dalam sistem POS. Setelah transaksi tercatat, sistem akan menyimpan data tersebut ke dalam basis data sebagai transaksi penjualan yang sah. Selain itu, sistem juga akan mencetak struk sebagai bukti pembelian yang dapat diberikan kepada pelanggan. *Tabel 3.10* memberikan spesifikasi lengkap mengenai proses pembuatan data transaksi ini, meliputi peran aktor yang terlibat, alur kerja sistem, kondisi prasyarat, dan hasil akhir yang diinginkan dari proses transaksi penjualan.

Tabel 3.10 *Spesifikasi Penggunaan Fungsional Pembuatan Data Transaksi*

Nama Use Case	**Melakukan Penambahan Data Transaksi**

Kode	**F48**
Deskripsi	Pengguna penambahan data transaksi/penjualan
Aktor	Pengguna
Kondisi Awal	Data Transaksi belum tercatat
Kondisi Akhir	Data Transaksi telah tercatat
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman transaksi 3. Pengguna mengisikan kode barcode menggunakan barcode scanner 4. Pengguna memasukkan dan mengatur data-data produk apa saja yang dibeli konsumen 5. Pengguna mengisi diskon 6. Pengguna memilih jenis pembayaran 7. Pengguna mengklik cetak nota	2.Sistem mengambil data yang diperlukan (produk yang tersedia) 8.Sistem melakukan pencetakan nota dan menyimpan data penjualan
Alur Alternatif	
Mencetak ulang nota sebelumnya	
Pengguna	**Sistem**
Mengklik cetak nota sebelumnya	Sistem memprint nota transaksi sebelumnya

3.1.2.9. Kasus Melakukan Proses Return Konsumen

Dalam kasus penggunaan ini, pengguna dapat melakukan revisi terhadap transaksi penjualan sebelumnya dengan menggunakan referensi nota. Proses ini digunakan ketika pelanggan melakukan pengembalian barang, baik karena kerusakan, ketidaksesuaian, atau alasan lainnya. Sistem akan mencatat return tersebut dan memperbarui data stok serta laporan keuangan yang terkait. *Tabel 3.11* mendeskripsikan secara komprehensif spesifikasi fungsional dari proses return konsumen, termasuk mekanisme pencarian transaksi berdasarkan referensi nota, validasi pengembalian barang, serta prosedur pembaruan data inventori dan laporan finansial.

Tabel 3.11 *Spesifikasi Penggunaan Fungsional Return Konsumen*

Nama Use Case	**Melakukan Return Konsumen**
Kode	**F55**
Deskripsi	Pengguna memasukkan data pengembalian barang ke konsumen
Aktor	Pengguna
Kondisi Awal	Data stok dan transaksi belum terupdate
Kondisi Akhir	Data stok dan transaksi terupdate
Alur Normal	
Pengguna	**Sistem**

1. Halaman Return 3. Pengguna memilih bagian return konsumen 4.Pengguna mengisikan kode/no restok kedalam form input 6. Pengguna mengatur data restok yang sesuai	2.Sistem menampilkan halaman pengembalian barang 5.sistem menampilkan data restok yang berkaitan 7.sistem memperbarui data penjualan dan restok

3.1.2.10. Kasus Melakukan Proses Return Supplier

Dalam kasus penggunaan ini, pengguna dapat melakukan perubahan terhadap riwayat restok barang jika terjadi ketidaksesuaian dalam pengiriman dari supplier. Proses return dilakukan dengan mencatat barang yang dikembalikan kepada supplier berdasarkan histori restok sebelumnya. Fitur ini memastikan data stok tetap akurat dan terintegrasi dengan pengelolaan hubungan dengan supplier. *Tabel 3.12* menjelaskan proses fungsional yang terjadi dari sisi pengguna dan sistem ketika menjalankan kasus ini.

Tabel 3.12 *Spesifikasi Penggunaan Fungsional Return Supplier*

Nama Use Case	**Melakukan Return Supplier**
Kode	**F56**
Deskripsi	Pengguna memasukkan data pengembalian barang ke supplier
Aktor	Pengguna
Kondisi Awal	Data stok dan transaksi belum terupdate
Kondisi Akhir	Data stok dan transaksi terupdate
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman Return 3. Pengguna memilih bagian return supplier 4.Pengguna mengisikan kode/no restok kedalam form input 6. Pengguna mengatur data restok yang sesuai	2.Sistem menampilkan halaman pengembalian barang 5.sistem menampilkan data restok yang berkaitan 7.sistem memperbarui data penjualan dan restok

3.1.2.11. Kasus Melakukan Pembuatan Data Pengeluaran

Dalam kasus penggunaan ini, pengguna melakukan pencatatan atas penggunaan dana kas kecil (*petty cash*) yang digunakan untuk keperluan operasional toko. Sebagaimana diuraikan dalam *Tabel 3.13*, setiap transaksi pengeluaran dicatat ke dalam sistem agar dapat dilacak dan dikontrol secara transparan. Dengan adanya pencatatan ini, toko dapat memantau arus kas keluar dan membuat laporan keuangan yang lebih akurat.

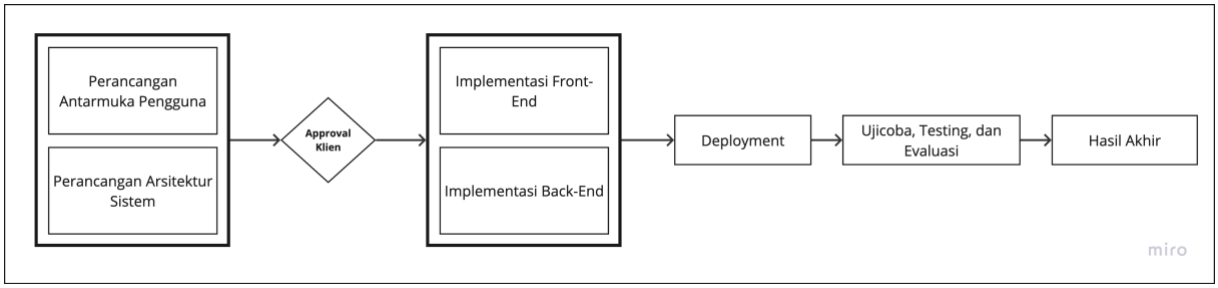
Tabel 3.13 *Spesifikasi Penggunaan Fungsional Pembuatan Data pengeluaran*

Nama Use Case	**Melakukan Pembuatan Data Pengeluaran**
Kode	**F28**
Deskripsi	Pengguna melakukan pengisian data pengeluaran toko

Aktor	Pengguna (owner)
Kondisi Awal	Data pengeluaran belum tersimpan
Kondisi Akhir	Data pengeluaran berhasil ditambahkan
Alur Normal	
Pengguna	**Sistem**
1. Pengguna mengakses halaman pengeluaran 3. Pengguna mengisikan data pengeluaran yang dilakukan 4. Pengguna meng-klik tombol submit	2. Sistem menampilkan form data pengeluaran 5.Sistem menyimpan data pengeluaran

3.1.3 Pengembangan Sistem Web

Setelah selesainya proses analisis kebutuhan, maka kita dapat masuk ke tahap pengembangan dari aplikasi web itu sendiri. Disini kita dapat menentukan bagaimana baiknya arsitektur dari aplikasi itu sendiri dilihat dari kebutuhan yang telah dikumpulkan dan preferensi atau keinginan klien. Gambar 3.5 akan menjelaskan langkah-langkah yang dilalui pada proses ini adalah, merancang desain antarmuka, perancangan arsitektur aplikasi, kemudian dilanjutkan dengan pengembangan sisi front-end dan back-end, lalu diakhiri dengan *deployment* pada server.



Gambar 3.5 *Alur Implementasi Aplikasi Web*

Dengan alur pengembangan seperti ini, maka kami memutuskan untuk menggunakan pendekatan dengan metodologi *Agile*, dimana setiap sprint/durasi/progress yang tercapai kami segera sampaikan kepada klien, sehingga memudahkan bagi klien untuk menambahkan dan meralat apabila adanya perubahan fitur dari yang telah dikumpulkan pada fase sebelumnya.

3.1.4 *Perancangan Desain Antarmuka*

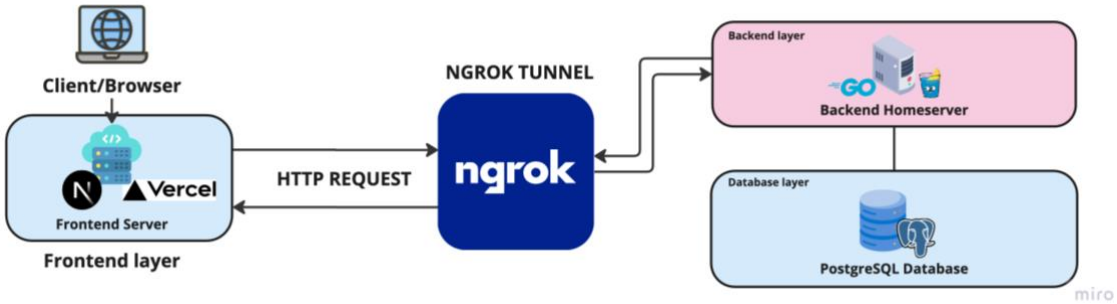
Pada tahap ini setelah dikumpulkannya kebutuhan-kebutuhan dari tahap sebelumnya, maka dapat dilanjutkan dengan pembuatan desain antarmuka. Dengan bantuan alat *Figma*, dilakukan perancangan desain antarmuka dari halaman *website* yang dibuat. Telah terlampir pada lampiran untuk dikunjungi desain antarmukanya. Dalam perancangan desain antarmuka ini, terdapat beberapa poin penting yang diinginkan dari pihak klien, seperti yang dijelaskan pada poin sebelumnya. Berdasarkan permintaan-permintaan tersebut, dapat kita gunakan untuk merancang antarmuka agar sesuai dengan keinginan klien. Untuk implementasi pada sistem, akan dijelaskan pada poin-poin berikutnya.

3.1.5 *Perancangan Desain Arsitektur*

Dalam pengembangan sistem penjualan untuk Toko XYZ, tidak adanya sistem yang sudah berjalan sebelumnya (*existing system*) memberikan kebebasan penuh dalam memilih teknologi dan merancang arsitektur sistem. Kondisi *greenfield* project ini memungkinkan pemilihan technology stack yang optimal tanpa terikat oleh *legacy system* atau keterbatasan infrastruktur yang sudah ada.

Berdasarkan analisis kebutuhan bisnis dan pertimbangan kompleksitas proyek, dipilih arsitektur monolithic dengan pola RESTful API sebagai solusi yang paling sesuai untuk skala dan karakteristik operasional Toko XYZ. Hal ini didasari oleh beberapa alasan, yaitu kemudahan pengembangan dan perawatan, sekaligus faktor ekonomis.

Sistem ini menggunakan arsitektur RESTful API, di mana pertukaran data dilakukan melalui protokol HTTP. Pengguna sebagai klien berinteraksi melalui antarmuka pada lapisan *frontend*, yang kemudian mengirimkan permintaan HTTP ke *backend*. Lapisan *backend* bertanggung jawab untuk mengelola data dengan berkomunikasi langsung ke *basis data*, memastikan proses penyimpanan dan pengambilan informasi berjalan dengan lancar. *Gambar 3.7* akan menjelaskan bagaimana cara kerja arsitektur dengan menjelaskan alur ketika sebuah klien mengakses halaman situs hingga bagaimana server berhubungan dengan yang lain.



Gambar 3.6 *Desain Arsitektur Sistem*

3.1.5.1 Architectural Concerns

Dalam merancang arsitektur sistem untuk Toko XYZ, kondisi greenfield project memberikan kebebasan penuh dalam pemilihan teknologi dan pendekatan arsitektur. Tidak adanya sistem existing yang harus diintegrasikan atau dimigrasikan, serta tidak adanya batasan teknologi spesifik dari klien, memungkinkan fokus sepenuhnya pada tiga concern utama: memaksimalkan performa sistem, minimalisir biaya, dan kemudahan pengembangan dan *maintenance.*

Performa sebagai *Concern* Utama menjadi prioritas mengingat karakteristik bisnis retail yang membutuhkan response time cepat, terutama pada modul kasir. Tanpa beban legacy system, arsitektur dapat dioptimalkan sejak awal untuk mencapai target performa yaitu waktu memuat halaman dan kemampuan menangani concurrent users. Kebebasan ini memungkinkan pemilihan teknologi berdasarkan benchmark performa murni tanpa perlu mempertimbangkan kompatibilitas dengan sistem lama.

Cost-Efficiency sebagai *Concern* lainnya juga sangat krusial mengingat target deployment pada mini PC lokal tanpa biaya cloud berulang. Architectural decisions diarahkan untuk meminimalkan:

1) Biaya Infrastruktur, Arsitektur dirancang efisien untuk berjalan optimal pada hardware terbatas (mini PC)
2) Biaya Lisensi**,** Seluruh teknologi yang dipilih bersifat open source tanpa biaya lisensi
3) Biaya Operasional, Tidak ada biaya cloud hosting, domain berbayar, atau subscription service
4) Biaya Maintenance, Arsitektur yang sederhana untuk meminimalkan kompleksitas maintenance

Karena proyek ini merupakan *greenfield project* dan klien juga tidak memiliki kriteria apapun dalam pemilihan alat ataupun *stacks*, maka pengembang dapat memilih alat-alat berdasarkan kriteria sebelumnya dengan lebih fleksibel. Berikut adalah hasil dari pemilihan alat-alat yang akan digunakan.

1) Next.js dipilih karena performa SSR/SSG yang terbukti cepat dan build size yang optimal
2) Go dipilih karena efisiensi resource dan performa tinggi dengan footprint minimal, ideal untuk mini PC
3) PostgreSQL dipilih karena gratis, powerful, dan ringan dibanding basis data komersial
4) NGROK sebagai solusi tunneling gratis menggantikan domain berbayar

3.1.5.2 Frontend Layer

Lapisan atau *layer* ini memiliki tugas untuk menampilkan hasil desain antarmuka pengguna yang telah dikembangkan sebelumnya. Lapisan ini juga bertugas untuk mengatur komunikasi antara lapisan *Backend* dengan menerima masukan/*input* dari pengguna. Layer *Frontend* ini dikembangkan menggunakan kerangka kerja *NextJS* dan *dideploy* menggunakan *Vercel*.

3.1.5.3 API Tunnel/NGROK

Ngrok adalah alat yang digunakan untuk membuat tunneling dari localhost ke internet, ngrok memungkinkan akses ke aplikasi yang berjalan di komputer lokal melalui URL publik. Mengapa kami menggunakan ngrok, hal ini dikarenakan klien tidak menginginkan adanya biaya bulanan ataupun tahunan dengan membeli domain. Ngrok memberikan sebuah domain gratis untuk kita gunakan dalam mengekspos service *Backend* ini.

3.1.5.4 Backend Layer

Lapisan *Backend* ini bertugas untuk memproses semua data yang masuk maupun keluar pada sistem. Lapisan *Backend* ini dikembangkan dengan menggunakan Go dan *dideploy* pada mini pc yang ada pada toko. *Backend* ini hanya memiliki 1 *service*/layanan yang bertugas sebagai server untuk melayani permintaan yang ditugaskan pada setiap *endpoint* yang ada pada layanan tersebut. Sebagaimana ditunjukkan dalam *Tabel 3.14*, berikut merupakan daftar dari setiap *endpoint* yang ada pada layanan API sistem.

Tabel 3.14 *Tabel Endpoint API*

No	Endpoint API	Metode	Keterangan
1	/api/user/login	POST	Digunakan untuk masuk ke akun masing-masing
2	/api/user	POST	Digunakan untuk membuat entry ke tabel user/akun
4	/api/user/{id}	DELETE	Digunakan untuk menghapus data dari tabel user/akun
5	/api/user/{id}	PATCH	Digunakan untuk merubah/data dari tabel user/akun
6	/api/user/{id}	GET	Digunakan untuk mengambil satu data tertentu berdasarkan id dari tabel user/akun
7	/api/user?params	GET	Digunakan untuk mengambil semua data dari tabel user/akun

8	/api/user/me	GET	Digunakan untuk mengambil data akun yang sedang digunakan saat ini
9	/api/user/download	GET	Digunakan untuk mengunduh data akun yang ada pada sistem
10	/api/cabang	POST	Digunakan untuk membuat entry ke tabel cabang/CV
11	/api/cabang/{id}	DELETE	Digunakan untuk menghapus data tertentu dari tabel cabang
12	/api/cabang/{id}	PATCH	Digunakan untuk mengubah data tertentu dari tabel cabang
13	/api/cabang/{id}	GET	Digunakan untuk mengambil data tertentu dari tabel cabang
14	/api/cabang?params	GET	Digunakan untuk mengambil seluruh data dari tabel cabang
15	/api/cabang/download	GET	Digunakan untuk mengunduh data-data cabang
16	/api/log	GET	Digunakan untuk mengambil seluruh log akses yang ada pada sistem
17	/api/pengeluaran	POST	Digunakan untuk membuat entry pada tabel pengeluaran
18	/api/pengeluaran/{id}	DELETE	Digunakan untuk menghapus data pengeluaran yang telah dibuat
19	/api/pengeluaran/{id}	PATCH	Digunakan untuk mengubah data pengeluaran yang telah dibuat
20	/api/pengeluaran/{id}	GET	Digunakan untuk mengambil salah satu data pengeluaran yang ada
21	/api/pengeluaran	GET	Digunakan untuk mengambil seluruh data pengeluaran
22	/api/pengeluaran/download	GET	Digunakan untuk mengunduh data pengeluaran
23	/api/produk/index	GET	Digunakan untuk mengambil informasi awal untuk membuat entry produk yang belum ada/baru
24	/api/produk/index-old	GET	Digunakan untuk mengambil informasi awal untuk membuat entry produk yang telah ada/lama

25	/api/produk	GET	Digunakan untuk mengambil seluruh data produk yang ada
26	/api/produk/{id}	GET	Digunakan untuk mengambil salah satu data produk
27	/api/produk/pending	GET	Digunakan untuk mengambil produk yang statusnya masih pending/belum dimasukkan dalam stok aktif
28	/api/produk/pending/{id}	GET	Digunakan untuk mengambil data detail dari salah satu produk pending
29	/api/produk/restok-history	GET	Digunakan untuk mengambil data histori restok/produk yang dimasukkan
30	/api/produk/index-final-stok	GET	Digunakan untuk mengambil informasi filter untuk halaman final-stok
31	/api/produk/final-stok	GET	Digunakan untuk mengambil perhitungan final stok produk yang ada pada saat itu
32	/api/produk/create	POST	Digunakan untuk membuat entry data produk baru
33	/api/produk/pending/insert/{id}	POST	Digunakan untuk mengubah status produk dari pending menjadi aktif
34	/api/produk/create-old	POST	Digunakan untuk membuat entry produk yang telah ada
35	/api/produk/pending	PATCH	Digunakan untuk mengubah data produk pending dari restok tertentu
36	/api/produk/pending/{id}	DELETE	Digunakan untuk menghapus data produk pending dari restok tertentu
37	/api/return/user/{id}	GET	Digunakan untuk mengambil informasi transaksi dari nomor nota tertentu untuk retur konsumen
38	/api/return/history/user	GET	Digunakan untuk mengambil informasi histori retur dari konsumen

39	/api/return/supplier/{id}	GET	Digunakan untuk mengambil informasi restok dari nomor restok tertentu untuk retur supplier
40	/api/return/history/supplier	GET	Digunakan untuk mengambil informasi histori retur dari supplier
41	/api/return/user	POST	Digunakan untuk membuat entry retur barang dari konsumen
42	/api/return/supplier	POST	Digunakan untuk membuat entry retur barang dari supplier
43	/api/jenis	POST	Digunakan untuk membuat entry data jenis/kategori
44	/api/jenis	GET	Digunakan untuk merubah entry data jenis/kategori
45	/api/jenis/{id}	DELETE	Digunakan untuk menghapus entry data jenis/kategori
46	/api/supplier/index	GET	Digunakan untuk mendapatkan informasi awal pembuatan data supplier
47	/api/supplier	GET	Digunakan untuk mendapatkan seluruh data supplier yang ada
48	/api/supplier/{id}	GET	Digunakan untuk mendapatkan salah satu informasi detail dari supplier
49	/api/supplier/download	GET	Digunakan untuk mengunduh data supplier yang ada
50	/api/supplier/{id}	PATCH	Digunakan untuk merubah data supplier
51	/api/supplier/supply/{id}	PATCH	Digunakan untuk merubah data merk yang disuplai oleh supplier
52	/api/supplier	POST	Digunakan untuk membuat entry data supplier
53	/api/supplier/{id}	DELETE	Digunakan untuk menghapus salah satu supplier yang ada
54	/api/transaksi/index	GET	Digunakan untuk mendapatkan informasi awal untuk membuat transaksi
55	/api/transaksi	GET	Digunakan untuk mendapatkan histori transaksi

56	/api/transaksi	POST	Digunakan untuk membuat entry data transaksi
57	/api/transaksi/download	GET	Digunakan untuk mengunduh data transaksi
58	/api/transaksi/print/{id}	GET	Digunakan untuk mendapatkan informasi print nota

3.1.5.5 Basis data Layer

Lapisan Basis data ini merupakan lapisan yang bertugas untuk memproses penyimpanan data pengguna dan sistem. Dalam sistem ini digunakan basis data relasional *PostgreSQL*, dengan optimasi di sisi *Indexing* pada tiap tabel. Optimasinya adalah dengan menggunakan metode *Index B-tree* atau *Binary Tree* yang lebih optimal dipasangkan dengan tipe data Integer pada *Primary Key*/*Index* tiap tabel.

3.1.6 Bahan dan peralatan yang digunakan

Dalam pengerjaan tugas akhir ini digunakan beberapa alat pendukung untuk mendukung pengembangan aplikasi web, yang berupa :

1. Sebuah Laptop Macbook pro M1 14 inch, untuk proses pengembangan web
2. Sebuah Tablet Samsung A9 8 inch untuk testing *mobile view*
3. Sebuah Mini Pc yang digunakan sebagai pc utama kasir dan web server dengan spesifikasi sebagai berikut:
4. Visual Studio Code sebagai editor teks untuk mengembangkan aplikasi.Dfa
5. GitHub sebagai sistem kontrol versi aplikasi atau penyimpanan rekam jejak pengembangan
6. Figma yang digunakan dalam membuat desain antarmuka aplikasi.Fa
7. Vertabelo untuk penggambaran ERD
8. Trello untuk pembagian tugas/tiket

3.3 Implementasi

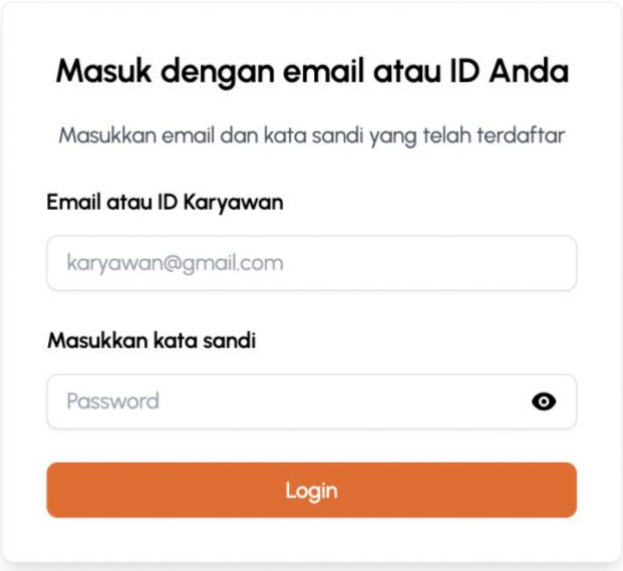
Bagian ini menjelaskan mengenai proses implementasi sistem yang dilakukan berdasarkan rancangan yang telah disusun pada tahap sebelumnya. Implementasi ini mencakup pembuatan dan penerapan desain antarmuka pengguna (*frontend*), serta perwujudan arsitektur aplikasi yang telah dirancang, termasuk struktur logika bisnis dan interaksi dengan basis data. Selain itu, bagian ini juga menyajikan cuplikan kode sumber yang digunakan untuk membangun sistem, guna memberikan gambaran teknis atas realisasi fitur-fitur utama dalam aplikasi berbasis web ini.

3.3.1 Implementasi Bagian Antarmuka (Frontend) Sistem

Bagian frontend berfungsi sebagai perantara bagi pengguna untuk berinteraksi dengan sistem. Implementasi frontend dalam sistem ini didasarkan pada desain antarmuka yang telah dirancang sebelumnya, kemudian diwujudkan dalam bentuk website yang dapat diakses oleh pengguna. Dalam aplikasi ini, pengembangan frontend dilakukan menggunakan kerangka kerja Next.js.

3.3.1.1 Implementasi Halaman Masuk (*Login*)

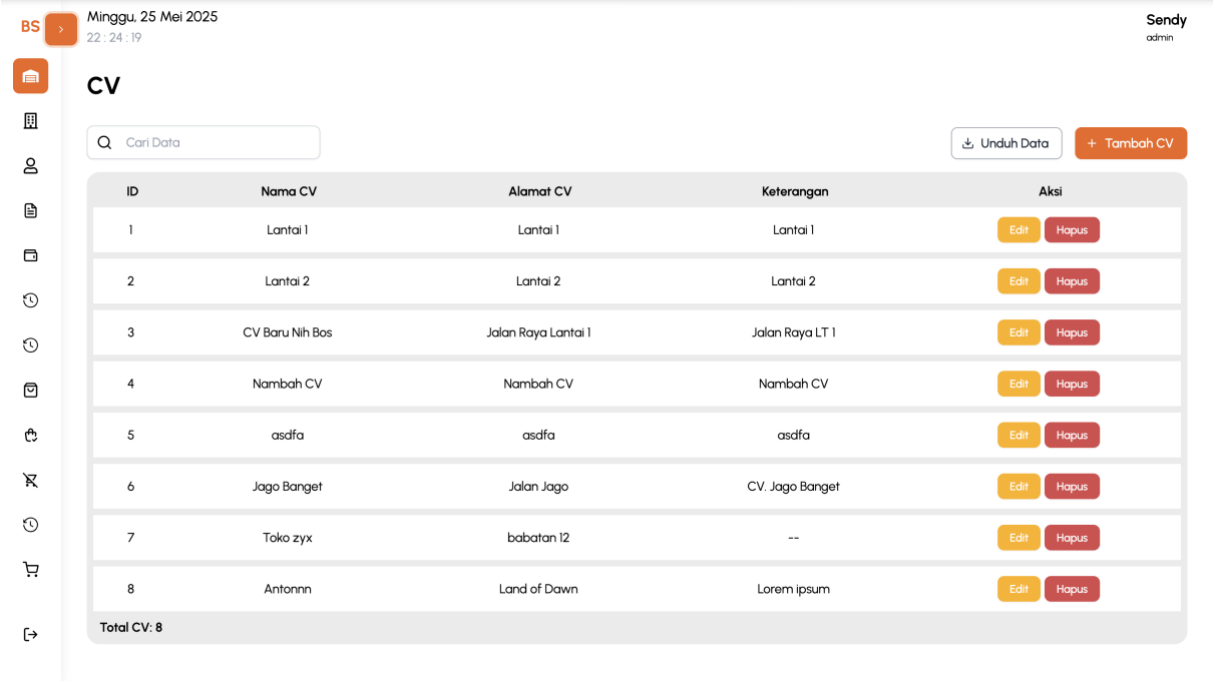
Setiap pengguna yang ingin mengakses aplikasi diharuskan untuk masuk menggunakan akun yang telah didaftarkan. Pengguna dapat masuk ke aplikasi dengan memasukkan email/username beserta kata sandi (password) dari akun yang telah didaftarkan seperti yang terlihat pada *Gambar 3.8.*



Gambar 3.7 *Halaman Masuk aplikasi*

3.3.1.2 Implementasi Halaman Cabang (CV)

Pada halaman ini, pengguna dapat melihat data cabang atau CV yang telah didaftarkan ke dalam sistem. Sebagaimana terlihat pada *Gambar 3.8*, pengguna dapat melihat informasi lebih detail dengan mengakses CV yang didaftarkan. Begitu pula dengan menghapus ataupun menambahkan data baru dari setiap CV.

