RANCANG BANGUN APLIKASI ANDROID "STOKITA" UNTUK PENGELOLAAN STOK DAN PENJUALAN ONLINE PADA FAHRI MART

TUGAS AKHIR

Diajukan Oleh:

Hafizul Furqan

NIM. 220705060

Mahasiswa Fakultas Sains dan Teknologi

Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI AR-RANIRY

BANDA ACEH

2026M/1448H

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI ANDROID "STOKITA" UNTUK PENGELOLAAN STOK DAN PENJUALAN ONLINE PADA FAHRI MART

TUGAS AKHIR

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Ar-Raniry Banda Aceh
Sebagai salah satu Beban Studi Memperoleh Gelar Sarjana (S1)
Dalam Ilmu/Prodi Teknologi Informasi

Oleh:

Hafizul Furqan
NIM. 220705060

Mahasiswa Fakultas Sains dan Teknologi

Program Studi Teknologi Informasi

Disetujui untuk Dimunaqasyahkan oleh:

Pembimbing I,	Pembimbing II,
Malahayati, M.T.	**Rahmad Ade Akbar, S.Pd., M.Pd.**
NIP. 198301272015032003	

Mengetahui,

Ketua Program Studi Teknologi Informasi,

Malahayati, M.T.
NIP. 198301272015032003

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI ANDROID "STOKITA" UNTUK PENGELOLAAN STOK DAN PENJUALAN ONLINE PADA FAHRI MART

TUGAS AKHIR

Telah Diuji oleh Panitia Ujian Munaqasyah Tugas Akhir
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan
Dinyatakan Lulus Serta Diterima Sebagai Salah Satu Beban Studi
Program Sarjana (S1)
Dalam Program Studi Teknologi Informasi

Pada Hari/Tanggal: Rabu, 5 Agustus 2026 M
21 Shafar 1448 H

Panitia Ujian Munaqasyah Tugas Akhir:

Ketua,	Sekretaris,
Malahayati, M.T.	**Rahmad Ade Akbar, S.Pd., M.Pd.**
NIP. 198301272015032003	
Penguji I,	Penguji II,
Ghufran Ibnu Yasa, M.T.	**Fathiah, M.Eng.**
NIP. 198409262014031005	**NIP. 198606152019032010**

Mengetahui:

Dekan Fakultas Sains dan Teknologi
UIN Ar-Raniry Banda Aceh,

Prof. Dr. Ir. Muhammad Dirhamsyah, M.T., IPU
NIP. 196210021988111001

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	: Hafizul Furqan
NIM	: 220705060
Program Studi	: Teknologi Informasi
Fakultas	: Sains dan Teknologi
Judul	: Rancang Bangun Aplikasi Android "Stokita" Untuk Pengelolaan Stok dan Penjualan Online Pada Fahri Mart

Dengan ini menyatakan bahwa dalam penulisan tugas akhir ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi terhadap naskah karya orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu bertanggungjawab atas karya ini.

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh, 5 Agustus 2026

Yang Menyatakan

Hafizul Furqan

ABSTRAK

Fahri Mart merupakan toko kelontong di Kabupaten Bireuen yang masih menjalankan proses bisnis secara konvensional, seperti pencatatan transaksi manual dan pengecekan stok visual, yang berisiko pada ketidakakuratan data dan keterbatasan jangkauan pasar. Penelitian ini bertujuan merancang dan membangun aplikasi Android "Stokita" menggunakan metode *SDLC Waterfall* untuk mendigitalisasi manajemen stok dan menyediakan layanan penjualan online. Sistem dibangun dengan arsitektur *Client-Server* memanfaatkan React Native sebagai *frontend*, Node.js dan Express.js sebagai *backend*, serta MySQL sebagai basis data. Pengujian sistem dilakukan melalui *API Testing* dan *Black Box Testing* untuk memastikan fungsionalitas sistem berjalan dengan baik. Hasil pengujian fungsional menunjukkan tingkat keberhasilan sebesar 100%. Evaluasi usability menggunakan metode *System Usability Scale* (SUS) terhadap 52 responden menghasilkan skor rata-rata 80,67, yang termasuk dalam kategori *Acceptable*, *Grade A*, dan predikat *Excellent*. Instrumen SUS dinyatakan sangat andal dengan koefisien *Cronbach's Alpha* 0,927. Dengan demikian, aplikasi Stokita layak diimplementasikan untuk meningkatkan efisiensi operasional dan daya saing digital Fahri Mart.

Kata Kunci: *Stokita, Android, React Native, Node.js, Waterfall, System Usability Scale (SUS).*

ABSTRACT

Fahri Mart is a grocery store in Bireuen Regency that still operates conventionally, characterized by manual transaction logging and visual stock monitoring, and lacks online sales services. These conditions result in operational inefficiencies, risks of data recording errors, and challenges in inventory management and sales reporting. This research aims to design and develop "Stokita," an Android-based application, using the SDLC Waterfall model to digitize stock management and provide online sales services. The system was built with a Client-Server architecture utilizing React Native as the frontend, Node.js and Express.js as the backend, and MySQL as the database. System testing was conducted through API Testing and Black Box Testing to ensure the system functions properly. The functional testing results indicate a 100% success rate. The usability evaluation using the System Usability Scale (SUS) method with 52 respondents yielded an average score of 80.67, categorized as "Acceptable", "Grade A", and an "Excellent" rating. The SUS instrument is declared highly reliable with a Cronbach's Alpha coefficient of 0.927. In conclusion, the Stokita application is feasible to be implemented as a digital solution to enhance operational efficiency and digital competitiveness for Fahri Mart.

Keywords: *Stokita, Android, React Native, Node.js, Waterfall, System Usability Scale (SUS).*

KATA PENGANTAR

Bismillahirrahmanirrahim.

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT, Tuhan Yang Maha Esa, atas segala rahmat, karunia, dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"Rancang Bangun Aplikasi Android Stokita untuk Pengelolaan Stok dan Penjualan Online pada Fahri Mart"** dengan baik. Shalawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, beserta keluarga, sahabat, dan seluruh pengikutnya hingga akhir zaman.

Tugas akhir ini disusun sebagai salah satu syarat untuk menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi. Dalam proses penyusunan tugas akhir ini, penulis menyadari bahwa tanpa bantuan, bimbingan, dan dukungan dari berbagai pihak, karya ilmiah ini tidak akan dapat terselesaikan dengan baik. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

Kedua orang tua tercinta yang senantiasa memberikan doa, dukungan, motivasi, dan kasih sayang yang tiada henti kepada penulis. Semoga Allah SWT membalas segala kebaikan mereka dengan balasan terbaik.

1. Ibu Malahayati, M.T., selaku Ketua Program Studi Teknologi Informasi, Pembimbing Akademik, sekaligus Dosen Pembimbing I, yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, arahan, masukan, serta motivasi kepada penulis selama proses penyusunan tugas akhir ini.
2. Bapak Rahmad Ade Akbar, S.Pd., M.Pd., selaku Dosen Pembimbing II, yang telah memberikan bimbingan, saran, koreksi, dan masukan yang sangat berharga sehingga tugas akhir ini dapat diselesaikan dengan baik.
3. Seluruh dosen Program Studi Teknologi Informasi yang telah memberikan ilmu, pengalaman, dan wawasan kepada penulis selama menempuh pendidikan.

4. Teman-teman seperjuangan yang telah memberikan dukungan, motivasi, serta semangat selama proses penyusunan tugas akhir ini.
5. Semua pihak yang tidak dapat disebutkan satu per satu yang telah membantu, baik secara langsung maupun tidak langsung, dalam penyelesaian tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki kekurangan, baik dari segi isi maupun penyajian. Oleh karena itu, penulis dengan rendah hati mengharapkan kritik dan saran yang membangun demi penyempurnaan di masa yang akan datang.

Akhir kata, semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan, khususnya di bidang teknologi informasi, serta menjadi kontribusi nyata dalam mendukung pengelolaan stok dan penjualan online pada Fahri Mart.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Banda Aceh, 5 Agustus 2026

Penulis,

Hafizul Furqan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era digital kini, banyak UMKM beralih ke penjualan online karena lebih efisien dan menjangkau pelanggan lebih luas. Salah satunya toko kelontong yang merupakan usaha mikro yang cukup vital bagi perekonomian lokal namun pengelolaannya masih cenderung konvensional. Pencatatan penjualan dan stok yang manual rentan kesalahan dan kehilangan data, sementara pemilik toko kesulitan memantau stok secara akurat. Fahri Mart sebagai contoh, hingga saat ini masih mencatat penjualan di buku tulis dan melakukan pengecekan stok barang secara visual. Pelaksanaan tersebut tentu membuat ketidaksesuaian jumlah barang, barang paling laku, dan pelaporan keuntungan terasa sulit. Selain itu, Fahri Mart tidak memiliki kanal online, karena seluruh transaksi dilakukan secara offline, maka jangkauan pasarnya juga terbatas.

Beberapa masalah utama pengelolaan toko kelontong secara manual antara lain: Pencatatan manual rentan kesalahan dan data hilang. Pelaku UMKM tanpa sistem digital sering mengalami kekeliruan hitung dan kehilangan data transaksi (Abadi A. N. 2024). Stok barang sulit dipantau akurat. Tanpa aplikasi, stok hanya dihitung fisik sehingga kerap tidak sesuai catatan penjualan, memicu kekurangan atau kelebihan stok. Tidak ada saluran penjualan online. Ketergantungan pada transaksi fisik membatasi pangsa pasar, padahal adopsi *e-commerce* terbukti memperluas pasar UMKM secara signifikan (Rezkia 2024). Laporan keuangan tidak otomatis. Tanpa sistem, pelaporan keuntungan dan analisis penjualan dilakukan manual yang memakan waktu dan rawan kelalaian.

Penelitian mutakhir menegaskan pentingnya digitalisasi dalam manajemen stok dan penjualan UMKM. Studi di Desa Cinta Rakyat (2025) menunjukkan bahwa implementasi aplikasi kasir Android membantu pelaku UMKM mencatat transaksi *real-time* dan memantau stok barang secara otomatis (Saragih R. 2025). Dengan pengenalan aplikasi kasir digital seperti Kasir Pintar atau Moka POS,

proses pencatatan menjadi lebih efisien dan akurat. Penelitian lain pada UMKM kosmetik 2025 melaporkan bahwa sistem informasi stok berbasis desktop mampu mengoptimalkan kontrol stok, meminimalkan kerugian produk kadaluarsa, serta menyediakan data akurat untuk pengambilan keputusan (Hanani et al. 2025).

Penerapan aplikasi kasir pintar berbasis Android pada UMKM NN Shop meningkatkan efisiensi dan ketepatan pencatatan laporan penjualan(Rahmat & Diyani 2024). Hasil penelitian mereka menemukan bahwa laporan penjualan menjadi lebih terstruktur dan rapi. Contoh lain, studi di UMKM Rilis Kosmetik melaporkan peningkatan kecepatan *input* data, akurasi stok yang lebih baik, serta kemudahan pencatatan transaksi penjualan dengan sistem berbasis Google Sheets (Saputra et al. 2025). Temuan-temuan ini memperkuat bahwa digitalisasi pencatatan dan manajemen stok mampu menggantikan proses manual menjadi lebih terstruktur dan efektif.

Penggunaan platform *e-commerce* membawa dampak positif signifikan bagi operasional bisnis UMKM. Penelitian tersebut menemukan bahwa *e-commerce* meningkatkan keamanan dan kenyamanan proses bisnis, memudahkan komunikasi dengan pelanggan, serta memperluas jangkauan pasar hingga ke skala global(Lyonita et al. 2024). Temuan ini menegaskan bahwa meskipun terdapat tantangan seperti kebutuhan pelatihan dan sumber daya manusia, adopsi platform digital tetap menjadi strategi kunci bagi UMKM untuk mencapai pasar lebih luas, meningkatkan efisiensi operasional, dan memberikan layanan yang lebih baik di era digital.

Penelitian-penelitian tersebut menyimpulkan bahwa solusi digital terintegrasi, baik berupa aplikasi mobile kasir maupun sistem inventaris, dapat menggantikan proses manual dan meningkatkan efisiensi manajemen UMKM. Selain itu, studi tentang *e-commerce* menunjukan bahwa platform digital menjadi pendorong utama pertumbuhan UMKM. Dalam konteks pengembangan aplikasi, penerapan *framework* cross-platform seperti React Native terbukti mempercepat pembuatan aplikasi Android/iOS dengan antarmuka modern dan proses pengembangan yang efisien (Asoka et al. 2024).

Melihat kondisi tersebut, diperlukan aplikasi mobile terpadu untuk stok dan penjualan. Penelitian ini mengusulkan Stokita, aplikasi Android berbasis React Native untuk toko kelontong Fahri Mart. React Native dipilih karena memungkinkan pengembangan lintas-platform yang cepat dan tampilan modern. Stokita mengintegrasikan fitur utama seperti manajemen stok otomatis, pencatatan transaksi online/offline, integrasi Admin–Pembeli, serta laporan dan analisis *real-time* untuk mendukung pengambilan keputusan. Dengan aplikasi Stokita, diharapkan Fahri Mart dapat mendigitalkan operasionalnya, sehingga stok terpantau akurat, pencatatan transaksi lebih andal, dan layanan penjualan daring terlaksana.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat dirumuskan beberapa pertanyaan penelitian sebagai berikut:

1) Bagaimana merancang dan membangun aplikasi Android "Stokita" yang mendukung pengelolaan stok barang dan penjualan online pada Fahri Mart?
2) Bagaimana implementasi dan pengujian fungsional aplikasi Android "Stokita" dalam menangani pengelolaan produk, stok barang, transaksi penjualan, dan pemesanan online pada Fahri Mart?
3) Bagaimana tingkat kemudahan penggunaan (*usability*) aplikasi Android "Stokita" berdasarkan persepsi pengguna di Fahri Mart menggunakan metode *System Usability Scale* (SUS)?

1.3 Ruang Lingkup

Untuk menjaga fokus dan ketercapaian tujuan penelitian, maka penelitian ini memiliki batasan sebagai berikut:

1) Penelitian ini mencakup proses perancangan dan pengembangan aplikasi Android “Stokita” menggunakan framework React Native yang terintegrasi dengan fitur utama pengelolaan stok dan penjualan online pada Fahri Mart. Fitur tersebut meliputi pencatatan stok masuk dan stok keluar, pemantauan stok habis atau menipis, pencatatan transaksi penjualan

offline dan online, notifikasi pesanan dan status pengiriman, serta laporan pendapatan.

2) Penelitian ini difokuskan pada aspek teknis dan *usability*, tanpa membahas secara mendalam aspek keamanan data, integrasi dengan sistem pembayaran eksternal, ataupun pengelolaan multi-cabang.
3) Pengujian aplikasi dilakukan pada lingkungan Fahri Mart sebagai studi kasus, tidak mencakup generalisasi untuk semua jenis toko kelontong.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1) Merancang dan membangun aplikasi Android "Stokita" yang yang mendukung pengelolaan stok barang dan penjualan online pada Fahri Mart.
2) Mengimplementasikan dan menguji fungsional aplikasi Android "Stokita" dalam menangani pengelolaan produk, stok barang, transaksi penjualan, dan pemesanan online pada Fahri Mart.
3) Mengukur tingkat kemudahan penggunaan (*usability*) aplikasi Android "Stokita" berdasarkan persepsi pengguna di Fahri Mart menggunakan metode *System Usability Scale* (SUS).

1.5 Manfaat Penelitian

Dengan adanya penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

A. Manfaat Praktis

Penelitian ini memberikan manfaat langsung bagi Fahri Mart dalam bentuk aplikasi "Stokita" yang dapat membantu proses pencatatan stok, transaksi penjualan, serta pemantauan barang habis atau menipis secara otomatis. Dengan adanya sistem ini, Fahri Mart dapat mengurangi kesalahan pencatatan manual, mempercepat proses transaksi, dan mulai membuka layanan penjualan online secara terintegrasi.

B. Manfaat Teoritis

Penelitian ini memberikan kontribusi pada bidang sistem informasi dan aplikasi mobile, khususnya dalam penerapan konsep *real-time notification*, *barcode scanning*, dan *online transaction management* berbasis Android. Konsep yang diterapkan dapat menjadi acuan bagi pengembangan sistem serupa pada usaha mikro dan kecil lainnya.

C. Manfaat Akademik

Secara akademik, penelitian ini memperkaya referensi dalam bidang rekayasa perangkat lunak dan *mobile computing*, khususnya pada penerapan sistem manajemen stok dan transaksi penjualan berbasis Android.

D. Manfaat Sosial dan Ekonomi

Aplikasi "Stokita" diharapkan dapat membantu pelaku UMKM seperti Fahri Mart untuk bertransformasi ke arah digitalisasi usaha, meningkatkan efisiensi operasional, memperluas jangkauan penjualan, serta meningkatkan daya saing di era ekonomi digital

E. Manfaat untuk Penulis

Penelitian ini memberikan pengalaman langsung kepada penulis dalam menerapkan teori pengembangan perangkat lunak ke dalam praktik nyata melalui pembangunan aplikasi mobile. Penulis memperoleh wawasan mendalam mengenai analisis kebutuhan sistem, proses perancangan antarmuka pengguna, serta pengujian fungsionalitas dan *usability*. Selain itu, pengembangan produk ini juga meningkatkan keterampilan penulis dalam menggunakan *framework* React Native dan mengelola proyek pengembangan sistem secara terstruktur, yang berguna sebagai bekal profesional di bidang teknologi informasi.

1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini disusun ke dalam lima bab utama sebagai berikut:

a) **BAB I PENDAHULUAN**: Menguraikan latar belakang masalah mengenai urgensi digitalisasi pada Fahri Mart, rumusan masalah, batasan ruang lingkup penelitian, tujuan, manfaat penelitian, serta sistematika penulisan.

b) **BAB II TINJAUAN PUSTAKA**: Menyajikan teori-teori relevan terkait sistem informasi, aplikasi mobile, metode Waterfall, serta ekosistem teknologi pengembang seperti React Native, Node.js, Express.js, MySQL, dan metode evaluasi SUS.

c) **BAB III METODE PENELITIAN**: Memaparkan prosedur rekayasa sistem yang meliputi metode pengumpulan data melalui observasi partisipatif dan wawancara, analisis kebutuhan fungsional dan non-fungsional, serta perancangan teknis terperinci yang mencakup *Use Case Diagram*, *Entity Relationship Diagram* (ERD), spesifikasi basis data, perancangan algoritma (*pseudocode*), arsitektur RESTful API, dan rancangan antarmuka pengguna (UI/UX).

d) **BAB IV HASIL DAN PEMBAHASAN**: Menyajikan hasil implementasi sistem pada sisi *backend* dan *frontend*, hasil pengujian RESTful API menggunakan Postman, serta pengujian fungsional *Black Box*. Bab ini juga membahas secara mendalam hasil pengujian *usability* SUS, termasuk uji validitas dan reliabilitas instrumen, serta interpretasi skor akhir terhadap efektivitas digitalisasi operasional toko.

e) **BAB V KESIMPULAN DAN SARAN**: Memuat kesimpulan dari seluruh rangkaian penelitian dan pengujian aplikasi Stokita, serta saran-saran konstruktif untuk pengembangan sistem di masa mendatang, seperti integrasi *payment gateway* dan pengembangan versi iOS.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Dalam upaya mengembangkan aplikasi manajemen stok dan penjualan untuk toko kelontong berbasis Android, berbagai penelitian sebelumnya telah dilakukan dengan pendekatan, teknologi, dan fokus yang berbeda-beda. Studi-studi ini memberikan landasan penting dalam memahami praktik terbaik dan tantangan yang mungkin dihadapi dalam pengembangan sistem serupa, khususnya untuk mendukung digitalisasi UMKM di sektor ritel.

Penelitian oleh Abadi & Ardiani (2024) merancang aplikasi Android untuk manajemen stok dan penjualan buku di Musi Bookstore dengan fitur pemindaian barcode menggunakan Google ML Kit. Aplikasi tersebut dibangun menggunakan Android Studio (Kotlin) dan basis data MySQL, serta mengikuti model pengembangan waterfall. Hasil pengujian menunjukkan aplikasi ini mampu meningkatkan akurasi dan efisiensi pencatatan transaksi serta stok secara keseluruhan. Demikian pula, Putra & Sancoko (2024) mengembangkan aplikasi *Point of Sale* berbasis Android untuk UMKM Toko Mapan dengan backend *Firebase Realtime Database*. Mereka menggunakan metode pengembangan *Extreme Programming* (XP) dalam siklusnya. Hasil uji *Black Box* menunjukkan aplikasi POS ini mempermudah pencatatan transaksi toko dan semua fungsinya berjalan baik sesuai spesifikasi.

Penelitian lainnya menekankan kemudahan penggunaan dan kepuasan pengguna. Ramadhani (2024) menerapkan metode *Design Thinking* untuk merancang aplikasi mobile bagi UMKM kuliner (UMKM Amplang Cita Rasa). Fokusnya adalah menyederhanakan proses transaksi UMKM dan meningkatkan efisiensi operasional. Hasil uji kepuasan dengan 10 responden menunjukkan lebih dari 80% menilai aplikasi berbasis mobile ini membantu kebutuhan operasional UMKM. Sejalan dengan upaya tersebut, Oktavianus dkk. (2023) mengembangkan sistem manajemen stok barang berbasis mobile untuk mempermudah pemantauan persediaan secara *real-time*. Penelitian ini berfokus pada penyediaan solusi kontrol

stok digital yang praktis agar pemilik usaha dapat melacak sirkulasi barang masuk dan keluar langsung melalui ponsel pintar guna menghindari kesalahan selisih stok fisik.

Manurung (2024) menyajikan contoh sistem informasi penjualan Android terintegrasi untuk UMKM (kasus Kugar Minamas Pansela) yang dibangun dengan metode *prototyping*. Aplikasi ini dikembangkan di Android Studio (Java/Kotlin) dan mengintegrasikan modul-modul terkait penjualan dan stok. Evaluasi dengan paired sample test menemukan bahwa waktu transaksi rata-rata berkurang signifikan (dari 7,42 menit menjadi 3,81 menit), artinya proses penjualan menjadi lebih cepat. Secara keseluruhan, penelitian-penelitian ini menyoroti transformasi digital UMKM melalui aplikasi mobile yang menyederhanakan pencatatan stok dan penjualan, walaupun model pengembangan dan teknologi yang dipakai beragam (misalnya *waterfall*, XP, *Design Thinking*, Flutter, Firebase, ML Kit, *prototyping*, dll.).

Beberapa penelitian terkait yang relevan dengan topik ini telah menjadi landasan penting dalam perumusan proposal ini. Ringkasan dari penelitian-penelitian tersebut disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terkait

No	Referensi	Metode	Deskripsi Singkat	Hasil
1	Abadi & Ardiani (2024)	*Waterfall*	Aplikasi Android untuk manajemen stok dan penjualan buku (Musi Bookstore) dengan scanner barcode ML Kit	Meningkatkan akurasi dan efisiensi pencatatan transaksi dan stok
2	Putra & Sancoko (2024)	*Extreme Programming* (XP)	Aplikasi *Point of Sale* (POS) Android dengan backend Firebase untuk toko	Aplikasi mempermudah proses transaksi toko, semua fungsi

No	Referensi	Metode	Deskripsi Singkat	Hasil
			kelontong Toko Mapan	berjalan baik sesuai uji Blackbox
3	Ramadhani dkk. (2024)	*Design Thinking*	Aplikasi mobile untuk UMKM Amplang Cita Rasa (kuliner) dengan fokus kemudahan transaksi dan UI/UX	>80% responden menilai aplikasi berbasis mobile ini membantu operasional UMKM (hasil kuesioner)
4	Oktavianus dkk. (2023)	–	Sistem manajemen stok barang berbasis mobile untuk mempermudah monitoring persediaan secara *real-time*.	Mempermudah pelacakan sirkulasi barang masuk dan keluar secara akurat dari perangkat seluler
5	Manurung dkk. (2024)	*Prototype*	Sistem informasi penjualan terintegrasi Android untuk UMKM (Kugar Minamas), dibangun di Android Studio (Java/Kotlin)	Waktu transaksi berkurang signifikan (rata-rata 7,42→3,81 menit) sehingga mempercepat proses penjualan

Berdasarkan hasil kajian terhadap beberapa penelitian terdahulu, dapat disimpulkan bahwa pengembangan aplikasi Android untuk pengelolaan stok dan penjualan telah terbukti memberikan manfaat nyata bagi pelaku UMKM, terutama dalam hal efisiensi waktu, ketepatan pencatatan, dan peningkatan layanan pelanggan. Namun, penelitian-penelitian tersebut masih memiliki fokus yang berbeda-beda, seperti pengelolaan stok dan penjualan buku, sistem Point of Sale

(POS), pemantauan persediaan, maupun peningkatan kemudahan penggunaan aplikasi.

Penelitian ini memiliki perbedaan dengan penelitian terdahulu pada integrasi fungsi yang dikembangkan dalam satu aplikasi. Aplikasi Stokita tidak hanya berfungsi untuk pengelolaan stok dan transaksi penjualan offline, tetapi juga menyediakan penjualan online yang terintegrasi dengan pengelolaan pesanan, perubahan status pesanan, pembaruan stok, serta laporan penjualan. Selain itu, Stokita menyediakan fitur pendukung pelanggan berupa keranjang belanja, produk favorit, daftar belanja, pengelolaan alamat, dan notifikasi. Pada sisi admin, aplikasi juga menyediakan pengelolaan produk, penambahan stok (stock in), kasir, pengelolaan pesanan, serta laporan penjualan.

Dengan adanya integrasi tersebut, Stokita diharapkan dapat mengisi kebutuhan yang belum secara keseluruhan dicakup oleh penelitian terdahulu, yaitu penggabungan pengelolaan stok, penjualan offline, penjualan online, dan layanan pendukung pelanggan dalam satu aplikasi Android yang diterapkan pada toko kelontong Fahri Mart.

2.2 Rancang Bangun

Rancang bangun merupakan proses merancang dan membangun suatu sistem berdasarkan kebutuhan pengguna untuk menyelesaikan permasalahan tertentu. Dalam pengembangan aplikasi, rancang bangun mencakup tahapan analisis kebutuhan, perancangan sistem dan antarmuka, implementasi, serta pengujian untuk menghasilkan aplikasi yang sesuai dengan kebutuhan pengguna.

Dalam penelitian ini, rancang bangun aplikasi Stokita menggunakan metode Waterfall, yaitu model pengembangan yang dilakukan secara berurutan melalui tahapan analisis kebutuhan, perancangan, implementasi, dan pengujian. Setiap tahapan diselesaikan sebelum dilanjutkan ke tahap berikutnya sehingga proses pengembangan dapat dilakukan secara sistematis dan terdokumentasi.

Metode Waterfall dipilih karena kebutuhan aplikasi Stokita telah dapat ditentukan sejak awal dan proses pengembangannya membutuhkan tahapan yang terstruktur. Dengan metode ini, proses analisis kebutuhan, perancangan,

implementasi, dan pengujian dapat dilakukan secara sistematis sehingga mendukung proses rancang bangun aplikasi Stokita secara terarah.

2.3 Sistem Informasi

Sistem Informasi (SI) merupakan suatu sistem yang terdiri atas elemen, teknologi, data, proses, dan sumber daya manusia yang saling berinteraksi untuk mengumpulkan, mengolah, menyimpan, dan menyajikan informasi guna mendukung kegiatan organisasi. SI berperan dalam mengotomatisasi proses bisnis serta menyediakan informasi yang terstruktur dan tepat waktu untuk mendukung pengambilan keputusan (Muhammad et al., 2025).

Dalam konteks UMKM atau toko kelontong, SI dapat digunakan untuk mengintegrasikan berbagai aktivitas operasional, seperti penjualan, pembelian, pengelolaan stok, data pelanggan, dan penyusunan laporan. Penerapan SI terbukti dapat meningkatkan efisiensi operasional dan akurasi pengelolaan data UMKM (Hasan R., 2024; Minasa S., 2024). Selain itu, aplikasi berbasis mobile dapat memberikan kemudahan akses terhadap informasi usaha dan mendukung pengelolaan transaksi secara terpadu (Muhammad et al., 2025; Nurlaela L., 2024).

Kemudahan penggunaan juga menjadi aspek penting dalam pengembangan SI untuk UMKM karena sistem umumnya digunakan oleh pengguna dengan tingkat kemampuan teknologi yang beragam. Oleh karena itu, antarmuka yang sederhana dan mudah dipahami diperlukan agar sistem dapat digunakan secara efektif dan meningkatkan penerimaan pengguna (Hasan R., 2024; Muhammad et al., 2025).

Berdasarkan konsep tersebut, aplikasi Stokita merupakan implementasi SI berbasis Android yang mengintegrasikan pengelolaan stok dan penjualan untuk mendukung kegiatan operasional Fahri Mart. Sistem ini menyediakan pengelolaan produk dan stok, transaksi penjualan offline, pemesanan online, serta laporan penjualan dalam satu aplikasi sehingga informasi dapat dikelola secara lebih terstruktur dan mudah diakses.

2.4 Aplikasi Mobile

Aplikasi mobile adalah perangkat lunak khusus yang dirancang untuk dijalankan pada perangkat bergerak seperti smartphone dan tablet, dengan

antarmuka serta fitur yang dioptimalkan untuk layar sentuh dan mobilitas pengguna. Pengembangan aplikasi mobile umumnya mempertimbangkan karakteristik unik ponsel (layar kecil, koneksi seluler, sensor, dll.), serta kebutuhan lintas platform (Android/iOS).

Aplikasi mobile memungkinkan UMKM meningkatkan efisiensi operasional dan memperluas pasar. Melalui aplikasi mobile, pelaku UMKM bisa mencatat transaksi penjualan secara *real-time*, memantau stok barang, dan menghasilkan laporan keuangan instan. Selain itu, aplikasi mobile dapat digunakan sebagai saluran pemasaran dan pemesanan online, sehingga UMKM meraih pelanggan lebih luas. Misalnya, Saragih R. (2025) menemukan bahwa penerapan aplikasi kasir Android (seperti Kasir Pintar atau Moka POS) memungkinkan UMKM pedesaan mencatat transaksi secara *real-time* dan memantau stok secara otomatis, meningkatkan akurasi pengelolaan keuangan.

Aplikasi mobile yang ditujukan untuk pelaku UMKM sering dirancang dengan antarmuka yang intuitif dan proses yang sederhana. Hal ini penting karena sebagian besar pengguna UMKM bukan ahli teknologi. Desain *user-centered* (berbasis kebutuhan pengguna) membantu memastikan aplikasi mudah dipelajari dan digunakan, mirip temuan Hasan dkk. (2024) untuk sistem informasi berbasis web UMKM. Selain itu, pelatihan pengguna dan dukungan teknis juga menjadi kunci keberhasilan implementasi, terutama untuk mengatasi kesenjangan literasi digital (Saragih R., 2025).

Konsep dan fungsi di atas diadopsi pada pengembangan Stokita. Sebagai aplikasi Android berbasis React Native, Stokita menggabungkan fungsi kasir (pencatatan penjualan), manajemen stok otomatis, dan laporan keuangan dalam satu platform mobile. Pendekatan React Native memungkinkan pengembangan cepat lintas Android/iOS dengan satu kode sumber. Dengan antarmuka yang ramah pengguna, Stokita menerapkan prinsip yang sama seperti aplikasi kasir digital lainnya: transaksi *real-time*, *tracking stock*, dan laporan keuangan instan. Pendekatan ini sejajar dengan literatur bahwa aplikasi mobile bagi UMKM mendukung automasi operasional harian dan digitalisasi usaha untuk meningkatkan efisiensi dan daya saing.

2.5 Pengelolaan Stok

Pengelolaan stok merupakan proses perencanaan dan pengendalian ketersediaan barang agar sesuai dengan kebutuhan serta mencegah terjadinya kelebihan maupun kekurangan persediaan. Aktivitas pengelolaan stok meliputi pencatatan barang masuk dan keluar, penyesuaian stok, serta pemantauan persediaan secara berkala untuk memastikan data stok tetap akurat.

Pada usaha kecil seperti toko kelontong, pengelolaan stok masih sering dilakukan secara manual sehingga rentan terhadap kesalahan pencatatan, kesulitan dalam pelacakan barang, dan keterlambatan memperoleh informasi persediaan. (Ho J. A., 2025) menunjukkan bahwa pencatatan manual dapat menurunkan efisiensi operasional dan menyulitkan pemantauan stok.

Penerapan sistem informasi dapat membantu mengatasi permasalahan tersebut melalui pencatatan dan pembaruan stok secara digital. Sistem pengelolaan stok berbasis digital dapat meningkatkan kecepatan pencatatan, mengurangi kesalahan manusia, serta menyediakan informasi persediaan yang lebih akurat dan terkini (Bisma Muhammad Hermawan et al., 2025; Christian & Voutama, 2024).

Dalam konteks aplikasi Stokita, fitur pengelolaan stok dirancang untuk mempermudah admin Fahri Mart dalam mencatat setiap transaksi pembelian dan penjualan, baik secara online maupun offline. Setiap transaksi secara otomatis mengurangi kuantitas stok pada sistem, dan admin akan menerima notifikasi saat stok barang mencapai batas minimum. Pendekatan ini tidak hanya meningkatkan efisiensi kerja toko, tetapi juga membantu pemilik usaha memantau kondisi persediaan secara *real-time* tanpa perlu melakukan pengecekan manual setiap saat.

2.6 Jual Beli Online

Jual beli online atau *e-commerce* adalah proses transaksi komersial yang dilakukan melalui internet. *E-commerce* dapat diartikan sebagai “proses transaksi jual beli yang dilakukan melalui internet” (Dermawan D. T. B., 2023). Melalui sistem ini, penjual dan pembeli dapat melakukan transaksi kapan saja dan di mana saja, tanpa perlu bertatap muka langsung, sehingga transaksi menjadi lebih efisien. Dalam praktiknya, *e-commerce* memanfaatkan jaringan komputer untuk pertukaran

barang, jasa, dan informasi secara elektronik, memungkinkan pembeli memperoleh produk yang diinginkan dengan cepat dan mudah.

Perkembangan teknologi informasi yang pesat telah mengubah pola perilaku konsumen, di mana banyak orang kini lebih memilih berbelanja online karena kenyamanan, kemudahan akses, dan keberagaman pilihan produk yang ditawarkan. Keuntungan lain dari berjualan secara online adalah jangkauan pasar yang sangat luas bahkan hingga tingkat global.

Implementasi sistem jual beli online perlu memperhatikan bukan hanya desain antarmuka pengguna (*User Interface*) tetapi juga manajemen data transaksi, keamanan sistem, dan kecepatan proses. Penelitian terkini menegaskan bahwa aspek kemudahan penggunaan (*usability*), kejelasan informasi produk, keamanan data pribadi, dan keandalan sistem merupakan faktor krusial dalam membangun kepercayaan pengguna pada platform *e-commerce*. Misalnya, studi oleh Anugraha A. Z. (2025) menunjukkan bahwa komponen penting dalam membangun kepercayaan konsumen meliputi keamanan data, kemudahan penggunaan aplikasi, serta kejelasan informasi produk. Dengan sistem yang aman, transparan, dan mudah digunakan, pelanggan akan merasa nyaman bertransaksi, sehingga kemungkinan mereka melakukan pembelian ulang dan loyalitas terhadap toko online semakin tinggi. Demikian pula, Septian et al. (2019) menegaskan bahwa kepercayaan konsumen (*e-trust*) menjadi salah satu kunci keberhasilan toko online. Secara keseluruhan, integrasi fitur yang responsif dan terpercaya pada aplikasi *e-commerce* sangat penting untuk mendukung pengalaman berbelanja pelanggan.

Dalam konteks Fahri Mart, penerapan fitur jual beli online pada aplikasi "Stokita" bertujuan memperluas jangkauan pemasaran sekaligus memudahkan pelanggan bertransaksi. Sebelumnya Fahri Mart hanya melayani transaksi secara langsung di toko, sehingga dengan hadirnya sistem Stokita, pelanggan dapat melihat daftar produk, melakukan pemesanan, dan menerima konfirmasi transaksi secara digital. Lebih lanjut, fitur jual beli online ini terintegrasi dengan manajemen stok barang. Setiap kali terjadi transaksi, data stok akan terupdate secara otomatis, sehingga persediaan produk selalu akurat dan *up-to-date*. Hasil penelitian menunjukkan bahwa penggunaan sistem omnichannel atau terintegrasi semacam ini

secara signifikan meningkatkan akurasi pencatatan stok, mempercepat proses restock, dan mengurangi risiko kehabisan stok (Gayatri, 2025). Dengan demikian, admin toko dapat terhindar dari risiko menjual produk yang sudah habis dan pelanggan tetap mendapatkan layanan yang memuaskan. Lyonita et al. (2024) juga menekankan bahwa *e-commerce* memungkinkan UMKM mencapai pasar lebih luas dan meningkatkan efisiensi operasional bisnis. Oleh karena itu, diharapkan penerapan sistem jual beli online melalui aplikasi Stokita akan meningkatkan efisiensi operasional Fahri Mart dan memperkuat daya saing usaha di era digital.

2.7 Fahri Mart

Fahri Mart merupakan toko kelontong yang berdiri sejak tahun 2012 dan dikelola oleh Ibu Nilawati. Toko ini berlokasi di Komplek Terminal Lama, Kecamatan Kota Juang, Kabupaten Bireuen, Aceh. Sebagai usaha mikro yang berorientasi pada kebutuhan masyarakat sekitar, Fahri Mart menyediakan berbagai produk kebutuhan harian, antara lain bahan sembako, makanan ringan, minuman, peralatan mandi, perlengkapan mencuci, rokok, serta layanan pulsa dan paket data. Toko ini beroperasi setiap hari dengan jam buka sekitar pukul 10.00 hingga 00.00 WIB.

Dalam aktivitas operasionalnya, Fahri Mart masih menggunakan sistem konvensional. Pencatatan transaksi penjualan dilakukan secara manual di buku tulis tanpa adanya sistem digital untuk mencatat data stok maupun transaksi harian. Pemantauan stok barang dilakukan secara langsung dengan melihat kondisi fisik barang di rak penjualan. Apabila terdapat produk yang habis atau hampir habis, pemilik baru melakukan pembelian ulang ke pemasok. Cara ini kerap menimbulkan kendala seperti kesulitan dalam melakukan rekap penjualan, keterlambatan mengetahui stok menipis, dan sering lupa terhadap jenis barang yang telah habis ketika akan berbelanja ulang.

Kondisi tersebut menunjukkan perlunya sistem yang mampu membantu proses pencatatan dan pemantauan stok secara otomatis. Selain itu, adanya keinginan untuk memperluas jangkauan penjualan melalui media digital menjadi faktor utama pengembangan aplikasi Stokita. Melalui aplikasi ini, pengelola dapat lebih mudah mencatat transaksi penjualan, memantau ketersediaan barang, serta memberikan

kemudahan bagi pelanggan untuk melakukan pembelian secara online tanpa harus datang langsung ke toko.

Penerapan Stokita diharapkan dapat membantu Fahri Mart bertransformasi menuju sistem pengelolaan toko yang lebih modern dan efisien. Dengan digitalisasi manajemen stok dan penjualan, Fahri Mart dapat meningkatkan akurasi data, efisiensi operasional, serta memperluas jangkauan pasar di wilayah sekitarnya.

2.8 React Native

React Native merupakan salah satu *framework open-source* hasil pengembangan Meta (Facebook) yang menggunakan bahasa pemrograman JavaScript dan pustaka React untuk membangun aplikasi mobile native Android maupun iOS dengan satu basis kode yang sama. Keunggulan utama React Native adalah konsep "*write once, run anywhere*", di mana kode yang ditulis dapat digunakan kembali di berbagai platform tanpa modifikasi besar. Hal ini tidak hanya memudahkan penyusunan antarmuka pengguna secara terstruktur, tetapi juga meningkatkan efisiensi pengembangan dan modularity aplikasi. Dengan pendekatan tersebut, komponen-komponen UI yang sama dapat digunakan kembali di berbagai bagian aplikasi, sehingga mempermudah perawatan dan perluasan sistem di masa mendatang.

Selain itu, React Native menyediakan fitur-fitur pengembangan yang mempercepat siklus kerja pengembang. Misalnya, keberadaan *hot-reloading* atau *live-reload* memungkinkan pengembang melihat perubahan kode secara *real-time* tanpa perlu membangun ulang aplikasi. Maulana (2024) bahkan mengamati bahwa React Native sangat populer karena memiliki satu basis kode tunggal yang menghemat waktu dan usaha pengembangan, didukung oleh komunitas pengembang yang besar dengan banyak plugin serta pustaka pihak ketiga, dan kemampuan Live Reload yang mempercepat iterasi pengembangan. Dengan kata lain, ekosistem React Native yang luas (ribuan pustaka dan plugin) memudahkan pengembang menambah fungsionalitas seperti navigasi, autentikasi, hingga integrasi fitur perangkat keras (kamera, sensor, dsb.) tanpa harus membangun semuanya dari nol.

Dari perspektif pengembangan aplikasi, React Native sangat mendukung integrasi dengan teknologi *backend* modern. Andrianto et al. (2025) menjelaskan bahwa penerapan React Native pada sisi *frontend* dapat mempercepat proses pengembangan aplikasi mobile lintas platform dan efisien dalam pengelolaan komponen UI. Mereka juga mencatat bahwa React Native mudah dihubungkan dengan *backend* berbasis Node.js/Express dan basis data MySQL, sehingga menghasilkan aplikasi yang ringan, fleksibel, dan andal. Dengan dukungan infrastruktur seperti ini, pengembang dapat merancang antarmuka pengguna yang dinamis dan interaktif sambil menjaga performa tetap optimal. Berdasarkan studi kasus nyata, penggunaan React Native memang terbukti efektif: Asoka et al. (2024) menemukan bahwa aplikasi *e-commerce* berbasis React Native berhasil meningkatkan efisiensi pengembangan dibanding metode native tradisional, serta menghasilkan kepuasan pengguna tinggi terutama dalam kemudahan penggunaan dan tampilan antarmuka.

Melalui sifat lintas-platform dan ekosistemnya yang kaya, React Native memungkinkan pengembangan fitur-fitur penting pada aplikasi Stokita secara efisien. Misalnya, aplikasi Stokita dapat menyertakan sistem pengelolaan stok barang, modul penjualan online, notifikasi *real-time*, maupun pemindaian kode batang (barcode) dengan responsivitas yang baik. Berdasarkan literatur di atas, dapat disimpulkan bahwa React Native adalah pilihan tepat untuk membangun aplikasi mobile modern yang memungkinkan pengembangan cepat dengan satu kode basis, tampilan antarmuka yang kaya komponen, serta dukungan integrasi penuh dengan *backend* dan database tanpa mengorbankan performa aplikasi.

2.9 Node.js dan Express.js

Node.js merupakan runtime environment JavaScript yang memungkinkan kode JavaScript dijalankan pada sisi server. Node.js menggunakan arsitektur event-driven dan non-blocking I/O sehingga mampu menangani berbagai permintaan secara efisien dan mendukung pengembangan aplikasi berbasis server (Pragestu S., 2023).

Express.js merupakan framework berbasis Node.js yang digunakan untuk membangun aplikasi web dan RESTful API. Express.js menyediakan mekanisme

routing dan middleware yang mempermudah pengembangan layanan API secara modular dan terstruktur (Hidayattullah M. S., 2025).

Penggunaan Node.js dan Express.js dalam pengembangan aplikasi Stokita diterapkan pada sisi backend untuk menangani komunikasi antara aplikasi Android dan basis data MySQL. Backend menyediakan RESTful API untuk mengelola autentikasi pengguna, data produk, stok, keranjang, pesanan, transaksi penjualan, dan laporan. Setiap permintaan dari aplikasi diproses oleh server dan dikembalikan dalam format JSON, sehingga proses pertukaran data antara aplikasi dan server dapat dilakukan secara terstruktur (Cahyono S. A. B., 2023).

2.10 MySQL

MySQL merupakan sistem manajemen basis data relasional (Relational Database Management System/RDBMS) open-source yang menggunakan SQL untuk menyimpan, mengelola, dan mengambil data. MySQL mendukung pengelolaan data secara terstruktur serta hubungan antar tabel, sehingga sesuai digunakan pada aplikasi pengelolaan stok dan penjualan.

MySQL dapat diintegrasikan dengan berbagai teknologi backend, termasuk Node.js dan Express.js. Integrasi tersebut memungkinkan aplikasi mengakses dan memproses data melalui RESTful API. Penggunaan MySQL pada aplikasi berbasis mobile juga telah diterapkan dalam penelitian Andrianto M. R., (2025) yang menggunakan React Native, Node.js, dan MySQL untuk pengembangan aplikasi mobile.

