BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini menghasilkan sebuah aplikasi *mobile* berbasis Flutter dengan *backend* Supabase yang dirancang untuk mendukung operasional Laundry LB Fresh di Bantul. Sistem dikembangkan melalui pendekatan rancang bangun yang mencakup tahapan identifikasi permasalahan, analisis kebutuhan, perancangan sistem, implementasi, serta pengujian awal. Permasalahan yang ditemukan di lapangan, seperti pencatatan transaksi secara manual yang rawan kesalahan, sulitnya pelacakan status *laundry*, dan belum tersedianya laporan keuangan otomatis, berhasil diatasi melalui sistem aplikasi yang terintegrasi.

Antarmuka pengguna dirancang responsif dengan mengacu pada prinsip Material Design dan dapat diakses berdasarkan peran pengguna seperti *Super Admin*, *Admin*, dan *User*. Struktur basis data menggunakan PostgreSQL melalui layanan Supabase, mencakup entitas utama seperti pengguna, pelanggan, layanan, transaksi, dan laporan, serta dilengkapi fitur *role-based access control*, *soft delete*, dan pencatatan waktu *(timestamping)* untuk mendukung histori dan audit sistem. Pengujian dilakukan secara internal menggunakan data simulasi dari Laundry LB Fresh, yang menunjukkan peningkatan efisiensi pencatatan transaksi dan kemudahan pelacakan data. Selanjutnya, pengujian juga dilakukan oleh pemilik bisnis dan staf secara langsung.

Secara keseluruhan, sistem yang dikembangkan dinilai telah memenuhi tujuan penelitian, yaitu menyediakan solusi digital yang mampu meningkatkan efisiensi, akurasi, dan kemudahan dalam pengelolaan bisnis *laundry* pada skala UMKM.

4.2 IMPLEMENTASI DESAIN ANTARMUKA

Antarmuka pengguna (*user interface*) dalam aplikasi pengelolaan bisnis *laundry* ini dirancang dengan prinsip Material Design untuk memberikan pengalaman penggunaan yang konsisten, bersih, dan intuitif. Setiap tampilan

menyesuaikan dengan peran pengguna seperti *Super Admin*, *Admin*, maupun *User* biasa, dan dapat diakses melalui perangkat Android secara responsif.

4.2.1 Implementasi Halaman *Splash*

Halaman *splash* merupakan tampilan awal yang muncul saat aplikasi pertama kali dijalankan. Layar ini berfungsi sebagai pengenal identitas aplikasi, dengan menampilkan logo secara terpusat, nama, serta kata-kata singkat untuk menciptakan kesan pertama yang profesional.

Gambar 4.1 Halaman *Splash*

Sebagaimana ditunjukkan pada Gambar 4.1, desain *splash screen* dibuat sederhana namun tetap elegan. Tampilan ini muncul selama kurang lebih 2 detik sebelum pengguna diarahkan ke halaman selanjutnya. Apabila pengguna belum melakukan *login*, sistem akan secara otomatis mengarahkan ke *login screen*.

Kemudian, jika sesi *login* masih aktif, pengguna akan langsung diarahkan ke *dashboard screen*.

4.2.2 Implementasi Halaman Masuk

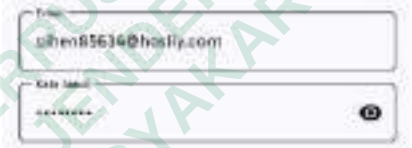
Halaman masuk merupakan gerbang utama bagi pengguna untuk mengakses aplikasi. Desainnya dibuat simpel dengan fokus pada kemudahan dan kejelasan memasukkan data. Halaman ini memungkinkan pengguna bisa masuk aplikasi menggunakan *email* dan kata sandi yang telah terdaftar di sistem.

Gambar 4.2 Halaman Masuk

Sebagaimana ditampilkan pada Gambar 4.2, halaman masuk terdiri atas dua kolom utama, yaitu *email* dan kata sandi, serta satu tombol “Masuk” dan satu tautan teks menuju halaman daftar. Sistem akan mendeteksi dan memvalidasi masukan pengguna ketika tombol masuk diklik. Jika format *email* tidak valid, sistem akan menampilkan peringatan pada kolom masukan.