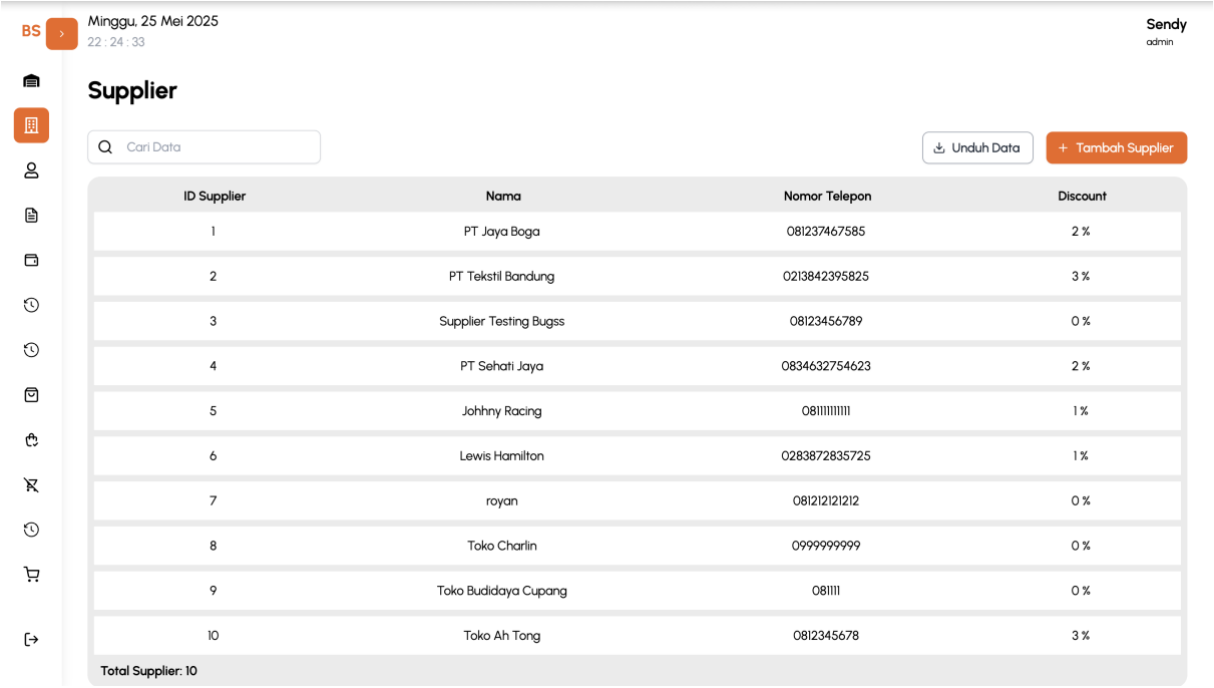


Gambar 3.8 *Halaman Cabang*

3.3.1.3 Implementasi Halaman Supplier

Pada halaman ini, pengguna dapat mengakses data-data supplier yang didaftarkan. Sebagaimana ditunjukkan pada *Gambar 3.9,* sebelum dapat memasukkan stok-stok barang, pengguna perlu memasukkan data supplier terlebih dahulu karena setiap data produk dicatat berdasarkan supplier yang menyuplai, dan diskon harga jual yang diberikan setiap supplier

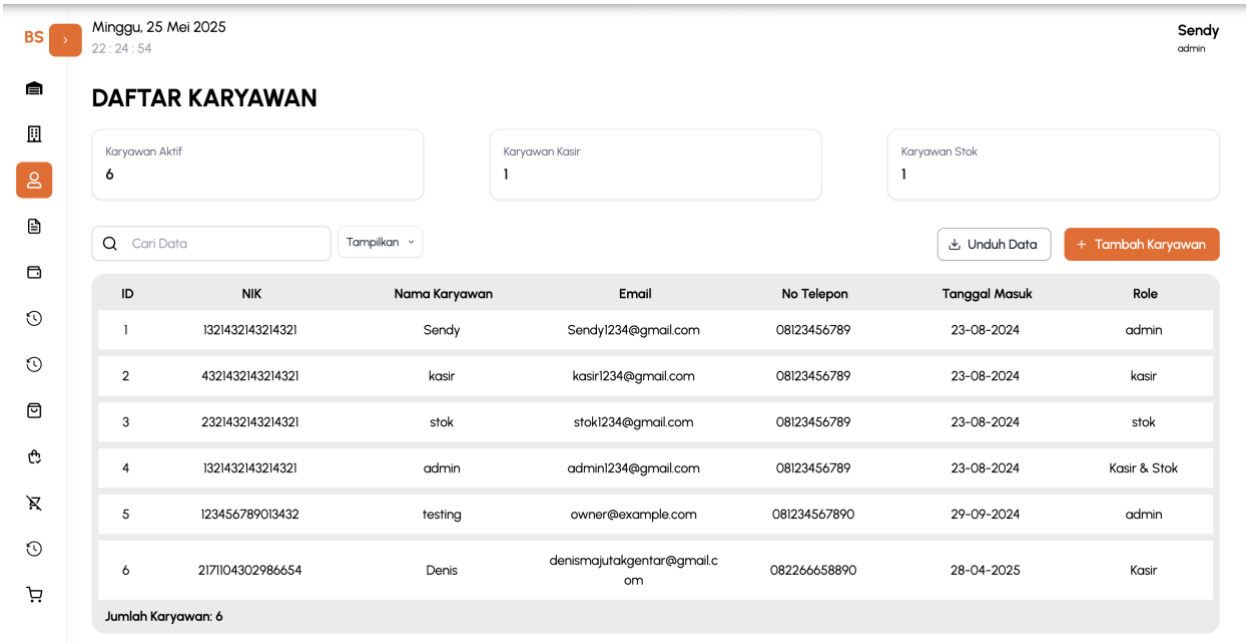
berbeda.



***Gambar 3.9** Halaman Supplier*

3.3.1.4 Implementasi Halaman Karyawan

Halaman ini merupakan halaman manajemen data karyawan dan sekaligus akun. Sebagaimana terlihat pada **Gambar 3.10**, setiap penambahan data karyawan akan ditambahkan juga akun untuk karyawan terkait. Masing-masing akun memiliki jabatan masing-masing.

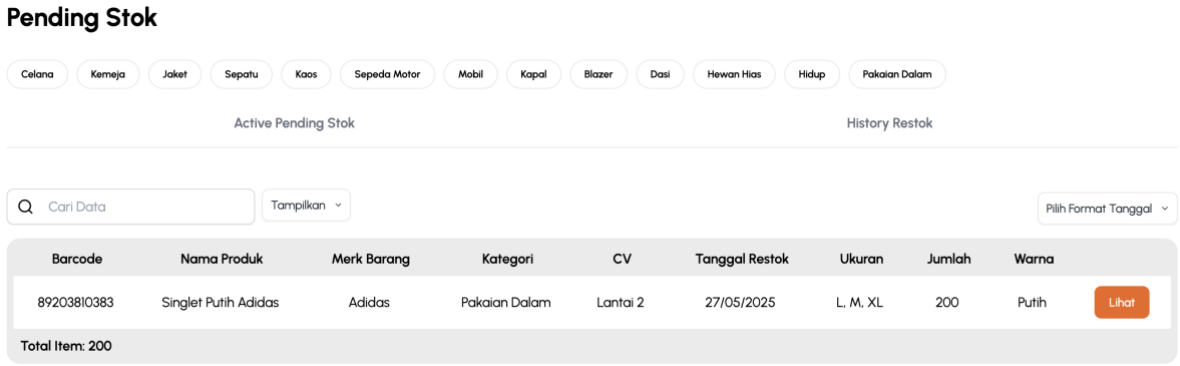


***Gambar 3.10** Halaman Karyawan*

3.3.1.5 Implementasi Halaman Pending Stok

Halaman ini menampilkan sistem manajemen "Pending Stok" dengan berbagai kategori

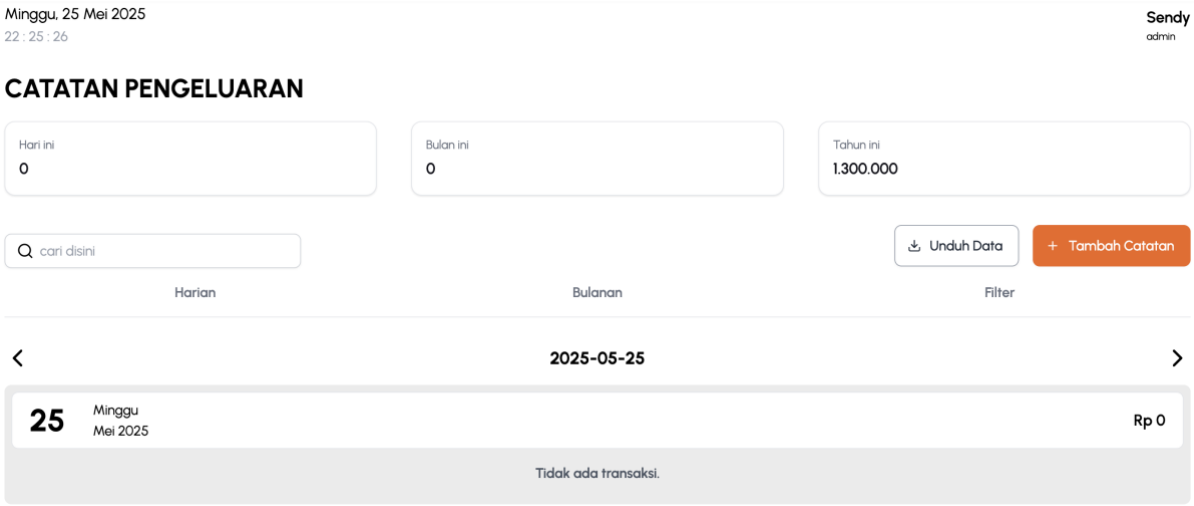
produk di bagian atas (Cabang, Keranjang, Jaket, dll). Sebagaimana ditunjukkan pada *Gambar 3.11*, tampilan utama berisi tabel isi item yang tertunda dan tersedia fitur pencarian, filter tampilan, dan tombol "Lihat" untuk memeriksa detail produk.



Gambar 3.11 *Halaman Pending Stok*

3.3.1.6 Implementasi Halaman Pengeluaran

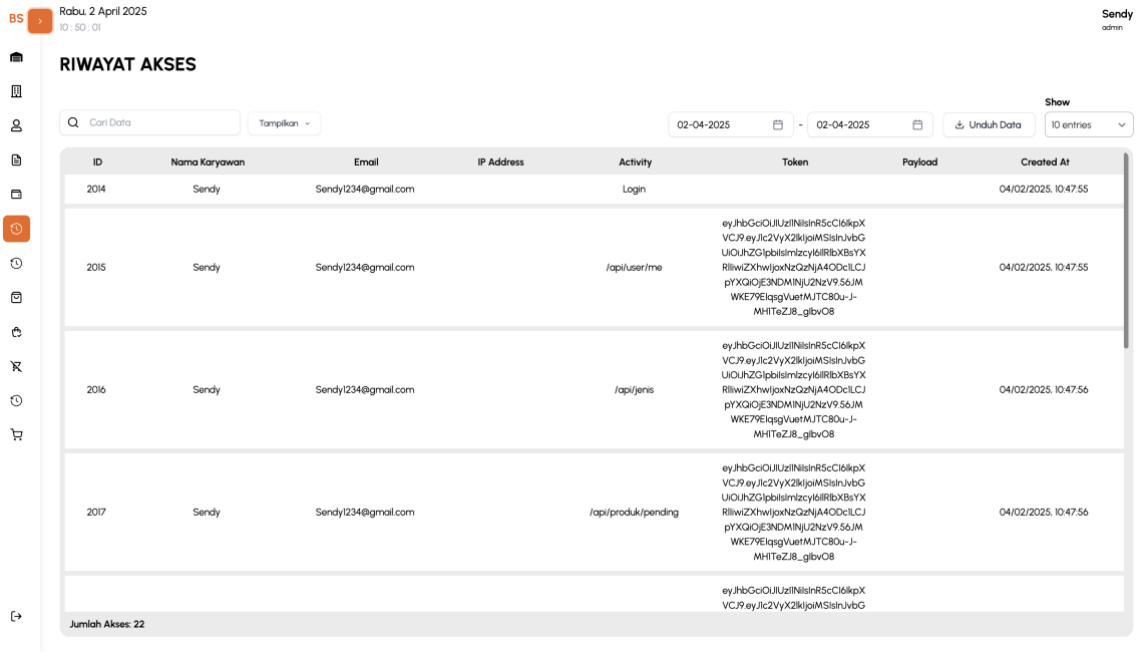
Halaman ini menampilkan "Catatan Pengeluaran" dengan ringkasan transaksi harian, bulanan, dan tahunan. Sebagaimana terlihat pada *Gambar 3.12*, di bagian atas terdapat statistik keuangan yang menunjukkan pengeluaran untuk hari ini, bulan ini, dan tahun ini. Lalu pada tabelnya akan berisi data pengeluaran yang telah dicatat. Terdapat fitur pencarian, tombol "Unduh Data", dan tombol "Tambah Catatan" berwarna oranye untuk menambahkan transaksi baru.



Gambar 3.12 *Halaman Pengeluaran*

3.3.1.7 Implementasi Halaman Riwayat Akses

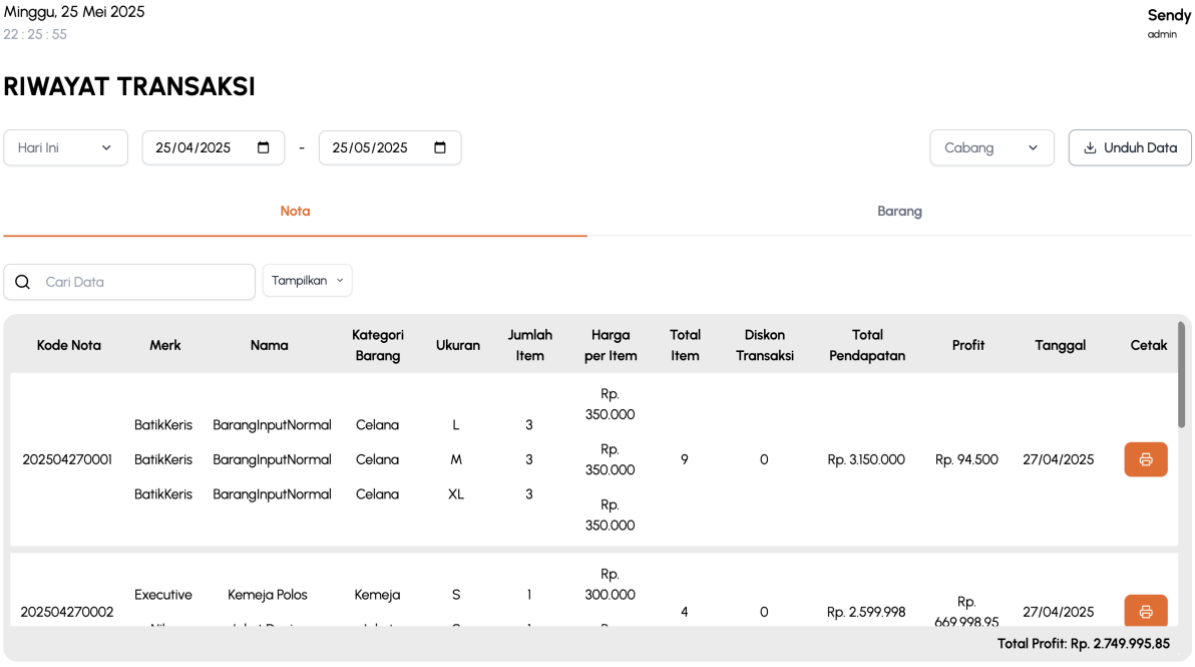
Halaman Riwayat Akses menampilkan catatan aktivitas pengguna dalam sistem, termasuk ID, nama karyawan, email, IP Address, aktivitas**,** serta token dan payload. Sebagaimana ditunjukkan pada *Gambar 3.13*, fitur pencarian dan filter tanggal memudahkan administrator dalam memantau dan mengaudit akses untuk menjaga keamanan sistem.



Gambar 3.13 *Halaman Riwayat Akses*

3.3.1.8 Implementasi Halaman Riwayat Transaksi

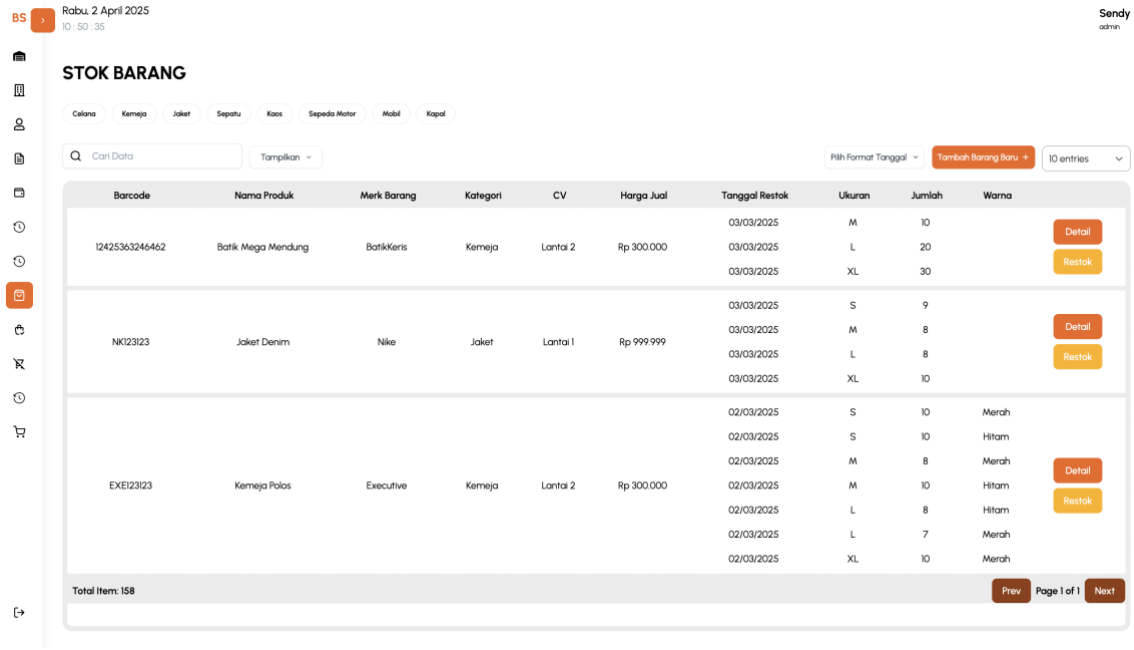
Halaman ini menampilkan Riwayat [transaksi/penjualan] dan memiliki filter dengan rentang tanggal serta filter cabang dan fitur mengunduh data. Sebagaimana terlihat pada *Gambar 3.14* tabel dilengkapi dengan detail seperti kode nota, merk, nama produk, kategori, ukuran, jumlah item, harga per item, diskon, dan total profit keseluruhan.



Gambar 3.14 *Halaman Riwayat Transaksi*

3.3.1.9 Implementasi Halaman Stok

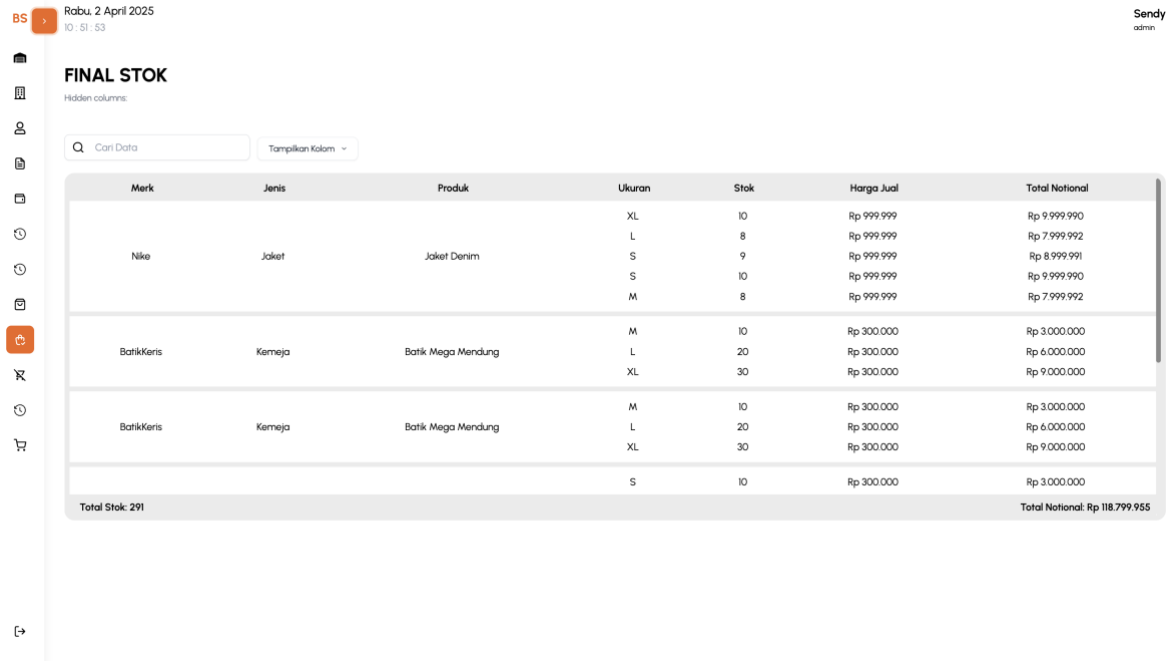
Halaman ini menampilkan implementasi Stok Barang. Sebagaimana ditunjukkan pada *Gambar 3.15* setiap produk mencantumkan detail seperti barcode, nama produk, merek, kategori, supplier (CV), harga jual, tanggal restok, ukuran, jumlah stok, dan warna. Tersedia tombol Detail dan Restok untuk setiap produk. Halaman dilengkapi fitur pencarian, filter, dan tombol Tambah Barang Baru dan tak lupa total keseluruhan stok.



Gambar 3.15 *Halaman Stok*

3.3.1.10 Implementasi Halaman Final Stok

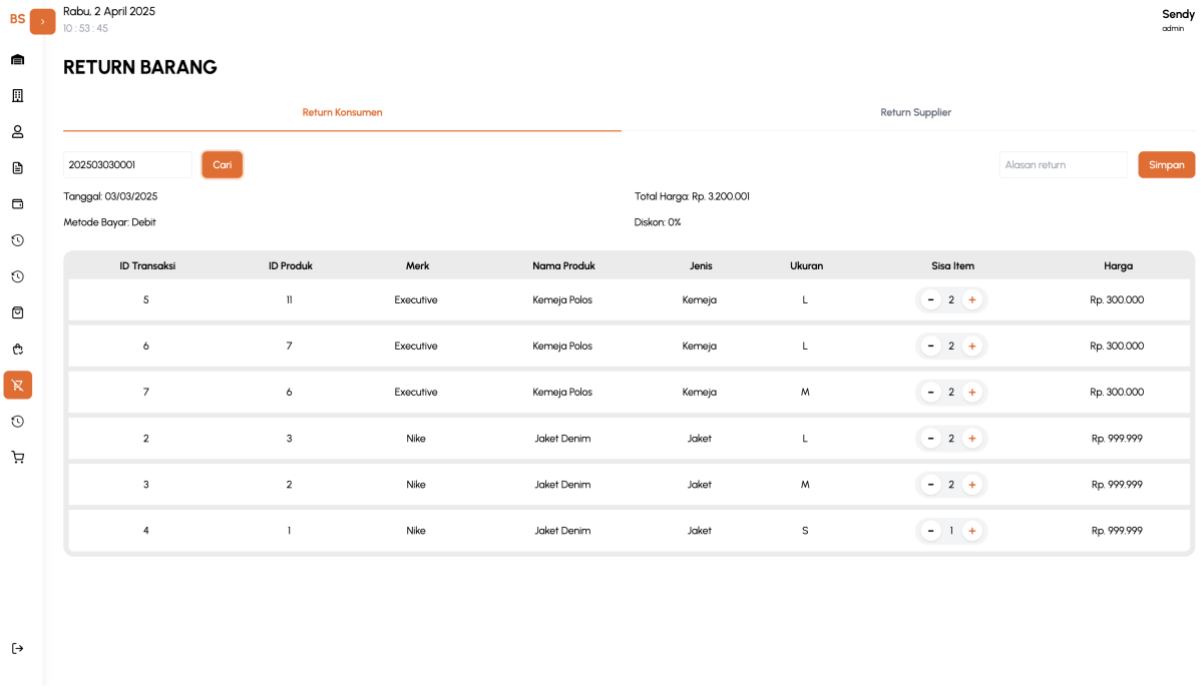
Halaman ini menampilkan implementasi Final Stok. Sebagaimana terlihat pada *Gambar 3.16,* setiap produk ditotalkan notionalnya berdasarkan harga jual mereka. Terdapat filter kolom, dimana apabila beberapa kolom dihilangkan maka susunan tabel juga akan ikut berubah mengikuti kolom yang ditampilkan.