Pada aplikasi Stokita, MySQL digunakan sebagai basis data utama untuk menyimpan dan mengelola data pengguna, kategori, produk, stok barang, keranjang, favorit, pesanan, transaksi penjualan, serta notifikasi. Data-data tersebut disusun dalam tabel yang saling berelasi menggunakan primary key dan foreign key sehingga pengelolaan data dapat dilakukan secara terstruktur dan konsisten.

Dengan demikian, MySQL dipilih sebagai basis data Stokita karena mendukung penyimpanan data relasional yang sesuai dengan kebutuhan pengelolaan stok, transaksi penjualan, dan pemesanan online.

2.11 Metode Waterfall

Metode Waterfall adalah model SDLC klasik beralur linier, diperkenalkan Royce (1970). Setiap fase dijalankan berurutan (analisis, desain, implementasi, verifikasi, pemeliharaan) dan tahap berikutnya baru dimulai setelah sebelumnya selesai. Model ini cocok untuk kebutuhan proyek yang jelas dan stabil, dengan dokumentasi lengkap. Menurut Wahid (2020) waterfall menuntut spesifikasi kebutuhan terdokumentasi sejak awal hingga pemeliharaan, tanpa mudah kembali ke tahap sebelumnya.

Berdasarkan literatur (Wahid 2020 dan Pranoto 2021), fase utama Waterfall meliputi:

1. Analisis Kebutuhan: Menampung kebutuhan pengguna dan sistem (*use-case*, SRS). Deliverable: dokumen kebutuhan, user stories, yang menjadi acuan sistem. Contoh Stokita: daftar fitur stok/penjualan yang diinginkan pemilik toko.
2. Desain Sistem: Merancang arsitektur, UI, dan database. Deliverable: diagram arsitektur, mockup UI, ERD. Misal: skema UI Stokita (input barang, kasir, laporan) dan skema basis data produk.
3. Implementasi (Pengkodean): Mengkode aplikasi (misalnya React Native untuk Stokita). Deliverable: kode sumber dan paket aplikasi jadi (.apk). Setiap modul diuji unit di tahap ini.
4. Verifikasi/Pengujian: Melakukan pengujian sistem (*black box*, *unit test*). Deliverable: laporan pengujian dan daftar bug. Misalnya, pengujian kasus stok habis, transaksi, login admin di Stokita.
5. Pemeliharaan: Menjalankan sistem di lingkungan nyata dan memperbaiki bug. Deliverable: pembaruan (*update*) dan dokumentasi pemeliharaan. Contoh: patch fitur scan barcode baru atau perbaikan tampilan.

Kelebihan Waterfall Menurut penelitian mutakhir, yaitu dapat menghasilkan produk berkualitas karena pelaksanaan bertahap. Struktur logisnya memudahkan estimasi biaya dan pemantauan kemajuan. Tahapan yang jelas memberi titik awal

dan akhir proyek terdefinisi. Hal ini sesuai untuk Stokita yang memiliki kebutuhan fungsi stok/penjualan terdefinisi (perubahan minimum).

Waterfall dipilih karena kebutuhan Stokita relatif terukur. Berdasarkan Wahid (2020) dan Kismawan, (2025), Waterfall cocok untuk proyek dengan kebutuhan terdefinisi jelas. Stokita melibatkan tim kecil yang dapat mengikuti tiap fase secara terstruktur. Dokumentasi minimal yang dipadatkan tetap disiapkan (misalnya SRS sederhana) sebagai panduan bersama. Dengan menggunakan Waterfall, pengembang dapat memastikan setiap fitur (*input* stok, transaksi, laporan) teruji lengkap sebelum lanjut fase selanjutnya. Pendekatan ini memudahkan evaluasi dan penyempurnaan sistem kasir Stokita secara sistematis.

2.12 Black Box Testing

Black Box Testing atau yang dikenal juga dengan sebutan pengujian fungsional (*functional Testing*) merupakan salah satu metodologi pengujian perangkat lunak klasik yang berfokus pada evaluasi perilaku eksternal sistem tanpa memperhatikan, menganalisis, atau mengetahui struktur kode internal, alur logika program, maupun implementasi database di dalamnya. Pengujian ini dirancang untuk memverifikasi apakah keluaran (*output*) yang dihasilkan oleh sistem sepenuhnya sesuai dengan spesifikasi kebutuhan fungsional yang diharapkan (*expected results*) ketika diberikan masukan (*input*) tertentu.

Konsep pengujian fungsional telah digunakan sejak masa awal perkembangan rekayasa perangkat lunak pada dasawarsa 1960-an dan 1970-an. Namun, formalisasi dan standardisasi metode *Black Box Testing* secara komprehensif pertama kali dikonseptualisasikan oleh ilmuwan komputer terkemuka Glenford J. Myers melalui bukunya yang sangat monumental berjudul "*The Art of Software Testing*" yang diterbitkan pada tahun 1979. Myers menegaskan bahwa pengujian harus dilakukan dengan sudut pandang pengguna akhir untuk menemukan ketidaksesuaian spesifikasi, kesalahan fungsi antarmuka (*interface errors*), serta penanganan galat (*Error Handling*) yang tidak konsisten. Seiring waktu, metode ini diperluas dengan berbagai teknik sistematis seperti *Boundary Value Analysis* (BVA) dan *Equivalence Partitioning* (EP).

Meskipun secara ideal pengujian dilakukan oleh tim jaminan kualitas independen (*independent Quality Assurance*), dalam praktiknya metode *Black Box Testing* dapat dan sangat lazim diuji secara mandiri oleh pengembang sistem itu sendiri (*developer self-Testing*). Karakteristik pengujian fungsional yang tidak terikat pada kode program justru mempermudah pengembang untuk melakukan validasi cepat terhadap kebenaran logika bisnis aplikasi yang sedang mereka bangun sebelum sistem tersebut dideploy ke lingkungan produksi.

Berdasarkan literatur ilmiah dalam rekayasa perangkat lunak, aktivitas pengujian memakan porsi yang sangat signifikan dalam pengembangan sistem, yaitu mencapai 35% hingga lebih dari 50% dari total waktu, tenaga, dan biaya pengembangan. Oleh karena itu, pengujian mandiri oleh pengembang di awal siklus pengembangan (*early testing*) merupakan langkah krusial untuk mencegah penumpukan bug.

Fakta ilmiah mengenai keterlibatan pengembang dalam menguji sistem mereka sendiri didukung oleh dua rujukan utama berikut:

1. Kurniawan A. E. (2020): Menunjukkan bahwa dalam siklus pengembangan, pengembang (*programmers*) secara rutin melakukan pengujian fungsional secara internal. Penelitian ini menegaskan bahwa terdapat risiko "redundant testing" di mana pengujian fungsional yang dilakukan oleh penguji independen sebenarnya mengulangi pengujian fungsional yang "telah dilakukan sebelumnya oleh para pemrogram/pengembang itu sendiri. Hal ini membuktikan bahwa pengembang secara mandiri telah menerapkan kasus-kasus uji fungsi dasar sebelum merilisnya.
2. Tyanafisya et al. (2025): Menggambarkan bagaimana tim pengembang sistem melakukan pengujian secara mandiri untuk mengevaluasi fungsionalitas modul krusial seperti halaman login dan registrasi menggunakan teknik *Equivalence Partitioning*. Pengujian mandiri oleh pengembang ini sangat efisien dalam membagi data masukan (*input*) menjadi kelas-kelas yang ekuivalen (valid dan tidak valid) untuk mendeteksi respons keliru, pesan kesalahan (*error message*) yang tidak

informatif, maupun ketidakcocokan antara validasi front-end dengan server backend. Penemuan kecacatan secara mandiri ini memungkinkan pengembang melakukan perbaikan logika sistem secara instan demi kenyamanan pengguna akhir.

2.13 System Usability Scale (SUS)

System Usability Scale (SUS) merupakan salah satu instrumen evaluasi *usability* yang paling populer, andal, dan efisien untuk mengukur persepsi kemudahan penggunaan suatu produk atau sistem, termasuk aplikasi perangkat lunak berbasis web maupun mobile. Penggunaan SUS dalam evaluasi sistem sangat disarankan karena sifatnya yang sederhana, mudah diterapkan, serta memiliki tingkat validitas dan reliabilitas yang tinggi meskipun jumlah sampel responden yang digunakan relatif terbatas.

Metode *System Usability Scale* pertama kali dirancang dan dikembangkan oleh John Brooke pada tahun 1986 saat ia bekerja di Digital Equipment Corporation (DEC), sebuah perusahaan komputer terkemuka asal Amerika Serikat. Metode ini awalnya dibuat sebagai instrumen evaluasi cepat yang bersifat fleksibel ("*quick and dirty*" *usability Scale*) untuk menilai sistem perkantoran elektronik terintegrasi yang sedang dikembangkan pada masa itu. Brooke kemudian memublikasikannya secara luas kepada komunitas akademik dan industri pada tahun 1996 melalui bab buku berjudul "SUS: *A Quick and Dirty Usability Scale*" dalam buku *Usability Evaluation in Industry* (Brooke, 1996). Penelitian lanjutan oleh Lewis (2018) menegaskan bahwa kegunaan SUS tetap sangat relevan dan terus berkembang pesat sebagai standar industri global dalam pengembangan antarmuka pengguna.

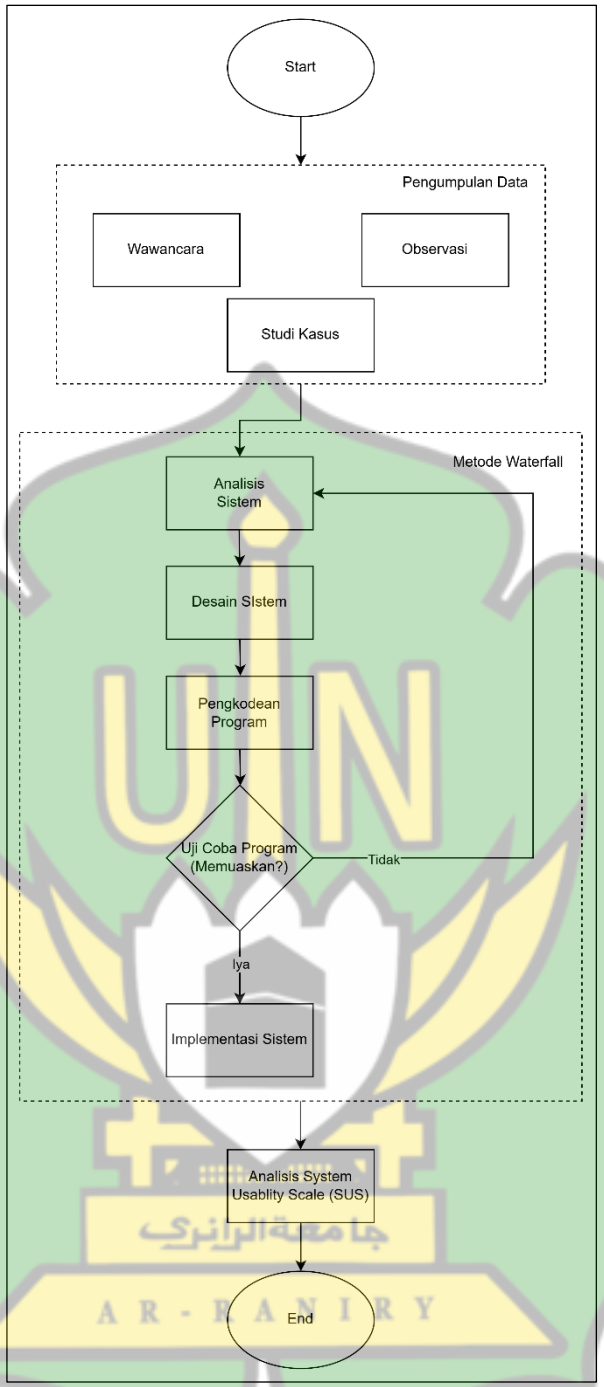
BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Penelitian ini merupakan penelitian rekayasa perangkat lunak (*software engineering research*) yang bertujuan untuk merancang dan membangun aplikasi Android Stokita sebagai sistem pengelolaan stok dan penjualan online pada Fahri Mart. Penelitian ini berfokus pada proses pengembangan aplikasi yang mampu membantu pengelolaan stok barang, pencatatan transaksi penjualan, serta penyediaan layanan penjualan online secara terintegrasi dalam satu platform berbasis Android.

Metode penelitian yang digunakan mencakup tahapan pengumpulan data, analisis kebutuhan sistem, perancangan sistem, implementasi aplikasi, serta pengujian sistem. Seluruh tahapan tersebut dilakukan secara sistematis untuk menghasilkan aplikasi yang sesuai dengan kebutuhan pengguna dan mampu menyelesaikan permasalahan yang dihadapi Fahri Mart dalam pengelolaan stok dan penjualan yang selama ini masih dilakukan secara manual.

Tahapan penelitian ini digambarkan dalam bentuk diagram alur penelitian pada Gambar 3.1. Diagram tersebut menunjukkan alur penelitian secara keseluruhan, mulai dari tahap pengumpulan data hingga tahap evaluasi aplikasi yang telah dikembangkan.



Gambar 3.1 Diagram Alur Penelitian

Berdasarkan alur penelitian pada Gambar 3.1, penelitian diawali dengan proses pengumpulan data melalui observasi dan wawancara dengan pemilik Fahri Mart untuk mengidentifikasi kondisi operasional toko, permasalahan yang dihadapi, serta kebutuhan sistem yang akan dikembangkan. Selanjutnya dilakukan analisis kebutuhan sistem yang menjadi dasar dalam perancangan aplikasi Stokita. Tahap

berikutnya adalah pengembangan aplikasi menggunakan metode Waterfall yang terdiri dari analisis kebutuhan, desain sistem, implementasi, dan pengujian.

Setelah aplikasi selesai dikembangkan, dilakukan pengujian sistem untuk memastikan seluruh fungsi berjalan sesuai kebutuhan. Selain itu, dilakukan pengujian *usability* menggunakan metode *System Usability Scale* (SUS) untuk mengukur tingkat kemudahan penggunaan aplikasi berdasarkan persepsi pengguna. Hasil pengujian tersebut digunakan sebagai dasar evaluasi untuk menilai kelayakan aplikasi Stokita sebagai solusi digital dalam pengelolaan stok dan penjualan pada Fahri Mart.

3.1.1 Pendekatan Penelitian

Pendekatan penelitian yang digunakan dalam penelitian ini adalah pendekatan kuantitatif deskriptif. Pendekatan ini digunakan pada tahap evaluasi aplikasi Stokita untuk memperoleh data yang terukur mengenai kinerja sistem, tingkat kemudahan penggunaan, serta efisiensi pengelolaan stok dan penjualan setelah aplikasi diterapkan pada Fahri Mart.

Pendekatan kuantitatif deskriptif dipilih karena penelitian ini tidak bertujuan untuk menguji hubungan antar variabel ataupun membuktikan hipotesis tertentu, melainkan untuk menggambarkan kondisi dan performa aplikasi yang telah dikembangkan berdasarkan data numerik yang diperoleh dari hasil pengujian. Melalui pendekatan ini, peneliti dapat memberikan gambaran yang objektif mengenai kelayakan aplikasi sebagai solusi digital dalam pengelolaan stok dan penjualan.

Adapun karakteristik pendekatan penelitian yang digunakan dapat dijelaskan sebagai berikut:

1. Jenis Pendekatan: Penelitian ini menggunakan pendekatan kuantitatif deskriptif yang difokuskan pada evaluasi aplikasi Stokita setelah proses pengembangan dan implementasi sistem selesai dilakukan.
2. Fokus Evaluasi: Evaluasi penelitian difokuskan pada tingkat kemudahan penggunaan (*usability*) aplikasi.

3. Sumber Data Kuantitatif: Data kuantitatif diperoleh dari hasil pengujian fungsional system dan hasil kuesioner *System Usability Scale* (SUS) yang diisi oleh pengguna aplikasi, yaitu admin toko dan pelanggan.
4. Bentuk Data dan Hasil Analisis: Data penelitian disajikan dalam bentuk skor, persentase, dan deskripsi numerik yang digunakan untuk menilai tingkat kelayakan dan kemudahan penggunaan.

Melalui pendekatan kuantitatif deskriptif ini, penelitian diharapkan mampu memberikan gambaran yang jelas dan terukur mengenai kualitas aplikasi Stokita dalam mendukung pengelolaan stok dan penjualan online pada Fahri Mart, sehingga dapat diketahui sejauh mana aplikasi yang dikembangkan mampu memenuhi kebutuhan pengguna dan meningkatkan efisiensi operasional toko.

3.1.2 Metode Pengumpulan Data

Pengumpulan data dalam penelitian ini bertujuan untuk memperoleh informasi empiris yang mendukung proses perancangan dan pengembangan aplikasi manajemen stok dan penjualan online "Stokita" pada Toko Fahri Mart. Proses ini dilakukan untuk memahami kondisi nyata operasional toko, sistem pencatatan yang berjalan, serta kebutuhan pengguna terhadap sistem baru yang akan dikembangkan.

Teknik pengumpulan data yang digunakan dalam penelitian ini meliputi observasi langsung, wawancara semi-terstruktur, Melalui kombinasi metode observasi dan wawancara, peneliti memperoleh gambaran menyeluruh mengenai alur kerja, kendala, serta kebutuhan pengguna, yang kemudian dijadikan dasar dalam proses perancangan sistem aplikasi "Stokita".

3.1.2.1 Teknik Observasi

Observasi dilakukan secara langsung di lokasi Toko Fahri Mart yang beralamat di Komplek Terminal Lama, Kecamatan Kota Juang, Kabupaten Bireuen, Aceh. Tujuan observasi ini adalah untuk memahami secara mendalam proses pengelolaan stok dan penjualan yang masih dilakukan secara manual serta mengidentifikasi permasalahan yang muncul dalam operasional harian toko.

Selama kegiatan observasi, peneliti tidak hanya mencatat aktivitas yang berlangsung, tetapi juga terlibat secara langsung dalam beberapa kegiatan

operasional toko, seperti membantu proses pencatatan penjualan, memeriksa stok barang di rak, serta mengamati proses transaksi antara penjual dan pembeli. Melalui keterlibatan tersebut, peneliti dapat memahami secara lebih komprehensif alur kerja yang diterapkan, termasuk cara pengelola menentukan barang yang perlu dibeli kembali serta kendala yang sering muncul, seperti stok habis tanpa terpantau, kesulitan dalam merekap data penjualan, dan kurangnya pencatatan stok masuk dan keluar secara sistematis.

Teknik observasi yang digunakan adalah observasi partisipatif, di mana peneliti berperan sebagai pengamat sekaligus ikut terlibat dalam aktivitas operasional dalam batas tertentu. Pendekatan ini memungkinkan peneliti memperoleh data yang lebih mendalam dan kontekstual karena memahami situasi kerja secara langsung dari sudut pandang pelaku usaha.

Hasil observasi partisipatif ini menjadi dasar dalam merumuskan kebutuhan fungsional aplikasi "Stokita", seperti fitur pencatatan stok otomatis, rekap penjualan digital, notifikasi stok menipis, serta sistem pemesanan online bagi pelanggan. Dengan demikian, aplikasi yang dirancang diharapkan benar-benar sesuai dengan kebutuhan nyata pengelola dan mampu meningkatkan efisiensi operasional Fahri Mart.

3.1.2.2 Teknik Wawancara

Teknik wawancara digunakan untuk menggali informasi yang tidak dapat diperoleh hanya melalui observasi, terutama yang berkaitan dengan persepsi, kebutuhan, dan harapan pengelola toko terhadap sistem pengelolaan stok dan penjualan digital. Wawancara dilakukan secara semi-terstruktur kepada pengelola Fahri Mart, Ibu Nilawati.

Pertanyaan wawancara disusun berdasarkan panduan tematik yang mencakup beberapa aspek utama, antara lain:

a. Kendala yang dihadapi dalam pencatatan stok dan rekap penjualan secara manual,
b. Kebutuhan fitur utama yang diharapkan dapat membantu pengelolaan stok dan transaksi,

c. Harapan terhadap aplikasi “Stokita”, seperti kemudahan dalam pemantauan stok dan pencatatan otomatis penjualan,
d. Pandangan terhadap fitur tambahan, seperti pemesanan online dan notifikasi stok habis,
e. Preferensi desain dan kemudahan penggunaan aplikasi, agar sesuai dengan kemampuan pengguna yang tidak terbiasa dengan sistem digital.

Data hasil wawancara ini digunakan sebagai bahan pertimbangan dalam perancangan sistem dan penentuan spesifikasi teknis aplikasi, sehingga aplikasi yang dikembangkan benar-benar berdasarkan kebutuhan nyata pengguna di lapangan. Dengan demikian, sistem “Stokita” diharapkan mampu memberikan solusi praktis bagi pengelola Fahri Mart dalam mengatasi kendala operasional dan meningkatkan efisiensi pengelolaan usaha.

3.1.3 Alat dan Bahan

Dalam pelaksanaan penelitian dan pengembangan aplikasi Android Stokita untuk pengelolaan stok dan penjualan online pada Fahri Mart, diperlukan perangkat keras (*hardware*) dan perangkat lunak (*software*) yang mendukung seluruh proses pengembangan sistem. Perangkat tersebut digunakan pada tahap analisis kebutuhan, perancangan sistem, implementasi, pengujian, hingga dokumentasi penelitian.

3.1.3.1 Perangkat Keras (*Hardware*)

Perangkat keras yang digunakan dalam penelitian ini berfungsi sebagai media utama untuk merancang, membangun, dan menguji aplikasi Stokita. Adapun spesifikasi perangkat keras yang digunakan dapat dilihat pada Tabel 3.1.

Tabel 3.1 Hardware

Komponen	Spesifikasi
Processor	Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz (2.81 GHz)
RAM	8,00 GB 2133 MHz RAM
Storage	238 GB
Graphic Card	Intel(R) HD Graphics 520

3.1.3.2 Perangkat Lunak (*Software*)

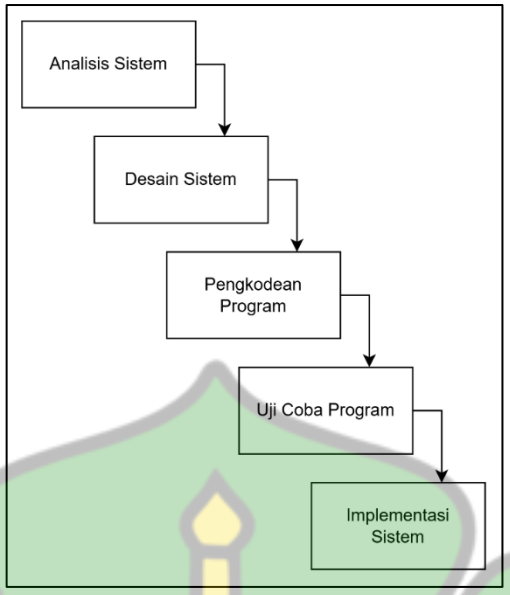
Perangkat lunak yang digunakan berfungsi sebagai alat bantu dalam proses pengembangan aplikasi, pengelolaan basis data, perancangan sistem, pengujian API, serta penyusunan dokumentasi penelitian. Daftar perangkat lunak yang digunakan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Software

Perangkat Lunak	Versi
Visual Studio Code	1.109.5
Android Studio	2024.2
React Native	0.83.1
Node.js	24.13.0
MySQL	8.0.45
Git	2.47.1
Postman	*Lastest Version*
Figma	*Lastest Version*
Draw.io	*Lastest Version*
Google Chrome	*Lastest Version*

3.1.4 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini mengacu pada model Waterfall. Model ini dipilih karena memiliki alur kerja yang terstruktur dan sistematis, di mana setiap tahap pengembangan dilakukan secara berurutan mulai dari analisis kebutuhan hingga pengujian.



Gambar 3.2 Metode Waterfall

Pendekatan ini cocok diterapkan dalam pengembangan aplikasi "Stokita", karena kebutuhan sistem sudah dapat didefinisikan dengan jelas sejak awal berdasarkan hasil observasi dan wawancara di Fahri Mart.

3.1.4.1 Analisis Kebutuhan

Tahap ini bertujuan untuk mengidentifikasi kebutuhan fungsional dan non-fungsional dari aplikasi Stokita. Analisis dilakukan melalui hasil observasi langsung di Fahri Mart dan wawancara semi-terstruktur dengan pengelola toko.

Dari hasil observasi, ditemukan bahwa pengelolaan stok masih dilakukan secara manual tanpa pencatatan digital. Barang yang habis baru diketahui saat pengelola melihat langsung ke rak toko, dan proses rekap penjualan masih ditulis di buku tulis. Selain itu, tidak ada sistem yang membantu pengingat stok menipis atau laporan penjualan otomatis.

Hasil wawancara juga menunjukkan bahwa pengelola menginginkan aplikasi yang dapat mempermudah pemantauan stok barang, melakukan pencatatan penjualan otomatis, menyediakan fitur pemesanan online bagi pelanggan sekitar, memberikan notifikasi stok habis, dan menghasilkan laporan penjualan harian dan bulanan.

Informasi tersebut kemudian dijadikan dasar untuk merumuskan kebutuhan sistem yang akan diterapkan pada aplikasi Stokita, baik dari sisi fungsionalitas maupun kemudahan penggunaannya.

3.1.4.2 Desain Sistem

Tahap desain sistem merupakan proses menerjemahkan kebutuhan sistem yang telah diperoleh pada tahap analisis ke dalam bentuk rancangan konseptual sebagai dasar pengembangan aplikasi Stokita. Pada tahap ini dilakukan perancangan gambaran umum sistem yang mencakup struktur sistem, interaksi pengguna dengan sistem, serta hubungan antar komponen yang terlibat dalam proses pengelolaan stok dan penjualan online pada Fahri Mart.

Desain sistem bertujuan untuk memberikan pemahaman mengenai bagaimana aplikasi akan bekerja dalam memenuhi kebutuhan pengguna. Melalui tahap ini, ditentukan fungsi-fungsi utama yang akan tersedia pada aplikasi, seperti pengelolaan data produk, pencatatan stok masuk dan stok keluar, transaksi penjualan, pengelolaan pesanan online, notifikasi stok menipis, serta penyajian laporan penjualan dan keuntungan.

Perancangan teknis secara detail, seperti *Use Case Diagram*, *Entity Relationship Diagram* (ERD), *Activity Diagram*, *User Flow*, struktur basis data, serta rancangan antarmuka pengguna, tidak dijelaskan pada tahap ini karena akan dibahas secara lengkap pada subbab Perancangan Sistem. Oleh karena itu, tahap desain sistem berfungsi sebagai gambaran umum mengenai arsitektur dan mekanisme kerja aplikasi Stokita sebelum memasuki tahap implementasi sistem.

3.1.4.3 Implementasi

Tahap implementasi dilakukan dengan menerjemahkan desain sistem ke dalam bentuk kode program. Aplikasi Stokita dibangun menggunakan React Native sebagai *framework* utama untuk pengembangan aplikasi Android, sedangkan Node.js digunakan untuk membangun *backend* API yang menghubungkan aplikasi dengan basis data MySQL.

Implementasi dilakukan secara bertahap mulai dari pembuatan struktur database, pengembangan API, hingga integrasi antar komponen sistem. *Version*

control menggunakan Git diterapkan untuk mencatat setiap perubahan kode dan menjaga konsistensi proyek selama proses pengembangan.

3.1.4.4 Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa aplikasi Stokita yang dikembangkan telah berfungsi sesuai dengan kebutuhan pengguna dan tujuan penelitian. Tahap ini bertujuan untuk mengevaluasi kinerja aplikasi dari aspek fungsionalitas, kemudahan penggunaan, serta efisiensi dalam mendukung pengelolaan stok dan penjualan pada Fahri Mart.

Pengujian fungsional dilakukan untuk memverifikasi bahwa setiap fitur utama aplikasi berjalan sesuai dengan kebutuhan yang telah ditentukan pada tahap analisis. Pengujian difokuskan pada fungsi-fungsi utama seperti proses autentikasi pengguna, pengelolaan data produk, pencatatan stok masuk dan stok keluar, transaksi penjualan offline dan online, pengelolaan pesanan, notifikasi stok menipis, serta penyajian laporan penjualan dan keuntungan. Melalui pengujian ini dapat diketahui apakah sistem mampu menghasilkan keluaran yang sesuai dengan masukan yang diberikan oleh pengguna.

Selain pengujian fungsional, dilakukan juga pengujian *usability* menggunakan instrumen *System Usability Scale* (SUS) untuk mengukur tingkat kemudahan penggunaan aplikasi berdasarkan persepsi pengguna. Pengujian ini bertujuan untuk mengetahui tingkat kemudahan dipelajari (*learnability*), efisiensi penggunaan, kenyamanan, serta kepuasan pengguna dalam berinteraksi dengan aplikasi Stokita.

Hasil dari pengujian tersebut digunakan sebagai dasar evaluasi untuk menilai kelayakan aplikasi Stokita sebagai sistem pengelolaan stok dan penjualan online. Apabila seluruh fungsi sistem berjalan dengan baik dan memperoleh tingkat *usability* yang memadai. aplikasi dinyatakan layak untuk digunakan dalam mendukung operasional Fahri Mart.

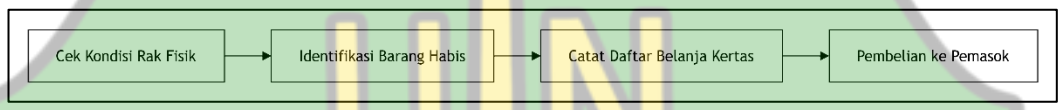
3.1.5 Analisis Proses Bisnis Fahri Mart

Analisis proses bisnis dilakukan untuk memetakan alur operasional manual yang diterapkan di Fahri Mart sebelum adanya pengembangan aplikasi Stokita. Berdasarkan hasil observasi partisipatif dan wawancara, saat ini seluruh kegiatan

operasional masih bersifat konvensional dan sangat bergantung pada ingatan serta pencatatan fisik di media kertas.

3.1.5.1 Alur Pengelolaan Stok Barang

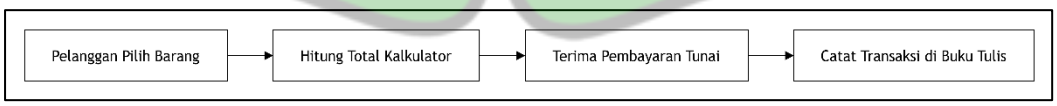
Proses manajemen persediaan di Fahri Mart dilakukan tanpa adanya dokumentasi stok masuk dan keluar yang terstruktur. Pemilik melakukan pemantauan stok dengan cara mengecek kondisi fisik barang di rak secara visual setiap harinya. Apabila terlihat barang mulai habis atau menipis, pemilik baru mencatat kebutuhan tersebut pada secarik kertas untuk dibawa saat berbelanja ke pemasok. Hal ini sering memicu keterlambatan dalam mengetahui stok yang habis serta risiko kesalahan dalam pengadaan barang. Alur proses pengelolaan stok barang pada Fahri Mart digambarkan pada Gambar 3.3.



Gambar 3.3 Blok Diagram Alur Pengelolaan Stok Barang

3.1.5.2 Alur Penjualan Offline

Transaksi penjualan saat ini dilakukan dengan melayani pelanggan yang datang langsung ke lokasi toko. Proses perhitungan total belanja pelanggan dilakukan oleh pengelola secara manual atau dengan bantuan alat kalkulator. Setelah pembayaran diterima secara tunai, setiap transaksi tersebut kemudian dituliskan kembali ke dalam buku besar laporan harian. Penggunaan buku tulis sebagai media utama laporan berisiko tinggi terhadap kehilangan data, kesalahan penulisan nilai, serta menyulitkan proses rekapitulasi laba-rugi setiap bulannya. Alur proses penjualan offline pada Fahri Mart digambarkan pada Gambar 3.4.

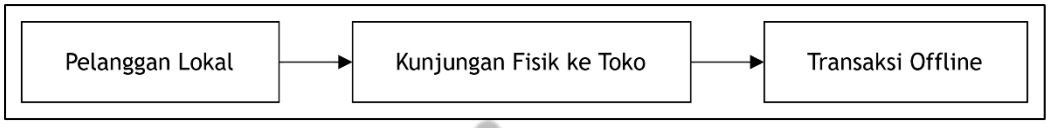


Gambar 3.4 Blok Diagram Alur Penjualan Offline

3.1.5.3 Ketiadaan Layanan Penjualan Online

Saat ini operasional Fahri Mart sepenuhnya mengandalkan transaksi fisik di toko. Ketiadaan saluran penjualan berbasis digital menyebabkan jangkauan pasar

terbatas hanya pada masyarakat di sekitar Komplek Terminal Lama, Bireuen, yang dapat berkunjung langsung ke lokasi. Hal ini menyebabkan potensi transaksi dari pelanggan luar daerah atau pelanggan yang menginginkan kepraktisan pemesanan digital belum dapat terakomodasi.



Gambar 3.5 Blok Diagram Keterbatasan Jangkauan Pasar

3.1.6 Perancangan Algoritma Sistem

Perancangan algoritma sistem dilakukan untuk menggambarkan alur logika proses yang berjalan pada aplikasi Stokita dalam mendukung pengelolaan stok dan penjualan online pada Fahri Mart. Tahap ini bertujuan untuk menerjemahkan kebutuhan sistem yang telah dianalisis sebelumnya ke dalam bentuk algoritma yang dapat digunakan sebagai acuan dalam proses implementasi aplikasi.

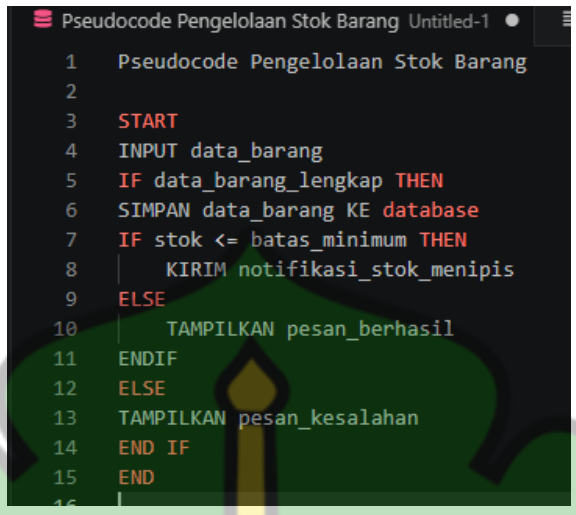
Perancangan algoritma sistem disajikan dalam bentuk *pseudocode* untuk mempermudah pemahaman terhadap logika proses yang diterapkan pada setiap fitur utama aplikasi. *Pseudocode* digunakan karena mampu menggambarkan urutan proses secara sistematis tanpa terikat pada sintaks bahasa pemrograman tertentu, sehingga lebih mudah dipahami baik oleh pengembang maupun pembaca penelitian.

Berdasarkan proses bisnis yang telah dianalisis pada subbab sebelumnya, perancangan sistem pada aplikasi Stokita difokuskan pada tiga proses utama, yaitu pengelolaan stok barang, penjualan offline, dan penjualan online. Ketiga proses tersebut merupakan fitur inti yang mendukung operasional Fahri Mart dalam melakukan pengelolaan persediaan barang dan transaksi penjualan secara terintegrasi.

3.1.6.1 Pseudocode Pengelolaan Stok Barang

Alur logika pengelolaan stok barang pada aplikasi Stokita direpresentasikan dalam bentuk *pseudocode* yang ditampilkan pada Gambar 3.6. *Pseudocode* ini digunakan untuk menggambarkan proses sistem dalam mencatat data barang,

memperbarui jumlah stok, serta memberikan notifikasi ketika stok barang mencapai batas minimum yang telah ditentukan.



Gambar 3.6 Pseudocode Pengelolaan Stok Barang

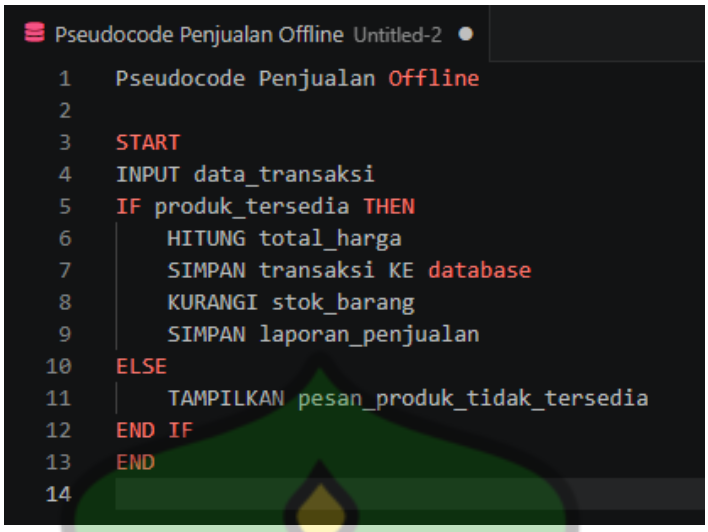
Berdasarkan Gambar 3.6, proses pengelolaan stok diawali dengan *input* data barang oleh admin melalui pengisian data produk atau pemindaian barcode. Sistem kemudian melakukan validasi terhadap data yang dimasukkan. Apabila data valid, sistem menyimpan informasi barang ke dalam basis data dan memperbarui jumlah stok yang tersedia.

Setelah proses penyimpanan berhasil dilakukan, sistem memeriksa jumlah stok barang yang tersisa. Jika jumlah stok berada di bawah batas minimum yang telah ditentukan, sistem akan mengirimkan notifikasi stok menipis kepada admin. Sebaliknya, apabila stok masih mencukupi, data akan langsung disimpan dan ditampilkan pada daftar produk.

Dengan mekanisme tersebut, admin dapat memantau ketersediaan stok barang secara *real-time*, melakukan penambahan stok ketika diperlukan, serta mengurangi risiko terjadinya kehabisan barang yang dapat mengganggu proses penjualan.

3.1.6.2 Pseudocode Penjualan Offline

Alur logika penjualan offline pada aplikasi Stokita direpresentasikan dalam bentuk *pseudocode* yang ditampilkan pada Gambar 3.7. *Pseudocode* ini menggambarkan proses transaksi penjualan yang dilakukan secara langsung di toko melalui fitur kasir yang tersedia pada aplikasi.



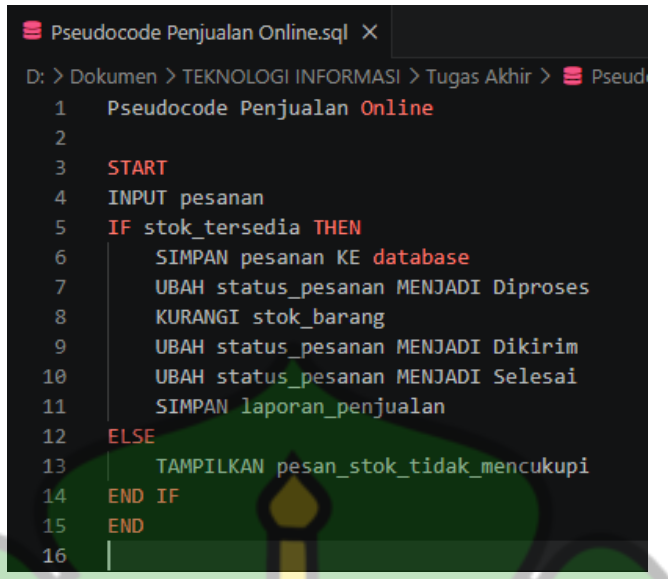
Gambar 3.7 Pseudocode Penjualan Offline

Berdasarkan Gambar 3.7, proses penjualan offline dimulai ketika admin memasukkan produk yang dibeli pelanggan melalui pemindaian barcode atau pemilihan produk dari daftar yang tersedia. Sistem kemudian menampilkan rincian transaksi beserta total harga yang harus dibayar pelanggan.

Setelah pembayaran diterima, admin melakukan konfirmasi transaksi dan sistem menyimpan data penjualan ke dalam basis data. Selanjutnya, sistem secara otomatis mengurangi jumlah stok barang sesuai dengan produk yang terjual dan mencatat transaksi ke dalam laporan penjualan. Apabila stok barang yang tersisa berada di bawah batas minimum, sistem akan mengirimkan notifikasi kepada admin sebagai pengingat untuk melakukan pengadaan ulang.

3.1.6.3 Pseudocode Penjualan Online

Alur logika penjualan online pada aplikasi Stokita direpresentasikan dalam bentuk *pseudocode* yang ditampilkan pada Gambar 3.8. *Pseudocode* ini digunakan untuk menggambarkan proses pemesanan produk oleh pelanggan melalui aplikasi hingga transaksi selesai diproses.



Gambar 3.8 Pseudocode Penjualan Online

Berdasarkan Gambar 3.8, proses penjualan online diawali ketika pelanggan memilih produk yang tersedia pada aplikasi dan menambahkan produk tersebut ke dalam keranjang belanja. Sistem memastikan jumlah pesanan tidak melebihi stok yang tersedia sehingga pelanggan hanya dapat memesan sesuai jumlah stok yang ada.

Setelah pelanggan melakukan checkout, sistem menyimpan data pesanan, lalu sistem secara otomatis mengubah status pesanan menjadi Diproses dan mengurangi jumlah stok. Admin kemudian menyiapkan pesanan sesuai dengan data yang diterima. Setelah pesanan dikirim, admin mengubah status pesanan menjadi Dikirim sehingga pelanggan dapat memantau perkembangan pesanan melalui aplikasi.

Ketika pesanan telah diterima pelanggan, admin mengubah status pesanan menjadi Selesai, lalu sistem mencatat transaksi ke dalam laporan penjualan. Dengan mekanisme tersebut, data stok dan transaksi penjualan selalu terintegrasi serta diperbarui secara *real-time*.

3.1.7 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk mengidentifikasi fungsi dan karakteristik yang harus dimiliki oleh aplikasi Stokita agar mampu mendukung

proses pengelolaan stok dan penjualan online pada Fahri Mart. Analisis ini disusun berdasarkan hasil observasi dan wawancara yang dilakukan dengan pemilik toko, serta mempertimbangkan kebutuhan operasional dalam mengelola data produk, stok barang, transaksi penjualan, dan pesanan pelanggan secara lebih efektif dan efisien.

Kebutuhan sistem menjadi dasar dalam proses perancangan dan pengembangan aplikasi sehingga sistem yang dihasilkan dapat sesuai dengan kebutuhan pengguna. Secara umum, kebutuhan sistem dibagi menjadi dua kategori, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

3.1.7.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fitur dan layanan yang harus disediakan oleh aplikasi Stokita agar dapat menjalankan seluruh proses bisnis yang telah dirancang. Adapun kebutuhan fungsional sistem adalah sebagai berikut:

1. **Manajemen Pengguna**: Sistem harus menyediakan fitur registrasi, login, dan logout untuk pengguna sesuai dengan peran masing-masing, yaitu admin dan pelanggan.
2. **Pengelolaan Data Produk**: Sistem harus menyediakan fitur untuk menambah, mengubah, menghapus, dan menampilkan data produk yang dijual pada Fahri Mart.
3. **Pengelolaan Kategori Produk**: Sistem harus menyediakan fitur untuk mengelola kategori produk sehingga produk dapat dikelompokkan dengan lebih terstruktur.
4. **Pengelolaan Stok Barang**: Sistem harus mampu mencatat stok barang masuk, memperbarui jumlah stok, serta menampilkan informasi stok secara *real-time*.
5. **Pemindaian Barcode**: Sistem harus menyediakan fitur pemindaian barcode untuk mempercepat proses pencarian dan input produk.
6. **Transaksi Penjualan Offline**: Sistem harus mendukung proses transaksi penjualan langsung di toko, mulai dari pemilihan produk hingga penyimpanan data transaksi.

7. **Penjualan Online**: Sistem harus memungkinkan pelanggan melakukan pemesanan produk melalui aplikasi tanpa harus datang langsung ke toko.
8. **Keranjang Belanja**: Sistem harus menyediakan fitur keranjang belanja untuk menampung produk yang dipilih pelanggan sebelum melakukan checkout.
9. **Pengelolaan Pesanan**: Sistem harus mampu mencatat dan mengelola status pesanan yang terdiri dari status Diproses, Dikirim, dan Selesai.
10. **Laporan Penjualan**: Sistem harus menyediakan informasi laporan penjualan yang dapat digunakan admin untuk memantau aktivitas transaksi dan performa penjualan.
11. **Notifikasi Stok Menipis**: Sistem harus memberikan notifikasi kepada admin apabila jumlah stok barang mencapai batas minimum yang telah ditentukan.
12. **Pencarian Produk**: Sistem harus menyediakan fitur pencarian produk untuk mempermudah pengguna menemukan barang yang diinginkan.

Kebutuhan fungsional tersebut memastikan bahwa aplikasi Stokita dapat mendukung seluruh aktivitas utama pengelolaan stok dan penjualan pada Fahri Mart secara terintegrasi.