Gambar 4.3 *Email Invalid*

Pada Gambar 4.3 ditunjukkan bagaimana sistem memberikan indikator kesalahan ketika format *email* yang dimasukkan tidak valid. Selain itu, pengguna dapat mengatur keterbacaan kata sandi melalui ikon tampilkan/sembunyikan kata sandi yang terletak di dalam kolom kata sandi. Fitur ini memudahkan pengguna dalam memastikan kata sandi sesuai sebelum dikirim.



Gambar 4.4 Kata Sandi Disembunyikan

Gambar 4.5 Kata Sandi Ditampilkan

Seperti yang tampak pada Gambar 4.4 dan Gambar 4.5, sistem memberikan kendali penuh kepada pengguna untuk melihat atau menyembunyikan karakter kata sandi dengan satu ketukan.

Jika pengguna menekan tombol masuk dengan data yang tidak sesuai, sistem akan menampilkan notifikasi kesalahan melalui notifikasi di bagian bawah layar.

Gambar 4.6 *Login Failed*

Gambar 4.6 memperlihatkan bagaimana sistem memberikan umpan balik secara visual ketika proses masuk gagal, misalnya karena kombinasi *email* dan kata sandi tidak cocok dengan yang ada di sistem. Selain itu, terdapat tombol teks di bagian bawah tombol masuk yang dapat digunakan untuk berpindah ke halaman daftar bagi pengguna baru yang belum memiliki akun.

Gambar 4.7 Kolom Tidak Diisi

Gambar 4.7 menunjukkan tampilan kesalahan ketika pengguna menekan tombol masuk tanpa mengisi semua kolom yang diwajibkan.

4.2.3 Implementasi Halaman Daftar

Halaman daftar disediakan bagi pengguna baru yang belum memiliki akun pada sistem. Halaman ini dirancang sederhana dan mudah dipahami seperti yang ada di halaman masuk, dengan fokus pada masukan data yang penting untuk proses autentikasi awal dan identitas pengguna.

Gambar 4.8 Halaman Daftar

Sebagaimana terlihat pada Gambar 4.8, halaman daftar terdiri atas empat kolom masukan, yaitu:

1. Nama
2. Email
3. Kata sandi
4. Konfirmasi kata sandi

Tombol “Daftar” berada di bagian bawah kolom, dan sistem akan memvalidasi semua masukan saat tombol tersebut ditekan. Jika terdapat masukan yang kosong atau tidak valid, sistem akan memberikan umpan balik berupa pesan kesalahan pada kolom terkait seperti yang terlihat pada Gambar 4.9 berikut.

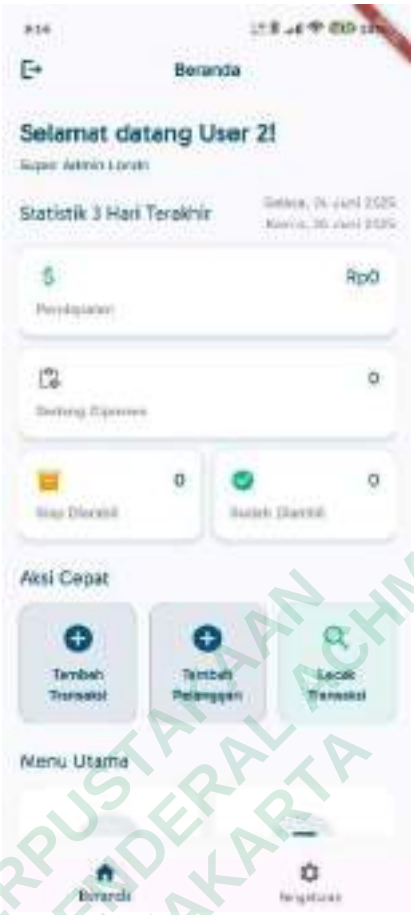
Gambar 4.9 Kolom Tidak Diisi

Validasi dan penanganan kesalahan masukan pada *register screen* mengikuti pola yang sama seperti yang telah dijelaskan pada *login screen*. Misalnya, sistem akan menampilkan peringatan apabila format *email* tidak valid, atau ketika kata sandi dan konfirmasi tidak cocok.

