Rancang Bangun Aplikasi *Mobile* untuk Usaha *Laundry* dengan Metode *User Centered Design* (UCD)

Disusun Oleh:

N a m a : Ricky Reza Ananda
NIM : 18523144

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

Rancang Bangun Aplikasi *Mobile* Untuk Usaha *Laundry* dengan Metode *User Centered Design* (UCD)

TUGAS AKHIR

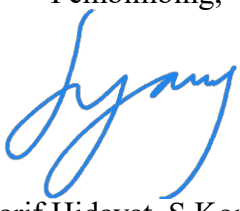
Disusun Oleh:

N a m a : Ricky Reza Ananda

NIM : 18523144

Yogyakarta, 19 Juni 2025

Pembimbing,



(Dr. Syarif Hidayat, S.Kom., M.I.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

Rancang Bangun Aplikasi *Mobile* Untuk Usaha *Laundry* dengan Metode *User Centered Design* (UCD)

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 19 Juni 2025

Tim Penguji

Dr. Syarif Hidayat S.Kom, M.I.T,

Dosen Penguji 1

Sri Mulyati, S.Kom., M.Kom.

Dosen Penguji 2

Dr. Novi Setiani, S.T., M.T.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ricky Reza Ananda

NIM : 18523144

Tugas akhir dengan judul:

Rancang Bangun Aplikasi *Mobile* Untuk Usaha *Laundry* dengan Metode *User Centered Design* (UCD)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 19 Juni 2025

(Ricky Reza Ananda)

HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan ke hadirat Allah Swt. yang telah memberikan kekuatan, kesehatan, kesabaran, serta kelancaran selama proses penyusunan skripsi ini. Tanpa izin dan rahmat-Nya, mungkin penyusunan skripsi ini tidak akan sampai pada tahap akhir.

Dengan penuh rasa hormat dan cinta, karya ini penulis persembahkan pertama-tama kepada kedua orang tua tercinta, Bapak Toha dan Ibu Suparni yang selalu memberikan kasih sayang, dukungan moral, serta doa yang tak pernah putus dalam setiap langkah kehidupan penulis. Terima kasih atas segala pengorbanan, motivasi, dan kepercayaan yang telah diberikan sehingga penulis dapat menyelesaikan penyusunan laporan skripsi ini. Terima kasih saya ucapkan kepada Bapak Syarif Hidayat selaku dosen pembimbing yang telah sabar dan konsisten membimbing, memberikan masukan yang membangun, serta menjadi tempat diskusi yang sangat berharga dalam proses penyusunan skripsi ini. Terima kasih juga untuk teman-teman yang telah memberikan doa, semangat dan dukungan kalian, terima kasih sudah menjadi bagian penting dalam proses ini.

Terakhir, karya ini persembahkan untuk diri sendiri. Atas segala usaha, waktu, dan semangat yang telah dicurahkan dalam menyelesaikan proses panjang ini. Semoga skripsi ini dapat menjadi bukti bahwa ketekunan dan dedikasi akan selalu membuahkan hasil. Lebih dari itu, penulis berharap karya ini dapat memberikan manfaat bagi siapa pun yang membacanya, menjadi tambahan wawasan, serta memberikan kontribusi positif dalam dunia pendidikan dan pengembangan ilmu pengetahuan.

HALAMAN MOTO

"life is the same as a festival. happiness lasts for fleeting moments, leaving behind some snapshot images of the moments. enjoy life and seize the fleeting moments".

— Kim Namjoom

KATA PENGANTAR

Puji dan puja Syukur saya panjatkan kepada Allah Swt. atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul "Rancang Bangun Aplikasi *Mobile* untuk Usaha *Laundry* dengan Metode *User-Centered Design* (UCD)". Skripsi ini disusun sebagai salah satu syarat untuk menyelesaikan studi pada Program Sarjana Informatika di Fakultas Teknologi Industri, Universitas Islam Indonesia.

Penyusunan skripsi ini tentu bukanlah hal yang mudah dan tidak terlepas dari berbagai tantangan. Namun berkat doa, dukungan, serta bantuan dari berbagai pihak, penulis dapat menyelesaikannya dengan baik. Oleh karena itu, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah Swt, atas segala rahmat, hidayah, dan kemudahan yang diberikan sehingga skripsi ini dapat diselesaikan.
2. Kedua orang tua tercinta yang selalu memberikan doa, dukungan dan semangat yang tak terhingga.
3. Bapak Dr. Syarif Hidayat S.Kom, M.I.T, selaku dosen pembimbing penulis yang menjadi tempat berdiskusi dan memberikan arahan serta masukan dalam penyusunan skripsi ini.
4. Ibu Arrie Kurniawardhani S.Si, M.Kom selaku dosen pembimbing akademik yang selalu mengingatkan dan memberikan arahan selama masa pembelajaran.
5. Seluruh dosen Program Sarjana Informatika yang telah memberikan ilmu, wawasan, dan pengalaman berharga selama masa pembelajaran.
6. Teman-teman Discord AVOCADO, yang menjadi ruang berbagi cerita, diskusi serta sumber semangat dan masukan yang sangat membantu penulis menyelesaikan skripsi ini.
7. Channel Youtube JavaScript Mastery yang telah menyediakan referensi vidio mengenai pembuatan aplikasi dan menjadi acuan penting dalam proses pengembangan aplikasi pada penelitian ini.

Akhir kata, penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa yang akan datang. Semoga skripsi ini dapat memberikan manfaat dan kontribusi positif, baik bagi penulis sendiri maupun bagi pembaca dan pihak lain yang berkepentingan.

Yogyakarta, 19 Juni 2025

(Ricky Reza Ananda)

SARI

Usaha laundry merupakan sebuah usaha yang umum ditemukan di sekitar tempat kita tinggal, usaha ini termasuk kedalam usaha mikro, kecil dan menengah (UMKM). Dalam pengoperasiannya, sebuah usaha akan melakukan pembukuan yang berguna untuk memonitori arus keuangan usaha tersebut. Dengan adanya kemajuan teknologi, kegiatan pembukuan bisa dilakukan dengan menggunakan smartphone kita. Penelitian ini bertujuan untuk merancang dan mengembangkan sebuah aplikasi mobile berbasis Android yang ditujukan untuk membantu pelaku usaha laundry dalam mengelola operasional bisnis, khususnya pencatatan keuangan dan sistem kasir. Pengembangan aplikasi dilakukan dengan pendekatan *User-Centered Design (UCD)* yang berfokus pada kebutuhan dan kenyamanan pengguna melalui tahapan analisis, perancangan, evaluasi prototipe, hingga implementasi.

Aplikasi ini dibangun menggunakan React Native dengan bantuan Expo agar dapat berjalan di platform Android. Dengan fitur-fitur seperti pencatatan transaksi, pengelolaan invoice, dan visualisasi laporan keuangan dalam bentuk grafik, aplikasi ini diharapkan dapat menjadi solusi digital yang memudahkan pelaku usaha *laundry* dalam mencatat dan memahami kondisi keuangan bisnis secara efisien. Evaluasi awal dilakukan pada tahap purwarupa dengan metode pengujian usabilitas. Hasil pengujian menunjukkan tingkat keberhasilan pengguna dalam menjalankan tugas sebesar 84% dan *error rate* sebesar 16%. Selain itu, penilaian menggunakan *System Usability Scale (SUS)* menghasilkan skor sebesar 75,5, yang mengindikasikan bahwa desain purwarupa tergolong baik dan layak untuk dikembangkan lebih lanjut. Pengujian *black-box* terhadap fitur utama seperti autentikasi, pengelolaan transaksi, dan tampilan dashboard. Hasilnya menunjukkan bahwa sebagian besar fungsi berjalan dengan baik sesuai kebutuhan pengguna.

Kata kunci: Aplikasi mobile, sistem kasir, UCD, pengujian usabilitas, blackbox

GLOSARIUM

API	*Application Programming Interface* atau antarmuka pemogramman aplikasi sebagai penghubung atau alat komunikasi perangkat lunak untuk bertukar data satu sama lain.
Backend	Bagian belakang layar sebuah aplikasi yang bertanggung jawab untuk pengelolaan dan pemrosesan data.
ERD	Entity Relationship Diagram adalah diagram penggambaran hubungan antar entitas dalam suatu basis data.
Frontend	Tampilan antarmuka pengguna dari suatu aplikasi atau perangkat lunak yang digunakan pengguna untuk berinteraksi dengan sistem.
High fidelity	Tampilan desain yang mirip dengan produk akhir baik secara visual dan fungsional.
Low fidelity	Tampilan sederhana sebuah aplikasi untuk menyampaikan ide dasar sebuah sistem.
Sitemap	Representasi visual struktur navigasi sebuah aplikasi atau sistem.
UCD	*User-Centered Design* dikenal metode pengembangan sistem yang berfokus kepada keperluaan dan keinginan pengguna.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Usaha mikro, kecil dan menengah atau yang biasa disingkat menjadi UMKM merupakan usaha produktif yang terbagi menjadi tiga jenis. Berdasarkan dari Undang-Undang Republik Indonesia Tahun 2008, UMKM terbagi menjadi tiga jenis yaitu usaha mikro, usaha kecil dan usaha menengah (Undang-Undang Republik Indonesia, 2008). Pembagian tersebut didasarkan pada definisi dan kriteria yang ditetapkan oleh peraturan perundang-undangan. Usaha mikro merupakan usaha produktif milik perorangan atau badan usaha perorangan yang memiliki kekayaan bersih sekitar 50 juta tidak termasuk bangunan dan tanah tempat usaha dan memiliki penjualan paling banyak 300 juta per tahun. Sementara usaha kecil merupakan usaha ekonomi produktif yang berdiri sendiri, yang dilakukan oleh orang perorangan atau badan usaha yang bukan merupakan anak atau cabang perusahaan yang dimiliki, dikuasai dan atau menjadi bagian baik langsung maupun tidak langsung dari usaha menengah atau besar. Usaha kecil memiliki kriteria sebagai berikut memiliki kekayaan bersih lebih dari 50 juta sampai dengan paling banyak 500 juta tidak termasuk bangunan dan tanah tempat usaha dan memiliki hasil penjualan tahunan lebih dari 300 juta sampai dengan 2,5 milyar. Dan yang terakhir untuk usaha menengah merupakan usaha ekonomi produktif yang berdiri sendiri, yang dilakukan oleh orang perorangan atau badan usaha yang bukan merupakan anak atau cabang perusahaan yang dimiliki, dikuasai atau menjadi bagian baik langsung maupun tidak langsung dengan usaha kecil atau usaha besar. Dengan kriteria memiliki kekayaan bersih lebih dari 50 juta hingga 10 milyar rupiah tidak dengan bangunan dan tanah tempat usaha, dan memiliki hasil penjualan tahunan lebih dari 2,5 milyar hingga 50 milyar rupiah.

Menurut data dari Kamar dagang dan Industri Indonesia atau Kadin, pada tahun 2023 jumlah pelaku usaha UMKM sekitar 66 juta, jumlah ini mengalami peningkatan dari tahun sebelumnya sebesar 1,52%(Kamar Dagang dan Industri Indonesia, 2023). Struktur UMKM didominasi oleh usaha mikro dengan proporsi sebesar 99%, sementara usaha kecil menempati 0,3% dan usaha menengah sebesar 0,06%. Komposisi tersebut berbanding lurus dengan kontribusi UMKM terhadap perekonomian nasional, dimana kontribusi UMKM mencapai 61% dari pendapatan Produk Domestik Bruto (PDB) setara 9,850 trilliun rupiah. Selain itu sektor UMKM juga berperan signifikan dalam penyerapan tenaga kerja, dengan menampung sekitar 117 juta pekerja, yang setara dengan 97% dari total tenaga kerja nasional.

Tidak hanya berkontribusi besar dalam penyerapan tenaga kerja, usaha mikro, kecil dan menengah (UMKM) dihadapkan pada tantangan transformasi digital yang kompleks. Keterbatasan sumber daya yang dimiliki, dan miniminya pengetahuan pemilik mengenai teknologi terkini menjadi hambatan utama pelaku UMKM dalam mengadopsi sistem digital. Dari berbagai sektor UMKM yang ada, hampir seluruhnya menghadapi kendala serupa dalam transformasi digital, dengan usaha *laundry* menjadi salah satu contoh konkret di mana keterbatasan teknologi masih sangat terasa dalam pengelolaan bisnis. Dalam pengoperasianya, masih banyak pelaku usaha *laundry* yang menggunakan metode pembukuan manual yang rentan akan kesalahan, tidak efisien dan sulit untuk menghasilkan laporan keuangan yang akurat. Selain pembukuan manual, proses transaksi dan pembuatan nota pada usaha *laundry* masih dilakukan secara konvensional dengan menggunakan tulisan tangan, yang rentan terhadap kesalahan pencatatan, sulit untuk dibaca dan mempersulit proses pengarsipan data transaksi di kemudian hari.

Dari tantangan yang ada, dibutuhkan solusi berupa pengembangan aplikasi dan sistem kasir digital yang dapat mengatasi kompleksitas pengelolaan usaha, memberikan kemudahan dalam pencatatan transaksi dan pelaporan keuangan secara *real-time* dan akurat. Kendala lain yang dihadapi pelaku UMKM adalah keterbatasan akses terhadap perangkat komputer atau laptop dengan sistem operasi Windows, yang memiliki harga relatif mahal. Hal ini menjadikan pengembangan aplikasi berbasis mobile melalui smartphone android sebagai pilihan strategis, mengingat hampir seluruh pelaku UMKM telah memiliki perangkat mobile dengan biaya yang jauh lebih terjangkau, sehingga memudahkan proses digitalisasi usaha *laundry* tanpa mengeluarkan investasi tambahan yang signifikan.

1.2 Rumusan Masalah

Pada penelitian ini terdapat dua rumusan masalah sebagai berikut:

1. Bagaimana merancang dan membangun aplikasi *mobile* sebagai sistem kasir digital untuk membantu pelaku usaha *laundry* dalam menjalankan operasional bisnisnya?
2. Bagaimana pemanfaatan data keuangan pada aplikasi tersebut dapat mendukung pelaku usaha *laundry* dalam menjalankan operasional bisnis secara lebih efisien?

1.3 Batasan Masalah

Terdapat beberapa batasan masalah dalam tugas akhir ini, diantaranya sebagai berikut:

1. Aplikasi *mobile* yang dikembangkan hanya untuk perangkat berbasis sistem operasi android.
2. Pengguna aplikasi dibatasi hanya untuk pemilik usaha *laundry* yang menjalankan semua proses bisnisnya.
3. Fungsionalitas aplikasi hanya mencakup kegiatan pencatatan dan pemantauan keuangan dengan dilengkapi sistem kasir.

1.4 Tujuan Penelitian

Penilitan ini bertujuan untuk mengembangkan sebuah aplikasi *mobile* yang berfungsi sebagai sistem kasir digital bagi pelaku usaha *laundry*, guna memudahkan pencatatan dan pemantauan arus keuangan, serta mendukung operasional bisnis secara efisien melalui pengelolaan transaksi dan penyajian data dalam bentuk yang mudah dipahami.

1.5 Manfaat Penelitian

Adapun manfaat yang ingin dicapai dari penelitian ini adalah sebagai berikut:

1. Bagi pemilik usaha *laundry*: memberikan kemudahan dalam mencatat dan memantau keuangan usaha secara digital, serta membantu mengambil keputusan operasional berdasarkan data yang ditampilkan dalam bentuk grafik.
2. Bagi penulis: menjadikan pengalaman dalam membangun aplikasi *mobile* menggunakan React Native dan Expo, serta menerapkan proses pengembangan aplikasi secara sistematis.

1.6 Metodologi Penelitian

Metode yang digunakan pada penelitian ini adalah metode *User Centered Design* (UCD). Metode ini dipilih karena merupakan metode yang berfokus kepada pengguna aplikasi itu sendiri. Dalam metode ini, terdapat iterasi yang berguna untuk menyesuaikan hasil dengan pendapat pengguna. Sehingga produk yang dihasilkan dapat menyelesaikan permasalahan dari pengguna dan sesuai dengan keinginan penggunanya (Kurnia & Awaludin, 2023).

1.7 Sistematika Penulisan

Sistematika penulisan pada laporan tugas akhir ini terbagi menjadi beberapa bab, Gambaran dari keseluruhan bab tersebut sebagai berikut:

a. BAB I PENDAHULUAN

Pada bab ini menjelaskan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitian, serta metodologi penelitian dan sistematika penulisan.

b. BAB II LANDASAN TEORI

Bab ini membahas mengenai teori-teori dan kajian yang berkaitan atau mendukung pemahaman untuk penilitian yang dijalankan.

c. BAB III METODOLOGI PENELITIAN

Bab ini berisikan mengenai uraian langkah-langah dalam metode yang digunakan untuk membuat aplikasi dalam penelitian ini.

d. BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai hasil dari pengembangan aplikasi, pembahasan fitur-fitur yang diimplementasikan serta pengujian terhadap produk.

e. BAB V KESIMPULAN DAN SARAN

Bab yang terakhir berisikan kesimpulan dari penelitian ini serta saran yang dapat dilakukan untuk penilitian atau pengembangan lebih lanjut di masa mendatang.

BAB II
LANDASAN TEORI

2.1 Proses Bisnis usaha laundry

Usaha *laundry* merupakan jenis usaha jasa yang menyediakan layanan pencucian, pengeringan, penyetrikaan, hingga pengemasan berbagai jenis pakaian. Dalam menjalankan operasionalnya, usaha *laundry* memiliki alur proses bisnis yang relatif standar namun krusial untuk dikelola dengan baik, mulai dari penerimaan order hingga pengembalian pakaian kepada pelanggan. Tahapan proses bisnis *laundry* dijelaskan sebagai berikut:

1. Penerimaan *order*

   Pelanggan datang untuk menyerahkan pakaian, pelayan mencatat data pelanggan, jumlah dan jenis pakaian, serta jenis layanan yang diinginkan. Informasi ini kemudian dicatat dalam sistem atau buku log sebagai transaksi pemasukan.
2. Pemeriksaan dan labeling

   Pakaian diperiksa dari segi kondisi dan diberi label untuk memastikan tidak tertukar antar pelanggan. Pakaian juga diklasifikasikan berdasarkan bahan atau jenis layanan yang dibutuhkan.
3. Proses pencucian dan pengeringan

   Pencucian dilakukan menggunakan mesin dan deterjen sesuai jenis bahan. Setelah dicuci, pakaian dikeringkan menggunakan mesin pengering atau dijemur tergantung jenis layanan.
4. Penyetrikaan dan pengemasan

   Setelah kering, pakaian disetrika, dilipat, dan dikemas. Pakaian yang sudah selesai dikemas akan disimpan hingga diambil/diantar ke pelanggan.
5. Pembayaran

   Pelanggan mengambil pakaian dan melakukan pembayaran sesuai tarif yang telah disepakati saat penyerahan. Pelanggan juga dapat melakukan pembayaran saat penyerahan barang kepada pelayan.

Adapun proses bisnis *laundry* koin yang merupakan usaha *laundry* dengan konsep swalayan (*self-service*), dimana pelanggan melakukan semua aktivitas seperti pencucian dan pengeringan secara mandiri menggunakan mesin otomatis berbasis koin atau token. Berikut adalah tahapan prosesnya:

1. Penerimaan Pelanggan

Pelanggan datang langsung ke tempat usaha laundry dan membawa pakaian kotor yang akan dicuci.

2. Pembelian Token atau Pembayaran

   Pelanggan membeli koin atau token dengan menggunakan uang tunai maupun digital, token atau koin tersebut digunakan untuk mengoperasikan mesin cuci dan mesin pengering.

3. Penggunaan Mesin

   Pelanggan memasukan pakain ke dalam mesin cuci dan menyalakan mesin menggunakan koin yang sudah dibeli. Setelah proses pencucian selesai pelanggan dapat melanjutkan ke mesin pengering dengan tahapan yang sama.

4. Penyelesaian dan Pengambilan

   Setelah proses pengeringan selesai, pelanggan mengambil pakaiannya. Beberapa tempat menyediakan area untuk melipat pakaian.

Selain transaksi yang menghasilkan pemasukan, usaha *laundry* juga memiliki berbagai jenis pengeluaran operasional yang perlu dikelola agar tidak membebani keuangan usaha. Beberapa jenis pengeluaran yang umum meliputi:

1. Gaji karyawan, untuk tenaga kerja yang membantu proses operasional.
2. Listrik dan air, seperti mesin cuci, pengering dan setrika yang membutuhkan konsumsi listrik dan air.
3. Pembelian barang habis, barang kebutuhan laundry meliputi deterjen, pewangi, plastik kemasan dan gas elpiji (setrika uap) yang harus selalu di stok agar tidak kehabisan.

2.2 Aplikasi Bergerak/Mobile

Aplikasi bergerak atau lebih dikenal sebagai *mobile applications* merupakan sebuah perangkat lunak (*software*) yang berjalan pada perangkat genggam khususnya *smartphone*. Mengambil dari namanya aplikasi *mobile* ini didesain dengan sedemikian rupa dengan memanfaatkan efisiensi dan portabilitas perangkat genggam sehingga pengguna dapat mengakses aplikasi dalam berbagai situasi dan aktivitas(Maurits Ivan, 2020).

Teknologi terus berjalan dan berevolusi, bermula dari aplikasi desktop (*Desktop application*). Aplikasi yang dikembangkan dan berjalan pada mesin komputer yang memiliki spesifikasi lebih besar. *Mobile application* melakukan penyederhanaan dan penyesuaian desain sehingga bisa dikatakan simplifikasi dari aplikasi *desktop*. Terlepas dari daya terbatas yang membuat komputasi aplikasi *mobile* lebih rendah kemudahan dan fleksibilitas yang ditawarkan

oleh aplikasi *mobile* lebih besar dibandingkan aplikasi *desktop*. Menggunakan perbandingan *head to head* berikut adalah perbandingan aplikasi *desktop* dan *mobile*:

1. Performa

   Performa aplikasi *desktop* jauh mengungguli aplikasi *mobile* karena orientasi aplikasi *desktop* adalah *resource-intensive task* sedangkan aplikasi *mobile* memiliki performa yang rendah dikarenakan daya dan penyimpanan yang terbatas.

2. Mobilitas dan portabilitas

   Mobilitas dan portabilitas sebuah aplikasi *desktop* jauh lebih rendah karena perangkat komputer yang digunakan memiliki banyak bagian seperti monitor, mouse, keyboard dan CPU. Dilain sisi aplikasi *mobile* jauh lebih tinggi dalam portabilitas dan mobilitas mengingat perangkat yang digunakan bisa digenggam menggunakan satu tangan.

3. Keamanan

   Secara keamanan data kedua belah sisi memiliki sistem pertahanan yang tidak jauh berbeda. Mengandalkan pengamanan kuat dari *third party software* aplikasi *desktop* maupun *mobile* dapat diamankan dengan baik.

4. Waktu penggunaan

   Penggunaan aplikasi *desktop* lebih memakan waktu melihat alur kerja aplikasi *desktop* yang memiliki interaksi lebih banyak dibandingkan dengan aplikasi *mobile* yang dirangcang untuk mengerjakan tugas *on-the-go* sehingga aplikasi *mobile* memiliki sedikit interaksi dan hemat waktu.

5. User experience

   Dengan ukuran layar yang lebih besar aplikasi *desktop* memiliki ruang bergerak dalam menyelesaikan tugas yang sangat luas dan mendetail. Aplikasi *mobile* yang memiliki ukuran layar yang lebih kecil menawarkan kemudahan dalam menyelesaikan tugas secara sederhana.

6. Kompleksitas pengembangan aplikasi

   Pengembangan aplikasi *desktop* dan *mobile* memiliki kompleksitas yang sejajar dengan *scope* sebuah aplikasi dibuat. Semakin besar sebuah target aplikasi dibuat akan semakain banyak syarat unik, alat, petunjuk desain, dan ekspektasi performa yang diinginkan oleh pengembang.

7. Layanan *online* dan *offline*

   Dari kedua belah sisi pemanfaatan sebuah jaringan dalam memaksimalkan penggunaan bergantung kepada desain aplikasi. Ketika berada dalam posisi *online* aplikasi desktop

menawarkan jaringan ethernet dengan kecepatan tinggi sedangkan aplikasi mobile mengandalkan WiFi untuk memaksimalkan mobilitas yang dimilikinya. Aplikasi *desktop* masih mendukung pengoperasian secara *offline* dibandingkan aplikasi *mobile* yang cukup bergantung kepada koneksi internet.

8. Jangkauan pasar

   Pasar aplikasi *desktop* jauh lebih sedikit daripada aplikasi *mobile* dikarenakan perangkat aplikasi *mobile* sangat mudah didapat dengan harga lebih murah daripada perangkat aplikasi *desktop*.

9. Perbaikan dan pembaharuan

   Pada aplikasi *desktop* pembaharuan dan perbaikian aplikasi umumnya dilakukan secara manual sedangkan aplikasi mobile lebih terintegrasi dengan adanya *appstore*.

Tingkat mobilitas tinggi aplikasi *mobile* membuat fungsionalitas aplikasi *mobile* menjadi lebih luas dan dapat memenuhi berbagai kebutuhan pengguna. Berdasarkan tipe, tujuan dan *platform* sebuah aplikasi ditujukan berikut adalah fungsionalitas umum yang dimiliki oleh aplikasi *mobile* (Debon et al., 2019):

1. Basic functionalities, merupakan fungsionalitas dasar sebuah aplikasi *mobile* bekerja dengan ada dan tidaknya internet.
2. Produktifitas dan Kebutuhan Umum, merupakan aplikasi *mobile* dengan fungsionalitas yang menitikberatkan manajemen pengalokasian waktu dan tugas sehingga pengguna dapat memaksimalkan keperluanya dengan efisien.
3. Media Sosial, aplikasi *mobile* dengan fungsionalitas memberikan kemampuan penggunanya dalam mengirimkan pesan, berbagi cerita, dan informasi secara *real-time*.
4. Hiburan dan Media, aplikasi bergerak juga berfungsi sebagai hiburan dengan menawarkan game, streaming dan media hiburan lainya.
5. *Finance* and *banking*, berfungsi mengatur, memperingatkan, dan mencatat semua hal yang berhubungan dengan keuangan.
6. Educational, menawarkan layanan belajar seperti kuis, kursus, *e-learning* dan *interactive tools* dalam meningkatkan kualitas belajar pengguna.
7. E-commerce, memudahkan dalam layanan jual beli secara *online* dan integrasi keuangan.

2.3 *User Centered Design*

User centered design (UCD) adalah sebuah metodologi yang memprioritaskan kebutuhan, kepentingan, dan pengalaman *end-user* dalam seluruh proses desain. Sebagai sebuah *system development cycle, user centered design* merupakan sistem yang sistematis dan terstrukur. Menggunakan lebih tepatnya empat iterasi pengembangan proses dilakukan dengan memahami dengan penuh *end user* pada kebutuhan, tantangan, serta kemauan mereka dalam menggunakan dan berinteraksi dengan produk sehingga hasil pengembangan akan menyelesaikan atau memberikan sebuah solusi yang memuaskan ekspektasi *end-user* (Fadli et al., 2024).

Dalam membentuk pondasi yang membangun seluruh sistem pengembangan setiap siklus iterasi permasalahan, kebutuhan, desain, dan evaluasi memerlukan *end-user feedback* yang kemudian digunakan sebagai bahan peningkatan sistem. Tujuan penggunaan user centered design adalah untuk menciptakan pelayanan yang efektif dan efisien dalam sudut pandang pengguna (Raihan & Hidayahtullah, 2022). Dalam penerapan *user centered design* pondasi pembangun sistem dibuat dengan fleksibel mengacu pada *scope* dan *resource* terbagi menjadi dua yaitu 4 tahapan model dan 6 tahapan model, tetapi ada kalanya sebuah sistem yang didedikasikan dengan maksud dan tujuan tertentu menggunakan sampai 7 tahapan (Petrenko et al., 2021). *Understanding context of use*, *specify user requirement*, *design & evaluation*, dan *implementation* adalah 4 tahapan *user centered design* yang umum digunakan dalam pengembangan sistem (Fadli et al., 2024). Menentukan kebutuhan dan persyaratan dalam mengembangkan teknologi kemudian mengidentifikasi kebutuhan pengguna serta sumber daya adalah tahapan memahami dan menganalisis *context of use* dan *user requirement*. Memasuki perancangan diagram, *flowchart*, *entity relationship diagram*, *wireframe*, dan *high fidelity design* merupakan tahapan *design solution & evaluation*. Menjalankan uji coba dengan menerapkan semua tahapan yang telah dirancang sedemikian rupa adalah tahapan implementasi. *Theory guided planning, adapataions for technology, initiate development, app refinement through focus groups input, beta testing prototypes, feasibulity and measure refinement*, dan *randomized trial* yang diusulkan oleh Petrenko menekankan secara detail dan terfokus dalam masalah yang akan diselesaikan. Menggunakan teori *guided planning* dalam menentukan sebuah roadmap pengembangan sehingga dapat berjalan dengan ekspektasi, *adaptation for technology* sebagai parameter penggunaan teknologi yang berkemungkinan besar diadaptasi dan dikembangkan. *Feasibility and measure refinement* menakar teknologi yang telah diadaptasi dan dikembangkan dengan memperbaiki kekurangan melalui feedback.

Randomized trial digunakan sebagai evaluasi yang menyerupai *usaibilty test* sehingga aplikasi bisa digunakan pada pengguna dalam skala yang ditentukan (Petrenko et al., 2021).

Meskipun dalam tahapan sistem user centered design memiliki ragam jenis berbeda berdasarkan tujuan dan solusi yang dinginkan, inti dari user centered design tetaplah sama dengan mengidentifikasi, memahami, menentukan, mendesain, mengebaluasi dan mengimplementasi dalam menyelesaikan sebuah masalah serta memenuhi ekspektasi end-user melalui feedback yang diberikan seiring berjalanya tahapan.