3.1.7.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional berkaitan dengan kualitas sistem, performa, keamanan, serta kemudahan penggunaan aplikasi. Kebutuhan non-fungsional yang ditetapkan dalam penelitian ini adalah sebagai berikut:

1. **Usability**: Sistem harus memiliki antarmuka yang sederhana, mudah dipahami, dan mudah digunakan oleh admin maupun pelanggan.
2. **Keamanan Data**: Sistem harus menyediakan mekanisme autentikasi login untuk melindungi data pengguna dan data transaksi dari akses yang tidak sah.
3. **Reliability**: Sistem harus mampu beroperasi dengan stabil dan menjalankan fungsi-fungsi utama tanpa mengalami gangguan yang signifikan.

4. **Performance**: Sistem harus mampu memproses data produk, stok, dan transaksi dengan waktu respons yang cepat.
5. **Portabilitas**: Aplikasi harus dapat dijalankan pada perangkat Android yang umum digunakan oleh pengguna.
6. **Konsistensi Antarmuka**: Sistem harus memiliki tampilan yang konsisten pada setiap halaman sehingga memberikan pengalaman penggunaan yang lebih baik.
7. **Maintainability**: Sistem harus dirancang dengan struktur yang mudah dipelihara dan dikembangkan untuk penambahan fitur di masa mendatang.
8. **Ketersediaan Data**: Sistem harus mampu menyimpan dan menampilkan data produk, stok, transaksi, dan pesanan secara akurat serta terintegrasi dengan basis data MySQL.

Kebutuhan non-fungsional tersebut bertujuan untuk memastikan bahwa aplikasi Stokita tidak hanya berfungsi sesuai kebutuhan pengguna, tetapi juga memiliki kualitas sistem yang baik, stabil, aman, dan nyaman digunakan dalam kegiatan operasional Fahri Mart.

3.1.8 Perancangan Sistem

Perancangan sistem merupakan tahap untuk menerjemahkan kebutuhan sistem yang telah dianalisis ke dalam bentuk rancangan teknis yang akan digunakan pada proses pengembangan aplikasi Stokita. Perancangan ini bertujuan untuk memberikan gambaran mengenai struktur sistem, interaksi pengguna dengan sistem, serta hubungan antar komponen yang terdapat dalam aplikasi.

Pada penelitian ini, perancangan sistem meliputi perancangan *Use Case Diagram*, perancangan basis data, perancangan alur sistem, dan perancangan antarmuka pengguna. Hasil perancangan tersebut menjadi acuan dalam proses implementasi aplikasi sehingga fitur yang dikembangkan sesuai dengan kebutuhan operasional Fahri Mart.

Melalui perancangan sistem yang terstruktur, diharapkan aplikasi Stokita mampu mendukung proses pengelolaan stok dan penjualan online secara efektif, efisien, dan mudah digunakan oleh pengguna.

3.1.8.1 Use Case Diagram

Perancangan *Use Case Diagram* bertujuan untuk memvisualisasikan interaksi antara pengguna (aktor) dengan aplikasi Stokita. Diagram ini digunakan untuk menggambarkan fungsi-fungsi utama sistem serta hak akses yang dimiliki oleh masing-masing aktor dalam menjalankan aktivitas pada aplikasi.

Gambar 3.9 Use Case Diagram

Berdasarkan rancangan pada Gambar 3.9, terdapat dua aktor utama yang terlibat dalam sistem, yaitu Admin dan Pelanggan. Kedua aktor tersebut memiliki peran yang berbeda sesuai dengan kebutuhan operasional aplikasi.

1. **Admin:** Admin merupakan aktor yang memiliki hak akses untuk mengelola seluruh data yang terdapat pada aplikasi Stokita. Admin bertanggung jawab terhadap pengelolaan produk, pengelolaan stok barang, transaksi penjualan, serta pengelolaan pesanan pelanggan. Adapun aktivitas yang dapat dilakukan oleh admin meliputi:
   a. Login ke dalam sistem.
   b. Mengelola data produk.

c. Mengelola kategori produk.
d. Mengelola stok barang.
e. Melakukan transaksi penjualan offline.
f. Memindai barcode produk.
g. Mengelola pesanan online.
h. Mengubah status pesanan (Diproses, Dikirim, dan Selesai).
i. Melihat laporan penjualan.
j. Logout dari sistem.

2. **Pelanggan:** Pelanggan merupakan pengguna aplikasi yang memanfaatkan fitur penjualan online untuk melakukan pembelian produk. Pelanggan memiliki akses terhadap fitur-fitur yang berhubungan dengan proses pemesanan barang. Adapun aktivitas yang dapat dilakukan oleh pelanggan meliputi:
   a. Registrasi akun.
   b. Login ke dalam sistem.
   c. Melihat daftar produk.
   d. Mencari produk.
   e. Menambahkan produk ke keranjang belanja.
   f. Melakukan checkout pesanan.
   g. Melihat status pesanan.
   h. Mengelola profil akun.
   i. Logout dari sistem.

Hubungan Antar Use Case (*Include Relationship*)

Pada *Use Case Diagram* terdapat beberapa hubungan include yang menggambarkan keterkaitan antar proses dalam sistem, yaitu:

1) **Kelola Stok Barang *include* Kelola Produk:** Hubungan ini menunjukkan bahwa proses pengelolaan stok tidak dapat dilakukan tanpa adanya data produk yang telah tersimpan pada sistem.
2) **Penjualan Offline *include* Kelola Stok Barang:** Hubungan ini menunjukkan bahwa setiap transaksi penjualan offline akan memengaruhi jumlah stok barang yang tersedia pada sistem.

3) **Checkout Pesanan *include* Keranjang Belanja:** Hubungan ini menunjukkan bahwa proses checkout hanya dapat dilakukan apabila pelanggan telah menambahkan produk ke dalam keranjang belanja.
4) **Checkout Pesanan *include* Kelola Alamat:** Hubungan ini menunjukkan bahwa pelanggan harus memilih salah satu alamat pengiriman yang telah tersimpan sebelum proses checkout dapat dilakukan. Informasi alamat yang dipilih kemudian disalin ke data pesanan sebagai alamat pengiriman.
5) **Kelola Pesanan *include* Ubah Status Pesanan:** Hubungan ini menunjukkan bahwa pengelolaan pesanan oleh admin melibatkan proses perubahan status pesanan sesuai tahapan pengiriman.
6) **Laporan Penjualan *include* Penjualan Offline dan Penjualan Online:** Hubungan ini menunjukkan bahwa data laporan penjualan diperoleh dari seluruh transaksi yang terjadi baik melalui penjualan offline maupun penjualan online.

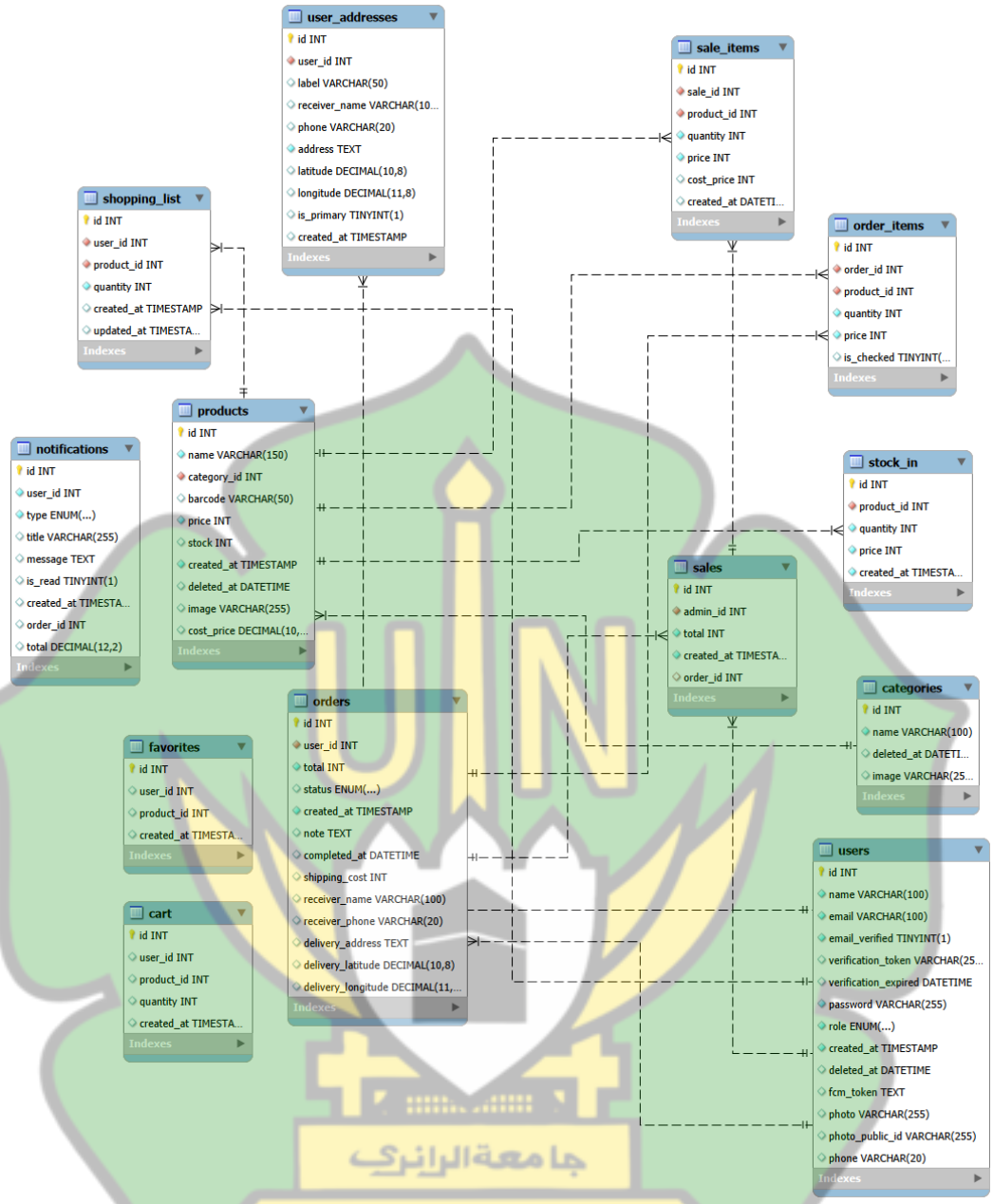
3.1.8.2 Perancangan Basis Data

Perancangan basis data dilakukan untuk mendefinisikan struktur penyimpanan data yang digunakan dalam aplikasi Stokita. Basis data dirancang menggunakan sistem manajemen basis data MySQL dengan tujuan untuk menyimpan, mengelola, dan mengintegrasikan seluruh data yang berkaitan dengan pengelolaan stok dan penjualan pada Fahri Mart.

Perancangan basis data dilakukan menggunakan *Entity Relationship Diagram* (ERD) untuk menggambarkan hubungan antar entitas yang terdapat dalam sistem. Struktur basis data dirancang agar mampu mendukung seluruh kebutuhan fungsional aplikasi, mulai dari pengelolaan pengguna, produk, stok barang, transaksi penjualan, hingga pengelolaan pesanan pelanggan.

a) Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) digunakan untuk menggambarkan struktur logis basis data dan hubungan antar entitas pada aplikasi Stokita. Dalam sistem ini, tabel users berfungsi sebagai entitas utama yang terhubung dengan tabel transaksi dan pengelolaan data lainnya.



Gambar 3.10 ERD Basis Data

Berdasarkan rancangan ERD pada Gambar 3.10, aplikasi Stokita terdiri atas beberapa entitas yang saling berhubungan untuk mendukung proses autentikasi pengguna, pengelolaan produk, manajemen stok, transaksi penjualan offline dan online, pengelolaan alamat pelanggan, notifikasi, serta fitur pendukung lainnya. Hubungan antar entitas dirancang menggunakan *foreign key* sehingga integritas data tetap terjaga. terdapat beberapa entitas utama yang digunakan dalam sistem, yaitu:

1. Users: Entitas users digunakan untuk menyimpan data pengguna aplikasi yang terdiri dari admin dan pelanggan. Setiap pengguna memiliki informasi akun yang digunakan untuk proses autentikasi dan otorisasi sistem.
2. User Addresses: Entitas user addresses digunakan untuk menyimpan beberapa alamat milik pelanggan. Sehingga satu pelanggan dapat memiliki lebih dari satu alamat. Alamat yang dipilih saat checkout akan digunakan sebagai alamat pengiriman.
3. Categories: Entitas categories digunakan untuk menyimpan data kategori produk sehingga produk dapat dikelompokkan berdasarkan jenisnya.
4. Products: Entitas products digunakan untuk menyimpan informasi produk yang dijual pada Fahri Mart, seperti nama produk, harga, stok, barcode, dan kategori produk.
5. Cart: Entitas cart digunakan untuk menyimpan daftar produk yang dimasukkan pelanggan ke dalam keranjang belanja sebelum melakukan proses checkout. Setiap data pada tabel ini merepresentasikan produk yang dipilih beserta jumlah pembelian sehingga pelanggan dapat meninjau kembali pesanan sebelum melakukan transaksi.
6. Favorites: Entitas favorites digunakan untuk menyimpan daftar produk favorit pengguna. Bagi pelanggan, fitur ini memudahkan dalam menemukan kembali produk yang sering dibeli sehingga produk dapat ditambahkan ke keranjang belanja dengan lebih cepat. Sementara itu, bagi admin, fitur ini membantu dalam mengakses produk-produk yang sering dikelola sehingga proses penambahan stok menjadi lebih mudah dan efisien.
7. Orders: Entitas orders digunakan untuk menyimpan data pesanan online yang dilakukan oleh pelanggan. Setiap pesanan memiliki informasi status pesanan yang terdiri dari Diproses, Dikirim, dan Selesai. Pada tabel ini juga disimpan snapshot alamat pengiriman berupa: receiver_name, receiver_phone, delivery_address, delivery_latitude, dan delivery_longitude. sehingga perubahan alamat pelanggan di kemudian hari tidak mempengaruhi riwayat pesanan.

8. Order Items: Entitas order_items digunakan untuk menyimpan rincian produk yang terdapat pada setiap pesanan pelanggan.
9. Sales: Entitas sales digunakan untuk menyimpan data transaksi penjualan yang dilakukan melalui fitur kasir atau penjualan offline.
10. Sale Items: Entitas sale_items digunakan untuk menyimpan rincian produk yang terdapat dalam setiap transaksi penjualan.
11. Stock In: Entitas stock_in digunakan untuk mencatat setiap penambahan stok barang yang dilakukan oleh admin. Data yang disimpan meliputi produk yang ditambahkan, jumlah stok yang masuk, harga pembelian, serta waktu pencatatan. Entitas ini berfungsi sebagai riwayat penambahan stok sekaligus mendukung pengelolaan persediaan barang.
12. Notifications: Entitas notifications digunakan untuk menyimpan data notifikasi yang dikirimkan kepada pengguna aplikasi. Notifikasi dapat berupa informasi pesanan baru, perubahan status pesanan, maupun informasi transaksi lainnya sehingga pengguna dapat memperoleh informasi secara cepat melalui aplikasi.
13. Shopping List: Entitas shopping_list digunakan untuk menyimpan daftar belanja yang dibuat oleh pelanggan sebelum melakukan pembelian. Melalui fitur ini pelanggan dapat menyusun daftar produk yang ingin dibeli terlebih dahulu, kemudian menambahkan produk tersebut ke dalam keranjang belanja ketika akan melakukan transaksi.

Hubungan antar entitas pada sistem dapat dijelaskan sebagai berikut:

1. Users – User Addresses (One-to-Many): Satu pengguna dapat memiliki lebih dari satu alamat pengiriman, sedangkan setiap alamat hanya dimiliki oleh satu pengguna.
2. Users – Orders (One-to-Many): Satu pelanggan dapat memiliki banyak pesanan, sedangkan satu pesanan hanya dimiliki oleh satu pelanggan.
3. Users – Cart (One-to-Many): Satu pengguna dapat memiliki banyak data keranjang belanja, sedangkan setiap data keranjang hanya dimiliki oleh satu pengguna.

4. Users – Favorites (One-to-Many): Satu pengguna dapat memiliki banyak produk favorit, sedangkan setiap data favorit hanya dimiliki oleh satu pengguna.
5. Users – Notifications (One-to-Many): Satu pengguna dapat menerima banyak notifikasi, sedangkan setiap notifikasi hanya dikirimkan kepada satu pengguna.
6. Users – Sales (One-to-Many): Satu admin dapat melakukan banyak transaksi penjualan offline, sedangkan setiap transaksi penjualan hanya dicatat oleh satu admin.
7. Categories – Products (One-to-Many): Satu kategori dapat memiliki banyak produk, sedangkan satu produk hanya dapat berada pada satu kategori.
8. Products – Cart (One-to-Many): Satu produk dapat tersimpan pada banyak keranjang belanja milik pengguna yang berbeda.
9. Products – Favorites (One-to-Many): Satu produk dapat ditandai sebagai produk favorit oleh banyak pengguna.
10. Products – Stock In (One-to-Many): Satu produk dapat memiliki banyak riwayat penambahan stok yang tercatat pada tabel stock_in.
11. Products – Order Items (One-to-Many): Satu produk dapat muncul pada banyak detail pesanan pelanggan.
12. Products – Sale Items (One-to-Many): Satu produk dapat tercatat pada banyak transaksi penjualan.
13. Orders – Order Items (One-to-Many): Satu pesanan dapat terdiri dari banyak produk yang tersimpan pada tabel order_items.
14. Orders – Notifications (One-to-Many): Satu pesanan dapat menghasilkan beberapa notifikasi, seperti notifikasi pesanan baru maupun perubahan status pesanan.
15. Orders – Sales (One-to-One): Satu pesanan online yang telah diselesaikan akan menghasilkan satu data transaksi penjualan pada tabel sales, sedangkan setiap transaksi penjualan yang berasal dari pesanan online hanya mengacu pada satu pesanan.

16. Sales – Sale Items (One-to-Many): Satu transaksi penjualan dapat memiliki banyak item produk yang dijual.

Hubungan tersebut memungkinkan seluruh data produk, stok, transaksi, dan pesanan dapat terintegrasi dengan baik sehingga mendukung operasional Fahri Mart secara efektif.

b) Spesifikasi Basis Data

Berikut merupakan spesifikasi tabel yang digunakan dalam basis data aplikasi Stokita. Tabel-tabel tersebut dirancang untuk mendukung proses pengelolaan stok barang, transaksi penjualan, serta pemesanan online pada Fahri Mart.

1. Tabel Users: Tabel users digunakan untuk menyimpan data akun pengguna yang terdiri dari admin dan pelanggan.

Tabel 3.3 Tabel Users

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas unik pengguna
name	VARCHAR(100)	Nama pengguna
email	VARCHAR(100)	Email pengguna
email_verified	TINYINT(1)	Verifikasi email
verification_token	VARCHAR(255)	Token untuk verifikasi email
verification_expired	DATETIME	Waktu kedaluwarsa verifikasi email
password	VARCHAR(255)	Password yang telah di-hash menggunakan bcrypt
fcm_token	TEXT	Token saat login
role	ENUM	Peran pengguna(admin/pelanggan)
phone	VARCHAR(20)	Nomor telepon
photo	VARCHAR(255)	Foto profil
photo_public_id	VARCHAR(255)	Public ID foto profil pada Cloudinary
created_at	TIMESTAMP	Waktu pembuatan akun
deleted_at	DATETIME	Waktu penghapusan akun

2. Tabel User Addresses: Tabel user addresses digunakan untuk menyimpan alamat pengiriman pelanggan.

Tabel 3.4 Tabel User Addresses

Nama Kolom	Tipe Data	Keterangan
id	INT(PK)	Identitas unik alamat pengguna
user_id	INT(FK)	Identitas pengguna
label	VARCHAR(50)	Label alamat
receiver_name	VARCHAR(100)	Nama penerima
phone	VARCHAR(20)	Nomor telepon penerima
address	TEXT	Alamat pengiriman
latitude	DECIMAL(10,8)	Latitude pengguna
longitude	DECIMAL(11,8)	Longitude pengguna
is_primary	TINYINT(1)	Alamat utama
created_at	TIMESTAMP	Waktu dibuat

3. Tabel Categories: Tabel categories digunakan untuk menyimpan data kategori produk.

Tabel 3.5 Tabel Categories

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas kategori
name	VARCHAR(100)	Nama kategori
image	VARCHAR(255)	Gambar kategori
deleted_at	DATETIME	Penanda data dihapus

4. Tabel Products: Tabel products digunakan untuk menyimpan informasi produk yang dijual pada Fahri Mart.

Tabel 3.6 Tabel Products

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas produk
name	VARCHAR(150)	Nama produk
category_id	INT (FK)	Relasi ke tabel categories
barcode	VARCHAR(50)	Kode barcode produk
price	INT	Harga jual produk
stock	INT	Jumlah stok produk
image	VARCHAR(255)	Gambar produk
cost_price	DECIMAL(10,2)	Harga modal produk

5. Tabel Stock_In: Tabel stock_in digunakan untuk mencatat penambahan stok barang.

Tabel 3.7 Tabel Stock_In

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas data stok masuk
product_id	INT (FK)	Produk yang ditambahkan
quantity	INT	Jumlah stok masuk
price	INT	Harga pembelian
created_at	TIMESTAMP	Tanggal penambahan stok

6. Tabel Orders: Tabel orders digunakan untuk menyimpan data pesanan online pelanggan.

Tabel 3.8 Tabel Orders

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas pesanan
user_id	INT (FK)	Pelanggan pemesan
total	INT	Total nilai pesanan
status	ENUM	Status pesanan
note	TEXT	Catatan pesanan
shipping_cost	INT	Biaya pengiriman
created_at	TIMESTAMP	Tanggal pesanan dibuat
completed_at	DATETIME	Tanggal pesanan selesai
receiver_name	VARCHAR(100)	Nama penerima
receiver_phone	VARCHAR(20)	No Hp penerima
delivery_address	TEXT	Alamat penerima
delivery_latitude	DECIMAL(10,8)	Latitude penerima
delivery_longitude	DECIMAL(11,8)	Longitude penerima

7. Tabel Order_Items: Tabel order_items digunakan untuk menyimpan detail produk dalam setiap pesanan.

Tabel 3.9 Tabel Order_Items

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas detail transaksi
sale_id	INT (FK)	Relasi ke tabel sales
product_id	INT (FK)	Produk yang dijual
quantity	INT	Jumlah produk
price	INT	Harga jual
cost_price	INT	Harga modal
created_at	DATETIME	Waktu pencatatan

8. Tabel Sales: Tabel sales digunakan untuk menyimpan transaksi penjualan offline.

Tabel 3.10 Tabel Sales

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas transaksi
admin_id	INT (FK)	Admin yang melakukan transaksi
total	INT	Total transaksi
order_id	INT (FK)	Referensi pesanan online
created_at	TIMESTAMP	Waktu transaksi

9. Tabel Sale_Items: Tabel sale_items digunakan untuk menyimpan detail produk pada setiap transaksi penjualan.

Tabel 3.11 Tabel Sale_Items

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas detail pesanan
order_id	INT (FK)	Relasi ke tabel orders
product_id	INT (FK)	Produk yang dipesan
quantity	INT	Jumlah produk
price	INT	Harga produk saat dipesan
cost_price	INT	Harga modal produk
is_checked	TINYINT(1)	Checklist produk

10. Tabel Cart: Tabel cart digunakan untuk menyimpan produk yang dimasukkan pelanggan ke dalam keranjang belanja.

Tabel 3.12 Tabel Cart

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas keranjang
user_id	INT (FK)	Pemilik keranjang
product_id	INT (FK)	Produk yang dipilih
quantity	INT	Jumlah produk
created_at	TIMESTAMP	Waktu ditambahkan

11. Tabel Favorites: Tabel favorites digunakan untuk menyimpan daftar produk favorit pelanggan.

Tabel 3.13 Tabel Favorites

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas favorit
user_id	INT (FK)	Pemilik data favorit
product_id	INT (FK)	Produk favorit
created_at	INT	Waktu penyimpanan

12. Tabel Notifications: Tabel notifications digunakan untuk menyimpan notifikasi yang dikirim kepada pengguna.

Tabel 3.14 Tabel Notifications

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas notifikasi
user_id	INT (FK)	Pengguna penerima
type	ENUM	Jenis notifikasi
title	VARCHAR(255)	Judul notifikasi
message	TEXT	Isi notifikasi
is_read	TINYINT(1)	Status dibaca
order_id	INT (FK)	Relasi ke pesanan
total	DECIMAL(12,2)	Nilai transaksi
created_at	TIMESTAMP	Waktu notifikasi dibuat

13. Tabel Shopping List: Tabel shopping list digunakan untuk menyimpan daftar belanja untuk pelanggan.

Tabel 3.15 Tabel Shopping List

Nama Kolom	Tipe Data	Keterangan
id	INT (PK)	Identitas daftar belanja
user_id	INT (FK)	Pemilik data daftar belanja
product_id	INT (FK)	Produk daftar belanja
quantity	INT	Jumlah produk
created_at	TIMESTAMP	Waktu produk ditambahkan
updated_at	TIMESTAMP	Waktu produk diupdate

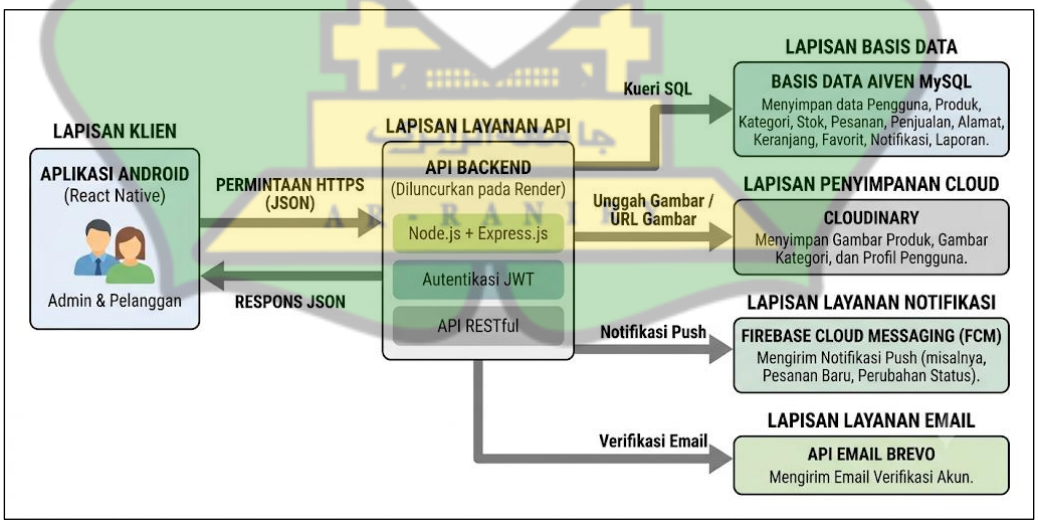
3.1.8.3 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem dilakukan untuk menggambarkan struktur utama aplikasi Stokita beserta hubungan antar komponen yang terlibat dalam proses pengelolaan stok dan penjualan online pada Fahri Mart. Arsitektur sistem berfungsi sebagai acuan dalam proses pengembangan aplikasi agar setiap komponen dapat saling berinteraksi secara terintegrasi sehingga proses pertukaran data dapat berlangsung dengan baik.

Aplikasi Stokita dikembangkan menggunakan arsitektur *Client-Server* berbasis RESTful API, yang memisahkan antarmuka pengguna (*frontend*) dengan logika bisnis (*backend*). Pada arsitektur ini, aplikasi Android berperan sebagai client yang mengirimkan permintaan (*request*) kepada server melalui protokol HTTPS. Selanjutnya, server memproses setiap permintaan, melakukan autentikasi pengguna, menjalankan logika bisnis, berinteraksi dengan basis data dan layanan pendukung lainnya, kemudian mengirimkan respons (*response*) kembali kepada aplikasi dalam format JSON.

Backend aplikasi dikembangkan menggunakan Node.js dan Express.js serta di-deploy pada layanan Render. Data utama aplikasi disimpan pada basis data MySQL yang di-hosting menggunakan Aiven, sedangkan penyimpanan gambar produk, kategori, dan foto profil memanfaatkan layanan Cloudinary. Selain itu, sistem juga terintegrasi dengan *Firebase Cloud Messaging* (FCM) untuk mengirimkan notifikasi kepada pengguna serta Brevo Email API untuk mengirimkan email verifikasi akun.

Secara umum, arsitektur sistem Stokita terdiri atas enam lapisan utama, yaitu *Client Layer*, *API Service Layer*, *Database Layer*, *Cloud Storage Layer*, *Notification Service Layer*, dan *Email Service Layer*. Visualisasi arsitektur sistem dapat dilihat pada Gambar 3.11.



Gambar 3.11 Arsitektur Sistem Stokita

1. Client Layer (Aplikasi Android)

Client Layer merupakan lapisan yang berinteraksi langsung dengan pengguna, yaitu admin dan pelanggan. Lapisan ini dibangun menggunakan *framework* React Native sehingga aplikasi dapat dijalankan pada perangkat Android. Melalui aplikasi mobile, pengguna dapat melakukan berbagai aktivitas seperti registrasi akun, login, mengelola alamat pengiriman, melihat produk, mengelola stok barang, melakukan transaksi penjualan offline, membuat pesanan online, mengelola keranjang belanja, menyimpan produk favorit, melihat daftar belanja, menerima notifikasi, serta memantau status pesanan. Komunikasi antara aplikasi dan server dilakukan menggunakan protokol HTTPS melalui RESTful API dengan format pertukaran data JSON.

2. API Service Layer (*Backend*)

API *Service Layer* merupakan lapisan *backend* yang dibangun menggunakan Node.js dan Express.js serta di-deploy pada layanan Render. Lapisan ini bertanggung jawab dalam menangani seluruh logika bisnis aplikasi, validasi data, autentikasi pengguna, pengelolaan transaksi, serta integrasi dengan berbagai layanan pendukung.

Beberapa fungsi utama pada lapisan *backend* meliputi:

a. Pengelolaan autentikasi dan otorisasi pengguna menggunakan JSON Web Token (JWT).
b. Pengelolaan verifikasi akun melalui Brevo Email API.
c. Pengelolaan data pengguna dan alamat pengiriman.
d. Pengelolaan data produk dan kategori.
e. Pengelolaan stok barang dan riwayat penambahan stok.
f. Pengelolaan transaksi penjualan offline.
g. Pengelolaan keranjang belanja dan produk favorit.
h. Pengelolaan daftar belanja pelanggan (*shopping list*).
i. Pengelolaan pesanan online beserta perubahan status pesanan.
j. Pengelolaan laporan penjualan.

k. Pengelolaan notifikasi sistem menggunakan Firebase Cloud Messaging (FCM).
l. Pengelolaan unggahan gambar melalui Cloudinary.

Backend menyediakan berbagai *endpoint* RESTful API yang digunakan oleh aplikasi Android untuk melakukan pertukaran data secara *real-time*.

3. Data Layer (Basis Data)

Data Layer merupakan lapisan yang bertanggung jawab dalam penyimpanan data aplikasi. Pada penelitian ini digunakan MySQL sebagai sistem manajemen basis data relasional. Basis data digunakan untuk menyimpan berbagai informasi yang diperlukan sistem, seperti: data pengguna, data kategori produk, data produk, data stok barang, data transaksi penjualan, data pesanan pelanggan, data keranjang belanja, data produk favorit, dan data notifikasi. Dengan penggunaan basis data relasional, integritas dan konsistensi data dapat terjaga sehingga mendukung operasional aplikasi secara optimal.

4. Cloud Storage Layer

Cloud Storage Layer digunakan untuk menyimpan file media yang diunggah oleh pengguna maupun admin. Pada aplikasi Stokita, penyimpanan gambar dilakukan menggunakan layanan Cloudinary sehingga file media tidak disimpan secara langsung pada server *backend*.

Layanan ini digunakan untuk menyimpan gambar produk, gambar kategori, serta foto profil pengguna. Setiap file yang berhasil diunggah akan menghasilkan URL dan public ID yang kemudian disimpan pada basis data untuk memudahkan proses pengelolaan maupun penghapusan file. Melalui layanan ini, proses unggah, penyimpanan, dan pengambilan gambar dapat dilakukan dengan lebih efisien serta mengurangi beban penyimpanan pada server *backend*.

5. Notification Service Layer

Notification Service Layer merupakan layanan yang digunakan untuk mengirimkan notifikasi kepada pengguna aplikasi. Pada penelitian ini digunakan *Firebase Cloud Messaging* (FCM) sebagai layanan push *notification*.

Layanan ini memungkinkan *backend* mengirimkan notifikasi secara *real-time* kepada perangkat Android, seperti notifikasi pesanan baru, perubahan status pesanan, maupun informasi transaksi lainnya. Dengan adanya layanan ini, pengguna dapat memperoleh informasi terbaru tanpa harus membuka aplikasi secara terus-menerus.

6. Email Service Layer

Email *Service Layer* digunakan untuk mengirimkan email secara otomatis kepada pengguna melalui Brevo Email API. Layanan ini dimanfaatkan pada proses registrasi akun untuk mengirimkan email verifikasi kepada pengguna. Melalui mekanisme tersebut, setiap pengguna diwajibkan melakukan verifikasi email sebelum dapat menggunakan seluruh fitur aplikasi. Penerapan layanan email ini membantu meningkatkan validitas data pengguna serta mendukung keamanan sistem.

Tabel 3.16 Deskripsi Komponen dan Layanan

Komponen	Teknologi	Keterangan Teknis
Frontend	React Native	Menyediakan antarmuka aplikasi Android
Backend	Node.js & Express.js (Render)	Menangani logika bisnis, RESTful API, dan integrasi layanan
Database	Aiven MySQL	Menyimpan data aplikasi secara terstruktur
Authentication	JSON Web Token (JWT)	Mengelola autentikasi dan otorisasi pengguna

Komponen	Teknologi	Keterangan Teknis
Cloud Storage	Cloudinary	Menyimpan gambar produk, kategori dan foto profil
Notification Service	*Firebase Cloud Messaging* (FCM)	Mengirimkan notifikasi kepada pengguna
Email Service	Brevo Email API	Mengirimkan email verifikasi akun
API Communication	REST API (JSON)	Media komunikasi antara aplikasi dan server

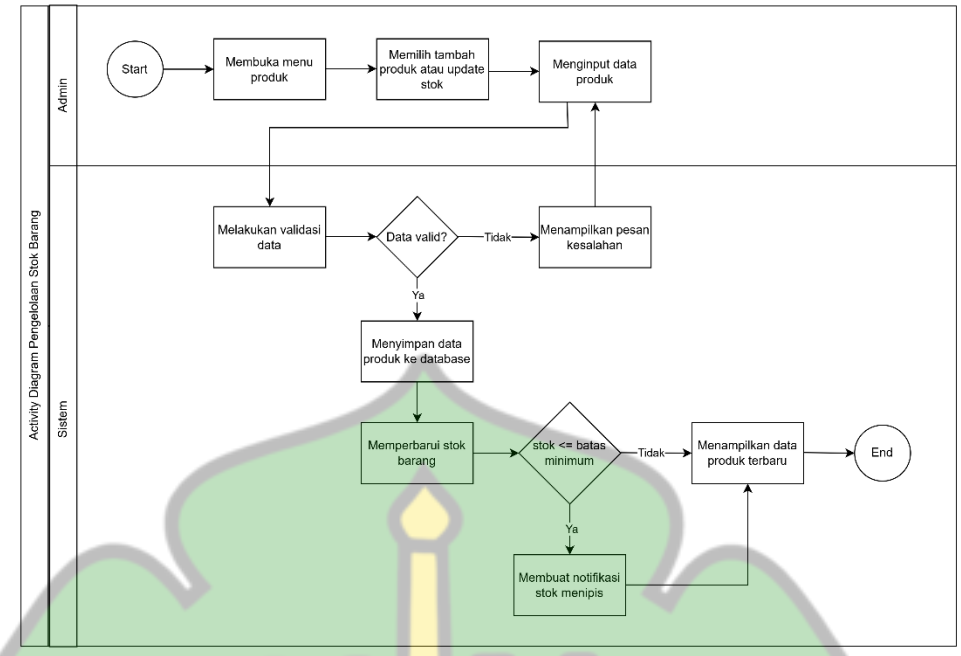
3.1.8.4 Perancangan Alur Sistem

Perancangan alur sistem dilakukan menggunakan *Activity Diagram* untuk menggambarkan urutan aktivitas yang terjadi pada aplikasi Stokita. *Activity Diagram* digunakan untuk memvisualisasikan interaksi antara pengguna dengan sistem serta menunjukkan bagaimana sistem memproses setiap aktivitas yang dilakukan oleh pengguna.

Berdasarkan hasil analisis kebutuhan dan proses bisnis yang telah dijelaskan sebelumnya, perancangan alur sistem pada aplikasi Stokita difokuskan pada tiga proses utama, yaitu: (1) Pengelolaan Stok Barang, (2) Penjualan Offline, dan (3) Penjualan Online. Ketiga proses tersebut merupakan aktivitas inti yang mendukung operasional Fahri Mart dalam mengelola persediaan barang dan transaksi penjualan.

a) Activity Diagram Pengelolaan Stok Barang

Diagram pada Gambar 3.12 menggambarkan aktivitas sistem dalam mengelola data stok barang yang tersedia pada Fahri Mart.



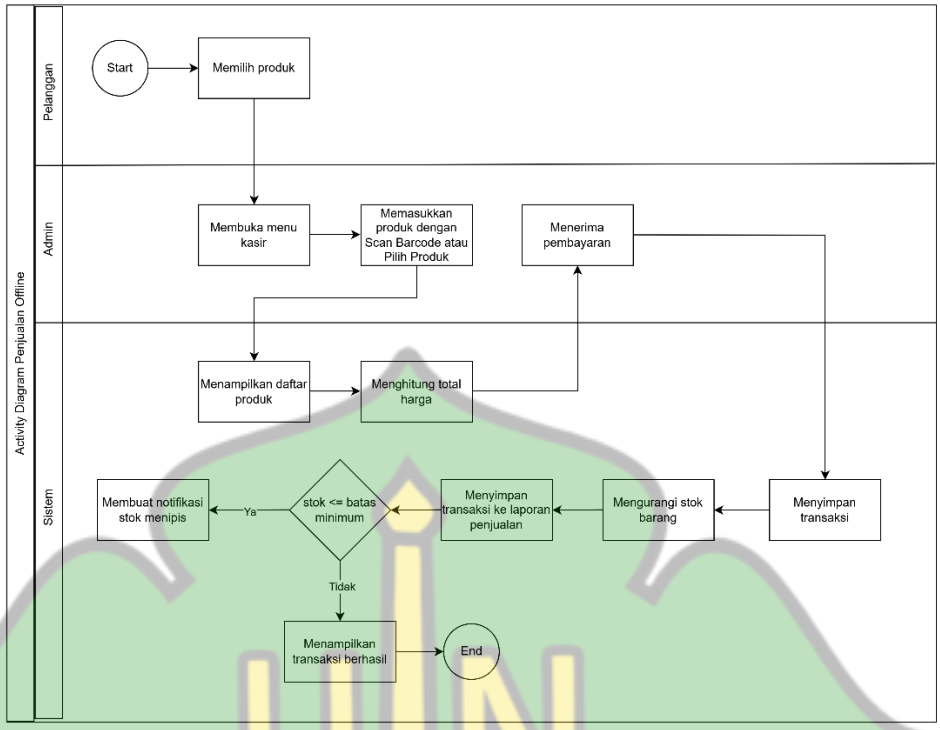
Gambar 3.12 Activity Diagram Pengelolaan Stok Barang

Proses pengelolaan stok barang diawali ketika admin membuka menu pengelolaan produk dan melakukan *input* data barang baru atau memperbarui data produk yang telah tersedia. Sistem kemudian melakukan validasi terhadap data yang dimasukkan oleh admin. Apabila data valid, sistem akan menyimpan informasi produk ke dalam basis data dan memperbarui jumlah stok barang.

Setelah data berhasil disimpan, sistem melakukan pemeriksaan terhadap jumlah stok yang tersedia. Jika jumlah stok berada di bawah batas minimum yang telah ditentukan, sistem akan menghasilkan notifikasi stok menipis kepada admin. Sebaliknya, apabila stok masih mencukupi, sistem langsung menampilkan data produk yang telah diperbarui. Proses berakhir ketika informasi stok berhasil tersimpan dan dapat diakses kembali oleh pengguna.

b) Activity Diagram Penjualan Offline

Diagram pada Gambar 3.13 menggambarkan aktivitas sistem dalam menangani transaksi penjualan yang dilakukan secara langsung di toko melalui fitur kasir pada aplikasi Stokita.



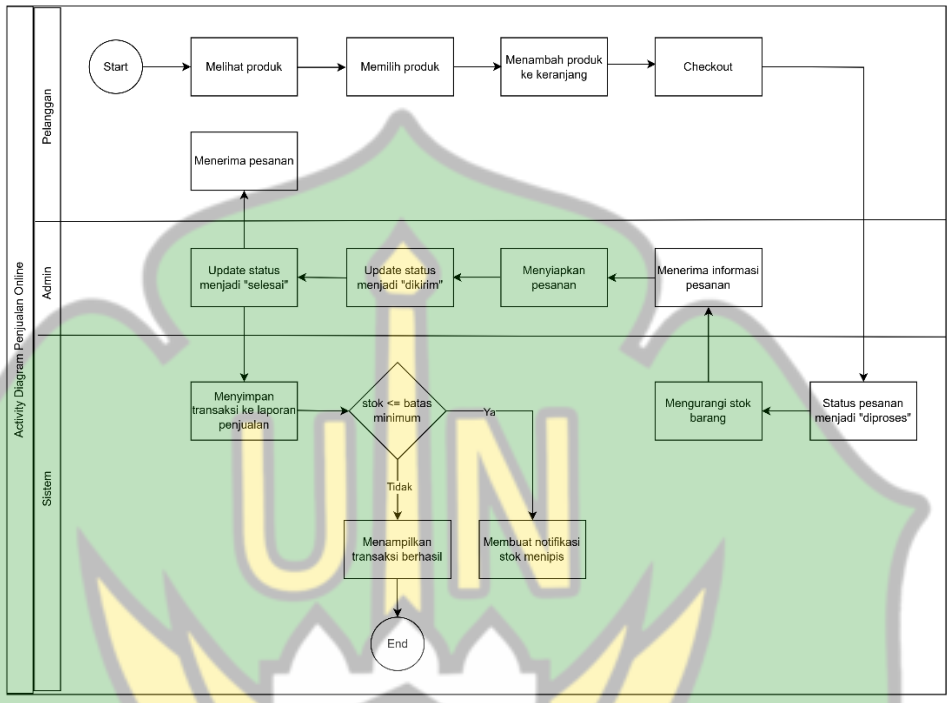
Gambar 3.13 Diagram Penjualan Offline

Proses dimulai ketika pelanggan memilih produk yang akan dibeli dan admin membuka menu transaksi penjualan. Selanjutnya admin memasukkan produk melalui pemindaian barcode atau memilih produk dari daftar yang tersedia pada sistem. Setelah seluruh produk berhasil dimasukkan, sistem menghitung total harga transaksi secara otomatis dan menampilkan rincian pembayaran.

Ketika pelanggan melakukan pembayaran, admin mengonfirmasi transaksi pada sistem. Sistem kemudian menyimpan data transaksi ke dalam basis data, mengurangi jumlah stok barang sesuai produk yang terjual, dan mencatat transaksi ke dalam laporan penjualan. Jika stok barang yang tersisa berada di bawah batas minimum, sistem akan mengirimkan notifikasi stok menipis kepada admin. Proses berakhir ketika transaksi berhasil disimpan dan laporan penjualan diperbarui.

c) Activity Diagram Penjualan Online

Diagram pada Gambar 3.14 menggambarkan aktivitas sistem dalam menangani proses pemesanan produk yang dilakukan pelanggan melalui aplikasi Stokita.



Gambar 3.14 Diagram Penjualan Online

Diagram pada Gambar 3.14 menggambarkan aktivitas sistem dalam menangani proses pemesanan produk yang dilakukan pelanggan melalui aplikasi Stokita. Proses diawali ketika pelanggan membuka aplikasi dan melihat daftar produk yang tersedia. Pelanggan kemudian memilih produk yang diinginkan dan menambahkannya ke dalam keranjang belanja. Setelah itu, pelanggan memilih alamat pengiriman yang akan digunakan dan melanjutkan proses checkout.

Pada saat proses checkout, sistem melakukan validasi ketersediaan stok untuk memastikan jumlah produk yang dipesan tidak melebihi stok yang tersedia. Apabila stok mencukupi, sistem menyimpan data pesanan beserta detail produk dan alamat pengiriman, kemudian secara otomatis mengurangi jumlah stok setiap produk yang dipesan serta mengubah status pesanan

menjadi Diproses. Selanjutnya, sistem mengirimkan notifikasi kepada admin mengenai adanya pesanan baru yang harus diproses.