Gambar 3.16 *Halaman Final Stok*

3.3.1.11 Implementasi Halaman Return Barang

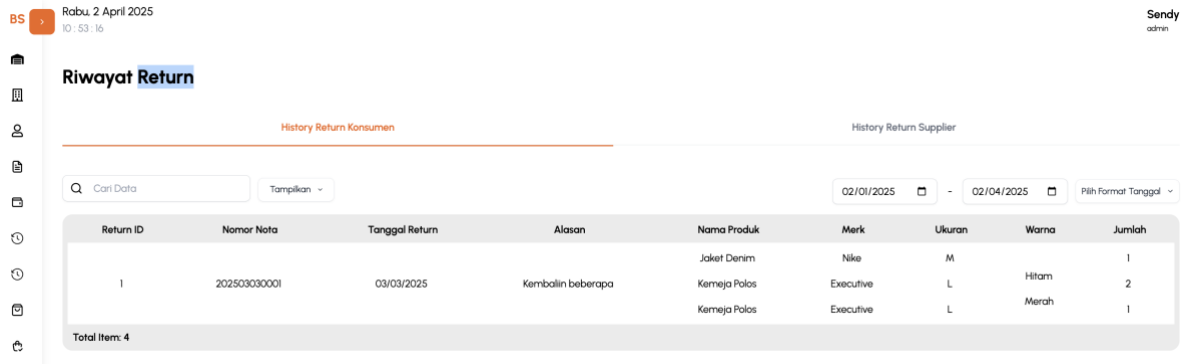
Halaman ini menampilkan proses return barang, dimana ketika dimasukkan nomor nota atau nomor restok sesuai dengan return yang diinginkan, kemudian akan ditampilkan produk-produk yang terjual pada nota tersebut. Sebagaimana ditunjukkan pada *Gambar 3.17,* selanjutnya dapat disesuaikan sisa item yang akan dibeli pada nota tersebut.



Gambar 3.17 *Halaman Return Barang*

3.3.1.12 Implementasi Halaman Riwayat Return

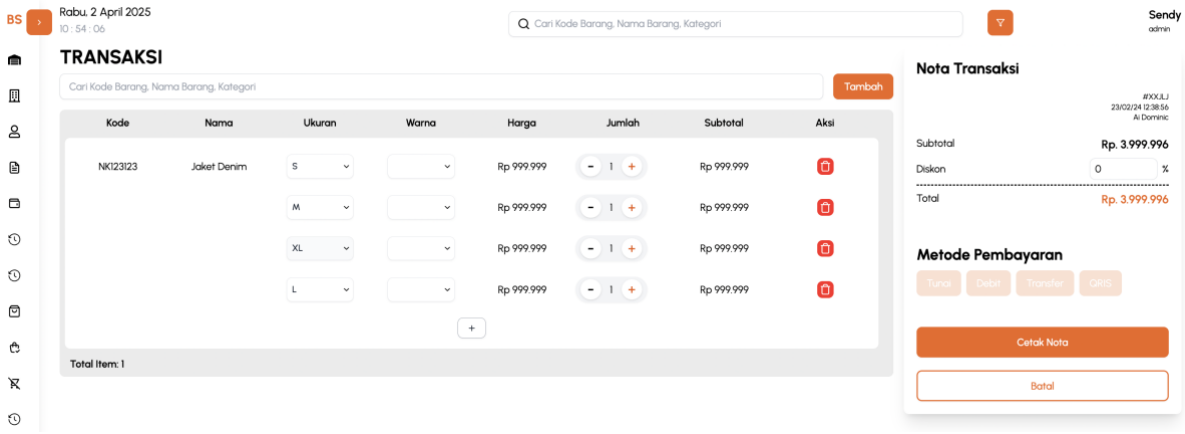
Halaman Riwayat Return ini merupakan halaman yang berisi riwayat pengembalian barang dari konsumen dan supplier. Sebagaimana terlihat pada *Gambar 3.18*, halaman ini dilengkapi dengan filter tanggal dan tabelnya menunjukkan barang apa saja dan jumlah barang yang dikembalikan.



Gambar 3.18 *Halaman Riwayat Return*

3.3.1.13 Implementasi Halaman Transaksi

Halaman ini mengimplementasikan fitur kasir, dimana kasir dapat memasukkan nomor barcode produk yang ada. Sebagaimana ditunjukkan pada *Gambar 3.19***,** pada setiap produk yang muncul dapat ditambahkan varian produk lain, kemudian dapat ditambahkan diskon dan pemilihan metode pembayaran.



Gambar 3.19 *Halaman Transaksi*

3.3.2 Implementasi Bagian Backend

Proses pertukaran informasi antara layanan backend dilakukan melalui API dengan protokol HTTP. Sebagian besar implementasi backend dikembangkan menggunakan teknologi tersebut. Rincian mengenai implementasi endpoint API yang telah disajikan dalam *Tabel 3.13* akan dijelaskan lebih lanjut pada subbab berikut.

3.3.2.1 Implementasi Endpoint API Masuk/*Login*

Implementasi API ini berfungsi untuk memungkinkan pengguna masuk ke dalam akun masing-masing melalui proses otentikasi yang aman. Pada bagian ini akan dijelaskan kode sumber utama yang digunakan dalam alur sistem login, mulai dari penerimaan data login (email dan password), validasi data, hingga pemberian token autentikasi jika data dinyatakan valid. Implementasi ini menjadi dasar dari sistem keamanan dan otorisasi pengguna, sehingga hanya pengguna yang terverifikasi yang dapat mengakses fitur-fitur yang sesuai dengan peran mereka dalam sistem.

```
1. func (s *userService) Verify(ctx context.Context, req dto.UserLoginRequest)
(dto.UserLoginResponse, error) {
 2.       check, flag, err := s.userRepo.CheckEmail(ctx, nil, req.Email)
 3.       if err != nil || !flag {
 4.                return dto.UserLoginResponse{}, dto.ErrEmailNotFound
 5.       }
 6.
 7.       checkPassword, err := helpers.CheckPassword(check.Password, []byte(req.Password))
 8.       if err != nil || !checkPassword {
 9.                return dto.UserLoginResponse{}, dto.ErrPasswordNotMatch
10.       }
11.
12.       token := s.jwtService.GenerateToken(strconv.Itoa(check.ID), check.Role)
13.
14.       s.logRepo.LogAkses(ctx, nil, check.ID)
15.
16.       return dto.UserLoginResponse{
17.                Token: token,
18.                Role:  check.Role,
19.       }, nil
```

Kode Sumber 3.1 *Implementasi untuk endpoint Masuk Akun*

Tabel 3.15 *Penjelasan Implementasi untuk endpoint Masuk Akun*

Baris	Penjelasan
1	Definisi fungsi Verify untuk melakukan verifikasi dari data akun yang berusaha masuk

2-5	Merupakan proses verifikasi email dengan memanggil fungsi check email dari lapisan repository. Dimana akan mengembalikan sebuah flag dan hasil pengecekan. Apabila terjadi error maka akan dikembalikan pesan error.
7-10	Pemanggilan fungsi untuk mengecek password, dengan mencocokan hasil hash yang disimpan dan kata sandi yang saat ini sedang digunakan untuk masuk kedalam akun terkait.
12	Pembuatan token JWT untuk akun yang sedang berusaha masuk
14	Pencatatan log akses untuk api login.
16-19	Mengembalikan hasil login berupa token dan jabatan yang dimiliki akun tersebut

3.3.2.2 Implementasi Endpoint API Pengambilan Data Cabang

Berikut merupakan implementasi API untuk pengambilan data cabang, yang bertujuan untuk menyediakan daftar cabang usaha yang terdaftar dalam sistem. API ini juga dilengkapi dengan fitur paginasi yang mengembalikan informasi mengenai jumlah total data, jumlah data per halaman, dan halaman saat ini. Hal ini memudahkan sisi frontend dalam menampilkan data secara terstruktur dan efisien, terutama saat data cabang berjumlah banyak.

```
 1. func (r *cabangRepository) GetAllCabangWithPagination(ctx context.Context, req
dto.PaginationRequest) (dto.GetAllCabangRepositoryResponse, error) {
 2.
 3.     var cabangs []entity.Cabang
 4.     var err error
 5.     var count int64
 6.
 7.     if req.PerPage == 0 {
 8.         req.PerPage = 10
 9.     }
10.
11.     if req.Page == 0 {
12.         req.Page = 1
13.     }
14.
15.     query := r.db.WithContext(ctx).Model(&entity.Cabang{}).Order("created_at ASC")
16.     if req.Search != "" {
17.         query = query.Where("name LIKE ?", "%"+req.Search+"%")
18.     }
19.
20.     err = query.Count(&count).Error
21.     if err != nil {
22.         return dto.GetAllCabangRepositoryResponse{}, err
23.     }
24.
25.     offset := (req.Page - 1) * req.PerPage
26.     maxPage := int64(math.Ceil(float64(count) / float64(req.PerPage)))
27.
28.     err = query.Offset(offset).Limit(req.PerPage).Find(&cabangs).Error
29.     if err != nil {
30.         return dto.GetAllCabangRepositoryResponse{}, err
31.     }
32.
33.     return dto.GetAllCabangRepositoryResponse{
34.         Data: cabangs,
35.         PaginationResponse: dto.PaginationResponse{
36.             Page:    req.Page,
37.             PerPage: req.PerPage,
38.             MaxPage: maxPage,
39.             Count:   count,
40.         },
41.     }, nil
42. }
43.
```

Kode Sumber 3.2 *Implementasi untuk pengambilan data cabang*

Tabel 3.16 *Penjelasan implementasi pengambilan data cabang*

Baris	Penjelasan
3-5	Mendeklarasikan variabel untuk menyimpan hasil data cabang, kemungkinan error, dan jumlah total data (count).
7-11	Menentukan nilai default untuk PerPage dan Page jika tidak diberikan oleh pengguna, yaitu 10 data per halaman dan halaman pertama.
13-17	Menyusun query dasar untuk mengambil data dari tabel cabang, diurutkan berdasarkan waktu pembuatan. Jika pengguna mengisi kolom pencarian (search), maka query akan ditambahkan kondisi filter.
18-22	Menghitung jumlah total data yang sesuai dengan filter pencarian, dan menyimpannya dalam variabel count. Jika terjadi kesalahan saat menghitung, maka proses akan dihentikan dan mengembalikan error.
24-25	Menghitung nilai offset berdasarkan halaman yang dipilih, dan menentukan total halaman (maxPage) dengan menggunakan rumus pembagian total data dengan jumlah data per halaman.
27-30	Mengambil data cabang dari basis data sesuai dengan halaman yang diminta, dengan menggunakan Offset dan Limit. Jika terjadi kesalahan saat pengambilan data, maka akan langsung dikembalikan sebagai error.
32-40	Mengembalikan hasil akhir berupa data cabang yang ditemukan, serta informasi lengkap terkait paginasi seperti halaman saat ini, jumlah data per halaman, jumlah halaman maksimum, dan total data keseluruhan.

3.3.2.3 Implementasi Endpoint API Pembuatan Data Supplier

Berikut merupakan implementasi API untuk melakukan pembuatan data supplier ke dalam sistem. Proses ini mencakup pengecekan apakah jenis dan merk barang yang dikaitkan dengan supplier tersebut sudah tersedia dalam basis data, serta otomatis menambahkan entri baru jika belum tersedia. Dengan demikian, integrasi antara data supplier dan data produk dapat berjalan secara dinamis dan mengurangi duplikasi data.

```
 1. func (s *supplierService) CreateSupplier(ctx context.Context, req dto.SupplierRequest)
(dto.SupplierResponse, error) {
 2.     _, flag, _ := s.supplierRepo.CheckSupplierName(ctx, nil, req.Name)
 3.     if flag {
 4.         return dto.SupplierResponse{}, dto.ErrSupplierAlreadyExists
 5.     }
 6.
 7.     supplierEntity := entity.Supplier{
 8.         Name:     req.Name,
 9.         NoHp:     req.NoHp,
10.         Discount: req.Discount,
11.     }
12.
13.     supplier, err := s.supplierRepo.CreateSupplier(ctx, nil, supplierEntity)
14.     if err != nil {
15.         return dto.SupplierResponse{}, err
16.     }
17.
18.     var merkResponses []dto.MerkResponse
19.     for _, merk := range req.MerkRequest {
20.         var jenisResponses []dto.JenisResponse
21.
22.         merkEntity := entity.Merk{
23.             Nama: merk.NamaMerk,
24.         }
25.
26.         merkRes, err := s.merkRepo.CreateMerk(ctx, nil, merkEntity)
27.         if err != nil {
```

```
28.             return dto.SupplierResponse{}, err
29.         }
30.
31.         if len(merk.JenisRequest) == 0 {
32.             allJenis, err := s.jenisRepo.GetAllJenis(ctx, nil)
33.             if err != nil {
34.                 return dto.SupplierResponse{}, err
35.             }
36.
37.             for _, jenis := range allJenis {
38.                 _, err = s.supplierRepo.CreateDetailMerkSupplier(ctx, nil,
entity.DetailMerkSupplier{
39.                     MerkID:     merkRes.ID,
40.                     JenisID:    jenis.ID,
41.                     SupplierID: supplier.ID,
42.                     Discount:   merk.Discount,
43.                 })
44.
45.                 if err != nil {
46.                     continue
47.                 }
48.
49.                 jenisResponses = append(jenisResponses, dto.JenisResponse{
50.                     NamaJenis: jenis.NamaJenis,
51.                 })
52.             }
53.
54.         } else {
55.             for _, jenis := range merk.JenisRequest {
56.                 jenisEntity := entity.Jenis{
57.                     NamaJenis: jenis.NamaJenis,
58.                 }
59.
60.                 jenis, err := s.jenisRepo.CreateJenis(ctx, nil, jenisEntity)
61.                 if err != nil {
62.                     continue
63.                 }
64.
65.                 _, err = s.supplierRepo.CreateDetailMerkSupplier(ctx, nil,
entity.DetailMerkSupplier{
66.                     MerkID:     merkRes.ID,
67.                     JenisID:    jenis.ID,
68.                     SupplierID: supplier.ID,
69.                     Discount:   merk.Discount,
70.                 })
71.
72.                 if err != nil {
73.                     continue
74.                 }
75.                 jenisResponses = append(jenisResponses, dto.JenisResponse{
76.                     NamaJenis: jenis.NamaJenis,
77.                 })
78.             }
79.         }
80.
81.         merkResponses = append(merkResponses, dto.MerkResponse{
82.             NamaMerk: merk.NamaMerk,
83.             Discount: merk.Discount,
84.             Jenis:    jenisResponses,
85.         })
86.     }
87.
88.     return dto.SupplierResponse{
89.         ID:       supplier.ID,
90.         Name:     supplier.Name,
91.         NoHp:     supplier.NoHp,
92.         Discount: supplier.Discount,
93.         Merk:     merkResponses,
```

```
94.     }, nil
95. }
96.
```

Kode Sumber 3.3 *Implementasi untuk pembuatan data Supplier*

Tabel 3.17 *Penjelasan implementasi pembuatan data Supplier*

Baris	Penjelasan
2-5	Pemanggilan fungsi untuk validasi nama supplier untuk menghindari data duplikat. Apabila sudah ada data tersebut atau terjadi error, maka akan dikembali pesan error.
6-11	Membuat objek **Supplier** dari data request.
12-14	Menyimpan entitas **supplier** ke basis data dan menangani kemungkinan error saat proses penyimpanan.
15-52	Melakukan iterasi terhadap setiap merk yang dikirimkan dalam request. Untuk setiap merk: Dibuat objek entitas merk. Disimpan ke dalam basis data. Jika JenisRequest kosong, maka mengambil semua jenis dari basis data (GetAllJenis). Jika JenisRequest tidak kosong, maka membuat jenis baru berdasarkan permintaan.
31-50	Jika tidak ada Jenis dalam permintaan, maka diasumsikan bahwa semua jenis yang sudah ada akan dikaitkan dengan merk tersebut melalui relasi DetailMerkSupplier.
53–77	Jika request menyertakan jenis, maka jenis baru dibuat dan juga dihubungkan dengan merk dan supplier melalui CreateDetailMerkSupplier.
78-83	Semua jenis yang telah diproses akan dikemas sebagai bagian dari MerkResponse dan disimpan dalam array merkResponses.
85-93	Mengembalikan objek SupplierResponse lengkap beserta daftar Merk dan Jenis terkait.

3.3.2.4 Implementasi Endpoint API Pengambilan Data Karyawan

API ini bertugas untuk mengambil data seluruh karyawan yang tersimpan di dalam sistem dan hanya dapat diakses oleh pengguna dengan peran admin atau owner. Akses terbatas ini memastikan keamanan dan privasi data pegawai tetap terjaga. Detail teknis mengenai alur permintaan hingga respon API akan dijelaskan melalui gambar dan tabel implementasi di bawah ini.

```
f 1. unc (r *userRepository) GetAllUserWithPagination(ctx context.Context, tx *gorm.DB, req
dto.PaginationRequest) (dto.GetAllUserRepositoryResponse, error) {
 2.     if tx == nil {
 3.         tx = r.db
 4.     }
 5.
 6.     var users []entity.User
 7.     var err error
 8.     var count int64
 9.     var countKasir int64
10.     var countStok int64
11.
12.     if req.PerPage == 0 {
13.         req.PerPage = 10
14.     }
```

```
15.     if req.Page == 0 {
16.         req.Page = 1
17.     }
18.
19.     query := tx.WithContext(ctx).Model(&entity.User{}).Where("role != ?", "owner")
20.
21.     if req.Search != "" {
22.         query = query.Where("name LIKE ?", "%"+req.Search+"%")
23.     }
24.
25.     err = query.Count(&count).Error
26.     if err != nil {
27.         return dto.GetAllUserRepositoryResponse{}, err
28.     }
29.
30.     err = tx.Model(&entity.User{}).Where("role = ?", "kasir").Count(&countKasir).Error
31.     if err != nil {
32.         return dto.GetAllUserRepositoryResponse{}, err
33.     }
34.
35.     err = tx.Model(&entity.User{}).Where("role = ?", "stok").Count(&countStok).Error
36.     if err != nil {
37.         return dto.GetAllUserRepositoryResponse{}, err
38.     }
39.
40.     offset := (req.Page - 1) * req.PerPage
41.     maxPage := int64(math.Ceil(float64(count) / float64(req.PerPage)))
42.
43.     err = query.Order("created_at ASC").Offset(offset).Limit(req.PerPage).Find(&users).Error
44.     if err != nil {
45.         return dto.GetAllUserRepositoryResponse{}, err
46.     }
47.
48.     return dto.GetAllUserRepositoryResponse{
49.         Users:          users,
50.         Total_Karyawan: count,
51.         Total_Kasir:    countKasir,
52.         Total_Stok:     countStok,
53.         Total_Admin:    count - countKasir - countStok,
54.         PaginationResponse: dto.PaginationResponse{
55.             Page:    req.Page,
56.             PerPage: req.PerPage,
57.             MaxPage: maxPage,
58.             Count:   count,
59.         },
60.     }, nil
61. }
62.
63.
```

Kode Sumber 3.4 *Implementasi pengambilan data karyawan*

Tabel 3.18 *Penjelasan implementasi pengambilan data karyawan*

Baris	Penjelasan
2-4	Fungsi GetAllUserWithPagination menerima parameter tx *gorm.DB. Jika pemanggil tidak menyediakan tx (nil), maka otomatis akan digunakan r.db (basis data utama). Ini berguna untuk menjamin konsistensi bila dipanggil di dalam transaksi yang lebih besar. Semua operasi selanjutnya akan memakai tx ini.
6-17	Pendeklarasian variabel-variabel yang akan digunakan dalam pembuatan dan penlaksanaan query
19-46	Membuat query berdasarkan variabel yang diberikan pada lapisan sebelumnya. Juga Menyusun query untuk menghitung karyawan-karyawan yang saat ini bekerja Mengatur limit pengambilan dan offset pengambilan data berdasarkan parameter

	yang ada
48-60	Menyusun data ke format response lalu mengembalikannya ke lapisan service

3.3.2.5 Implementasi Endpoint API Pembuatan Data Pengeluaran

API berikut digunakan untuk mencatat data pengeluaran uang atau petty cash yang dikelola oleh toko, dan fitur ini hanya dapat digunakan oleh pengguna dengan otoritas admin atau owner. Proses pencatatan ini akan menyimpan rincian penggunaan dana, termasuk kategori pengeluaran, jumlah uang, dan deskripsi singkat. Rincian teknis dan ilustrasi implementasi dapat dilihat pada dokumentasi berupa gambar dan tabel yang tersedia.

```
 1. func (r *pengeluaranRepository) CreatePengeluaran(ctx context.Context, pengeluaran
entity.Pengeluaran) (dto.PengeluaranResponse, error) {
 2.     err := r.db.WithContext(ctx).Create(&pengeluaran).Error
 3.     if err != nil {
 4.         return dto.PengeluaranResponse{}, err
 5.     }
 6.
 7.     return dto.PengeluaranResponse{
 8.         ID:                 pengeluaran.ID,
 9.         NamaPengeluaran:    pengeluaran.NamaPengeluaran,
10.         TipePembayaran:     pengeluaran.TipePembayaran,
11.         TanggalPengeluaran: pengeluaran.TanggalPengeluaran,
12.         Description:        pengeluaran.Description,
13.         KategoriPengeluaran: pengeluaran.KategoriPengeluaran,
14.         Jumlah:             pengeluaran.Jumlah,
15.         Tujuan:             pengeluaran.Tujuan,
16.     }, nil
17. }
18.
19.
```

Kode Sumber 3.5 *Implementasi pembuatan data pengeluaran*

Tabel 3.19 *Penjelasan implementasi pembuatan data pengeluaran*

Baris	Penjelasan
3-6	Fungsi CreatePengeluaran menerima context dan objek pengeluaran (entitas). Pada baris ini dipanggil r.db.WithContext(ctx).Create(&pengeluaran), yang akan menyimpan data pengeluaran ke dalam tabel terkait di basis data. Hasil eksekusi .Error langsung disimpan ke variabel err.
8-17	Setelah penyimpanan berhasil (error nil), dibuat objek dto.PengeluaranResponse yang memetakan semua properti penting dari entitas pengeluaran—mulai dari ID, NamaPengeluaran, TipePembayaran, TanggalPengeluaran, Description, KategoriPengeluaran, Jumlah, hingga Tujuan. Objek ini kemudian dikembalikan bersama nil untuk menandakan operasi sukses.

3.3.2.6 Implementasi Pembuatan Log Akses

Dibawah ini merupakan implementasi dari sebuah *middleware* yang berguna sebagai pencatat akses yang terjadi pada situs ini. Setiap permintaan yang terjadi akan melewati *middleware* ini, dan dicatat informasi-informasi terkait permintaan dan aktor pembuat permintaanya.

```
 1. func LogUserActivityMiddleware(logAksesService service.LogAksesService, jwtService
service.JWTService) gin.HandlerFunc {
 2.     return func(ctx *gin.Context) {
 3.
 4.         if ctx.Request.URL.Path == "/api/user/login" {
```

```
 5.             ctx.Next()
 6.             return
 7.         }
 8.
 9.         token := ctx.GetHeader("Authorization")
10.         if !strings.HasPrefix(token, "Bearer ") {
11.             response := utils.BuildResponseFailed(dto.MESSAGE_FAILED_PROSES_REQUEST,
"Authorization header does not contain Bearer", nil)
12.             ctx.AbortWithStatusJSON(http.StatusUnauthorized, response)
13.             return
14.         }
15.
16.         tokenString := strings.TrimSpace(strings.Replace(token, "Bearer ", "", 1))
17.         // Read the request body
18.         bodyBytes, err := io.ReadAll(ctx.Request.Body)
19.         if err != nil {
20.             ctx.JSON(http.StatusInternalServerError, gin.H{"error": "Unable to read request
body"})
21.             return
22.         }
23.
24.         // Restore the request body so it can be read again by subsequent handlers
25.         ctx.Request.Body = io.NopCloser(bytes.NewBuffer(bodyBytes))
26.
27.         userID, err := jwtService.GetUserIDByToken(tokenString)
28.         if err != nil {
29.             res := utils.BuildResponseFailed(dto.MESSAGE_FAILED_DENIED_ACCESS, err.Error(),
nil)
30.             ctx.JSON(http.StatusUnauthorized, res)
31.             ctx.Abort()
32.             return
33.         }
34.
35.         ctx.Next()
36.
37.         statusCode := ctx.Writer.Status()
38.         id, _ := strconv.Atoi(userID)
39.         logAksesID := logAksesService.GetActiveLogAksesForUser(ctx, id)
40.         var payload string
41.         if ctx.Request.URL.RawQuery != "" {
42.             payload = ctx.Request.URL.RawQuery
43.         } else {
44.             payload = string(bodyBytes)
45.         }
46.
47.         detailAkses := entity.DetailAkses{
48.             Activity:   ctx.Request.URL.Path,
49.             LogAksesID: logAksesID,
50.             Token:      tokenString,
51.             IP:         ctx.ClientIP(),
52.             Payload:    payload,
53.             Status:     strconv.Itoa(statusCode),
54.         }
55.
56.         logAksesService.LogDetailAkses(ctx, detailAkses)
57.     }
58. }
```

Kode Sumber 3.6 *Implementasi middleware log akses*

Tabel 3.20 *Penjelasan implementasi middleware log akses*

Baris	Penjelasan
5-8	jika path request adalah /api/user/login, middleware langsung memanggil ctx.Next() (melanjutkan ke handler berikutnya) dan tidak mencatat apa pun, agar proses login itu sendiri tidak di-log.
10-17	Mengambil bearer token atau akses token yang digunakan akun karyawan sebagai

	identifikasi akses nya
19-23	Mengambil request body dari API request tersebut
28-34	Mengambil ID User/Pengguna berdasarkan token yang telah diambil sebelumnya
38-46	Menangkap hasil eksekusi & menentukan payload
48-55	Menyusun entitas log akses untuk kemudian dicatat didalam basis data berdasarkan data-data yang telah dikumpulkan sebelumnya.