2.4 Teknologi yang digunakan

Penggunaan teknologi sangat membantu dalam mengerjakan penelitian ini, teknologi yang digunakan dijelaskan sebagai berikut:

2.4.1 Figma

Berperan besar dalam mengembangkan desain *user interface, user experience* dan purwarupa penelitian ini. Figma adalah aplikasi mobile berbasis web dengan kombinasi fungsionalitas web browser dan aplikasi native. Dengan antarmuka yang ramah pengguna, mendukung alur kerja kolaboratif, dan manajemen aset yang terorganisir aplikasi figma sangat populer dikalangan desainer (Agus Muhyidin et al., 2020). Berikut adalah fungsionalitas yang membuat figma menjadi pilihan utama para desainer:

1. Alat desain, fitur desain dasar yang meliputi artboard, frames, grids, layouts, typography, dan pen tool mendukung kreativitas pengguna dalam membuat desain.
2. *Componen-based design,* fitur dimana pengguna dapat membuat desain yang bisa dialokasikan dan disinkron kan antar artboard membuat proses pengembangan menjadi lebih efisien.
3. Purwarupa interaktif, menggunakan setiap komponen yang sudah ada figma memberikan fitur interactive component, transition, animation, dan device preview sehingga pengguna dapat membuat sebuah aplikasi dinamis yang bisa diuji dalam bentuk purwarupa.
4. Kolaborasi *real-time,* memanfaatkan teknologi *cloud* sebagai perantara antar pengguna dalam menyampaikan komentar dalam satu wadah secara *real-time.*
5. Penyimpanan berbasis *cloud*, cloud adalah sebuah ruang penyimpanan online yang bisa diakses selama terhubung jaringan internet. Figma memanfaatkan teknologi

penyimpanan cloud sehingga pengguna bisa mengakses desain di setiap saat dan tidak harus menyimpan data pada penyimpanan lokal.

2.4.2 React

React js adalah sebuah library yang ada dalam bahasa pemrograman javascript yang digunakan untuk membuat sebuah *UI components* dan *user interface*. React js sangat populer dan menjadi framework sebanyak 39.5% diseluruh dunia terhitung 2024 (Vailshery, 2024). Dalam studi yang dilakukan oleh Songtao menggunakan react mampu mengubah sebuah susunan pengembangan web dengan ekosistem yang besar dan fitur virtual DOM membuat performa aplikasi menjadi meningkat dengan signifikan (Chen et al., 2019).

Berikut adalah fitur react js yang bisa meningkatkan performa aplikasi:

1. Components, adalah sebuah *logical reusable part* atau bagian dalam sebuah code yang independen dan dapat digunakan secara berulang sebagai *function.*
2. Jsx, merupakan fitur berupa extension yang memungkinkan pengembang menggunakan javascript dan html dalam satu waktu.
3. Hooks, sebuah fitur yang dikembangkan didalam react js yang memungkin kan pengembang untuk menggunakan react fitur tanpa memanggil sebuah *class.*
4. DOM, *document object model* atau DOM adalah sebuah representasi dari struktur dokumen berupa *tree object*.

2.4.3 Expo

Expo adalah sebuah framework di dalam react native yang digunakan untuk mempermudah langkah pengembangan aplikasi react native menggunakan fitur expo application service. Selain memberikan layanan dan alat dalam membantu pengembang membangun sebuah aplikasi, pengembang juga dapat melakukan build, test dan deploy pada android dan iOS.

Berawal dari keinginan untuk membuat para developer dapat mengembangkan aplikasi dengan environment antar platform yang saling terhubung *cross platform* Charlie Cheever dan Evan Bacon di tahun 2015 pada awal perilisian expo ditujukan sebagai *toolchain* sebuah kumpulan alat untuk memproses software development dalam menyederhanakan pengembangan aplikasi antar platform.

Memanfaatkan *variable environment* adalah hal yang harus diutamakan dalam sebuah alur pembuatan aplikasi. EAS atau expo application service adalah sebuah integrated cloud services yang terintegrasi didalam expo dan react native apps. penggunaan environment variable dalam expo menjadikan data penting dalam sistem aman dan dapat dikonfigurasi serta peluncuran juga bug fix yang lebih ringkas. EAS dilain sisi membuat pengembangan, produksi, dan staging menjadi lebih fleksible karena EAS memiliki variable yang didesain untuk menjaga semua environment menjadi tersinkronisasi (Dziedzic et al., 2024).

2.4.4 Appwrite

Open source back-end-as-a-service(BaaS) yang memberikan sekumpulan *API* seperti *aunthentification, storage management, database, function, messaging* dan *real-time service*. Kumpulan API tersebut digunakan untuk mengurangi waktu pengerjaan pada proses pembuatan layanan *back-end* aplikasi bergerak, web dan *server side-application.* Appwrite juga memberikan kumpulan SDKs seperti Javascript, react, react native flutter, android, dan iOS.

Appwrite didesain tunuk mengerjakan frontend dan SDKs secara seamless sehingga cocok digunakan oleh developer dalam skala kecil. Selain itu berikut keuntungan yang didapat ketika mengadaptasi appwrite (Oberai, 2024):

1. Clean environment, memberikan layanan yang sesuai dengan kebutuhan pengembang.
2. Toolkits, memiliki banyak alat terutama API dalam meringkas proses pengembangan.
3. Ease of Use, appwrite memiliki API dan SDK yang ramah dengan pengguna baru sehingga para pengembang dengan skala kecil hingga besar dapat menggunakan appwrite.
4. Open source, dari komunitas ke komunitas appwrite memberikan kontrol penuh layanan pada pengembang.

2.4.5 Bluestacks

Virtual machine atau mesin virtual yang memproyeksikan sebuah lingkungan baru atau sama dengan induknya dalam satu ekosistem. Bluestacks adalah virtual machine atau dalam skala kecil bisa disebut sebagai emulator yang populer karena bluestacks

memproyeksikan lingkungan android dalam ekosistem windows sehingga aplikasi yang memeiliki format *.apk* bisa berjalan dengan sempurna. Berjalan pertamakali pada tahun 2011 bluestacks juga memiliki penyaing seperti LDPlayer dan NOX.

Dalam menjalankan sebuah virtual machine, perangkat komputer harus memenuhi minimum persyaratan penggunaan dikarenakan pembagian resource yang akan dilakukan virtual machine ketika dijalankan. Berikut adalah syarat minimum dijalankanya bluestacks (Bluestack, 2024):

1. Operating system, windows 7 dan diatasnya.
2. RAM, 2 GB RAM sudah bisa menjalankan bluestacks namun, sebaiknya menggunakan 4 GB RAM dengan ekpektasi penggunakan tidak mengalami crash.
3. Storage, minimal 5GB storage kosong untuk membuat environment android. Belum terhitung dengan aplikasi yang diinstall.
4. Processor, Intel atau AMD dengan single thread benchmark score diatas 500. Dengan ekspektasi penggunaan tidak mengalami crash.
5. Graphic card, Nvidia atau AMD graphic card.

Selayaknya virtual machine pada umumnya bluestacks digunakan sebagai mesin kedua dalam melakukan pengembangan dan menggunakan aplikasi yang memiliki environment berbeda. Dalam penggunaan secara global bluestacks digunakan sebagai berikut:

1. Gaming, dengan mampu menjalankan environment android dalam windows, para *gamer* menjadi lebih memilih bluestacks karena memiliki spesifikasi yang jauh mengungguli smartphone pada umumnya.
2. App testing & development, para pengembang dapat menghubungkan bluestacks dengan berbagai project mereka untuk diuji tanpa harus bergantung pada smartphone.
3. Phone substitue, environment android dalam bluestacks juga bisa menjadi pengganti smartphone android ketika sebuah aplikasi hanya bisa digunakan pada environment android.

2.4.6 Android Studio

Integrated Development Environtment (IDE) merupakan software aplikasi yang memberikan fasilistas, fungsionalitas dan gabungan berbagai macam alat pengembangan pada satu *user interface* untuk mengembangkan sebuah software

aplikasi. Android studio menyediakan lingkungan pengembangan android dilengkapi dengan *code editor, emulator, layout editor, testing, debugging,* dan *deploying* aplikasi android (Android Studio, 2024).

Diperkenalkan secara global oleh google pada tahun 2013 android studio berkembang pesat hingga sekarang. Mendukung bahasa pemrograman kotlin dari mulanya java, compose design tools yang digunakan untuk membantu proses animasi dan layout, intelligent code editor untuk menulis code dengan baik dan cepat disertai live edit untuk melihat perubahan secara langsung, hingga yang terbaru yaitu mengintegrasikan AI yaitu Gemini sebagai *AI Assistant* untuk menjawab hampir semua tantangan dalam pengembangan aplikasi.

2.4.7 Visual Studio Code

Dibuat dan didesain untuk pengembangan software, layanan *open source* gratis visual studio code bisa bekerja secara *cross platform* pada windows, macOS, dan linux. Dirawat dan dikembangkan oleh Microsoft, visual studio code memberikan kebebasan pengembangan ekosistem seluas – luasnya. Dikenalkan kepada dunia pertamakali pada tahun 2015, visual studio code menjadi ramai dibincangkan oleh para developer terutama web developer dan developer kecil karena ukuran yang lebih kecil dari pendahulunya yaitu visual studio performa yang diberikan oleh visual studio code luar biasa kuat.

Di dalam environment visual studio code pengembang dengan bebas menggunakan bahasa yang mereka sukai karena visual studio code mendukung banyak sekali bahasa pemrograman. Hal ini juga menjadikan visual studio code menjadi sangat populer di hampir semua kalangan pengembang aplikasi.

Menjadi IDE kesayangan para developer berikut adalah poin plus menggunakan visual studio code:

1. Ringan dan kencang, lebih ringan dari kakaknya visual studio code mampu berjalan hanya dengan 1 GB ram dan processor dengan benchmark diatas atau sama dengan 200.
2. Open source, dari komunitas ke komunitas visual studio code visual studio code memberikan akses penuh layanan kepada para penggunanya.

3. Extension, merupakan fitur yang sangat diandalkan oleh para pengembang dengan menawarkan customization pada environment visual studio code seluas – luasnya kepada para penggunanya.
4. Integrated Intellisense & AI, hadir sebagai AI asistant memberikan *smart code completion* kepada penggunanya berdasarkan library yang digunakan sehingga penulisan code lebih cepat, rapi dan terstruktur.

2.4.8 Draw.io

Dewasa ini dikenal sebagai diagrams.net, merupakan sebuah alat pembuat diagram online seperti *flowcharts, network diagram, entity relationships diagram, wireframe, UML diagrams, organizational charts,* dan desain yang berhubungan dengan diagram sistem. Layanan gratis dan *open source*, memiliki integrasi secara *cloud,* dan bisa berkolaborasi antar desainer aplikasi berbasis web ini menjadi pilihan dalam membentuk diagram pengembangan sistem.

Serupa dengan figma berikut adalah kelebihan dengan fitur yang ditawarkan oleh draw.io:

1. Web based, memiliki tampilan yang ramah kepada pengguna awam draw.io mengandalkan kefamiliaran pengguna dengan menggabungkan website dan aplikasi desain native. Draw.io juga memiliki aplikasi yang dapat digunakan pada platform windows.
2. Template, memiliki banyak kerangka desain mulai dari umum, lanjut, flowchart, relasi entitas, dan uml membantu penggunanya dalam mempersingkat waktu pengerjaan desain. Selain itu draw.io juga menawarkan fitur berupa shape libraries dengan menghadirkan *pre-designed templates and shapes* yang terus diperbaruhi seiring berkembangnya teknologi.
3. Cloud storage, dengan hadirnya cloud storage pada draw.io membuat proses pengembangan desain menjadi lebih fleksible dan dapat diakses selama terkoneksi dengan jaringan internet.
4. Collaboration, draw.io juga menerapkan fitur yang sama seperti figma dengan mempersilahkan pengguna dengan akun berbeda bergabung mengerjakan sebuah project dalam satu tempat. Collaboration sangat berguna bagi pengembang yang bekerja secara berkelompok dalam satu waktu.

5. Open source, dari komunitas ke komunitas draw.io membebaskan pengguna dalam menggunakan layanan yang diberikanya.

2.5 Review penelitian sejenis

Ada penelitian yang memiliki topik serupa mengenai mendukung aktivitas operasional bisnis pelaku umkm yang sudah dilakukan oleh peneliti terdahulu. Berikut beberapa penelitian tersebut:

1. Merujuk kepada penelitian yang dilakukan oleh Calvin dengan judul "Perancangan UI/UX untuk Aplikasi Bank Jago menggunakan Metode User Centered Design". Metode yang digunakan adalah campuran dengan User Centered Design sebagai metode pendukung pengembangan aplikasi. Dengan tujuan mengolah fitur terkhusus dalam area asuransi yang mempertimbangkan prinsip usability seperti learnability, memorability, dan satisfaction. Hasil dari penelitian ini berupa terbentuknya purwarupa dengan pemenuhan prinsip usability diantaranya learnability 65%, memorability 50%, satisfaction 70%, dan efficiency 55% .
2. Pada penelitian yang dilakukan oleh Ivan dengan judul "Analisis Dan Implementasi Aplikasi Pembukuan Berbasis Android Untuk Memenuhi Kebutuhan Pada Usaha Kecil Menengah". Metode yang digunakan adalah kualitatif. Dengan tujuan membantu pelaku usaha dalam mengelola data transaksi keuangan sehingga informasi ketersediaan barang menjadi lebih terkoordinasi. Hasil dari penelitian ini berupa terbentuknya sebuah aplikasi berbasis android dengan fitur utama digitalisasi pembukuan usaha dan diuji menggunakan analisis PIECES.
3. Penelitan yang dilakukan oleh Eva dengan judul "Pelatihan dan Pembinaan terhadap Pengembangan Usaha Laundry di Kabupaten Tegal". Metode yang digunakan adalah campuran. Dengan tujuan menciptakan sebuah *digital marketing strategy* pada usaha laundry yang ada di kabupaten Tegal. Hasil dari penelitian ini adalah peningkatan pendapatan para pelaku bisnis laundry setelah menggunakan *digital marketing strategy*.
4. Penelitian yang dilakukan oleh Dhandy dengan judul "Perancangan Aplikasi Pembukuan Menggunakan Metode Agile Scrum". Metode yang digunakan adalah campuran dengan Agile Scrum sebagai metode pengembangan aplikasi. Tujuan dalam penelitian ini adalah memaksimalkan pengelolaan produk secara efektif dengan memanfaatkan aplikasi mobile serta membantu analisa penjualan di masa depan. Hasil

dari penelitian ini adalah sebuah aplikasi berbasis android yang memiliki fungsi finance terutama bookeeping dan lolos uji *black-box testing*.

5. Penelitian yang dilakukan oleh Xena dengan judul "Pemanfaatan Software Pembukuan Akuntansi Sebagai Solusi Atas Sistem Pembukuan Manual Pada Umkm". Metode yang digunakan adalah deskriptif dengan bantuan kualitatif pada penembangan aplikasi. Tujuan penelitian ini adalah terciptanya sebuah solusi yang menjawab susahnya pembukuan secara manual. Hasil dari penelitan ini berupa sebuah software berbasis android dengan fungsionalitas finance lengkap dengan panduan pengguna.

Tabel perbandingan antara penelitian terdahulu yang sejenis dengan penelitan yang akan dilakukan dapat dilihat seperti pada Tabel 2.1.

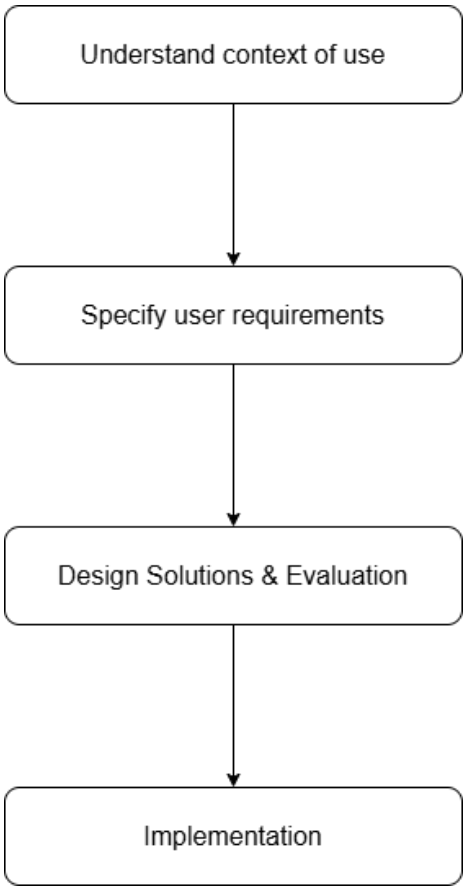
Tabel 2.1 Perbandingan antara penelitian sejenis

	Penelitian 1	Penelitian 2	Penelitian 3	Penelitian 4	Penelitian 5
Penelitian berupa pengembangan aplikasi	✓	✓		✓	
Objek penelitian merupakan pengusaha laundry			✓		
Tujuan penelitan mendigitalisasi proses bisnis umkm			✓		✓
Metode yang digunakan berupa UCD	✓				
Hasil pengembangan berupa aplikasi mobile	✓	✓		✓	

BAB III
METODOLOGI PENELITIAN

3.1 Tahapan penelitian

Dalam penelitian ini, metodologi yang digunakan merupakan metode *User Centered Design* (UCD). Dalam metode tersebut terdapat beberapa tahapan yang dilakukan guna untuk mencapai tujuan penelitian. Tahapan-tahapan tersebut dapat dilihat seperti pada Gambar 3.1



Gambar 3.1 Tahapan metode UCD

Dari gambar di atas, tahapan dimulai dengan *understand context of use*, *specify user requirements*, *design solutions & evaluation* dan yang terakhir *implemantion*. Tahapan-tahapan tersebut dijelaskan sebagai berikut:

3.1.1 *Understand the Context of Use*

Pada tahap awal untuk metode user centered design, penulis harus menentukan produk apa yang akan dibuat, seperti kegunaan produk dan menentukan target pengguna dari produk tersebut. Dalam kasus ini, produk yang akan dibuat merupakan produk yang dapat membantu

pemilik usaha *laundry* dalam kegiatan operasionalnya, seperti proses pencatatan arus keuangan atau pembukuan dan proses pelayanan kasir. Kriteria untuk target pengguna dari aplikasi ini sendiri dapat dilihat seperti yang ada pada Tabel 3.1.

Tabel 3.1 Kriteria target pengguna

Kategori	**Kriteria**
Umur	15+
Jenis Kelamin	Semua jenis
Pekerjaan	Pemilik usaha laundry

3.1.2 *Specify User Requirements*

Dalam tahapan kedua metode *user centered design* untuk pengembangan aplikasi pembukuan dan kasir usaha *laundry*, dilakukan pengumpulan data melalui wawancara terhadap calon pengguna. Wawancara dilakukan guna mendapatkan informasi mengenai kebutuhan dan masalah yang ada pada pengguna. Target narasumber untuk wawancara merupakan pelaku usaha *laundry* baik yang sudah maupun belum menggunakan teknologi untuk membantu dalam pengoperasian bisnisnya. Untuk daftar pertanyaan yang diajukan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Daftar pertanyaan wawancara

No	**Pertanyaan**
1	Apakah anda memiliki sebuah smartphone?
2	Apakah anda mahir dalam menggunakan smartphone?
3	Sudah berapa lama anda menjalankan bisnis anda?
4	Untuk sekarang metode apa yang anda gunakan untuk pembukuan dan pelayanan kasir?
5	Apakah anda sudah pernah mencoba menggunakan aplikasi yang membantu dalam pembukuan/kasir?
6	Jika iya, apakah anda memiliki masalah/keluhan terhadap aplikasi tersebut?
7	Jika anda akan menggunakan aplikasi serupa, fitur apa yang anda inginkan pada aplikasi tersebut?

Wawancara kemudian dilakukan terhadap beberapa narasumber yang merupakan pelaku usaha *laundry*. Sampel hasil wawancara diambil dari 4 narasubmber, hasil dari wawancara dapat dilihat pada Tabel 3.3 hingga Tabel 3.6.

Tabel 3.3 Hasil wawancara narasumber 1

No	Pertanyaan	Respon
1	Apakah anda memiliki sebuah smartphone?	Iya
2	Apakah anda mahir dalam menggunakan smartphone?	Mahir
3	Sudah berapa lama anda menjalankan bisnis anda?	1,5 tahun
4	Untuk sekarang metode apa yang anda gunakan untuk pembukuan dan pelayanan kasir?	Untuk pelayanan kasir menggunakan aplikasi dan pembukuan menggunakan tulis tangan dan Microsoft excel
5	Apakah anda sudah pernah mencoba menggunakan aplikasi yang membantu dalam pembukuan/kasir?	Iya
6	Jika iya, apakah anda memiliki masalah/keluhan terhadap aplikasi tersebut?	Karena menggunakan dua aplikasi berbeda menjadi kurang praktis, dan aplikasi kasir berbayar tiap bulan
7	Jika anda akan menggunakan aplikasi serupa, fitur apa yang anda inginkan pada aplikasi tersebut?	Fitur kasir dan juga pencatatan arus uang dalam satu aplikasi

Tabel 3.4 Hasil wawancara narasumber 2

No	Pertanyaan	Respon
1	Apakah anda memiliki sebuah smartphone?	Iyaa
2	Apakah anda mahir dalam menggunakan smartphone?	Lumayan mahir

3	Sudah berapa lama anda menjalankan bisnis anda?	5 tahun
4	Untuk sekarang metode apa yang anda gunakan untuk pembukuan dan pelayanan kasir?	Masih menggunakan tulis tangan
5	Apakah anda sudah pernah mencoba menggunakan aplikasi yang membantu dalam pembukuan/kasir?	Tidak pernah
6	Jika iya, apakah anda memiliki masalah/keluhan terhadap aplikasi tersebut?	-
7	Jika anda akan menggunakan aplikasi serupa, fitur apa yang anda inginkan pada aplikasi tersebut?	Fitur pencatatan pemasukan dan pengeluaran dan nota online

Tabel 3.5 Hasil wawancara narasumber 3

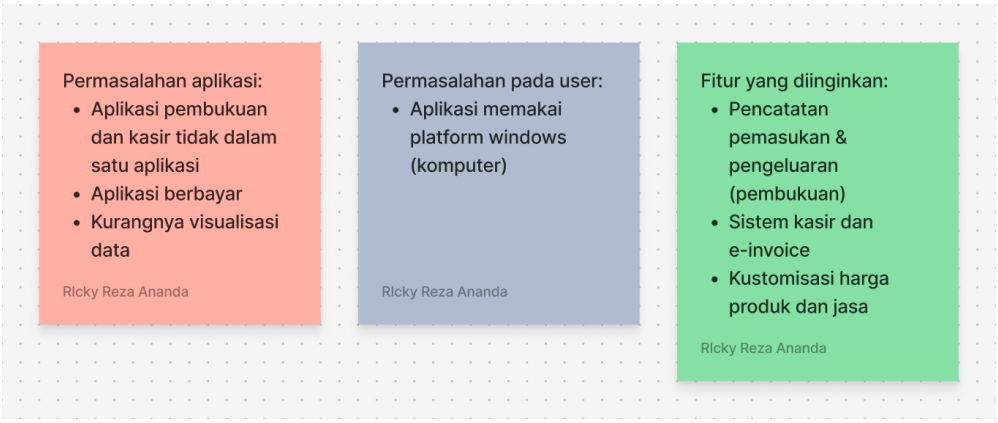
No	**Pertanyaan**	**Respon**
1	Apakah anda memiliki sebuah smartphone?	Iya
2	Apakah anda mahir dalam menggunakan smartphone?	Lumayan mahir
3	Sudah berapa lama anda menjalankan bisnis anda?	3 tahun
4	Untuk sekarang metode apa yang anda gunakan untuk pembukuan dan pelayanan kasir?	Masih menggunakan metode tulis tangan
5	Apakah anda sudah pernah mencoba menggunakan aplikasi yang membantu dalam pembukuan/kasir?	Iya
6	Jika iya, apakah anda memiliki masalah/keluhan terhadap aplikasi tersebut?	Kurang paham dengan alur aplikasi karena terlalu banyak fitur

7	Jika anda akan menggunakan aplikasi serupa, fitur apa yang anda inginkan pada aplikasi tersebut?	Aplikasi yang berfokus pada laundry, untuk fitur seperti pelayan kasir, kustom harga jasa dan pembukuan

Tabel 3.6 Hasil wawancara narasumber 4

No	Pertanyaan	Respon
1	Apakah anda memiliki sebuah smartphone?	Iya
2	Apakah anda mahir dalam menggunakan smartphone?	Mahir
3	Sudah berapa lama anda menjalankan bisnis anda?	2 tahun
4	Untuk sekarang metode apa yang anda gunakan untuk pembukuan dan pelayanan kasir?	Untuk kasir menggunakan aplikasi dan pembukuan menggunakan excel
5	Apakah anda sudah pernah mencoba menggunakan aplikasi yang membantu dalam pembukuan/kasir?	Iya
6	Jika iya, apakah anda memiliki masalah/keluhan terhadap aplikasi tersebut?	Karena menggunakan excel secara normal jadi tidak ada tampilan grafik yang mendukung
7	Jika anda akan menggunakan aplikasi serupa, fitur apa yang anda inginkan pada aplikasi tersebut?	Pelayanan kasir dan fitur pembukuan didukung oleh grafik untuk memudahkan pemahaman

Dari hasil wawancara tersebut dilakukan analisa dan pengelompokan mengenai informasi yang didapatkan. Informasi tersebut dikelompokan seperti yang ada pada Gambar 3.2. Berdasarkan informasi tersebut, maka diputuskanlah aplikasi yang akan dibuat merupakan aplikasi pencatatan arus keuangan dan sistem kasir berbasis *mobile* untuk usaha *laundry* yang dapat digunakan siapa saja secara gratis, dengan didukung visualisasi data menggunakan grafik dan fitur-fitur pendukung seperti kustomisasi harga dan juga *e-invoice*.



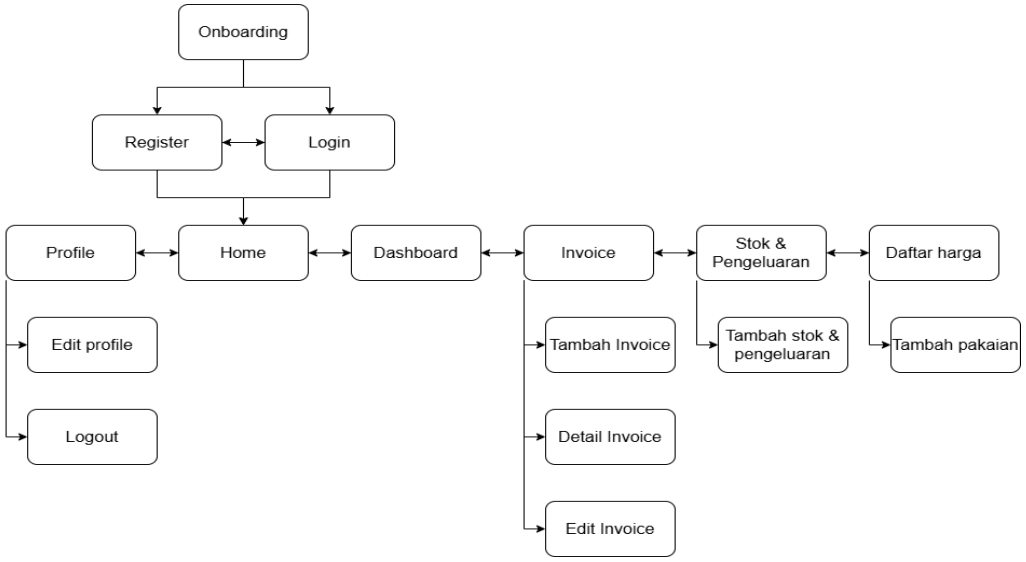
Gambar 3.2 Hasil pengelompokan informasi

3.1.3 *Design Solution & Evaluation*

Dalam tahap ketiga, dilakukan proses perancangan sistem dan desain aplikasi sebagai dasar pengembangan lebih lanjut. Tahap ini diawali dengan perancangan proses bisnis melalui pembuatan *sitemap*, *usecase*, dan *activity* diagram untuk menggambarkan alur dan kebutuhan fungsional aplikasi. Selanjutnya, dilakukan perancangan basis data melalui pembuatan *Entity Relationship Diagram* (ERD), serta perancangan antarmuka pengguna melalui pengembangan purwarupa *low fidelity* dan *high fidelity*. Sebagai bentuk validasi terhadap desain yang telah dibuat, dilakukan pengujian usabilitas pada purwarupa untuk memastikan kesesuaian dengan kebutuhan pengguna.

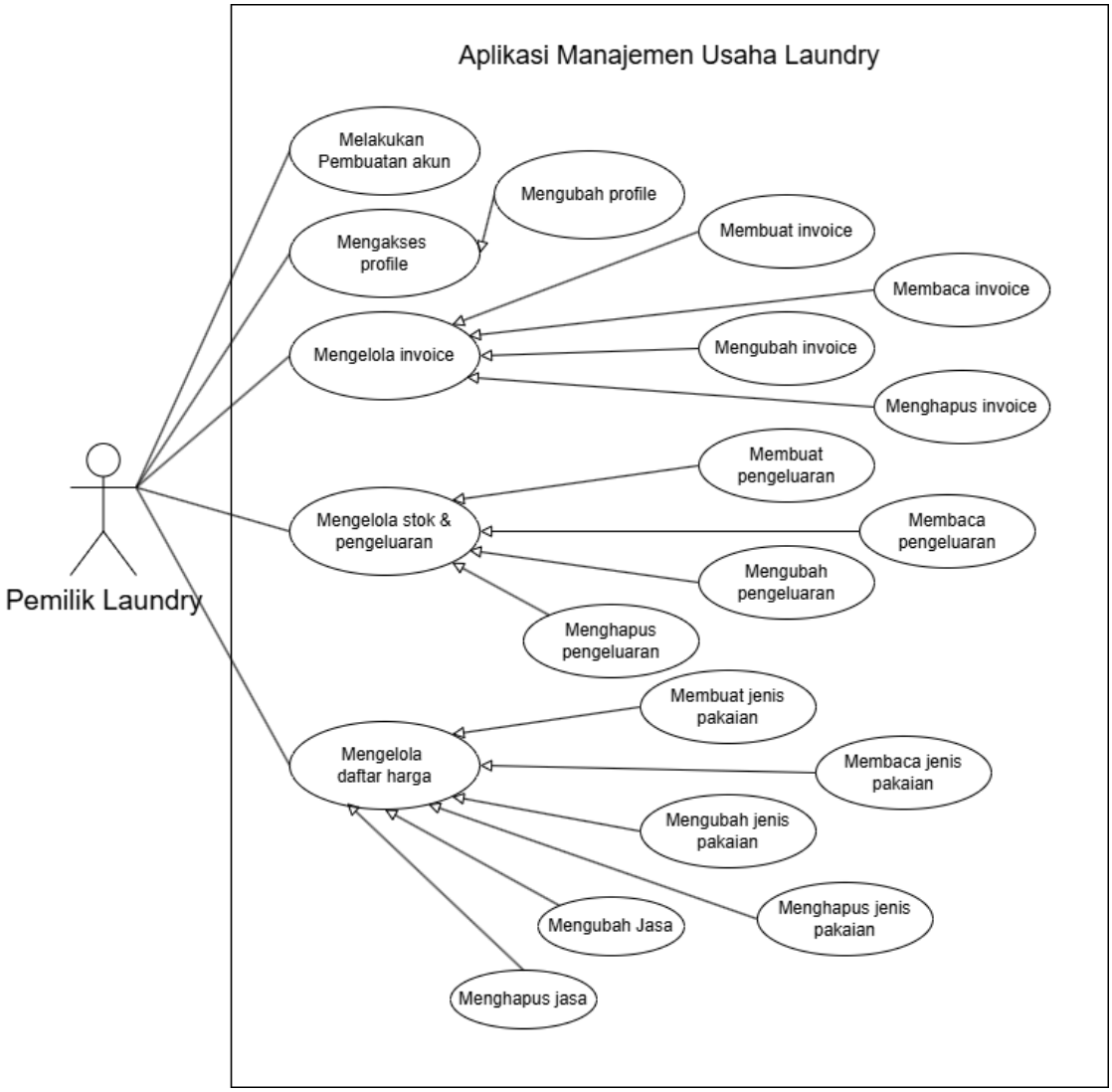
3.1.3.1 Perancangan proses bisnis

Sebelum membuat usecase diagram dan activity diagram, diperlukan pembuatan rancangan *sitemap*. *Sitemap* merupakan kerangka daftar halaman pada suatu aplikasi, sitemap digunakan untuk memudahkan penulis dalam membuat *activity* diagram dan proses desain tampilan aplikasi. Untuk rancangan *sitemap* aplikasi dapat dilihat pada Gambar 3.3.