Admin kemudian menyiapkan barang sesuai pesanan pelanggan. Setelah pesanan dikirimkan, admin mengubah status pesanan menjadi Dikirim sehingga pelanggan dapat memantau perkembangan pengiriman melalui aplikasi. Setelah pesanan diterima oleh pelanggan, admin mengubah status pesanan menjadi Selesai sebagai tanda bahwa proses transaksi telah berakhir.

Seluruh data pesanan dan transaksi penjualan akan tersimpan pada basis data sehingga dapat digunakan sebagai informasi pada laporan penjualan. Dengan demikian, proses penjualan online dapat berlangsung secara terintegrasi mulai dari pemesanan, pengelolaan stok, hingga penyelesaian transaksi.

3.1.8.5 Perancangan Alur Pengguna

Perancangan alur pengguna (*User Flow*) bertujuan untuk menggambarkan urutan interaksi yang dilakukan pengguna saat menggunakan aplikasi Stokita. *User Flow* digunakan untuk memvisualisasikan perpindahan antar halaman (*screen*) serta proses yang harus dilalui pengguna untuk menyelesaikan suatu aktivitas pada sistem.

Pada aplikasi Stokita terdapat dua jenis pengguna utama, yaitu Admin dan Pelanggan. Masing-masing pengguna memiliki alur penggunaan yang berbeda sesuai dengan hak akses dan kebutuhan operasionalnya. Admin berfokus pada pengelolaan produk, stok barang, transaksi penjualan, dan pesanan pelanggan, sedangkan Pelanggan berfokus pada proses pencarian produk dan pembelian secara online.

Perancangan *User Flow* pada aplikasi Stokita dibagi menjadi dua alur utama, yaitu *User Flow* Admin dan *User Flow* Pelanggan. Kedua alur tersebut terintegrasi dalam satu sistem yang sama dan saling berhubungan melalui proses transaksi penjualan online.

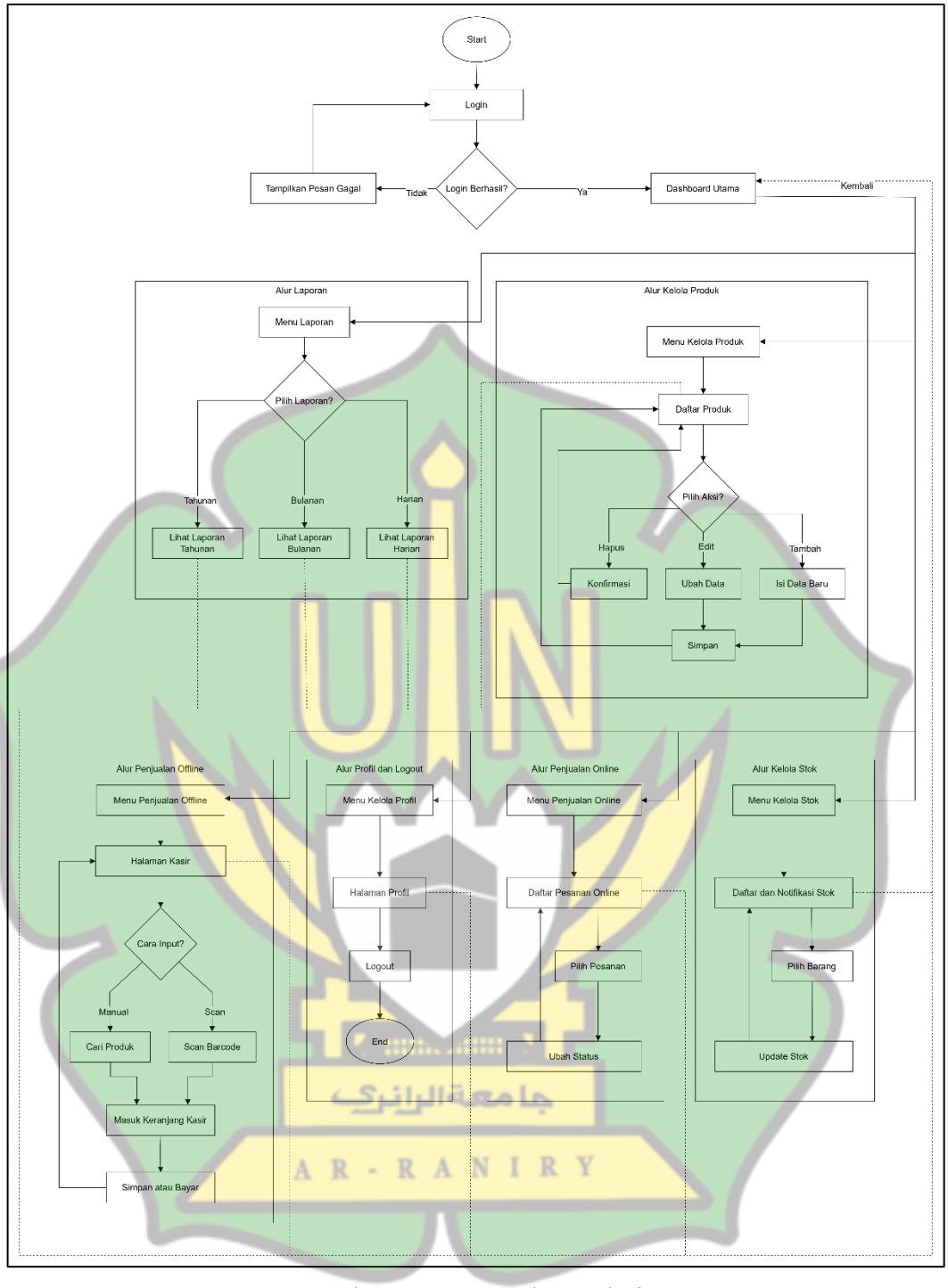
a) User Flow Admin

User Flow Admin menggambarkan alur interaksi pengguna dengan peran admin dalam mengelola operasional Fahri Mart melalui aplikasi Stokita. Proses diawali ketika admin melakukan login ke dalam sistem menggunakan akun yang telah terdaftar.

Setelah berhasil login, admin diarahkan ke halaman utama (*Home Dashboard*) yang menyediakan akses ke seluruh fitur utama aplikasi. Dari halaman tersebut admin dapat mengelola data produk, kategori produk, stok barang, transaksi penjualan offline, pesanan online, laporan penjualan, serta notifikasi sistem.

Pada proses pengelolaan produk, admin dapat menambah, mengubah, atau menghapus data produk. Pada proses pengelolaan stok, admin dapat melakukan pembaruan jumlah stok barang dan memantau notifikasi stok menipis. Untuk transaksi penjualan offline, admin dapat melakukan pencatatan transaksi melalui fitur kasir yang terintegrasi dengan sistem stok.

Selain itu, admin juga dapat mengelola pesanan online yang dibuat pelanggan. Admin dapat memantau status pesanan, mengubah status pesanan menjadi Dikirim dan Selesai, serta melihat laporan penjualan yang dihasilkan dari transaksi online maupun offline. Setelah seluruh aktivitas selesai dilakukan, admin dapat keluar dari sistem melalui fitur *logout*.



Gambar 3.15 User Flow Admin

b) User Flow Pelanggan

User Flow Pelanggan menggambarkan alur interaksi pelanggan dalam melakukan pembelian produk melalui aplikasi Stokita. Proses dimulai ketika

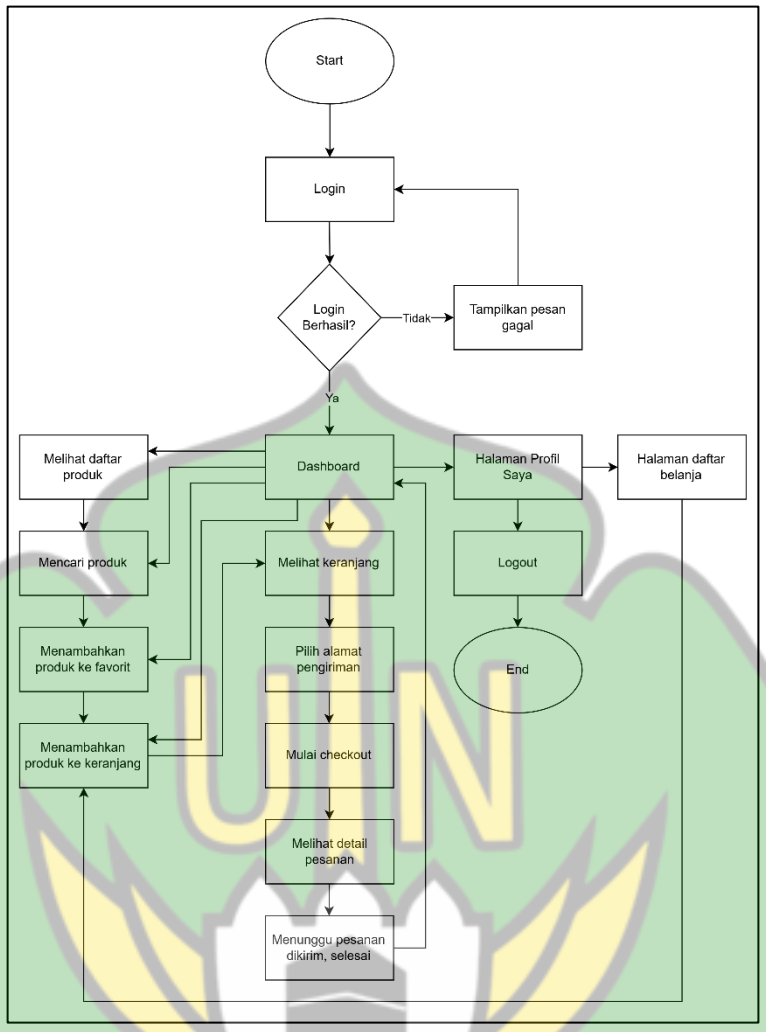
pelanggan melakukan registrasi akun atau login menggunakan akun yang telah dimiliki.

Setelah berhasil login, pelanggan akan diarahkan ke halaman utama aplikasi yang menampilkan daftar produk dan kategori produk yang tersedia. Pelanggan dapat melakukan pencarian produk, melihat detail produk, serta menambahkan produk yang diinginkan ke dalam keranjang belanja.

Setelah produk ditambahkan ke keranjang, pelanggan dapat melanjutkan proses checkout untuk membuat pesanan. Sistem kemudian menyimpan data pesanan dan secara otomatis mengubah status pesanan menjadi Diproses.

Selanjutnya pelanggan dapat memantau perkembangan pesanan melalui halaman pesanan. Ketika admin mengubah status pesanan menjadi Dikirim, informasi tersebut akan ditampilkan pada aplikasi pelanggan. Setelah pesanan diterima dan status diubah menjadi Selesai, transaksi akan tercatat sebagai transaksi yang berhasil diselesaikan.

Selain melakukan pembelian, pelanggan juga dapat mengelola profil akun, melihat daftar produk favorit, serta melihat riwayat pesanan yang pernah dilakukan. Proses penggunaan aplikasi berakhir ketika pelanggan melakukan *logout* dari sistem.



Gambar 3.16 User Flow Pelanggan

3.1.8.6 Perancangan Antarmuka

Perancangan antarmuka pengguna (*User Interface*) dilakukan sebagai tahap konseptual untuk menjembatani interaksi antara aktor (Admin dan Pelanggan) dengan logika sistem Stokita. Desain ini disusun dengan mengacu pada prinsip *User-Centered Design*, yang menitikberatkan pada aspek kemudahan navigasi, keterbacaan informasi, dan konsistensi elemen visual di seluruh halaman aplikasi.

Strategi perancangan antarmuka pada aplikasi Stokita dibagi menjadi dua kelompok utama berdasarkan fungsi operasionalnya:

1. **Antarmuka Sisi Admin (Manajemen Operasional)**: Dirancang dengan tata letak yang mengutamakan efisiensi pemrosesan data. Fokus utama pada sisi admin meliputi kemudahan akses fitur pemindaian *barcode*, visualisasi

dasbor ringkasan stok, serta manajemen status pesanan pelanggan secara *real-time*.

2. **Antarmuka Sisi Pelanggan (Pengalaman Belanja Digital)**: Dirancang dengan konsep *e-commerce* modern yang intuitif. Penekanan desain diberikan pada kemudahan pencarian produk, kejelasan informasi harga, serta penyederhanaan alur keranjang belanja dan proses *checkout*.

Dalam pengembangannya, antarmuka ini dirancang untuk berjalan secara responsif pada perangkat Android menggunakan komponen modular dari *framework* React Native. Seluruh elemen antarmuka tersebut diintegrasikan langsung dengan layanan RESTful API untuk memastikan data yang ditampilkan kepada pengguna selalu sinkron dengan basis data pusat.

Adapun rincian perancangan halaman utama yang disiapkan meliputi halaman Login, Registrasi, Beranda (Dashboard), Daftar Produk, Keranjang Belanja, Detail Pesanan, serta Laporan Penjualan bagi admin. Visualisasi hasil rancangan dan implementasi antarmuka pengguna secara detail disajikan pada Bab IV bagian 4.1.4.

3.1.9 Perancangan Pengujian Sistem

Perancangan pengujian sistem dilakukan untuk memastikan bahwa aplikasi Stokita yang dikembangkan telah berfungsi sesuai dengan kebutuhan fungsional dan proses bisnis yang telah dirancang. Pengujian bertujuan untuk memverifikasi bahwa setiap fitur dapat berjalan dengan baik, menghasilkan keluaran yang sesuai, serta mampu mendukung proses pengelolaan stok dan penjualan pada Fahri Mart.

Pengujian dalam penelitian ini terdiri atas *API Testing*, *Black Box Testing*, dan *System Usability Scale* (SUS). *API Testing* digunakan untuk memverifikasi fungsi endpoint RESTful API pada sisi backend berdasarkan request dan response yang dihasilkan. *Black Box Testing* digunakan untuk menguji fungsi aplikasi dari sisi pengguna berdasarkan masukan (*input*) dan keluaran (*output*) tanpa memperhatikan struktur kode program. Sementara itu, pengujian SUS digunakan untuk mengukur tingkat kemudahan penggunaan (*usability*) aplikasi berdasarkan persepsi pengguna.

Pengujian dilakukan pada komponen aplikasi yang saling terintegrasi dalam arsitektur *Client-Server*, yaitu aplikasi Android, *backend API*, dan basis data. *API Testing* dilakukan menggunakan Postman. Sementara itu, *Black Box Testing* dilakukan dengan mengoperasikan aplikasi Stokita secara langsung berdasarkan skenario pengujian yang telah ditentukan. Pengujian SUS dilakukan dengan memberikan kuesioner kepada pengguna setelah menggunakan aplikasi.

Fokus pengujian sistem meliputi:

1. Setiap endpoint API dapat memproses request dan menghasilkan response sesuai dengan kebutuhan sistem.
2. Fitur autentikasi, meliputi registrasi dan login, dapat berjalan sesuai dengan hak akses pengguna.
3. Pengelolaan produk dan stok barang dapat dilakukan sesuai dengan fungsi yang dirancang.
4. Transaksi penjualan *offline* dapat diproses dan tersimpan ke dalam basis data.
5. Proses pemesanan *online* dapat dilakukan sesuai dengan alur yang telah dirancang.
6. Perubahan status pesanan dari Diproses, Dikirim, hingga Selesai dapat berjalan dengan baik.
7. Sistem dapat memperbarui stok barang sesuai dengan transaksi yang terjadi.
8. Laporan penjualan dapat menampilkan data transaksi sesuai dengan periode yang dipilih.
9. Fitur-fitur pada aplikasi dapat digunakan sesuai dengan skenario *Black Box Testing*.
10. Aplikasi memiliki tingkat kemudahan penggunaan yang dapat diukur berdasarkan hasil pengujian SUS.

3.1.9.1 Skenario Pengujian Black Box

Skenario pengujian *Black Box* disusun sebagai panduan untuk memverifikasi bahwa setiap fungsionalitas pada aplikasi Stokita berjalan sesuai dengan kebutuhan sistem yang telah didefinisikan pada tahap analisis. Pengujian ini difokuskan sepenuhnya pada perilaku eksternal sistem dengan memberikan berbagai variasi

data masukan (*input*) dan mengamati keluaran (*output*) yang dihasilkan, tanpa melibatkan analisis terhadap struktur kode internal program maupun logika *database* di dalamnya.

Perancangan skenario pengujian ini mencakup seluruh fitur utama aplikasi yang dioperasikan oleh Admin Fahri Mart maupun pelanggan, dengan rincian cakupan sebagai berikut:

1. **Modul Autentikasi dan Keamanan**: Menguji alur registrasi pelanggan, verifikasi email, serta validasi hak akses login berdasarkan peran pengguna (*admin* atau pelanggan) menggunakan mekanisme *Role-Based Access Control* (RBAC).
2. **Manajemen Produk dan Inventaris**: Menguji kemampuan admin dalam menambah, mengubah, dan menghapus data produk, kategori, serta akurasi fungsionalitas penambahan stok barang (*Stock In*).
3. **Fitur Belanja Pelanggan**: Menguji interaksi pelanggan dalam pencarian produk, penggunaan fitur daftar belanja (*shopping list*), manajemen keranjang belanja, hingga pemilihan alamat pengiriman.
4. **Alur Transaksi Digital**: Menguji validitas proses *checkout* pemesanan online, sinkronisasi stok otomatis saat pesanan dibuat, serta manajemen status pesanan oleh admin (Diproses, Dikirim, Selesai).
5. **Operasional Kasir Offline**: Menguji keandalan fitur pemindaian *barcode* produk dan fungsionalitas pencatatan transaksi penjualan langsung di toko melalui antarmuka kasir.
6. **Pelaporan dan Profil**: Menguji akurasi penyajian laporan penjualan harian, bulanan, dan tahunan, serta fitur pengelolaan informasi akun pengguna.

Setiap skenario dirancang untuk mendeteksi potensi kesalahan pada antarmuka pengguna, kesalahan fungsi, serta konsistensi penanganan galat (*error handling*) ketika sistem menerima masukan data yang tidak valid. Adapun detail setiap kasus uji (*test case*) beserta hasil aktual dan validasi tingkat keberhasilannya disajikan secara terperinci pada Bab IV bagian 4.2.2 Pengujian Fungsional.

3.1.9.2 Skenario Pengujian API

Skenario pengujian API dirancang untuk memvalidasi seluruh fungsionalitas *backend* dan integritas pertukaran data antara aplikasi Android dengan server melalui protokol HTTPS. Pengujian ini menggunakan perangkat lunak Postman sebagai alat bantu untuk mengirimkan berbagai variasi permintaan (*request*) serta menganalisis respons (*response*) yang diberikan oleh server Node.js. Fokus utama pengujian adalah memastikan integritas logika bisnis, akurasi validasi data, serta keandalan mekanisme keamanan akses.

Perancangan skenario pengujian API mencakup modul-modul krusial sebagai berikut:

1. **Modul Autentikasi dan Profil**: Memvalidasi alur registrasi pelanggan, manajemen token akses (*JSON Web Token*), serta fungsionalitas pengelolaan data profil dan alamat pengiriman pengguna.
2. **Manajemen Katalog dan Inventaris**: Menguji *endpoint* untuk penambahan data produk, pengelolaan kategori, pencarian produk berbasis *barcode*, serta mekanisme penambahan stok barang (*Stock In*).
3. **Fungsionalitas Belanja**: Menguji logika sistem dalam mengelola daftar produk favorit, daftar belanja (*shopping list*), dan pemrosesan data pada keranjang belanja pelanggan.
4. **Transaksi dan Alur Pesanan**: Memvalidasi alur pembuatan pesanan online (*checkout*), manajemen perubahan status pesanan (Diproses, Dikirim, Selesai), serta integrasi pencatatan transaksi penjualan kasir offline.
5. **Laporan dan Sistem Notifikasi**: Menguji keandalan penarikan data laporan penjualan harian maupun bulanan, serta fungsionalitas pengiriman dan status pembacaan notifikasi sistem.
6. **Keamanan dan Otorisasi (RBAC)**: Memverifikasi pembatasan hak akses guna memastikan *endpoint* sensitif yang dikhususkan bagi admin tidak dapat diakses oleh pelanggan atau pengguna tanpa token yang valid.

Seluruh detail kasus uji (*test case*) API, termasuk metode HTTP yang digunakan, status kode respons, dan sinkronisasi data pada basis data MySQL disajikan secara komprehensif pada Bab IV bagian 4.2.1 Pengujian RESTful API.

3.1.9.3 Perancangan Pengujian System Usability Scale (SUS)

Perancangan pengujian *System Usability Scale* (SUS) dilakukan untuk mengukur tingkat kemudahan penggunaan (*usability*) aplikasi Stokita berdasarkan persepsi pengguna setelah menggunakan aplikasi. Pengujian ini bertujuan untuk mengetahui sejauh mana aplikasi mudah dipelajari, mudah digunakan, serta mampu memberikan pengalaman penggunaan yang baik bagi pengguna.

Pengujian SUS dilaksanakan setelah seluruh fitur utama aplikasi selesai diimplementasikan dan telah melewati pengujian fungsional menggunakan *Black Box Testing*. Sebelum mengisi kuesioner, responden diminta untuk mencoba fitur-fitur utama aplikasi sesuai dengan hak akses masing-masing, seperti proses login, pengelolaan produk, transaksi penjualan, pemesanan online, serta pengelolaan profil pengguna. Setelah seluruh skenario penggunaan selesai dilakukan, responden diminta mengisi kuesioner *System Usability Scale* (SUS).

Instrumen SUS terdiri atas sepuluh butir pernyataan dengan menggunakan skala Likert lima tingkat, mulai dari Sangat Tidak Setuju (1) hingga Sangat Setuju (5). Daftar pernyataan yang digunakan pada pengujian ini dijelaskan pada Subbab 3.3.2 Kuesioner *System Usability Scale* (SUS).

Skor hasil pengisian kuesioner kemudian dihitung menggunakan prosedur perhitungan standar *System Usability Scale* (SUS). Untuk setiap pernyataan bernomor ganjil, nilai yang diberikan responden dikurangi 1, sedangkan untuk setiap pernyataan bernomor genap, nilai diperoleh dari 5 dikurangi skor jawaban responden. Selanjutnya, seluruh skor dijumlahkan dan dikalikan dengan faktor 2,5 sehingga diperoleh nilai SUS dalam rentang 0–100.

Nilai SUS yang diperoleh digunakan untuk mengevaluasi tingkat *usability* aplikasi Stokita. Semakin tinggi nilai SUS yang dihasilkan, maka semakin baik tingkat kemudahan penggunaan aplikasi berdasarkan penilaian responden.

Tabel 3.17 Interpretasi Nilai System Usability Scale (SUS)

Rentang Nilai SUS	Interpretasi
> 80,3	Sangat Baik (Excellent)
68 – 80,3	Baik (Good)
68	Rata-rata (Average)
51 – 67	Kurang (Marginal)
< 51	Buruk (Poor)

3.2 Populasi dan Sampel

Penelitian ini dilakukan pada UMKM Fahri Mart sebagai objek penerapan aplikasi Stokita. Subjek penelitian difokuskan pada pengguna yang terlibat secara langsung dalam proses pengelolaan stok barang dan transaksi penjualan menggunakan aplikasi yang dikembangkan. Penentuan subjek penelitian dilakukan untuk memperoleh data yang dapat digunakan dalam proses pengujian fungsional maupun pengujian tingkat kemudahan penggunaan (*usability*) aplikasi.

Mengingat penelitian ini merupakan penelitian pengembangan perangkat lunak (*software engineering research*), maka teknik pengambilan sampel menggunakan pendekatan *non-probability sampling*, yaitu pemilihan responden berdasarkan keterlibatan mereka dalam penggunaan aplikasi Stokita.

3.2.1 Populasi

Menurut Sugiyono (2019), populasi adalah wilayah generalisasi yang terdiri atas objek atau subjek yang memiliki karakteristik tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya.

Populasi dalam penelitian ini adalah seluruh pengguna yang terlibat dalam operasional Fahri Mart dan menggunakan aplikasi Stokita, baik sebagai pengelola sistem maupun pengguna aplikasi untuk melakukan transaksi. Populasi tersebut terdiri atas admin toko dan pelanggan yang berinteraksi langsung dengan fitur-fitur aplikasi seperti pengelolaan produk, pengelolaan stok barang, transaksi penjualan offline, serta pemesanan produk secara online.

3.2.2 Sampel

Teknik pengambilan sampel yang digunakan dalam penelitian ini adalah Purposive Sampling. Menurut Sugiyono (2019), purposive sampling adalah teknik penentuan sampel dengan pertimbangan atau kriteria tertentu untuk menyesuaikan dengan tujuan penelitian. Teknik ini dipilih karena evaluasi kegunaan (usability) sistem memerlukan responden yang memenuhi syarat teknis spesifik, yaitu pengguna yang benar-benar telah berinteraksi langsung dengan sistem yang dikembangkan agar penilaian yang diberikan bersifat objektif dan relevan.

Untuk mendapatkan sampel yang representatif dan sesuai dengan tujuan evaluasi aplikasi Stokita menggunakan metode System Usability Scale (SUS), peneliti menetapkan kriteria inklusi (inclusion criteria) sebagai berikut:

1. Kriteria Sampel Admin (Pengelola Toko):
    a) Merupakan pemilik atau karyawan aktif yang mengelola operasional harian Fahri Mart.
    b) Bertanggung jawab langsung dalam manajemen persediaan stok, pencatatan transaksi kasir, dan pemrosesan pesanan online.
    c) Telah mencoba seluruh fitur utama pada sisi admin dashboard.
2. Kriteria Sampel Pelanggan:
    a) Merupakan mahasiswa atau masyarakat umum sebagai representasi kelompok masyarakat digital (early adopters) yang memiliki literasi teknologi baik untuk menguji sistem transaksi online.
    b) Memiliki perangkat smartphone bersistem operasi Android.
    c) Telah mencoba seluruh alur fitur utama aplikasi Stokita pada sisi pelanggan, mulai dari pencarian produk, fitur daftar belanja, keranjang belanja, hingga melakukan simulasi proses checkout pemesanan online.
    d) Tidak disyaratkan harus berada di lokasi fisik toko Fahri Mart, karena fokus pengujian ditekankan pada kegunaan (usability) sistem aplikasi penjualan online secara digital.
    e) Bersedia secara sukarela berpartisipasi sebagai responden pengujian usability dengan mengisi kuesioner SUS yang disediakan.

Berdasarkan kriteria pertimbangan tersebut, distribusi sampel dalam penelitian ini terdiri dari:

1. Admin Fahri Mart : 2 Orang
2. Pelanggan Fahri Mart : 50 Orang

Dengan demikian, jumlah keseluruhan sampel dalam penelitian ini adalah 52 responden. Penetapan jumlah ini disesuaikan dengan jumlah pengguna aktual yang berpartisipasi aktif dalam tahapan pengujian fungsionalitas dan uji coba kegunaan (usability) aplikasi Stokita di lapangan

3.3 Instrumen Penelitian

Instrumen penelitian merupakan alat yang digunakan untuk memperoleh data selama proses penelitian. Pada penelitian ini, instrumen penelitian digunakan untuk mengumpulkan data mengenai kinerja fungsional sistem serta tingkat kemudahan penggunaan aplikasi Stokita. Instrumen yang digunakan disesuaikan dengan tujuan penelitian, yaitu menguji apakah aplikasi telah berfungsi sesuai kebutuhan serta mengetahui tingkat penerimaan pengguna terhadap aplikasi yang dikembangkan.

Instrumen penelitian yang digunakan terdiri atas dua jenis, yaitu lembar pengujian fungsional (*Black Box Testing*) dan kuesioner *System Usability Scale* (SUS).

3.3.1 Lembar Pengujian Fungsional (Test Case)

Instrumen pengujian fungsional yang digunakan pada penelitian ini berupa lembar pengujian *Black Box Testing*. Instrumen ini digunakan sebagai pedoman untuk mencatat hasil pengujian terhadap setiap fungsi utama pada aplikasi Stokita berdasarkan skenario pengujian yang telah dirancang pada Subbab 3.1.9.

Lembar pengujian memuat informasi mengenai fitur yang diuji, skenario pengujian, data uji (*test case*), hasil yang diharapkan, hasil pengujian, serta kesimpulan pengujian. Melalui instrumen ini dapat diketahui apakah setiap fungsi pada aplikasi telah berjalan sesuai dengan kebutuhan fungsional yang telah ditentukan.

Pengujian dilakukan terhadap fitur-fitur utama aplikasi Android serta layanan RESTful API menggunakan aplikasi Postman. Pengujian pada aplikasi Android bertujuan untuk memastikan setiap fungsi yang berinteraksi langsung dengan pengguna dapat berjalan dengan baik, sedangkan pengujian API dilakukan untuk memverifikasi bahwa setiap *endpoint* mampu memproses permintaan (*request*) dan menghasilkan respons (*response*) sesuai dengan spesifikasi yang telah ditetapkan.

Fitur-fitur yang diuji meliputi: Registrasi pengguna, Login pengguna, Pengelolaan profil pengguna, Pengelolaan alamat pengguna, Pengelolaan produk, Pengelolaan stok barang, Pencarian produk menggunakan barcode, Keranjang belanja, Produk favorit, Daftar belanja, Pemesanan produk secara online, Penjualan offline, Laporan penjualan, Notifikasi, dan Keamanan sistem menggunakan autentikasi JSON Web Token (JWT).

Setiap skenario pengujian akan dinilai berdasarkan kesesuaian antara hasil yang diperoleh dengan hasil yang diharapkan. Pengujian dinyatakan Berhasil apabila sistem menghasilkan keluaran sesuai dengan skenario pengujian, sedangkan pengujian dinyatakan Gagal apabila keluaran yang dihasilkan tidak sesuai dengan hasil yang diharapkan.

Tabel 3.18 Format Lembar Pengujian Black Box Testing

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Login dengan data valid	Email dan password valid	Berhasil masuk ke halaman utama	…	…

3.3.2 Kuesioner System Usability Scale (SUS)

Instrumen penelitian yang digunakan untuk mengukur tingkat *usability* aplikasi Stokita adalah kuesioner *System Usability Scale* (SUS). Instrumen ini terdiri atas sepuluh butir pernyataan baku yang digunakan untuk memperoleh persepsi pengguna terhadap kemudahan penggunaan aplikasi. Responden diminta

memberikan penilaian terhadap setiap pernyataan menggunakan skala Likert lima tingkat, yaitu:

a) Skor 1 : Sangat Tidak Setuju
b) Skor 2 : Tidak Setuju
c) Skor 3 : Netral
d) Skor 4 : Setuju
e) Skor 5 : Sangat Setuju

Daftar pernyataan *System Usability Scale* (SUS) yang digunakan dalam penelitian ditunjukkan pada Tabel 3.19 berikut.

Tabel 3.19 Daftar Pernyataan System Usability Scale (SUS)

No	Kode	Butir Pernyataan	Sifat Pernyataan
1	Q1	Saya merasa akan sering menggunakan sistem ini.	Positif
2	Q2	Saya merasa sistem ini terlalu rumit.	Negatif
3	Q3	Saya merasa sistem ini mudah digunakan.	Positif
4	Q4	Saya merasa perlu bantuan teknis untuk dapat menggunakan sistem ini.	Negatif
5	Q5	Fitur-fitur dalam sistem ini terintegrasi dengan baik.	Positif
6	Q6	Saya merasa sistem ini memiliki terlalu banyak inkonsistensi.	Negatif
7	Q7	Saya membayangkan kebanyakan orang akan cepat belajar menggunakan sistem ini.	Positif
8	Q8	Saya merasa sistem ini terlalu membingungkan saat digunakan.	Negatif
9	Q9	Saya merasa percaya diri saat menggunakan sistem ini.	Positif
10	Q10	Saya harus banyak belajar sebelum dapat menggunakan sistem ini.	Negatif

Pernyataan pada instrumen SUS terdiri atas pernyataan positif (nomor ganjil) dan pernyataan negatif (nomor genap). Pembagian tersebut bertujuan untuk mengurangi bias jawaban responden sehingga hasil evaluasi lebih objektif. Skor yang diperoleh dari setiap responden selanjutnya dihitung menggunakan prosedur

perhitungan standar *System Usability Scale* (SUS) untuk memperoleh nilai *usability* aplikasi Stokita.

3.4 Teknik Analisis Data

Teknik analisis data merupakan proses mengolah data yang diperoleh selama penelitian sehingga dapat menghasilkan informasi yang sesuai dengan tujuan penelitian. Pada penelitian ini, teknik analisis data digunakan untuk menganalisis hasil pengujian fungsional aplikasi menggunakan *Black Box Testing* serta hasil evaluasi kemudahan penggunaan menggunakan *System Usability Scale* (SUS).

Teknik analisis data yang digunakan dalam penelitian ini adalah statistik deskriptif. Menurut Sugiyono (2019), statistik deskriptif merupakan teknik analisis data yang digunakan untuk mendeskripsikan atau menggambarkan data yang telah terkumpul tanpa bermaksud membuat generalisasi terhadap populasi yang lebih luas.

Pada penelitian ini, data hasil pengujian *Black Box Testing* dianalisis untuk mengetahui tingkat keberhasilan fungsi-fungsi aplikasi, sedangkan data hasil pengisian kuesioner SUS dianalisis untuk mengetahui tingkat *usability* aplikasi Stokita berdasarkan persepsi pengguna.

3.4.1 Analisis Skor Usability

Data hasil pengisian kuesioner *System Usability Scale* (SUS) dianalisis menggunakan prosedur perhitungan standar SUS sehingga diperoleh skor *usability* pada rentang 0–100.

Tahapan perhitungan skor SUS adalah sebagai berikut:

1. Untuk pernyataan bernomor ganjil (1, 3, 5, 7, dan 9), skor kontribusi diperoleh dari: Skor Jawaban − 1
2. Untuk pernyataan bernomor genap (2, 4, 6, 8, dan 10), skor kontribusi diperoleh dari: 5 − Skor Jawaban
3. Seluruh skor kontribusi dijumlahkan.
4. Hasil penjumlahan kemudian dikalikan dengan 2,5 sehingga diperoleh skor akhir SUS.

Selanjutnya skor SUS dari seluruh responden dihitung nilai rata-ratanya untuk mengetahui tingkat *usability* aplikasi Stokita. Interpretasi skor dilakukan menggunakan *Acceptability Range*, *Grade Scale*, dan *Adjective Rating* sehingga dapat diketahui tingkat penerimaan serta tingkat kemudahan penggunaan aplikasi berdasarkan persepsi pengguna.

3.4.2 Analisis Validitas dan Reliabilitas Instrumen

Sebelum data hasil kuesioner System Usability Scale (SUS) digunakan untuk mengukur tingkat usability aplikasi Stokita, terlebih dahulu dilakukan pengujian validitas terhadap instrumen penelitian. Uji validitas bertujuan untuk mengetahui sejauh mana setiap butir pernyataan pada kuesioner SUS mampu mengukur variabel yang dimaksud secara tepat dan konsisten.

Pengujian validitas dilakukan menggunakan teknik korelasi item-total (*corrected item-total correlation*) dengan pendekatan *Pearson Product Moment*, yaitu dengan mengorelasikan skor setiap butir pernyataan (yang telah dikonversi ke dalam skala 0–4 sesuai aturan perhitungan SUS) dengan skor total dari seluruh butir lainnya. Rumus korelasi Pearson Product Moment yang digunakan adalah sebagai berikut.

$$r\ hitung = \frac{n\sum XY - (\sum X)(\sum Y)}{\sqrt{[n\sum X^2 - (\sum X)^2]\,[n\sum Y^2 - (\sum Y)^2]}}$$

Keterangan:

- r hitung = koefisien korelasi antara skor item dengan skor total
- X = skor tiap butir pernyataan
- Y = skor total seluruh butir
- n = jumlah responden

Nilai r hitung yang diperoleh kemudian dibandingkan dengan nilai r tabel pada taraf signifikansi α = 0,05 dengan derajat kebebasan (df) = n − 2. Suatu butir pernyataan dinyatakan valid apabila nilai r hitung > r tabel, dan dinyatakan tidak valid apabila r hitung ≤ r tabel.

Selain uji validitas, dilakukan pula uji reliabilitas instrumen menggunakan koefisien Cronbach's Alpha, yaitu teknik yang digunakan untuk mengukur konsistensi internal antar butir pernyataan dalam suatu instrumen. Rumus Cronbach's Alpha yang digunakan adalah sebagai berikut.

$$\alpha = \left(\frac{k}{k-1}\right)\left(1 - \frac{\sum {\sigma_i}^2}{{\sigma_t}^2}\right)$$

Keterangan:

- α = koefisien reliabilitas Cronbach's Alpha
- k = jumlah butir pernyataan
- $\Sigma\sigma_i^2$ = jumlah varians skor tiap butir pernyataan
- σ_t^2 = varians skor total

Tahapan perhitungan Cronbach's Alpha dilakukan sebagai berikut:

1. Menghitung varians skor setiap butir pernyataan (σ_i^2) dari seluruh responden.
2. Menjumlahkan seluruh varians butir pernyataan ($\Sigma\sigma_i^2$).
3. Menghitung varians skor total (σ_t^2), yaitu varians dari penjumlahan skor seluruh butir pada setiap responden.
4. Memasukkan nilai-nilai tersebut ke dalam rumus Cronbach's Alpha untuk memperoleh koefisien reliabilitas.

Instrumen dinyatakan reliabel apabila nilai $\alpha > 0{,}60$, dengan kategori interpretasi sebagai berikut: $\alpha \leq 0{,}20$ (sangat rendah), $0{,}20 < \alpha \leq 0{,}40$ (rendah), $0{,}40 < \alpha \leq 0{,}60$ (cukup), $0{,}60 < \alpha \leq 0{,}80$ (tinggi), dan $0{,}80 < \alpha \leq 1{,}00$ (sangat tinggi).

3.4.3 Analisis Fungsional

Data hasil pengujian *Black Box Testing* dianalisis secara deskriptif berdasarkan kesesuaian antara hasil pengujian dengan hasil yang diharapkan pada setiap skenario pengujian. Selain itu, tingkat keberhasilan pengujian dihitung menggunakan persentase keberhasilan (*Success Rate*) untuk mengetahui tingkat keberhasilan implementasi fungsi-fungsi aplikasi Stokita. Persentase keberhasilan dihitung menggunakan rumus berikut:

$$\text{Persentase Keberhasilan} = \frac{\text{Jumlah Fitur Berhasil}}{\text{Jumlah Fitur yang Diuji}} \times 100\%$$

Hasil perhitungan digunakan untuk mengetahui tingkat keberhasilan implementasi aplikasi Stokita. Sistem dinyatakan berfungsi dengan baik apabila seluruh fitur yang diuji menghasilkan keluaran sesuai dengan hasil yang diharapkan dan memperoleh tingkat keberhasilan sebesar 100% pada pengujian *Black Box*.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Tahap implementasi sistem merupakan proses penerapan hasil perancangan yang telah dijelaskan pada Bab III ke dalam bentuk aplikasi yang dapat dijalankan. Pada tahap ini seluruh kebutuhan fungsional, perancangan basis data, arsitektur sistem, serta antarmuka pengguna diimplementasikan menggunakan teknologi yang telah ditentukan.

Aplikasi Stokita dikembangkan menggunakan arsitektur *Client–Server* yang memisahkan aplikasi Android sebagai client dan *backend* sebagai penyedia layanan (*service provider*). Komunikasi antara aplikasi Android dan *backend* dilakukan melalui RESTful API menggunakan format pertukaran data JSON melalui protokol HTTPS.

Backend dikembangkan menggunakan Node.js dengan *framework* Express.js, sedangkan *frontend* dibangun menggunakan React Native sehingga dapat dijalankan pada perangkat Android. Penyimpanan data menggunakan Aiven MySQL sebagai basis data relasional, sedangkan penyimpanan gambar produk, kategori, dan foto profil menggunakan layanan Cloudinary. Selain itu, sistem juga memanfaatkan *Firebase Cloud Messaging* (FCM) untuk pengiriman notifikasi secara *real-time* dan Brevo Email API untuk proses verifikasi email pengguna.

Implementasi sistem dibagi menjadi dua bagian utama, yaitu implementasi *backend* sebagai penyedia layanan API serta implementasi *frontend* sebagai aplikasi Android yang digunakan oleh admin maupun pelanggan.

4.1.1 Implementasi Arsitektur Sistem

Implementasi arsitektur sistem dilakukan dengan menerapkan arsitektur *Client–Server* berbasis RESTful API. *Backend* dibangun menggunakan Node.js dan Express.js, sedangkan aplikasi Android dikembangkan menggunakan React Native.

Backend bertanggung jawab menangani seluruh logika bisnis aplikasi, autentikasi pengguna menggunakan JSON Web Token (JWT), pengelolaan data produk, kategori, stok barang, transaksi penjualan offline, pemesanan online, pengelolaan alamat pengguna, daftar favorit, daftar belanja, notifikasi, laporan penjualan, serta komunikasi dengan layanan pihak ketiga seperti Cloudinary, *Firebase Cloud Messaging*, dan Brevo Email API. *Frontend* bertugas menyediakan antarmuka pengguna serta mengirimkan permintaan (*request*) kepada *backend* melalui *endpoint* REST API.

Struktur proyek *backend* disusun secara modular sehingga setiap komponen memiliki tanggung jawab yang jelas dan mudah dipelihara. Struktur direktori *backend* ditunjukkan pada Gambar 4.1.

Gambar 4.1 Struktur Direktori Backend Sistem

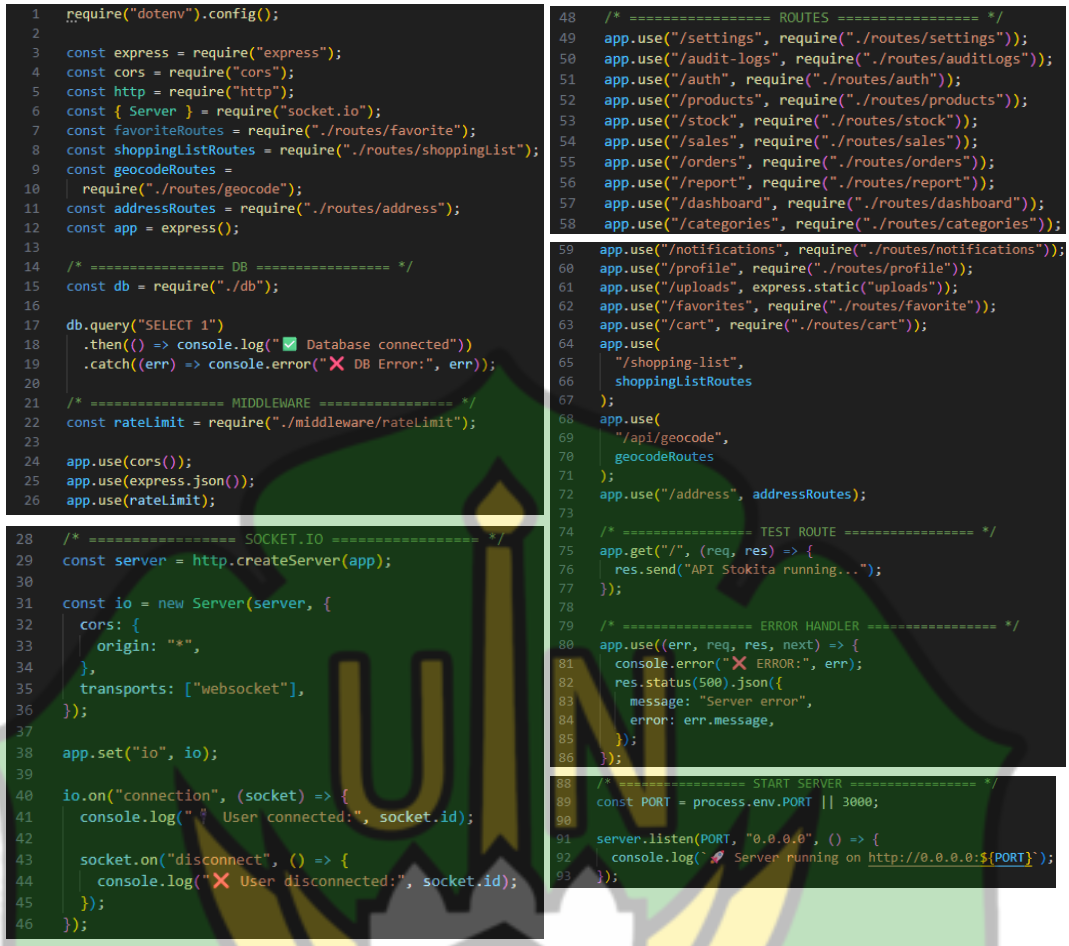
Berdasarkan Gambar 4.1, proyek *backend* terdiri atas beberapa folder utama yang memiliki fungsi berbeda. Folder routes berisi seluruh *endpoint* REST API yang mengelola berbagai fitur aplikasi, seperti autentikasi pengguna, produk, kategori, stok barang, penjualan, pesanan, laporan penjualan, profil pengguna, alamat pengguna, keranjang belanja, daftar favorit, daftar belanja, notifikasi, dashboard, audit log, pengaturan aplikasi, serta layanan geocoding.