3.3.2.7 Implementasi Endpoint API Pembuatan Data Transaksi

Implementasi ini menjelaskan API untuk pembuatan transaksi penjualan yang dijalankan melalui modul kasir. Seluruh produk yang telah dimasukkan ke dalam keranjang oleh kasir akan diproses menjadi data transaksi final yang disimpan ke dalam basis data. Proses ini juga akan menyesuaikan stok produk secara otomatis dan menghasilkan struk penjualan sebagai bukti transaksi.

```
 1. func (t *transaksiService) CreateTransaksi(ctx context.Context, createTransaksi
dto.CreateTransaksi, userID string) (dto.TransaksiResponse, error) {
 2.     CountHargaBe := 0.0
 3.     for _, produk := range createTransaksi.Produks {
 4.         produkDetail, err := t.transaksiRepo.GetProdukByDetailID(ctx, nil,
produk.DetailProdukID)
 5.         if err != nil {
 6.             return dto.TransaksiResponse{}, err
 7.         }
 8.         CountHargaBe += produkDetail.HargaJual * float64(produk.JumlahProduk)
 9.     }
10.
11.     if createTransaksi.Diskon > 0 && createTransaksi.Diskon <= 100 {
12.         CountHargaBe = CountHargaBe * ((100 - createTransaksi.Diskon) / 100)
13.     }
14.
15.     if CountHargaBe != createTransaksi.TotalHarga {
16.         return dto.TransaksiResponse{}, fmt.Errorf("total price mismatch: calculated total is
%.2f, but provided total is %.2f", CountHargaBe, createTransaksi.TotalHarga)
17.     }
18.
19.     for _, produk := range createTransaksi.Produks {
20.         produkDetail, err := t.transaksiRepo.GetDetailProdukStok(ctx, nil,
produk.DetailProdukID)
21.         if err != nil {
22.             return dto.TransaksiResponse{}, err
23.         }
24.
25.         if produkDetail.Stok < produk.JumlahProduk {
26.             return dto.TransaksiResponse{}, fmt.Errorf("stock is not enough for product %d",
produk.DetailProdukID)
27.         }
28.     }
29.
30.     // Generate transaction ID as int64
31.     today := time.Now().Format("20060102") // Format as YYYYMMDD
32.     latestID, err := t.transaksiRepo.GetLatestTransaksiID(ctx, today)
33.     if err != nil {
34.         return dto.TransaksiResponse{}, err
35.     }
36.
37.     // Determine the sequence number
38.     sequence := 1
39.     if latestID != 0 {
40.         // Extract the sequence part and increment it
41.         sequence = int(latestID%10000) + 1
42.     }
```

```
43.
44.     // Create the transaction ID as int64
45.     datePrefix, _ := strconv.ParseInt(today, 10, 64)
46.     transactionID := datePrefix*10000 + int64(sequence)
47.
48.     transaksi := entity.Transaksi{
49.         ID:               transactionID, // Assign the generated ID
50.         TanggalTransaksi: time.Now(),
51.         TotalHarga:       createTransaksi.TotalHarga,
52.         MetodeBayar:      createTransaksi.MetodeBayar,
53.         CreatedBy:        userID,
54.         Diskon:           createTransaksi.Diskon,
55.     }
56.
57.     Transaksi, err := t.transaksiRepo.CreateTransaksi(ctx, nil, transaksi)
58.     if err != nil {
59.         return dto.TransaksiResponse{}, err
60.     }
61.
62.     for _, produk := range createTransaksi.Produks {
63.         detailTransaksi := entity.DetailTransaksi{
64.             JumlahProduk:   produk.JumlahProduk,
65.             TransaksiID:    Transaksi.ID,
66.             DetailProdukID: produk.DetailProdukID,
67.         }
68.
69.         _, err := t.transaksiRepo.CreateDetailTransaksi(ctx, nil, detailTransaksi)
70.         if err != nil {
71.             return dto.TransaksiResponse{}, err
72.         }
73.     }
74.
75.     return dto.TransaksiResponse{
76.         ID:               Transaksi.ID,
77.         TanggalTransaksi: Transaksi.CreatedAt,
78.         TotalHarga:       Transaksi.TotalHarga,
79.         MetodeBayar:      Transaksi.MetodeBayar,
80.         Diskon:           Transaksi.Diskon,
81.     }, nil
82. }
```

Kode Sumber 3.7 *Implementasi Service pembuatan data transaksi*

Tabel 3.21 *Penjelasan implementasi Service pengambilan data karyawan*

Baris	Penjelasan
2-17	Melakukan validasi perhitungan total harga yang telah diproses dengan diskon. Untuk mencocokan data dari frontend dan data yang diterima di backend sama
19-28	Melakukan validasi stok dari data yang dikirimkan frontend. Untuk memastikan semua barang yang dibeli valid bahwa stok produknya masih ada dan dapat dijual.
31-46	Membuat ID Transaksi dan memastikan untuk id transaksi sebelumnya tidak digunakan dalam pembuatan ID yang baru.
57-60	Memanggil fungsi dari repository untuk membuat data transaksi
62-73	Melakukan iterasi untuk membuat data detail transaksi berdasarkan produk-produk yang ada dalam transaksi tersebut
75-81	Mengembalikan data hasil pembuatan transaksi ke lapisan sebelumnya berdasarkan transaksi yang telah dibuat.

3.3.2.8 Implementasi Endpoint API Pengambilan Data Riwayat Transaksi

Berikut merupakan penjelasan tentang pengambilan data Riwayat Transaksi. Dimana pengambilan datanya dapat dipisahkan menjadi dua versi, yaitu pengambilan Riwayat berdasarkan nota dan berdasarkan produk, berikut penjelasannya.

```go
func (t *transaksiService) GetHistoryTransaksi(ctx context.Context, req dto.TransactionPaginationRequest) (any, error) {

    if req.Filter == "" || req.Filter == "produk" {
        History, err := t.transaksiRepo.GetTransaksiByProduk(ctx, nil, req)
        if err != nil {
            return nil, err
        }

        return History, nil

    } else if req.Filter == "nota" {

        transaksiData, err := t.transaksiRepo.GetTransaksiByNota(ctx, nil, req)
        if err != nil {
            return dto.GetTransaksiNotaResponse{}, err
        }

        var SumTotalProfit float64
        var detailedTransaksiList []dto.GetTransaksiNota

        for i := range transaksiData.Data {
            items := transaksiData.Data[i]
            details, err := t.transaksiRepo.GetTransaksiNotaDetail(ctx, nil, items.IDTransaksi)
            if err != nil {
                return dto.GetTransaksiNotaResponse{}, err
            }

            // Calculate total profit for this transaction
            var totalProfit float64
            for _, detail := range details {
                totalProfit += detail.TotalProfit // Summing up profit from each detail item
            }

            // Construct detailed transaction
            detailedTransaksi := dto.GetTransaksiNota{
                IDTransaksi:      items.IDTransaksi,
                TotalProduk:      items.TotalProduk,
                TotalPendapatan:  items.TotalPendapatan,
                TanggalTransaksi: items.TanggalTransaksi,
                DiskonTransaksi:  items.DiskonTransaksi,
                TotalProfit:      totalProfit, // Assign the calculated total profit
                DetailTransaksi:  details,
            }

            // Sum up total profit
            SumTotalProfit += totalProfit
            detailedTransaksiList = append(detailedTransaksiList, detailedTransaksi)
        }

        response := dto.GetTransaksiNotaResponse{
            Data:           detailedTransaksiList,
            SumTotalProfit: SumTotalProfit,
            PaginationResponse: dto.PaginationResponse{
                PerPage: req.PerPage,
                Page:    req.Page,
                MaxPage: transaksiData.PaginationResponse.MaxPage,
                Count:   transaksiData.PaginationResponse.Count,
            },
        }

        return response, nil
    }

    return nil, nil}
```

Kode Sumber 3.8 *Implementasi pengambilan Riwayat Transaksi*

***Tabel 3.22** Penjelasan implementasi pengambilan Riwayat Transaksi*

Baris	Penjelasan
3	Validasi jenis filter, yang mengecek untuk filter produk dan juga apabila parameter filter itu kosong, maka pengambilan data membuat data per produk sebagai nilai standar nya.
4-7	Pemanggilan fungsi untuk mengambil data Riwayat Transaksi per produk, apabila terjadi error pada pengambilan maka akan dikembalikan hasil error pengambilan dan data kosong.
11	Validasi jenis filter, yang mengecek untuk filter pengambilan berdasarkan nota.
13-16	Pemanggilan fungsi untuk mengambil data Riwayat Transaksi berdasarkan nota, apabila terjadi error pada pengambilan maka akan dikembalikan hasil error pengambilan dan data kosong.
18-19	Inisiasi variabel untuk menyimpan data total profit dan deret detail data transaksi per nota
21-26	Dilakukan iterasi, per data nota yang ada dan diambil dari proses sebelumnya. Dilakukan pengambilan detail produk yang ada pada tiap nota. Apabila terjadi kesalahan maka proses dihentikan dan dikembalikan error.
29-32	Dilakukan perhitungan total keuntungan dari detail produk yang ada pada Langkah iterasi saat ini.
34-43	Disusunlah data yang telah diambil dan dibuat tersebut kedalam sebuah format DTO untuk kemudian ditempelkan pada format hasil akhir dari pengambilan data ini.
46	Ditambahkan total pendapatan dari nota ini, ke sebuah variabel untuk total pendapatan keseluruhan
47	Ditempelkan data yang telah dibuat pada baris 34-43 ke deret yang diinisiasi diawal
50-61	Disusunlah seluruh data yang telah diambil dan dibuat ke sebuah format akhir, yang berisikan data per nota dan seluruh total penjumlahan pendapatan. Lalu dikembalikan ke proses yang memanggil fungsi ini.

3.3.2.9 Implementasi Endpoint API Pembuatan Stok

Disini akan dijelaskan bagaimana proses pembuatan stok atau proses pemasukan data produk kedalam sistem. Mulai dari validasi dan pembuatan merk jenis dan sebagainya akan dijelaskan pada penjelasan dibawah.

```
 1. func (s *produkService) CreateProduk(ctx context.Context, produk dto.ProdukRequest)
(dto.CreateProdukResponse, error) {
 2.     existingProduk, err := s.produkRepo.GetProdukByBarcodeID(ctx, produk.BarcodeId)
 3.     if err != nil {
 4.         return dto.CreateProdukResponse{}, err
 5.     }
 6.     if existingProduk.ID != 0 {
 7.         return dto.CreateProdukResponse{}, errors.New("product with the same barcode ID
already exists")
 8.     }
 9.
10.     DetailMerkSupply, err := s.produkRepo.GetDetailMerkSupplier(ctx, nil, produk.MerkId,
produk.JenisId, produk.SupplierId)
11.     if err != nil {
12.         return dto.CreateProdukResponse{}, err
13.     }
14.
15.     tanggalRestok := utils.ParseDate(produk.TanggalRestok)
```

```
16.
17.     produkEntity := entity.Produk{
18.         NamaProduk: produk.NamaProduk,
19.         BarcodeID:  produk.BarcodeId,
20.         CabangID:   produk.CabangId,
21.         HargaJual:  produk.HargaJual,
22.     }
23.
24.     createdProduk, err := s.produkRepo.CreateProduk(ctx, nil, produkEntity)
25.     if err != nil {
26.         return dto.CreateProdukResponse{}, err
27.     }
28.
29.     restokEntity := entity.Restok{
30.         SupplierID:    produk.SupplierId,
31.         ProdukID:      createdProduk.ID,
32.         TanggalRestok: tanggalRestok,
33.     }
34.
35.     Restok, err := s.produkRepo.CreateRestok(ctx, nil, restokEntity)
36.     if err != nil {
37.         log.Printf("Error creating Restok: %v\n", err)
38.         return dto.CreateProdukResponse{}, err
39.     }
40.
41.     var details []dto.DetailProdukResponse
42.
43.     for _, item := range produk.Detail {
44.         produkItemEntity := entity.DetailProduk{
45.             ProdukID:             createdProduk.ID,
46.             DetailMerkSupplierID: DetailMerkSupply.DetailMerkSupplierID,
47.             Ukuran:               item.Ukuran,
48.             Stok:                 item.Stok,
49.             Warna:                item.Warna,
50.             Status:               0,
51.             HargaBeli:            createdProduk.HargaJual - (createdProduk.HargaJual *
(float64(DetailMerkSupply.Discount) / 100)),
52.         }
53.
54.         createdDetailProduk, err := s.produkRepo.CreateDetailProduk(ctx, nil,
produkItemEntity)
55.         if err != nil {
56.             log.Printf("Error creating DetailProduk: %v\n", err)
57.             return dto.CreateProdukResponse{}, err
58.         }
59.
60.         detailRestok := entity.DetailRestok{
61.             Jumlah:         item.Stok,
62.             RestokID:       Restok.ID,
63.             DetailProdukID: createdDetailProduk.ID,
64.         }
65.
66.         _, err = s.produkRepo.CreateDetailRestok(ctx, nil, detailRestok)
67.         if err != nil {
68.             log.Printf("Error creating DetailRestok: %v\n", err)
69.             return dto.CreateProdukResponse{}, err
70.         }
71.
72.         detailResponse := dto.DetailProdukResponse{
73.             Ukuran:    createdDetailProduk.Ukuran,
74.             Stok:      createdDetailProduk.Stok,
75.             Warna:     createdDetailProduk.Warna,
76.             BarcodeID: createdProduk.BarcodeID,
77.         }
78.
79.         details = append(details, detailResponse)
80.     }
81.
```

```
82.     produkResponse := dto.CreateProdukResponse{
83.         ID:             createdProduk.ID,
84.         NamaProduk:     createdProduk.NamaProduk,
85.         BarcodeID:      createdProduk.BarcodeID,
86.         HargaJual:      createdProduk.HargaJual,
87.         TanggalRestok: tanggalRestok,
88.         Details:        details,
89.     }
90.
91.     return produkResponse, nil
92. }
```

Kode Sumber 3.9 *Implementasi pembuatan stok*

Tabel 3.23 *Penjelasan implementasi pembuatan stok*

Baris	Penjelasan
2-8	Validasi produk berdasarkan ID Barcode produk, validasi ini memastikan bahwa produk yang dimasukkan tidak menduplikasi produk yang telah ada didalam sistem. Apabila produk sudah ada maka akan dikembalikan pesan error untuk peringatan bahwa produk telah ada didalam sistem, dan dapat digunakan fitur restok saja
10-13	Pemanggilan fungsi untuk mengecek detail dari produk tersebut, seperti supplier, merk, dan jenis dari produk yang dimasukkan itu benar adanya. Supaya tidak terjadi pemasukan data yang tidak ada.
15	Pembuatan tanggal restok sesuai format yang diinginkan
17-22	Pembuatan entitas stok untuk nantinya digunakan untuk dimasukkan didalam basis data.
24-27	Pemanggilan fungsi dengan entitas stok pada Langkah sebelumnya sebagai parameternya, fungsi ini berfungsi untuk menjalankan query pada lapisan repository untuk dimasukkannya data stok tersebut.
29-33	Pembuatan entitas restok untuk nantinya digunakan untuk dimasukkan didalam basis data.
35-39	Pemanggilan fungsi dengan entitas restok pada Langkah sebelumnya sebagai parameternya, fungsi ini berfungsi untuk menjalankan query pada lapisan repository untuk dimasukkannya data restok tersebut.
41	Inisiasi variabel untuk menyimpan data hasil pembuatan detail produk
43	Inisiasi iterasi yang sepanjang data detail produk yang dimasukkan
44-58	Dibuatnya data entitas detail produk dari produk yang akan dimasukkan, seperti ukuran, merk, warna, dan suppliernya. Kemudian dipanggil fungsi untuk memasukkannya kedalam basis data sebagai nilai baru
60-70	Dibuatnya data entitas detail restok dari produk yang akan dimasukkan, seperti id restok yang bersangkutan dan id detail produk yang digunakan. Kemudian dipanggil fungsi untuk memasukkannya kedalam basis data sebagai nilai baru
82-89	Disusun data untuk hasil respons pembuatan produk yang disusun dari data-data hasil proses diatas, dan dikembalikan ke proses induk/pemanggil

3.3.2.10 Implementasi Endpoint API Penyetujuan Data Pending Stok

Berikut merupakan implementasi API untuk proses mengaktifkan data produk yang sebelumnya berstatus belum aktif atau "pending". Proses ini biasanya dilakukan oleh pengguna dengan otoritas tertentu (owner) setelah melakukan verifikasi terhadap kelengkapan dan kebenaran data produk. Penjelasan lengkap mengenai alur proses serta cuplikan kode sumber utama yang digunakan dalam implementasi ini akan disampaikan pada bagian berikutnya.

```
 1. func (r *produkRepository) InsertProduk(ctx context.Context, tx *gorm.DB, restokID string)
(entity.Produk, error) {
 2.
 3.     if tx == nil {
 4.         tx = r.db
 5.     }
 6.
 7.     var ids []int
 8.
 9.     if err := tx.WithContext(ctx).
10.         Table("restoks r").
11.         Select("dr.detail_produk_id").
12.         Joins("JOIN detail_restoks dr ON r.id = dr.restok_id").
13.         Where("r.id = ?", restokID).
14.         Scan(&ids).Error; err != nil {
15.         return entity.Produk{}, err
16.     }
17.
18.     if err := tx.WithContext(ctx).
19.         Table("detail_produks").
20.         Where("id IN ?", ids).
21.         Update("status", 1).Error; err != nil {
22.         return entity.Produk{}, err
23.     }
24.
25.     var produk entity.Produk
26.     if err := tx.WithContext(ctx).
27.         Table("produks p").
28.         Select("p.id, p.nama_produk, p.barcode_id, p.harga_jual").
29.         Joins("JOIN detail_produks dp ON p.id = dp.produk_id").
30.         Where("dp.id IN ?", ids).
31.         First(&produk).Error; err != nil {
32.         return entity.Produk{}, err
33.     }
34.
35.     return produk, nil
36. }
37.
```

Kode Sumber 3.10 *Implementasi penyetujuan data stok*

Tabel 3.24 *Penjelasan implementasi penyetujuan data stok*

Baris	Penjelasan
7	Inisiasi variabel array untuk menyimpan id yang akan diambil
9-16	Menjalankan query untuk mengambil id dari detail produk yang ingin disetujui data produk/stok
18-23	Menjalankan query untuk memperbarui data status dari detail produk yang id nya telah diambil dari langkah sebelumnya
25-33	Mengambil data produk dari detail produk yang telah diperbarui tadi

3.3.2.11 Implementasi Endpoint API Pengambilan Data Final Stok

Berikut akan dijelaskan tentang alur dan mekanisme pengambilan data final stok produk. Disini dilakukan pengambilan data dan perhitungan seluruh total notional dari produk-produk yang ada didalam sistem.

```
1. func (r *produkRepository) GetFinalStokProduk(ctx context.Context)
([]dto.FinalStokProdukResponse, error) {
 2.     // Step 1: Get all merks
 3.     merks, err := getAllMerks(r.db)
 4.     if err != nil {
 5.         return nil, err
 6.     }
```

```
 7.
 8.      var responses []dto.FinalStokProdukResponse
 9.
10.      // Step 2: Iterate through each merk
11.      for _, merk := range merks {
12.          // Create the response entry for the merk
13.          merkResponse := dto.FinalStokProdukResponse{
14.              Merk: merk.Nama,
15.          }
16.
17.          // Step 3: Get all jenis for the current merk
18.          jenisList, err := getJenisByMerk(r.db, merk.ID)
19.          if err != nil {
20.              return nil, err
21.          }
22.
23.          // Step 4: Populate data for each jenis
24.          for _, jenisName := range jenisList {
25.              var jenisID int
26.              r.db.Table("jenis").Select("id").Where("nama_jenis = ?",
jenisName).Scan(&jenisID)
27.
28.              jenisResponse := dto.FinalStokProdukDetailJenis{
29.                  Jenis: jenisName,
30.              }
31.
32.              // Step 5: Get all produk for the current merk and jenis
33.              produkDetails, err := getProdukDetailsByMerkAndJenis(r.db, merk.ID, jenisID)
34.              if err != nil {
35.                  return nil, err
36.              }
37.
38.              // Step 6: Group produk details by produk name
39.              produkMap := make(map[string][]dto.FinalStokProdukDetail)
40.              for _, detail := range produkDetails {
41.                  produkMap[detail.Produk] = append(produkMap[detail.Produk],
dto.FinalStokProdukDetail{
42.                      Ukuran:        detail.Ukuran,
43.                      Stok:          detail.Stok,
44.                      HargaJual:     detail.HargaJual,
45.                      TotalNotional: detail.TotalNotional,
46.                  })
47.              }
48.
49.              // Step 7: Transform the produkMap into ProdukDetail slices
50.              for produkName, details := range produkMap {
51.                  // Calculate the sum of TotalNotional for the current produk
52.                  var sumTotalNotional float64
53.                  for _, detail := range details {
54.                      sumTotalNotional += detail.TotalNotional
55.                  }
56.
57.                  // Append the produk detail to the jenisResponse
58.                  jenisResponse.Produk = append(jenisResponse.Produk,
dto.FinalStokProdukProdukDetail{
59.                      Produk:           produkName,
60.                      Detail:           details,
61.                      SumTotalNotional: sumTotalNotional,
62.                  })
63.              }
64.
65.              // Add the jenis response to the merk response
66.              merkResponse.Jenis = append(merkResponse.Jenis, jenisResponse)
67.          }
68.
69.          // Add the completed merk response to the final responses
70.          responses = append(responses, merkResponse)
71.      }
```

```
72.
73.     return responses, nil
74. }
75.
```

Kode Sumber 3.11 *Implementasi penyetujuan data stok*

Tabel 3.25 *Penjelasan implementasi penyetujuan data stok*

Baris	Penjelasan
3-6	Mengambil seluruh merk yang ada didalam basis data dan menyimpannya kedalam sebuah variabel merks
8	Membuat variabel responses yang memiliki format respon akhir dari fungsi ini
11-71	Adalah sebuah proses iterasi/looping dengan jumlah sepanjang isi dari merks yang telah diambil, proses iterasi tersebut terdiri dari : Tiap data merk yang di Langkah iterasi, dimasukkan kedalam format final respon Lalu diambil seluruh jenis yang ada pada merk tersebut, dan dilakukan iterasi kedalam array jenis tersebut Lalu diambil setiap produk yang memiliki kombinasi jenis dan merk tersebut Kemudian dari hasil data yang telah diambil, dilakukan kalkulasi untuk total notional produk tersebut Dari produk yang telah diambil tersebut, maka dimasukkan kedalam format respon akhir yang dibuat diawal
74	Data respon hasil dikembalikan ke proses induk yang memanggil

3.3.2.12 Implementasi Endpoint API Pembuatan Data Return Barang

Berikut merupakan penjelasan dari implementasi proses pengembalian atau return barang, disini akan dijelaskan untuk pengembalian barang dari konsumen kepada penjual. Untuk implementasi sisi supplier memiliki alur yang sama, tetapi perbedaanya hanya pada entitas yang diperbarui saja. Berikut penjelasannya.

```
 1. func (rs *restokService) CreateReturnUser(ctx context.Context, returnData
dto.CreateReturnUser) ([]dto.CreateReturnUserResponse, error) {
 2.     // Fetch old transaction data
 3.     oldData, err := rs.GetReturnUser(ctx, strconv.FormatInt(returnData.TransaksiID, 10))
 4.     if err != nil {
 5.         return nil, err
 6.     }
 7.
 8.     // Map old transaction details for quick lookup
 9.     oldDetailMap := make(map[int]dto.DetailReturnUser)
10.     for _, oldItem := range oldData.DetailTransaksi {
11.         oldDetailMap[oldItem.DetailTransaksiID] = oldItem
12.     }
13.
14.     // Prepare return summary list
15.     var returnSummaries []dto.ReturnSummary
16.     for _, newItem := range returnData.DetailTransaksi {
17.         oldItem, exists := oldDetailMap[newItem.DetailTransaksiID]
18.         if !exists {
19.             return nil, fmt.Errorf("invalid detail_transaksi_id: %d",
newItem.DetailTransaksiID)
20.         }
21.
22.         returnedQty := oldItem.JumlahItem - newItem.JumlahItem
23.         if returnedQty < 0 || returnedQty > oldItem.JumlahItem {
24.             return nil, fmt.Errorf("invalid return quantity for detail_transaksi_id: %d",
newItem.DetailTransaksiID)
25.         }
26.
```

```
27.         if returnedQty > 0 {
28.             // Store return details in struct
29.             returnSummaries = append(returnSummaries, dto.ReturnSummary{
30.                 DetailProdukID:    newItem.DetailProdukID,
31.                 DetailTransaksiID: newItem.DetailTransaksiID,
32.                 JumlahReturn:      returnedQty,
33.             })
34.         }
35.     }
36.
37.     // Process basis data updates
38.     if err := rs.processReturnsUser(ctx, returnSummaries, oldData.TransaksiID,
oldData.Diskon); err != nil {
39.         return nil, err
40.     }
41.
42.     // Now create the ReturnUser entity and DetailReturnUser records
43.     returnUser := entity.ReturnUser{
44.         TransaksiID: returnData.TransaksiID,
45.         Alasan:      returnData.Alasan,
46.     }
47.
48.     returnRes, err := rs.returnRepo.CreateReturnUser(ctx, returnUser)
49.     if err != nil {
50.         return nil, fmt.Errorf("failed to create return record: %v", err)
51.     }
52.
53.     // Create DetailReturnUser records
54.     for _, summary := range returnSummaries {
55.         detailReturnUser := entity.DetailReturnUser{
56.             JumlahProduk:      summary.JumlahReturn,
57.             DetailProdukID:    summary.DetailProdukID,
58.             DetailTransaksiID: summary.DetailTransaksiID,
59.             ReturnUserID:      returnRes.ID,
60.         }
61.
62.         if _, err := rs.returnRepo.CreateDetailReturnUser(ctx, detailReturnUser); err != nil
{
63.             return nil, fmt.Errorf("failed to create return detail record: %v", err)
64.         }
65.
66.     }
67.
68.     var returnResponses []dto.CreateReturnUserResponse
69.     for _, summary := range returnSummaries {
70.         oldItem, exists := oldDetailMap[summary.DetailTransaksiID]
71.         if !exists {
72.             continue
73.         }
74.
75.         returnResponses = append(returnResponses, dto.CreateReturnUserResponse{
76.             Merk:        oldItem.Merk,
77.             NamaProduk:  oldItem.NamaProduk,
78.             Jenis:       oldItem.Jenis,
79.             Ukuran:      oldItem.Ukuran,
80.             JumlahRetur: summary.JumlahReturn,
81.             HargaProduk: oldItem.HargaProduk,
82.         })
83.     }
84.
85.     return returnResponses, nil
86. }
87.
```