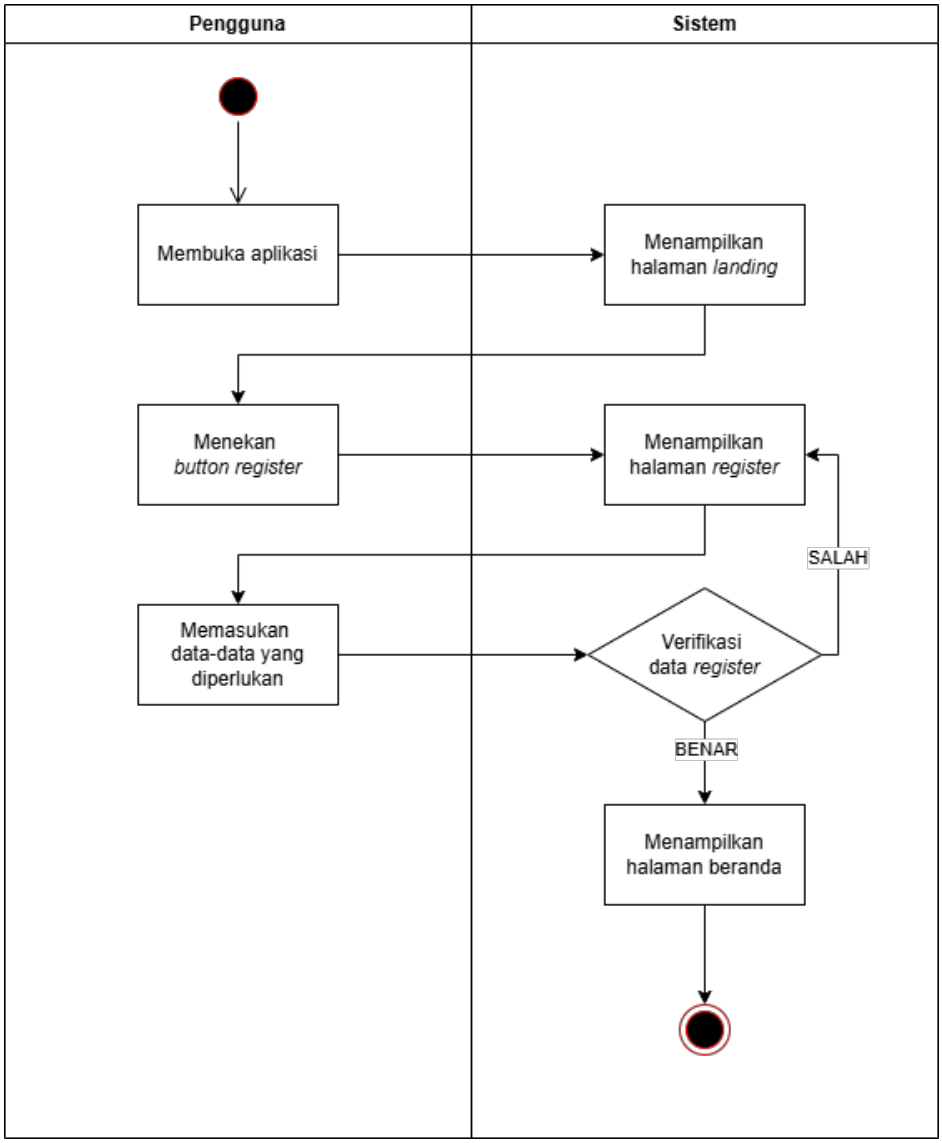
Gambar 3.3 *Sitemap* aplikasi

Selanjutnya dilakukan pembuatan *usecase & activity diagram* yang merupakan bagian dari *unified modelling language* (UML). UML sendiri adalah sebuah bahasa pemodelan visual yang terstruktur dan sistematis untuk perancangan desain sistem (Pranoto et al., 2024). *Usecase diagram* digunakan untuk menggambarkan interaksi pengguna dengan sistem, bagian-bagian dari *usecase* antara lain adalah aktor, aktivitas dan hubungan. Dalam usecase aplikasi ini yang berperan sebagai aktor adalah pemilik *laundry*. Rancangan usecase diagram dapat dilihat pada Gambar 3.4.



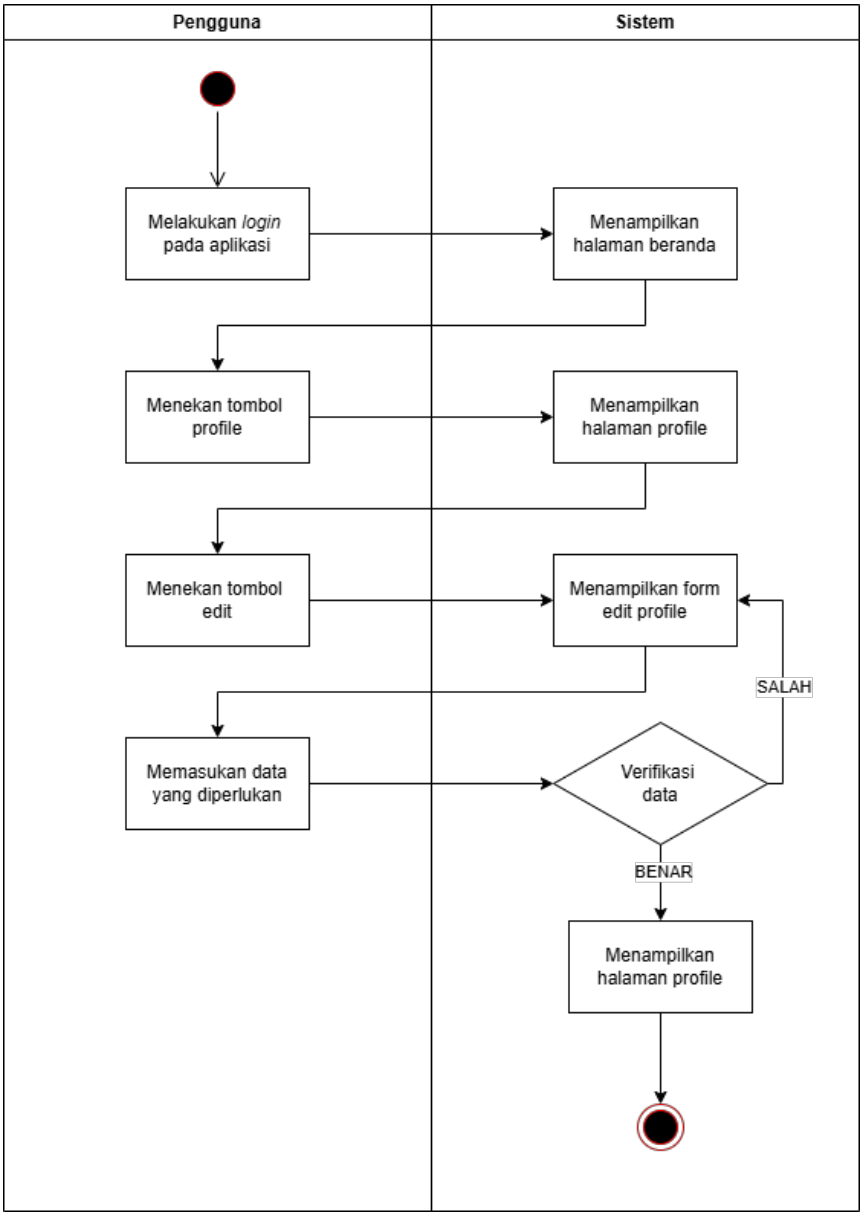
Gambar 3.4 *Usecase diagram* aplikasi

Seperti yang ada pada gambar di atas, di dalam *usecase diagram* memiliki aktivitas-aktivitas yang digambarkan dengan objek oval dengan keterangan nama aktivitas tersebut. Pembuatan *activity* diagram akan mengikuti aktivitas-aktivitas yang ada dalam usecase tersebut. *Activity* diagram digunakan untuk menjelaskan gambaran alur atau proses menyeluruh untuk setiap aktivitas yang ada. Pada aplikasi ini terdapat 16 *activity* diagram yang dijelaskan sebagai berikut:



Gambar 3.5 *Activity diagram register*

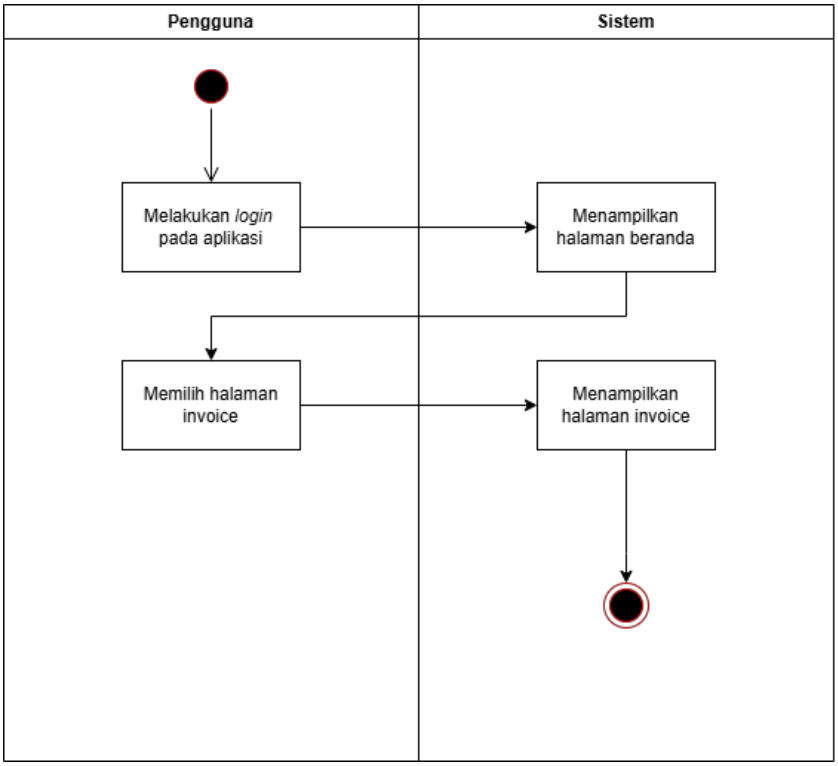
Dalam *activity diagram register* atau daftar akun yang terdapat pada Gambar 3.5, aktivitas dimulai dari sisi pengguna dengan membuka aplikasi. Kemudian sistem akan menampilkan halaman *landing*/*onboarding*, dalam halaman tersebut pengguna harus menekan tombol dengan keterangan *register*. Sistem melakukan navigasi dengan membukan halaman register, pengguna diharuskan mengisi data-data yang diperlukan untuk melanjutkan proses pendaftaran. Selanjutnya sistem akan mengecek data yang dimasukan apakah sudah sesuai, jika data yang dimasukan salah pengguna akan tetap pada halaman tersebut hingga memasukan data dengan benar. Jika data yang dimasukan benar maka sistem akan melanjutkan navigasi menuju halaman beranda.



Gambar 3.6 *Activity diagram edit profile*

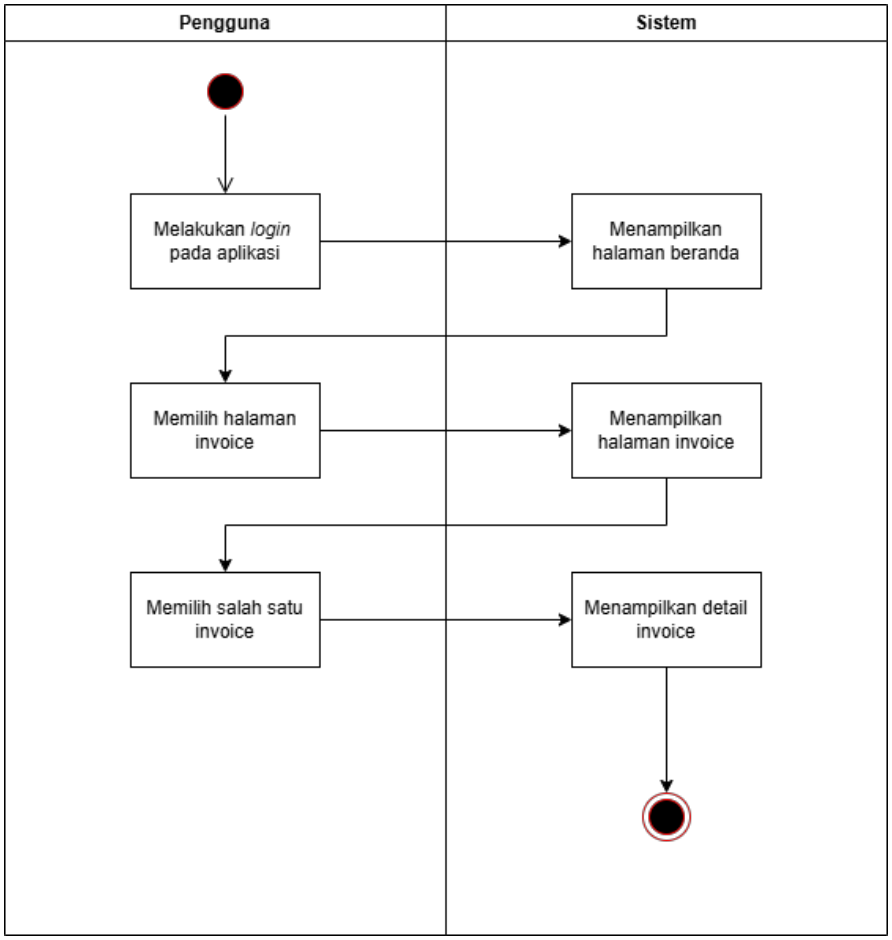
Dalam *activity diagram edit profile* yang ada pada Gambar 3.6, aktivitas dimulai dari sisi pengguna dengan melakukan login pada aplikasi. Sistem kemudian menampilkan halaman beranda, pada halaman ini pengguna harus menekan tombol *profile* sehingga sistem akan menampilkan halaman *profile*. Pada halaman *profile* pengguna harus menekan tombol *edit* dan sistem akan menampilkan halaman *form edit profile*. Pengguna dipersilahkan memasukan atau mengganti data yang diinginkan, Sistem akan mengecek data yang dimasukan, jika data yang dimasukan salah pengguna harus memasukan data

dengan benar, tetapi jika data yang dimasukan sudah benar sistem akan menampilkan halaman profile.



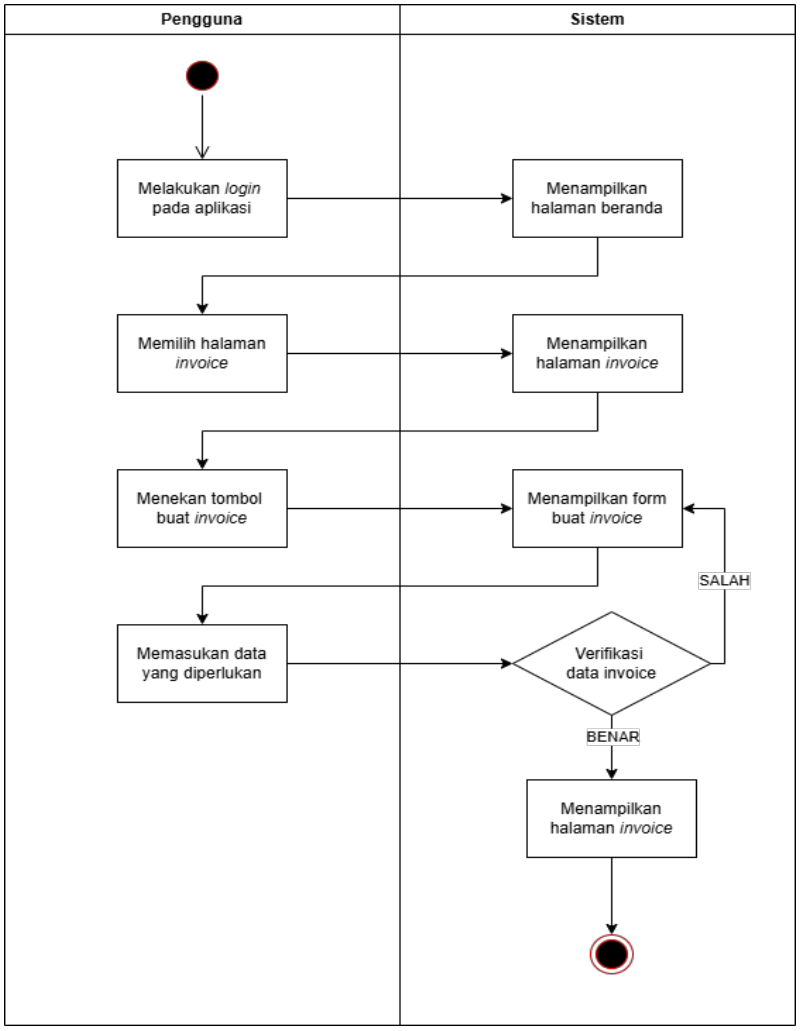
Gambar 3.7 *Activity diagram* buka halaman *invoice*

Berdasarkan pada Gambar 3.7 *Activity diagram* buka halaman *invoice*, *activity diagram* buka halaman *invoice* aktivitas diawali dengan pengguna diharuskan melakukan *login* terlebih dahulu, kemudian sistem akan menampilkan halaman beranda. Pada halaman beranda pengguna hanya perlu menekan tombol yang memiliki keterangan *invoice*. Selanjutnya sistem akan mengalihkan pengguna ke dalam halaman *invoice*.



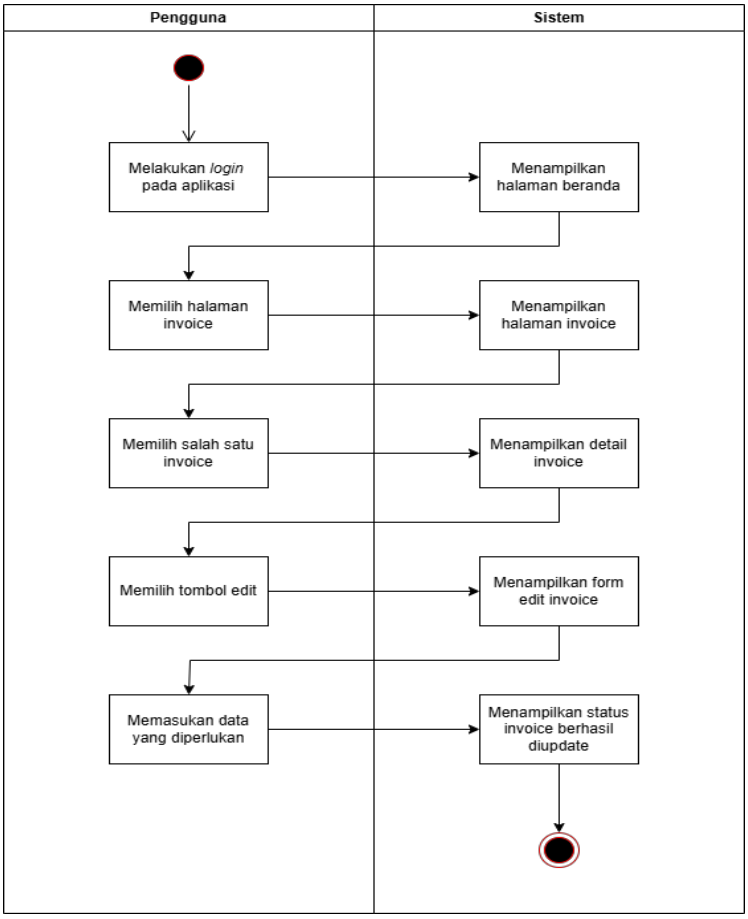
Gambar 3.8 *Activity diagram* detail *invoice*

Seperti yang dapat dilihat pada Gambar 3.8, pada *activity diagram* detail *invoice* diawali dengan pengguna melakukan *login* pada aplikasi terlebih dahulu, dilanjutkan dengan sistem yang menampilkan halaman beranda. Pada halaman beranda pengguna menekan tombol *invoice* yang akan mengarahkan pengguna menuju halaman *invoice.* Pada halaman *invoice* nantinya terdapat daftar *invoice*/nota dari usaha *laundry* pengguna, untuk aktivitas ini pengguna menekan salah satu *invoice* pada daftar yang disajikan. Kemudian sistem akan menampilkan detail dari *invoice* tersebut pada halaman detail *invoice*.



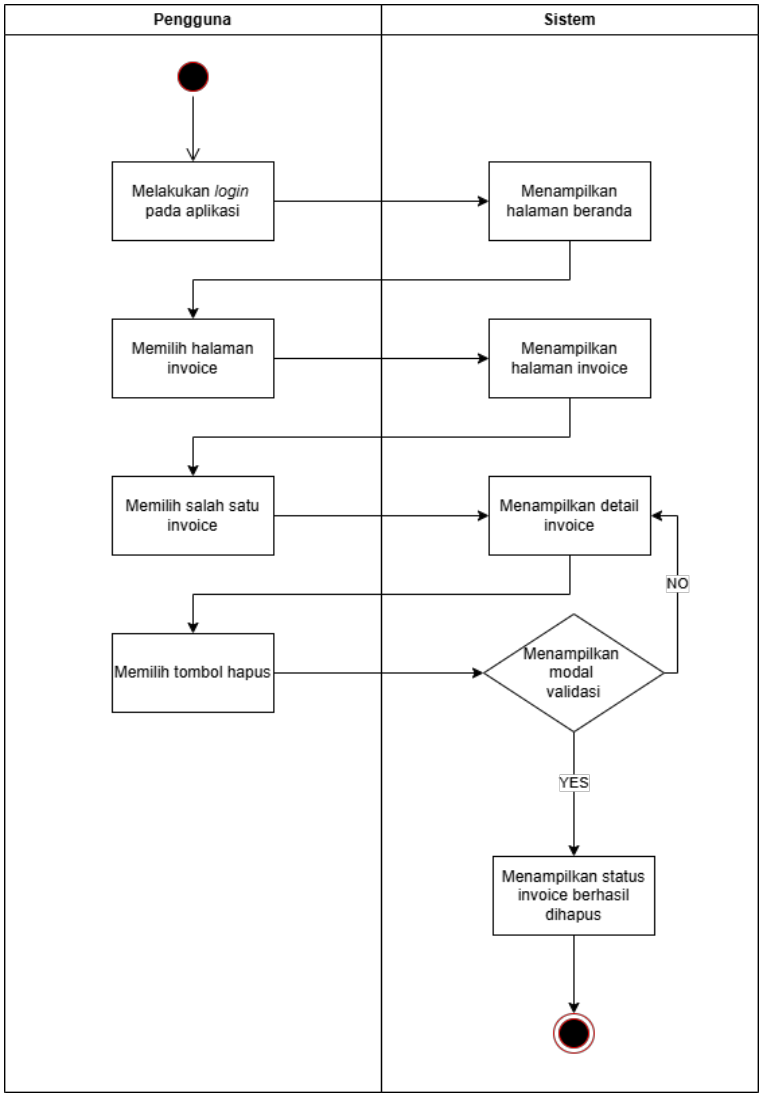
Gambar 3.9 *Activity diagram* buat *invoice*

Seperti yang dapat dilihat pada Gambar 3.9, pada *activity diagram* buat *invoice* diawali dengan pengguna melakukan login pada aplikasi terlebih dahulu, dilanjutkan dengan sistem yang akan menampilkan halaman beranda. Pada halaman beranda pengguna menekan tombol *invoice* yang akan mengarahkan pengguna menuju halaman *invoice*. Pada halaman *invoice*, pengguna diharuskan menekan tombol buat *invoice* selanjutnya sistem akan menampilkan halaman *form* buat *invoice*. Pengguna diminta memasukan data yang diperlukan, kemudian sistem akan mengecek data yang dimasukan sudah benar atau salah. Jika data yang dimasukan salah pengguna diminta memasukan data yang benar pada halaman tersebut dan jika data yang dimasukan sudah benar sistem akan membawa pengguna kembali pada halaman *invoice*.



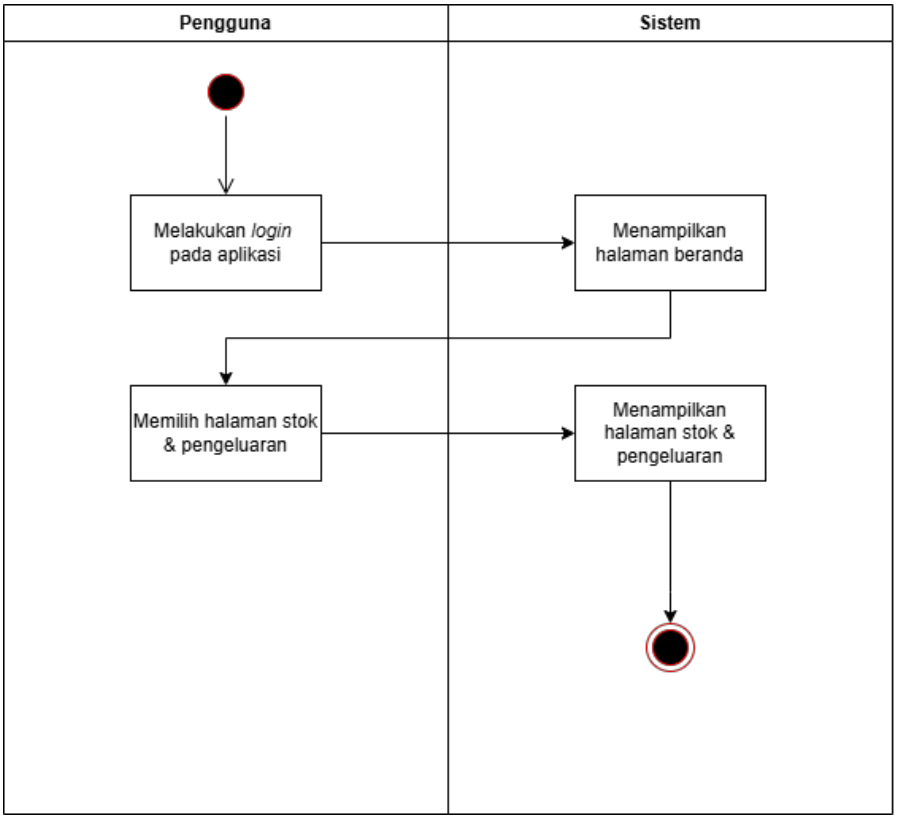
Gambar 3.10 *Activity diagram edit invoice*

Seperti yang dapat dilihat pada Gambar 3.10, pada *activity diagram edit invoice* diawali dengan pengguna melakukan *login* pada aplikasi terlebih dahulu, dilanjutkan dengan sistem yang akan menampilkan halaman beranda. Pada halaman beranda pengguna menekan tombol *invoice* yang akan mengarahkan pengguna menuju halaman *invoice.* Pada halaman *invoice* nantinya terdapat daftar *invoice*/nota dari usaha *laundry* pengguna, untuk aktivitas ini pengguna menekan salah satu *invoice* pada daftar yang disajikan. Kemudian sistem akan menampilkan detail dari *invoice* tersebut pada halaman detail *invoice*. Pada halaman tersebut pengguna dapat menekan tombol edit sehingga sistem menampilkan halaman form edit invoice, kemudian pengguna dapat memasukan data yang ingin diganti dan menekan tombol simpan. Sistem akan menampilkan *alert* status invoice berhasil diperbaharui pada halaman invoice.