Folder *middleware* digunakan untuk menyimpan *middleware* yang menangani autentikasi menggunakan JSON Web Token (JWT), otorisasi hak akses admin, pembatasan jumlah permintaan (*rate limiting*), serta proses upload berkas.

Folder utils berisi berbagai fungsi pendukung yang digunakan oleh beberapa modul sistem, seperti integrasi Cloudinary, *Firebase Cloud Messaging*, Brevo Email API, pemeriksaan stok produk, pengiriman notifikasi, pembuatan audit log, serta template email.

Selain itu terdapat beberapa berkas utama seperti app.js sebagai *entry point* aplikasi, db.js sebagai konfigurasi koneksi basis data menggunakan *connection pool*, berkas .env sebagai penyimpan konfigurasi lingkungan, serta serviceAccountKey.json yang digunakan untuk autentikasi layanan Firebase Admin SDK.

Inisialisasi *backend* dilakukan pada berkas app.js yang berfungsi sebagai titik awal (*entry point*) aplikasi. Tahap ini meliputi pemuatan konfigurasi lingkungan, inisialisasi koneksi basis data, pendaftaran *middleware*, konfigurasi komunikasi *real-time* menggunakan Socket.IO, registrasi seluruh *endpoint* API, serta menjalankan server.



Gambar 4.2 Implementasi Inisialisasi Server

Berdasarkan Gambar 4.2, proses inisialisasi diawali dengan pemanggilan konfigurasi lingkungan menggunakan package dotenv. Selanjutnya sistem membuat instance aplikasi Express melalui fungsi express() dan menginisialisasi koneksi basis data menggunakan modul db.js.

Setelah koneksi berhasil dibuat, sistem mendaftarkan beberapa *middleware*, antara lain cors untuk mengatur akses lintas domain, express.json() untuk memproses data berformat JSON, serta *middleware rate limiting* untuk membatasi jumlah permintaan yang diterima server dalam periode tertentu.

Backend kemudian membuat HTTP Server menggunakan modul http dan mengintegrasikan Socket.IO sehingga aplikasi dapat mendukung komunikasi *real-time*, seperti pengiriman notifikasi kepada pengguna tanpa harus melakukan penyegaran (*refresh*) aplikasi secara manual.

Selanjutnya sistem mendaftarkan seluruh *endpoint* REST API yang menangani berbagai fitur aplikasi, antara lain autentikasi, produk, kategori, stok barang, penjualan, pesanan, laporan, dashboard, profil pengguna, alamat pengguna, keranjang belanja, daftar favorit, daftar belanja, notifikasi, geocoding, pengaturan aplikasi, serta audit log.

Pada bagian akhir, *backend* menjalankan server menggunakan fungsi server.listen() sehingga aplikasi siap menerima permintaan (*request*) dari aplikasi Android melalui REST API maupun komunikasi *real-time* menggunakan Socket.IO.

4.1.2 Implementasi Basis Data

Implementasi basis data pada aplikasi Stokita menggunakan sistem manajemen basis data MySQL yang dihosting pada layanan Aiven MySQL. Basis data berfungsi sebagai media penyimpanan seluruh data aplikasi, mulai dari data pengguna, produk, kategori, transaksi penjualan, pemesanan online, hingga data pendukung lainnya. Seluruh tabel dibangun berdasarkan hasil perancangan *Entity Relationship Diagram* (ERD) pada Bab III sehingga setiap entitas saling terhubung melalui penggunaan *Primary Key* dan *Foreign Key*.

Struktur basis data dirancang menggunakan model relasional sehingga mampu menjaga integritas data, meminimalkan redundansi, serta mendukung proses pengelolaan data secara konsisten. Berdasarkan implementasi yang telah dilakukan, basis data aplikasi Stokita terdiri atas tabel utama users, user_addresses, categories, products, orders, order_items, sales, sale_items, serta beberapa tabel pendukung seperti stock_in, cart, favorites, shopping_list, notifications, dan audit_logs.

Berikut merupakan implementasi beberapa tabel utama yang digunakan dalam aplikasi.

1) Implementasi Tabel Users

Tabel users digunakan untuk menyimpan data akun seluruh pengguna aplikasi, baik admin maupun pelanggan.

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	users	INT	binary	11
2	name	stokita	users	VARCHAR	utf8mb4	100
3	email	stokita	users	VARCHAR	utf8mb4	100
4	email_verified	stokita	users	TINYINT	binary	1
5	verification_token	stokita	users	VARCHAR	utf8mb4	255
6	verification_expired	stokita	users	DATETIME	binary	19
7	password	stokita	users	VARCHAR	utf8mb4	255
8	role	stokita	users	ENUM	utf8mb4	9
9	created_at	stokita	users	TIMESTAMP	binary	19
10	deleted_at	stokita	users	DATETIME	binary	19
11	fcm_token	stokita	users	TEXT	utf8mb4	65535
12	photo	stokita	users	VARCHAR	utf8mb4	255
13	photo_public_id	stokita	users	VARCHAR	utf8mb4	255
14	phone	stokita	users	VARCHAR	utf8mb4	20

Gambar 4.3 Struktur Tabel Users

Berdasarkan Gambar 4.3, tabel users memiliki kolom id sebagai *Primary Key* yang digunakan sebagai identitas unik setiap pengguna. Tabel ini menyimpan informasi identitas pengguna seperti nama, email, nomor telepon, kata sandi (password), peran (role), foto profil, serta informasi pendukung autentikasi.

Password pengguna disimpan dalam bentuk hash sehingga keamanan akun tetap terjaga. Selain itu, tabel ini juga mengimplementasikan mekanisme verifikasi email melalui kolom email_verified, verification_token, dan verification_expired. Untuk mendukung pengiriman notifikasi menggunakan *Firebase Cloud Messaging*, sistem menyimpan fcm_token, sedangkan photo_public_id digunakan untuk menyimpan public ID gambar pada layanan Cloudinary.

2) Implementasi Tabel Categories

Tabel categories digunakan untuk menyimpan data kategori produk yang tersedia pada Fahri Mart.

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	categories	INT	binary	11
2	name	stokita	categories	VARCHAR	utf8mb4	100
3	deleted_at	stokita	categories	DATETIME	binary	19
4	image	stokita	categories	VARCHAR	utf8mb4	255

Gambar 4.4 Struktur Tabel Categories

Setiap kategori memiliki identitas unik berupa id, nama kategori, gambar kategori, serta informasi waktu penghapusan (*soft delete*). Kolom id digunakan

sebagai *Primary Key* yang selanjutnya direferensikan oleh tabel products melalui kolom category_id.

3) Implementasi Tabel Products

Tabel products digunakan untuk menyimpan seluruh informasi produk yang dijual pada Fahri Mart.

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	products	INT	binary	11
2	name	stokita	products	VARCHAR	utf8mb4	150
3	category_id	stokita	products	INT	binary	11
4	barcode	stokita	products	VARCHAR	utf8mb4	50
5	price	stokita	products	INT	binary	11
6	stock	stokita	products	INT	binary	11
7	created_at	stokita	products	TIMESTAMP	binary	19
8	deleted_at	stokita	products	DATETIME	binary	19
9	image	stokita	products	VARCHAR	utf8mb4	255
10	cost_price	stokita	products	DECIMAL	binary	12

Gambar 4.5 Struktur Tabel Products

Berdasarkan Gambar 4.5, tabel products menyimpan informasi berupa nama produk, barcode, harga jual, harga modal (*cost price*), jumlah stok, gambar produk, serta kategori produk. Relasi dengan tabel categories diterapkan melalui *Foreign key* category_id, sehingga setiap produk hanya berada pada satu kategori.

Informasi harga modal digunakan sebagai dasar perhitungan laba pada laporan penjualan, sedangkan data stok akan diperbarui secara otomatis ketika terjadi transaksi penjualan maupun proses checkout pemesanan online.

4) Implementasi Tabel Orders dan Order Items

Proses pemesanan online diimplementasikan menggunakan dua tabel yang saling berhubungan, yaitu orders dan order_items.

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	orders	INT	binary	11
2	user_id	stokita	orders	INT	binary	11
3	total	stokita	orders	INT	binary	11
4	status	stokita	orders	ENUM	utf8mb4	8
5	created_at	stokita	orders	TIMESTAMP	binary	19
6	note	stokita	orders	TEXT	utf8mb4	65535
7	completed_at	stokita	orders	DATETIME	binary	19
8	shipping_cost	stokita	orders	INT	binary	11
9	receiver_name	stokita	orders	VARCHAR	utf8mb4	100
10	receiver_phone	stokita	orders	VARCHAR	utf8mb4	20
11	delivery_address	stokita	orders	TEXT	utf8mb4	65535
12	delivery_latitude	stokita	orders	DECIMAL	binary	12
13	delivery_longitude	stokita	orders	DECIMAL	binary	13

Gambar 4.6 Struktur Tabel Orders

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	order_items	INT	binary	11
2	order_id	stokita	order_items	INT	binary	11
3	product_id	stokita	order_items	INT	binary	11
4	quantity	stokita	order_items	INT	binary	11
5	price	stokita	order_items	INT	binary	11
6	is_checked	stokita	order_items	TINYINT	binary	1
7	cost_price	stokita	order_items	INT	binary	11

Gambar 4.7 Struktur Tabel Order_Items

Tabel orders digunakan untuk menyimpan informasi utama setiap transaksi pemesanan pelanggan, seperti identitas pelanggan, total transaksi, status pesanan, biaya pengiriman, catatan pesanan, serta waktu transaksi.

Selain itu, tabel ini juga menyimpan snapshot alamat pengiriman berupa nama penerima, nomor telepon, alamat pengiriman, serta koordinat lokasi (latitude dan longitude). Penyimpanan snapshot alamat dilakukan agar data alamat pada pesanan tetap tersimpan meskipun pengguna mengubah alamat utama pada kemudian hari.

Sementara itu, tabel order_items digunakan untuk menyimpan rincian setiap produk yang terdapat pada suatu pesanan, meliputi produk yang dibeli, jumlah pembelian, harga jual, harga modal, serta status pemeriksaan item. Hubungan antara kedua tabel menerapkan relasi One-to-Many, yaitu satu pesanan dapat terdiri atas lebih dari satu produk.

5) Implementasi Tabel Sales dan Sale Items

Tabel sales dan sale_items digunakan untuk menyimpan transaksi penjualan offline yang dilakukan melalui fitur kasir pada aplikasi Stokita.

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	sales	INT	binary	11
2	admin_id	stokita	sales	INT	binary	11
3	total	stokita	sales	INT	binary	11
4	created_at	stokita	sales	TIMESTAMP	binary	19
5	order_id	stokita	sales	INT	binary	11

Gambar 4.8 Struktur Tabel Sales

	Field	Schema	Table	Type	Character Set	Display Size
1	id	stokita	sale_items	INT	binary	11
2	sale_id	stokita	sale_items	INT	binary	11
3	product_id	stokita	sale_items	INT	binary	11
4	quantity	stokita	sale_items	INT	binary	11
5	price	stokita	sale_items	INT	binary	11
6	cost_price	stokita	sale_items	INT	binary	11
7	created_at	stokita	sale_items	DATETIME	binary	19

Gambar 4.9 Struktur Tabel Sale_Items

Tabel sales menyimpan informasi utama transaksi, seperti admin yang melakukan transaksi, total pembayaran, waktu transaksi, serta referensi terhadap pesanan online melalui kolom order_id apabila transaksi berasal dari pemesanan pelanggan.

Sementara itu, tabel sale_items menyimpan rincian setiap produk yang terjual, meliputi produk, jumlah pembelian, harga jual, harga modal, serta waktu pencatatan transaksi. Data pada kedua tabel digunakan sebagai sumber informasi dalam proses penyusunan laporan penjualan dan analisis keuntungan.

6) Implementasi Tabel Pendukung

Selain tabel utama, aplikasi Stokita juga mengimplementasikan beberapa tabel pendukung yang berperan dalam mendukung berbagai fungsi sistem, yaitu user_addresses, stock_in, cart, favorites, shopping_list, notifications, dan audit_logs.

Tabel user_addresses digunakan untuk menyimpan lebih dari satu alamat milik pengguna beserta label alamat, nama penerima, nomor telepon, koordinat lokasi, serta penanda alamat utama (*primary address*). Tabel stock_in digunakan untuk mencatat riwayat penambahan stok barang.

Selanjutnya, tabel cart berfungsi menyimpan daftar produk yang dipilih pelanggan sebelum melakukan proses checkout. Tabel favorites digunakan untuk menyimpan produk favorit pengguna, sedangkan tabel shopping_list digunakan sebagai daftar belanja yang dapat dibuat oleh pelanggan. Lalu tabel notifications digunakan untuk menyimpan riwayat notifikasi yang dikirimkan kepada pengguna.

Implementasi seluruh tabel tersebut menghasilkan struktur basis data yang saling terintegrasi sehingga mampu mendukung proses autentikasi pengguna, pengelolaan produk dan stok, transaksi penjualan offline, pemesanan online, hingga penyajian laporan penjualan secara konsisten.

4.1.3 Implementasi Logika Bisnis dan API

Implementasi logika bisnis pada aplikasi Stokita dibangun pada sisi *backend* menggunakan Node.js dengan *framework* Express.js. Seluruh fungsi utama aplikasi diimplementasikan dalam bentuk layanan RESTful API yang berfungsi sebagai media komunikasi antara aplikasi Android dan server. Setiap permintaan (*request*) yang dikirimkan oleh aplikasi diproses oleh *backend* sesuai dengan aturan bisnis yang telah ditentukan, kemudian hasilnya dikembalikan dalam format JSON.

Logika bisnis pada *backend* diimplementasikan ke dalam beberapa modul utama, meliputi autentikasi pengguna, pengelolaan profil dan alamat pengguna, kategori produk, produk, stok barang, keranjang belanja, daftar favorit, daftar belanja, transaksi penjualan offline, pemesanan online, laporan penjualan, notifikasi, serta pengelolaan hak akses pengguna. Setiap modul menerapkan proses validasi data, autentikasi, serta otorisasi sesuai dengan kebutuhan sistem.

Daftar *endpoint* API yang telah diimplementasikan ditunjukkan pada Tabel 4.1.

Tabel 4.1 Daftar Implementasi Endpoint API Sistem

No	Modul	Metode	*Endpoint* (URL)	Deskripsi
1	Autentikasi	POST	/auth/register	Registrasi akun pengguna baru
		POST	/auth/login	Login pengguna dan menghasilkan JWT Token
2	Profil Pengguna	GET	/profile	Menampilkan data profil pengguna
		PUT	/profile	Memperbarui data profil pengguna
		POST	/profile/upload	Mengupload foto profil
3	Alamat Pengguna	POST	/address	Menambahkan alamat pengguna
		GET	/address	Menampilkan daftar alamat pengguna

No	Modul	Metode	*Endpoint* (URL)	Deskripsi
		PUT	/address/:id	Memperbarui alamat pengguna
		DELETE	/address/:d	Menghapus alamat pengguna
4	Produk	POST	/products	Menambahkan produk baru
		GET	/products	Menampilkan seluruh produk
		PUT	/products/:id	Mengedit data produk
		DELETE	/products/:id	Menghapus produk
		GET	/products/barcode/:code	Mencari produk berdasarkan barcode
5	Stok Barang	POST	/stock	Menambahkan stok barang
6	Barcode	GET	/products/barcode/:code	Mencari produk berdasarkan barcode
7	Favorit	POST	/favorite/toggle/:productId	Menambah atau menghapus produk dari daftar favorit
		GET	/favorite	Menampilkan daftar produk favorit
		DELETE	/favorite/:productId	Menghapus produk dari daftar favorit
8	Daftar Belanja	POST	/shoppingList	Menambahkan produk ke daftar belanja
		GET	/shoppingList	Menampilkan daftar belanja

No	Modul	Metode	*Endpoint* (URL)	Deskripsi
		PUT	/shoppingList/:id	Memperbarui data daftar belanja
		DELETE	/shoppingList/:id	Menghapus produk dari daftar belanja
		POST	/shoppingList/add-all-to-cart	Memindahkan seluruh daftar belanja ke keranjang
9	Keranjang Belanja	POST	/cart	Menambahkan produk ke keranjang
		GET	/cart	Menampilkan isi keranjang
		PUT	/cart/:id	Mengupdate isi keranjang
		DELETE	/cart/:id	Menghapus produk dari keranjang
10	Pesanan Online	POST	/orders	Membuat pesanan baru
		GET	/orders	Menampilkan daftar pesanan
		PUT	/orders/:id/status	Memperbarui status pesanan
11	Penjualan Offline	POST	/sales	Menyimpan transaksi penjualan offline
		GET	/sales	Menampilkan data transaksi penjualan
12	Laporan Penjualan	GET	/report/daily	Menampilkan laporan harian
		GET	/report/monthly	Menampilkan laporan bulanan
		GET	/report/best-seller	Menampilkan produk terlaris

No	Modul	Metode	*Endpoint* (URL)	Deskripsi
13	Notifikasi	GET	/notifications	Menampilkan daftar notifikasi pengguna
		PUT	/notifications/read-all	Menandai seluruh notifikasi sebagai telah dibaca

Selain menyediakan layanan RESTful API, *backend* juga menerapkan beberapa aturan logika bisnis untuk memastikan keamanan data dan konsistensi proses bisnis aplikasi.

1) Logika Autentikasi dan Verifikasi Pengguna

Autentikasi pengguna diimplementasikan menggunakan JSON Web Token (JWT). Setelah pengguna berhasil melakukan proses login, sistem menghasilkan token autentikasi yang harus dikirimkan pada setiap permintaan API melalui *Authorization Header* dengan format Bearer Token. Apabila token tidak tersedia atau tidak valid, sistem akan mengembalikan respons *401 Unauthorized*.

Selain autentikasi, sistem juga menerapkan proses verifikasi email menggunakan layanan Brevo Email API. Setelah proses registrasi berhasil, sistem mengirimkan email verifikasi kepada pengguna. Akun hanya dapat digunakan setelah proses verifikasi email berhasil dilakukan.

2) Logika Hak Akses Pengguna

Aplikasi menerapkan mekanisme *Role-Based Access Control* (RBAC) untuk membatasi hak akses setiap pengguna berdasarkan perannya. Pengguna dengan peran admin memiliki hak untuk mengelola kategori produk, produk, stok barang, transaksi penjualan offline, pesanan pelanggan, laporan penjualan, serta dashboard administrasi. Sementara itu, pengguna dengan peran pelanggan hanya dapat mengakses fitur yang berkaitan dengan pencarian produk, keranjang belanja, daftar favorit, daftar belanja, pemesanan online, pengelolaan alamat, serta profil pengguna. Apabila pengguna mencoba mengakses *endpoint*

yang tidak sesuai dengan hak aksesnya, sistem akan mengembalikan respons *403 Forbidden*.

3) Logika Pengelolaan Stok Barang

Sistem menerapkan mekanisme pembaruan stok secara otomatis untuk menjaga kesesuaian antara jumlah stok pada basis data dengan kondisi barang yang tersedia. Penambahan stok dilakukan melalui modul Stock In, sedangkan pengurangan stok dilakukan secara otomatis ketika pelanggan berhasil melakukan proses checkout pada pemesanan online maupun ketika transaksi penjualan offline berhasil disimpan. Sebelum pesanan diproses, sistem terlebih dahulu memvalidasi ketersediaan stok untuk memastikan jumlah produk yang dipesan tidak melebihi stok yang tersedia.

4) Logika Pemesanan Online

Pada proses pemesanan online, pelanggan memilih produk yang akan dibeli melalui keranjang belanja kemudian menentukan alamat pengiriman. Sistem selanjutnya memvalidasi stok setiap produk dan menyimpan snapshot alamat pengiriman, yang meliputi nama penerima, nomor telepon, alamat lengkap, serta koordinat lokasi.

Apabila seluruh data valid, sistem secara otomatis membuat data pesanan dengan status awal Diproses dan mengurangi stok produk sesuai jumlah yang dipesan. Selanjutnya admin dapat memperbarui status pesanan menjadi Dikirim dan Selesai setelah menchecklist semua item pesanan pelanggan.

5) Logika Penyimpanan Media

Aplikasi Stokita mengimplementasikan penyimpanan gambar menggunakan layanan Cloudinary. Ketika admin menambahkan atau memperbarui data produk, kategori, maupun foto profil pengguna, berkas gambar terlebih dahulu diunggah ke Cloudinary. Selanjutnya sistem menyimpan URL gambar beserta public ID pada basis data sehingga gambar dapat ditampilkan kembali tanpa membebani penyimpanan server.

6) Logika Notifikasi

Sistem mengimplementasikan layanan notifikasi menggunakan *Firebase Cloud Messaging* (FCM). Token perangkat pengguna disimpan pada basis data sehingga *backend* dapat mengirimkan notifikasi ketika terjadi perubahan status pesanan atau aktivitas penting lainnya. Selain dikirimkan ke perangkat pengguna, riwayat notifikasi juga disimpan pada tabel notifications sehingga dapat ditampilkan kembali pada aplikasi.

Dengan implementasi logika bisnis tersebut, aplikasi Stokita mampu mengelola proses autentikasi pengguna, pengelolaan produk, transaksi penjualan offline, pemesanan online, penyimpanan media, pengiriman notifikasi, serta pencatatan aktivitas sistem secara terintegrasi melalui layanan RESTful API. Pemisahan logika bisnis pada sisi *backend* juga meningkatkan keamanan, mempermudah proses pengembangan dan pemeliharaan aplikasi, serta memungkinkan aplikasi untuk dikembangkan lebih lanjut pada berbagai platform yang mendukung komunikasi berbasis API.

4.1.4 Implementasi Antarmuka Pengguna

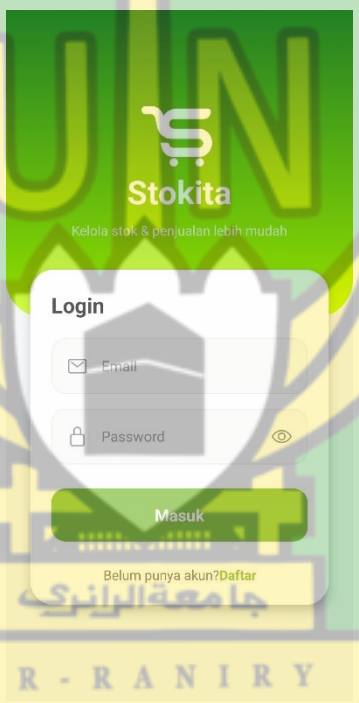
Implementasi antarmuka pengguna (*User Interface*) merupakan tahap realisasi dari rancangan antarmuka yang telah disusun pada Bab III. Aplikasi Stokita dikembangkan menggunakan React Native sehingga dapat dijalankan pada perangkat Android. Antarmuka dirancang dengan konsep modern, responsif, dan mudah digunakan (*user friendly*) agar mampu memberikan pengalaman penggunaan yang baik bagi admin maupun pelanggan.

Implementasi antarmuka mengacu pada prinsip *usability*, yaitu kemudahan navigasi, konsistensi tampilan, keterbacaan informasi, serta efisiensi dalam menyelesaikan tugas pengguna. Seluruh halaman aplikasi telah terintegrasi dengan layanan RESTful API sehingga data yang ditampilkan selalu diperbarui berdasarkan isi basis data.

Berikut merupakan implementasi beberapa halaman utama pada aplikasi Stokita:

a) Implementasi Halaman Login

Implementasi halaman Login ditunjukkan pada Gambar 4.10. Halaman ini merupakan gerbang autentikasi pengguna sebelum dapat mengakses aplikasi. Pengguna diminta memasukkan alamat email dan kata sandi yang telah terdaftar. Setelah tombol Masuk ditekan, aplikasi mengirimkan permintaan autentikasi ke *backend* melalui *endpoint* /auth/login. Apabila data yang dimasukkan valid, sistem menghasilkan JSON Web Token (JWT) yang digunakan sebagai autentikasi pada setiap permintaan API, kemudian mengarahkan pengguna ke halaman utama sesuai dengan peran pengguna (admin atau pelanggan). Pengguna yang belum melakukan verifikasi email tidak dapat menggunakan aplikasi hingga proses verifikasi berhasil dilakukan.

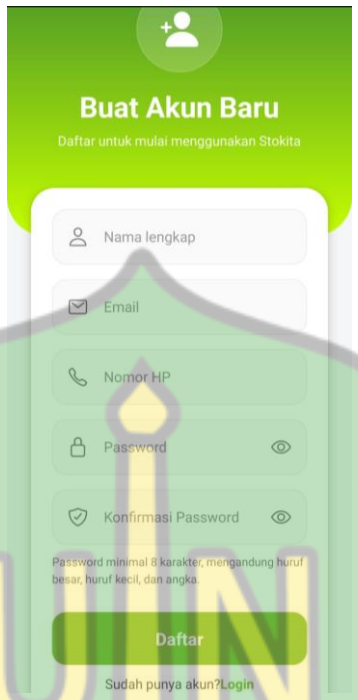


Gambar 4.10 Halaman Login

b) Implementasi Halaman Register

Implementasi halaman Register ditunjukkan pada Gambar 4.11. Halaman ini digunakan oleh pelanggan untuk membuat akun baru dengan mengisi data berupa nama, email, nomor telepon, dan kata sandi. Setelah proses registrasi berhasil, sistem menyimpan data pengguna ke dalam basis data kemudian mengirimkan email verifikasi menggunakan layanan Brevo Email API.

Pengguna dapat melakukan login setelah akun berhasil diverifikasi melalui tautan yang dikirimkan ke alamat email.

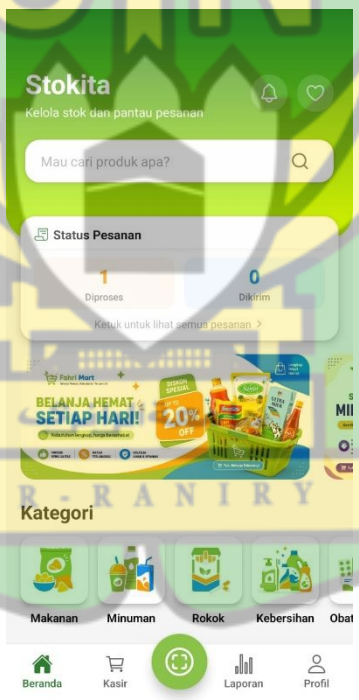


Gambar 4.11 Halaman Register

c) Implementasi Dashboard Admin dan Pelanggan

Implementasi Dashboard Admin dan Pelanggan ditampilkan pada Gambar 4.12 dan 4.13. Bagi Pelanggan, halaman ini berfungsi sebagai pusat navigasi utama aplikasi. Antarmuka menampilkan kolom pencarian produk, kategori produk, produk terlaris (*best seller*), serta daftar produk yang tersedia. Selain itu, pengguna juga dapat menambahkan produk ke keranjang belanja secara langsung melalui tombol yang tersedia pada setiap produk. Bagi Admin, halaman ini merupakan pusat pengelolaan aplikasi. Dashboard menampilkan ringkasan informasi seperti jumlah pesanan masuk, stok barang, serta akses cepat menuju fitur pengelolaan produk, stok barang, transaksi penjualan, dan laporan penjualan.

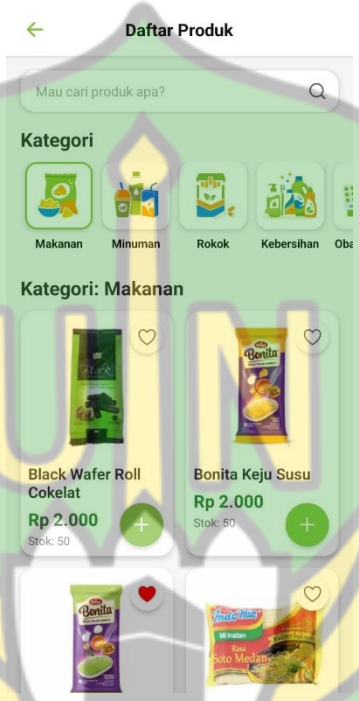
Gambar 4.12 Dashboard Pelanggan



Gambar 4.13 Dashboard Admin

d) Implementasi Halaman Daftar Produk

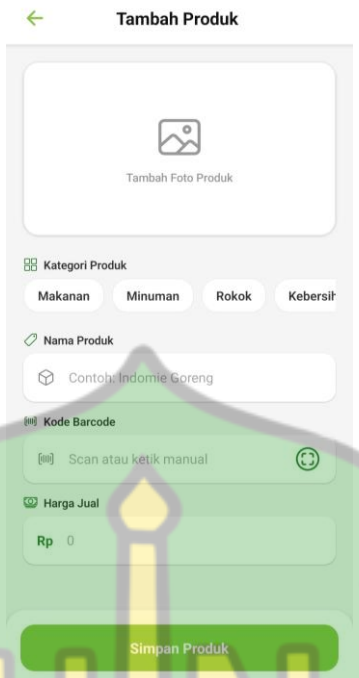
Implementasi halaman Produk ditunjukkan pada Gambar 4.14. Halaman ini menampilkan seluruh produk yang tersedia beserta informasi nama produk, harga, stok, kategori, dan gambar produk. Pengguna dapat mencari produk menggunakan fitur pencarian maupun memfilter produk berdasarkan kategori yang dipilih.



Gambar 4.14 Halaman Daftar Produk

e) Implementasi Halaman Tambah Produk

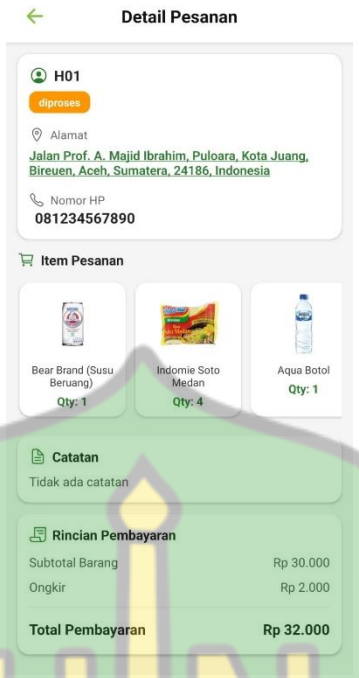
Implementasi halaman Kelola Stok Barang ditunjukkan pada Gambar 4.15. Halaman ini digunakan untuk menambah produk yang belum tersedia di aplikasi. Admin menambah produk dengan mengisi kolom-kolom rincian produk, seperti nama, kategori, dan harga jual. Adapun kode barcode terisi otomatis setelah barcode discan.



Gambar 4.15 Halaman Tambah Produk

f) Implementasi Halaman Detail Pesanan

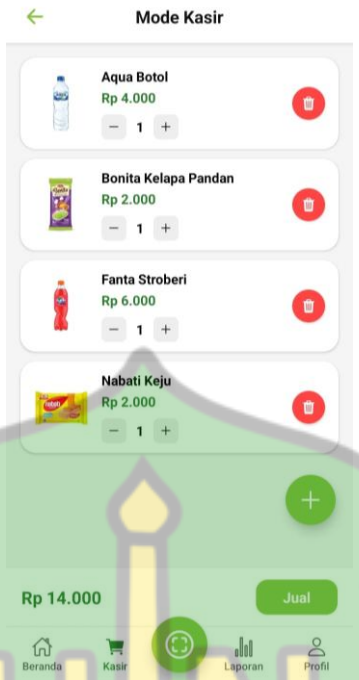
Implementasi halaman Detail Pesanan ditunjukkan pada Gambar 4.16. Halaman ini menampilkan informasi lengkap mengenai pesanan pelanggan, meliputi nama penerima, nomor telepon, alamat pengiriman, daftar produk yang dipesan, catatan pesanan, biaya pengiriman, total pembayaran, serta status pesanan. Status pesanan ditampilkan secara bertahap mulai dari Diproses, Dikirim, hingga Selesai, sehingga pelanggan dapat memantau perkembangan pesanan secara langsung.



Gambar 4.16 Halaman Detail Pesanan

g) Implementasi Halaman Kasir

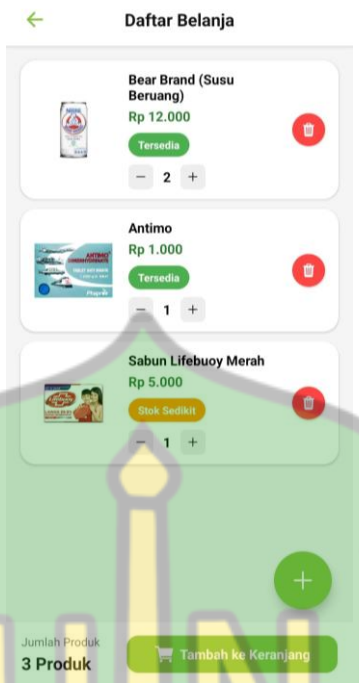
Implementasi halaman Kasir ditunjukkan pada Gambar 4.17. Halaman ini berfungsi sebagai fitur kasir yang digunakan admin untuk mencatat transaksi penjualan secara langsung di toko. Produk dapat ditambahkan melalui pemindaian barcode maupun pemilihan dari daftar produk. Setelah transaksi berhasil disimpan, sistem secara otomatis mencatat data penjualan ke dalam basis data.



Gambar 4.17 Halaman Kasir

h) Implementasi Halaman Daftar Belanja

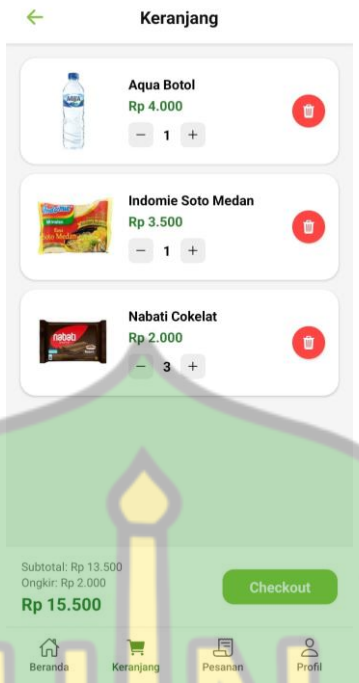
Implementasi halaman Daftar Belanja ditunjukkan pada Gambar 4.18. Halaman ini digunakan oleh pelanggan untuk menyusun daftar produk yang akan dibeli pada kesempatan berikutnya. Pengguna dapat menambahkan produk ke dalam daftar belanja, mengubah jumlah produk, maupun menghapus produk yang sudah tidak diperlukan. Selain itu, aplikasi juga menyediakan fitur untuk memindahkan seluruh produk pada daftar belanja ke keranjang belanja secara otomatis melalui satu kali proses. Fitur ini memudahkan pelanggan dalam merencanakan pembelian tanpa harus menambahkan produk satu per satu ketika akan melakukan checkout.



Gambar 4.18 Halaman Daftar Belanja

i) Implementasi Halaman Keranjang

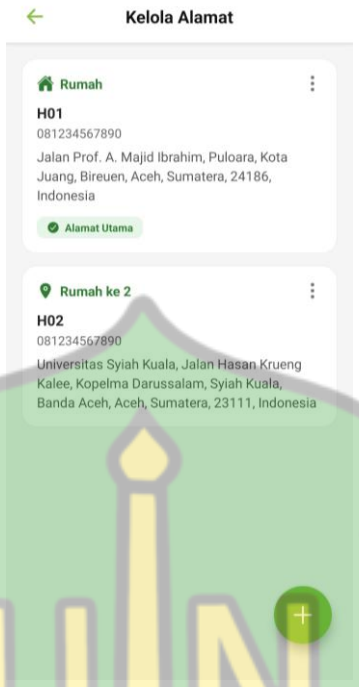
Implementasi halaman Keranjang ditunjukkan pada Gambar 4.19. Halaman ini menampilkan daftar produk yang telah dipilih pelanggan sebelum melakukan checkout. Pengguna dapat mengubah jumlah produk, menghapus produk dari keranjang, melihat total harga transaksi, serta melanjutkan proses checkout.



Gambar 4.19 Halaman Keranjang

j) Implementasi Halaman Kelola Alamat

Implementasi halaman Kelola Alamat ditunjukkan pada Gambar 4.20. Halaman ini digunakan oleh pelanggan untuk mengelola alamat pengiriman yang akan digunakan pada proses pemesanan online. Pengguna dapat menambahkan alamat baru, mengubah informasi alamat, menghapus alamat yang tidak digunakan, serta menentukan alamat utama (*primary address*). Setiap alamat menyimpan informasi berupa label alamat, nama penerima, nomor telepon, alamat lengkap, serta koordinat lokasi (latitude dan longitude). Pada saat proses checkout, pelanggan dapat memilih salah satu alamat yang telah tersimpan sebagai tujuan pengiriman pesanan sehingga proses pemesanan menjadi lebih cepat dan praktis.



Gambar 4.20 Halaman Kelola Alamat

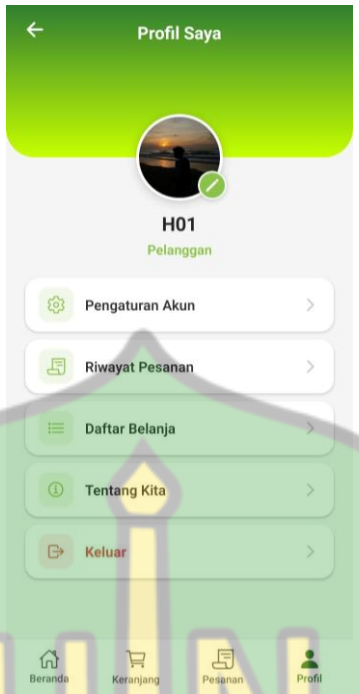
k) Implementasi Halaman Laporan Penjualan

Implementasi halaman Laporan Penjualan ditunjukkan pada Gambar 4.21. Halaman ini digunakan untuk menampilkan laporan penjualan harian, bulanan, dan tahunan. Laporan Penjualan yang ditampilkan berupa total penjualan, total profit, HPP, grafik penjualan, serta grafik profit. Informasi laporan penjualan berasal dari data transaksi yang tersimpan pada basis data sehingga admin dapat memantau perkembangan penjualan dan kondisi usaha secara lebih efektif.

Gambar 4.21 Halaman Laporan Penjualan

l) Implementasi Halaman Profil

Implementasi halaman Profil ditunjukkan pada Gambar 4.22. Halaman ini menampilkan foto profil dan nama pengguna. Pengguna dapat beralih ke pengaturan akun, melihat riwayat penjualan bagi admin, melihat riwayat pesanan bagi pelanggan, melihat informasi tentang aplikasi Stokita, serta beralih ke halaman daftar belanja bagi pelanggan. Selain itu, pengguna dapat memperbarui foto profil maupun melakukan proses *logout* melalui halaman ini.



Gambar 4.22 Halaman Profil

4.2 Pengujian Sistem

Pengujian sistem merupakan tahapan yang dilakukan untuk memastikan bahwa aplikasi Stokita yang telah dikembangkan dapat berfungsi sesuai dengan kebutuhan pengguna dan spesifikasi sistem yang telah dirancang pada Bab III. Pengujian dilakukan setelah seluruh proses implementasi selesai sehingga setiap fungsi aplikasi dapat divalidasi sebelum digunakan oleh admin maupun pelanggan di Fahri Mart.

Sesuai dengan metode penelitian yang telah ditetapkan, pengujian sistem pada penelitian ini dilakukan melalui tiga tahapan, yaitu pengujian RESTful API (API *Testing*), pengujian fungsional aplikasi menggunakan *Black Box Testing*, serta pengujian *usability* menggunakan *System Usability Scale* (SUS).

Pengujian RESTful API bertujuan untuk memastikan bahwa setiap *endpoint* API mampu menerima permintaan (*request*), memproses data sesuai logika bisnis, serta menghasilkan respons (*response*) yang sesuai. Pengujian dilakukan menggunakan aplikasi Postman.

Selanjutnya, pengujian *Black Box Testing* dilakukan pada aplikasi Android untuk memastikan seluruh fitur yang berinteraksi langsung dengan pengguna dapat berjalan sesuai kebutuhan fungsional tanpa memperhatikan struktur kode program.

Tahap terakhir adalah pengujian *System Usability Scale* (SUS) yang melibatkan admin Fahri Mart dan beberapa pelanggan sebagai responden. Pengujian ini bertujuan mengetahui tingkat kemudahan penggunaan aplikasi berdasarkan persepsi pengguna setelah menggunakan aplikasi.

Hasil dari ketiga tahapan pengujian tersebut digunakan sebagai dasar dalam mengevaluasi kualitas, fungsi, serta tingkat kemudahan penggunaan aplikasi Stokita.

4.2.1 Pengujian RESTful API (API Testing)

Pengujian RESTful API dilakukan untuk memastikan bahwa setiap *endpoint* API yang telah diimplementasikan mampu memproses permintaan (*request*) dari aplikasi Android dan menghasilkan respons (*response*) sesuai dengan spesifikasi sistem.

Pengujian dilakukan menggunakan aplikasi Postman dengan mengirimkan berbagai jenis permintaan HTTP ke setiap *endpoint* pada *backend* Node.js. Setiap *endpoint* diuji menggunakan data valid maupun data tidak valid untuk memverifikasi proses validasi data, autentikasi, otorisasi, serta implementasi logika bisnis yang diterapkan pada sistem.

Fokus pengujian meliputi:

1. Autentikasi pengguna menggunakan JSON Web Token (JWT).
2. Pengelolaan profil pengguna.
3. Pengelolaan alamat pengguna.
4. Pengelolaan produk.
5. Pengelolaan stok barang.
6. Pencarian produk menggunakan barcode.
7. Pengelolaan produk favorit.
8. Pengelolaan daftar belanja.
9. Pengelolaan keranjang belanja.

10. Proses checkout dan pemesanan online.
11. Transaksi penjualan offline.
12. Laporan penjualan.
13. Notifikasi pengguna.
14. Keamanan endpoint menggunakan JWT dan Role-Based Access Control.

Suatu pengujian dinyatakan berhasil apabila server mengembalikan HTTP Status Code yang sesuai, respons JSON memiliki struktur yang benar, serta perubahan data pada basis data berlangsung sesuai dengan logika bisnis yang telah ditentukan.

Berikut merupakan hasil pengujian terhadap *endpoint-endpoint* utama RESTful API aplikasi Stokita.

4.2.1.1 Pengujian Register

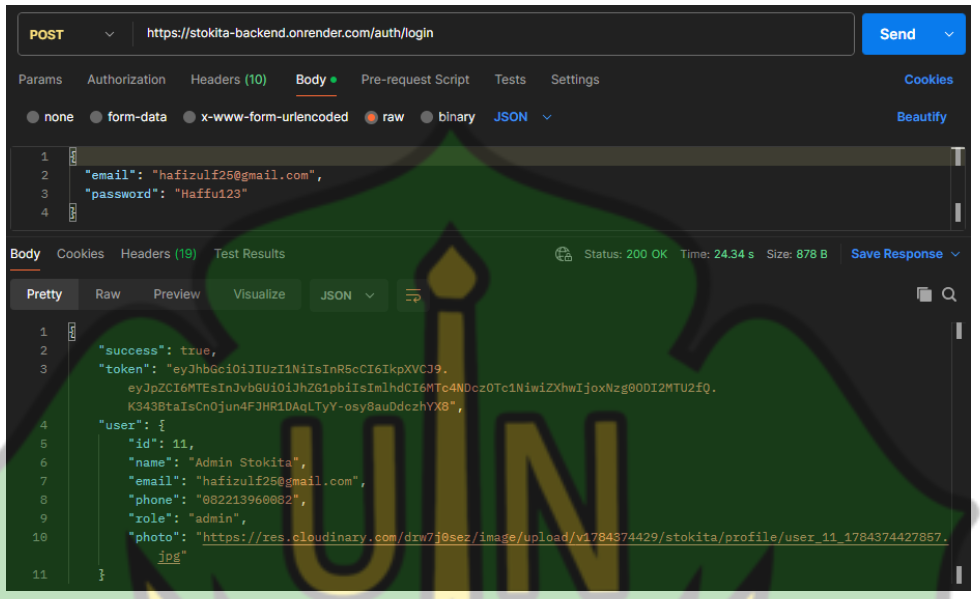
a) Skenario: Daftar akun baru menggunakan data yang valid.
b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK dan sistem mengirim link verifikasi ke email pengguna.

Gambar 4.23 Pengujian Register

4.2.1.2 Pengujian Login

a) Skenario: Login menggunakan email dan password yang valid.

b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK beserta token JWT dan data pengguna.