Kode Sumber 3.12 *Implementasi penyetujuan data stok*

Tabel 3.26 *Penjelasan implementasi penyetujuan data stok*

Baris	Penjelasan
3-6	Mengambil data transaksi dari id transaksi (id nota) yang diberikan
8-35	Memproses data transaksi tersebut, untuk dibuat kedalam format yang mencatat detail produk yang dijual dan jumlah produk tersebut. Untuk digunakan merubah jumlah yang ada di stok dan jumlah yang ada pada data transaksi
37-41	Memproses data yang telah dirangkum sebelumnya untuk memperbarui data pada basis data entitas produk dan transaksi
42-74	Mencatat hasil return yang diproses kedalam entitas return dan return detail untuk mencatat histori pengembalian barang
75-85	Membuat data respon sukses berisikan data produk yang dikembalikan dan mengembalikan data tersebut ke fungsi induk yang memanggil

3.3.2.13 Implementasi Endpoint API Pengambilan Data Riwayat Return

Berikut akan dijelaskan bagaimana implementasi pengambilan data Riwayat pengembalian atau return, setelah dijelaskan mekanisme pengembalian barang pada poin sebelumnya. Berikut adalah penjelasan kode sumber.

```
 1. func (r *returnRepository) GetReturnUserHistoryWithDetails(ctx context.Context, start string,
stop string) ([]dto.HistoryReturnUser, error) {
 2.     var result []dto.HistoryReturnUser
 3.
 4.     start = start + " 00:00:00"
 5.     stop = stop + " 23:59:59"
 6.
 7.     // Map to group details by ReturnID
 8.     historyMap := make(map[int64]*dto.HistoryReturnUser)
 9.
10.     rows, err := r.db.WithContext(ctx).Raw(`
11.         SELECT
12.             ru.id AS return_id,
13.             ru.alasan,
14.             ru.created_at AS tanggal_return,
15.             ru.transaksi_id AS nomor_nota,
16.             dru.jumlah_produk AS jumlah_return,
17.             p.nama_produk,
18.             p.barcode_id,
19.             dp.ukuran,
20.             dp.warna,
21.             m.nama AS merk
22.         FROM return_users ru
23.         LEFT JOIN detail_return_users dru ON ru.id = dru.return_user_id
24.         LEFT JOIN detail_produks dp ON dru.detail_produk_id = dp.id
25.         LEFT JOIN produks p ON dp.produk_id = p.id
26.         LEFT JOIN detail_merk_suppliers dms ON dp.detail_merk_supplier_id =
dms.detail_merk_supplier_id
27.         LEFT JOIN merks m ON dms.merk_id = m.id
28.         WHERE ru.created_at BETWEEN ? AND ?
29.     `, start, stop).Rows()
30.
31.     if err != nil {
32.         return nil, err
33.     }
34.     defer rows.Close()
35.
36.     for rows.Next() {
37.         var (
38.             returnID      int64
39.             alasan        string
40.             tanggalReturn time.Time
41.             nomorNota     string
42.             jumlahReturn  int
```

```
43.             namaProduk    string
44.             barcodeID     string
45.             ukuran        string
46.             warna         string
47.             merk          string
48.         )
49.
50.         // Scan the row
51.         err := rows.Scan(&returnID, &alasan, &tanggalReturn, &nomorNota, &jumlahReturn,
&namaProduk, &barcodeID, &ukuran, &warna, &merk)
52.         if err != nil {
53.             return nil, err
54.         }
55.
56.         // Check if the return_id already exists in the map
57.         if _, exists := historyMap[returnID]; !exists {
58.             historyMap[returnID] = &dto.HistoryReturnUser{
59.                 ReturnID:      returnID,
60.                 Alasan:        alasan,
61.                 TanggalReturn: tanggalReturn,
62.                 NomorNota:     nomorNota,
63.             }
64.         }
65.
66.         // Add detail if available
67.         if namaProduk != "" { // Only add if there's a detail
68.             historyMap[returnID].DetailHistoryReturnUser =
append(historyMap[returnID].DetailHistoryReturnUser, dto.DetailHistoryReturnUser{
69.                 NamaProduk:   namaProduk,
70.                 BarcodeId:    barcodeID,
71.                 JumlahReturn: jumlahReturn,
72.                 Merk:         merk,
73.                 Ukuran:       ukuran,
74.                 Warna:        warna,
75.             })
76.         }
77.     }
78.
79.     // Convert map to slice
80.     for _, v := range historyMap {
81.         result = append(result, *v)
82.     }
83.
84.     return result, nil
85. }
86.
```

Kode Sumber 3.13 *Implementasi penyetujuan data stok*

Tabel 3.27 *Penjelasan implementasi penyetujuan data stok*

Baris	Penjelasan
2-8	Menginisiasi variabel result untuk hasil akhir dan mengatur untuk nilai start dan stop dimulai pada detik, menit, dan jam 00:00:00, dan juga membuat variabel berbentuk map yang menggunakan key intejer untuk menyimpan informasi di proses selanjutnya.
10-33	Membuat query untuk pengambilan data dan menyimpan query tersebut dalam variabel rows
36-54	Menjalankan query tersebut dan membuat nama kolom yang ingin diambil dan kemudian menyimpan data hasil query tersebut dalam map yang sebelumnya dibuat
67-76	Menambahkan detail produk yang dikembalikan dalam map return tersebut
79-84	Hasil map tadi, dikonversi menjadi slice terlebih dahulu, sebelum dikembalikan ke fungsi induk yang memanggil

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini, dilakukan pengujian terhadap sistem yang telah dikembangkan untuk memastikan bahwa website berfungsi sesuai dengan kebutuhan dan spesifikasi yang telah ditetapkan. Pengujian mencakup *functional testing* untuk mengevaluasi apakah setiap fitur berjalan dengan baik, *usability testing* untuk menilai pengalaman pengguna dalam berinteraksi dengan antarmuka sistem, serta *performance testing* guna mengukur kinerja website dalam berbagai kondisi penggunaan. Hasil dari pengujian ini akan digunakan sebagai dasar untuk mengevaluasi kualitas sistem serta mengidentifikasi perbaikan yang diperlukan sebelum implementasi akhir.

4.1 Functional Testing

Functional test dalam pengujian perangkat lunak bertujuan untuk memastikan bahwa setiap fitur dalam sistem bekerja sesuai dengan spesifikasi yang telah ditetapkan. Pengujian ini dilakukan dengan memberikan input tertentu dan memeriksa apakah output yang dihasilkan sesuai dengan yang diinginkan. Dalam pengujian ini, digunakan metode black box, yaitu pendekatan yang menilai fungsionalitas perangkat lunak tanpa memperhatikan struktur internal atau kode programnya.

4.1.1. Pengujian Melakukan Masuk/Login

Pengujian melakukan masuk atau *login* akun merupakan pengujian kemapuan sistem dalam menangani proses *login* akun pengguna. Rincian kasus pengujian dapat dilihat pada tabel 4.1 dibawah

Tabel 4.1 *Skenario pengujian masuk akun*

Kode Uji	FT-001
Nama	Melakukan masuk kedalam akun
Kondisi Awal	Pengguna belum masuk kedalam akun
Skenario 1 – Penguji Melakukan Login Dengan Mengisi Formulir Login	
Data Uji	“email”: “Sendy1234@gmail.com” “password”: “Sendy1234”,
Prosedur Pengujian	1. Penguji membuka halaman login 2. Penguji memasukkan data uji kedalam formulir 3. Penguji menekan tombol login
Hasil Yang Diinginkan	1. Muncul *Toast* sukses apabila berhasil 2. Muncul *Toast* error apabila gagal atau ada kesalahan data 3. Aplikasi beralih ke halaman awal dan menampilkan halaman yang dapat diakses sesuai dengan jabatan yang dimiliki
Hasil Yang Diperoleh	1. Berhasil masuk kedalam akun 2. Berhasil berpindah ke halaman yang diinginkan 3. Berhasil menampilkan halaman sesuai dengan jabatan yang digunakan
Kesimpulan Pengujian	Berhasil

4.1.2. Pengujian Membuat Data Karyawan

Pengujian melakukan pembuatan data karyawan, merupakan pengujian kemampuan sistem dalam menangani proses pembuatan data karyawan sekaligus membuat akun pengguna. Rincian kasus pengujian dapat dilihat pada Tabel 4.2 dibawah.

Tabel 4.2 *Skenario pengujian pembuatan data karyawan*

Kode Uji	FT-002
Nama	Melakukan pembuatan data karyawan
Kondisi Awal	Data karyawan belum dibuat
Skenario 1 – Penguji Melakukan Pengisian Data Dengan Mengisi Formulir	
Data Uji	"nik": "123456789013432", "name": "testing", "email": "testing@example.com", "password": "nasigoreng", "no_hp": "081234567890", "role": "kasir", "tanggal_masuk": "2024-09-28T17:00:00.000Z", "tempat_lahir": "Jakarta", "tanggal_lahir": "2024-09 28T17:00:00.000Z", "alamat": "Jl. Merdeka No. 1, Jakarta"
Prosedur Pengujian	1. Penguji memasukkan tiap data pada masing-masing formulir 2. Penguji kemudian melanjutkan dengan membuat kata sandi untuk data karyawan yang telah dimasukkan tersebut 3. Apabila dirasa sudah benar, maka penguji dapat menekan tombol submit dan data akan disimpan.
Hasil Yang Diinginkan	1. Muncul *Toast* sukses apabila berhasil 2. Muncul *Toast* error apabila gagal atau ada kesalahan data 3. Data karyawan berhasil ditambahkan dan Kembali ke halaman awal bagian karyawan
Hasil Yang Diperoleh	1. Data karyawan berhasil ditambahkan 2. Berhasil berpindah ke halaman awal bagian karyawan
Kesimpulan Pengujian	Berhasil

4.1.3. Pengujian Membuat Supplier

Pengujian melakukan pembuatan data supplier, merupakan pengujian kemampuan sistem dalam menangani proses pembuatan data supplier yang cukup rumit, karena diperlukan pengecekan dan pembuatan data merk, kategori, dan diskon masing-masing. Rincian kasus pengujian dapat dilihat pada Tabel 4.3 dibawah.

Tabel 4.3 *Skenario pengujian pembuatan data supplier*

Kode Uji	FT-003
Nama	Melakukan pembuatan supplier
Kondisi Awal	Data supplier dan merk belum dibuat
Skenario 1 – Penguji Melakukan Pengisian Data Dengan Formulir	
Data Uji	"name": "Supplier Testing", "no_hp": "08123456789", "discount": 0, "merk_request": [{ "nama_merk": "Ecinos",

	"discount_merk": 25,
	"jenis_request": [
	{
	"nama_jenis": "Celana Panjang"
	},
	{
	"nama_jenis": "Celana Pendek"
	}
	]
	},
	{
	"nama_merk": "Executive",
	"discount_merk": 50,
	"jenis_request": [
	{
	"nama_jenis": "Kaos"
	},
	{
	"nama_jenis": "Kemeja"
	}
	]
	}
	]

Prosedur Pengujian	1. Penguji masuk ke halaman supplier 2. Penguji memilih opsi menambah supplier 3. Penguji memasukkan data supplier 4. Penguji mengisi daftar merk yang diinginkan 5. Penguji mengisi jenis yang berhubungan dengan tiap merk yang disuplai 6. Penguji mengisikan diskon berdasarkan merk-merk tersebut
Hasil Yang Diinginkan	1. Muncul Toast sukses apabila berhasil 2. Muncul Toast error apabila gagal atau ada kesalahan data 3. Data berhasil dimasukkan 4. Data Merk-Jenis-Diskon berhasil dimasukkan dan sesuai dengan *input*
Hasil Yang Diperoleh	1. Data supplier berhasil dimasukkan dan muncul toast sukses
Kesimpulan Pengujian	Berhasil

4.1.4. Pengujian Membuat Stok

Pengujian melakukan pembuatan data stok barang, merupakan pengujian kemampuan sistem dalam menangani proses pembuatan data stok yang dibagi menjadi dua tipe, yaitu produk baru dan produk lama. Rincian kasus pengujian dapat dilihat pada Tabel 4.4 dibawah.

Tabel 4.4 *Skenario pengujian pembuatan data stok barang*

Kode Uji	FT-004
Nama	Melakukan pengisian stok
Kondisi Awal	Stok belum dimasukkan

Skenario 1 – Penguji Memasukkan Data Stok Baru	

Data Uji	"barcode_id" : "1231AKSSDD",
	"nama_produk": "Kaos Lengan Pendek",
	"merk_id": 1,
	"supplier_id": 1,
	"cabang_id": 1,
	"jenis_id": 1,
	"tanggal_restok": "2024-12-13",
	"harga_jual": 250000,
	"detail": [
	{
	"ukuran_produk": "S",
	"stok_produk": 30,
	"warna_produk":""
	},
	{
	"ukuran_produk": "S",
	"stok_produk": 20,
	"warna_produk":""
	}
	]

Prosedur Pengujian	1. Penguji memilih opsi menambahkan stok baru 2. Penguji mengisikan data produk secara lengkap 3. Penguji mengisi jumlah produk beserta ukuran dan warna
Hasil Yang Diinginkan	1. Aplikasi memunculkan *toast* sukses apabila berhasil 2. Aplikasi memunculkan *toast* error apabila gagal 3. Aplikasi Kembali ke halaman awal dari halaman produk 4. Data produk dapat tersimpan dengan sesuai
Hasil Yang Diperoleh	1. Aplikasi memunculkan *toast* sesuai dengan kondisi 2. Aplikasi Kembali ke halaman yang sesuai 3. Data tersimpan dengan sesuai
Kesimpulan Pengujian	Berhasil
Skenario 2 – Penguji Memasukkan Data Stok lama	
Data Uji	"produk_id": 4, "merk_id": 2, "supplier_id": 1, "cabang_id": 1, "jenis_id": 4, "tanggal_restok": "2024-11-24", "harga_jual": 250000.00, "details": [{ "ukuran_produk": "XXL", "stok_produk": 70, "warna_produk":"merah" }, {

	"ukuran_produk": "XXL", "stok_produk": 70, "warna_produk":"lilac" }]
Prosedur Pengujian	1. Penguji memilih opsi menambahkan stok lama 2. Penguji memilih data produk yang ingin di tambah stoknya 3. Penguji mengisi jumlah produk beserta ukuran dan warna
Hasil Yang Diinginkan	1. Aplikasi memunculkan *toast* sukses apabila berhasil 2. Aplikasi memunculkan *toast* error apabila gagal 3. Aplikasi Kembali ke halaman awal dari halaman produk 4. Data produk dapat tersimpan dengan sesuai
Hasil Yang Diperoleh	1. Aplikasi memunculkan *toast* sesuai dengan kondisi 2. Aplikasi Kembali ke halaman yang sesuai 3. Data tersimpan dengan sesuai
Kesimpulan Pengujian	Berhasil

4.1.5. Pengujian Menyetujui Pending Stok

Pengujian melakukan penyetujuan stok pending, dimana pemilik harus menyetujui hasil input restok pada poin sebelumnya yang dilakukan oleh karyawan. Rincian kasus pengujian dapat dilihat pada Tabel 4.5 dibawah.

Tabel 4.5 *Skenario pengujian penyetujuan pending stok*

Kode Uji	FT-005
Nama	Melakukan Penyetujuan Stok Pending
Kondisi Awal	Stok masih berstatus pending (tidak bisa dijual)
Skenario 1 – Penguji Melakukan Penyetujuan Produk	
Data Uji	-
Prosedur Pengujian	1. Penguji melakukan navigasi ke halaman pending stok 2. Penguji memilih gelombang restok yang telah dilakukan 3. Penguji memilih opsi setuju 4. Data stok otomatis diperbarui dan berhasil dimasukkan kedalam stok
Hasil Yang Diinginkan	1. Aplikasi memunculkan *toast* sukses apabila berhasil 2. Aplikasi memunculkan *toast* error apabila gagal 3. Data stok diperbarui dengan sesuai
Hasil Yang Diperoleh	1. Aplikasi memunculkan *toast* sesuai dengan kondisi 2. Data diperbarui dengan sesuai
Kesimpulan Pengujian	Berhasil

4.1.6. Pengujian Membuat Penjualan

Pengujian melakukan pembuatan data penjualan/transaksi, yang mana data pembuatannya dapat berlapis-lapis dengan banyak produk. Rincian kasus pengujian dapat dilihat pada Tabel

4.6 dibawah.

Tabel 4.6 *Skenario pengujian pembuatan data transaksi/penjualan*

Kode Uji	FT-006
Nama	Melakukan penjualan barang
Kondisi Awal	Penjualan belum dicatat
Skenario 1 – Penguji Melakukan Penjualan Barang Pada Halaman Transaksi	

Data Uji	"metode_bayar": "QRIS",
	"total_harga": 8250000.00,
	"diskon": 0,
	"produks": [
	{
	"detail_produk_id": 15,
	"jumlah_produk": 10
	},
	{
	"detail_produk_id": 16,
	"jumlah_produk": 10
	},
	{
	"detail_produk_id": 17,
	"jumlah_produk": 10
	}
	]
	}

Prosedur Pengujian	1. Penguji melakukan navigasi ke halaman transaksi 2. Penguji memasukkan kode barcode dengan secara manual atau menggunakan scanner 3. Penguji memilih jumlah produk yang ingin dibeli 4. Penguji menambahkan variasi dari produk yang telah ditambahkan 5. Penguji memilih jumlah diskon dan metode pembayaran 6. Penguji mencetak nota pembayaran
Hasil Yang Diinginkan	1. Muncul *toast* sesuai dengan kondisi yang terjadi 2. Data produk dapat muncul secara akurat sesuai dengan kode barcode yang diberikan 3. Nota pembayaran dapat tercetak dengan sesuai
Hasil Yang Diperoleh	1. *Toast* muncul sesuai dengan kondisi yang terjadi 2. Data produk tampil dengan akurat 3. Nota tercetak dengan sesuai
Kesimpulan Pengujian	Berhasil

4.1.7. Pengujian Melakukan Retur

Pengujian ini dilakukan untuk memastikan bahwa proses pembuatan retur barang berjalan dengan benar sesuai dengan logika bisnis yang telah ditentukan. Dalam pengujian ini, perlu diperhatikan perubahan yang terjadi pada data penjualan, data stok barang yang dikembalikan, serta histori restok yang terhubung dengan transaksi tersebut. Rincian lebih lanjut mengenai

skenario dan hasil pengujian dapat dilihat pada *Tabel 4.7* di bawah ini sebagai dokumentasi dari proses validasi sistem.

***Tabel 4.7** Skenario pengujian pembuatan retur barang*

Kode Uji	FT-007
Nama	Melakukan pengembalian barang
Kondisi Awal	Stok dan riwayat transaksi/restok belum diperbarui
Skenario 1 – Penguji Melakukan Retur Konsumen	

Data Uji	"id_transaksi": 202502030003,
	"alasan": "Cacat",
	"detail_transaksi": [
	{
	"detail_transaksi_id": 52,
	"detail_produk_id": 15,
	"jumlah_item": 2
	},
	{
	"detail_transaksi_id": 53,
	"detail_produk_id": 16,
	"jumlah_item": 2
	},
	{
	"detail_transaksi_id": 54,
	"detail_produk_id": 17,
	"jumlah_item": 2
	}
	]

Prosedur Pengujian	1. Penguji melakukan navigasi ke halaman retur 2. Penguji memilih opsi retur konsumen 3. Penguji memasukkan nomor nota/transaksi 4. Penguji menyesuaikan jumlah produ k yang akan di retur 5. Penguji mengisi alasan retur 6. Penguji mengsubmit data yang telah diisikan
Hasil Yang Diinginkan	1. Muncul *toast* sesuai dengan kondisi yang terjadi 2. Data Transaksi diperbarui sesuai dengan hasil retur 3. Data Stok diperbarui sesuai dengan hasil retur
Hasil Yang Diperoleh	1. *Toast* berhasil muncul sesuai dengan kondisi 2. Data Transaksi berhasil diperbarui 3. Data Stok berhasil diperbarui
Kesimpulan Pengujian	Berhasil
Skenario 1 – Penguji Melakukan Retur Supplier	
Data Uji	"restok_id": 16, "alasan": "Barang rusak", "detail_transaksi": [{ "detail_restok_id": 29, "detail_produk_id": 32,

	"jumlah_item": 10
	},
	{
	"detail_restok_id": 30,
	"detail_produk_id": 33,
	"jumlah_item": 10
	}
	]

Prosedur Pengujian	1. Penguji melakukan navigasi ke halaman retur 2. Penguji memilih opsi retur Supplier 3. Penguji memasukkan nomor nota/transaksi 4. Penguji menyesuaikan jumlah produ k yang akan di retur 5. Penguji mengisi alasan retur 6. Penguji mengsubmit data yang telah diisikan
Hasil Yang Diinginkan	1. Muncul *toast* sesuai dengan kondisi yang terjadi 2. Data Restok diperbarui sesuai dengan hasil retur 3. Data Stok diperbarui sesuai dengan hasil retur
Hasil Yang Diperoleh	1. *Toast* berhasil muncul sesuai dengan kondisi 2. Data Restok berhasil diperbarui 3. Data Stok berhasil diperbarui
Kesimpulan Pengujian	Berhasil

4.2 Usability Testing

Usability test adalah metode yang digunakan untuk mengevaluasi seberapa mudah dan efisien pengguna dapat menggunakan sistem berbasis web yang telah dikembangkan. Tujuan utama dari pengujian ini adalah untuk mengidentifikasi kendala dalam penggunaan sistem, memahami pengalaman pengguna, serta mengukur tingkat kepuasan mereka terhadap sistem.

Pengujian dilakukan menggunakan metode System Usability Scale (SUS) karena metode ini telah terbukti efektif dalam mengevaluasi aplikasi berbasis web. Proses pengujian melibatkan partisipan pengguna nyata, dalam hal ini yang seharusnya mengisi kuisioner SUS adalah karyawan toko XYZ (sebagai user utama). Tetapi karena keterbatasan akses dan kesibukan yang dialami oleh klien, partisipan yang digunakan adalah mahasiswa ITS yang diambil secara acak dari berbagai jurusan, baik itu Sistem Informasi, Teknik Informatika, maupun jurusan lain. Partisipan diminta menyelesaikan sejumlah tugas umum dalam sistem seperti membuat data supplier, data stok, dan mencoba membuat atau mencatat transaksi.

Tabel 4.8 *Tabel pertanyaan kuisioner SUS*

Kode	Pertanyaan
Q1	Saya merasa sistem ini terlalu rumit
Q2	Saya merasa sistem ini terlalu lambat
Q3	Saya merasa banyak hal yang tidak konsisten
Q4	Saya membutuhkan bantuan dalam menggunakan sistem ini
Q5	Saya tidak mengalami bug dan error saat mengoperasikan sistem
Q6	Saya rasa sistem sudah terintegrasi dengan baik
Q7	Saya rasa sistem sudah responsif saat digunakan
Q8	Navigasi pada sistem sudah jelas dan mudah digunakan

Q9	Saya rasa proses dan alur dari tiap modul sistem sudah efisien
Q10	Saya merasa puas dengan pengalaman saya menggunakan sistem ini

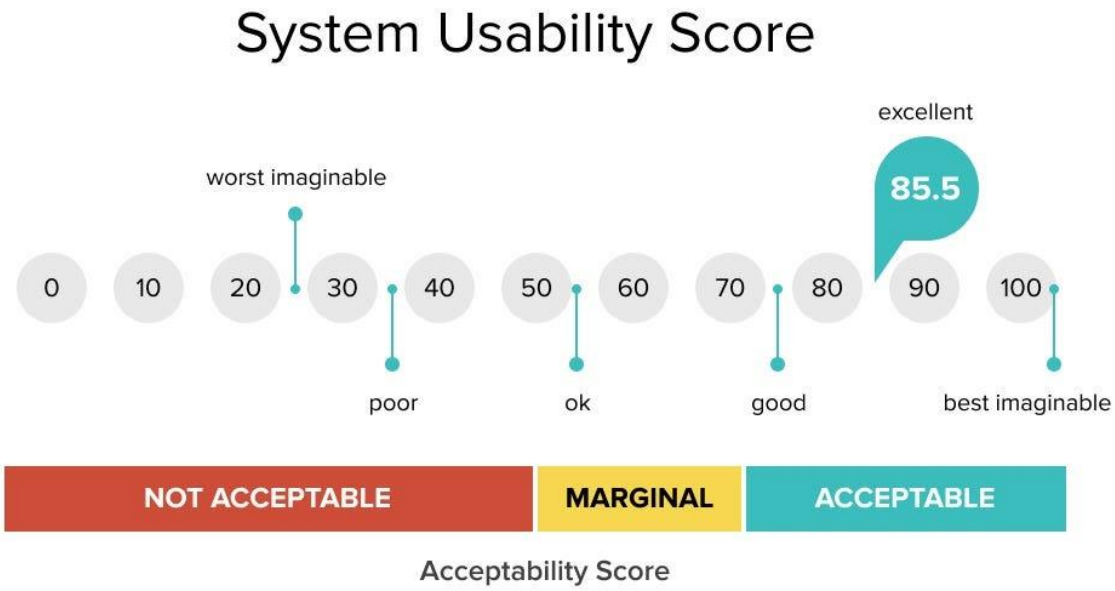
Dari kuisioner SUS yang telah disebar dan dilakukan, didapatkan *36 data partisipan* yang mengisi dan mengerjakan kuisioner SUS. Data partisipan bersifat anonim sehingga tidak akan dianalisa pada buku ini, tetapi persebarannya terbagi pada rentang umur mahasiswa dari beberapa kampus di Surabaya. Juga tidak terbatas pada jurusan teknik informatika, melainkan juga jurusan-jurusan non teknik ataupun non IT. Berikut data hasil yang telah diolah dengan pengkalian bobot jumlah poin masing-masing.