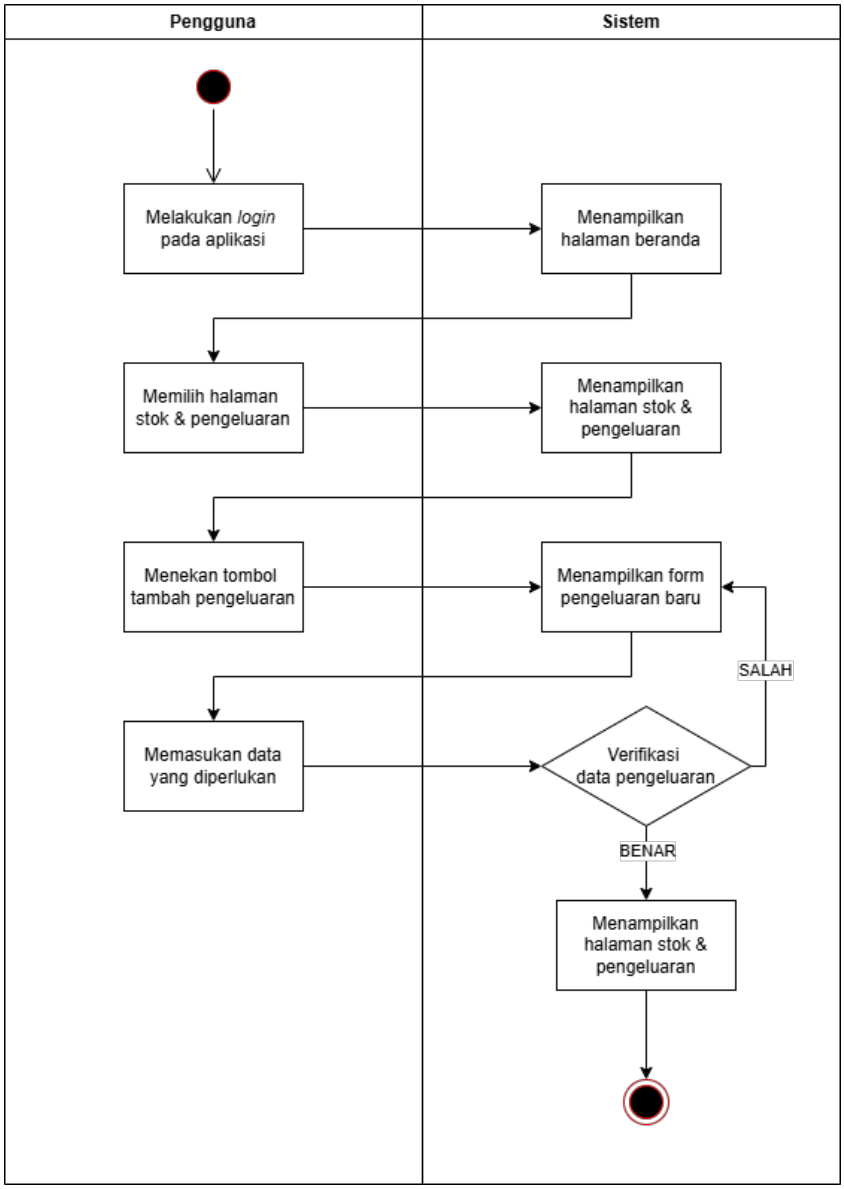
Gambar 3.11 *Activity diagram* hapus *invoice*

Seperti yang dapat dilihat pada Gambar 3.11, pada *activity diagram* hapus *invoice* diawali dengan pengguna melakukan *login* pada aplikasi terlebih dahulu, dilanjutkan dengan sistem yang akan menampilkan halaman beranda. Pada halaman beranda pengguna menekan tombol *invoice* yang akan mengarahkan pengguna menuju halaman *invoice.* Pada halaman *invoice* nantinya terdapat daftar *invoice*/nota dari usaha *laundry* pengguna, untuk aktivitas ini pengguna menekan salah satu *invoice* pada daftar yang disajikan. Kemudian sistem akan menampilkan detail dari *invoice* tersebut pada halaman detail *invoice*. Pada halaman tersebut pengguna dapat menekan tombol hapus, sistem kemudian menampilkan modal validasi. Jika pengguna menekan “Tidak/Batal” maka sistem akan tetap pada halaman detail invoice tetapi jika pengguna menekan “Ya” sistem akan melanjutkan menampilkan alert status bahwa *invoice* berhasil dihapus pada halaman *invoice*.



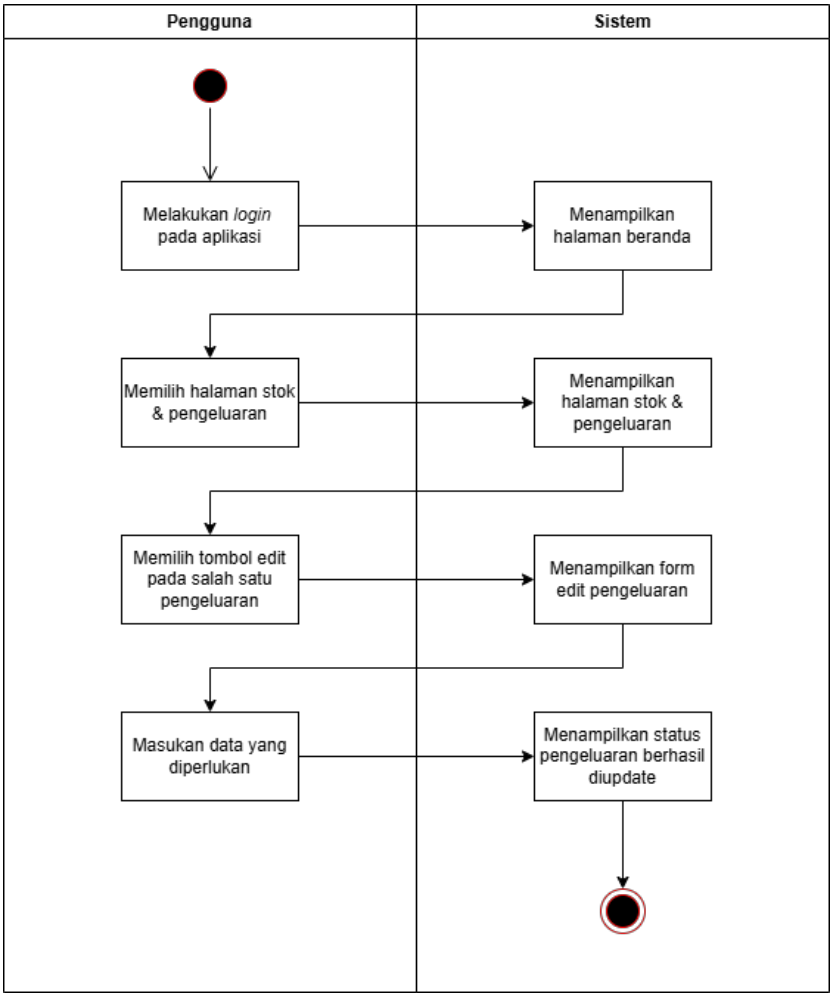
Gambar 3.12 *Activity diagram* buka halaman stok & pengeluaran

Dari activity diagram buka halaman stok & pengeluaran yang ada Gambar 3.12, aktivitas diawali dengan pengguna sudah melakukan login pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan stok & pengeluaran yang akan mengarahkan pengguna ke halaman stok & pengeluaran.



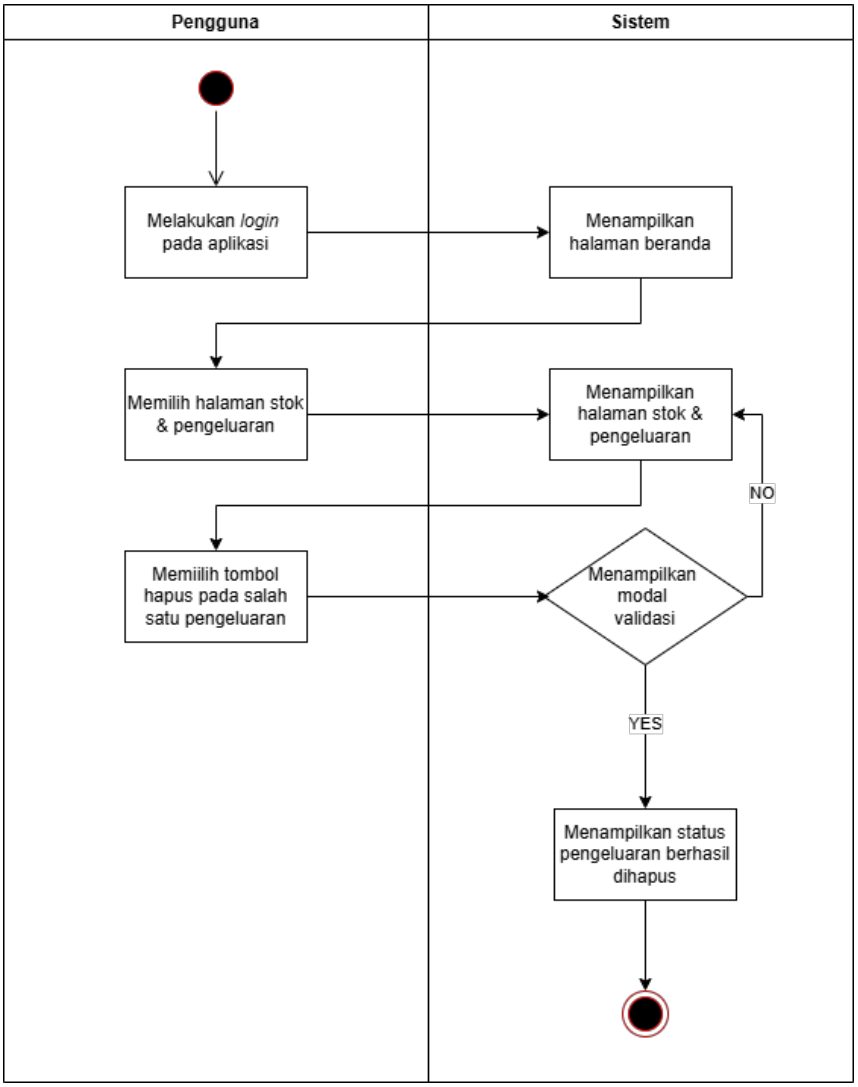
Gambar 3.13 *Activity diagram* buat pengeluaran

Dari *activity diagram* buat pengeluaran yang ada pada Gambar 3.13, aktivitas diawali dengan pengguna sudah melakukan login pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan stok & pengeluaran yang akan mengarahkan pengguna ke halaman stok & pengeluaran. Pada halaman tersebut pengguna dapat membuat pengeluaran baru dengan menekan tombol tambah pengeluaran yang akan memerintah sistem untuk menampilkan halaman *form* buat pengeluaran baru. Pengguna diminta memasukan data yang diperlukan kemudian sistem akan melakukan pengecekan data, jika data yang dimasukan sudah benar sistem akan menampilkan halaman stok & pengeluaran.



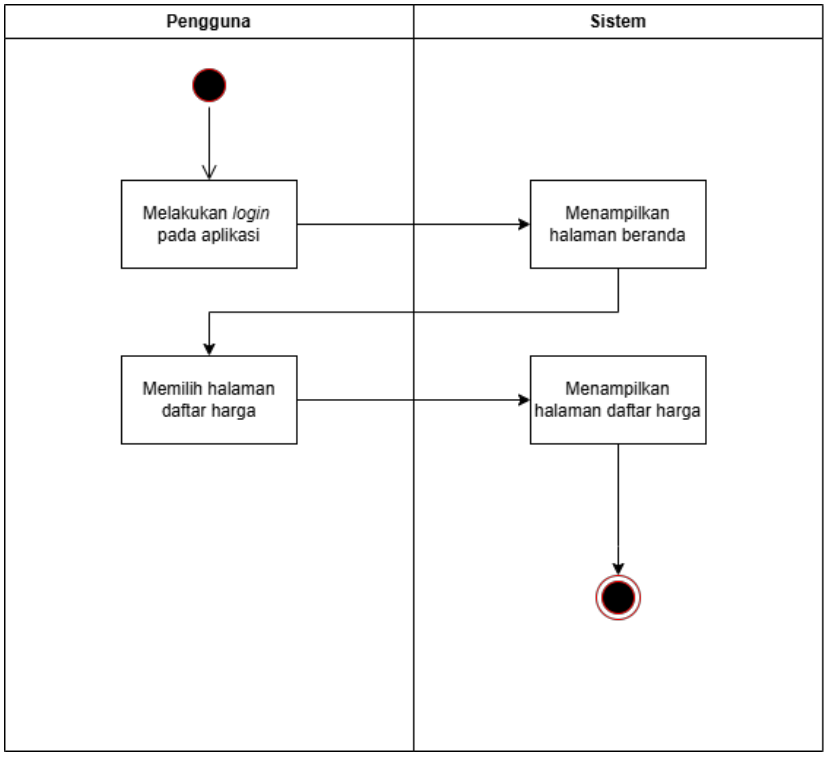
Gambar 3.14 *Activity diagram edit* pengeluaran

Dari *activity diagram edit* pengeluaran yang ada pada Gambar 3.14, aktivitas diawali dengan pengguna sudah melakukan login pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan stok & pengeluaran yang akan mengarahkan pengguna ke halaman stok & pengeluaran. Pada halaman tersebut pengguna dapat memilih untuk mengganti data pada pengeluaran yang sudah ada dengan menekan tombol *edit* yang akan memerintah sistem untuk menampilkan halaman *form edit* pengeluaran. Pengguna diminta memasukan data yang ingin diganti kemudian sistem akan melakukan pengecekan data, jika data yang dimasukan sudah benar sistem akan menampilkan *alert* status bahwa pengeluaran berhasil diperbaharui pada halaman stok & pengeluaran.



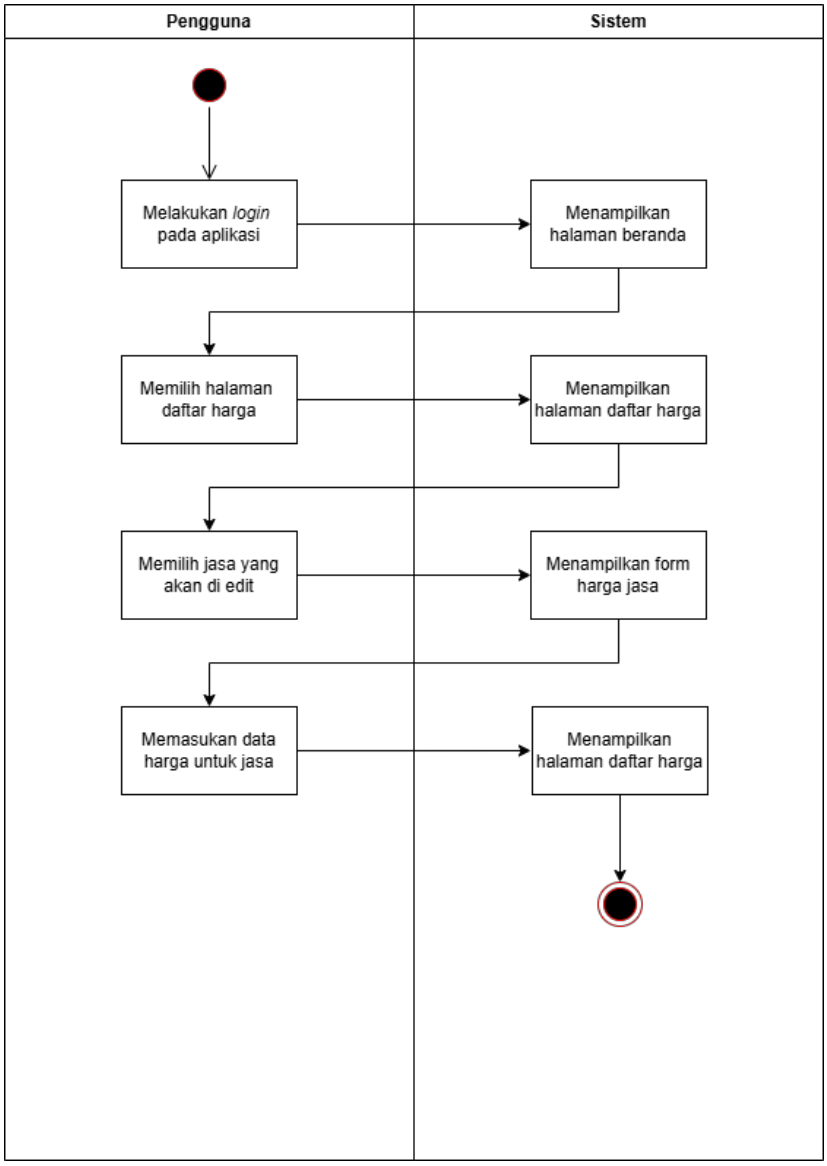
Gambar 3.15 *Activity diagram* hapus pengeluaran

Dari *activity diagram edit* pengeluaran yang ada pada Gambar 3.15, aktivitas diawali dengan pengguna sudah melakukan login pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan stok & pengeluaran yang akan mengarahkan pengguna ke halaman stok & pengeluaran. Pada halaman tersebut pengguna dapat memilih untuk menghapus pengeluaran yang sudah ada dengan menekan tombol hapus yang akan memerintah sistem untuk menampilkan modal validasi. Pengguna diminta menekan "Ya" dan sistem akan menampilkan *alert* status bahwa pengeluaran berhasil dihapus pada halaman stok & pengeluaran. Jika pengguna menekan "Tidak/Batal" maka sistem akan tidak akan menghapus pengeluaran tersebut dan tetap menampilkan halaman stok & pengeluaran.



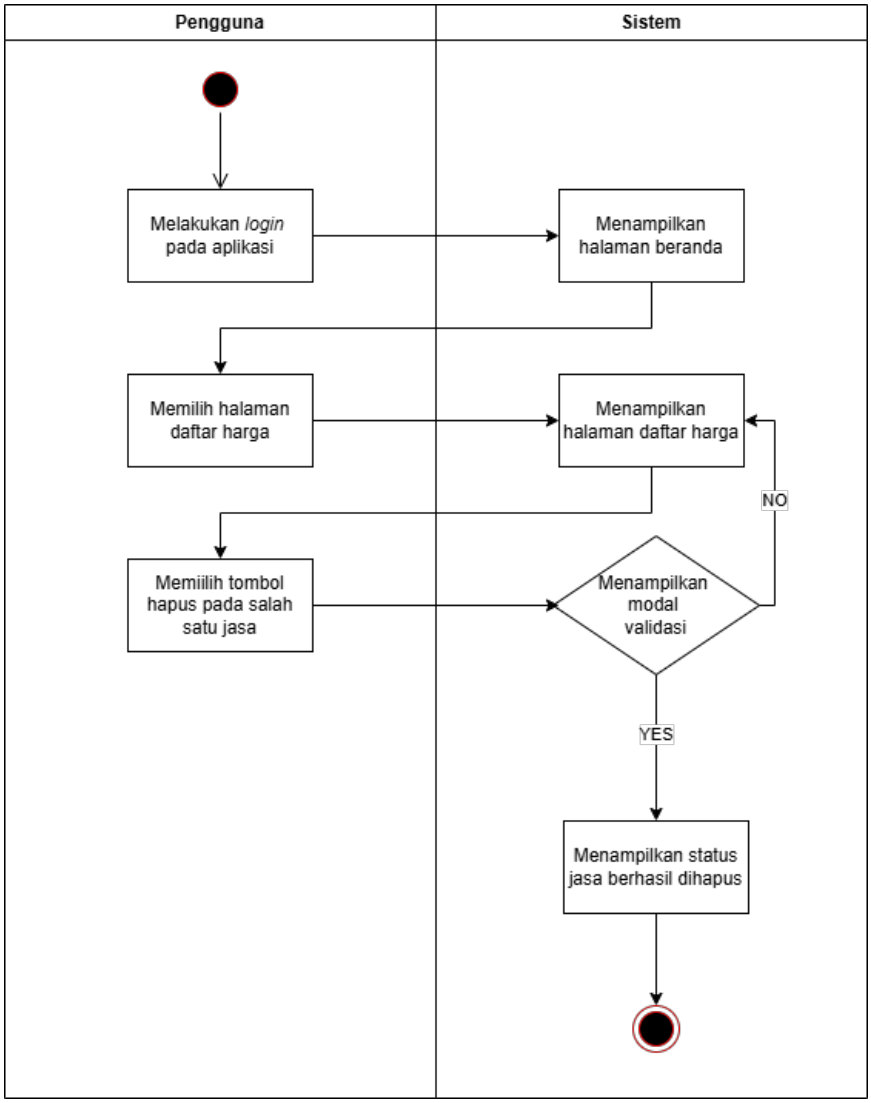
Gambar 3.16 *Activity diagram* buka halaman daftar harga

Berdasarkan pada Gambar 3.16, *activity diagram* buka halaman daftar harga aktivitas diawali dengan pengguna diharuskan melakukan *login* terlebih dahulu, kemudian sistem akan menampilkan halaman beranda. Pada halaman beranda pengguna hanya perlu menekan tombol yang memiliki keterangan daftar harga. Selanjutnya sistem akan mengalihkan pengguna ke dalam halaman daftar harga.



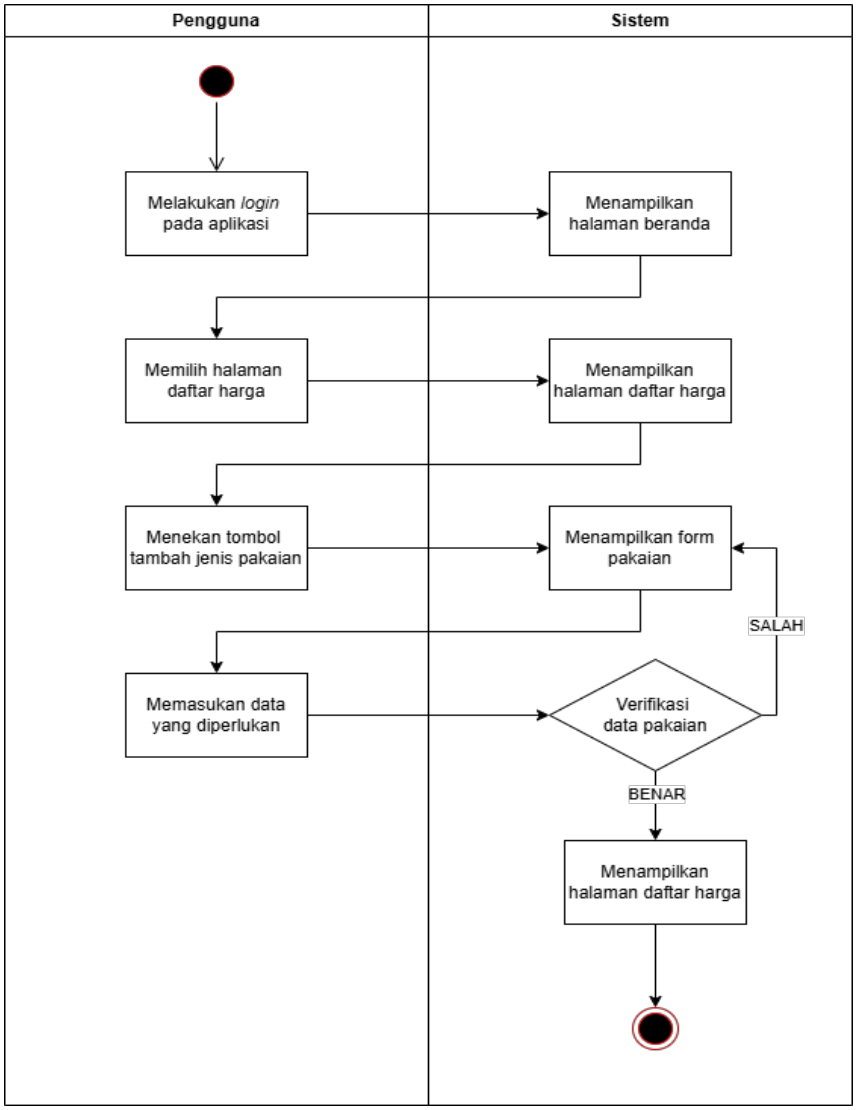
Gambar 3.17 *Activity diagram edit* jasa

Berdasarkan pada Gambar 3.17, *activity diagram edit* jasa, aktivitas diawali dengan pengguna diharuskan melakukan *login* terlebih dahulu, kemudian sistem akan menampilkan halaman beranda. Pada halaman beranda pengguna hanya perlu menekan tombol yang memiliki keterangan daftar harga. Selanjutnya sistem akan mengalihkan pengguna ke dalam halaman daftar harga. Pada halaman ini pengguna dapat mengganti data yang ada pada jasa dengan menekan jasa yang akan diubah, kemudian sistem akan menampilkan form untuk pengguna mengganti data yang ada. Setelah itu sistem akan menampilkan halaman daftar harga.



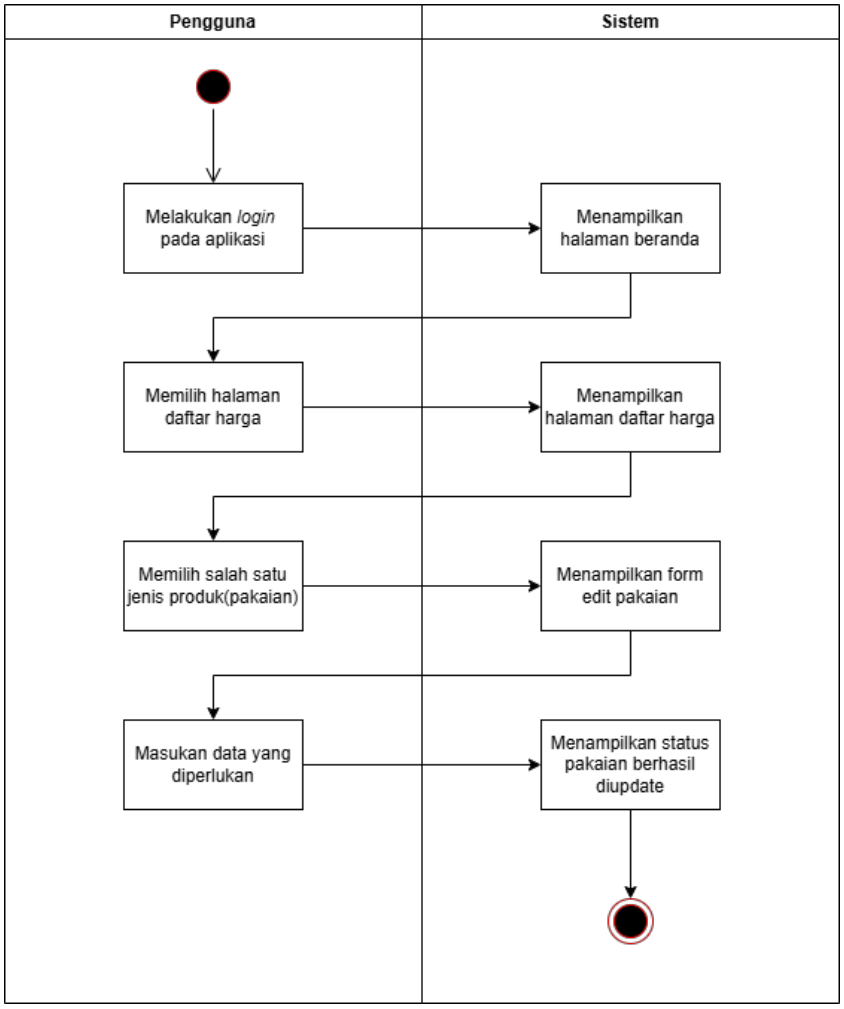
Gambar 3.18 *Activity diagram* hapus jasa

Berdasarkan pada Gambar 3.18, *activity diagram* hapus jasa, aktivitas diawali dengan pengguna diharuskan melakukan *login* terlebih dahulu, kemudian sistem akan menampilkan halaman beranda. Pada halaman beranda pengguna hanya perlu menekan tombol yang memiliki keterangan daftar harga. Selanjutnya sistem akan mengalihkan pengguna ke dalam halaman daftar harga. Pada halaman ini pengguna dapat menghapus jasa dengan menekan jasa yang akan dihapus, kemudia pengguna menakan tombol hapus. Dan sistem akan menampilkan modal validasi untuk pengguna, jika pengguna menekan "Tidak/Batal" sistem akan menutup modal validasi tersebut tetapi jika pengguna menekan "Ya" sistem akan menampilkan *alert* status bahwa jasa berhasil dihapus. Setelah itu sistem akan menampilkan halaman daftar harga.



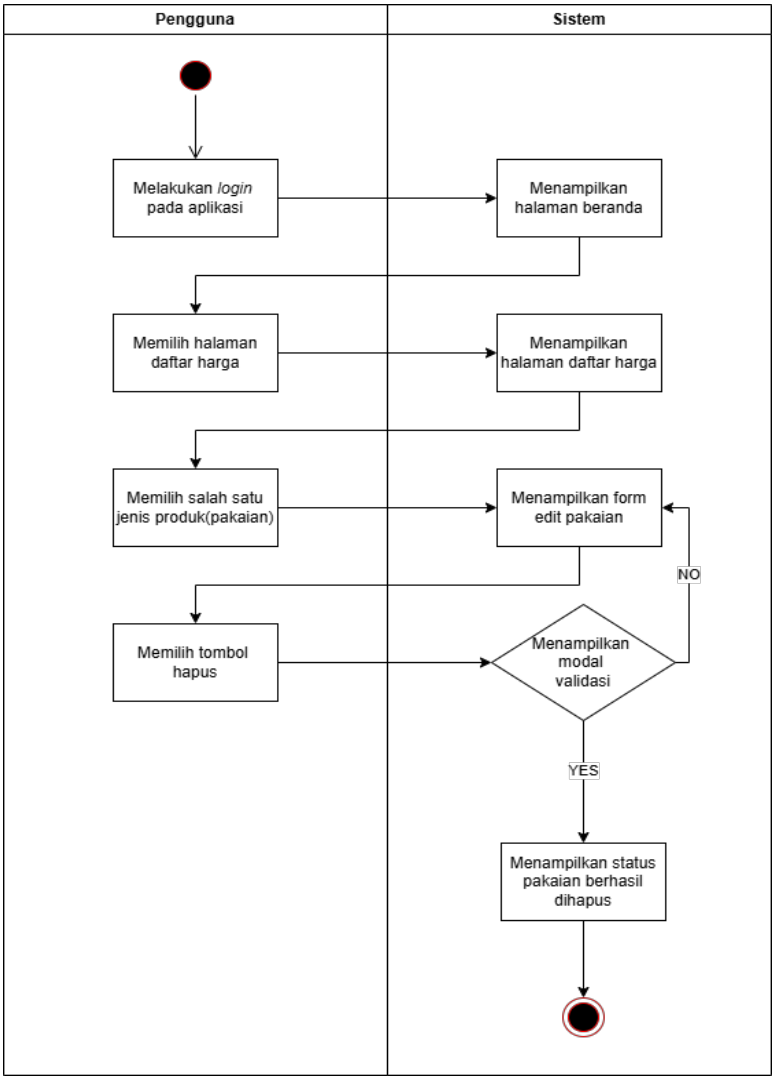
Gambar 3.19 *Activity diagram* buat jenis pakain

Dalam *activity diagram* buat jenis pakaian yang terdapat pada Gambar 3.19, aktivitas ini dimulai dengan pengguna melakukan *login* pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan harga untuk berpindah ke halaman daftar harga, dalam halaman tersebut pengguna dapat menambahkan jenis pakaian baru dengan menekan tombol tambah produk. Selanjutnya sistem akan menampilkan halaman *form* buat pakaian dimana pengguna diminta untuk mengisi data yang diperlukan. Setelah mengisi data yang diperlukan sistem akan mengecek data yang dimasukan sudah sesuai atau tidak, jika data tidak sesuai pengguna diharuskan mengisi data dengan benar dan jika data yang dimasukan sudah benar maka sistem akan menampilkan halaman daftar harga.