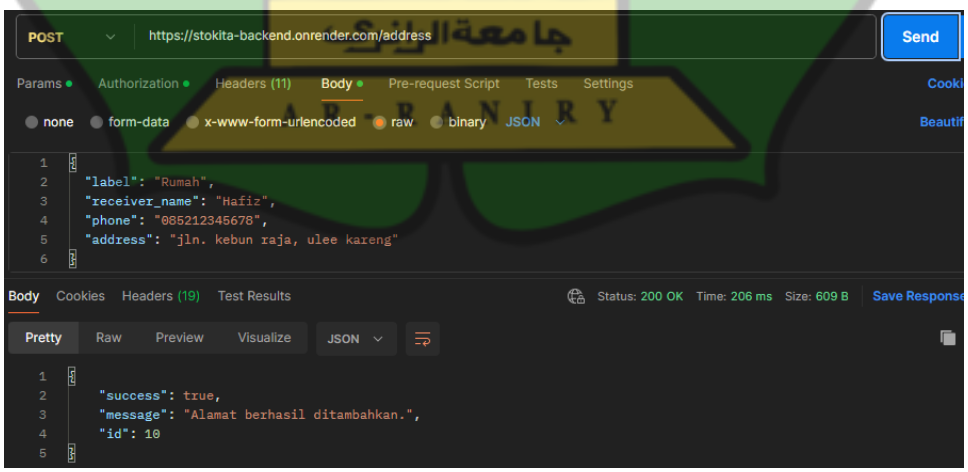


Gambar 4.24 Pengujian Login

4.2.1.3 Pengujian Alamat

a) Skenario: Pelanggan menambahkan alamat baru.

b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK dan alamat baru berhasil ditambahkan.

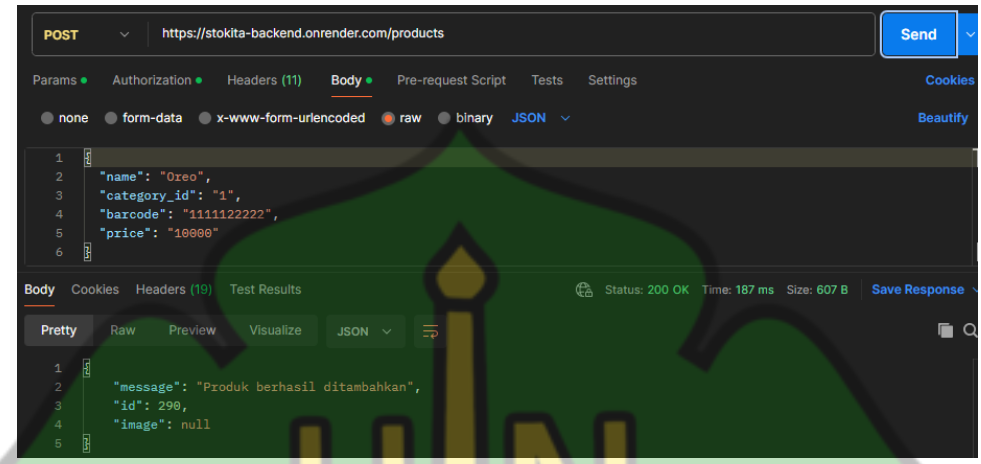


Gambar 4.25 Pengujian Alamat

4.2.1.4 Pengujian Tambah Produk

a) Skenario: Admin menambahkan produk baru.

b) Hasil yang diharapkan: Server mengembalikan status HTTP 200 OK dan data produk berhasil disimpan ke dalam basis data.

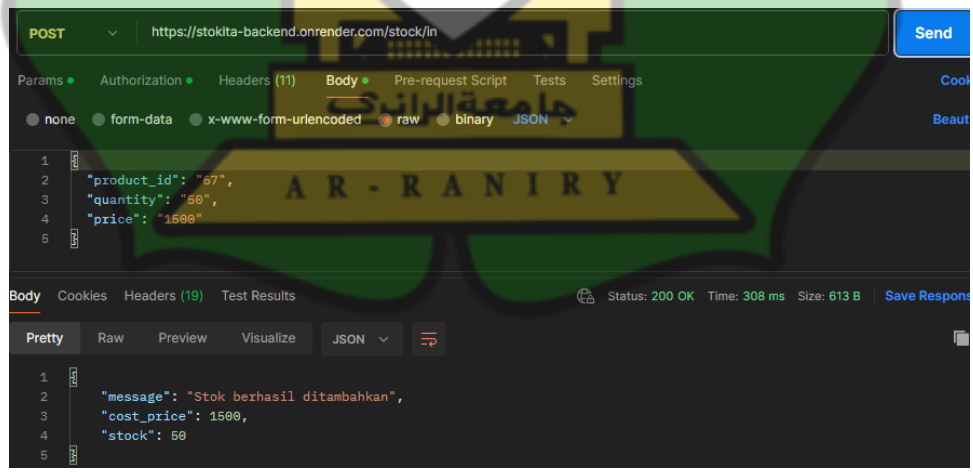


Gambar 4.26 Pengujian Data Produk

4.2.1.5 Pengujian Tambah Stok

a) Skenario: Admin menambahkan stok produk.

b) Hasil yang diharapkan: Server mengembalikan status HTTP 200 OK dan stok produk berhasil ditambahkan.

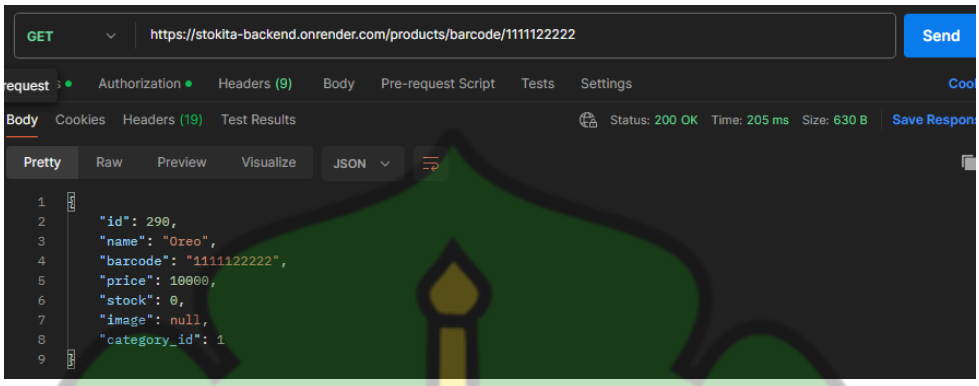


Gambar 4.27 Pengujian Tambah Stok

4.2.1.6 Pengujian Scan Barcode

a) Skenario: Barcode produk yang valid dipindai.

b) Hasil yang diharapkan: Sistem menampilkan informasi produk sesuai barcode.

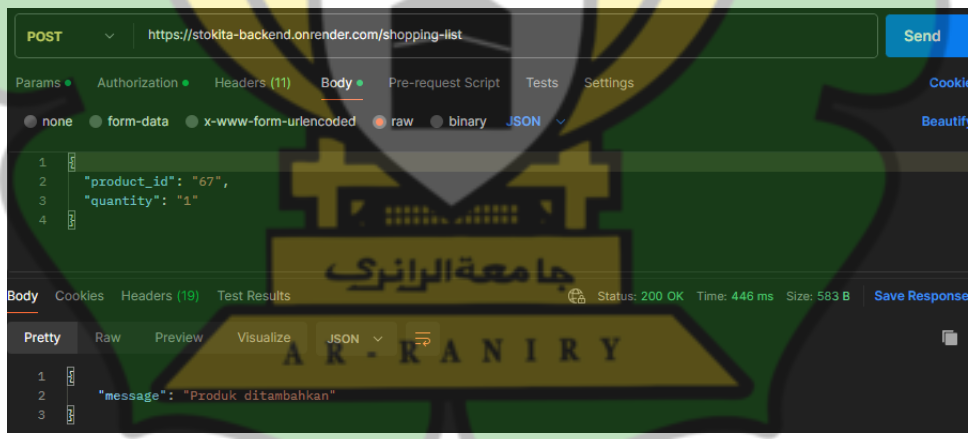


Gambar 4.28 Pengujian Scan Barcode

4.2.1.7 Pengujian Daftar Belanja

a) Skenario: Pelanggan menambahkan produk ke daftar belanja.

b) Hasil yang diharapkan: Server mengembalikan status HTTP 200 OK dan produk berhasil masuk ke daftar belanja.

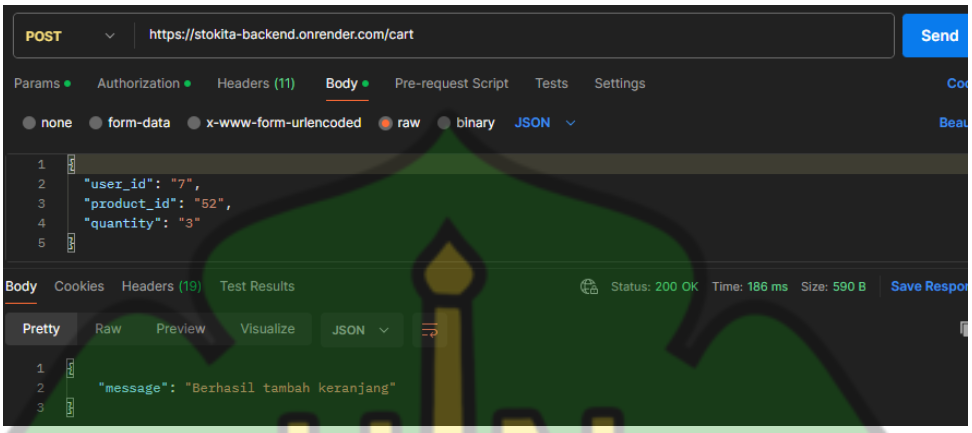


Gambar 4.29 Pengujian Daftar Belanja

4.2.1.8 Pengujian Keranjang Belanja

a) Skenario: Pelanggan menambahkan produk ke keranjang.

b) Hasil yang diharapkan: Server mengembalikan status HTTP 200 OK dan produk berhasil masuk ke keranjang.

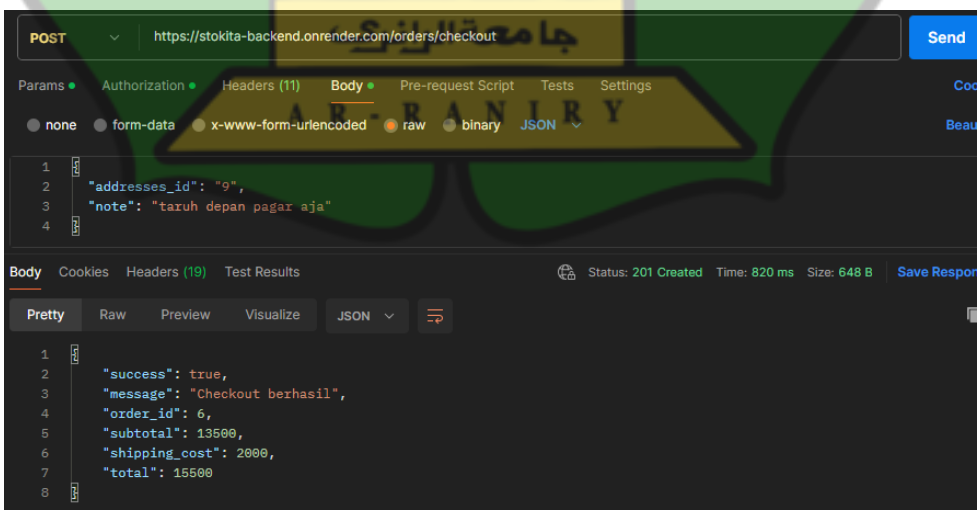


Gambar 4.30 Pengujian Keranjang Belanja

4.2.1.9 Pengujian Checkout Pesanan

a) Skenario: Pelanggan melakukan checkout barang yang sudah dimasukkan ke keranjang.

b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK, data pesanan berhasil dibuat dengan status awal Diproses, snapshot alamat pengiriman berhasil disimpan, dan stok produk otomatis berkurang sesuai jumlah yang dipesan.

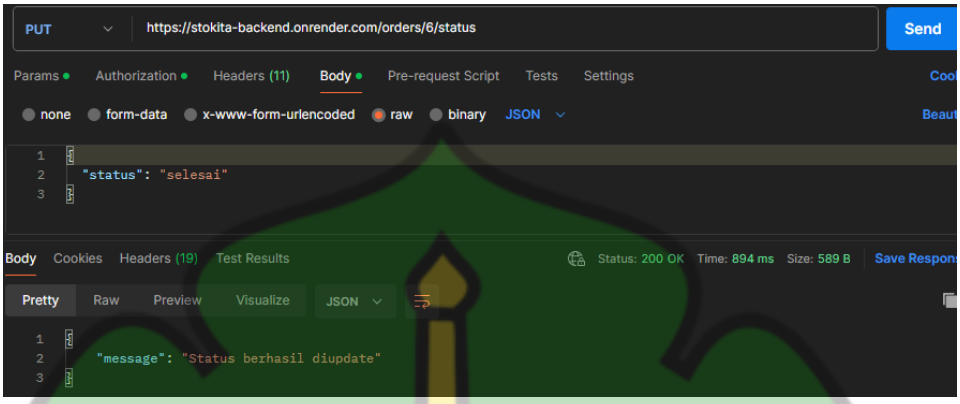


Gambar 4.31 Pengujian Checkout Pesanan

4.2.1.10 Pengujian Update Status Pesanan

a) Skenario: Admin mengubah status pesanan menjadi Selesai.

b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK dan status pesanan berhasil diperbarui menjadi Selesai.



Gambar 4.32 Pengujian Update Status Pesanan

4.2.1.11 Pengujian Penjualan Kasir

a) Skenario: Admin melakukan transaksi penjualan di mode kasir.

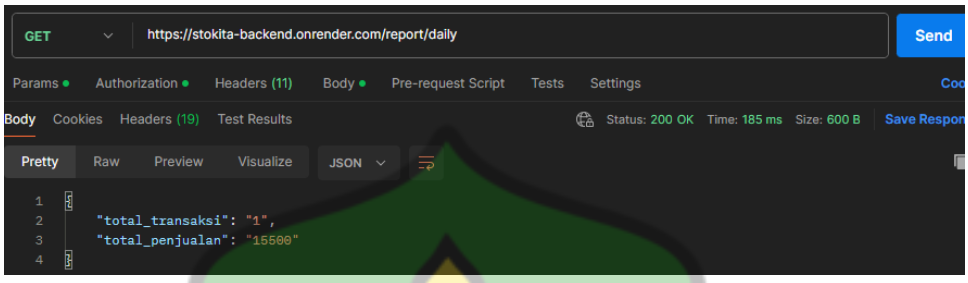
b) Hasil yang diharapkan: Server mengembalikan HTTP 200 OK dan transaksi penjualan berhasil disimpan.

Gambar 4.33 Pengujian Penjualan Kasir

4.2.1.12 Pengujian Laporan Penjualan

a) Skenario: Admin membuka laporan penjualan harian/bulanan.

b) Hasil yang diharapkan: Sistem menampilkan data penjualan sesuai periode yang dipilih.

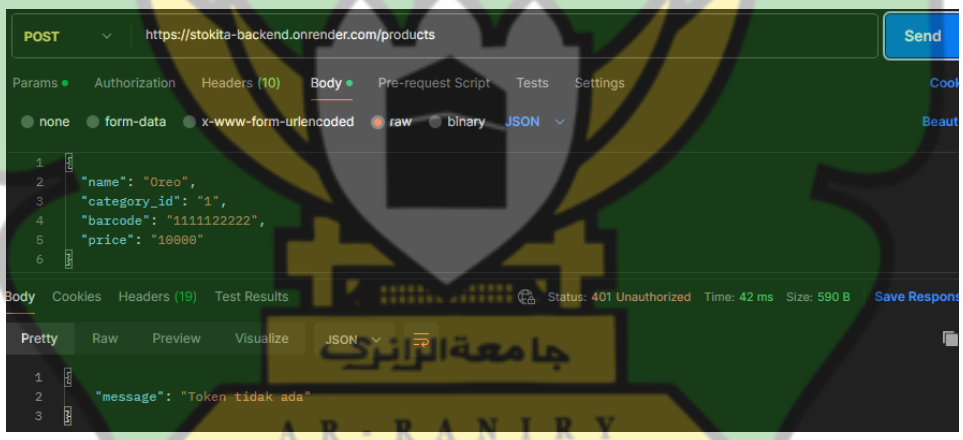


Gambar 4.34 Pengujian Laporan Penjualan

4.2.1.13 Pengujian JWT Security

a) Skenario: Mengakses endpoint tanpa token autentikasi

b) Hasil yang diharapkan: Server mengembalikan 401 Unauthorized dan akses ditolak



Gambar 4.35 Pengujian JWT Security

4.2.1.14 Rekapitulasi Hasil Pengujian Fungsional

Setelah seluruh skenario pengujian API dilakukan menggunakan metode *Black Box Testing*, diperoleh hasil bahwa setiap *endpoint* utama pada aplikasi Stokita mampu berjalan sesuai dengan fungsi yang telah dirancang. Pengujian dilakukan menggunakan aplikasi Postman dengan mengirimkan berbagai variasi permintaan (*request*) ke server serta memverifikasi respons (*response*) yang dihasilkan.

Rekapitulasi hasil pengujian fungsional dapat dilihat pada Tabel 4.2 berikut.

Tabel 4.2 Rekapitulasi Hasil Pengujian API

No	Fitur/Modal	Skenario Pengujian	Hasil Aktual (Respon Server)	Status
1	Login	Login menggunakan email dan password yang valid	Login berhasil, server mengembalikan HTTP 200 beserta token JWT	Valid
2	Login	Login menggunakan password yang salah	Server mengembalikan HTTP *401 Unauthorized*	Valid
3	Profil	Menampilkan data profil pengguna	Data profil berhasil ditampilkan (HTTP *200 OK*)	Valid
4	Alamat	Menambahkan alamat pengiriman	Data alamat berhasil disimpan (HTTP *200 OK*)	Valid
5	Produk	Menambahkan data produk baru	Data produk berhasil disimpan (HTTP *200 OK*)	Valid
6	Produk	Mengubah data produk	Data produk berhasil diubah (HTTP *200 OK*)	Valid
7	Produk	Menghapus produk	Produk berhasil dihapus (HTTP *200 OK*)	Valid
8	Barcode	Mencari produk berdasarkan barcode	Data produk berhasil ditampilkan sesuai barcode (HTTP *200 OK*)	Valid
9	Favorit	Menambahkan produk ke daftar favorit	Produk berhasil ditambahkan ke daftar favorit (HTTP *200 OK*)	Valid
10	Daftar Belanja	Menambahkan produk ke daftar belanja	Produk berhasil ditambahkan ke daftar belanja (HTTP *200 OK*)	Valid
11	Keranjang	Menambahkan produk ke keranjang	Produk berhasil ditambahkan ke keranjang (HTTP *200 OK*)	Valid

No	Fitur/Modal	Skenario Pengujian	Hasil Aktual (Respon Server)	Status
12	Checkout	Pelanggan melakukan checkout pesanan	Pesanan berhasil dibuat dengan status Diproses dan stok produk berkurang sesuai jumlah yang dipesan (HTTP *200 OK*)	Valid
13	Status Pesanan	Admin mengubah status pesanan menjadi Dikirim	Status pesanan berhasil diperbarui (HTTP *200 OK*)	Valid
14	Status Pesanan	Admin mengubah status pesanan menjadi Selesai	Status pesanan berhasil diperbarui menjadi Selesai (HTTP *200 OK*)	Valid
15	Kasir	Admin melakukan penjualan di kasir	Data transaksi berhasil disimpan (HTTP *200 OK*)	Valid
16	Laporan Penjualan	Menampilkan laporan penjualan	Data laporan berhasil ditampilkan sesuai data transaksi (HTTP *200 OK*)	Valid
17	JWT Security	Mengakses *endpoint* tanpa token autentikasi	Server mengembalikan HTTP *401 Unauthorized*	Valid
18	Authorization	Pelanggan mengakses *endpoint* khusus admin	Server mengembalikan HTTP *403 Forbidden*	Valid

Berdasarkan hasil pengujian pada Tabel 4.2, seluruh skenario pengujian RESTful API berhasil dijalankan sesuai dengan hasil yang diharapkan. Setiap *endpoint* mampu memproses permintaan (*request*) dari aplikasi Android dan menghasilkan respons (*response*) yang sesuai dengan logika bisnis yang telah dirancang. Pengujian menunjukkan bahwa proses autentikasi menggunakan JSON Web Token (JWT), pengelolaan profil dan alamat pengguna, pengelolaan produk, keranjang belanja, daftar belanja, transaksi penjualan, pemesanan online, serta penyajian laporan penjualan telah berjalan dengan baik tanpa ditemukan kesalahan

fungsional. Selain itu, mekanisme otorisasi berdasarkan peran pengguna (*Role-Based Access Control*) juga berfungsi dengan baik dengan mengembalikan respons *401 Unauthorized* maupun *403 Forbidden* ketika terjadi pelanggaran hak akses.

Dengan demikian, tingkat keberhasilan pengujian fungsional aplikasi Stokita mencapai 100%, sehingga dapat disimpulkan bahwa seluruh fitur utama aplikasi telah berfungsi sesuai dengan kebutuhan fungsional yang telah ditetapkan dan layak untuk digunakan pada proses pengelolaan stok barang dan penjualan online di Fahri Mart.

4.2.2 Pengujian Fungsional (Black Box Testing)

Pengujian fungsional menggunakan metode *Black Box Testing* dilakukan untuk memastikan bahwa setiap fitur pada aplikasi Android Stokita dapat berjalan sesuai dengan kebutuhan fungsional sistem. Pada metode ini, pengujian dilakukan dengan memberikan berbagai masukan (*input*) pada aplikasi dan mengamati keluaran (*output*) yang dihasilkan tanpa memperhatikan struktur kode program maupun proses internal yang terjadi pada sistem.

Pengujian dilakukan secara langsung pada aplikasi Android yang dibangun menggunakan React Native. Setiap fitur diuji berdasarkan skenario penggunaan yang umum dilakukan oleh admin maupun pelanggan, kemudian dibandingkan dengan hasil yang diharapkan. Suatu pengujian dinyatakan berhasil apabila aplikasi menghasilkan keluaran yang sesuai dengan spesifikasi kebutuhan system.

Fitur yang diuji meliputi proses autentikasi pengguna, pengelolaan produk, keranjang belanja, daftar belanja, pengelolaan alamat, pemesanan online, transaksi penjualan, laporan penjualan, serta pengelolaan profil pengguna.

Tabel 4.3 Black Box Testing Halaman Login

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Email dan password dikosongkan kemudian menekan tombol Login	Email (kosong), Password (kosong)	Sistem menampilkan pesan bahwa email dan password harus diisi.	Sistem menampilkan pesan validasi bahwa email dan password wajib diisi.	Valid

2	Mengisi email tetapi password dikosongkan	Email valid, Password (kosong)	Sistem menampilkan pesan bahwa password harus diisi.	Sistem menampilkan pesan bahwa password wajib diisi.	Valid
3	Mengisi email dan password yang salah	Email valid, Password salah	Sistem menampilkan pesan bahwa email atau password tidak sesuai.	Sistem menampilkan pesan kesalahan login dan proses login dibatalkan.	Valid
4	Mengisi email dan password yang benar	Email valid, Password valid	Sistem berhasil login dan menampilkan halaman utama sesuai hak akses pengguna.	Pengguna berhasil login dan diarahkan ke halaman utama sesuai perannya.	Valid

Tabel 4.4 Black Box Testing Halaman Register

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Seluruh data dikosongkan lalu menekan tombol Daftar	Semua field kosong	Sistem menampilkan validasi bahwa seluruh data wajib diisi.	Sistem menampilkan pesan validasi pada seluruh field yang kosong.	Valid
2	Mengisi data dengan email yang sudah digunakan	Email terdaftar	Sistem menampilkan pesan bahwa email sudah digunakan.	Sistem menolak registrasi dan menampilkan pesan bahwa email telah terdaftar.	Valid
3	Mengisi seluruh data dengan benar	Nama, email, password valid	Sistem berhasil membuat akun dan mengirim email verifikasi.	Akun berhasil dibuat dan email verifikasi berhasil dikirim ke pengguna.	Valid

Tabel 4.5 Black Box Testing Halaman Kelola Alamat

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan daftar alamat	Membuka halaman	Daftar alamat pengguna	Seluruh alamat pengguna berhasil	Valid

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
		kelola alamat	berhasil ditampilkan.	ditampilkan pada halaman kelola alamat.	
2	Menambahkan alamat baru	Mengisi data alamat dengan benar	Alamat baru berhasil disimpan.	Data alamat berhasil disimpan dan ditampilkan pada daftar alamat.	Valid
3	Mengubah data alamat	Mengubah informasi alamat	Data alamat berhasil diperbarui.	Informasi alamat berhasil diperbarui sesuai perubahan yang dilakukan pengguna.	Valid
4	Menghapus alamat	Memilih alamat yang akan dihapus	Alamat berhasil dihapus dari sistem.	Data alamat berhasil dihapus dari daftar alamat pengguna.	Valid
5	Menentukan alamat utama	Memilih salah satu alamat sebagai alamat utama	Alamat yang dipilih menjadi alamat utama dan digunakan sebagai alamat default saat checkout.	Alamat utama berhasil diperbarui dan digunakan sebagai alamat default pada proses checkout.	Valid

Tabel 4.6 Black Box Testing Halaman Daftar Produk

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan daftar produk	Membuka halaman produk	Daftar produk berhasil ditampilkan.	Seluruh produk berhasil ditampilkan sesuai data pada basis data.	Valid
2	Admin menambahkan stok produk	Data stok dan harga beli valid	Stok produk berhasil ditambahkan.	Stok produk berhasil diperbarui sesuai jumlah yang dimasukkan.	Valid

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
3	Admin mengubah data produk	Data produk baru	Produk berhasil diperbarui.	Data produk berhasil diperbarui dan ditampilkan pada daftar produk.	Valid
4	Admin menghapus produk	Pilih produk	Produk berhasil dihapus.	Produk berhasil dihapus dari sistem.	Valid
5	Pelanggan menambah produk ke keranjang	Pilih produk	Produk berhasil ditambahkan ke keranjang.	Produk berhasil masuk ke keranjang belanja.	Valid

Tabel 4.7 Black Box Testing Halaman Daftar Belanja

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan daftar belanja	Membuka halaman daftar belanja	Daftar produk pada daftar belanja berhasil ditampilkan.	Seluruh produk pada daftar belanja berhasil ditampilkan sesuai data pengguna.	Valid
2	Menambahkan produk ke daftar belanja	Memilih produk	Produk berhasil ditambahkan ke daftar belanja.	Produk berhasil ditambahkan ke daftar belanja dan muncul pada daftar belanja pengguna.	Valid
3	Mengubah jumlah produk pada daftar belanja	Mengubah jumlah produk	Jumlah produk berhasil diperbarui.	Jumlah produk berhasil diperbarui sesuai *input* pengguna.	Valid
4	Menghapus produk dari daftar belanja	Pilih produk	Produk berhasil dihapus dari daftar belanja.	Produk berhasil dihapus dari daftar belanja.	Valid
5	Menambahkan seluruh produk ke keranjang	Menekan tombol Tambah ke Keranjang	Seluruh produk pada daftar belanja berhasil dipindahkan ke	Seluruh produk berhasil ditambahkan ke keranjang belanja sesuai	Valid

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
			keranjang belanja.	data pada daftar belanja.	

Tabel 4.8 Black Box Testing Halaman Keranjang

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan isi keranjang	Membuka halaman keranjang	Seluruh produk dalam keranjang berhasil ditampilkan.	Daftar produk pada keranjang berhasil ditampilkan.	Valid
2	Mengubah jumlah produk	Jumlah baru	Jumlah produk berhasil diperbarui.	Jumlah produk berhasil diperbarui dan total belanja berubah otomatis.	Valid
3	Menghapus produk dari keranjang	Pilih produk	Produk berhasil dihapus dari keranjang.	Produk berhasil dihapus dari keranjang	Valid
4	Checkout dengan produk dan alamat yang valid	Keranjang berisi produk, alamat dipilih	Pesanan berhasil dibuat, stok diperbarui otomatis, dan status menjadi Diproses.	Produk Pesanan berhasil dibuat dengan status Diproses dan stok produk berkurang sesuai jumlah yang dipesan.	Valid

Tabel 4.9 Black Box Testing Halaman Daftar Pesanan Admin

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan daftar pesanan	Membuka halaman daftar pesanan	Daftar pesanan berhasil ditampilkan.	Seluruh pesanan berhasil ditampilkan sesuai status masing-masing.	Valid
2	Admin mengubah status menjadi Dikirim	Pilih pesanan	Status berubah menjadi Dikirim.	Status pesanan berhasil diperbarui menjadi Dikirim.	Valid

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
3	Admin mengubah status menjadi Selesai	Pilih pesanan	Status berubah menjadi Selesai.	Status pesanan berhasil diperbarui menjadi Selesai.	Valid
4	Admin mengubah status menjadi Dikirim dan Selesai tanpa checklist item pesanan terlebih dahulu	Pilih pesanan	Sistem menampilkan pesan bahwa item pesanan belum ter-checklist.	Sistem menolak perubahan status dan menampilkan pesan bahwa seluruh item harus di-checklist terlebih dahulu.	Valid

Tabel 4.10 Black Box Testing Halaman Kasir

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Scan barcode produk	Barcode valid	Produk berhasil ditemukan dan ditambahkan ke kasir.	Produk berhasil ditemukan berdasarkan barcode dan masuk ke daftar transaksi.	Valid
2	Menambah produk secara manual	Pilih produk	Produk berhasil ditambahkan ke kasir.	Produk berhasil ditambahkan ke daftar transaksi kasir.	Valid
3	Melakukan transaksi penjualan	Produk dan jumlah stok valid	Transaksi berhasil disimpan dan stok berhasil diperbarui.	Transaksi berhasil disimpan dan stok produk berkurang sesuai jumlah yang terjual.	Valid

Tabel 4.11 Black Box Testing Halaman Laporan Penjualan

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan laporan harian	Memilih laporan harian	Data laporan harian berhasil ditampilkan.	Data laporan harian berhasil ditampilkan sesuai transaksi.	Valid
2	Menampilkan laporan bulanan	Memilih laporan bulanan	Data laporan bulanan berhasil ditampilkan.	Data laporan bulanan berhasil ditampilkan	Valid

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
				sesuai periode yang dipilih.	
3	Menampilkan laporan tahunan	Memilih laporan tahunan	Data laporan tahunan berhasil ditampilkan.	Data laporan tahunan berhasil ditampilkan sesuai periode yang dipilih.	Valid

Tabel 4.12 Black Box Testing Halaman Profil

No	Skenario Pengujian	Test Case	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Menampilkan profil pengguna	Membuka halaman profil	Data profil berhasil ditampilkan.	Informasi profil pengguna berhasil ditampilkan sesuai data pada basis data.	Valid
2	Mengubah foto profil	File gambar (.jpg/.jpeg)	Foto profil berhasil diunggah dan ditampilkan pada profil.	Foto profil berhasil diperbarui dan ditampilkan pada halaman profil.	Valid

Berdasarkan hasil pengujian pada Tabel 4.3 sampai 4.12, seluruh fitur utama pada aplikasi Android Stokita berhasil dijalankan sesuai dengan kebutuhan fungsional yang telah ditetapkan. Setiap fitur mampu menerima masukan dari pengguna dan menghasilkan keluaran yang sesuai dengan proses bisnis sistem. Tidak ditemukan kesalahan fungsional yang menyebabkan kegagalan proses selama pengujian dilakukan.

Dengan demikian, hasil pengujian menggunakan metode *Black Box Testing* menunjukkan bahwa aplikasi Stokita telah memenuhi kebutuhan fungsional pengguna dan layak digunakan sebagai media pengelolaan stok barang serta penjualan online pada Fahri Mart.

4.2.3 Pengujian Usability (Metode System Usability Scale)

Pengujian *usability* dilakukan untuk mengetahui tingkat kemudahan penggunaan aplikasi Stokita berdasarkan persepsi pengguna setelah menggunakan sistem. Pengujian menggunakan metode *System Usability Scale* (SUS), yang terdiri atas 10 butir pernyataan dengan skala penilaian 1 sampai 5, mulai dari sangat tidak setuju hingga sangat setuju.

Pengujian SUS melibatkan 52 responden yang terdiri dari pengguna aplikasi Stokita. Responden diminta memberikan penilaian terhadap setiap pernyataan berdasarkan pengalaman mereka setelah menggunakan aplikasi. Sebelum hasil SUS digunakan untuk mengukur tingkat usability, instrumen terlebih dahulu diuji validitas dan reliabilitas untuk memastikan bahwa setiap butir pernyataan layak dan konsisten digunakan sebagai instrumen pengukuran.

4.2.4.1 Uji Validitas Instrumen SUS

Uji validitas dilakukan untuk mengetahui apakah setiap butir pernyataan pada instrumen *System Usability Scale* (SUS) mampu mengukur aspek usability aplikasi Stokita secara tepat. Instrumen SUS yang digunakan dalam penelitian ini terdiri atas 10 butir pernyataan, yaitu Q1 sampai Q10, dengan pernyataan positif pada nomor ganjil dan pernyataan negatif pada nomor genap.

Uji validitas dilakukan menggunakan metode korelasi *Pearson Product Moment* dengan mengorelasikan skor masing-masing butir pernyataan terhadap skor total. Sebelum dilakukan uji korelasi, skor jawaban responden dikonversi terlebih dahulu sesuai aturan perhitungan SUS, yaitu skor butir ganjil dikurangi 1, sedangkan skor butir genap dihitung dengan 5 dikurangi skor jawaban.

Pengujian dilakukan terhadap 52 responden dengan derajat kebebasan (*df*) = n − 2 = 50 dan taraf signifikansi α = 0,05, sehingga diperoleh nilai r tabel sebesar 0,2732. Kriteria pengambilan keputusan adalah apabila nilai r-hitung > r-tabel, maka butir pernyataan dinyatakan valid.

Tabel 4.13 Hasil Uji Validitas Instrumen SUS

Kode	Pernyataan	r hitung	r tabel	Keterangan
Q1	Saya merasa akan sering menggunakan sistem ini.	0,712	0,2732	Valid
Q2	Saya merasa sistem ini terlalu rumit.	0,839	0,2732	Valid
Q3	Saya merasa sistem ini mudah digunakan.	0,801	0,2732	Valid
Q4	Saya merasa perlu bantuan teknis untuk dapat menggunakan sistem ini.	0,614	0,2732	Valid
Q5	Fitur-fitur dalam sistem ini terintegrasi dengan baik.	0,723	0,2732	Valid
Q6	Saya merasa sistem ini memiliki terlalu banyak inkonsistensi.	0,662	0,2732	Valid
Q7	Saya membayangkan kebanyakan orang akan cepat belajar menggunakan sistem ini.	0,680	0,2732	Valid
Q8	Saya merasa sistem ini terlalu membingungkan saat digunakan.	0,709	0,2732	Valid
Q9	Saya merasa percaya diri saat menggunakan sistem ini.	0,752	0,2732	Valid
Q10	Saya harus banyak belajar sebelum dapat menggunakan sistem ini.	0,712	0,2732	Valid

Berdasarkan Tabel 4.13, seluruh butir pernyataan SUS memperoleh nilai r-hitung lebih besar daripada r-tabel 0,2732. Nilai korelasi terendah terdapat pada Q4 sebesar 0,614, sedangkan nilai tertinggi terdapat pada Q2 sebesar 0,839. Dengan demikian, seluruh 10 butir pernyataan SUS dinyatakan valid dan dapat digunakan dalam pengukuran *usability* aplikasi Stokita.

4.2.4.2 Uji Reliabilitas Instrumen SUS

Setelah pengujian validitas dilakukan, langkah selanjutnya adalah melakukan uji reliabilitas untuk mengukur konsistensi internal dari instrumen System Usability Scale (SUS). Reliabilitas instrumen diuji menggunakan metode Cronbach's Alpha, di mana suatu instrumen dinyatakan reliabel apabila nilai koefisien $\alpha > 0{,}60$.

Berdasarkan data jawaban asli dari 52 responden menggunakan skala Likert 1–5, diperoleh nilai varians masing-masing butir pernyataan dan varians total yang digunakan dalam perhitungan Cronbach's Alpha sebagai berikut:

Tabel 4.14 Hasil Perhitungan Varians dan Reliabilitas SUS

No	Kode Butir	Varians Butir (σ_i^2)
1	Q1	0,573
2	Q2	0,484
3	Q3	0,534
4	Q4	0,505
5	Q5	0,499
6	Q6	0,629
7	Q7	0,617
8	Q8	0,523
9	Q9	0,550
10	Q10	0,484
Jumlah Varians Butir ($\Sigma\sigma_i^2$)		**5,397**
Varians Skor Total (σ_t^2)		**32,632**
Koefisien Cronbach's Alpha (α)		**0,927**

Berdasarkan hasil pengolahan data pada Tabel 4.14, diperoleh nilai koefisien Cronbach's Alpha (α) sebesar 0,927. Nilai ini kemudian dibandingkan dengan standar reliabilitas yang ditetapkan, yaitu:

1. Nilai α (0,927) > 0,60, sehingga instrumen penelitian dinyatakan Reliabel.

2. Berdasarkan kategori interpretasi, nilai 0,927 berada pada rentang $0{,}80 < \alpha \leq 1{,}00$, yang berarti instrumen SUS dalam penelitian ini memiliki tingkat reliabilitas yang Sangat Tinggi.

Dengan hasil tersebut, dapat disimpulkan bahwa kuesioner SUS yang digunakan telah konsisten dan handal untuk mengukur tingkat *usability* aplikasi Stokita, sehingga data yang diperoleh dapat dilanjutkan ke tahap analisis skor *usability*.

4.2.4.3 Hasil Pengukuran Usability dengan SUS

Setelah instrumen SUS dinyatakan valid dan reliabel, tahap selanjutnya adalah melakukan pengukuran tingkat usability aplikasi Stokita menggunakan metode System Usability Scale (SUS). Pengukuran dilakukan berdasarkan jawaban dari 52 responden terhadap 10 pernyataan SUS dengan skala penilaian 1 sampai 5.

Sebelum dilakukan perhitungan skor SUS, jawaban responden terlebih dahulu dikonversi ke dalam skala 0–4 sesuai dengan aturan perhitungan *System Usability Scale*, yaitu:

a) Pernyataan bernomor ganjil (Q1, Q3, Q5, Q7, dan Q9) dihitung menggunakan rumus $(x - 1)$.
b) Pernyataan bernomor genap (Q2, Q4, Q6, Q8, dan Q10) dihitung menggunakan rumus $(5 - x)$.

Hasil konversi skor setiap butir pertanyaan untuk masing-masing responden disajikan pada Tabel 4.15.

Tabel 4.15 Distribusi Skor Jawaban Responden (Skala 0–4)

Kode Responden	Role	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total Skor
R1	Pelanggan	3	3	2	3	2	2	2	4	3	3	27
R2	Pelanggan	3	4	3	4	4	2	4	2	3	4	33
R3	Pelanggan	4	4	3	3	3	3	4	3	3	3	33
R4	Pelanggan	4	3	4	4	3	4	3	3	3	3	34
R5	Pelanggan	4	4	4	4	4	4	4	4	4	4	40

Kode Responden	Role	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total Skor
R6	Pelanggan	4	3	3	3	4	3	3	3	3	3	32
R7	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R8	Pelanggan	3	4	3	3	4	4	4	4	4	4	37
R9	Pelanggan	4	4	4	4	4	3	4	4	4	4	39
R10	Pelanggan	3	4	3	3	4	4	4	4	4	4	37
R11	Pelanggan	2	2	3	2	3	2	2	2	2	3	23
R12	Pelanggan	2	2	2	4	3	2	2	2	2	3	24
R13	Pelanggan	2	2	2	2	2	2	2	2	3	2	21
R14	Pelanggan	4	4	4	4	4	3	2	3	4	4	36
R15	Pelanggan	4	3	3	3	4	3	3	3	2	4	32
R16	Pelanggan	4	4	4	3	3	4	4	4	4	4	38
R17	Pelanggan	3	3	4	4	4	3	3	4	3	4	35
R18	Pelanggan	4	3	3	3	4	4	3	4	3	3	34
R19	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R20	Pelanggan	3	3	4	3	4	4	3	4	3	4	35
R21	Pelanggan	3	3	3	4	3	4	3	3	3	3	32
R22	Pelanggan	2	3	3	2	2	2	2	3	3	3	25
R23	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R24	Pelanggan	3	4	4	4	4	3	3	3	4	4	36
R25	Pelanggan	3	3	3	2	3	3	2	2	3	2	26
R26	Pelanggan	3	3	3	3	2	3	3	3	2	3	28
R27	Pelanggan	4	4	3	4	3	3	3	3	3	3	33
R28	Pelanggan	2	2	2	2	3	2	2	2	2	2	21
R29	Pelanggan	3	3	3	4	3	3	4	3	3	3	32
R30	Pelanggan	4	4	4	4	4	3	4	4	4	4	39
R31	Pelanggan	2	2	2	2	2	2	2	2	2	2	20
R32	Pelanggan	3	4	4	4	3	3	4	4	3	4	36
R33	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R34	Pelanggan	3	3	3	3	3	4	3	3	3	3	31

Kode Responden	Role	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total Skor
R35	Pelanggan	3	3	3	4	3	2	4	4	3	2	31
R36	Pelanggan	2	3	4	4	3	3	4	3	4	3	33
R37	Pelanggan	4	4	4	3	4	3	3	3	3	4	35
R38	Pelanggan	4	3	4	3	3	4	3	3	4	3	34
R39	Pelanggan	2	2	2	3	2	2	2	3	2	2	22
R40	Pelanggan	2	3	2	2	3	3	3	3	2	4	27
R41	Pelanggan	4	3	3	3	4	4	4	3	3	3	34
R42	Pelanggan	4	4	3	3	3	3	3	3	3	3	32
R43	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R44	Pelanggan	4	3	3	3	3	3	2	4	3	3	31
R45	Pelanggan	3	3	3	2	3	2	3	3	2	4	28
R46	Pelanggan	4	4	4	4	4	4	4	4	4	4	40
R47	Pelanggan	3	3	3	3	4	2	4	3	2	3	30
R48	Pelanggan	2	2	2	3	2	2	3	2	2	2	22
R49	Pelanggan	3	3	3	3	3	4	2	3	4	3	31
R50	Pelanggan	3	3	2	3	3	4	3	2	3	3	29
R51	Admin	3	4	4	3	4	4	4	4	4	3	37
R52	Admin	3	4	4	3	4	3	3	3	3	3	33
Total												1678

Berdasarkan data pada Tabel 4.15, diperoleh total skor konversi SUS dari seluruh responden sebesar **1678** yang selanjutnya dihitung menggunakan rumus *System Usability Scale* sebagai berikut.

$$\text{Skor SUS} = \frac{\text{Total Skor Mentah}}{\text{Jumlah Responden}} \times 2{,}5$$

Sehingga diperoleh skor rata-rata SUS sebagai berikut:

$$\text{Skor SUS} = \frac{1678}{52} \times 2{,}5 = 80{,}67$$

Hasil perhitungan tersebut menunjukkan bahwa aplikasi Stokita memperoleh nilai *System Usability Scale* sebesar **80,67**. Berdasarkan kategori penilaian SUS yang digunakan dalam penelitian ini, nilai tersebut termasuk ke dalam kategori *Acceptability*: **"Acceptable"**, *Grade Scale*: **"A"**, dan *Adjective Rating*: **"Excellent"** yang sesuai dengan rentang skor yang diperoleh.

4.3 Pembahasan

Pembahasan merupakan tahap analisis terhadap hasil implementasi dan pengujian aplikasi Stokita yang telah dilakukan. Pada bagian ini dibahas sejauh mana aplikasi mampu menjawab rumusan masalah penelitian, memenuhi kebutuhan pengguna, serta memberikan solusi terhadap permasalahan pengelolaan stok barang dan penjualan online yang sebelumnya dihadapi oleh Fahri Mart.

Analisis dilakukan secara mendalam berdasarkan perbandingan kondisi sistem sebelum dan sesudah implementasi, hasil pengujian fungsional dengan metode *Black Box Testing* dan API *Testing*, serta evaluasi tingkat kemudahan penggunaan menggunakan *System Usability Scale* (SUS).

4.3.1 Pembahasan Hasil Implementasi Sistem

Implementasi aplikasi Stokita berhasil mengubah proses pengelolaan stok barang dan transaksi penjualan di Fahri Mart yang sebelumnya dilakukan secara manual menjadi sistem digital berbasis Android. Aplikasi dibangun menggunakan arsitektur *Client–Server* dengan *backend* Node.js dan Express.js, *frontend* React Native, serta basis data MySQL, sehingga seluruh proses bisnis dapat dijalankan secara terintegrasi melalui RESTful API.

Implementasi sistem menghasilkan beberapa peningkatan dibandingkan dengan proses yang sebelumnya diterapkan. Perbandingan kondisi sebelum dan sesudah implementasi aplikasi ditunjukkan pada Tabel 4.16 berikut.