Tabel 4.9 *Hasil evaluasi sus terhadap 36 responden acak*

Kode Partisipan	Kode Pertanyaan										Total	Skor Akhir
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		
P1	2	1	1	1	5	5	4	5	5	5	38	95
P2	2	3	2	4	4	4	4	5	5	5	30	75
P3	1	1	1	1	5	5	5	5	5	5	40	100
P4	4	4	4	3	3	3	4	4	4	4	21	52,5
P5	4	2	3	4	3	4	4	4	4	2	22	55
P6	3	3	4	5	3	3	3	3	4	5	20	50
P7	2	2	3	2	2	3	3	3	2	2	20	50
P8	3	2	3	2	3	2	3	3	2	3	20	50
P9	2	1	2	1	5	5	5	4	5	5	37	92,5
P10	2	2	2	3	4	4	4	4	4	4	29	72,5
P11	3	2	2	4	2	3	3	4	3	4	22	55
P12	2	2	1	2	4	4	5	5	5	5	35	87,5
P13	3	3	3	3	3	3	3	3	3	3	20	50
P14	1	3	2	2	1	4	5	5	4	3	28	70
P15	4	3	3	4	4	4	4	5	5	4	26	65
P16	3	2	2	3	4	4	4	5	4	4	29	72,5
P17	2	2	2	1	1	4	5	4	4	4	29	72,5
P18	2	3	2	2	4	4	4	4	4	3	28	70
P19	4	3	4	5	4	4	5	3	4	4	22	55
P20	1	1	1	1	5	5	5	5	5	5	40	100
P21	2	3	2	2	3	4	4	4	3	3	26	65
P22	1	1	1	1	5	5	5	5	5	5	40	100
P23	2	2	1	4	4	5	5	4	5	4	32	80
P24	1	2	2	3	4	4	5	4	5	4	32	80
P25	2	2	3	4	4	4	5	4	4	4	28	70
P26	2	2	2	1	2	3	4	4	4	4	28	70
P27	2	2	2	2	4	4	4	4	4	4	30	75
P28	1	1	1	1	5	5	5	5	5	5	40	100
P29	3	2	2	4	5	4	5	5	5	5	32	80
P30	1	2	1	2	3	5	5	4	4	4	33	82,5

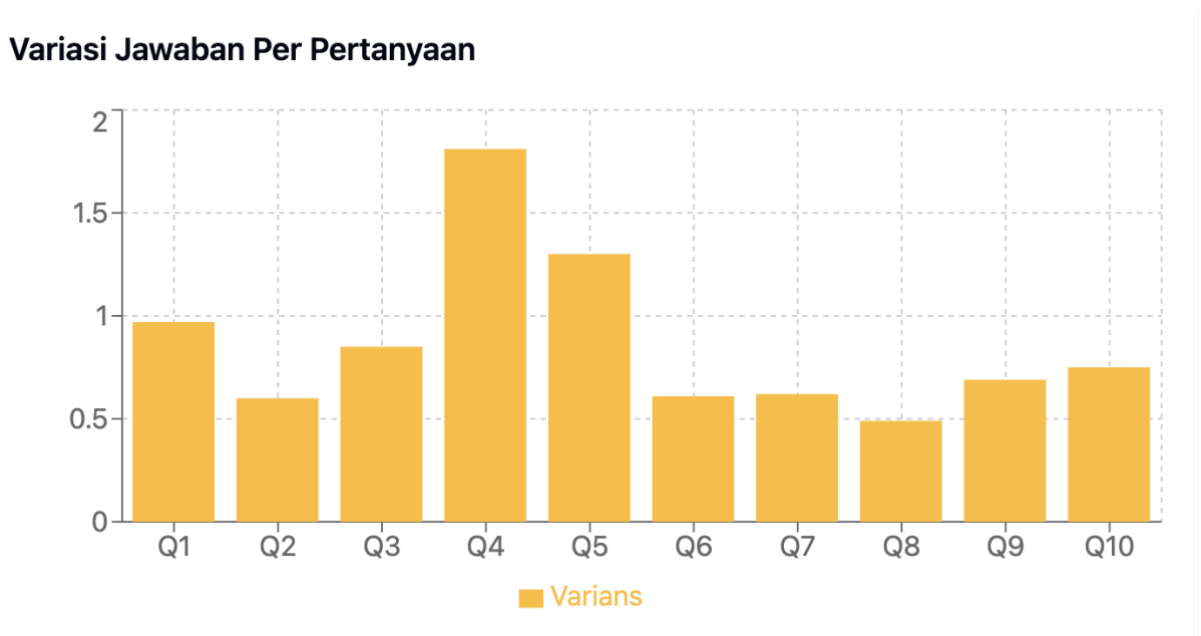
P31	3	1	1	5	5	5	5	5	5	5	34	85
P32	3	3	3	4	3	4	3	4	4	3	22	55
P33	2	2	2	3	3	4	4	4	4	4	28	70
P34	1	1	1	1	5	5	5	5	5	5	40	100
P35	4	3	3	5	3	3	3	3	3	3	17	42,5
P36	1	2	1	1	2	4	3	4	4	4	30	75
Rata-rata skor											**29,1**	**72,8**

Dari uji SUS yang telah dilakukan terhadap 36 responden, didapatkan hasil skor SUS sebesar 72,8. Skor ini masuk dalam kategori baik atau memuaskan seperti yang dapat kita lihat dari rentang nilai dibawah ini.



Gambar 4.1 *Rentang nilai evaluasi SUS*

Nilai yang diperoleh menunjukkan bahwa sistem memiliki tingkat kegunaan yang memuaskan dan berada di atas rata-rata industri untuk aplikasi berbasis web sejenis. Hasil ini menjadi indikator positif bahwa sistem yang dikembangkan telah memenuhi ekspektasi dasar penggunaan dan dapat diterima oleh pengguna akhir sebagai alat yang efektif untuk mengelola proses bisnis Toko XYZ.



Gambar 4.2 *Grafik Varians jawaban tiap pertanyaan SUS*

Namun pada pertanyaan Q4 mendapatkan nilai Varians paling tinggi mengindikasikan bahwa jawaban sangat beragam, dan nilai skor yang didapat mengidikasikan bahwa pengguna membutuhkan bantuan dalam menggunakan sistem. Hal ini dapat dikarenakan oleh beberapa hal yaitu

1. Kurangnya Onboarding yang Efektif, Responden hanya diberikan flowchart alur dan penjelasan berupa paragraf, tanpa demonstrasi atau tutorial interaktif yang komprehensif. Hal ini menyebabkan pengguna merasa tidak memiliki panduan yang cukup saat pertama kali berinteraksi dengan sistem. Tentunya untuk klien akan diberikan panduan penuh baik itu berupa video dan teks/dokumen.
2. Kompleksitas Alur Kerja, Sistem yang dikembangkan memiliki beberapa modul terintegrasi dengan alur kerja yang mungkin tidak intuitif bagi pengguna yang belum familiar dengan proses bisnis retail. Kompleksitas ini terlihat dari implementasi fitur seperti "pending stok" dan proses multi-langkah dalam manajemen supplier.
3. Terminologi Teknis, Penggunaan istilah spesifik dalam domain (seperti "pending stok", "detail merk supplier", "retur supplier") tanpa penjelasan kontekstual yang memadai dapat menyulitkan pengguna yang tidak memiliki pengetahuan dasar tentang sistem.

Kemudian juga setelah dilakukan pengujian terhadap responden acak, penulis juga mendapat hasil pengujian dari *end user* sistem itu sendiri. Yang mana pengujian dilakukan kepada 4 karyawan utama yang mengurus stok dan kasir. Karyawan-karyawan tersebut dibekali dengan pelatihan langsung oleh pemilik dan adanya panduan manual untuk penggunaan. Berikut hasil dari pengujian tersebut. Berikut merupakan hasil dari evaluasi SUS yang telah dilakukan dan mendapatkan nilai yang sangat memuaskan yakni di angka 81,8.

Tabel 4.10 *Hasil Evaluasi SUS terhadap 4 karyawan toko*

Kode Partisipan	Kode Pertanyaan										Total	Skor Akhir
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10		
P1	3	2	2	4	5	4	5	5	5	5	32	80
P2	1	2	1	2	3	5	5	4	4	4	33	82,5

P3	3	1	1	5	5	5	5	5	5	5	34	85
P4	1	2	2	3	4	4	5	4	5	4	32	80
Rata-rata skor											**32,7**	**81,8**

4.3 Uji Performa Sistem

Pengujian performa dilakukan untuk mengevaluasi kecepatan dan kestabilan sistem berbasis web ini dalam kondisi jaringan dan perangkat yang berbeda. Pengujian dilakukan dengan menggunakan fitur *Performance Test* pada *Google Chrome DevTools*, yang menggunakan tiga parameter utama yang diuji dalam pengujian ini adalah LCP, INP, dan CLS. Berikut adalah parameter-parameter yang digunakan dalam uji performa sistem ini.

1. Api Service yang digunakan di deploy pada VPS dengan spesifikasi 2 core CPU dan 8 GB RAM, hal ini dikarenakan home server klien belum dipasang saat buku ini dibuat
2. Uji coba dilakukan pada modul/proses yang dirasa paling berat
3. Uji coba dilakukan menggunakan laptop penulis, yaitu macbook pro 14 inci

Dengan harapan uji performa ini mendapatkan nilai dibawah batas toleransi masing-masing yang mana nilai toleransi tersebut telah digunakan pacuan oleh Google dan banyak developer lainnya, berikut adalah jarak nilai performa yang diinginkan untuk tiap uji coba.

Tabel 4.11 *Tabel rentang nilai parameter yang diuji*

Uji Coba	Batas Nilai		
	Sangat Baik	Baik/Bisa dikembangkan	Buruk
LCP	Dibawah 2.5 Detik	2.5 – 4 Detik	Diatas 4 Detik
INP	Dibawah 200ms	200 – 500ms	Diatas 500ms
CLS	Dibawah 0,1	0,1 – 0,25	Diatas 0,25

4.3.1 Pengujian LCP *(Largest Contentful Paint)*

Pada uji coba ini, kita akan mengukur seberapa cepat LCP atau mengukur waktu yang dibutuhkan untuk menampilkan elemen utama terbesar di layar. Pada uji coba ini ditargetkan untuk mendapatkan nilai atau waktu dibawah 2,5 detik.

Tabel 4.12 *Tabel hasil pengujian LCP*

Parameter	Kasus	Tanpa Perlambatan	Throttling Network	Throttling Network + CPU
LCP	Pembuatan Supplier	1,91	2,33	2,66
	Akses Data Supplier	1,09	1,65	1,81
	Akses Data Detail Supplier	2,24	2,58	2,74
	Pembuatan Stok	1,37	1,68	1,87
	Akses Data Stok	2,1	2,47	2,61

	Akses Riwayat Transaksi	1,91	2,46	2,62
	Pembuatan Transaksi	2,2	2,34	2,57
	Akses Final Stok	1,2	1,62	1,88

Dari hasil yang didapatkan, didapatkan hasil yang memuaskan pada uji coba tanpa dan dengan perlambatan jaringan. Tetapi ketika CPU dari perangkat uji diperlambat, didapakan hasil yang berada diatas 2,5 detik, tetapi masih masih berada di rentang 2 detik. Yang mana hal ini masih berada di rentang nilai yang dapat diterima dan masih dapat disebut cepat mengingat hasil yang berada diatas 2,5 detik merupakan hasil dari perlambatan jaringan dan perangkat uji coba.

4.3.2 Pengujian INP *(Interaction to Next Paint)*

Pada uji coba ini, kita akan mengukur seberapa cepat INP atau mengukur waktu respons sistem terhadap interaksi pengguna. Pada uji coba ini ditargetkan untuk mendapatkan nilai atau waktu dibawah 200 ms untuk tiap kasus percobaan.

Tabel 4.13 *Tabel hasil pengujian INP*

Parameter	Kasus	Tanpa Perlambatan	Throttling Network	Throttling Network + CPU
INP	Pembuatan Supplier	43ms	44ms	80ms
	Filter Data Supplier	24ms	36ms	40ms
	Akses Detail Supplier	-	-	-
	Pembuatan Stok	32ms	48ms	52ms
	Filter Data Stok	35ms	36ms	32ms
	Akses Riwayat Transaksi	36ms	32ms	56ms
	Pembuatan Transaksi	36ms	40ms	56ms
	Filter Data Final Stok	24ms	40ms	40ms

Dapat dilihat, dari seluruh hasil percobaan yang dilakukan, didapatkan hasil yang sangat memuaskan, yang mana tiap nilai INP dari sistem yang dikembangkan berada jauh dibawah 200ms.

4.3.3 Pengujian CLS *(Cumulative Layout Shift)*

Dan terakhir merupakan uji coba CLS yang mengukur stabilitas tata letak halaman saat dimuat. Pada uji coba ini juga ditargetkan untuk mendapatkan nilai atau waktu dibawah 0.1 untuk tiap kasus.

Tabel 4.14 *Tabel hasil pengujian CLS*

Parameter	Kasus	Tanpa Perlambatan	Throttling Network	Throttling Network + CPU
CLS	Pembuatan Supplier	0,01	0,01	0,02
	Akses Data Supplier	0,02	0,02	0,02
	Akses Data Detail Supplier	0,00	0,00	0,00
	Pembuatan Stok	0,00	0,00	0,00
	Akses dan Filter Data Stok	0,01	0,01	0,01
	Akses Riwayat Transaksi	0,01	0,01	0,01
	Pembuatan Transaksi	0,01	0,01	0,01
	Akses Final Stok	0,00	0,00	0,00

Dari hasil yang telah dilakukan pada uji CLS, juga didapatkan hasil yang memuaskan yakni seluruh nilai CLS yang didapatkan berada dibawah 0,1. Sehingga dapat disimpulkan bahwa CLS sistem sudah optimal.

Dari seluruh uji performa yang dilakukan, dapat ditemukan bahwa hanya pada uji CLS ketika perangkat uji yang digunakan diberikan perlambatan dalam jaringan dan CPU, sistem mendapat hasil diatas 2,5 detik. Hal ini dapat dikarenakan oleh berbagai alasan seperti jaringan yang tidak stabil, perangkat yang digunakan, atau server sistem itu sendiri karena server sistem uji coba kali ini tidak dilakukan pada home server utama yang memiliki spesifikasi lebih baik daripada cloud hosting yang digunakan saat ini.

4.4 Uji Keamanan Sistem

Uji keamanan sistem merupakan komponen kritis dalam evaluasi sistem berbasis web untuk Toko XYZ. Pengujian ini dirancang untuk mengidentifikasi dan mengevaluasi potensi kerentanan yang dapat dieksploitasi oleh pihak tidak bertanggung jawab. Dalam konteks sistem yang mengelola data sensitif seperti informasi supplier, harga produk, dan transaksi penjualan, keamanan menjadi aspek fundamental yang memerlukan perhatian khusus.

Pengujian keamanan yang dilakukan meliputi dua pendekatan utama: pengujian SQL Injection untuk mengevaluasi keamanan lapisan basis data, dan pengujian ketahanan sistem terhadap serangan DDoS untuk menilai stabilitas sistem di bawah kondisi beban tinggi. Kedua metode ini dipilih karena relevansinya dengan ancaman keamanan yang umum dihadapi aplikasi web modern, khususnya sistem yang memproses data bisnis kritis.

Pengujian dilakukan dalam lingkungan terkontrol yang mereplikasi sistem produksi, dengan fokus pada identifikasi kerentanan dan penilaian efektivitas mekanisme keamanan yang telah diimplementasikan.

4.4.1 Uji Coba SQL Injection

SQL Injection merupakan teknik serangan yang memanfaatkan kerentanan pada aplikasi web untuk menyisipkan kode SQL berbahaya ke dalam kueri basis data. Serangan ini berpotensi mengakibatkan akses tidak sah ke data sensitif, modifikasi data, atau bahkan pengambilalihan kontrol sistem.

Dalam pengujian SQL Injection yang dilakukan pada sistem Toko XYZ, fokus utama diarahkan pada endpoint autentikasi (login) sebagai titik masuk kritis ke dalam sistem. Pengujian dilakukan dengan memasukkan berbagai pola input berbahaya yang dirancang untuk mengganggu atau memanipulasi kueri SQL yang dijalankan oleh backend. Pengujian menggunakan pendekatan *black-box*, di mana berbagai payload SQL Injection dimasukkan ke dalam formulir login melalui antarmuka web. Beberapa payload yang diuji meliputi:

1. Input Klasik untuk *bypass* autentikasi	a. `' OR '1'='1` b. `" OR "1"="1` c. `or 1=1--` d. `' OR '1'='1' --`
2. Input untuk mendeteksi kerentanan blind SQL injection:	a. `' AND (SELECT 1) AND '1'='1` b. `' AND (SELECT 1 FROM dual WHERE 1=2) AND '1'='1`
3. Input dengan karakter khusus untuk menguji sanitasi input:	a. `'; DROP TABLE users; --` b. `' UNION SELECT username,password FROM users --`

Berdasarkan rangkaian pengujian SQL Injection yang dilakukan, dapat disimpulkan bahwa sistem Toko XYZ menunjukkan ketahanan yang baik terhadap serangan SQL Injection. Beberapa hasil kunci yang diamati:

1. Penolakan Input Berbahaya, Sistem secara konsisten menolak semua payload SQL Injection yang diuji, dengan mengembalikan pesan kesalahan generik tanpa mengungkapkan informasi teknis tentang struktur basis data.
2. Tidak Ada Kebocoran Informasi, Tidak terdeteksi adanya kebocoran informasi teknis dalam pesan kesalahan yang dapat dimanfaatkan untuk serangan lebih lanjut.

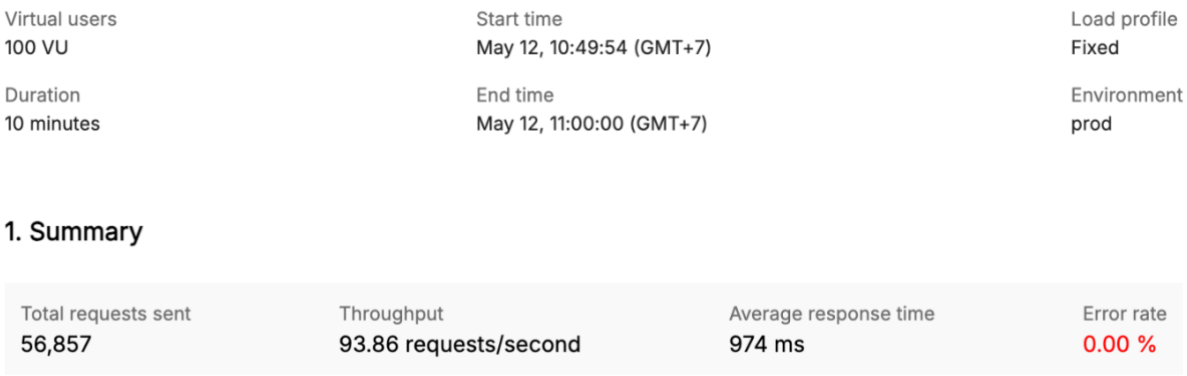
4.4.2 Uji Coba DDOS dan *Load Testing*

Pengujian ketahanan terhadap serangan Distributed Denial of Service (DDoS) merupakan komponen penting dalam evaluasi keamanan sistem Toko XYZ. Pengujian ini bertujuan untuk menilai kemampuan sistem dalam menangani lonjakan traffic yang signifikan

dan memastikan ketersediaan layanan tetap terjaga meskipun dalam kondisi beban tinggi. Pendekatan pengujian mengkombinasikan load testing konvensional dan simulasi pola serangan DDoS sederhana untuk memberikan gambaran komprehensif tentang ketahanan sistem.

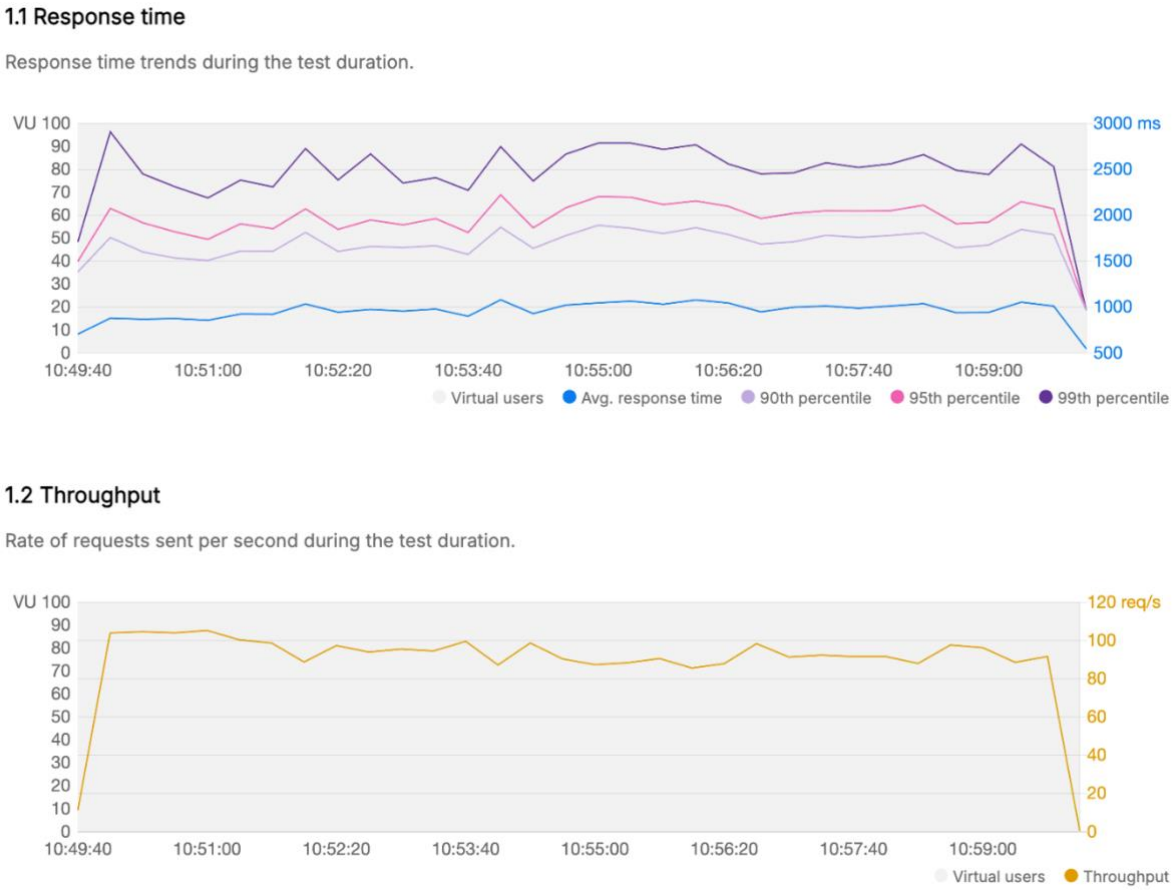
Pengujian memanfaatkan Postman untuk melakukan serangan yang mensimulasikan 100 pengguna yang secara bersamaan yang menyerang/mengakses satu atau beberapa *endpoint* API yang ada dengan semaksimal mungkin hingga batas maksimal server. Berikut merupakan hasil yang didapatkan dari melakukan pengujian menggunakan postman dengan beberapa kondisi yaitu:

1. Dilakukan pengujian terhadap beberapa endpoint halaman dengan menggunakan 100 *virtual user*



***Gambar 4.2** Hasil pengujian terhadap banyak endpoint selama 10 menit*

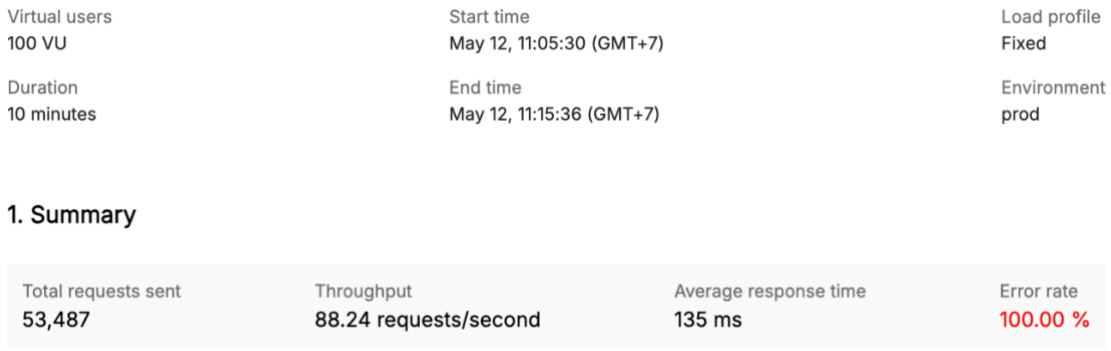
Dari hasil yang didapatkan diatas, dapat kita lihat total permintaan yang dikirim mencapai 24.000 permintaan dengan rata-rata request 93,86 request per detik dan rata-rata waktu yang diperlukan untuk menyelesaikan permintaan di angka 974ms atau 0,974 detik. Error rate yang didapat juga 0% yang menandakan setiap permintaan yang dilakukan berhasil dikirim dan diterima atau dikembalikan secara sukses.



Gambar 4.3 *Grafik response time dan throughput dari pengujian endpoint*

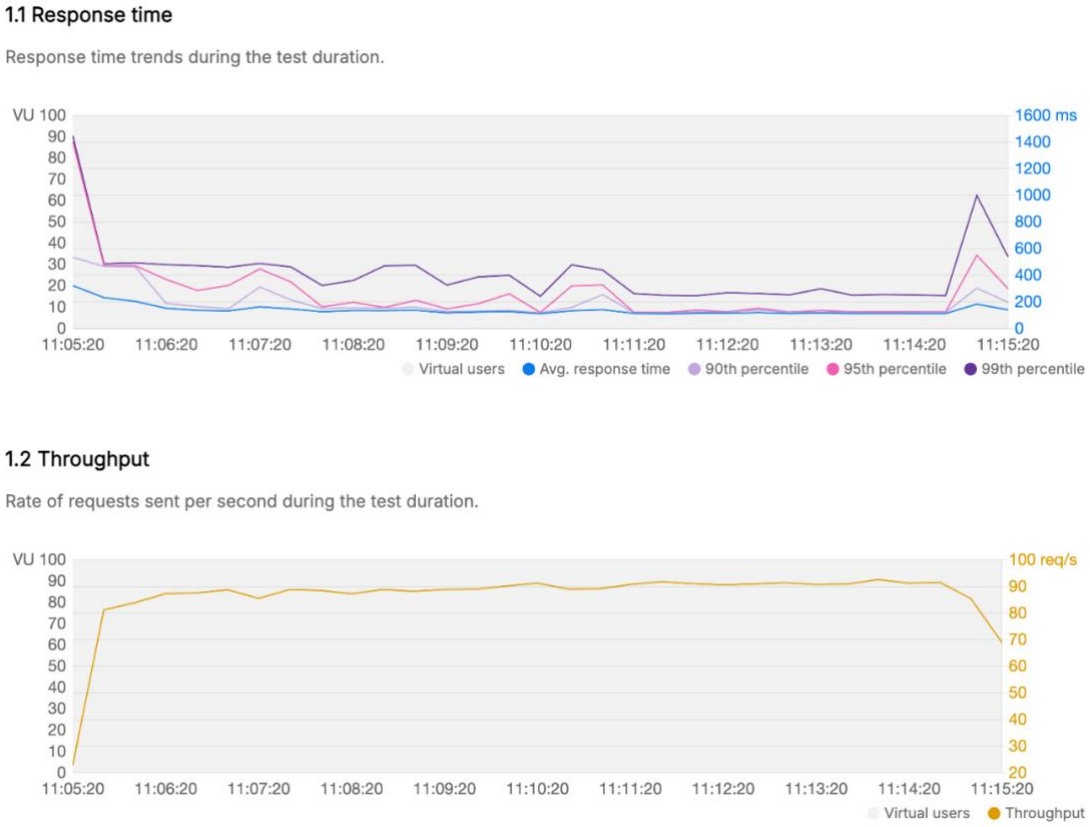
Hasil pengujian beban pada sistem Toko XYZ menunjukkan performa yang sangat memuaskan dengan ketahanan yang baik terhadap beban tinggi. *Grafik 4.3* juga menampilkan bahwa sistem mampu mempertahankan throughput konsisten sekitar 90-100 request per detik selama periode pengujian 10 menit dengan 100 pengguna virtual secara simultan, menunjukkan kapasitas pemrosesan yang melebihi kebutuhan operasional toko. Waktu respons rata-rata sistem bertahan stabil di kisaran 900-1000 ms (< 1 detik), sementara persentil ke-99 (skenario terburuk) tetap di bawah 2500 ms, bahkan pada puncak beban.