Gambar 3.20 *Activity diagram edit* jenis pakaian

Dalam *activity diagram edit* jenis pakaian yang terdapat pada Gambar 3.20, aktivitas ini dimulai dengan pengguna melakukan *login* pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan harga untuk berpindah ke halaman daftar harga, dalam halaman tersebut akan menampilkan daftar jenis pakaian yang sudah dibuat. Untuk mengganti data yang ada pada jenis pakaian tersebut pengguna menekan salah satu jenis pakaian yang ingin diubah, dan sistem akan menampilkan *form edit* pakaian. Pengguna dapat mengganti data yang ada, setelah itu sistem akan menampilkan bahwa jenis pakaian berhasil diperbaharui.



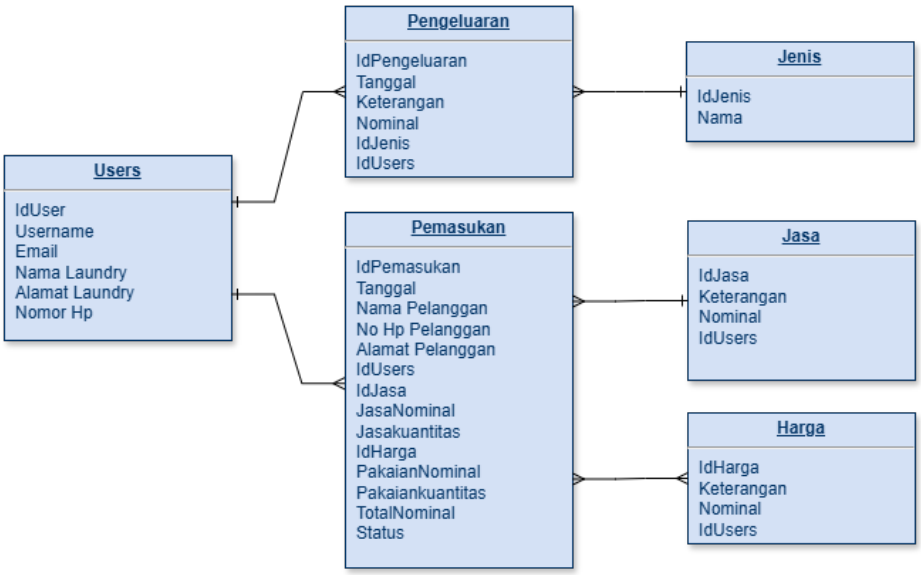
Gambar 3.21 *Activity diagram* hapus jenis pakaian

Dalam *activity diagram* hapus jenis pakaian yang terdapat pada Gambar 3.21, aktivitas ini dimulai dengan pengguna melakukan *login* pada aplikasi, sistem kemudian menampilkan halaman beranda. Pengguna menekan tombol dengan keterangan harga untuk berpindah ke halaman daftar harga, dalam halaman tersebut akan menampilkan daftar jenis pakaian yang sudah dibuat. Untuk menghapus salah satu jenis pakaian yang ada pengguna menekan salah satu jenis pakaian yang ingin dihapus, dan sistem akan menampilkan *form edit* pakaian. Pengguna dapat menekan tombol hapus yang ada pada halaman tersebut, setelah itu sistem akan menampilkan modal validasi. Jika pengguna menekan "Ya" sistem akan menampilkan alert status bahwa jenis pakaian berhasil dihapus, dan jika pengguna menekan "Tidak/Batal" sistem akan menghilangkan tampilan modal validasi.

3.1.3.2 Perancangan basis data

Pada bagian ini dilakukan pembuatan desain basis data sebagai landasan untuk pengelolaan informasi dalam sistem. Salah satu langkah penting dalam tahap ini adalah penyusunan ERD (*Entity Relationship Diagram*) yaitu representasi visual dari struktur basis data yang menggambarkan hubungan antar entitas atau objek (Hadi & Dwi Gustina, 2024). ERD ini berperan penting dalam mempermudah proses pengembangan *backend* aplikasi yang berhubungan dengan basis data. Rancangan ERD untuk aplikasi ini dapat dilihat seperti pada

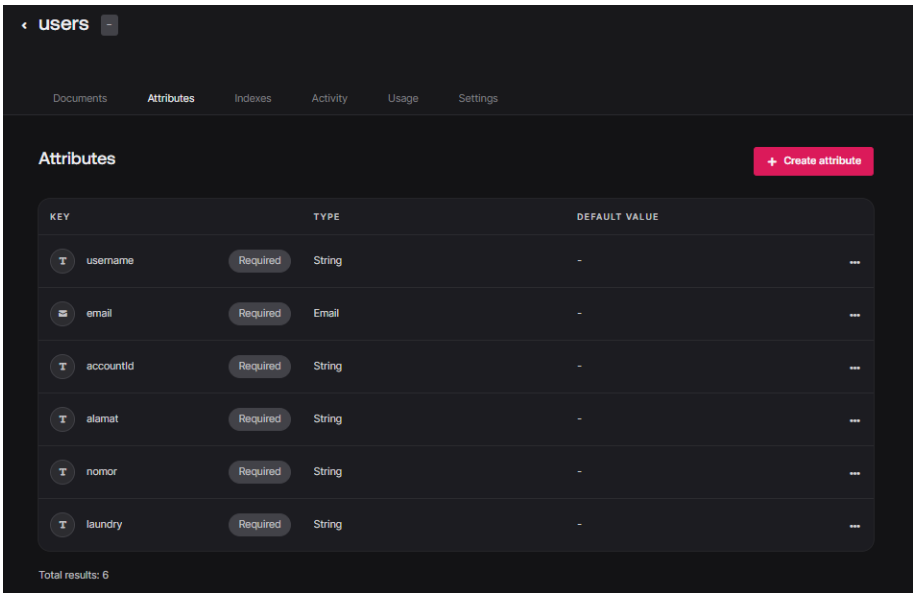
Gambar 3.22, dalam gambar tersebut memiliki enam *collections* utama yaitu: *users*, pengeluaran, pemasukan, jenis, jasa, dan harga. Masing-masing *collections* memiliki *attribute* yang menyimpan informasi spesifik terkait fungsinya. Implentasi bagian *backend* dilakukan menggunakan Appwrite, sebuah layanan penyedia *backend service* serupa Google Firebase yang mendukung pengelolaan *database*,autentikasi dan penyimpanan data.



Gambar 3.22 Rancangan ERD

Dalam *collections users*, informasi yang disimpan merupakan informasi mengenai data diri dan usaha pengguna. Seperti yang bisa dilihat pada Gambar 3.23, dalam gambar tersebut menampilkan *users collections* beserta *attribute*-nya dalam appwrite *databases*. *Collections* ini terdiri dari 6 *attribute* yaitu:

1. *Username* dengan tipe *string* memiliki status *required* yang dimana *attribute* ini tidak boleh kosong. *Attribute* ini digunakan untuk menyimpan nama pemilik usaha laundry.
2. Email dengan tipe email memiliki status *required, attribute* ini menyimpan email pengguna yang digunakan untuk melakukan login.
3. *AccountId* dengan tipe *string* memiliki status *required*, merupakan *attribute* yang memiliki hubungan dengan fitur *auth* pada Appwrite. *Attribute* ini menyimpan *ID* dari fitur *auth* yang ada.
4. Alamat dengan tipe *string* memiliki status *required*, *attribute* ini menyimpan informasi mengenai alamat tempat usaha pengguna.
5. Nomor dengan tipe *string* memiliki status *required, attribute* ini digunakan untuk menyimpan informasi mengenai kontak nomor telepon usaha pengguna.
6. *Laundry* dengan tipe *string* memiliki status *required*, *attribute* ini digunakan untuk menyimpan nama usaha *laundry* pengguna.

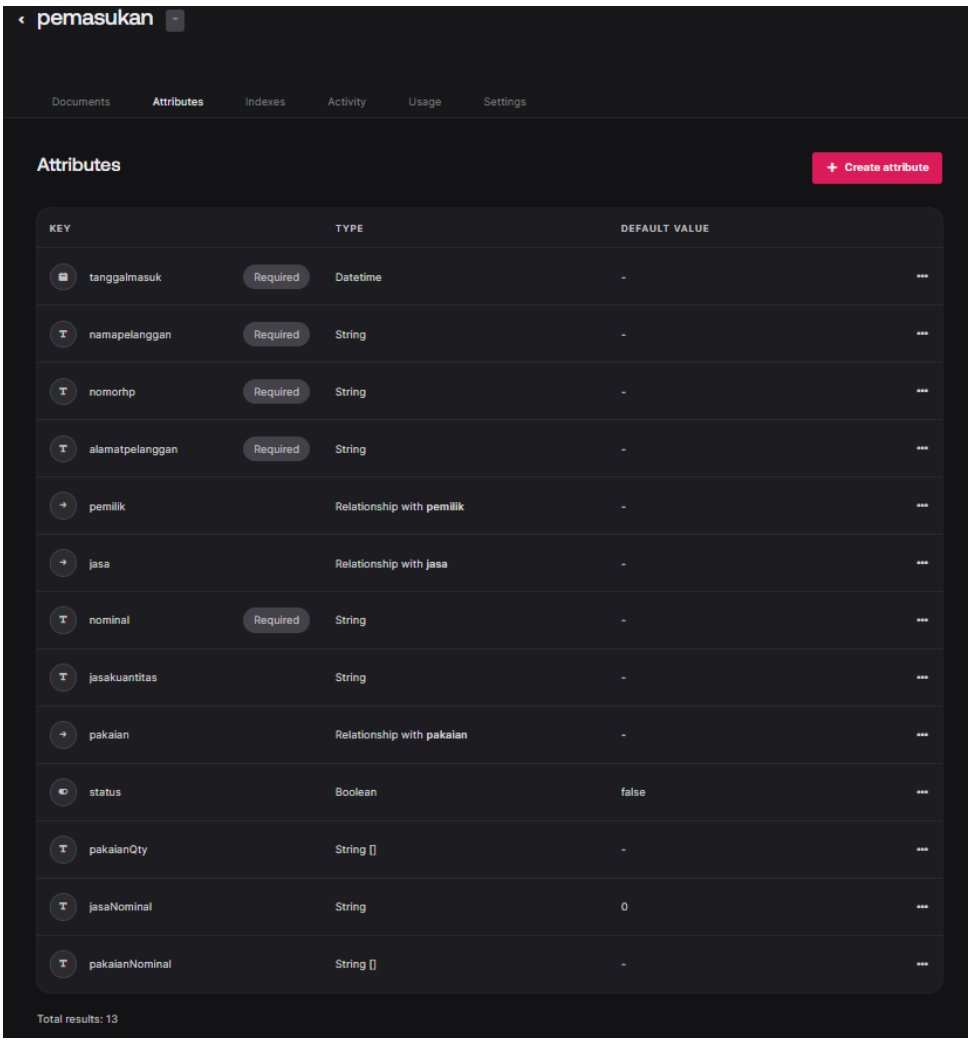


Gambar 3.23 *Users collections*

Untuk pemasukan *collections*, merupakan *collections* dengan *attribute* terbanyak dikarenakan *collections* ini memiliki hubungan dengan tiga *collections* lainnya. Informasi yang disimpan pada *collections* ini merupakan data mengenai pemasukan pengguna yang biasanya ditulis pada *invoice*/nota laundry. Seperti yang ada pada Gambar 3.24 *collections* ini memiliki 13 *attribute* dengan rincian sebagai berikut:

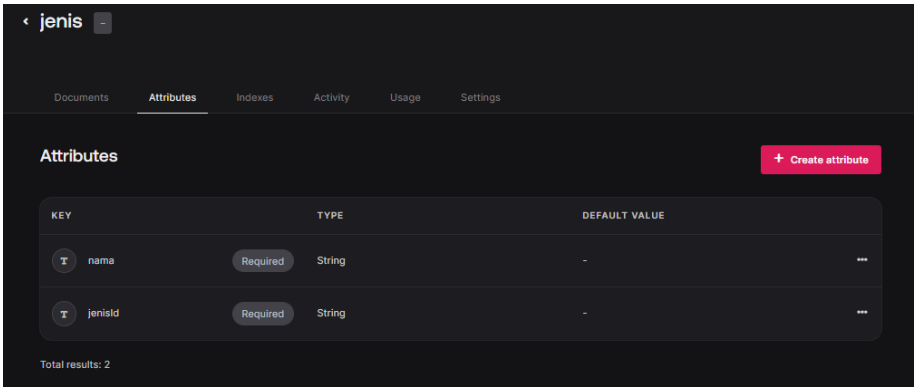
1. Tanggalmasuk merupakan *attribute* dengan status *required* dan tipe *datetime* yang termasuk tipe khusus untuk data yang berupa tanggal. Informasi yang disimpan pada *attribute* ini berupa tanggal dimana barang atau pakaian masuk.
2. Namapelanggan merupakan *attribute* dengan status *required* dan tipe *string*, informasi yang disimpan berupa nama pelanggan yang melakukan transaksi.
3. Nomorhp merupakan *attribute* dengan status *required* dan tipe *string*, informasing yang disimpan berupa nomor kontak telepon dari pelanggan yang melakukan transaksi.
4. Alamatpelanggan merupakan *attribute* dengan status *required* dan tipe *string*, informasi yang disimpan berupa alamat dari pelanggan yang melakukan transaksi.
5. Pemilik merupakan *attribute relationship* dengan *user collections* memiliki tipe hubungan *many-to-one* yang berarti pemasukan memiliki satu pemilik tetapi pemilik dapat memiliki banyak pemasukan. Informasi yang disimpan berupa *ID* pada *user collections* yang merujuk pada pemilik pemasukan tersebut.
6. Jasa merupakan *attribute relationship* dengan jasa *collections* memiliki tipe hubungan *many-to-one* yang berarti pemasukan memiliki satu jasa tetapi jasa dapat memiliki banyak pemasukan. Informasi yang disimpan berupa *ID* pada jasa *collections* yang merujuk pada jenis jasa yang digunakan pada transaksi tersebut.
7. Nominal merupakan *attribute* dengan status *required* dan tipe *string*, informasi yang disimpan berupa total nominal dari transaksi yang dilakukan oleh pelanggan.
8. Jasakuantitas merupakan *attribute* dengan tipe *string*, informasi yang disimpan berupa jumlah kuantitas dari barang atau pakaian pada saat transaksi.
9. Pakaian merupakan *attribute relationship* dengan harga *collections* memiliki tipe hubungan *many-to-many* yang berarti pemasukan dapat memiliki banyak pakaian dan pakaian dapat memiliki banyak pemasukan. Informasi yang disimpan berupa *ID* pada harga *collections* yang merujuk pada pakaian yang digunakan pada transaksi tersebut.
10. Status memiliki tipe *boolean* dengan *default value false*, *attribute* ini digunakan sebagai status bahwa nota sudah lunas atau belum dibayar. Jika data yang disimpan *false* berarti transaksi tersebut belum dibayar dan data true untuk transaksi yang sudah lunas.

11. Pakaianqty merupakan *attribute* dengan tipe *array string*, karena *attribute* ini digunakan sebagai data kuantitas dari *attribute relationship* pakaian yang memiliki hubungan *many-to-many*.
12. Jasanominal merupakan *attribute* dengan tipe *string*, attribute ini digunakan untuk menyimpan nominal jasa yang digunakan pada saat nota dibuat. Karena harga jasa dapat diganti kapan saja sehingga dibutuhkan *attribute* ini.
13. Pakaiannominal merupakan *attribute* dengan tipe *array string*, *attribute* ini digunakan untuk menyimpan nominal pakaian yang digunakan pada saat nota dibuat. Karena harga pakaian dapat diganti kapan saja sehingga dibutuhkan *attribute* ini.



Gambar 3.24 Pemasukan *collections*

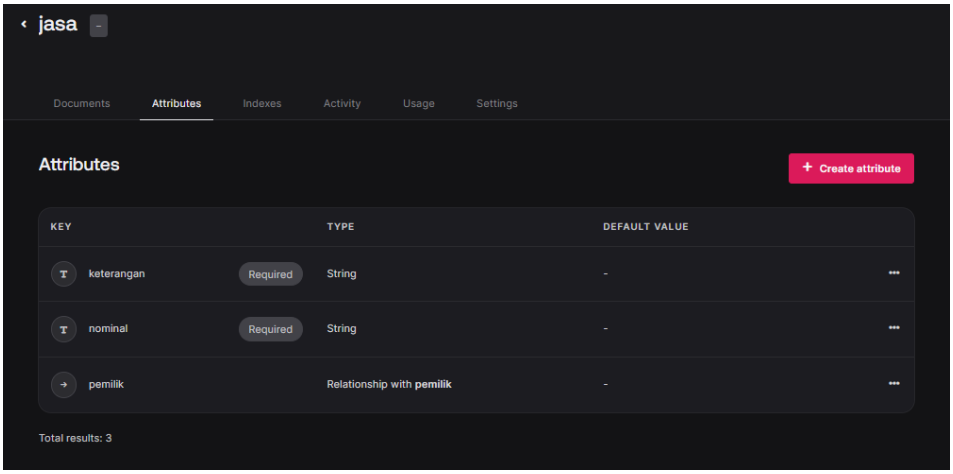
Jenis *collections* digunakan untuk menyimpan dua data yang membedakan jenis pengeluaran yaitu *stock* dan *expenses*. *Attribute* yang ada pada *collections* tersebut dapat dilihat seperti pada Gambar 3.25. Pada *attribute* nama digunakan untuk menyimpan variable "*Stock*" dan "*Expenses*" dan jenisId digunakan untuk menyimpan nomor ID untuk setiap variable. jenisId 1 digunakan untuk variable "*Stock*" dan 2 untuk "*Expenses*".



Gambar 3.25 Jenis *collections*

Pada jasa *collections* yang dapat dilihat pada Gambar 3.26 memiliki tiga *attribute*, *collections* ini digunakan untuk menyimpan jasa yang ada pada usaha pengguna. Nantinya jasa tersebut akan terbuat secara otomatis ketika pengguna melakukan pendaftaran pada aplikasi. Ketiga *attribute* dijelaskan sebagai berikut:

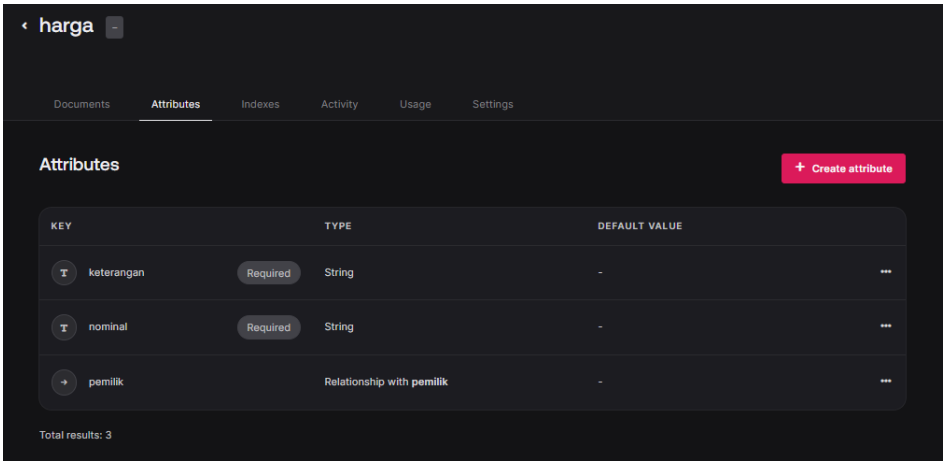
1. Keterangan yang menyimpan data dalam bentuk *string* dan merupakan *attribute required*, yang harus diisi dengan nama dari jasa yang ada.
2. Nominal yang menyimpan data dalam bentuk *string,* penggunaan tipe data *string* untuk data yang memuat angka digunakan untuk memudahkan dalam implementasi. Data yang disimpan dalam *attribute* ini merupakan harga dari jasa tersebut.
3. Pemilik yang merupakan *attribute relationship* dengan *user collections*, sehingga saat pengguna menggunakan aplikasi jasa yang disajikan merupakan jasa yang berkolerasi dengan pengguna itu sendiri tidak seluruh jasa dalam basis data.



Gambar 3.26 Jasa *collections*

Collections yang terakhir yaitu harga *collections,* digunakan untuk menyimpan data pakaian untuk cuci satuan pada usaha pengguna. *Collections* ini memiliki tiga *attribute* yang dapat dilihat pada Gambar 3.27. *Attribute* tersebut dijelaskan sebagai berikut:

1. *Attribute* keterangan menyimpan data dengan bentuk *string*, data yang disimpan berupa keterangan mengenai nama pakaian.
2. Attribue nominal yang menyimpan data dalam bentuk *string,* penggunaan tipe data *string* untuk data yang memuat angka digunakan untuk memudahkan dalam implementasi. Data yang disimpan dalam *attribute* ini merupakan harga dari pakaian tersebut.
3. Pemilik yang merupakan *attribute relationship* dengan *user collections*, sehingga saat pengguna menggunakan aplikasi pakaian yang disajikan merupakan pakaiahn yang berkolerasi dengan pengguna itu sendiri tidak seluruh pakaian dalam basis data.



Gambar 3.27 Harga *collections*

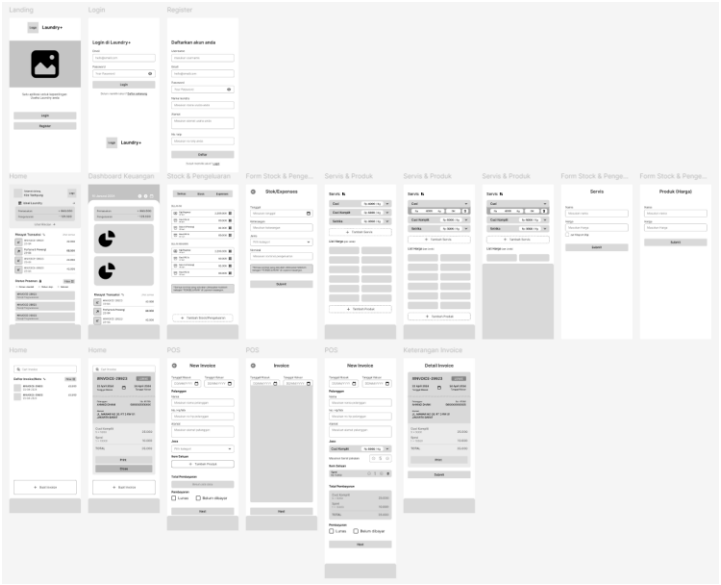
3.1.3.3 Perancangan antarmuka

Dalam tahapan ini berfokus kepada tampilan antarmuka aplikasi, aplikasi Figma digunakan dalam keseluruhan proses perancangan antarmuka. Hal-hal yang perlu dilakukan dalam proses ini yaitu pembuatan *design system*, *low fidelity*, dan *high fidelity prototype*. Design system adalah kumpulan dari berbagai aspek desain yang akan digunakan dalam suatu produk agar menjaga konsistensinya (Bergman, 2024). Aspek tersebut meliputi seperti logo, *typhography*, *colour palette* dan lain lain, untuk *design system* dari aplikasi ini dapat dilihat seperti pada Gambar 3.28.

Gambar 3.28 *Design system*

Dari pembuatan *design system* tersebut akhirnya diputuskanlah nama aplikasi yang sedang dibuat yaitu LaPlus yang merupakan singkatan dari *Laundry Plus*. Setelah memiliki

design system selanjutnya yaitu melakukan perancangan *low fidelity prototype*. *Low fidelity* digunakan untuk mendapatkan ide awal dari bentuk sederhana desain sebuah produk. *Low fidelity* untuk aplikasi LaPlus dapat dilihat pada Gambar 3.29, dalam gambar tersebut terlihat hanya berupa blok-blok dari *element* dan teks yang akan digunakan masih belum ada warna yang masuk dalam desain tersebut.



Gambar 3.29 Desain *low fidelity*

3.1.3.4 Proses pengujian

Proses pengujian dilakukan dalam dua tahap, yaitu pada tahap purwarupa dan tahap aplikasi akhir. Pengujian pertama pada purwarupa bertujuan untuk menguji aspek usabilitas, yang mencakup pengukuran persentase keberhasilan dan perhitungan skor SUS. Dalam pengujian persentase keberhasilan, pengguna diberikan sejumlah tugas yang harus diselesaikan. Tugas-tugas tersebut dirancang berdasarkan aktivitas yang terdapat dalam *usecase diagram*, dengan daftar tugas yang tercantum pada Tabel 3.7. Dari tugas yang diberikan, akan dihitung persentase keberhasilan untuk mengetahui sejauh mana pengguna dapat menyelesaikan seluruh tugas dengan sukses. Pengujian ini dilakukan pada lima pengguna, dan perhitungan persentase keberhasilan serta *error rate* menggunakan rumus yang tercantum pada persamaan (3.1) dan (3.2).

Tabel 3.7 Daftar tugas

No	Tugas
T1	Pengguna dapat membuat akun
T2	Pengguna dapat melakukan *edit profile*
T3	Pengguna dapat membuka halaman *invoice*
T4	Pengguna dapat membuka halaman detail *invoice*
T5	Pengguna dapat membuat invoice baru
T6	Pengguna dapat menyunting invoice
T7	Pengguna dapat menghapus invoice
T8	Pengguna dapat membuka halaman stok & pengeluaran
T9	Pengguna dapat membuat pengeluaran baru
T10	Pengguna dapat menyunting pengeluaran
T11	Pengguna dapat menghapus pengeluaran
T12	Pengguna dapat membuka halaman daftar harga
T13	Pengguna dapat menyunting harga jasa
T14	Pengguna dapat menghapus jasa
T15	Pengguna dapat membuat jenis pakaian
T16	Pengguna dapat menyunting jenis pakaian
T17	Pengguna dapat menghapus jenis pakaian

$$\%\ \text{Keberhasilan} = \frac{\text{Jumlah tugas yang sukses}}{\text{Jumlah seluruh tugas}} . 100\% \tag{3.1}$$

$$\%\ \text{Error} = 100\% - \%\ \text{Keberhasilan} \tag{3.2}$$

Untuk pengujian selanjutnya yaitu *System Usability Scale* (SUS) yang merupakan pengujian usabilitas popular yang dikembangkan oleh John Brooke pada tahun 1986. Pengujian ini bertujuan untuk mendapatkan masukan dari pengguna dengan menjawab 10 pertanyaan yang diberikan, daftar pertanyaan tersebut dapat dilihat pada Tabel 3.8. Pengguna dapat menjawab dengan memilih pilihan jawaban yang tersedia, terdapat 5 pilihan jawaban yang memiliki skor dari 1-5 dengan keterangan sangat tidak setuju hingga sangat setuju.

Tabel 3.8 Daftar pertanyaan untuk SUS

No	Pertanyaan SUS
1	Saya berpikir akan menggunakan sistem ini lagi
2	Saya merasa sistem ini rumit untuk digunakan
3	Saya merasa sistem ini mudah digunakan
4	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini
5	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya
6	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi pada sistem ini)
7	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat
8	Saya merasa sistem ini membingungkan
9	Saya merasa tidak ada hambatan dalam menggunakan sistem ini
10	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini

Pada pengujian SUS terdapat beberapa aturan dalam menghitung hasil skor dari pengguna, berikut 3 aturan yang ada pada perhitungan SUS (Susilo, 2019):

1. Setiap pertanyaan dengan nomor ganjil, skor akan dikurang 1.
2. Setiap pertanyaan dengan nomor genap, skor yang didapat pengguna akan menjadi nilai pengurangan dari skor maksimal.
3. Skor SUS yang didapat dari hasil penjumlahan skor seluruh pertanyaan dikali 2,5.

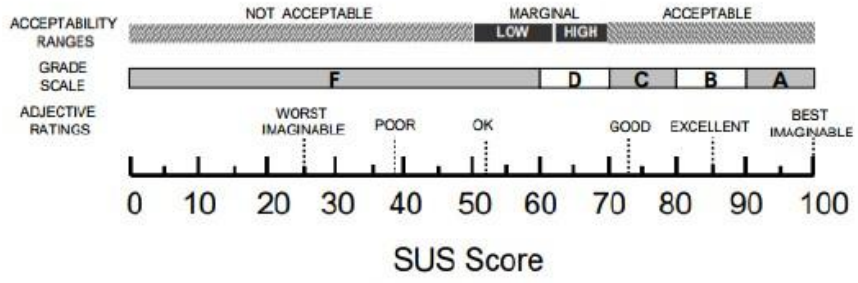
Setelah memperoleh data hasil pengujian, langkah berikutnya adalah menghitung rata-rata dari seluruh nilai yang telah diperoleh. Rumus perhitungan yang digunakan dapat dilihat persamaan (3.3) dengan keterangan yang dijelaskan pada Gambar 3.30. Hasil perhitungan ini akan menjadi nilai akhir rata-rata skor SUS yang kemudian disesuaikan dengan skala penilaian SUS sebagaimana ditunjukkan pada Gambar 3.31. Selanjutnya, nilai tersebut akan digunakan untuk menentukan keterangan terkait hasil pengujian SUS yang telah dilakukan.

$$\bar{x} = \frac{\sum x}{n} \quad (3.3)$$

$\bar{x}$ = skor rata-rata
$\sum x$ = jumlah skor SUS
n = jumlah responden

Gambar 3.30 Keterangan rumus SUS

Sumber : (Susilo, 2019)



Gambar 3.31 Penilaian SUS Skor

Sumber : (Susilo, 2019)

Pengujian kedua dilakukan pada aplikasi akhir dengan metode uji fungsionalitas untuk memastikan setiap fitur yang dikembangkan berjalan sesuai dengan kebutuhan dan spesifikasi yang telah ditetapkan. Pengujian ini bertujuan untuk mengevaluasi apakah aplikasi dapat berfungsi dengan baik tanpa adanya *bug* atau *error* yang signifikan. Setiap fungsi dalam aplikasi diuji berdasarkan skenario pengujian yang dirancang, mencakup validasi *input*, navigasi antar halaman, serta respon aplikasi terhadap berbagai tindakan pengguna. Pengujian ini juga dilakukan untuk memastikan integrasi antara komponen berjalan secara harmonis.

3.1.4 *Implementation*

Tahapan implementasi ini bertujuan untuk merealisasikan rancangan antarmuka yang telah ditentukan sebelumnya menjadi sebuah aplikasi yang fungsional. Expo, sebagai salah satu *library* dari React Native, memudahkan proses pengembangan aplikasi lintas platform dengan

berbagai fitur yang mendukung efisiensi kerja, dan kemudahan integrasi dengan modul-modul lainnya. Dalam tahap ini, pengembang memperbaiki kekurangan desain sistem dan mengubah tampilan desain kedalam kode sumber menggunakan JavaScript, membangun komponen antarmuka pengguna, dan mengintegrasikan logika bisnis serta fungsionalitas aplikasi.