Tabel 4.16 Perbandingan Sistem Manual dan Sistem Stokita

No	Aspek Pembanding	Sistem Lama	Sistem Stokita
1	Pengelolaan Stok	Dicatat secara manual sehingga sering terjadi kesalahan pencatatan dan keterlambatan pembaruan stok.	Stok barang tersimpan pada database dan diperbarui secara otomatis sesuai transaksi.
2	Penjualan (Kasir)	Transaksi dicatat secara manual sehingga membutuhkan waktu lebih lama.	Transaksi dilakukan melalui aplikasi dan langsung tersimpan ke database.
3	Pemesanan Online	Belum bisa memesan secara online	Pelanggan dapat melakukan pemesanan secara langsung melalui aplikasi Android.
4	Pencarian Produk	Dilakukan secara manual sehingga membutuhkan waktu lebih lama.	Produk dapat dicari berdasarkan nama maupun filter kategori secara cepat.
5	Laporan Penjualan	Rekap penjualan dilakukan secara manual menggunakan buku	Laporan penjualan harian, bulanan, dan produk terlaris dapat ditampilkan secara otomatis.
6	Keamanan Data	Data mudah hilang atau rusak karena hanya tersimpan pada pencatatan manual.	Data tersimpan pada database MySQL dan dilindungi menggunakan autentikasi JWT.

Berdasarkan perbandingan pada Tabel 4.4, digitalisasi ini memberikan dampak nyata pada efisiensi harian Fahri Mart. Dari sisi administrasi, laporan penjualan harian dan bulanan kini dapat diakses secara instan oleh admin. Penerapan arsitektur *Client-Server* juga memastikan seluruh data tersimpan terpusat pada server Aiven MySQL, sedangkan file gambar produk, kategori, dan foto profil

tersimpan aman di Cloudinary. Hal ini mengeliminasi risiko duplikasi data serta memberikan jaminan ketersediaan data yang konsisten dan andal.

4.3.2 Pembahasan Hasil Pengujian Fungsional

Hasil pengujian fungsional baik pada tingkat RESTful API (menggunakan Postman) maupun antarmuka aplikasi Android (menggunakan *Black Box Testing*) menunjukkan tingkat keberhasilan sebesar 100% (*Success Rate*). Seluruh skenario pengujian yang dirancang berhasil dieksekusi dengan hasil aktual yang sepenuhnya sesuai dengan hasil yang diharapkan.

Beberapa aspek logika bisnis krusial yang berhasil dievaluasi dan dibahas meliputi:

1. **Keakuratan Mekanisme Sinkronisasi Stok Otomatis**: Mekanisme pembaruan stok otomatis merupakan salah satu fitur inti Stokita untuk menjaga konsistensi antara data digital di server dengan kondisi fisik produk di rak toko. Berdasarkan hasil pengujian fungsional, pengurangan dan penambahan stok terjadi secara presisi dan *real-time* melalui skenario berikut:
    a) Transaksi Offline (Kasir): Stok produk berkurang secara otomatis seketika setelah admin atau kasir mengonfirmasi transaksi pembayaran pada halaman kasir.
    b) Transaksi Online (Pelanggan): Pengurangan stok produk terjadi secara otomatis pada saat pelanggan menekan tombol Checkout. Sistem terlebih dahulu memvalidasi sisa stok di database, dan jika mencukupi, stok langsung dikurangi dan status pesanan diubah menjadi "Diproses".
    c) Fungsi Status "Selesai": Ketika pesanan online telah diterima pelanggan dan admin mengubah statusnya menjadi "Selesai", sistem tidak melakukan pengurangan stok lagi. Pada tahap ini, status "Selesai" murni berfungsi untuk memfinalisasi transaksi, mengunci data penjualan, dan mencatatkan nilai transaksi ke dalam laporan keuangan serta perhitungan keuntungan.

d) Riwayat Penambahan Stok: Setiap aktivitas restok atau penambahan pasokan produk baru dicatat secara transparan oleh admin melalui modul Stock In, yang riwayatnya tersimpan rapi dalam entitas stock_in di database.

2. **Keandalan Autentikasi dan Otorisasi (RBAC)**: Pengujian fungsional membuktikan efektivitas keamanan berbasis token JWT. Sistem berhasil menerapkan pembatasan hak akses menggunakan metode *Role-Based Access Control* (RBAC):
   a) Jika pengguna tidak menyertakan token JWT yang valid dalam header authorization, server secara konsisten mengembalikan respons *HTTP 401 Unauthorized*.
   b) Jika pengguna dengan peran Pelanggan mencoba menembus pertahanan sistem dengan mengakses endpoint khusus admin (seperti menambah produk atau melihat laporan profit), server secara andal menolak dan mengembalikan respons *HTTP 403 Forbidden*.
3. **Konsistensi Penanganan Kesalahan (*Error Handling*)**: Pengujian fungsional pada API juga mengonfirmasi keandalan sistem dalam menangani kesalahan *input* pengguna. Sistem memberikan respons kesalahan yang deskriptif dan konsisten menggunakan kode status standar HTTP (misalnya *400 Bad Request* untuk *input* tidak valid atau *404 Not Found* untuk pencarian barcode yang tidak terdaftar), sehingga memudahkan penanganan galat (*error handling*) pada sisi aplikasi Android.

4.3.3 Pembahasan Hasil Pengujian Usability SUS

Pembahasan tingkat *usability* sistem bertujuan untuk menafsirkan skor *System Usability Scale* (SUS) yang diperoleh secara objektif dan terstruktur. Berdasarkan hasil pengujian yang melibatkan 52 responden (2 admin dan 50 pelanggan), aplikasi Stokita memperoleh skor SUS rata-rata sebesar 80,67. Interpretasi terhadap hasil ini didukung oleh analisis keandalan data dan standar penilaian SUS global sebagai berikut:

1. **Validitas dan Reliabilitas sebagai Dasar Keandalan Data**: Sebelum dilakukan interpretasi skor, instrumen kuesioner SUS telah dinyatakan memenuhi syarat ilmiah pada tahap pengujian instrumen di subbab 4.2.3. Seluruh butir pernyataan (Q1-Q10) terbukti valid dengan nilai r-hitung > 0,2732. Selain itu, tingkat reliabilitas instrumen berada pada kategori Sangat Tinggi dengan koefisien Cronbach's Alpha sebesar 0,927. Hal ini menunjukkan bahwa skor 80,67 yang diperoleh bukan merupakan hasil kebetulan, melainkan refleksi konsisten dari persepsi pengguna terhadap aplikasi Stokita.
2. **Interpretasi Berdasarkan Standar Kelayakan SUS**: Mengacu pada kerangka teoretis standar penilaian SUS global, skor 80,67 memberikan gambaran kualitas aplikasi dari tiga sudut pandang:
    a) *Acceptability Ranges*: Skor ini berada jauh di atas nilai rata-rata standar kelayakan (skor 68), sehingga masuk dalam kategori *Acceptable*. Artinya, sistem telah memenuhi tingkat penerimaan pengguna secara optimal untuk mendukung operasional harian di Fahri Mart.
    b) *Adjective Rating*: Berdasarkan penilaian kualitatif, aplikasi Stokita mendapatkan predikat *Excellent*, yang mengindikasikan bahwa sistem dinilai sangat mudah digunakan dan memberikan kepuasan tinggi bagi pengguna.
    c) *Grade Scale*: Pada skala kelas, skor ini berada pada rentang *Grade* A, yang membuktikan kualitas usability Stokita memiliki standar tinggi dibandingkan aplikasi sejenis.
3. **Analisis Pola Jawaban Responden**: Tingginya skor SUS didukung oleh pola jawaban responden yang menunjukkan konsistensi persepsi. Pada pernyataan positif (nomor ganjil) seperti kemudahan penggunaan (Q3) dan integrasi fitur (Q5), mayoritas responden memberikan penilaian tinggi (skor 4 dan 5). Sebaliknya, pada pernyataan negatif (nomor genap) seperti kerumitan sistem (Q2) dan kebutuhan bantuan teknis (Q4), responden memberikan skor rendah (1 dan 2). Pola ini menegaskan bahwa antarmuka

aplikasi yang dibangun menggunakan React Native berhasil memberikan interaksi yang intuitif bagi pengguna di tingkat UMKM ritel.

4. **Kesimpulan Efektivitas Digitalisasi**: Hasil pengujian *usability* ini membuktikan bahwa transisi operasional Fahri Mart dari sistem konvensional (buku tulis) ke platform digital berbasis Android telah dicapai dengan sukses. Aplikasi Stokita tidak hanya unggul secara fungsional dengan tingkat keberhasilan 100%, tetapi juga berhasil mengadopsi prinsip desain user-centered yang memudahkan pemilik toko dalam mengelola stok serta memberikan kenyamanan bagi pelanggan dalam bertransaksi secara online. Dengan demikian, Stokita dinyatakan sangat layak dan siap untuk diimplementasikan sepenuhnya pada Fahri Mart.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan pada penelitian ini, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Aplikasi "Stokita" berbasis Android untuk pengelolaan stok dan penjualan online pada Fahri Mart telah berhasil dirancang dan dibangun menggunakan arsitektur *Client-Server* berbasis RESTful API. Aplikasi ini menggunakan *framework* React Native pada sisi *frontend*, Node.js dan Express.js pada sisi *backend*, serta MySQL sebagai basis data relasional. Sistem ini mampu mendigitalisasi operasional manual Fahri Mart secara terintegrasi.
2. Implementasi fitur fungsional sistem telah berjalan dengan sangat baik dan andal. Seluruh modul krusial seperti manajemen pengguna berbasis *Role-Based Access Control* (RBAC) dengan token JWT, pengelolaan data produk/kategori, pemindaian barcode produk, transaksi kasir offline harian, keranjang belanja dan checkout pesanan online, notifikasi stok menipis, hingga rekap laporan keuangan harian dan bulanan otomatis telah berfungsi secara presisi. Berdasarkan pengujian RESTful API menggunakan Postman dan pengujian antarmuka menggunakan *Black Box Testing*, seluruh fitur yang diuji memperoleh persentase keberhasilan (*Success Rate*) sebesar 100%.
3. Berdasarkan pengujian tingkat kebergunaan (*usability*) menggunakan metode *System usability Scale* (SUS) terhadap 52 responden (2 admin Fahri Mart dan 50 pelanggan), aplikasi ini memperoleh skor rata-rata sebesar 80,67. Mengacu pada standar penilaian SUS global, skor tersebut menempatkan aplikasi Stokita ke dalam kategori *Acceptability*: "Acceptable", *Grade Scale* "A", dan *Adjective Rating* "Excellent". Hasil ini mengonfirmasi bahwa Stokita sangat mudah digunakan, memiliki kurva

pembelajaran yang sangat pendek, dan siap diadaptasikan dengan nyaman oleh pemilik toko maupun pelanggan awam sekalipun.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat digunakan sebagai bahan pertimbangan untuk pengembangan sistem selanjutnya, yaitu:

1. Pengembangan aplikasi versi iOS perlu dipertimbangkan di masa depan. Mengingat Stokita dibangun menggunakan *framework* lintas platform (*cross-platform*) React Native, pengemasan aplikasi untuk platform iOS dapat dilakukan secara efisien menggunakan satu basis kode (*single codebase*) yang sama demi memperluas aksesibilitas pengguna.
2. Integrasi dengan sistem gerbang pembayaran eksternal (*payment gateway*) seperti Midtrans atau Xendit. Penambahan fitur ini pada modul transaksi online pelanggan akan memungkinkan proses verifikasi pembayaran digital (seperti *e-wallet*, transfer bank, atau QRIS) berjalan secara otomatis dan *real-time* tanpa memerlukan pemeriksaan manual oleh admin toko.
3. Pengembangan fitur manajemen stok multi-satuan (*multi-unit of measurement*), seperti penambahan stok grosir (per dus atau per pak) di sisi admin serta fitur checkout grosir di sisi pelanggan. Saat ini, sistem Stokita hanya mendukung pencatatan produk dan transaksi dalam satuan tunggal (eceran/satuan). Dengan menerapkan konversi satuan yang terintegrasi secara matematis di dalam basis data MySQL (misalnya, 1 dus berisi 24 satuan), admin Fahri Mart dapat melakukan restock barang berskala besar secara lebih efisien, sementara pelanggan dapat melakukan pembelian dalam jumlah grosir (dus/pak) secara praktis pada menu penjualan online.

DAFTAR PUSTAKA

Abadi A. N., A. F. (2024). Perancangan Sistem Informasi Manajemen Stok dan Penjualan Buku Berbasis Android pada Musi Bookstore. *Jurnal Teknologi Informasi*, *5*(3). https://doi.org/10.46576/djtechno

Andrianto M. R., F. B. R. P. F. F. E. & H. M. M. (2025). Pengembangan Aplikasi Mobile " MyAgenda " dengan React Native , Node . js , dan MySQL. *Prosiding Seminar Nasional Informatika Bela Negara (SANTIKA)*, *5*(1).

Anugraha A. Z., N. M. I. P. & S. S. S. A. (2025). Peran Sistem Informasi dalam Meningkatkan Kepercayaan Pengguna pada Platform E-Commerce. *Kohesi: Jurnal Multidisiplin Saintek*, *8*(7). https://doi.org/10.8734/Kohesi.v1i2.365

Asoka, E., Rahmi, L., & Hapsari, Y. (2024). *Penggunaan Framework React Native Dalam Perancangan Aplikasi Penjualan Goodday Garden*. *8*(2).

Bisma Muhammad Hermawan, Muhammad Abdulrahman Hakim, Rijal Arifin, & Edy Susena. (2025). Sistem Informasi Manajemen Stok Toko Kelontong Berbasis Web untuk Pencatatan Stok di Toko Dandung. *Switch : Jurnal Sains Dan Teknologi Informasi*, *3*(4), 19–34. https://doi.org/10.62951/switch.v3i4.507

Brooke, J. (1996). SUS - A quick and dirty usability scale Industrial usability evaluation. *Usability Evaluation in Industry*, 189–194.

Cahyono S. A. B., S. S. & F. R. (2023). Implementasi Otentikasi Website Node JS Express Menggunakan Passport. *JSITIK: Jurnal Sistem Informasi Dan Teknologi Informasi Komputer*, *2*(1).

Christian, C., & Voutama, A. (2024). RANCANG BANGUN APLIKASI SISTEM INFORMASI INVENTARIS BERBASIS WEBSITE. *Jurnal Informatika Dan Teknik Elektro Terapan*, *12*(2). https://doi.org/10.23960/jitet.v12i2.4259

Dermawan D. T. B., & M. D. (2023). E-Commerce: Definisi, Perkembangan dan Hukum dalam Pandangan Agama Islam. *Jurnal Ekonomi Manajemen Akuntansi*, *29*(1).

Gayatri, G. T. S. (2025). Meningkatkan Efisiensi Pengelolaan Stok pada UMKM Online Shop X: Studi Kasus Penggunaan Ginee Omnichannel. *Jurnal Pendidikan Tambusai*, *9*(2).

Hanani, N. F., Razendri Ingnasia, C., Pratama, L. P., Alfarizi, S., Zaki, A., Akbar, A., & Akbar, A. (2025). *Optimalisasi Manajemen Stok UMKM Kosmetik Melalui Aplikasi Inventaris dengan Notifikasi Kadaluarsa Otomatis* (Vol. 5).

Hasan R., S. K. & M. L. B. F. (2024). Sistem Informasi Administrasi pada UMKM Kema Sama Berbasis Web. *Jurnal Sistem Informasi Dan Teknik Komputer*, *9*(2).

Hidayattullah M. S., H. M. I. S. H. & P. A. (2025). Pengembangan Sistem Pendaftaran dan Pengelolahan Laporan Magang di Kominfo Kota Palembang Berbasis Website Menggunakan Next Js dan Express. *Jurnal Informatika Dan Tekonologi Komputer*, *5*(3).

Ho J. A., B. J. T. P. N. J. R. S. S. A. & F. D. (2025). Perancangan Sistem Manajemen Stok Toko Mas XYZ Berbasis Web. *Journal of Information Technology and Computer Science (INTECOMS)*, *8*(6).

Kismawan, L. (2025). PERBANDINGAN METODE SDLC WATERFALL, AGILE, DAN SPIRAL DALAM PENGEMBANGAN SISTEM. *Prosiding UNID Yang Belum Dipublikasikan*.

Kurniawan A. E., & H. R. (2020). Automatic Testing with Black Box Testing Methods and Boundary Value Analysis Techniques at PT. Hartono Istana Teknologi. *PROXIES*, *4*(1), 1–17.

Lewis, J. R. (2018). The System Usability Scale: Past, Present, and Future. *International Journal of Human–Computer Interaction*, *34*(7), 577–590. https://doi.org/10.1080/10447318.2018.1455307

Lyonita, I., Maghfirah, P. D., Mediana, S. P., Saleha, B., Zebua, A., Nasution, D. P., Pembangunan, J. E., Sains, S., & Pancabudi, U. P. (2024). PENGARUH PENGGUNAAN E-COMMERCE TERHADAP BISNIS UMKM DALAM MENINGKATKAN DAYA SAING EKONOMI DI ERA DIGITAL. *Jurnal Ekonomi Manajemen Dan Bisnis*, *1*(3), 172–176. https://doi.org/10.62017/jemb

Manurung R., S. T. T. & N. B. R. M. (2024). Sistem Informasi Penjualan Terintegrasi Android: Solusi Digitalisasi UMKM dalam Era Ekonomi Digital (Studi Kasus: KUGAR Minamas Pansela). *Jurnal Informatika Dan Teknologi*, *10*(2).

Maulana, A. (2024, December 19). *React Native vs Kotlin: Pilih yang Mana untuk Proyek Mobile?* https://www.codepolitan.com/blog/react-native-vs-kotlin-pilih-yang-mana-untuk-proyek-mobile-1pibs3/

Minasa S., S. F. M. M. N. A. J. B. & S. A. (2024). Sistem Informasi Pengelolaan Inventaris UMKM Berbasis Web dengan Pendekatan Agile. *Jurnal Teknologi Informasi Dan Elektronika*, *9*(2). https://doi.org/10.32897/infotronik.2024.9.2.3783

Muhammad, R., Putra Pratama, S., Alijoyo, F. A., & Sofiah, E. (2025). OPTIMALISASI PROSES BISNIS UMKM MELALUI USER INTERFACE SISTEM POINT OF SALE (STUDI KASUS: CIRENG ASGARD). In *Jurnal Sistem Informasi* (Vol. 7, Number 2).

Nurlaela L., S. D. & W. F. T. (2024). Penerapan Metode User Centered Design pada Aplikasi UMKM Berbasis Android (Studi Kasus: UMKM Kenyot Susu Boyolali). *JITU: Journal Informatic Technology and Communication*, *8*(1).

Oktavianus R., R. O. H. D. B. J. S. E. & W. Y. (2023). Sistem Manajemen Stok Barang Berbasis Mobile. *Jurnal Sistem Cerdas Dan Rekayasa (JSCR)*, *5*(2).

Pragestu S., S. H. & R. E. F. (2023). Analisis Skalabilitas Web Server Apache Tomcat, Node.Js dan Go pada Protokol Hypertext Transfer Protocol (HTTP) dan Message Queue Telemetry Transport (MQTT). *JUSTIN (Jurnal Sistem Dan Teknologi Informasi)*, *11*(4). https://doi.org/10.26418/justin.v11i4.71607

Pranoto, H. Y. & G. T. (2021). Monitoring Aplikasi Produksi Produk Busa Menggunakan Metode UML dan Waterfall. *JISA (Jurnal Informatika Dan Sains)*, *4*(2).

Putra H. B. P., & S. S. D. (2024). Penerapan Sistem Point of Sale Berbasis Android untuk Peningkatan Kinerja Usaha. *Jurnal Informatika Dan Teknologi*, *7*(1).

Rahmat, M., & Diyani, L. A. (2024). Diploma Tiga Akuntansi; Universitas Bina Insani. *Universitas Bina Insani; Jl. Raya Siliwangi*, *9*(3), 82400924.

Ramadhani Z., P. P. & F. M. (2024). Implementasi Metode Design Thinking pada Perancangan Aplikasi UMKM Berbasis Mobile (Studi Kasus UMKM Amplang Cita Rasa di Kecamatan Muara Badak). *Sebatik*, *28*(2). https://doi.org/10.46984/sebatik.v28i2.0000

Rezkia, D. P. (2024). Analisis Penggunaan E-Commerce bagi UMKM di Era Digital. *Seminar Nasional Prosiding Ilmu Manajemen Kewirausahaan Dan Bisnis*, *1*(1).

Saputra, I., Ritonga, A., Wasliono, A., & Azura Lubis, S. (2025). Manajemen Stok dan Penjualan UMKM Rilis Kosmetik Berbasis Google Sheets. *Jurnal Kemitraan Dan Pengabdian*, *1*(2), 89–99. https://doi.org/10.65358/jurnamitra.v1i2.71

Saragih R., G. I. S. & K. F. (2025). Pemanfaatan Aplikasi Kasir Digital Berbasis Android untuk UMKM di Desa Cinta Rakyat. *Jurnal Pengabdian Masyarakat Berdampak (JUBEMPA)*, *1*(1).

Septian, O., Andhika Kusuma, W., & Wahyuni, E. D. (2019). *Analisis Perbandingan Usability Dan User Experience Terhadap E-Trust Pada Situs Ecommerce C2C Menggunakan Heart Dan Pulse Framework*. *1*(1), 27–38.

Sugiyono. (2019). *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Alfabeta.

Tyanafisya, A., Faras, A., Nashwandra, N. B., Pratama, R., Thahir, R. M., Wicaksono, A., & Mindara, G. P. (2025). *Black Box Testing using Equivalence Partitioning Technique on Bakkar Website under a Creative Commons Attribution-NonCommercial ShareAlike 4.0 International (CC BY-NC-SA 4.0)*. *10*(1). https://journal.uin-alauddin.ac.id/index.php/instek

Wahid, A. A. (2020). Analisis Metode Waterfall untuk Pengembangan Sistem Informasi. *Jurnal Ilmu-Ilmu Informatika Dan Manajemen STMIK*.

LAMPIRAN

Lampiran 1. Pertanyaan Pengujian Black Box

Skenario: Mengisi email dan password yang salah.

Hasil yang diharapkan: Sistem menampilkan pesan kesalahan login dan proses login dibatalkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengisi email dan password yang benar

Hasil yang diharapkan: Sistem berhasil login dan menampilkan halaman utama sesuai hak akses pengguna.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 3 dari 11

Pengujian Black Box

Halaman Register

Skenario: Seluruh data dikosongkan lalu menekan tombol Daftar

Hasil yang diharapkan: Sistem menampilkan validasi bahwa seluruh data wajib diisi.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengisi data dengan email yang sudah digunakan

Hasil yang diharapkan: Sistem menampilkan pesan bahwa email sudah digunakan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengisi seluruh data dengan benar *

Hasil yang diharapkan: Sistem berhasil membuat akun dan mengirim email verifikasi.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 4 dari 11

Pengujian Black Box Kelola Alamat

Halaman Kelola Alamat

Skenario: Menampilkan daftar alamat *

Hasil yang diharapkan: Daftar alamat pengguna berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menambahkan alamat baru *

Hasil yang diharapkan: Alamat baru berhasil disimpan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengubah data alamat *

Hasil yang diharapkan: Data alamat berhasil diperbarui.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menghapus alamat *

Hasil yang diharapkan: Alamat berhasil dihapus dari sistem.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menentukan alamat utama *

Hasil yang diharapkan: Alamat yang dipilih menjadi alamat utama dan digunakan sebagai alamat default saat checkout.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 5 dari 11

Pengujian Black Box

Halaman Daftar Produk

Skenario: Menampilkan daftar produk *

Hasil yang diharapkan: Daftar produk berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Admin mengubah data produk *

Hasil yang diharapkan: Produk berhasil diperbarui.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Admin menghapus produk *

Hasil yang diharapkan: Produk berhasil dihapus.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Pelanggan menambah produk ke keranjang *

Hasil yang diharapkan: Produk berhasil ditambahkan ke keranjang.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 6 dari 11

Pengujian Black Box

Halaman Daftar Belanja

Skenario: Menampilkan daftar belanja *

Hasil yang diharapkan: Daftar produk pada daftar belanja berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menambahkan produk ke daftar belanja *

Hasil yang diharapkan: Produk berhasil ditambahkan ke daftar belanja.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengubah jumlah produk pada daftar belanja *

Hasil yang diharapkan: Jumlah produk berhasil diperbarui.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menghapus produk dari daftar belanja *

Hasil yang diharapkan: Produk berhasil dihapus dari daftar belanja.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menambahkan seluruh produk ke keranjang *

Hasil yang diharapkan: Seluruh produk pada daftar belanja berhasil dipindahkan ke keranjang belanja.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 7 dari 11

Pengujian Black Box

Halaman Keranjang

Skenario: Menampilkan isi keranjang *

Hasil yang diharapkan: Seluruh produk dalam keranjang berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengubah jumlah produk *

Hasil yang diharapkan: Jumlah produk berhasil diperbarui.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menghapus produk dari keranjang *

Hasil yang diharapkan: Produk berhasil dihapus dari keranjang.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Checkout dengan produk dan alamat yang valid *

Hasil yang diharapkan: Pesanan berhasil dibuat, stok diperbarui otomatis, dan status menjadi Diproses.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 8 dari 11

Pengujian Black Box

Halaman Daftar Pesanan Admin

Skenario: Menampilkan daftar pesanan *

Hasil yang diharapkan: Daftar pesanan berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Admin mengubah status menjadi Dikirim *

Hasil yang diharapkan: Status berubah menjadi Dikirim.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Admin mengubah status menjadi Selesai *

Hasil yang diharapkan: Status berubah menjadi Selesai.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Admin mengubah status menjadi Dikirim dan Selesai tanpa checklist item pesanan terlebih dahulu *

Hasil yang diharapkan: Sistem menampilkan pesan bahwa item pesanan belum ter-checklist.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 9 dari 11

Pengujian Black Box

Halaman Kasir

Skenario: Scan barcode produk *

Hasil yang diharapkan: Produk berhasil ditemukan dan ditambahkan ke kasir.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menambah produk secara manual *

Hasil yang diharapkan: Produk berhasil ditambahkan ke kasir.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Melakukan transaksi penjualan *

Hasil yang diharapkan: Transaksi berhasil disimpan dan stok berhasil diperbarui.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 10 dari 11

Pengujian Black Box

Halaman Laporan Penjualan

Skenario: Menampilkan laporan harian *

Hasil yang diharapkan: Data laporan harian berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menampilkan laporan bulanan *

Hasil yang diharapkan: Data laporan bulanan berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Menampilkan laporan tahunan *

Hasil yang diharapkan: Data laporan tahunan berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Bagian 11 dari 11

Pengujian Black Box

Halaman Profil

Skenario: Menampilkan profil pengguna *

Hasil yang diharapkan: Data profil berhasil ditampilkan.

- [ ] Berhasil
- [ ] Tidak Berhasil

Skenario: Mengubah foto profil *

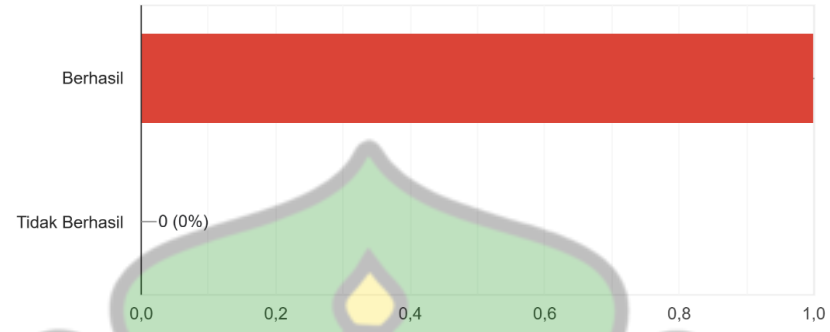
Hasil yang diharapkan: Foto profil berhasil diunggah dan ditampilkan pada profil.

- [ ] Berhasil
- [ ] Tidak Berhasil

Lampiran 2. Jawaban Pengujian Black Box

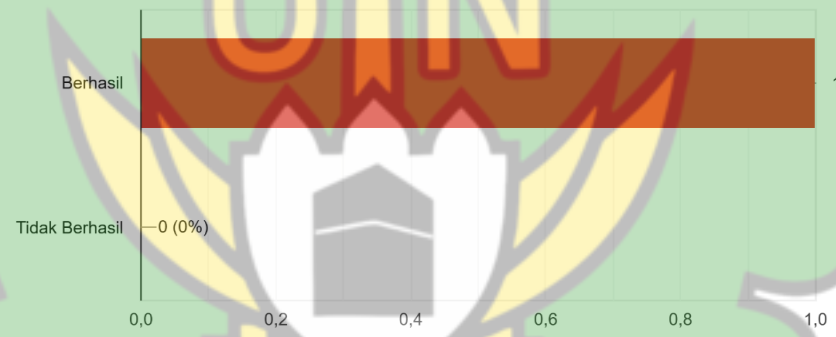
Skenario: Email dan password dikosongkan kemudian menekan tombol Login. Hasil yang diharapkan: Sistem menampilkan pesan bahwa email dan password harus diisi.

1 jawaban



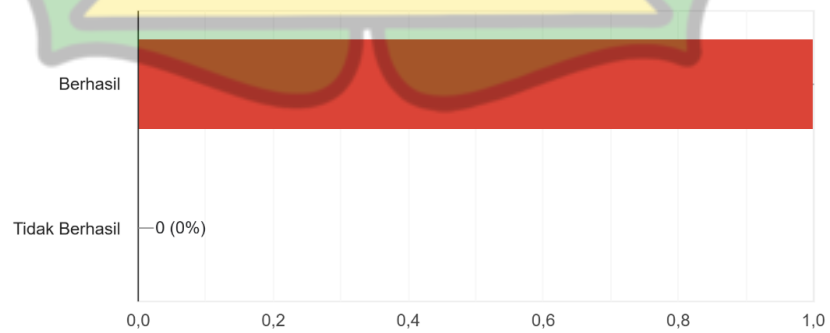
Skenario: Mengisi email tetapi password dikosongkan. Hasil yang diharapkan: Sistem menampilkan pesan bahwa password wajib diisi.

1 jawaban



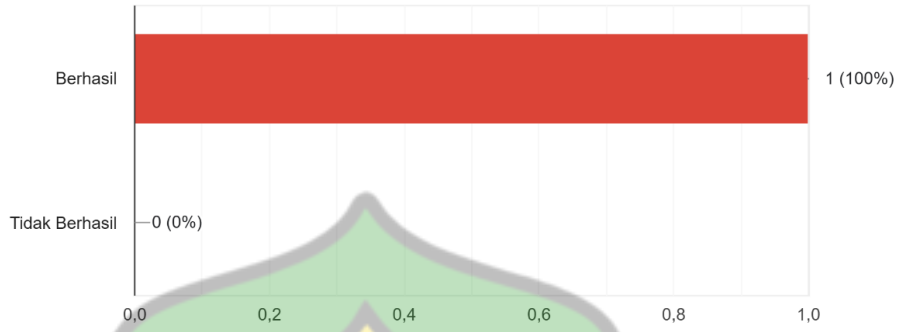
Skenario: Mengisi email dan password yang salah. Hasil yang diharapkan: Sistem menampilkan pesan kesalahan login dan proses login dibatalkan.

1 jawaban



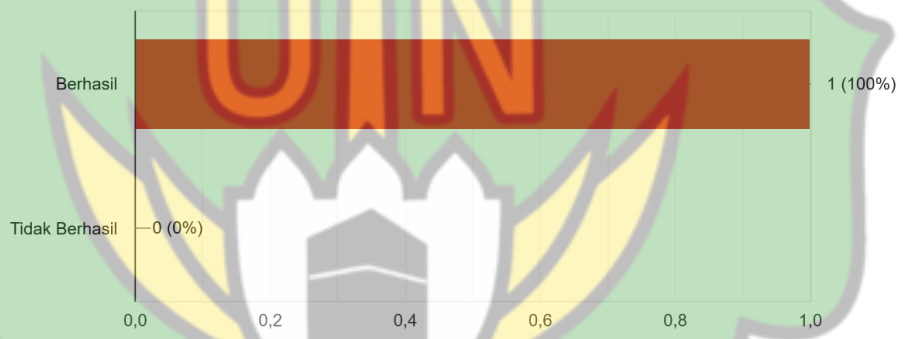
Skenario: Mengisi email dan password yang benar Hasil yang diharapkan: Sistem berhasil login dan menampilkan halaman utama sesuai hak akses pengguna.

1 jawaban



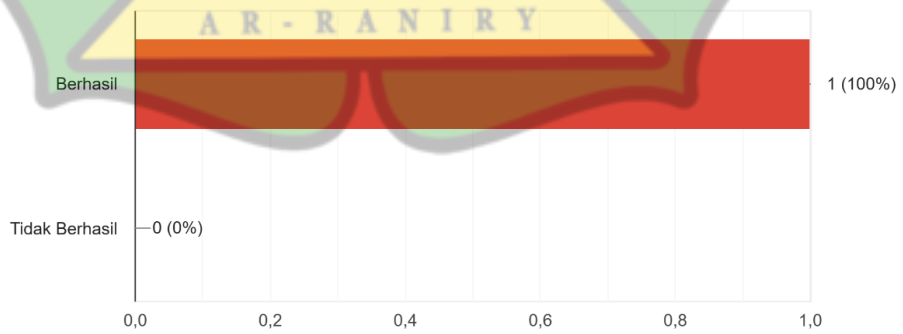
Skenario: Mengisi data dengan email yang sudah digunakan Hasil yang diharapkan: Sistem menampilkan pesan bahwa email sudah digunakan.

1 jawaban



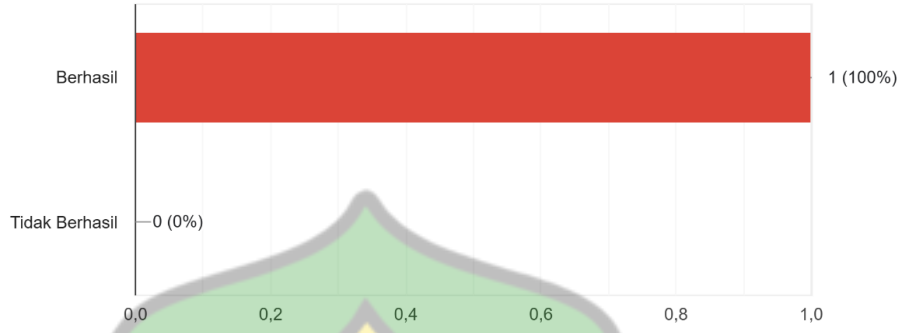
Skenario: Seluruh data dikosongkan lalu menekan tombol Daftar Hasil yang diharapkan: Sistem menampilkan validasi bahwa seluruh data wajib diisi.

1 jawaban



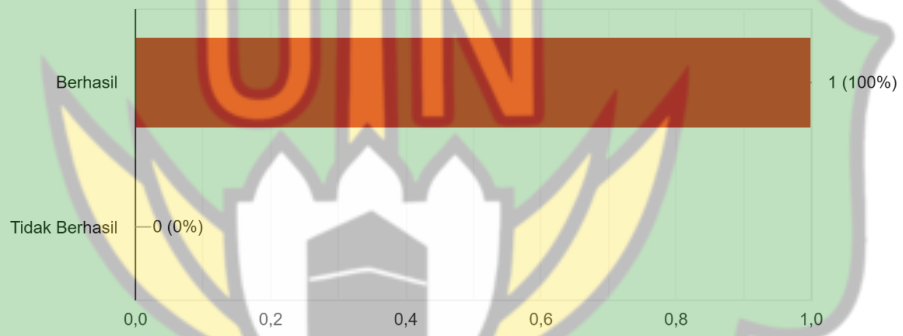
Skenario: Mengisi seluruh data dengan benar Hasil yang diharapkan: Sistem berhasil membuat akun dan mengirim email verifikasi.

1 jawaban



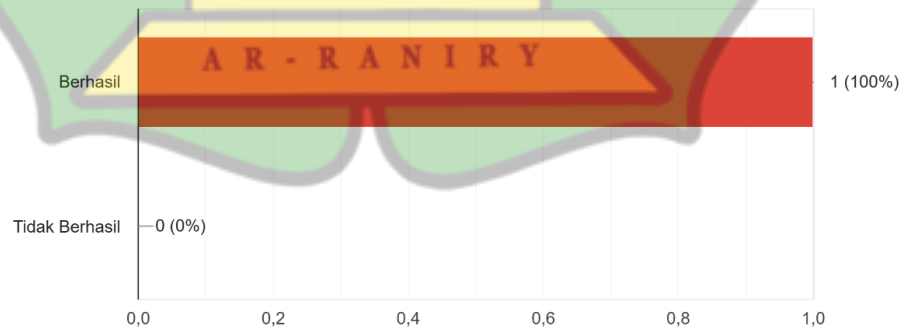
Skenario: Menampilkan daftar alamat Hasil yang diharapkan: Daftar alamat pengguna berhasil ditampilkan.

1 jawaban



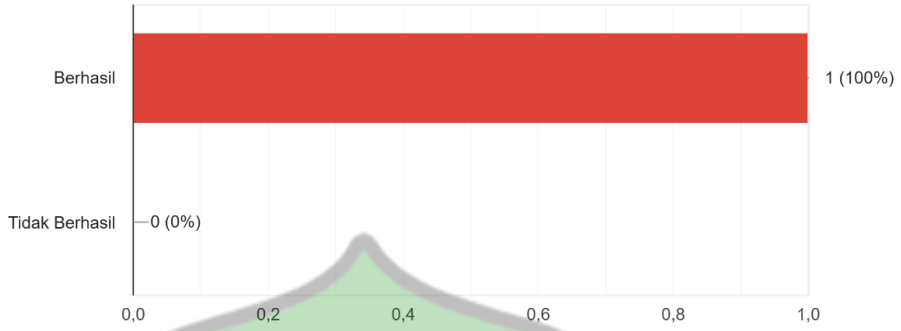
Skenario: Menambahkan alamat baru Hasil yang diharapkan: Alamat baru berhasil disimpan.

1 jawaban



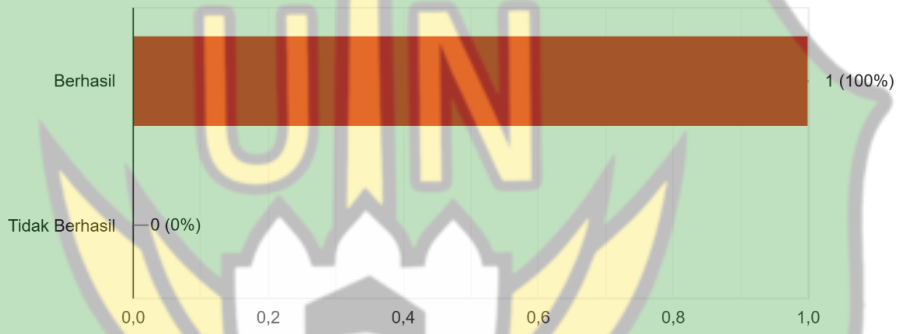
Skenario: Mengubah data alamat Hasil yang diharapkan: Data alamat berhasil diperbarui.

1 jawaban



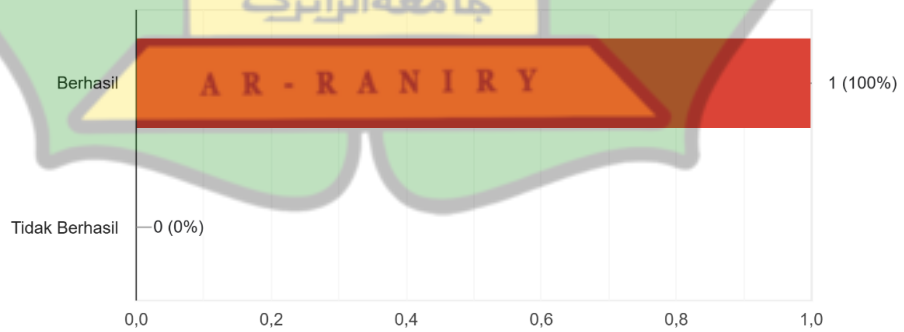
Skenario: Menghapus alamat Hasil yang diharapkan: Alamat berhasil dihapus dari sistem.

1 jawaban



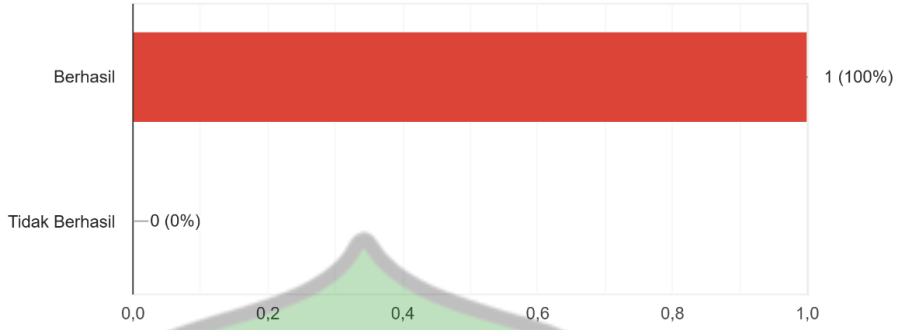
Skenario: Menentukan alamat utama Hasil yang diharapkan: Alamat yang dipilih menjadi alamat utama dan digunakan sebagai alamat default saat checkout.

1 jawaban



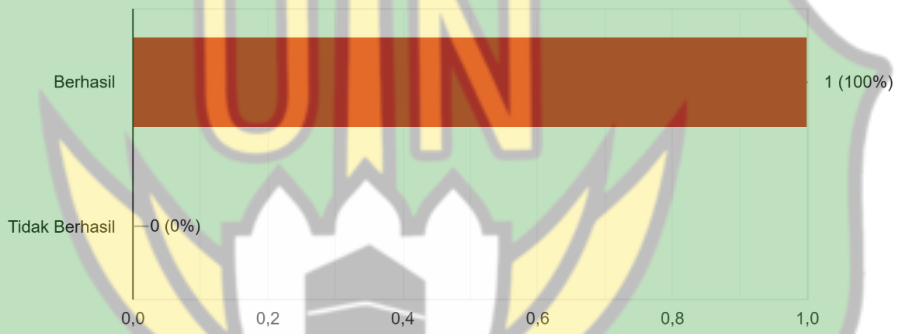
Skenario: Menampilkan daftar produk Hasil yang diharapkan: Daftar produk berhasil ditampilkan.

1 jawaban



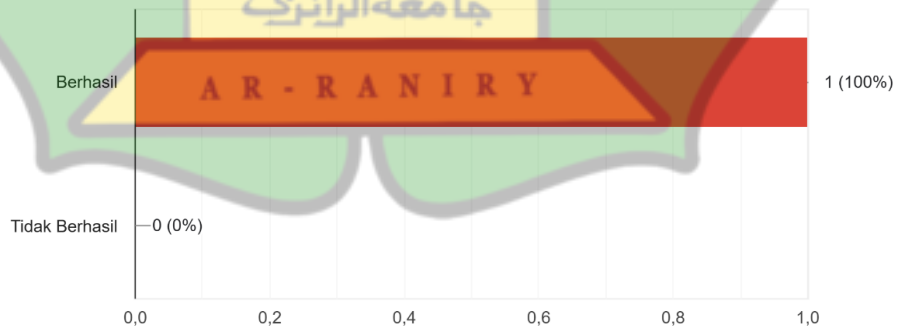
Skenario: Admin menambahkan stok produk Hasil yang diharapkan: Stok produk berhasil ditambahkan.

1 jawaban



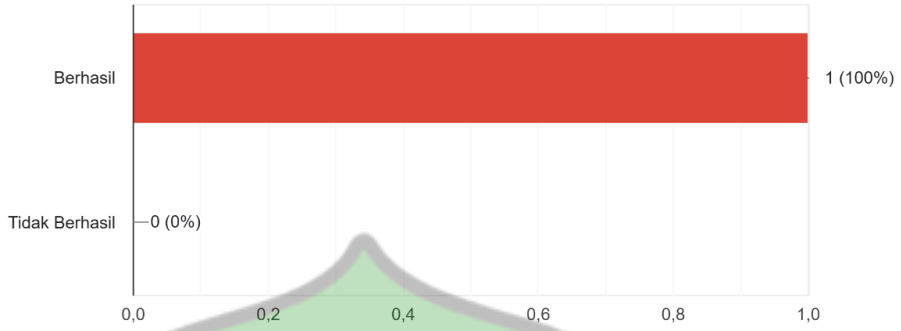
Skenario: Admin mengubah data produk Hasil yang diharapkan: Produk berhasil diperbarui.