2. Dilakukan pengujian terhadap endpoint login saja dengan menggunakan 100 *virtual user*



Gambar 4.4 *Hasil pengujian terhadap endpoint login selama 10 menit*

Dari hasil pengujian beban endpoint login, dapat kita lihat bahwa sistem tetap menerima dan melayani 53.487 permintaan dan memberikan rata-rata waktu 135 ms. 100% error rate yang didapat dikarenakan konfigurasi data login yang digunakan disengaja menggunakan data yang salah. Hal ini untuk menguji apakah dalam beban yang diberikan sistem tetap dapat melayani permintaan login dari user yang memang memiliki data yang benar.



Gambar 4.5 *Grafik Response dan Throughput terhadap pengujian endpoint login selama 10 menit*

Hasil pengujian performa sistem Toko XYZ menunjukkan optimasi yang sangat baik dalam menangani beban tinggi. Grafik menampilkan bahwa sistem mengalami penurunan waktu respons yang signifikan di awal pengujian, dari sekitar 350ms menjadi stabil di kisaran 150-200ms setelah fase warm-up, menunjukkan optimasi caching dan connection pooling yang efektif. Meskipun diuji dengan beban yang konsisten sekitar 85-90 request per detik selama periode 10 menit, performa sistem sangat stabil dengan waktu respons rata-rata yang rendah, bahkan di persentil ke-99 tetap bertahan di bawah 400ms untuk sebagian besar durasi pengujian. Terdapat lonjakan kecil pada menit terakhir pengujian yang mencapai sekitar 900ms pada persentil ke-99, namun rata-rata tetap stabil, menunjukkan ketahanan sistem terhadap variasi beban

Setiap pengujian dilakukan, penulis mencoba mengakses dan menggunakan website sebagaimana mestinya digunakan. Berdasarkan hasil yang didapatkan, meskipun sistem telah diberikan beban oleh banyak pengguna dan dengan beban yang cukup berat. Sistem masih dapat melayani permintaan dari user dengan tidak adanya perbedaan performa yang signifikan dari hasil yang diuji sebelumnya.

4.5 Analisis Implementasi Arsitektur terhadap Performa Sistem

Berdasarkan hasil pengujian performa yang telah dilakukan, dapat disimpulkan bahwa arsitektur sistem Toko XYZ menunjukkan performa yang sangat baik. Parameter utama pengujian seperti LCP, INP, dan CLS menunjukkan nilai yang memenuhi standar, bahkan pada kondisi jaringan dan perangkat yang diberi perlambatan. Hal ini mengindikasikan bahwa pemilihan dan implementasi teknologi pada sisi *frontend* dan *backend* telah berhasil mendukung kebutuhan bisnis dengan optimal. Arsitektur sistem yang terdiri dari Next.js pada *frontend*, Go sebagai *backend*, PostgreSQL sebagai basis data, dan NGROK sebagai tunnel API telah berhasil terintegrasi dengan baik, memberikan respons cepat dan pengalaman pengguna yang mulus.

4.5.1 *Frontend*

Implementasi frontend menggunakan Next.js terbukti memberikan kontribusi signifikan terhadap performa sistem. Hasil pengujian LCP (*Largest Contentful Paint*) menunjukkan nilai rata-rata 1,75 detik tanpa perlambatan dan 2,19 detik dengan perlambatan jaringan, keduanya berada di bawah ambang batas optimal 2,5 detik.Nilai CLS (*Cumulative Layout Shift*) yang konsisten di bawah 0,02 menunjukkan stabilitas tata letak halaman yang sangat baik. Hasil memuaskan tersebut didapat dari beberapa faktor arsitektur *frontend* yang berkontribusi besar terhadap performa ini yang meliputi:

1. Server-Side Rendering (SSR): Implementasi SSR pada Next.js memungkinkan konten halaman diproses di server sebelum dikirimkan ke klien, sehingga mempercepat proses rendering awal. Fitur ini sangat menguntungkan pada halaman-halaman data-heavy seperti halaman stok dan final stok yang menampilkan jumlah data signifikan.

2. Static Site Generation (SSG) untuk halaman-halaman statis seperti halaman login dan formulir input, memungkinkan halaman di-pre-render pada saat build time sehingga pengiriman ke klien menjadi sangat cepat.

3. Code Splitting dan Dynamic Import, Implementasi ini memungkinkan pemecahan bundle JavaScript menjadi chunk-chunk yang lebih kecil dan dimuat sesuai kebutuhan. Hal ini terbukti pada nilai INP yang selalu di bawah 80ms, menunjukkan responsivitas tinggi terhadap interaksi pengguna.

4. Lazy Loading Components, Komponen-komponen yang tidak segera terlihat pada *viewport*, seperti tabel data pada halaman dengan pagination, dimuat secara lazy, mengurangi initial load time.

5. Minifikasi dan Kompresi Aset: Implementasi minifikasi dan kompresi JavaScript, CSS, dan HTML pada build production berkontribusi pada pengurangan ukuran payload dan peningkatan kecepatan unduh.

Optimasi-optimasi ini secara kolektif berkontribusi pada ketahanan sistem terhadap pengujian keamanan. Misalnya, implementasi validasi input pada sisi client melalui *React Hook Form* memberikan lapisan pertahanan pertama terhadap serangan *SQL Injection*, sementara arsitektur berbasis token JWT dengan *cookies* meningkatkan keamanan autentikasi. Nilai CLS yang konsisten di bawah 0,02 juga menunjukkan stabilitas tata letak halaman yang sangat baik, mencegah potensi serangan *clickjacking*.

4.5.2 *Backend*

Arsitektur backend yang terdiri dari Go, PostgreSQL, dan NGROK menunjukkan performa yang sangat baik dalam menangani permintaan dari frontend. Bahasa Go terbukti menjadi pilihan yang tepat berdasarkan hasil pengujian performa bahkan didalam perlambatan jaringan sistem tetap menunjukkan performa yang memuaskan. Beberapa faktor kunci arsitektur backend yang berkontribusi terhadap performa dan keamanan sistem yang optimal:

1. Basis data Connection Pooling, Implementasi connection pooling pada koneksi basis data GORM memastikan penggunaan koneksi basis data yang efisien, mengurangi overhead pembuatan koneksi baru, dan meningkatkan throughput sistem, sebagaimana terlihat pada pengujian DDoS yang menunjukkan kemampuan menangani 93,86 request per detik.
2. *B-tree Indexing* pada PostgreSQL**,** Optimasi indexing menggunakan metode B-tree pada kolom primary key dan foreign key meningkatkan kecepatan query, terutama pada operasi join kompleks seperti yang terlihat pada implementasi pengambilan data riwayat transaksi, berkontribusi pada response time rata-rata 1164ms meskipun di bawah beban maksimal yang dapat ditangani oleh server.
3. *Parameterized Queries* dengan GORM, Implementasi konsisten *parameterized queries* dan *prepared statements* melalui GORM memberikan perlindungan efektif terhadap serangan SQL Injection, terbukti dari hasil pengujian SQL Injection yang tidak menunjukkan adanya kerentanan.
4. *Middleware Chain Architecture,* Penggunaan arsitektur *middleware chain* pada Gin *framework* memungkinkan *middleware* seperti autentikasi dan logging diaplikasikan secara terpisah, meningkatkan modularitas dan *maintainability* kode.
5. JWT (*JSON Web Token*) *Authentication***,** Mekanisme autentikasi berbasis token dengan validasi signature menggunakan algoritma HMAC-SHA256 memberikan lapisan keamanan yang kuat dan *stateless*, mengurangi beban server dalam mengelola sesi.
6. Error Handling Terstruktur, Implementasi error handling yang terstruktur dan konsisten, dengan lapisan abstraksi error yang memisahkan domain error dari HTTP response, mencegah kebocoran informasi teknis yang dapat dimanfaatkan penyerang.
7. Stateless Design, Desain backend yang stateless memungkinkan scaling horizontal yang lebih mudah dan mengurangi beban memori server, meningkatkan ketahanan terhadap lonjakan traffic.

Arsitektur yang diimplementasikan dalam sistem ini telah terbukti efektif dalam pengujian *SQL Injection* dan *DDoS*, di mana sistem mampu menolak semua *payload* berbahaya tanpa mengekspos informasi internal, serta tetap responsif bahkan di bawah beban tinggi dengan 93,86 request per detik. Kemampuan ini menunjukkan bahwa pemilihan Go sebagai bahasa backend, dikombinasikan dengan penerapan praktik keamanan yang solid, berhasil menciptakan sistem yang tidak hanya berkinerja tinggi tetapi juga aman dari ancaman umum.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari seluruh proses yang telah dilakukan meliputi proses perancangan, implementasi, dan pengujian aplikasi untuk Toko Baju XYZ, telah dilakukan evaluasi komprehensif terhadap hasil sistem. Pengujian meliputi *functional testing* untuk memverifikasi fitur-fitur sistem, *usability testing* dengan metode SUS untuk mengukur tingkat kegunaan, *performance testing* untuk mengevaluasi kecepatan dan kestabilan, serta *security testing* untuk menilai ketahanan sistem terhadap serangan. Berdasarkan rangkaian pengujian tersebut dan observasi implementasi pada lingkungan yang disesuaikan semirip mungkin dengan di toko, didapatkan hasil pengamatan yang dapat disimpulkan sebagai berikut.

1. Sistem berbasis web yang dikembangkan khusus untuk UD. XYZ telah dikembangkan dengan baik dengan spesifikasi dan kustomisasi sesuai dengan yang diinginkan oleh klien. Tahap pengembangan sistem telah dinyatakan selesai dan saat ini sedang dalam tahap testing/pengujian oleh klien langsung pada lokasi. Pihak pengembang atau penulis memberikan garansi 4 bulan setelah masa pengujian ini, sehingga saat ini penulis masih terikat kontrak untuk memperbaiki sistem apabila ditemukan adanya *bugs.*
2. Sistem juga terbukti membantu dalam proses bisnis toko, dimana setiap proses yang sebelumnya dilakukan pencatatan dan perhitungan yang dilakukan secara manual telah diintegrasikan dan diotomatisasi dalam masing-masing halaman yang telah diimplementasikan dalam sistem.
3. Tingkat kegunaan sistem telah diukur menggunakan *usability test* dengan metode SUS. Skor SUS yang didapatkan adalah 72,8 yang tergolong dalam tingkat kegunaan yang baik. Selain pengujian terhadap responden acak, dilakukan juga pengujian terhadap karyawan toko, yang mana mendapat hasil 81,8 yang mana sangatlah baik.
4. Penerapan arsitektur Next.js dengan optimisasi SSR/SSG *hybrid***,** Go untuk *concurrent backend processing*, dan PostgreSQL dengan indexing serta *query optimization* yang tepat, berhasil menghasilkan sistem penjualan berkinerja tinggi untuk Toko XYZ. Kombinasi teknologi ini memberikan performa optimal**,** skalabilitas yang baik, dan pengalaman pengguna yang responsif, mampu menangani *concurrent transactions*, *real-time inventory updates*, dan *fast page loading* sesuai kebutuhan operasional bisnis retail modern.
5. Pengujian dan evaluasi performa dan keamanan membuktikan bahwa sistem memiliki performa yang sangat memuaskan dan juga keamanan sistem terbukti dapat menghadapi beberapa pengujian keamanan yang diberikan. Berupa *SQL Injection* dan *Load testing* yang juga membuktikan performa sistem dalam menangani permintaan pengguna.

5.2 Saran

Sebagai pihak pengembang yang telah melakukan masa pengembangan dan evaluasi dari sistem ini. Didapatkan hasil yang telah dirangkum pada poin sebelumnya, kemudian berdasarkan hasil-hasil pengujian dan pengembangan yang telah dilakukan dapat diberikan beberapa saran untuk pihak klien untuk dipertimbangkan dalam pengembangan lebih lanjut yang akan datang apabila direncanakan.

1. Integrasi dengan Sistem Lain, untuk pengembangan sistem di masa depan sangat direkomendasikan untuk mengimplementasikan integrasi dengan sistem-sistem lain yang mendukung dan memperluas fungsionalitas sistem. Contohnya seperti integrasi ke platform pembayaran digitas seperti QRIS, OVO, Gopay, dkk. Selain itu, dapat dilakukan juga integrasi ke marketplace online seperti Tokopedia atau Shopee, mengingat tren belanja online yang semakin meningkat.

2. *Data Analytics*, Implementasi modul *Data Analytics* merupakan tingkat pengembangan berikutnya yang dapat meningkatkan nilai sistem bagi pebisnis. Dengan *adanya Data Analytics*, sistem dapat menganalisa penjualan untuk mendapatkan tren atau *insight* seperti perilaku pelanggan, performa produk, pola penualan yang dapat menjadi wawasan untuk melakukan keputusan bisnis. Data yang telah diperoleh dan diolah kemudian baiknya divisualisasikan untuk pemilik bisnis dapat berinteraksi dan membacanya lebih mudah.
3. Skalabilitas, Untuk mengantisipasi pertumbuhan bisnis Toko XYZ, sistem perlu dirancang dengan mempertimbangkan skalabilitas. Pendekatan horizontal dan vertikal dapat digunakan untuk menangani peningkatan beban. Backend Go mendukung konkurensi melalui goroutines, namun perlu diperkuat dengan connection pooling dan caching menggunakan Redis. Pemisahan modul menjadi microservices serta penggunaan load balancer seperti Nginx juga disarankan untuk efisiensi dan fleksibilitas saat bisnis berkembang.

DAFTAR PUSTAKA

Chaffey, Dave. (2015). *Digital business and E-commerce management : strategy, implementation and practice*. Pearson Education Limited.

Fariz, M., Lazuardy, S., & Anggraini, D. (2022). Modern Front End Web Architectures with React.Js and Next.Js. *International Research Journal of Advanced Engineering and Science*, *7*(1), 132–141.

Harrison Oke Ekpobimi. (2024a). Building high-performance web applications with NextJS. *Computer Science & IT Research Journal*, *5*(8), 1963–1977. https://doi.org/10.51594/csitrj.v5i8.1459

Harrison Oke Ekpobimi. (2024b). Building high-performance web applications with NextJS. *Computer Science & IT Research Journal*, *5*(8), 1963–1977. https://doi.org/10.51594/csitrj.v5i8.1459

Julia Kagan. (2024, April 17). *Understanding Tax Deductions: Itemized vs. Standard Deduction*. Https://Www.Investopedia.Com/Terms/t/Tax-Deduction.Asp.

Monica Pakpahan, W., Febrian, A., Retning Jati, P., Winardi, S., & Adiputra Pardosi, I. (2022). ANALISIS DAN PERANCANGAN SISTEM APLIKASI KASIR DIGITAL (POINT OF SALES). *Jl*, *23*(1).

Pandya, D. N., Suryadharma, D., Wulandhari, L. A., & Alam, I. N. (2024). Assessing University Website Performance: A Comparative Analysis Using GTmetrix. *International Journal of Computer Science and Humanitarian AI*, *1*(1), xx–xx. https://doi.org/10.21512/ijcshai.v1i1.12152

Purnomo, S., Nurmalitasari, N., & Nurchim, N. (2024). Digital transformation of MSMEs in Indonesia: A systematic literature review. *Journal of Management and Digital Business*, *4*(2), 301–312. https://doi.org/10.53088/jmdb.v4i2.1121

Rosmala, D., & Rasyidin, I. (2024). Data Storage Basis data PostgreSQL and JSON File Format in Golang Using Gin-Gonic and Gin-Gonic with GORM. *Industrial Sciencetech Jurnal*, *1*(1). https://doi.org/10.26760/jrh

Soegoto, E. S., & Palalungan, A. F. (2020). Web Based Online Inventory Information System. *IOP Conference Series: Materials Science and Engineering*, *879*(1). https://doi.org/10.1088/1757-899X/879/1/012125

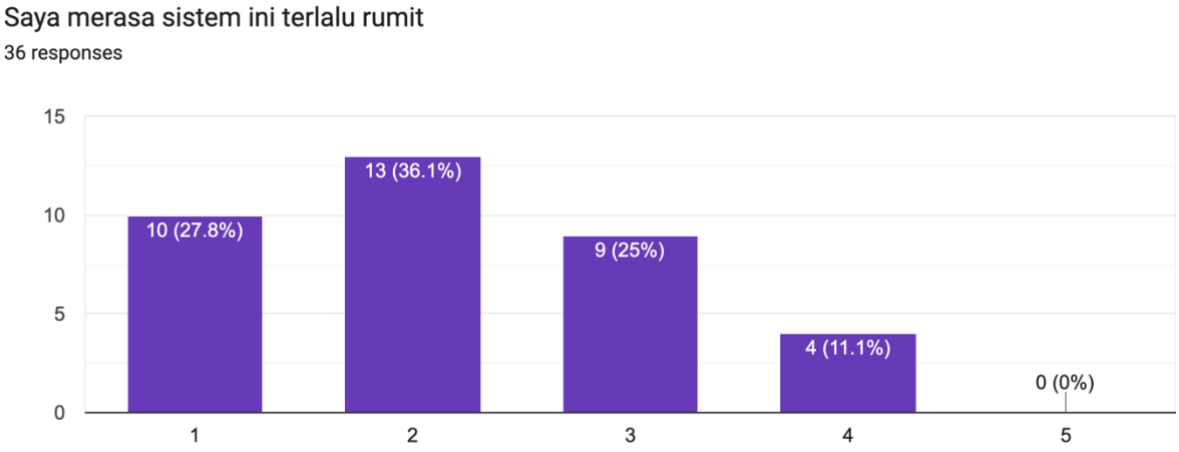
Susanto, F. G. P., Fadlan, N. I. Y., & Haryani, P. (2023). Design of Web-Based Management Information System for Student Organizations in Kendal Regency Using Next.js Framework. *Compiler*, *12*(1), 9. https://doi.org/10.28989/compiler.v12i1.1616

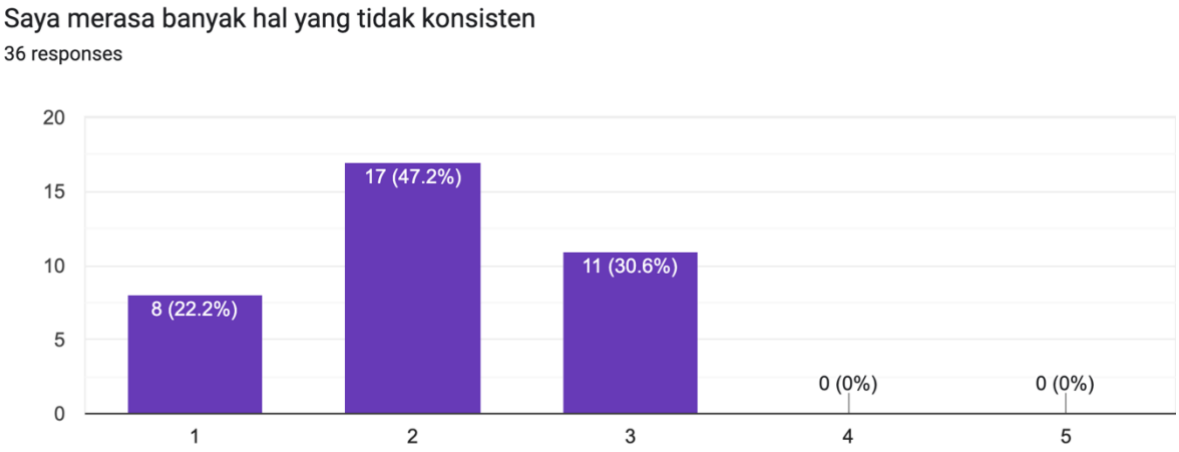
Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2024). *Overview of Web Application Performance Optimization Techniques*. http://arxiv.org/abs/2412.07892

(Halaman ini sengaja dikosongkan)

LAMPIRAN-LAMPIRAN ATAU APPENDIK

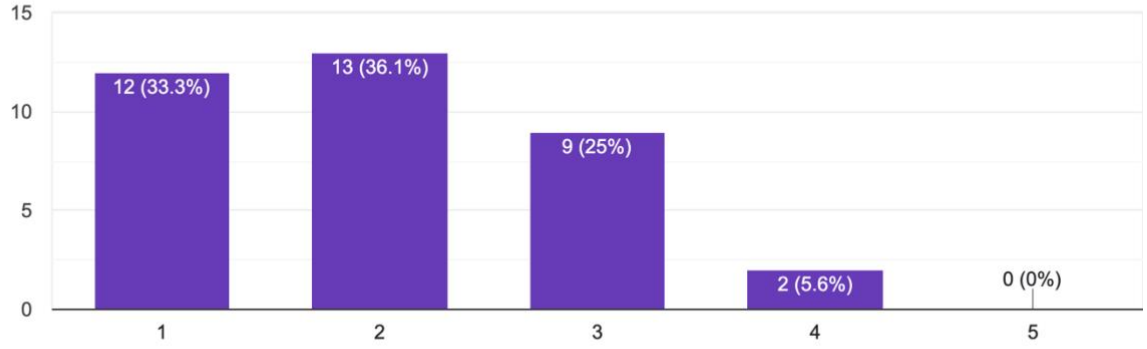
Lampiran 1
Jawaban Kuisioner SUS





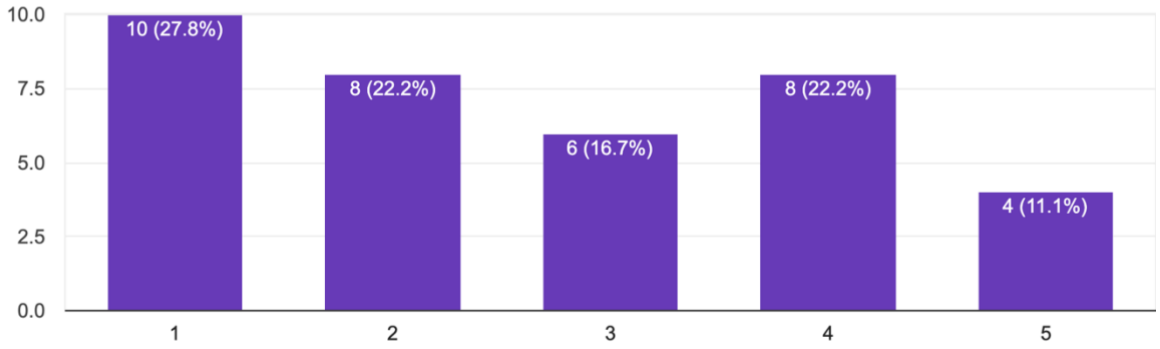
Saya merasa performa sistem ini lambat

36 responses



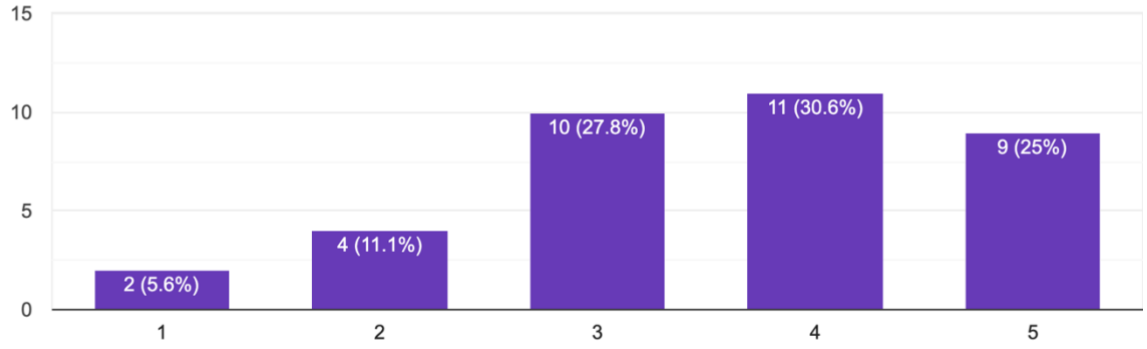
Saya membutuhkan bantuan dalam menggunakan sistem ini

36 responses



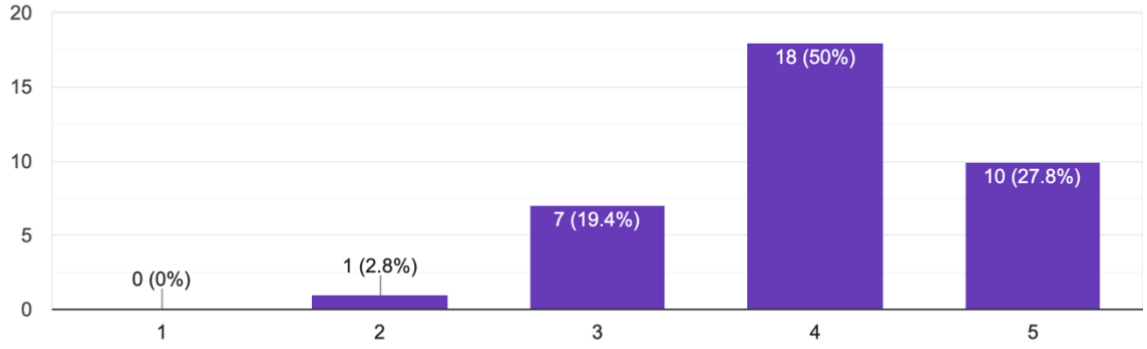
Saya tidak mengalami bug dan error saat mengoperasikan sistem