BAB IV
HASIL DAN PEMBAHASAN

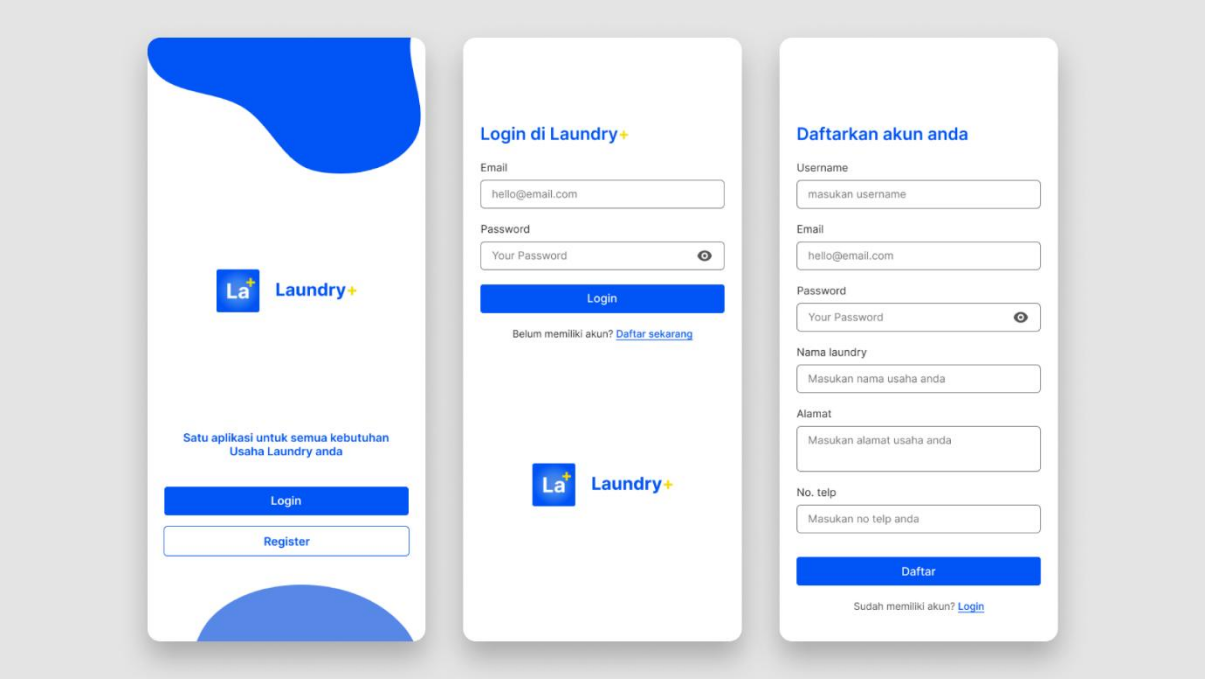
4.1 Hasil desain *high fidelity*

Hasil desain *high fidelity* pada aplikasi Laplus terbagi ke dalam enam kategori utama, yaitu autentikasi, menu utama, *profile*, *invoice*, stok & pengeluaran, serta daftar harga. Bagian autentikasi mencakup tampilan untuk proses autentikasi pengguna seperti login dan registrasi. Menu utama dirancang untuk memberikan akses cepat ke fitur-fitur utama aplikasi. Bagian *profile* menampilkan informasi pengguna yang dapat diedit sesuai kebutuhan. *Invoice* berfungsi untuk mengelola data transaksi dan tagihan secara sistematis. Stok & pengeluaran digunakan untuk mencatat pengeluaran yang terjadi. Terakhir, daftar harga memuat informasi tentang harga jasa atau jenis pakaian satuan yang ditawarkan, sehingga mempermudah pengguna dalam melakukan penyesuaian atau pengecekan harga.

4.1.1 Autentikasi

Pada bagian autentikasi pada aplikasi dirancang untuk mendukung proses akses awal pengguna dengan menyediakan fitur-fitur penting yang berkaitan dengan autentikasi dan pengenalan aplikasi. Bagian ini terdiri dari 3 halaman seperti yang bisa dilihat pada Gambar 4.1, dijelaskan sebagai berikut:

1. *Onboarding* adalah halaman pertama yang dilihat oleh pengguna saat membuka aplikasi untuk pertama kali, halaman ini berfungsi untuk pengenalan singkat tentang aplikasi dan navigasi antara halaman *login* dan *register*.
2. *Login* untuk autentikasi pengguna yang sudah terdaftar, pada halaman ini pengguna diharuskan memasukan email dan password sesuai dengan akun yang mereka miliki.
3. *Register* yang digunakan pengguna untuk pembuatan akun baru. Pengguna diminta memasukan data yang dibutuhkan meliputi nama pemilik dan usaha, nomor telepon, alamat, email, dan *password*.



Gambar 4.1 Halaman pada bagian autentikasi

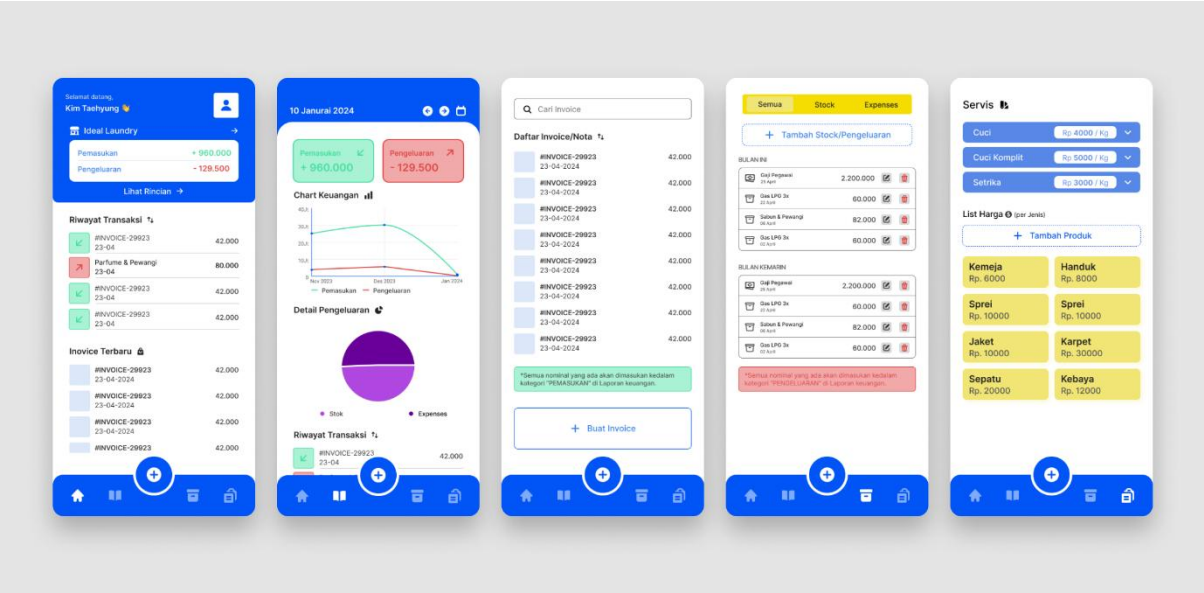
4.1.2 Menu utama

Bagian menu utama adalah bagian kunci pada aplikasi Laplus, yang dirancang untuk memudahkan pengguna mengakses fitur-fitur utama aplikasi. Bagian ini terdiri dari beberapa halaman seperti yang dapat dilihat pada Gambar 4.2, dijelaskan sebagai berikut:

1. *Home* adalah halaman yang terbuka saat pengguna login ke dalam aplikasi. Halaman ini menampilkan informasi penting seperti total pemasukan dan pengeluaran, serta dilengkapi dengan riwayat transaksi terbaru dan invoice terbaru untuk memudahkan pengguna memantau aktivitas keuangan. Selain itu pada halaman ini, pengguna dapat mengakses halaman *profile* melalui tombol dengan ikon pengguna yang terletak di bagian kanan atas.
2. *Dashboard* adalah halaman yang dirancang untuk memungkinkan pengguna melihat seluruh arus keuangan usaha mereka secara menyeluruh. Halaman ini dilengkapi dengan berbagai grafik yang memudahkan pengguna dalam memahami dan mengolah informasi keuangan yang tersedia. Selain itu, pengguna juga dapat menyaring data keuangan berdasarkan bulan untuk mempermudah analisis dan pemantauan keuangan sesuai kebutuhan.
3. *Invoice* adalah halaman yang dirancang untuk mendukung pelayanan transaksi atau *Point of Sales*(POS) serta menampilkan seluruh daftar *invoice*. Halaman ini

memudahkan pengguna dalam mengelola transaksi dengan pelanggan dan memantau riwayat *invoice* secara terstruktur.

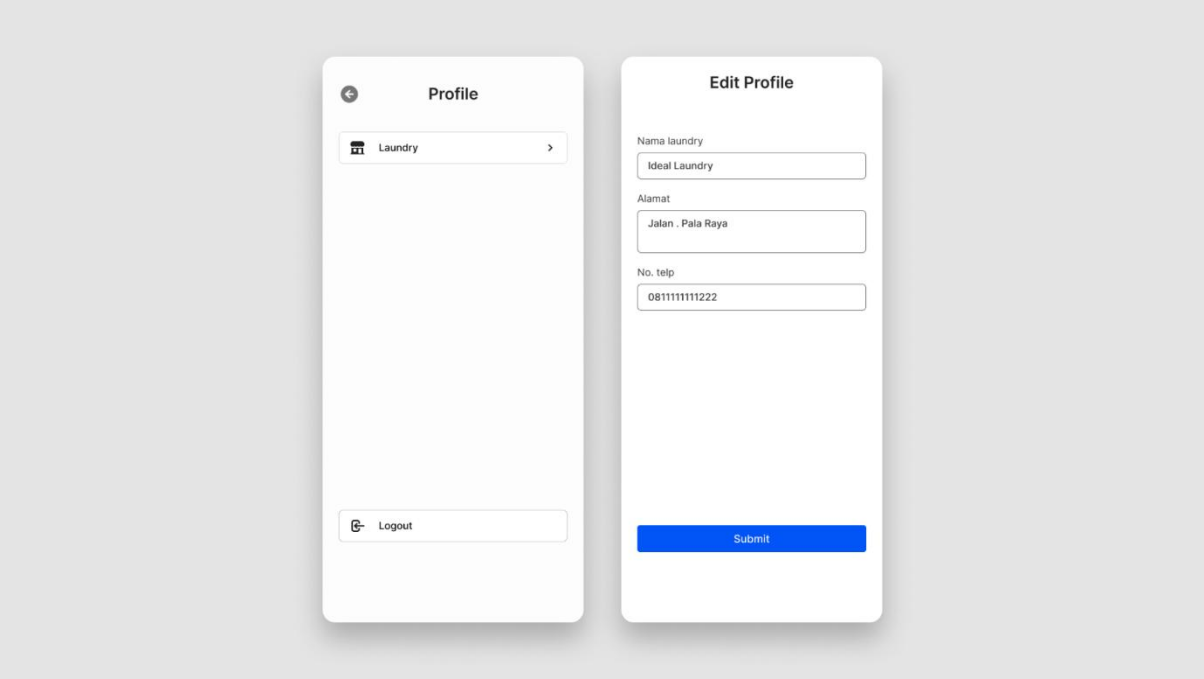
4. Stok/Pengeluaran adalah halaman yang dirancang untuk memantau pengeluaran usaha pengguna. Halaman ini membagi pengeluaran ke dalam dua kategori utama, yaitu pengeluaran untuk stok, yang berkaitan dengan pembelian barang atau bahan baku, dan pengeluaran untuk *expenses*, yang mencakup biaya operasional lainnya. Halaman ini membantu pengguna dalam mengelola dan mencatat setiap jenis pengeluaran secara terperinci.
5. Daftar harga adalah halaman yang berfungsi untuk mengelola dan mengatur harga jasa yang disediakan oleh pengguna, serta menetapkan harga jenis pakaian untuk layanan pencucian satuan. Halaman ini memungkinkan pengguna untuk melakukan penyesuaian harga dengan mudah sesuai kebutuhan bisnis.



Gambar 4.2 Halaman pada bagian menu utama

4.1.3 *Profile*

Bagian *profile* merupakan bagian yang berhubungan langsung dengan akun pengguna dan informasi terkait. Seperti yang dapat dilihat pada Gambar 4.3, bagian ini terdiri dari dua halaman utama. Halaman pertama adalah halaman *profile*, di mana pengguna dapat melihat informasi akun dan memiliki opsi untuk menekan tombol untuk *edit profile* atau *logout* dari aplikasi. Halaman kedua adalah halaman *edit profile*, yang dirancang untuk memungkinkan pengguna mengganti atau memperbarui informasi terkait usaha laundry mereka dengan mudah.



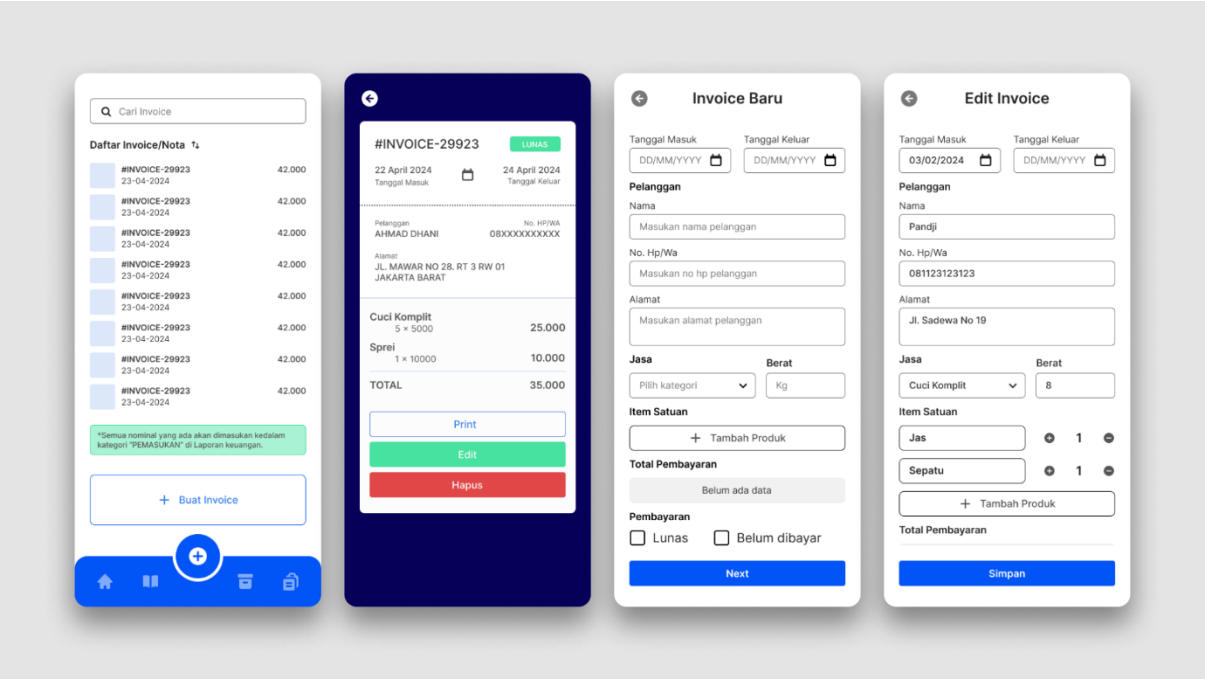
Gambar 4.3 Halaman pada bagian *profile*

4.1.4 Invoice

Bagian *invoice* merupakan elemen yang mendukung pelayanan transaksi atau *Point of Sales* (POS). Bagian ini tediri dari 4 halaman yang bisa dilihat pada Gambar 4.4, dijelaskan sebagai berikut:

1. Halaman invoice berisi daftar seluruh *invoice* yang telah dibuat oleh pengguna. Dalam halaman ini, pengguna dapat melakukan pencarian *invoice* berdasarkan nama atau nomor pelanggan yang tercantum. Selain itu, pengguna juga dapat mengakses halaman detail *invoice* dengan memilih salah satu *invoice* dari daftar tersebut..
2. Halaman detail *invoice* menyajikan seluruh informasi terkait suatu *invoice* secara lengkap. Pada halaman ini, pengguna memiliki pilihan untuk mengedit, menghapus, mengirim, atau mencetak invoice ke dalam bentuk fisik.
3. Halaman buat invoice adalah halaman yang dirancang untuk memungkinkan pengguna membuat invoice baru. Halaman ini berisi formulir yang harus diisi oleh pengguna, mencakup informasi seperti data pelanggan, tanggal, jenis jasa dan pakaian yang digunakan serta status pembayaran *invoice* tersebut.
4. Halaman *edit invoice* dirancang untuk memungkinkan pengguna melakukan perubahan pada *invoice* yang sudah dibuat sebelumnya. Setelah selesai, pengguna dapat

menyimpan perubahan tersebut agar data pada *invoice* diperbarui sesuai dengan kebutuhan.



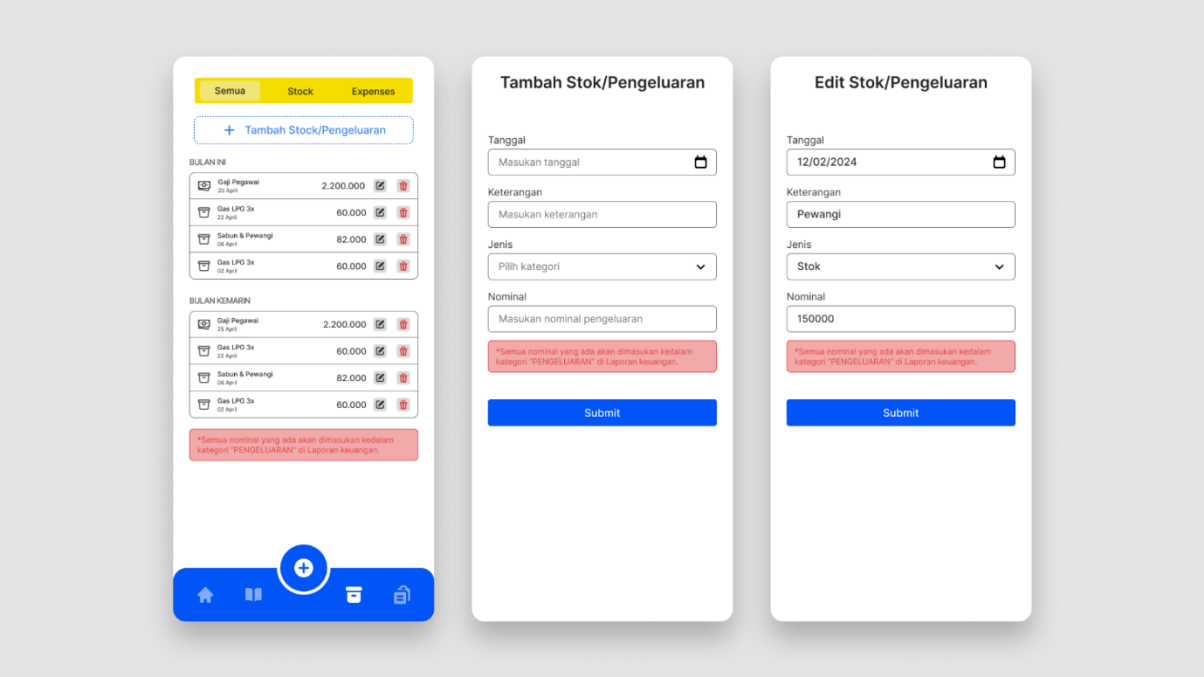
Gambar 4.4 Halaman pada bagian *invoice*

4.1.5 Stok/Pengeluaran

Bagian stok/pengleuaran digunakan untuk memantau dan mengelola pengeluaran dari usaha pengguna. Pengguna dapat membuat, menghapus, dan menyunting pengeluaran yang dimiliki. Seperti yang terdapat pada Gambar 4.5 bagian ini terdiri dari 3 halaman sebagai berikut:

1. Halaman stok/pengeluaran menampilkan daftar seluruh pengeluaran yang telah dibuat oleh pengguna. Pada halaman ini, pengguna dapat menyaring pengeluaran berdasarkan jenisnya, yaitu stok atau *expenses*. Selain itu, pengguna juga memiliki opsi untuk menyunting atau menghapus pengeluaran yang tercantum di halaman ini sesuai kebutuhan.
2. Halaman tambah stok/pengeluaran dirancang untuk memungkinkan pengguna membuat pengeluaran baru. Pada halaman ini, pengguna diminta untuk mengisi informasi yang diperlukan, seperti keterangan, nominal, dan jenis pengeluaran yang akan dicatat.
3. Halaman edit stok/pengeluaran digunakan untuk memperbarui data pengeluaran yang telah tercatat sebelumnya. Pada halaman ini, pengguna dapat mengubah informasi

seperti keterangan, nominal, atau jenis pengeluaran sesuai kebutuhan, lalu menyimpan perubahan untuk memperbarui data secara efektif.



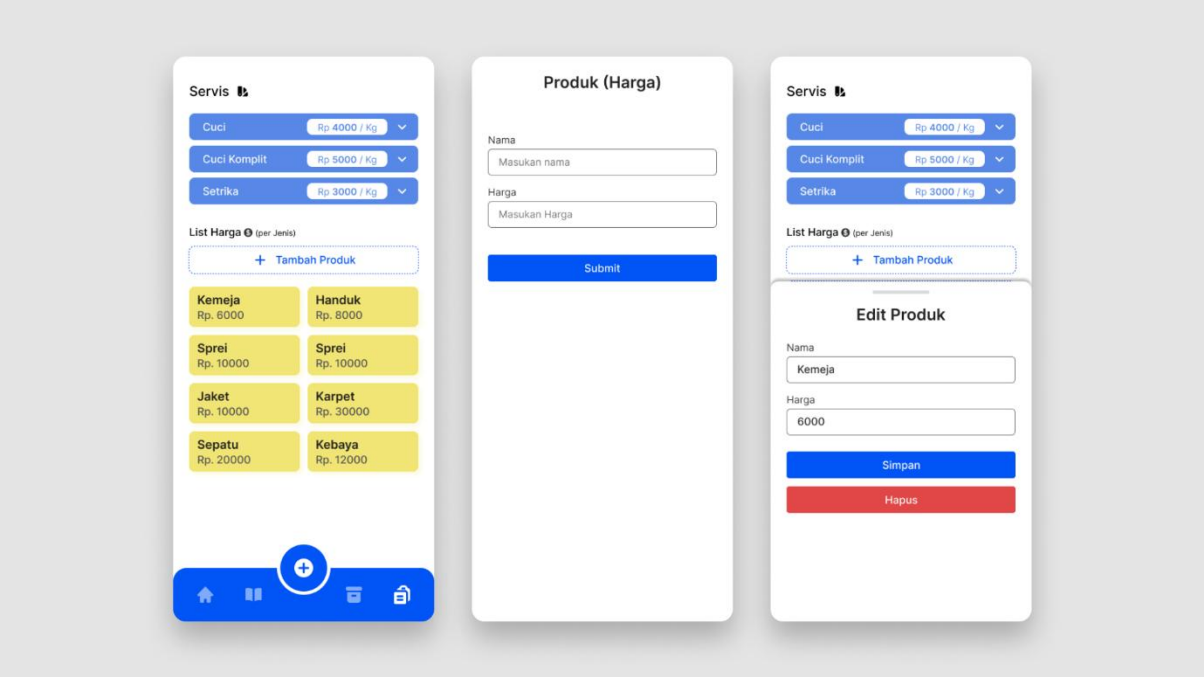
Gambar 4.5 Halaman pada bagian stok/pengeluaran

4.1.6 Daftar harga

Bagian daftar harga berfungsi untuk mengelola dan mengatur harga jasa yang disediakan oleh pengguna, serta menetapkan harga jenis pakaian untuk layanan pencucian satuan. Pada Gambar 4.6 dibawah ini dapat dilihat bahwa bagian ini terdiri dari 3 halaman dijelaskan sebagai berikut:

1. Halaman daftar harga menampilkan informasi lengkap mengenai harga barang atau jasa yang tersedia. Pada halaman ini, pengguna dapat melihat rincian harga yang telah ditetapkan, yang dapat digunakan sebagai referensi dalam transaksi atau penyusunan invoice.
2. Halaman buat jenis pakaian digunakan untuk menambahkan pakaian baru ke dalam sistem. Pada halaman ini, pengguna diminta mengisi informasi penting seperti keterangan, dan nominal harga yang diperlukan sebelum pakaian disimpan ke dalam daftar.
3. Halaman edit jenis pakaian memungkinkan pengguna memperbarui informasi terkait jenis pakaian yang telah ada. Pada halaman ini, pengguna dapat mengubah keterangan

dan nominal harga, lalu menyimpan perubahan untuk memastikan data pakaian tetap sesuai kebutuhan.



Gambar 4.6 Halaman pada bagian daftar harga

4.2 Hasil pengujian purwarupa

Pengujian pertama yang dilakukan pada purwarupa merupakan pengujian presentase keberhasilan, pengujian ini diberikan kepada 5 orang calon pengguna. Pengguna diminta melakukan tugas yang tertera pada Tabel 3.7, jika pengguna dapat melakukan tugas tersebut maka akan diberikan tanda (✓) dan jika gagal diberikan tanda (×). Tujuan dari pengujian ini adalah untuk mengetahui apakah hasil desain mudah dipahami dan dijalankan oleh pengguna. Hasil dari pengujian tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil pengujian presentase keberhasilan

No	Responden 1	Responden 2	Responden 3	Responden 4	Responden 5
T1	✓	✓	✓	✓	✓
T2	✓	×	×	✓	✓
T3	✓	×	×	✓	✓
T4	×	✓	✓	×	×
T5	✓	✓	✓	✓	✓
T6	✓	✓	✓	✓	✓
T7	✓	✓	✓	✓	✓

T8	✓	×	✓	✓	×
T9	✓	✓	✓	✓	✓
T10	✓	×	×	✓	✓
T11	✓	✓	×	✓	✓
T12	✓	×	✓	✓	✓
T13	✓	✓	✓	✓	✓
T14	✓	✓	✓	✓	✓
T15	✓	✓	✓	✓	✓
T16	✓	✓	✓	✓	✓
T17	✓	✓	✓	✓	✓

Berdasarkan pengujian pada purwarupa aplikasi, diperoleh hasil presentase keberhasilan sebesar 84% dan *error rate* sebesar 16%. Nilai ini menunjukkan bahwa sebagian besar alur dan elemen antarmuka dalam purwarupa dapat dipahami dan digunakan oleh pengguna dengan cukup baik. Namun, masih terdapat beberapa kendala yang dialami oleh responden selama pengujian. Salah satu masalah yang paling banyak dikeluhkan adalah ketiadaan nama atau label pada menu halaman, yang menyebabkan kebingungan, terutama bagi pengguna yang pertama kali mencoba purwarupa. Oleh karena itu, meskipun skor keberhasilan mencapai angka yang cukup baik, perbaikan pada aspek visual dan navigasi, seperti penambahan teks penjelas atau ikon yang lebih informatif, menjadi hal penting yang perlu diimplementasikan dalam versi final aplikasi agar meningkatkan kemudahan penggunaan dan pengalaman pengguna secara keseluruhan.

$$\% \text{ Keberhasilan} = \frac{72}{85}.100\% = 84\%$$

$$\% \text{ Error} = 100\% - 84\% = 16\%$$

Tabel 4.2 Hasil pengujian SUS skor

No	Responden	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Jumlah	Nilai (Jumlah x 2,5)
1	Pengguna 1	3	3	3	3	4	4	2	3	3	3	31	77.5
2	Pengguna 2	2	2	2	3	3	4	3	2	2	2	25	62.5
3	Pengguna 3	3	3	2	2	4	4	2	3	2	3	28	70
4	Pengguna 4	4	4	3	3	4	4	3	3	3	4	35	87.5
5	Pengguna 5	4	3	3	4	4	4	2	3	3	2	32	80
Skor rata-rata													75.5

Berdasarkan hasil pengujian usability menggunakan metode System Usability Scale (SUS), diperoleh skor akhir sebesar 75,5. Nilai ini menunjukkan bahwa prototipe aplikasi berada pada kategori "baik" (*good usability*) dan mendekati batas atas dari kategori "*acceptable*" menurut standar interpretasi SUS. Skor ini menandakan bahwa secara umum, pengguna merasa cukup puas terhadap kemudahan penggunaan, struktur navigasi, serta tampilan antarmuka yang terdapat dalam purwarupa. Pengguna mampu menyelesaikan tugas-tugas yang diuji tanpa mengalami hambatan yang signifikan, meskipun terdapat beberapa aspek minor yang perlu diperbaiki. Meskipun demikian, kesan keseluruhan terhadap aplikasi tetap positif dan dapat dijadikan dasar yang kuat untuk pengembangan ke tahap implementasi. Dengan skor 75,5, prototipe dinilai layak untuk dilanjutkan ke tahap pengembangan, namun tetap perlu penyempurnaan pada aspek kegunaan agar pengalaman pengguna menjadi lebih optimal.