1 jawaban



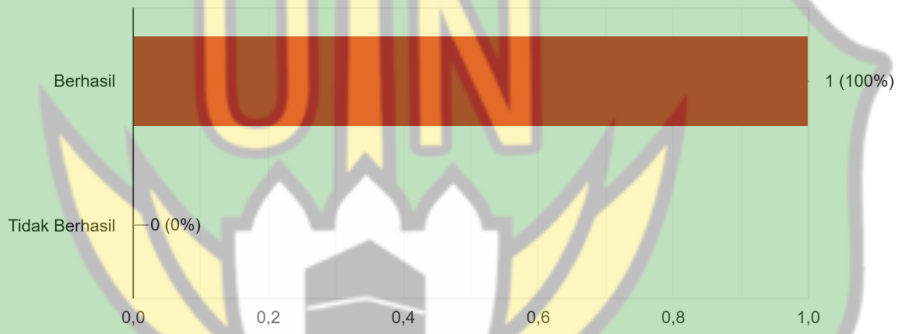
Skenario: Admin menghapus produk Hasil yang diharapkan: Produk berhasil dihapus.

1 jawaban



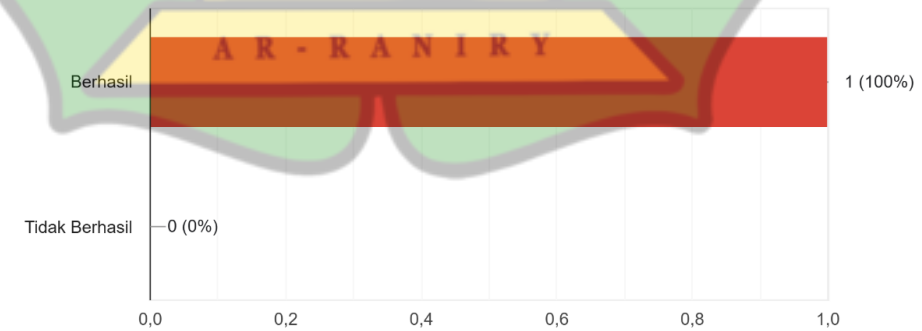
Skenario: Pelanggan menambah produk ke keranjang Hasil yang diharapkan: Produk berhasil ditambahkan ke keranjang.

1 jawaban



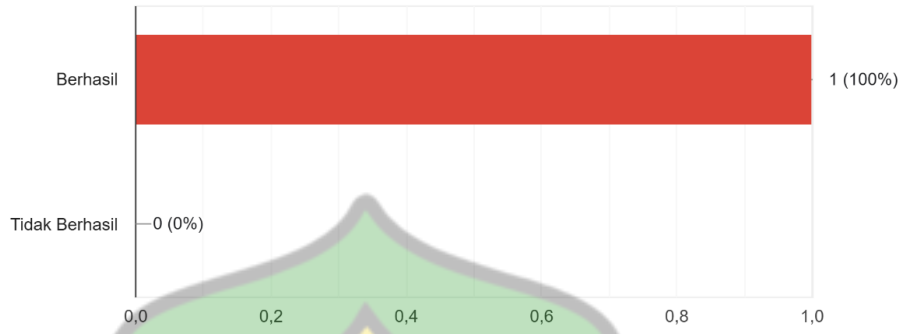
Skenario: Menampilkan daftar belanja Hasil yang diharapkan: Daftar produk pada daftar belanja berhasil ditampilkan.

1 jawaban



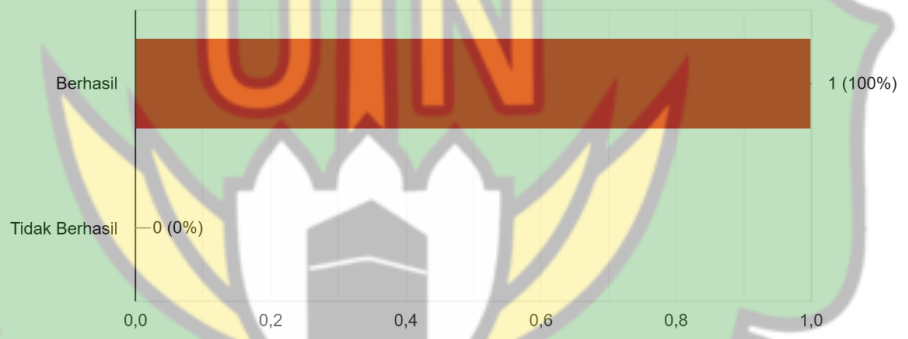
Skenario: Menambahkan produk ke daftar belanja Hasil yang diharapkan: Produk berhasil ditambahkan ke daftar belanja.

1 jawaban



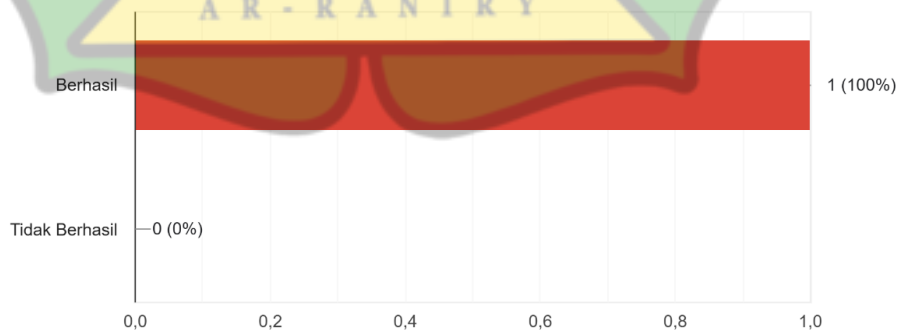
Skenario: Mengubah jumlah produk pada daftar belanja Hasil yang diharapkan: Jumlah produk berhasil diperbarui.

1 jawaban



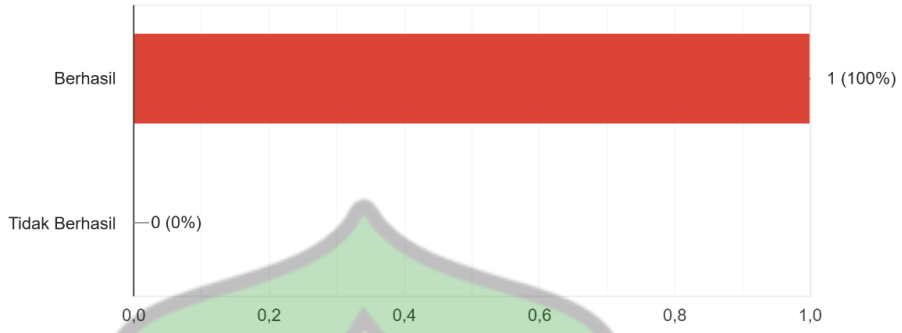
Skenario: Menghapus produk dari daftar belanja Hasil yang diharapkan: Produk berhasil dihapus dari daftar belanja.

1 jawaban



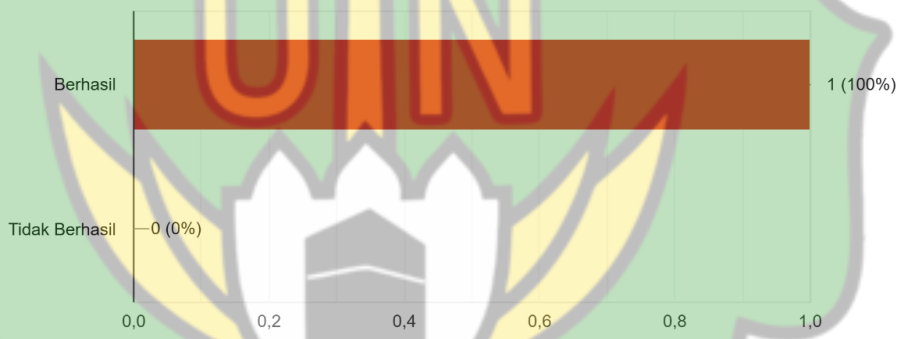
Skenario: Menambahkan seluruh produk ke keranjang Hasil yang diharapkan: Seluruh produk pada daftar belanja berhasil dipindahkan ke keranjang belanja.

1 jawaban



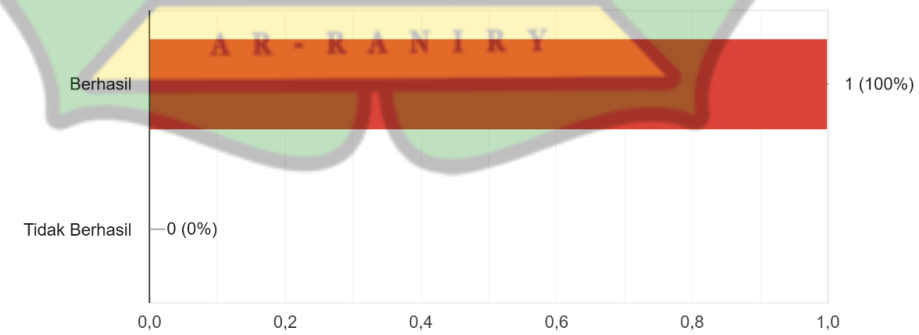
Skenario: Menampilkan isi keranjang Hasil yang diharapkan: Seluruh produk dalam keranjang berhasil ditampilkan.

1 jawaban



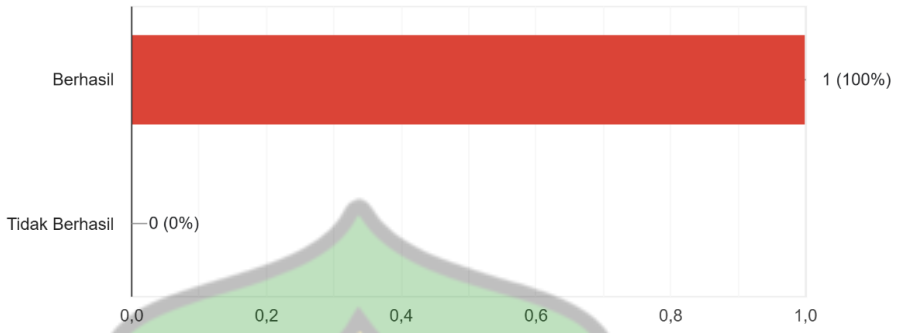
Skenario: Mengubah jumlah produk Hasil yang diharapkan: Jumlah produk berhasil diperbarui.

1 jawaban



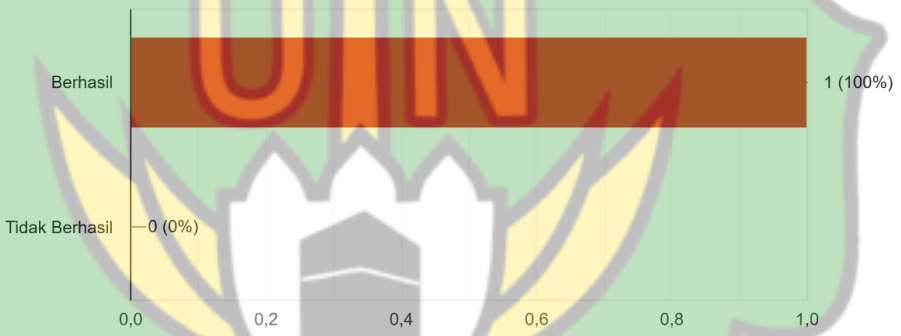
Skenario: Menghapus produk dari keranjang Hasil yang diharapkan: Produk berhasil dihapus dari keranjang.

1 jawaban



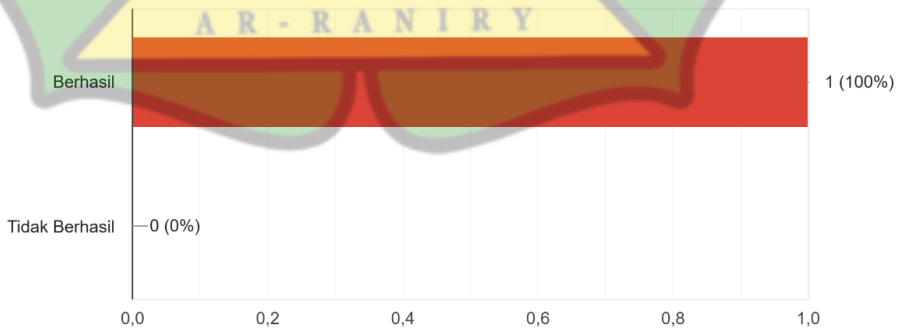
Skenario: Checkout dengan produk dan alamat yang valid Hasil yang diharapkan: Pesanan berhasil dibuat, stok diperbarui otomatis, dan status menjadi Diproses.

1 jawaban



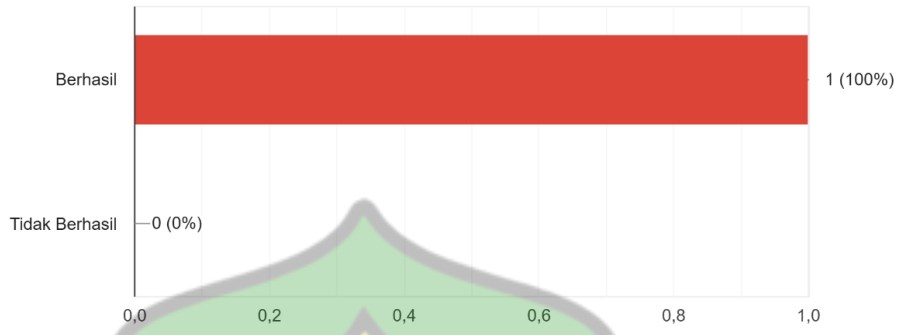
Skenario: Menampilkan daftar pesanan Hasil yang diharapkan: Daftar pesanan berhasil ditampilkan.

1 jawaban



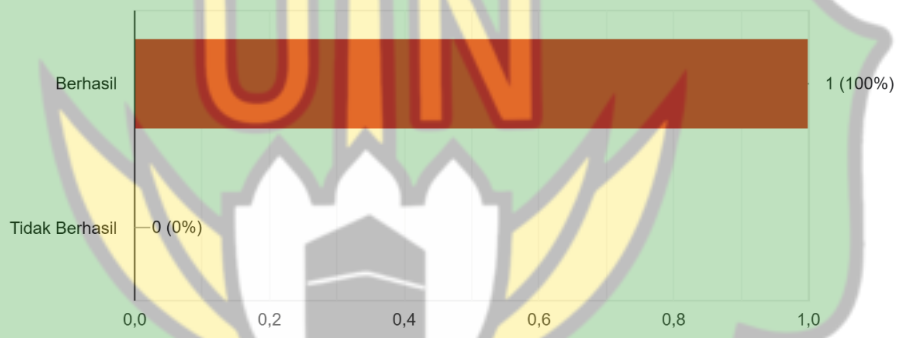
Skenario: Admin mengubah status menjadi Dikirim Hasil yang diharapkan: Status berubah menjadi Dikirim.

1 jawaban



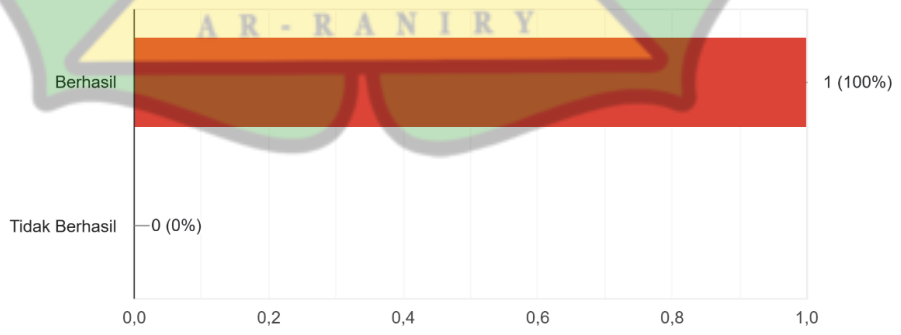
Skenario: Admin mengubah status menjadi Selesai Hasil yang diharapkan: Status berubah menjadi Selesai.

1 jawaban



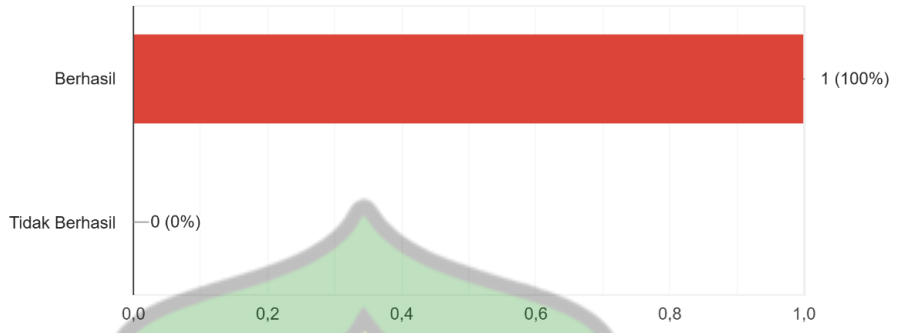
Skenario: Admin mengubah status menjadi Dikirim dan Selesai tanpa checklist item pesanan terlebih dahulu Hasil yang diharapkan: Sistem m...kan pesan bahwa item pesanan belum ter-checklist.

1 jawaban



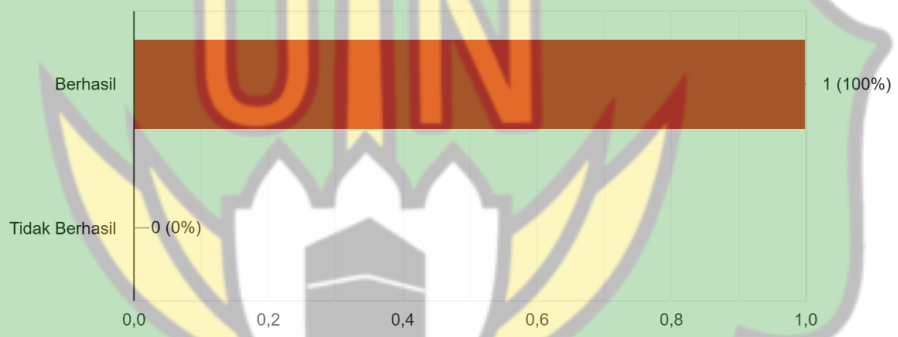
Skenario: Scan barcode produk Hasil yang diharapkan: Produk berhasil ditemukan dan ditambahkan ke kasir.

1 jawaban



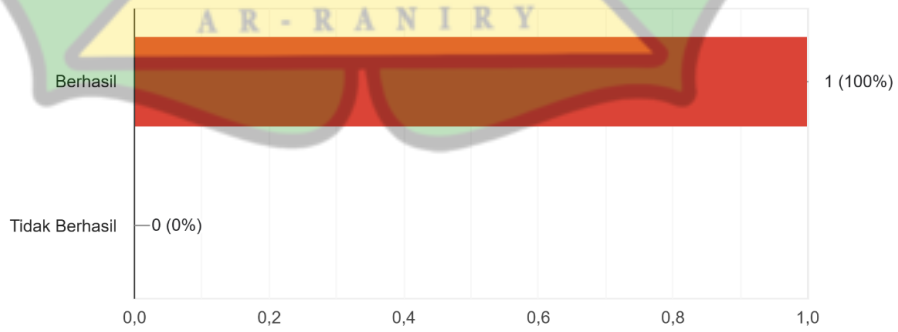
Skenario: Menambah produk secara manual Hasil yang diharapkan: Produk berhasil ditambahkan ke kasir.

1 jawaban



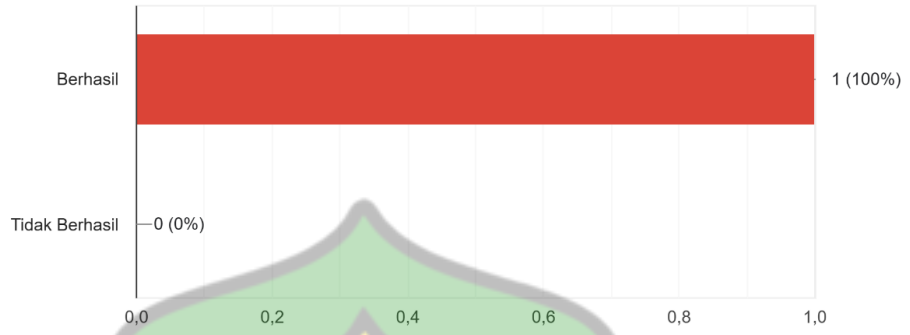
Skenario: Melakukan transaksi penjualan Hasil yang diharapkan: Transaksi berhasil disimpan dan stok berhasil diperbarui.

1 jawaban



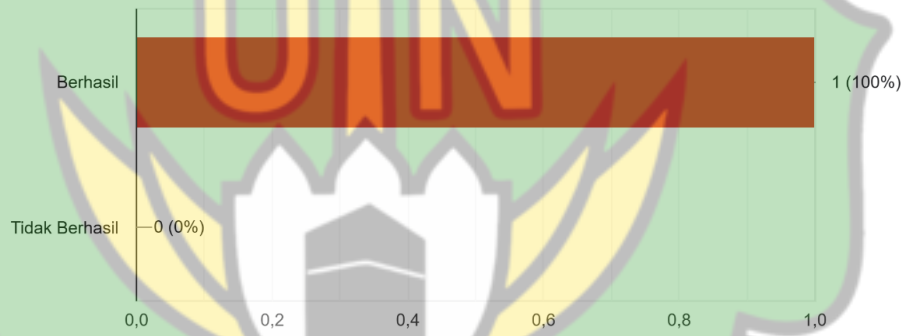
Skenario: Menampilkan laporan harian Hasil yang diharapkan: Data laporan harian berhasil ditampilkan.

1 jawaban



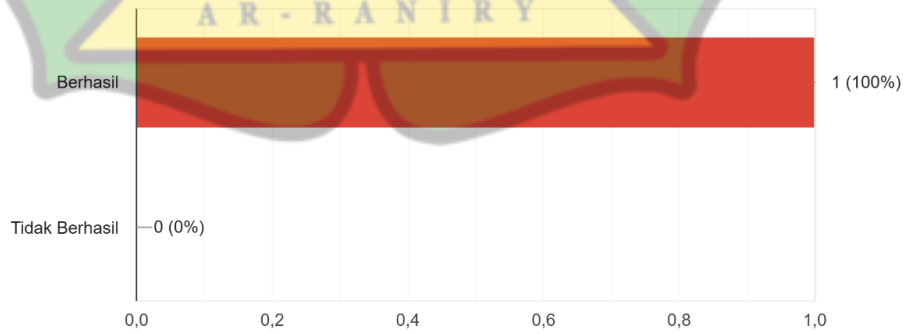
Skenario: Menampilkan laporan bulanan Hasil yang diharapkan: Data laporan bulanan berhasil ditampilkan.

1 jawaban



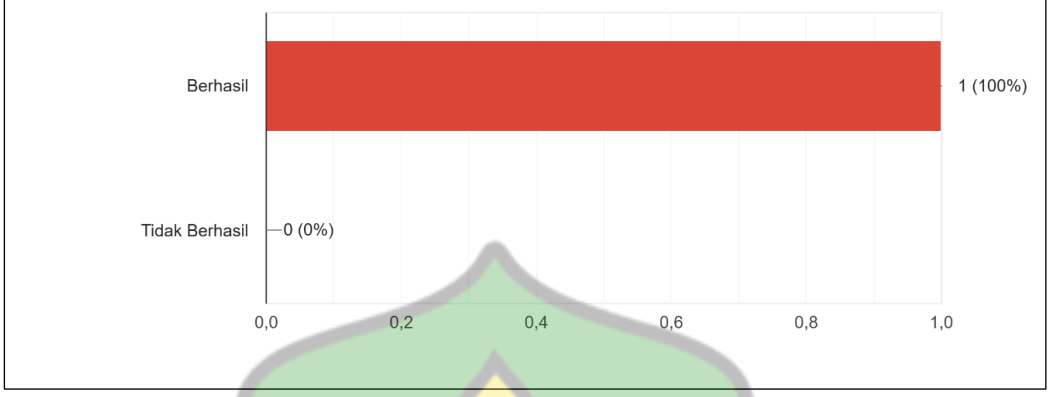
Skenario: Menampilkan laporan tahunan Hasil yang diharapkan: Data laporan tahunan berhasil ditampilkan.

1 jawaban



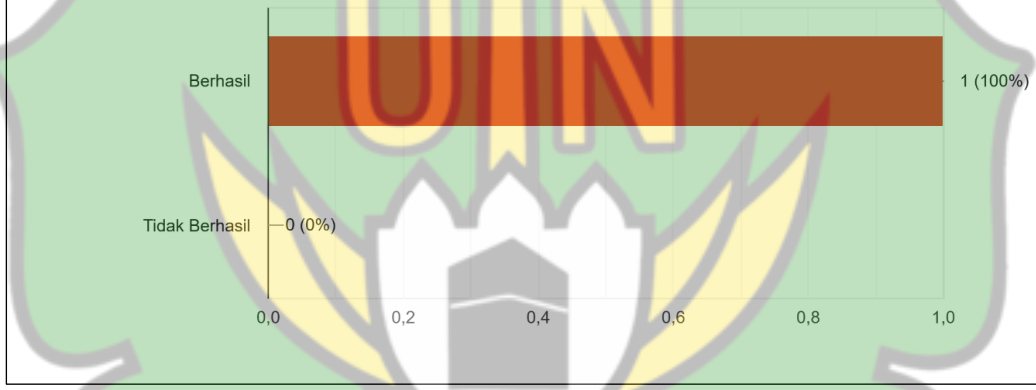
Skenario: Menampilkan profil pengguna Hasil yang diharapkan: Data profil berhasil ditampilkan.

1 jawaban



Skenario: Mengubah foto profil Hasil yang diharapkan: Foto profil berhasil diunggah dan ditampilkan pada profil.

1 jawaban



Lampiran 3. Pertanyaan Pengujian System Usability Scale (SUS)

Kuesioner untuk Mengukur System Usability Scale (SUS) Aplikasi Stokita

Deskripsi formulir

Nama *

Teks jawaban singkat

Saya merasa akan sering menggunakan sistem ini *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya merasa sistem ini terlalu rumit *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya merasa sistem ini mudah digunakan *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya merasa perlu bantuan teknis untuk menggunakan sistem ini *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Fitur-fitur dalam sistem ini terintegrasi dengan baik *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya merasa sistem ini memiliki terlalu banyak inkonsistensi *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya membayangkan kebanyakan orang akan cepat belajar menggunakan sistem ini *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Saya merasa sistem ini terlalu membingungkan saat digunakan *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

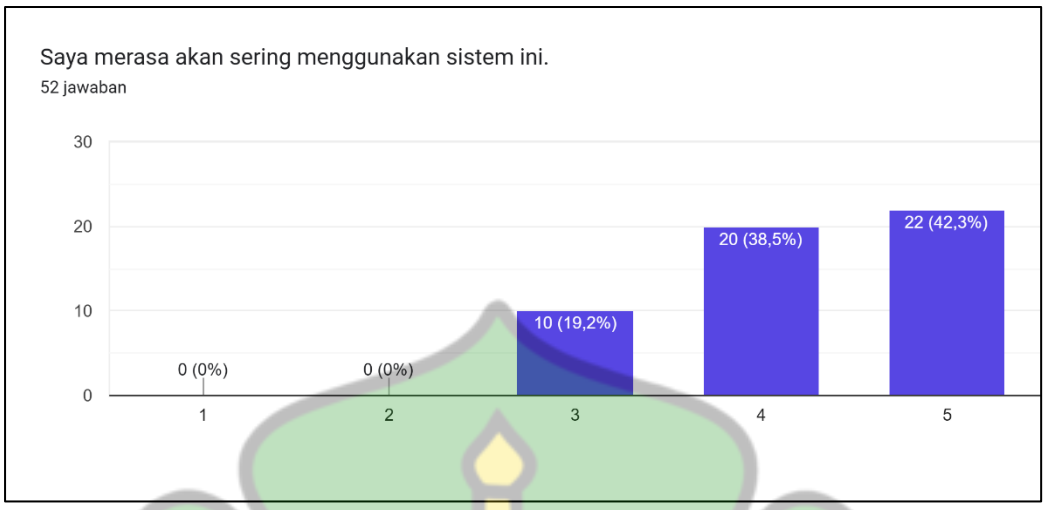
Saya merasa percaya diri saat menggunakan sistem ini *

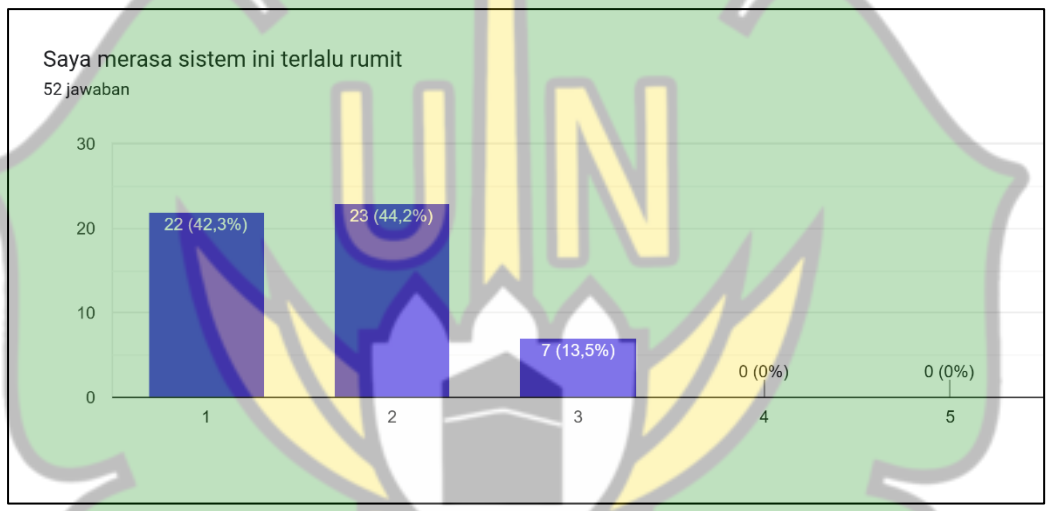
	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

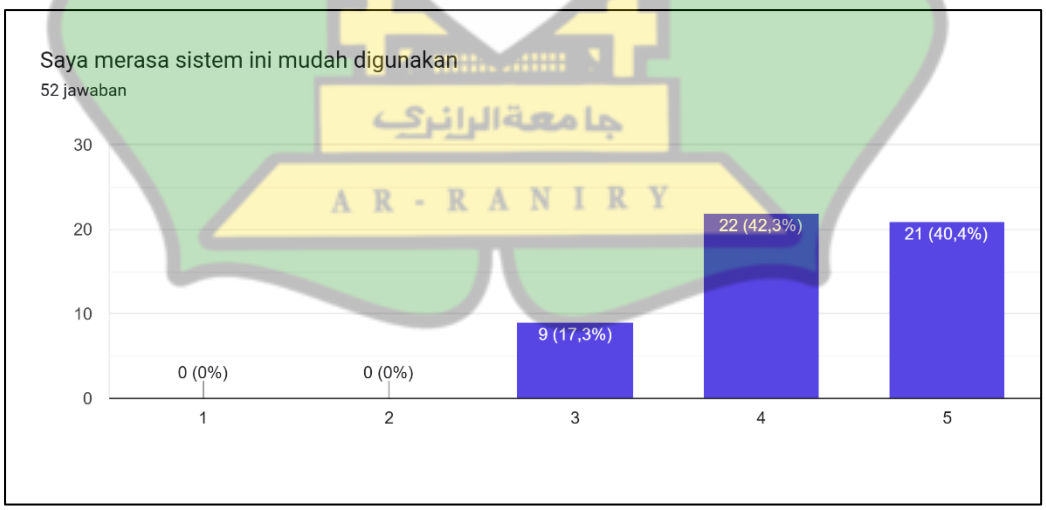
Saya harus banyak belajar sebelum dapat menggunakan sistem ini *

	1	2	3	4	5	
Sangat Tidak Setuju	○	○	○	○	○	Sangat Setuju

Lampiran 4. Jawaban Pengujian System Usability Scale (SUS)

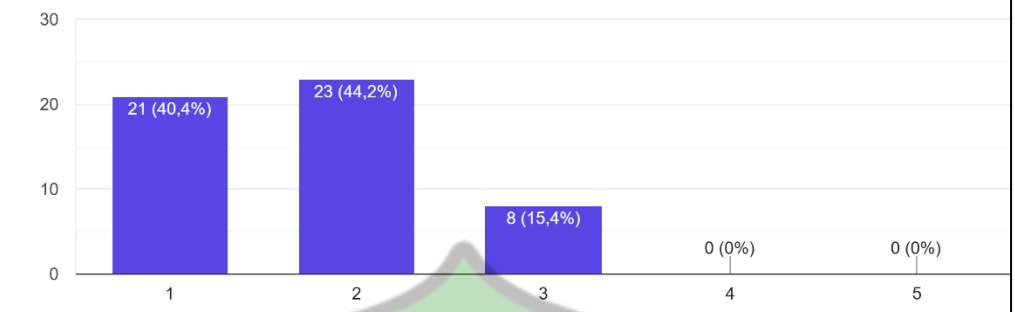






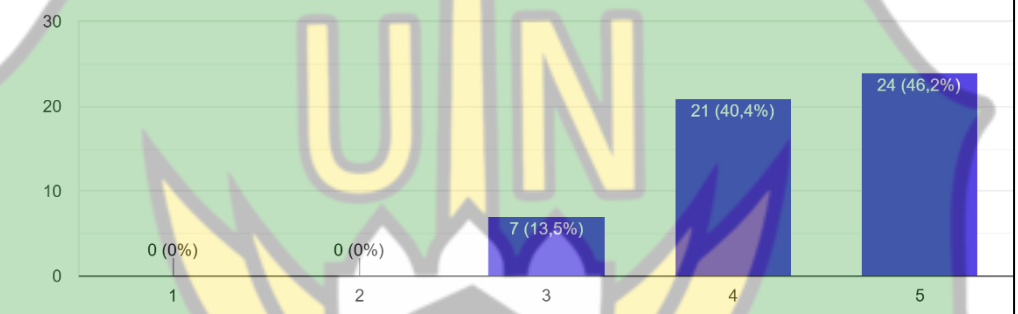
Saya merasa perlu bantuan teknis untuk dapat menggunakan sistem ini.

52 jawaban



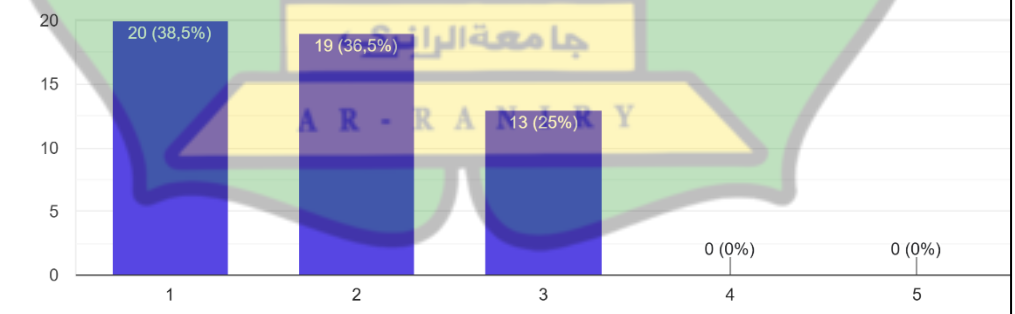
Fitur-fitur dalam sistem ini terintegrasi dengan baik

52 jawaban



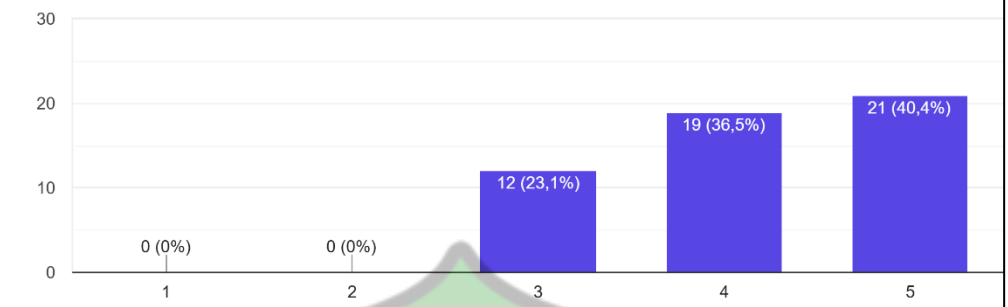
Saya merasa sistem ini memiliki terlalu banyak inkonsistensi

52 jawaban



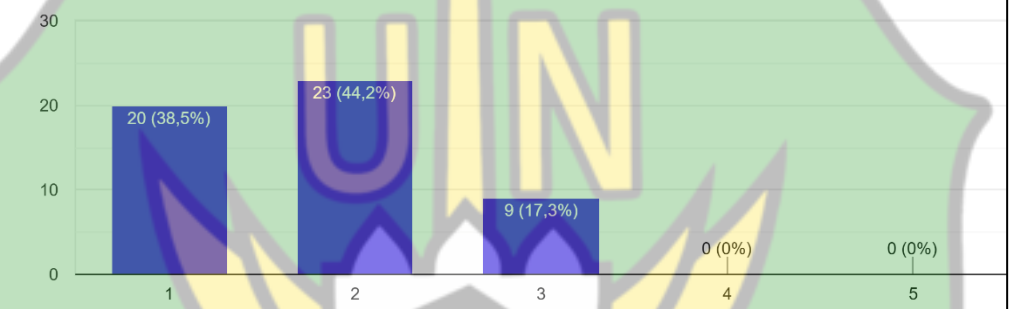
Saya membayangkan kebanyakan orang akan cepat belajar menggunakan sistem ini

52 jawaban



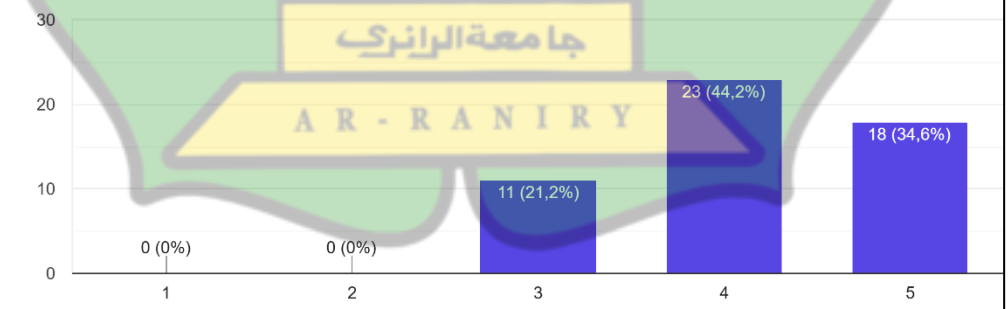
Saya merasa sistem ini terlalu membingungkan saat digunakan.

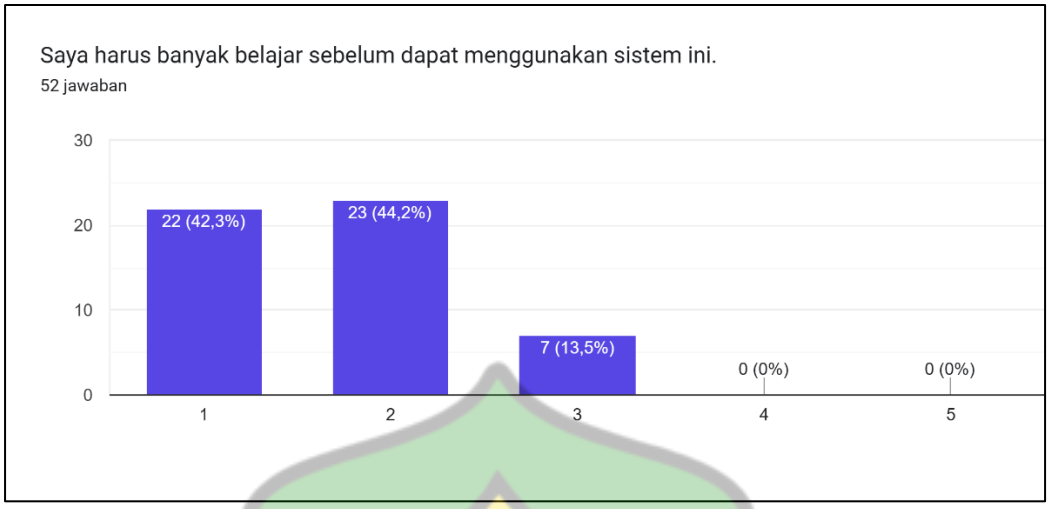
52 jawaban



Saya merasa percaya diri saat menggunakan sistem ini.

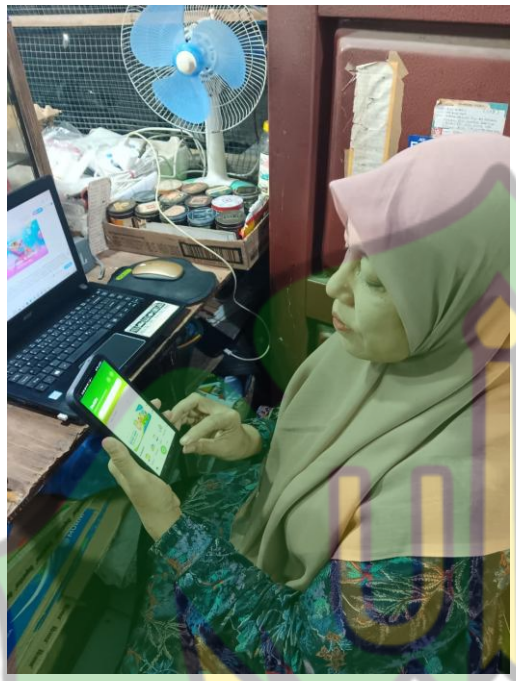
52 jawaban

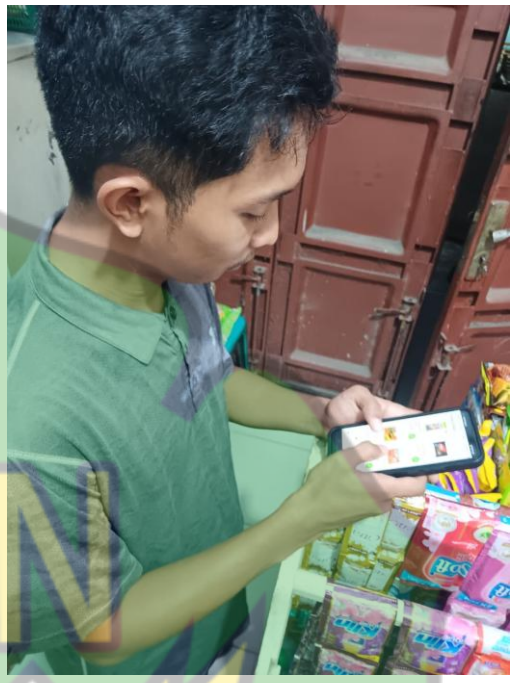




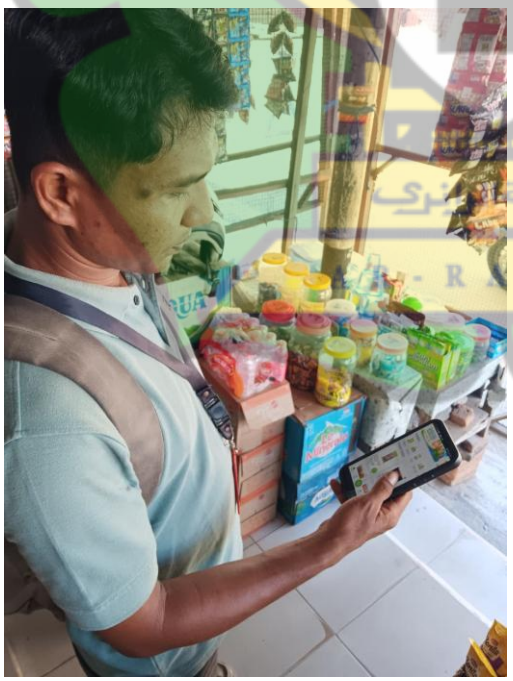
Lampiran 5. Dokumentasi Pelaksanaan Pengujian Usability Menggunakan System Usability Scale (SUS)

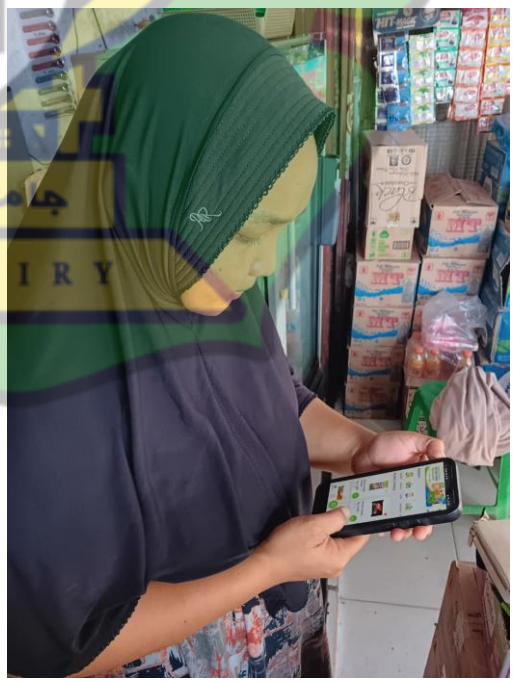
Admin





Pelanggan





Pelanggan