36 responses



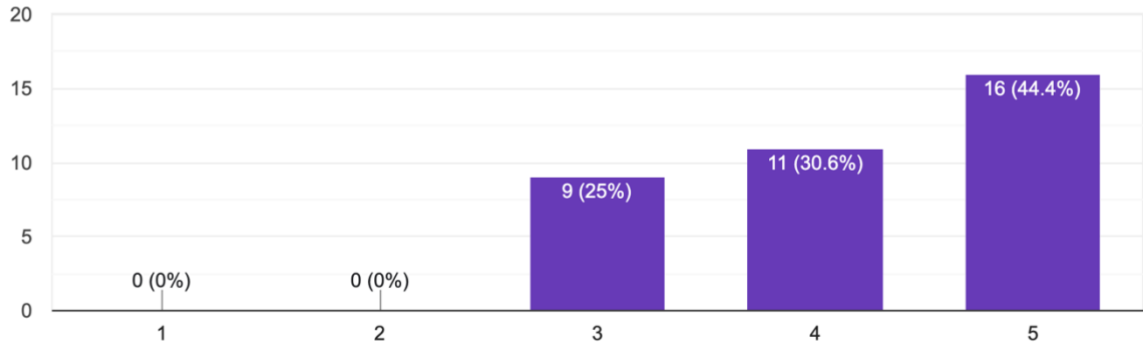
Saya rasa sistem sudah terintegrasi dengan baik

36 responses



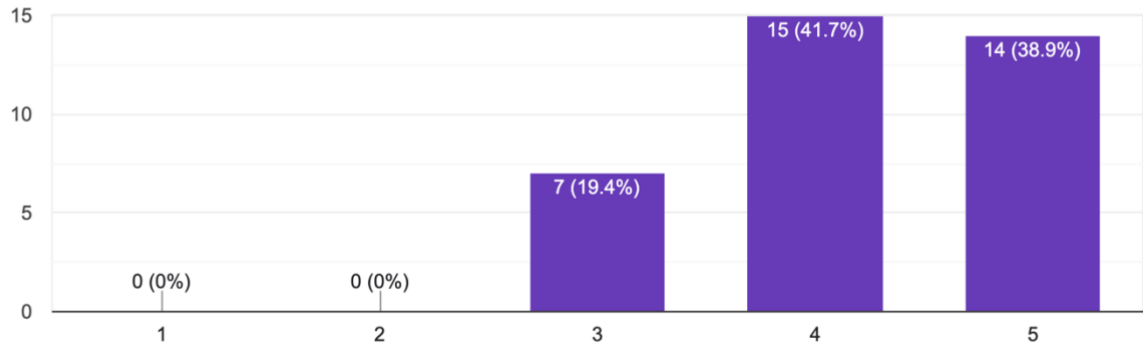
Saya rasa sistem sudah responsif saat digunakan

36 responses



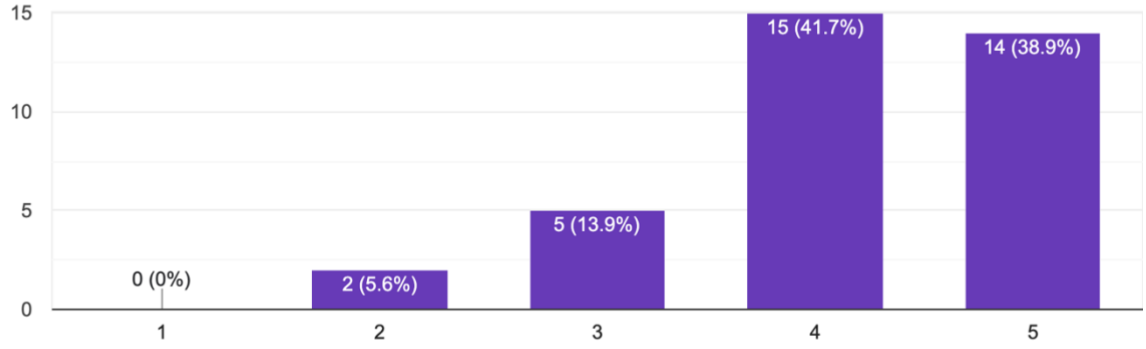
Navigasi pada sistem sudah jelas dan mudah digunakan

36 responses



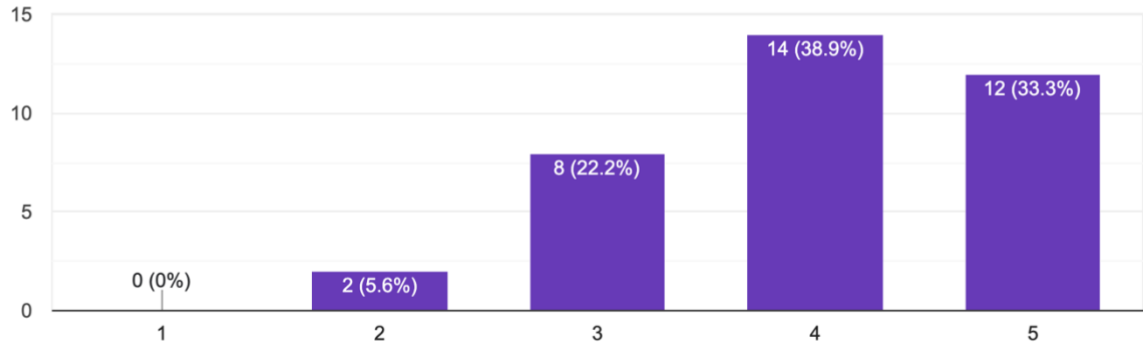
Saya rasa proses dan alur dari tiap modul sistem sudah efisien

36 responses



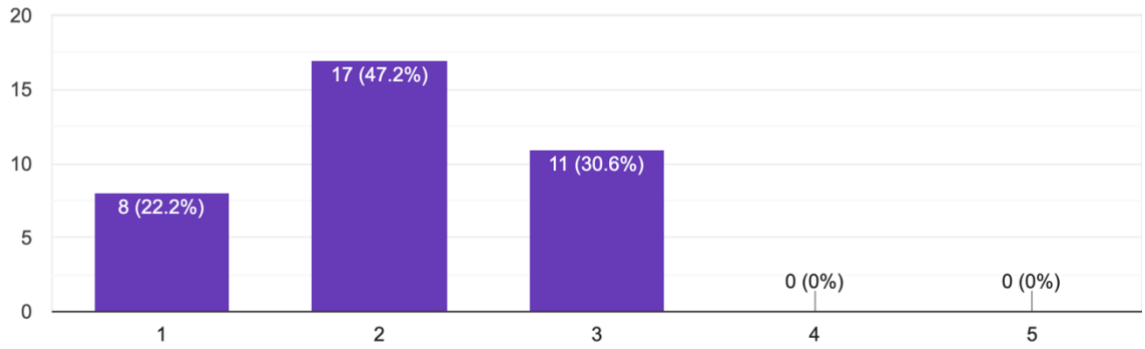
Saya merasa puas dengan pengalaman saya menggunakan sistem ini

36 responses



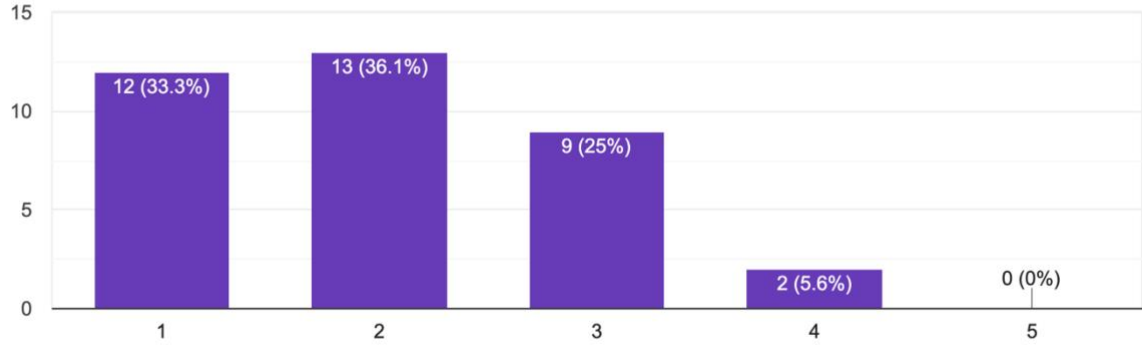
Saya merasa banyak hal yang tidak konsisten

36 responses



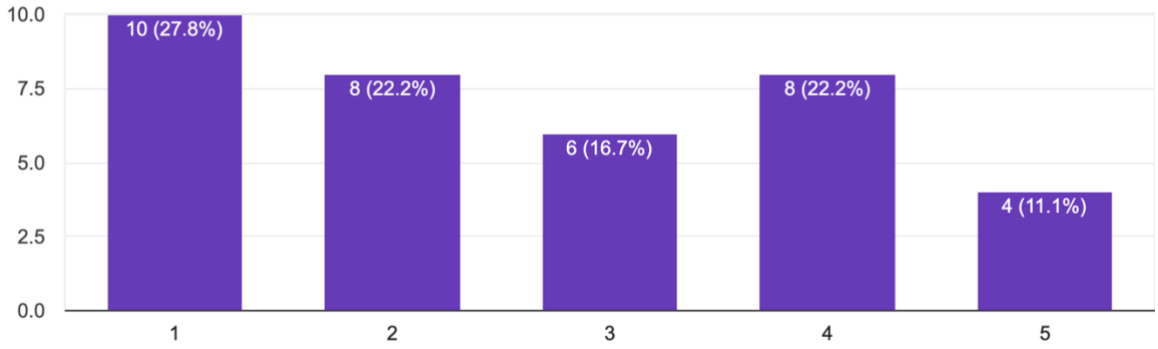
Saya merasa performa sistem ini lambat

36 responses



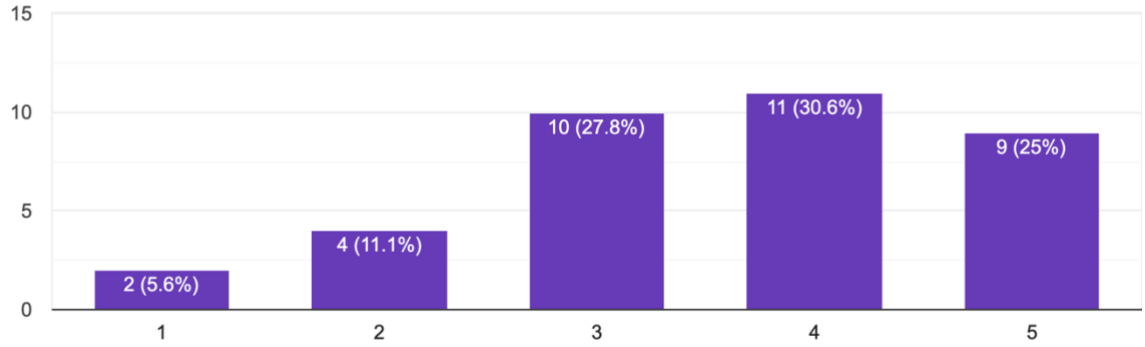
Saya membutuhkan bantuan dalam menggunakan sistem ini

36 responses



Saya tidak mengalami bug dan error saat mengoperasikan sistem

36 responses



MOU

Merupakan kontrak Kerjasama dari kedua belah pihak untuk pengembangan dan pembuatan sistem, berikut adalah cuplikan beberapa isi dari MOU tersebut. Yang mana dokumen lengkap dapat diakses pada https://docs.google.com/document/d/1U6hbeC0pgTFYv3Y7iO8eU9yikpkXWZiDZ3nazy_okDM/edit?usp=sharing

SURAT PERJANJIAN KERJA SAMA
TOKO BUMI SUBUR

Perjanjian ini dibuat di Surabaya, pada **19 Mei 2024** oleh dan antara :

1. Nama : Adrian Karuna Soetikno
   Jabatan : *Project Manager* dan *Web Developer*
   Alamat : Sutorejo Selatan VI no. 11A, Surabaya

2. Nama : Rizky Alifiyah Rahma
   Jabatan : *Web Developer* dan *UI/UX Designer*
   Alamat : Jl. Medayu Sentosa No. 130, Rungkut, Surabaya, Jawa Timur 60295

Dalam hal ini akan berperan sebagai pihak **Pengembang** Situs Web POS untuk toko "Bumi Subur" yang selanjutnya akan disebut sebagai **PIHAK PERTAMA**.

3. Nama : Rosiendy Eddy Tanair
   Jabatan : CEO Bumi Subur
   Alamat : Jalan Pemuda, Tanjung Redeb, Berau, Kalimantan Timur

Dimana pada surat ini akan berperan sebagai **Klien** dari pengembangan Situs Web untuk toko "Bumi Subur" yang selanjutnya akan disebut sebagai **PIHAK KEDUA**.

Dengan ditandatanganinya surat ini maka kedua belah pihak dinyatakan sepakat untuk mengadakan kerja sama dengan ketentuan-ketentuan sebagai berikut :

PASAL 1
Jangka Waktu

1. **PIHAK PERTAMA** bersedia menerima dan mengerjakan pekerjaan yang diberikan oleh **PIHAK KEDUA** dengan durasi waktu yang dijelaskan pada **Pasal 5 Poin 1.** Dilanjutkan dengan durasi selama 5 Bulan untuk testing dan perbaikan *bug* yang ditemukan pada masa tersebut dari sistem yang telah dibuat oleh **PIHAK PERTAMA**.
2. **PIHAK KEDUA** selaku pemberi pekerjaan bersedia memberikan data dan informasi yang dibutuhkan guna pengembangan Sistem selama periode pengerjaan tersebut. Terutama pada masa analisis kebutuhan sistem, dimana masa ini krusial untuk mengembangkan sistem yang sesuai dengan kebutuhan dan keinginan **PIHAK KEDUA**.

Product Requirement Document

Merupakan dokumen yang berisikan kebutuhan kebutuhan secara singkat guna administrasi dan panduan pihak pengembang untuk lebih memahami keinginan klien

Fitur Sistem :

Fitur-fitur yang diharapkan ada pada sistem ERP dari Bumi Subur ini adalah sebagai berikut :

- Sistem Inventaris Stok barang
    - Penambahan stok dapat menambahkan barang baru, dan membuat barcode baru untuk tipe barang tersebut.
    - Sistem dapat mengakomodasi permintaan klien dengan membuat inventarisasi berdasarkan jenis pakaian yang memiliki banyak ukuran (tiap pakaian memiliki 1 barcode id namun dapat memiliki banyak ukuran.)
    - Penambahan stok oleh karyawan harus melalui persetujuan owner terlebih dahulu (masuk sebagai pending stok) sebelum dapat masuk kedalam tabel produk dari *database* sistem.
    - Sistem dapat menyimpan harga jual dan harga beli, dimana harga beli otomatis dipotong dari harga jual yang dimasukkan berdasarkan merk atau supplier (dapat dipilih untuk menggunakan yang mana)
    - Sistem dapat menentukan status barang yang sudah termasuk stok lama berdasarkan durasi yang ditentukan.
    - Sistem dapat memberikan diskon/merubah harga
    - Sistem dapat melakukan perubahan informasi dari data stok tersebut Sistem dapat mencetak laporan kondisi stok barang yang ada di gudang tersebut

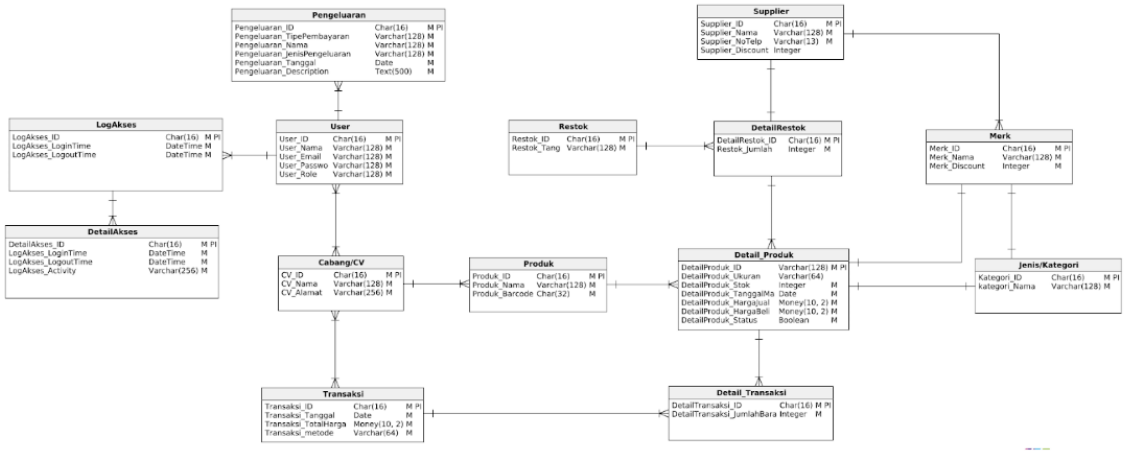
- Sistem dapat melayani penjualan/barang keluar (dapat digunakan dengan fitur barcode scanner yang telah dimiliki oleh toko/input langsung kode unik yang ada di barcode tersebut)
    - Sistem dapat melayani operasi layaknya aplikasi kasir atau **POS**
    - Sistem dapat memisahkan penjualan dari satu toko/CV dengan CV lain tetapi penjualan tetap tercetak dalam satu struk atau nota.
    - Sistem harus menyimpan semua laporan penjualan, dan dapat mencetak laporan penjualan harian atau pada jangka waktu tertentu
    - Sistem menyimpan histori transaksi tiap cv masing masing terpisah.

- Sistem dapat digunakan untuk mencatat pembukuan dari pengeluaran dan pemasukan yang dilakukan.
- Sistem dapat mencatat log akses tiap user mulai dari login hingga aktivitas yang dilakukan
- Sistem menyimpan data dari supplier yang dimiliki dan merk-merk apa saja yand di supply olehnya. Dari data merk dan supplier ini nantinya akan digunakan untuk pemotongan harga jual dari produk.

User Scenarios :

Kategori Pengguna	Tugas	Hak Akses ke aplikasi	Kemampuan yang harus dimiliki
Pemilik	Mengatur dan mengontrol keseluruhan bisnis	• Dapat mengakses semua halaman yang dapat diakses oleh karyawan • Dapat mengakses halaman "histori akses" • Dapat mengakses halaman "pengeluaran" • Dapat mengakses halaman "*pending* stok"	Mampu mengoperasikan *Smartphone* atau *PC* atau laptop
Kasir	Mencatat penjualan sebagai kasir	• Dapat mengakses halaman transaksi • Dapat mengakses halaman riwayat transaksi, yang dibatasi hingga riwayat hari tersebut • Dapat melihat Stok produk CV	Mampu mengoperasikan *Smartphone* atau *PC* atau laptop
Stock Keeper	Mengontrol stok toko dan gudang masing-masing	• Dapat mengakses halaman Stok • Dapat menambahkan stok barang baru ataupun lama. • Dapat menghapus stok barang baru ataupun lama.	Mampu mengoperasikan *Smartphone*/laptop/PC untuk pengelolaan data

Object Description (Database) :



UI Design:

Berikut untuk UI dari sistem Bumi Subur : Link Figma Design

Success Metric :

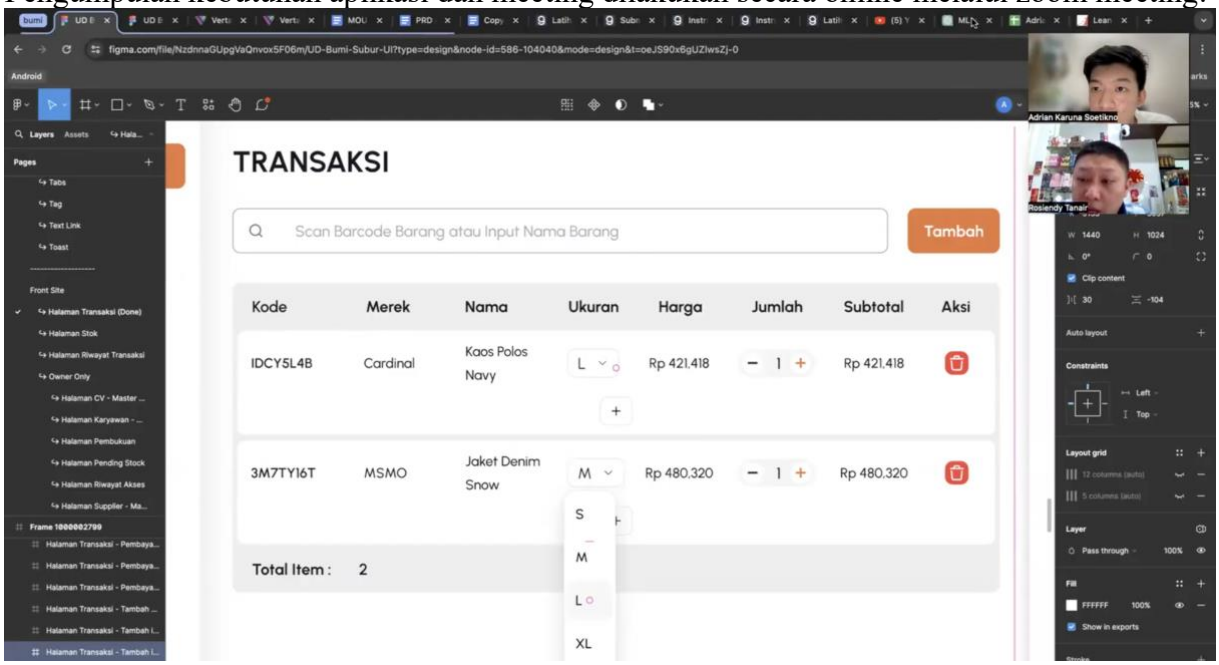
1. **Sebagai Kasir, saya dapat melakukan login ke akun milik saya**
   Acceptance Criteria :
   a. Halaman transaksi dapat dibuka melalui link dan menampilkan:
      i. Kolom form Username
      ii. Kolom form Password
      iii. Tombol login
   b. Halaman akan berpindah sesuai dengan otoritas yang dimiliki pegawai
   c. Sistem menampilkan halaman-halaman yang dapat diakses oleh pegawai tersebut

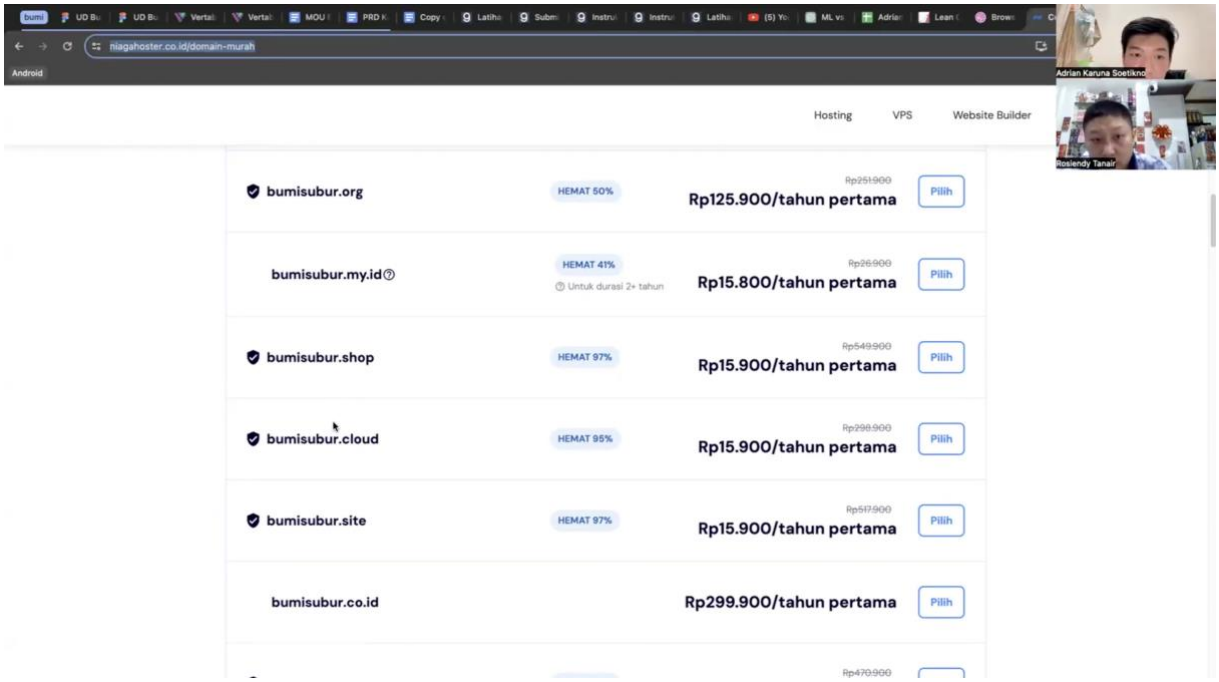
2. **Sebagai Kasir, saya dapat menambahkan Transaksi dengan halaman “transaksi”.**
   Acceptance Criteria :
   a. Halaman transaksi dapat dibuka melalui link/menu pada sidebar dan menampilkan :
      i. Informasi subtotal transaksi
      ii. Kolom penambahan produk dengan BarcodeID atau nama produk.
      iii. Pilihan metode pembayaran
      iv. Kolom perhitungan kembalian

Untuk dokumen lengkap, dapat diakses pada
https://docs.google.com/document/d/1vMHaC2Zl7xdqBF5-z8qCVwoLGJqD0IZRFzFJuGmKJRQ/edit?usp=sharing

Dokumentasi Pengumpulan Kebutuhan Sistem

Pengumpulan kebutuhan aplikasi dan meeting dilakukan secara online melalui zoom meeting.





BIODATA KLIEN

Nama : Bapak X
Alamat : Berau, Kalimantan Timur
Alamat Toko : Berau, Kalimantan Timur

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Penulis dilahirkan di Kota Rembang pada 29 April 2003 dan merupakan anak pertama dari 3 bersaudara. Penulis menempuh Pendidikan masa SMA pada SMAN 1 PATI yang mana setelah lulus ia menempuh Pendidikan tinggi pada Teknik Informatika ITS melalui jalur SNMPTN pada tahun 2021 dengan nomor NRP 5025211019.
Semasa Aktif sebagai mahasiswa ITS, penulis sempat aktif dalam organisasi kerohanian TPKB ITS, Aktif dalam event tahunan Schematics, dan kerap kali mengisi sebagai Asisten Pengajar/Dosen pada beberapa mata kuliah.