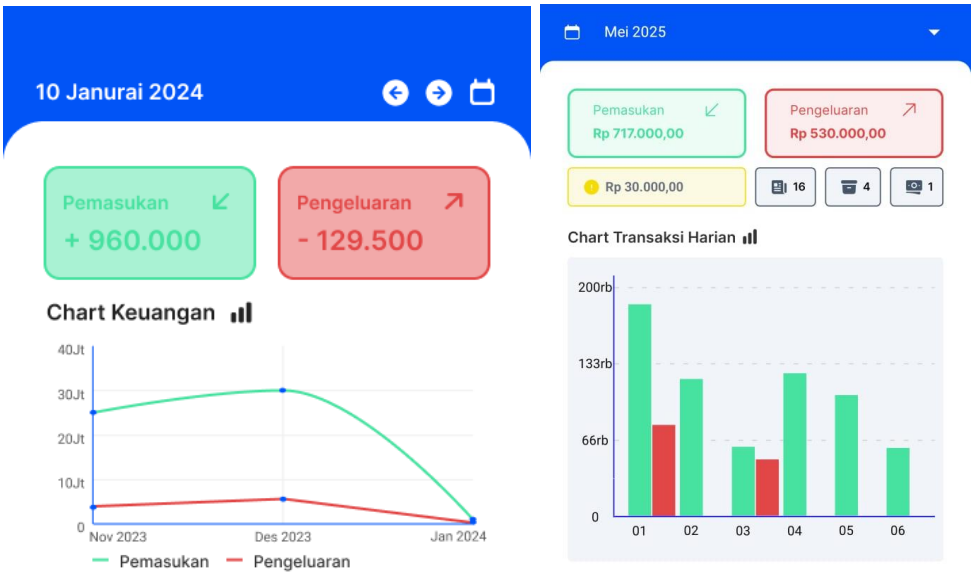
4.3 Hasil Implementasi

Implementasi ini merupakan proses mengubah desain purwarupa menjadi aplikasi *mobile* yang fungsional. Teknologi yang digunakan dalam proses ini adalah Expo, salah satu library dari React Native, yang dipilih karena aplikasi yang dikembangkan ditujukan untuk perangkat mobile. Pada bagian ini akan dijelaskan perubahan-perubahan yang terjadi pada beberapa halaman dari desain awal, serta pengimplementasian beberapa potongan kode sumbernya. Penjelasan mengenai halaman-halaman yang mengalami perubahan disampaikan sebagai berikut:

4.3.1 Halaman dashboard

Pada halaman dashboard perubahan yang terjadi mencakup banyak elemen, dari perubahan *line chart* menjadi *bar chart*, dan penambahan satu *chart* untuk memantau pemasukan harian serta detail total jumlah masing-masing pengeluaran dan total pembayaran yang belum lunas. Perubahan ini dilakukan agar pengguna dapat melihat lebih detail arus keuangan usahanya sehingga dapat membantu keputusan pengguna dalam melakukan operasional usahanya. Contohnya seperti melakukan promosi pada hari di mana arus pemasukan mengalami penurunan. Penjelasan mengenai kode halaman dashboard sebagai berikut:

4.3.1.1 Bagian atas dan chart harian



(a) Desain purwarupa (b) Desain final

Gambar 4.7 Perbandingan (a)desain purwarupa dan (b)desain final bagian atas halaman *dashboard*

Seperti yang ditunjukan pada gambar di atas, bagian atas halaman *dashboard* mencakup fitur, antara lain *filter* berdasarkan bulan, total pemasukan dan pengeluaran, detail pembayaran belum lunas, jumlah *invoice*, stok, serta data pengeluaran (*expenses*). Kode untuk bagian antarmuka pengguna (*frontend*) dapat dilihat pada Gambar 4.8 dan Gambar 4.9. Sementara itu, Gambar 4.10 menampilkan implementasi pemanggilan fungsi API yang berperan dalam pengelolaan data pada sisi *backend*.

```
<SafeAreaView className="flex-1 bg-white h-full">
 <View className="bg-Primary w-full h-20 px-4">
  <View className="flex-row items-center">
   <CalendarIcon color='#fff' height={16} />
   <Picker
      selectedValue={selectedMonth}
      onValueChange={(value) => {
        setSelectedMonth(value);
      }}
      style={{ height: 50, width: (screenWidth - 48),  color: 'white' }}
      dropdownIconColor={'white'}
   >
    <Picker.Item label="Pilih bulan" value={null} />
    {formattedMonths.map((month) => (
     <Picker.Item key={month.value} label={month.label} value={month.value} />
    ))}
   </Picker>
  </View>
 </View>
 <View className="flex-1 w-full h-full px-6 py-6 bg-white rounded-t-xl
   mt-[-30px]">
  <ScrollView
     ref={scrollViewRef}
     refreshControl={
       <RefreshControl
         refreshing={refreshing}
         onRefresh={onRefresh}
       />
     }
     showsVerticalScrollIndicator={false}
     showsHorizontalScrollIndicator={false}
  >
  <View className='flex flex-row justify-between gap-4 mb-2'>
   <View className="flex-1 bg-Hijau-2/20 border-solid border-Hijau border-2
    rounded-lg py-1 px-4 gap-1">
    <View className="flex flex-row items-center justify-between">
     <Text className="text-Hijau font-iregular text-sm">Pemasukan</Text>
      <ArrowDownLeftIcon color='#47E1A0' height={16} />
    </View>
    <Text className="text-Hijau text-sm font-ibold mb-2">
     {formatRupiah(totalPemasukan)}
    </Text>
   </View>
```

```
42    <View className="flex-1 bg-Merah-2/20 border-solid border-Merah border-2
        rounded-lg px-4 py-1 gap-1">
43     <View className="flex flex-row items-center justify-between">
44      <Text className="text-Merah font-iregular text-sm">Pengeluaran</Text>
45       <ArrowUpRightIcon color='#E14747' height={16}/>
46     </View>
47     <Text className="text-Merah text-sm font-ibold mb-2">
48      {formatRupiah(totalPengeluaran)}
49     </Text>
50    </View>
51   </View>
52   <View className="w-full flex flex-row space-x-2 mb-4">
53    <View className="flex-1 flex-row items-center bg-slate-100 border-slate-600
       p-2 rounded border">
54     <ExclamationCircleIcon color='#475569' height={16} />
55     <Text className="text-xs font-isemibold text-Hitam-1">
56      {formatRupiah(totalPending)}
57     </Text>
58    </View>
59    <View className="flex flex-row items-center bg-slate-100 border-slate-600
       p-2 rounded border">
60     <NewspaperIcon color='#475569' height={16} />
61     <Text className="text-xs font-isemibold text-Hitam-1">
62      {pemasukanCount}
63     </Text>
64    </View>
65    <View className="flex flex-row items-center bg-slate-100 border-slate-600
       p-2 rounded border">
66     <ArchiveBoxIcon color='#475569' height={16} />
67     <Text className="text-xs font-isemibold text-Hitam-1">{totalStock}</Text>
68    </View>
69    <View className="flex flex-row items-center bg-slate-100 border-slate-600
       p-2 rounded border">
70     <BanknotesIcon color='#475569' height={16} />
71     <Text className="text-xs font-isemibold text-Hitam-1">
72      {totalExpenses}
73     </Text>
74    </View>
75   </View>
```

Gambar 4.8 Kode *frontend* untuk bagian atas halaman *dashboard*

```
<View className="flex flex-row items-center">
 <Text className="text-Hitam-2 text-base font-isemibold">
  Chart Pemasukan Harian
 </Text>
 <ChartBarIcon color="#333" height={16} strokeWidth={2}/>
</View>
<View className="mt-2 mb-8 bg-slate-100 px-2 py-4 rounded">
<BarChart
  data={barDataByDate}
  width={chartWidth}
  height={200}
  barWidth={20}
  spacing={4}
  initialSpacing={12}
  xAxisThickness={1}
  yAxisThickness={1}
  yAxisTextStyle={{ color: 'black' }}
  yAxisColor={'blue'}
  xAxisColor={'blue'}
  noOfSections={3}
  yAxisLabelWidth={32}
  formatYLabel={formatYLabel}
  maxValue={Math.max(...barDataByDate.map((item) => item.value)) + 1000}
 />
</View>
```

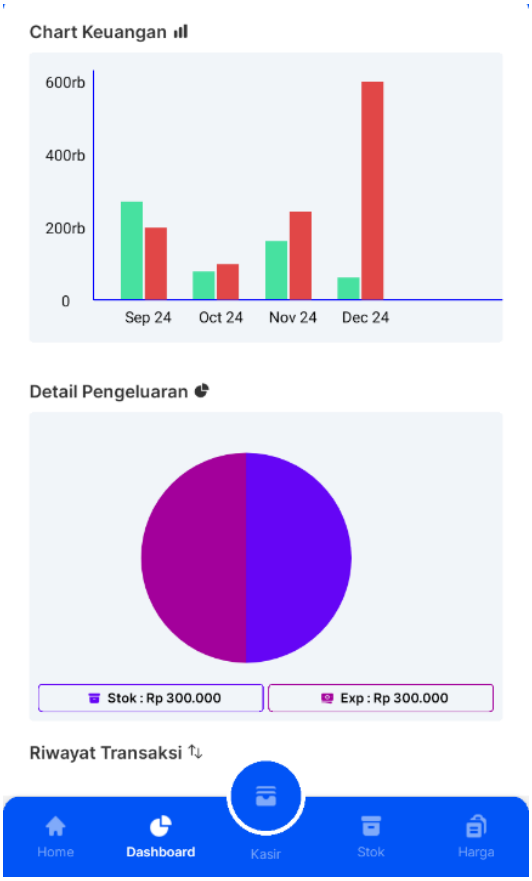
Gambar 4.9 Kode *frontend* untuk bagian *chart* harian

```
//Get All Pemasukan
export async function getUserPemasukan(userId){
    try {
        const pemasukan = await databases.listDocuments(
            config.databaseId,
            config.pemasukanCollectionId,
            [
                Query.equal('pemilik', userId)
            ]
        );
        return pemasukan.documents;
    } catch (error) {
        throw new Error(error);
    }
}
//Get All Pengeluaran
export async function getUserPengeluaran(userId){
    try {
        const pengeluaran = await databases.listDocuments(
            config.databaseId,
            config.pengeluaranCollectionId,
            [
                Query.equal('pemilik', userId)
            ]
        );
        return pengeluaran.documents;
    } catch (error) {
        throw new Error(error);
    }
}

//pemanggilan API pada bagian frontend
const { data: pemasukan, refetch: refetchPemasukan } = useAppwrite(() =>
getUserPemasukan(user.$id));
  const { data: pengeluaran, refetch: refetchPengeluaran } = useAppwrite(() =>
getUserPengeluaran(user.$id));
```

Gambar 4.10 Kode *backend* untuk halaman *dashboard*

4.3.1.2 Bagian chart keuangan dan chart pengeluaran



Gambar 4.11 Tampilan akhir bagian *chart* keuangan bulanan dan pengeluaran

Pada halaman dashboard terdapat dua *chart* lagi selain *chart* keuangan harian, yaitu *chart* pemasukan dan pengeluaran bulanan, dan detail pengeluaran seperti yang ditunjukan pada Gambar 4.11. *Chart* keuangan bulanan menggunakan *bar chart* untuk menyajikan datanya, implementasi kode untuk bagian *chart* keuangan dapat dilihat pada Gambar 4.12 sedangkan pada Gambar 4.13 merupakan kode untuk bagian detail pengeluaran yang menggunakan *pie chart*. Penggunaan *react-native-gifted-chart* dari *react library* yang digunakan untuk membuat *chart* pada halaman ini.

```
<View className="flex flex-row items-center">
 <Text className="text-Hitam-2 text-base font-isemibold">
  Chart Keuangan
 </Text>
 <ChartBarIcon color="#333" height={16} strokeWidth={2}/>
</View>
<View className="mt-2 mb-8 bg-slate-100 px-2 py-4 rounded">
 <BarChart
    data={flatBarData}
    width={chartWidth}
    height={200}
    barWidth={20}
    spacing={24}
    hideRules
    xAxisThickness={1}
    yAxisThickness={1}
    yAxisTextStyle={{ color: 'black' }}
    yAxisColor={'blue'}
    xAxisColor={'blue'}
    noOfSections={3}
    maxValue={maxValue}
    formatYLabel={formatYLabel}
    yAxisLabelWidth={50}
    onPress={handleBarPress}
 />
 {selectedBar && (
   <View style={{ position: 'absolute', left: 70+selectedBar.index*30, top:
    220, }}>
    <Text className="p-1 bg-white border border-Hitam-4 text-Hitam-2
     font-isemibold rounded">
 {selectedBar.value.toLocaleString('id-ID',{style:'currency',currency:'IDR'})}
    </Text>
   </View>
 )}
</View>
```

Gambar 4.12 Kode *frontend* untuk bagian chart keuangan

```
<View className="flex">
 <View className="flex flex-row items-center">
  <Text className="text-Hitam-2 text-base font-isemibold">
   Detail Pengeluaran
  </Text>
  <ChartPieIcon color="#333" height={16}/>
 </View>
 <View className="mt-2 flex bg-slate-100 p-2 rounded mb-4">
 <PieChart
    data={dataChart}
    width={screenWidth}
    height={240}
    chartConfig={{ color: (opacity = 1) => `rgba(0, 0, 0, ${opacity})`,}}
    accessor="amount"
    backgroundColor="#f1f5f9"
    borderRadius="20"
    hasLegend={false}
    center={[70, 5]}
    absolute
 />
 <View className="flex flex-row justify-around w-full space-x-1">
  {dataChart.map((item, index) => (
    <View key={index} className="flex-1 flex-row items-center justify-center
     border border-solid py-1 rounded" style={{ borderColor: item.color }}>
     <View>
      { item.name === "Stok : " ?
        <ArchiveBoxIcon color='#6505f5' height={12} /> :
        <BanknotesIcon color='#a3009b' height={12} />}
     </View>
     <Text className="text-xs font-isemibold">{item.name}</Text>
     <Text className="text-xs font-isemibold">
      {formatRupiah(item.amount)}
     </Text>
    </View>
  ))}
 </View>
</View>
</View>
```

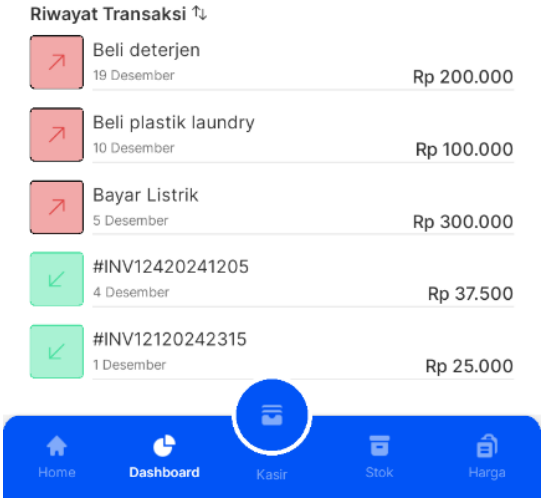
Gambar 4.13 Kode *frontend* untuk bagian detail pengeluaran

4.3.1.3 Bagian chart penggunaan jasa

Gambar 4.14 Tampilan *chart* penggunaan jasa

Penambahan *chart* penggunaan jasa dalam aplikasi bertujuan untuk memberikan gambaran visual mengenai frekuensi penggunaan masing-masing jenis jasa oleh pelanggan dalam periode tertentu. Fitur ini sangat berguna bagi pemilik usaha *laundry* karena dapat membantu dalam menganalisis layanan mana yang paling diminati dan memiliki kontribusi tertinggi terhadap pendapatan. Melalui informasi ini, pemilik dapat mengambil keputusan strategis seperti penyesuaian kapasitas layanan, promosi pada jasa tertentu, atau bahkan pengembangan jasa baru yang sesuai dengan preferensi pelanggan. Tampilan dari fitur ini disajikan dalam bentuk grafik yang informatif dan mudah dipahami seperti yang ditunjukkan pada Gambar 4.14.

4.3.1.4 Bagian riwayat transaksi



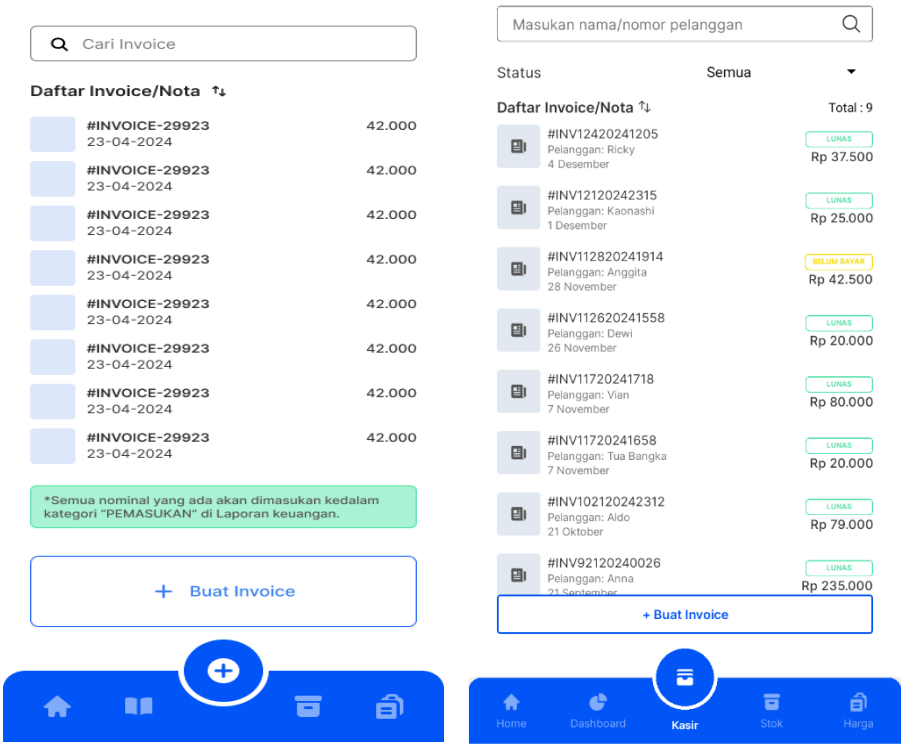
Gambar 4.15 Tampilan akhir bagian riwayat transaksi pada halaman *dashboard*

Pada bagian akhir halaman *dashboard* terdapat komponen riwayat transaksi yang menampilkan pemasukan dan pengeluaran berdasarkan urutan waktu pembuatan. Bagian akhir halaman tersebut ditunjukan pada Gambar 4.15. Komponen riwayat transaksi tidak hanya ditampilkan pada halaman *dashboard*, tetapi juga pada tersedia pada halaman beranda. Implementasi kode *frontend* untuk bagian riwayat transaksi dapat dilihat pada Gambar 4.16.

```
1  const renderItem = ({ item }) => (
2   <View className='my-2 flex flex-row w-full '>
3   <View className={`${item.bgColor} flex w-12 h-12 rounded border
     border-  ${item.color} mr-2 items-center justify-center`}>
4   {item.type === 'pemasukan' ? <ArrowDownLeftIcon color='#47E1A0' height={20}/>
    : <ArrowUpRightIcon color='#E14747' height={20} />}
5   </View>
6   <View className="flex flex-1" >
7    <Text className="text-sm text-Hitam-2 font-imedium">
8   {item.type === 'pemasukan' ? `${item.invoiceNumber}` : `${item.customerName}`}
9    </Text>
10  <View className="flex flex-row justify-between items-center space-x-1
     border-b-[0.5px] border-Hitam-4/30 ">
11  <Text className= "text-xs font-iregular text-Hitam-4">
12   {formatDate(item.date)}
13  </Text>
14  <Text className={`text-sm text-Hitam-2 font-imedium`}>
15   {formatRupiah(item.amount)}
16  </Text>
17  </View>
18  </View>
19  <Text className="text-sm text-gray-500"></Text>
20  </View>
21 );
```

Gambar 4.16 Kode frontend untuk bagian riwayat transaksi

4.3.2 Halaman invoice



(a) Desain purwarupa (b) Desain final

Gambar 4.17 Perbandingan (a)desain purwarupa dan (b)desain final halaman *invoice*

(b) Perubahan pada halaman *invoice* (kasir) yaitu penambahan *filter* untuk status pembayaran *invoice* dan penambahan detail kecil seperti nama pelanggan dan keterangan status pada *invoice card* yang disajikan. Perbandingan tampilan purwarupa dan tampilan akhir untuk halaman *invoice* bisa dilihat pada Desain purwarupa (b) Desain final

Gambar 4.17. Untuk pembuatan fitur *filter* menggunakan *react picker* yang terdapat pada *react library* contoh implementasinya terdapat pada Gambar 4.18 dalam baris ke 15-24. Dan untuk kode bagian *invoice card*-nya terdapat pada Gambar 4.19, kode tersebut nantinya dipanggil pada fungsi *flatlist* yang terdapat pada baris ke 37 dalam Gambar 4.18.

```
<SafeAreaView className="bg-white h-full p-8">
 <View className="relative bg-Merah mb-10">
  <FormField
     value={searchTerm}
     placeholder="Masukan nama/nomor pelanggan"
     handleChangeText={setSearchTerm}
     otherStyles='flex-1 top-0'
  />
  <View className="absolute right-3 top-2.5">
   <MagnifyingGlassIcon color="#333" height={24} />
   </View>
  </View>
  <View className="flex-row items-center mt-2 justify-between">
   <Text className="text-Hitam-2 text-base font-iregular">Status</Text>
   <Picker
      selectedValue={statusFilter}
      onValueChange={(value) => setStatusFilter(value)}
      style={{ height: 50, width: 200}}
      dropdownIconColor={'black'}
   >
     <Picker.Item label="Semua" value="Semua" />
     <Picker.Item label="Lunas" value="Lunas" />
     <Picker.Item label="Belum Bayar" value="Belum Bayar" />
   </Picker>
  </View>
  <View className="flex flex-row items-center justify-between">
   <View className="flex flex-row items-center">
    <Text className="text-Hitam-2 text-base font-isemibold">
     Daftar Invoice/Nota
    </Text>
    <ArrowsUpDownIcon color="#333" height={16}/>
   </View>
   <Text>Total : {totalInvoice}</Text>
  </View>
  <FlatList
     data={filteredData}
     renderItem={invoiceOnly}
     keyExtractor={(item) => item.id}
     refreshControl={
       <RefreshControl
         refreshing={refreshing}
         onRefresh={onRefresh}
       />
     }
```

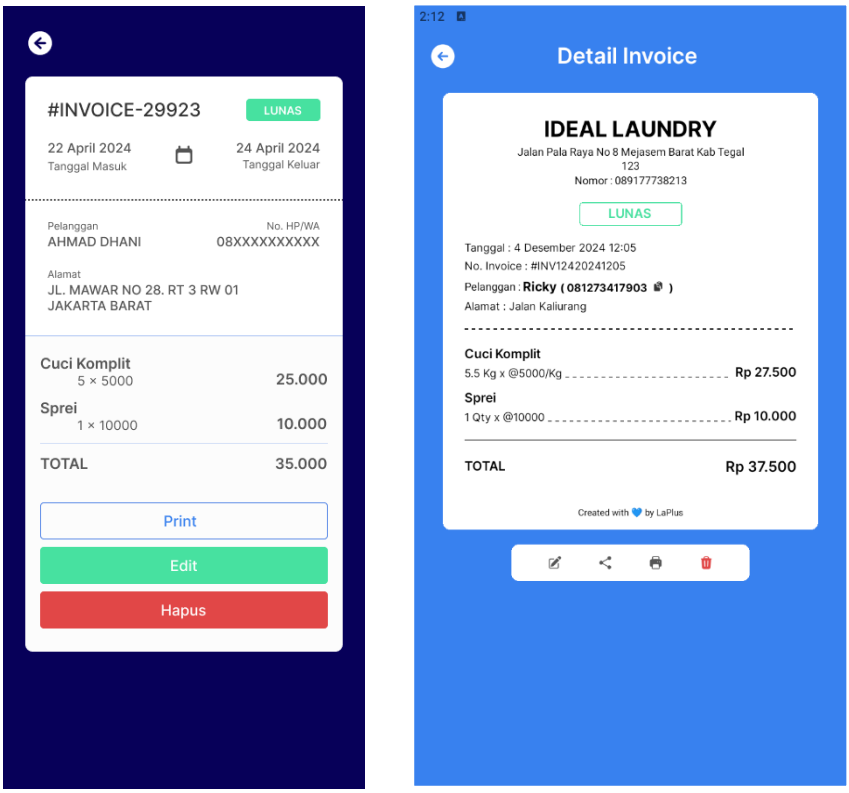
```
45      ListEmptyComponent={
46        <View className="mt-2 bg-slate-50 w-full h-20 border-[0.5px]
47          border-Hitam-4/10 rounded justify-center">
48          <Text className="text-center text-sm font-imedium text-Hitam-4">
49           Tidak ada invoice
50          </Text>
51         </View>
52      }
53      scrollEnabled={true}
54   />
55   <PrimaryButton
56      title= '+ Buat Invoice'
57      handlePress= {() => router.push('/createInvoice')}
58      containerStyles='w-full mb-4 bg-white border-solid border-2
                         border-Primary'
59      textStyle='text-sm text-Primary font-isemibold'
60   />
61 </SafeAreaView>
```

Gambar 4.18 Kode frontend halaman invoice

```
//Components Invoice card
1  const invoiceOnly = ({ item }) => (
2   <TouchableOpacity
3      className="my-2 flex flex-row"
4      onPress={ ()=> (
5        handleDetail(item)
6      )}
7   >
8    <View className="bg-slate-200 w-12 h-12 rounded mr-2 items-center
      justify-center">
9     <NewspaperIcon color='#555' height={20} />
10   </View>
11   <View className="flex flex-row flex-1 flex-wrap justify-between
      items-center">
12    <View className="flex justify-start">
13     <View className="flex flex-row items-center">
14      <Text className="text-sm text-Hitam-2 font-iregular">
15        {item.invoiceNumber}
16      </Text>
17     </View>
18     <Text className="text-xs font-iregular text-Hitam-4">
19       Pelanggan: {item.customerName}
20     </Text>
21     <Text className="text-xs font-iregular text-Hitam-4">
22       {formatDate(item.date)}
23     </Text>
24    </View>
25    <View className="flex-wrap items-end">
26     <Text className={`min-w-[75px] bg-white rounded px-2 py-0.5
        text-[8px] text-center font-isemibold border ${item.status ===
        true ? 'text-Hijau border-Hijau' :'text-Secondary
        border-Secondary'}`}>
27         {item.status ? 'LUNAS' : 'BELUM BAYAR'}
28     </Text>
29     <Text className='text-sm text-Hitam-2 font-imedium'>
30       {formatRupiah(item.amount)}
31     </Text>
32    </View>
33   </View>
```

Gambar 4.19 Kode render untuk components card invoice

4.3.3 Halaman detail invoice



(a) Desain purwarupa (b) Desian final

Gambar 4.20 Perbandingan (a)desain purwarupa dan (b)desain final detail *invoice*

Halaman detail invoice merupakan halaman yang muncul setelah kita menekan salah satu *invoice card* yang tersedia. Halaman ini berisikan seluruh informasi mengenai *invoice*/nota sebagai data pemasukan, informasi yang ada antara lain, tanggal transaksi, nama dan nomor hp pelanggan, jenis jasa dan pakaian serta total jumlah tagihan. Perubahan yang terjadi pada halaman ini terdapat pada perubahan tombol menu aksi yang diganti menggunakan ikon, perbandingan tampilan akhir dan tampilan sebelumnya untuk halaman detail *invoice* dapat dilihat pada Gambar 4.20. Implementasi kode *frontend* untuk halaman ini ditunjukan pada Gambar 4.21, sedangkan Gambar 4.22 merupakan kode untuk komponen detail *card* yang dipanggil pada baris ke 11 pada Gambar 4.21. Pada halaman ini terdapat menu aksi yang memiliki fitur sendiri, antara lain *edit*, *print*, *share*, dan hapus.

```
<SafeAreaView className="bg-Primary-2 h-full p-4">
 <View className="flex flex-row items-center justify-between mb-6">
  <TouchableOpacity onPress={router.back}>
   <ArrowLeftCircleIcon color="#FFF" width={32} height={32} />
  </TouchableOpacity>
  <Text className="text-2xl text-white font-isemibold ml-[-40px]">
   Detail Invoice
  </Text>
  <View></View>
 </View>
 {renderCard()} //Komponen detail card
 <View className="px-4 mt-4 items-center">
  <View className="w-50% flex-row bg-white rounded-lg px-10 py-2 space-x-10">
   <TouchableOpacity
     className="flex-row space-x-2 h-6 items-center justify-start"
     onPress={() => (handleEdit(item))}
   >
    <PencilSquareIcon color="#555" width={16} height={16} />
   </TouchableOpacity>
   <TouchableOpacity
     className="h-6 justify-center"
     onPress={sendWhatsAppMessage}
   >
    <ShareIcon color="#555" width={16} height={16} />
   </TouchableOpacity>
   <TouchableOpacity className="h-6 justify-center" onPress={printPDF}>
    <PrinterIcon color="#555" width={16} height={16} />
   </TouchableOpacity>
   <TouchableOpacity
     className="h-6 justify-center"
     onPress={() => handleDelete(item.id)}
   >
    <TrashIcon color="#E14747" width={16} height={16} />
   </TouchableOpacity>
  </View>
 </View>
</SafeAreaView>
```

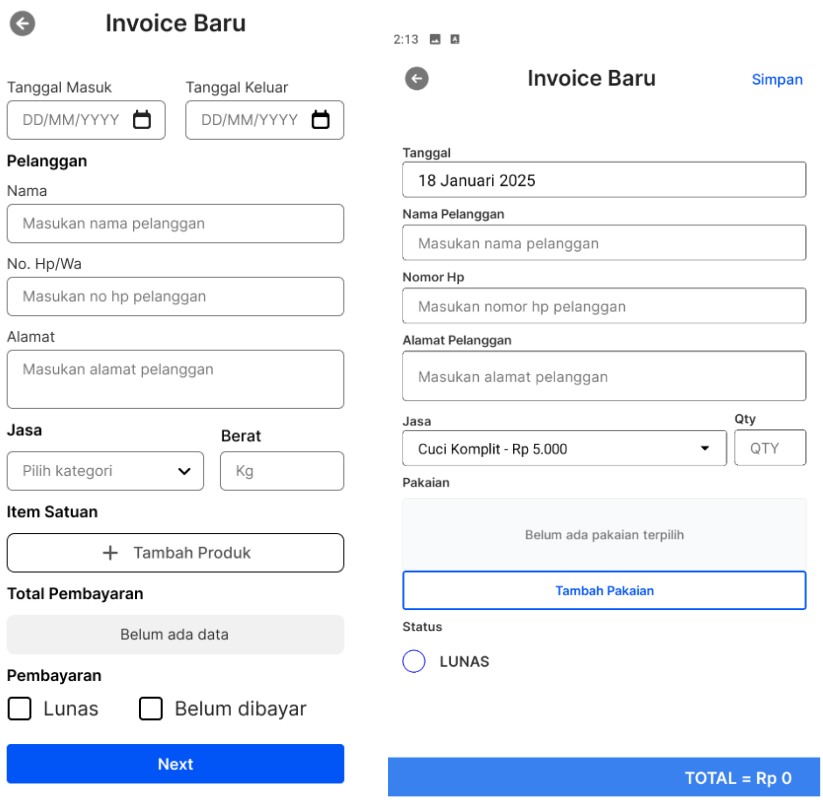
Gambar 4.21 Kode *frontend* halaman detail *invoice*

```
1  const renderCard = () => {
2  if (item) {
3  return (
4  <View className="px-4">
5  <View className="bg-white rounded-lg px-6 pt-6 pb-3 flex items-center">
6   <Text className="text-2xl font-ibold">{user.laundry.toUpperCase()}</Text>
7   <Text className="text-center text-xs w-[70%]">{user.alamat}</Text>
8   <Text className="text-xs mb-4">Nomor : {user.nomor}</Text>
9   <Text className={`min-w-[75px] bg-white rounded px-8 py-0.5 text-sm
     font-isemibold border ${item.status === true ? 'text-Hijau border-Hijau'
     :'text-Secondary border-Secondary'}`}>
10    {item.status ? "LUNAS" : "BELUM BAYAR"}
11  </Text>
12  <View className="w-full mt-4 space-y-1">
13   <Text className="text-xs font-iregular">
14    Tanggal : {formatDate(item.date)}
15   </Text>
16   <Text className="text-xs font-iregular">
17    No. Invoice : #INV{formatName(item.date)}
18   </Text>
19   <View className="flex flex-row items-center">
20    <Text className="text-xs">Pelanggan : </Text>
21    <View className="flex-row items-end space-x-1">
22     <Text className="text-sm font-ibold">{item.customerName}</Text>
23     <TouchableOpacity className="items-center" onPress={copyToClipboard}>
24      <Text className="text-xs font-ibold">
25       ({item.nomorhp}<ClipboardDocumentIcon color="#333" height={12}/>)
26      </Text>
27     </TouchableOpacity>
28    </View>
29   </View>
30   <Text className="text-xs font-iregular">Alamat : {item.alamat}</Text>
31  </View>
32  <View className="w-full border-b-[1px] border-dashed mt-4"></View>
33  <View className="w-full">
34  {item.jasa && (
35   <View className="mt-4">
36    <Text className="text-sm font-isemibold">{item.jasa}</Text>
37    <View className="flex flex-row justify-between items-end space-x-1">
38     <Text className="text-xs font-iregular">
39      {item.jasaqty} {item.jasa === "Koin" ? "Koin" : "Kg"} x
        @{jasaNominal}{item.jasa === "Koin" ? "/Koin" : "/Kg"}
40     </Text>
```

```
41      <View className="flex-1 border-Hitam-4 border-b-[1px] border-dashed
        mb-1"></View>
42      <Text className="text-sm font-isemibold">
43       {formatRupiah(totalJasaHarga)}
44      </Text>
45     </View>
46    </View>
47    )}
48   {item.pakaian.map((pakaianItem,index) => (
49    <View key={index} className="mt-2">
50     <Text className="text-sm font-isemibold">
51      {pakaianItem.keterangan}
52     </Text>
53     <View className="flex flex-row justify-between items-end space-x-1">
54      <Text className="text-xs font-iregular">
55       {item.pakaianqty[index]} Qty x @{item.pakaianNominal[index]}
56      </Text>
57      <View className="flex-1 border-Hitam-4 border-b-[1px] border-dashed
          mb-1"></View>
58       <Text className="text-sm font-isemibold">
59        {formatRupiah(Number.parseInt(item.pakaianNominal[index])
          *Number.parseInt(item.pakaianqty[index]))}
60       </Text>
61      </View>
62     </View>
63    ))}
64   </View>
65   <View className="w-full border-b-0.5 mt-4"></View>
66    <View className="flex flex-row justify-between items-center space-x-1
      mt-4 mb-8">
67     <Text className="flex-1 text-sm font-isemibold">TOTAL</Text>
68     <Text className="text-base font-isemibold">{formatRupiah(item.amount)}</Text>
71    </View>
72    <Text className="text-[10px]">Created with 💚 by LaPlus</Text>
73    </View>
74   </View>
75   )
76 }
77 return <Text>Empty </Text>
78 }
```

Gambar 4.22 Kode untuk komponen detail *card*

4.3.4 Halaman tambah invoice

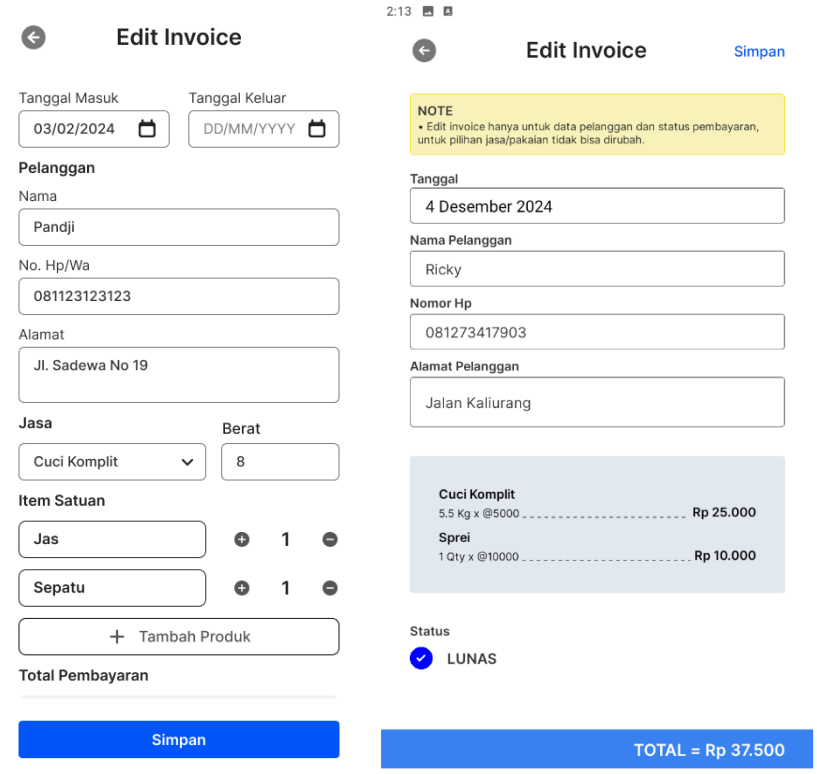


(a) Desain purwarupa (b) Desain final

Gambar 4.23 Perbandingan (a)desain purwarupa dan (b)desain final halaman tambah invoice

Perubahan yang terjadi pada halaman tambah *invoice* tidak begitu signifikan, hanya mencakup pengurangan dan pembaharuan desain agar lebih selaras dengan tema aplikasi. Salah satu perubahan utama adalah dihilangkannya *input* untuk tanggal keluar sehingga hanya terdapat satu jenis tanggal pada halaman ini. Keputusan ini diambil karena adanya kendala teknis dalam mengimplementasikan dua jenis tanggal yang terkait, serta untuk menyederhanakan pengelolaan data transaksi. Perubahan tersebut tidak mempengaruhi fungsionalitas, karena halaman ini tetap berfungsi sebagai pencatatan transaksi *laundry*, memungkinkan pengguna untuk memasukan informasi pelanggan, jenis layanan, serta rincian pembayaran ke dalam sistem. Hasil tampilan akhir untuk halaman tambah *invoice* dapat dilihat pada Gambar 4.23 bagian (b), sedangkan tampilan sebelumnya disajikan pada bagian (a).

4.3.5 Halaman edit invoice

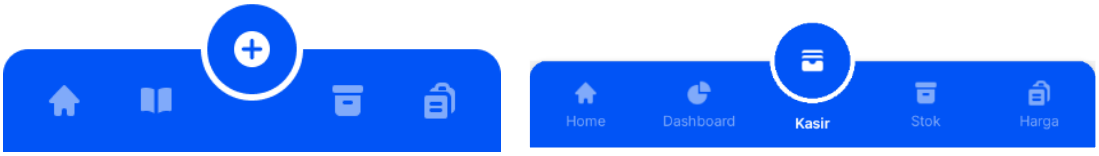


(a) Desain purwarupa (b) Desain final

Gambar 4.24 Perbandingan (a)desain purwarupa dan (b)desain final halaman *edit invoice*

Perbandingan antara tampilan akhir dan tampilan sebelumnya dari halaman *edit invoice* ditunjukkan pada Gambar 4.24, pada halaman ini terdapat pembatasan terhadap perubahan data yang ada. Jika pada desain purwarupa sebelumnya pengguna dapat merubah seluruh data *invoice*, termasuk jenis jasa dan pakaian, serta total tagihan maka pada implementasi akhirnya hal tersebut dibatasi. Pembatasan ini disesuaikan dengan kendala teknis pada basis data yang harus menyimpan informasi riwayat harga jenis pakaian atau jasa. Informasi atau data yang bisa diubah pada halaman ini meliputi tanggal, data pelanggan seperti nama, nomor dan alamat, serta status pembayaran *invoice* tersebut.

4.3.6 Tab menu



(a) Desain purwarupa (b) Desain final

Gambar 4.25 Perbandingan (a)desain purwarupa dan (b)desain final pada *tab menu*

Hal yang menjadi masalah pada saat melakukan pengujian purwarupa adalah tidak adanya label yang menandakan suatu halaman pada *tab* menu. Sehingga dilakukan penambahan label dan penyesuaian ikon guna mempermudah pengguna dalam mengidentifikasi halaman tersebut. Hasil perubahan yang ada pada *tab* menu disajikan dalam Gambar 4.25 bagian (b) dan tampilan sebelumnya pada bagian (a).

4.4 Hasil pengujian aplikasi

Setelah aplikasi dinyatakan selesai dalam masa implementasi, tugas selanjutnya adalah melakukan pengujian pada aplikasi tersebut. Pengujian yang dilakukan merupakan pengujian *blackbox* dengan *manual testing*, pengujian dilakukan untuk mengetahuai apakah fitur-fitur dalam aplikasi berjalan sesuai dengan yang diharapkan. Untuk hasil pengujiannya fitur-fitur tersebut dikelompokan sesuai dengan bagiannya masing-masing seperti bagian autentikasi, *dashboard*, *invoice*, pengeluaran dan kelola harga. Rincian hasil pengujian pada aplikasi dijelaskan sebagai berikut:

4.4.1 Bagian autentikasi

Tabel 4.3 Hasil pengujian pada fungsi login

Test Scenario :	Fungsi login	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan identitas yang benar	Sistem akan melanjutkan ke halaman beranda	Sistem melanjutkan ke halaman beranda	Pass

Memasukan identitas yang salah	Sistem akan menampilkan alert error dengan keterangan invalid credential	Sistem menampilkan alert error	Pass

Tabel 4.4 Hasil pengujian pada fungsi register

Test Scenario :	Fungsi register	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan semua identitas yang benar	Sistem akan melanjutkan ke halaman beranda	Sistem melanjutkan ke halaman beranda	Pass
Memasukan identitas yang tidak sesuai format	Sistem akan menampilkan alert error dengan keterangan format tidak sesuai	Sistem menampilkan alert error	Pass
Tidak memasukan identitas yang diminta	Sistem akan menampilkan alert dengan keterangan “Tolong masukan semua data”	Sistem menampilkan alert error dengan keterangan yang sama	Pass

Tabel 4.5 Hasil pengujian pada fungsi logout

Test Scenario :	Fungsi logout	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**

Menekan tombol logout	Sistem akan menampilkan halaman onboarding	Sistem melanjutkan ke halaman onboarding	Pass

Tabel 4.6 Hasil pengujian pada fungsi edit profile

Test Scenario :	Fungsi edit profile	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan semua identitas yang benar	Sistem akan melanjutkan ke halaman beranda	Sistem melanjutkan ke halaman beranda	Pass
Memasukan identitas yang tidak sesuai format	Sistem akan menampilkan alert error dengan keterangan format tidak sesuai	Sistem menampilkan alert error	Pass
Tidak memasukan identitas yang diminta	Sistem akan menampilkan alert dengan keterangan "Tolong masukan semua data"	Sistem menampilkan alert error dengan keterangan yang sama	Pass

Berdasarkan hasil pengujian, seluruh fungsi tersebut berhasil dijalankan dengan benar. Hasil pengujian dari setiap fungsi dijelaskan sebagai berikut:

1. *Login*: berhasil mengautentikasi pengguna dengan email dan password yang valid, serta memberikan notifikasi yang tepat untuk input yang salah.
2. *Register*: berhasil menyimpan data pengguna baru dan memberikan validasi untuk input yang tidak lengkap atau tidak sesuai format.
3. *Logout*: mengeluarkan pengguna dari sesi aktif dan mengarahkan kembali ke halaman *onboard* tanpa error.

4. *Edit profile*: berhasil memperbarui data pengguna dan menyimpan perubahan secara real-time ke basis data.

Secara keseluruhan, hasil pengujian menunjukkan bahwa sistem bagian autentikasi berfungsi dengan baik dan stabil, serta sudah layak digunakan dalam lingkungan operasional aplikasi. Tidak ditemukan error atau kegagalan fungsional dalam skenario pengujian yang dilakukan, sehingga fitur autentikasi dinyatakan lulus pengujian black-box.

4.4.2 Bagian dashboard

Tabel 4.7 Hasil pengujian pada fitur filter dashboard

Test Scenario :	Fungsi filter dashboard	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih periode bulan	Sistem akan menampilkan data sesuai dengan bulan yang dipilih	Sistem menampilkan data yang sesuai	Pass

Pengujian black-box pada halaman dashboard dilakukan hanya pada fitur *filter* data berdasarkan bulan, yang berfungsi untuk menampilkan data keuangan (pemasukan dan pengeluaran) sesuai dengan periode waktu yang dipilih oleh pengguna. Hasil pengujian menunjukkan bahwa fungsi filter berhasil dijalankan dengan benar. Ketika pengguna memilih bulan tertentu, sistem secara otomatis memperbarui tampilan total pemasukan, pengeluaran, dan informasi keuangan lainnya sesuai dengan data yang tersedia untuk bulan tersebut.

4.4.3 Bagian invoice

Test Scenario :	Fungsi pencarian invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**

Action	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
Memasukan kata kunci sesuai nama/nomor pengguna	Sistem akan menampilkan invoice sesuai kata kunci yang dimasukan	Sistem menampilkan invoice sesuai kata kunci yang dimasukan	Pass
Memasukan kata kunci tidak sesuai nama/ nomor pengguna	Sistem akan menampilkan keterangan bahwa tidak ada invoice	Sistem menampilkan keterangan bahwa tidak ada invoice	Pass

Test Scenario :	Fungsi filter invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih kategori “Semua”	Sistem akan menampilkan seluruh data invoice	Sistem menampilkan seluruh data invoice	Pass
Memilih kategori “Lunas”	Sistem akan menampilkan data invoice yang sudah lunas	Sistem menampilkan data invoice yang sudah lunas	Pass
Memilih kategori “Belum bayar”	Sistem akan menampilkan data invoice yang belum bayar	Sistem menampilkan data invoice yang belum bayar	Pass

Test Scenario :	Fungsi tambah invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**

Action	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa invoice berhasil dibuat	Sistem menampilkan invoice berhasil dibuat	Pass
Tidak memasukan informasi yang diminta	Sistem akan menampilkan bahwa semua data harus di isi	Sistem menampilkan bahwa semua data harus di isi	Pass

Test Scenario :	Fungsi edit invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa invoice berhasil diperbaharui	Sistem menampilkan invoice berhasil diperbaharui	Pass

Test Scenario :	Fungsi hapus invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Menekan tombol hapus	Sistem akan menampilkan modal peringatan	Sistem menampilkan modal peringatan	Pass
Memilih pilihan iya pada modal	Sistem akan menampilkan pemberitahuan	Sistem menampilkan pemberitahuan invoice berhasil dihapus	Pass

	invoice berhasil dihapus		

Test Scenario :	Fungsi share invoice	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih tombol share pada halaman detail invoice	Sistem akan menampilkan fitur berbagi dari smartphone pengguna	Sistem menampilkan fitur berbagi dari smartphone pengguna	Pass

Test Scenario :	Fungsi print invoice	**TestCase (Pass/Fail/Not executed)**	**Fail**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih tombol print	Sistem akan menampilkan fitur berbagi dari smartphone pengguna dan mencetak invoice	Sistem menampilkan fitur print tetapi tidak bisa mencetak invoice	Fail

Hasil pengujian pada bagian *invoice* menunjukkan bahwa enam fungsi berhasil berjalan dengan baik, sedangkan satu fungsi yaitu *print invoice* belum dapat berjalan secara optimal. Berikut adalah ringkasan hasil pengujian:

1. Pencarian *invoice*: sistem berhasil menampilkan hasil pencarian berdasarkan nama pelanggan atau nomor *invoice* secara akurat.
2. *Filter invoice*: filter berdasarkan status pembayaran berfungsi dengan baik dan menampilkan data yang sesuai.

3. Tambah *invoice*: *invoice* baru dapat ditambahkan dengan benar dan langsung muncul dalam daftar.
4. *Edit invoice*: data *invoice* dapat disunting dan diperbarui sesuai *input* pengguna.
5. Hapus *invoice*: data berhasil dihapus dari sistem dan tidak lagi muncul pada daftar *invoice*.
6. *Share invoice*: i*nvoice* dapat dibagikan melalui platform eksternal seperti WhatsApp.
7. *Print invoice*: Fungsi ini belum berjalan optimal, terutama dalam hal konektivitas dengan printer Bluetooth. Halaman cetak memang dapat dibuka, namun proses pencetakan tidak dapat dilanjutkan secara langsung dari aplikasi.

Untuk mengatasi kegagalan pada fitur *print invoice*, disarankan agar menggunakan aplikasi aplikasi pihak ketiga seperti RawBT. RawBT memungkinkan pengguna mencetak dokumen melalui koneksi *bluetooth* dengan lebih mudah dan kompatibel dengan banyak jenis printer yang umum digunakan pelaku usaha kecil.

4.4.4 Bagian pengeluaran

Test Scenario :	Fungsi filter pengeluaran	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih kategori “Semua”	Sistem akan menampilkan seluruh data pengeluaran	Sistem menampilkan seluruh data pengeluaran	Pass
Memilih kategori “Stok”	Sistem akan menampilkan data stok pengeluaran	Sistem akan menampilkan data stok pengeluaran	Pass
Memilih kategori “Expenses”	Sistem akan menampilkan data Expenses pengeluaran	Sistem akan menampilkan data Expenses pengeluaran	Pass

Test Scenario :	Fungsi tambah pengeluaran	**TestCase (Pass/Fail/Not executed)**	**Pass**

Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa pengeluaran berhasil dibuat	Sistem menampilkan pengeluaran berhasil dibuat	Pass
Tidak memasukan informasi yang diminta	Sistem akan menampilkan bahwa semua data harus di isi	Sistem menampilkan bahwa semua data harus diisi	Pass

Test Scenario :	Fungsi edit pengeluaran	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa pengeluaran berhasil diperbaharui	Sistem menampilkan pengeluaran berhasil diperbaharui	Pass

Test Scenario :	Fungsi hapus pengeluaran	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Menekan tombol hapus	Sistem akan menampilkan modal peringatan	Sistem menampilkan modal peringatan	Pass

Memilih pilihan iya pada modal	Sistem akan menampilkan pemberitahuan pengeluaran berhasil dihapus	Sistem menampilkan pemberitahuan pengeluaran berhasil dihapus	Pass

4.4.5 Bagian harga

Test Scenario :	Fungsi edit harga jasa	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memilih jenis jasa	Sistem akan menampilkan input untuk melakukan perubahan	Sistem menampilkan input untuk melakukan perubahan	Pass
Memasukan data yang diubah	Sistem akan menampilkan alert harga berhasil diperbaharui	Sistem menampilkan alert harga berhasil diperbaharui	Pass

Test Scenario :	Fungsi hapus jasa	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Menekan tombol hapus	Sistem akan menampilkan modal peringatan	Sistem menampilkan modal peringatan	Pass
Memilih pilihan iya pada modal	Sistem akan menampilkan pemberitahuan Jasa berhasil dihapus	Sistem menampilkan pemberitahuan Jasa berhasil dihapus	Pass

Test Scenario :	Fungsi tambah jenis pakaian	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa pakaian berhasil dibuat	Sistem menampilkan pakaian berhasil dibuat	Pass

Test Scenario :	Fungsi edit jenis pakaian	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Memasukan informasi sesuai dengan format	Sistem akan menampilkan bahwa pakaian berhasil diperbaharui	Sistem menampilkan pakaian berhasil diperbaharui	Pass

Test Scenario :	Fungsi hapus jenis pakaian	**TestCase (Pass/Fail/Not executed)**	**Pass**
Action	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
Menekan tombol hapus	Sistem akan menampilkan modal peringatan	Sistem menampilkan modal peringatan	Pass
Memilih pilihan iya pada modal	Sistem akan menampilkan pemberitahuan	Sistem menampilkan pemberitahuan pakaian berhasil dihapus	Pass

	pakaian berhasil dihapus		

Berdasarkan pengujian yang dilakukan pada bagian harga, terdapat lima fungsi yang di uji antara lain: edit harga jasa, hapus jasa, tambah, edit, dan hapus jenis pakaian. Untuk hasil pengujian dari setiap fungsi dijelaskan sebagai berikut

1. *Edit* harga jasa: sistem berhasil menyimpan perubahan harga jasa yang dilakukan pengguna dan menampilkannya kembali dengan nilai yang diperbarui.
2. Hapus jasa: Pengguna dapat menghapus data jasa yang tidak lagi digunakan, dan data tersebut tidak lagi muncul dalam daftar jasa.
3. Tambah jenis pakaian: *Form input* dapat digunakan untuk menambahkan harga jenis pakaian baru. Data yang ditambahkan langsung tercatat dan muncul dalam daftar jenis pakaian.
4. *Edit* jenis pakaian: Perubahan data pakaian berhasil dilakukan dan data yang diperbarui tersimpan dengan baik serta ditampilkan secara akurat.
5. Hapus jenis pakaian: Fungsi penghapusan bekerja sebagaimana mestinya, di mana pakaian yang dihapus tidak lagi tersedia dalam sistem.

Secara keseluruhan, seluruh fungsi pada fitur harga berhasil dijalankan tanpa ditemukan bug atau error, sehingga fitur ini dinyatakan stabil dan layak digunakan dalam lingkungan operasional aplikasi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari hasil penelitian yang telah dilakukan mengenai pengembangan aplikasi mobile menggunakan metode UCD untuk membantu operasional usaha laundry. Kesimpulan disusun berdasarkan analisis terhadap rumusan masalah, hasil implementasi aplikasi, serta pengujian yang telah dilakukan selama proses pengembangan. Berikut merupakan kesimpulan yang dapat diambil dari penelitian ini:

1. **Bagaimana merancang dan membangun aplikasi mobile sebagai sistem kasir digital untuk membantu pelaku usaha laundry dalam menjalankan operasional bisnisnya?,** aplikasi mobile berhasil dikembangkan menggunakan React Native dengan bantuan Expo dan berfungsi sebagai sistem kasir digital yang memudahkan pelaku usaha laundry dalam mencatat dan memantau transaksi keuangan harian secara efisien. Aplikasi ini memiliki fitur pencatatan pemasukan dan pengeluaran, riwayat transaksi, serta visualisasi data keuangan dalam bentuk grafik. Hasil dari pengujian pertama pada purwarupa dengan pengujian presentase keberhasilan mencapai 84%, dan hasil *System Usability Scale* (SUS) menunjukkan skor sebesar 75.5, yang menunjukkan aplikasi termasuk dalam kategori baik atau *acceptable*. Pengujiannya selanjutnya dilakukan pada produk jadi dengan menggunakan metode *black-box testing* yang menunjukkan hampir seluruh fungsi berjalan sesuai harapan.
2. **Bagaimana pemanfaatan data keuangan pada aplikasi tersebut dapat mendukung pelaku usaha laundry dalam menjalankan operasional bisnis secara lebih efisien?,** data keuangan yang tercatat dalam aplikasi dapat digunakan oleh pelaku usaha untuk mendukung pengambilan keputusan dalam operasional bisnis. Melalui tampilan grafik keuangan yang ada, pelaku usaha dapat dengan mudah memahami kondisi keuangan harian hingga bulanan, serta mengamati tren pemasukan dan pengeluaran. Dengan demikian, aplikasi ini tidak hanya menjadi alat pencatatan, tetapi juga sarana analisis sederhana untuk meningkatkan efektivitas pengelolaan usaha.

5.2 Saran

Berdasarkan hasil pengujian dan implementasi aplikasi, terdapat beberapa hal yang dapat dijadikan pertimbangan untuk pengembangan di masa mendatang agar aplikasi semakin optimal dan fungsional bagi pelaku usaha *laundry*, antara lain:

1. Perbaikan fitur print

   Fitur print saat ini belum mendukung untuk melakukan pencetakan langsung pada aplikasi sehingga masih membutuhkan aplikasi pihak ketiga, disarankan kedepannya pembuatan fitur print ini dapat digunakan tanpa membutuh aplikasi pihak ketiga lagi.

2. Manajemen pelanggan

   Saat ini dalam aplikasi data pelanggan masih masuk kedalam data *invoice* saja tidak ada model khusus yang hanya menyimpan data tersebut. Pengembangan fitur manajemen pelanggan seperti pencatatan nama, nomor dan riwayat transaksi dapat meningkatkan efisiensi serta kualitas layanan kepada pelanggan.

3. Sistem notifikasi pelanggan

   Fitur yang ada pada aplikasi hanya bisa mengirimkan invoice kepada pelanggan lewat whatsapp ataupun email. Untuk kedepannya disarankan agar aplikasi bisa mengirim pesan notifikasi melalui whatsapp untuk memberikan invoice dan keterangan mengenai kapan proses pencucian selesai secara otomatis setelah invoice dibuat.

4. Penerapan multi-user dan hak akses berbeda

   Aplikasi dapat diperluas agar mendukung sistem multi-user dengan peran dan hak akses yang berbeda, seperti admin (pemilik usaha) dan kasir (karyawan), untuk meningkatkan keamanan data dan efisiensi operasional.

DAFTAR PUSTAKA

Agus Muhyidin, M., Sulhan, M. A., & Sevtiana, A. (2020). Perancangan UI/UX Aplikasi My CIC Layanan Informasi Akademik Mahasiswa Menggunakan Aplikasi Figma. *Jurnal Digit: Digital of Information Technology*, *10*(2), 208–219. https://doi.org/10.51920/jd.v10i2.171

Android Studio. (2024). *Meet Android Studio*. https://developer.android.com/studio/intro

Annisa, R., Agung Ananda, R., & Sulistiono, W. E. (2024). Implementasi Golang Clean Architecture pada Perancangan Backend Point of Sales Website. *Jurnal Informatika Dan Teknik Elektro Terapan*, *12*(2), 1518–1523. https://doi.org/10.23960/jitet.v12i2.4668

Bergman, C. (2024). *Design systems 101: What is a design system?* https://www.figma.com/blog/design-systems-101-what-is-a-design-system/

Bluestack. (2024). *System requirements for BlueStacks 5*. https://support.bluestacks.com/hc/en-us/articles/360056129211-System-requirements-for-BlueStacks-5

Chen, S., Thaduri, U. R., & Ballamudi, V. K. R. (2019). Front-End Development in React: An Overview. *Engineering International*, *7*(2), 117–126. https://doi.org/10.18034/ei.v7i2.662

Debon, R., Coleone, J. D., Bellei, E. A., & De Marchi, A. C. B. (2019). Mobile health applications for chronic diseases: A systematic review of features for lifestyle improvement. In *Diabetes and Metabolic Syndrome: Clinical Research and Reviews* (Vol. 13, Issue 4, pp. 2507–2512). Elsevier Ltd. https://doi.org/10.1016/j.dsx.2019.07.016

Dziedzic, S., Kurak, K., Samp, J., & Szeremeta, P. (2024). *Environment variables in EAS: new changes to simplify setup*. https://expo.dev/blog/environment-variables

Fadli, B., Ramadlan, N., Wulandari, S., Hajar, R. R., Sejati, P., & Suhendar, A. (2024). Penerapan Metode UCD (User Centered Design) Pada Sistem Perpustakaan Sekolah Berbasis Android. *KLIK: Kajian Ilmiah Informatika Dan Komputer*, *4*(5), 2430–2441. https://doi.org/10.30865/klik.v4i5.1803

Hadi, H. S., & Dwi Gustina, E. (2024). Web-based Information System for the Recapitulation of the Election of the Chairman of RW011 Padang Sarai Permai Housing. *Jurnal Manajemen Teknologi Informatika*, *2*(1), 219–230. https://doi.org/10.70038/jentik.v2i1.87

Kamar Dagang dan Industri Indonesia. (2023). *Data dan Statistik UMKM Indonesia*. https://kadin.id/data-dan-statistik/umkm-indonesia/

Kurnia, J., & Awaludin, M. (2023). Penerapan Metode UCD (user centered design) Sistem Informasi Penggajian Karyawan Berbasis Web pada Koperasi Karyawan Air Timur Jakarta (Kopkar-Atj). *Jurnal Sistem Informasi Universitas Suryadarma*, *10*(2), 131–138. https://doi.org/10.35968/jsi.v10i2.1082

Maurits Ivan. (2020). Analisis dan Implementasi Aplikasi Pembukuan Berbasis Android Untuk Memenuhi Kebutuhan Pada Usaha Kecil Menengah. *UG Jurnal*, *14*(11), 21–32.

Oberai, A. (2024). *mySHOEFITTER: Solving 75% of online shoe order returns*. https://appwrite.io/blog/post/case-study-myshoefitter

Petrenko, C. L. M. G., Kautz-Turnbull, C. C., Roth, A. R., Parr, J. E., Tapparello, C., Demir, U., & Olson, H. C. (2021). Initial feasibility of the "Families moving forward connect" mobile health intervention for caregivers of children with fetal alcohol spectrum disorders: Mixed method evaluation within a systematic user-centered design approach. *JMIR Formative Research*, *5*(12). https://doi.org/10.2196/29687

Pranoto, S., Sutiono, S., & Nasution, D. (2024). Penerapan UML Dalam Perancangan Sistem Informasi Pelaporan Dan Evaluasi Pembangunan Pada Bagian Administrasi Pembangunan Sekretariat Daerah Kota Tebing Tinggi. *Surplus: Jurnal Ekonomi Dan Bisnis*, *2*(2), 384–401.

Raihan, M. R., & Hidayahtullah, D. (2022). Pengembangan Sistem Point Of Sale Berbasis User Centered Design. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *6*(1), 74. https://doi.org/10.30865/mib.v6i1.3412

Susilo, E. (2019). *Cara Menggunakan System Usability Scale (SUS) Pada Evaluasi Usability*. https://www.edisusilo.com/cara-menggunakan-system-usability-scale/

Undang-Undang Republik Indonesia. (2008). *Undang-Undang Republik Indonesia tentang Usaha Kecil, Mikro dan Menengah (UU Nomor 20 Tahun 2008)*.

Vailshery, L. S. (2024). *Most used web frameworks among developers worldwide, as of 2024*. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

LAMPIRAN