Halaman ini juga terhubung dengan *login screen* melalui tautan teks di bagian bawah, sehingga pengguna dapat dengan mudah berpindah antar halaman sesuai kebutuhannya.

4.2.4 Implementasi Halaman *Dashboard*

Halaman *dashboard* merupakan pusat utama kontrol dalam aplikasi yang ditampilkan setelah pengguna berhasil *login*. Tampilan ini berfungsi sebagai ringkasan informasi penting dan akses cepat ke fitur-fitur utama, yang disesuaikan berdasarkan peran pengguna (*Super Admin*, *Admin*, atau *User* biasa).



Gambar 4.10 Halaman *Dashboard* Bagian Atas

Pada halaman *dashboard* bagian atas, seperti ditampilkan pada Gambar 4.10, terdapat beberapa elemen utama:

1. *Header* ucapan selamat datang, yang menampilkan nama pengguna dan peran pengguna.
2. Statistik 3 harian, yang menyajikan informasi berikut:
    a. Total pendapatan
    b. Jumlah transaksi yang sedang diproses (*On Progress*)
    c. Transaksi yang siap diambil (*Ready for Pickup*)
    d. Transaksi yang telah diambil (*Picked Up*)

Statistik ini diperbarui secara *real-time* dan membantu pengguna untuk melihat performa layanan dalam periode tiga hari terakhir secara ringkas.

Selain itu, di bagian kiri atas halaman ini terdapat tombol *logout* yang ditampilkan dalam bentuk ikon *logout*. Tombol ini memungkinkan pengguna untuk keluar dari akun yang telah *login*.

Gambar 4.11 Halaman *Dashboard* Bagian Bawah

Gambar 4.11 menampilkan halaman *dashboard* bagian bawah, yang terdiri atas:

1. Aksi Cepat (*Quick Action*) berupa tombol:
    a. Tambah Transaksi
    b. Tambah Pelanggan
    c. Lacak Transaksi untuk semua *role*
2. Menu Utama, yang memberikan navigasi ke fitur-fitur inti, meliputi:
    a. Kelola Pegawai (khusus *Super Admin*)
    b. Kelola Layanan (khusus *Super Admin*)
    c. Kelola Transaksi

d. Ekspor Laporan

Pada bagian bawah juga ada *navigation bar* untuk mengakses halaman *dashboard* dan halaman pengaturan. Dengan komposisi ini, pengguna dapat langsung mengakses fungsi operasional sehari-hari dari satu halaman secara efisien.

Proses *logout* dimulai saat ikon *logout* di kiri atas ditekan, sistem akan menampilkan dialog konfirmasi untuk memastikan bahwa pengguna benar-benar ingin *logout* dari aplikasi.

Gambar 4.12 Dialog Konfirmasi *Logout*

Gambar 4.12 memperlihatkan tampilan dialog yang menanyakan konfirmasi sebelum pengguna *logout*. Hal ini untuk menghindari *logout* yang tidak disengaja. Setelah pengguna mengonfirmasi, sistem akan menghapus sesi pengguna dan menampilkan notifikasi bahwa *logout* berhasil dilakukan.

Gambar 4.13 *Success Logout*

Sebagaimana tampak pada Gambar 4.13, aplikasi memberikan umpan balik visual bahwa proses *logout* telah berhasil, kemudian akan mengarahkan kembali ke *login screen*.

4.2.5 Implementasi Halaman Pengaturan

Halaman pengaturan memungkinkan pengguna menyesuaikan preferensi tampilan dari aplikasi, khususnya dalam pemilihan bahasa. Tampilan halaman ini dibuat sederhana dan efisien, dengan satu komponen utama agar pengguna tidak merasa kebingungan oleh banyak menu.

Gambar 4.14 Halaman Pengaturan

Halaman pengaturan hanya berisi satu menu saja, sebagaimana terlihat pada Gambar 4.14, halaman ini menampilkan pilihan bahasa yang saat ini sedang aktif. Ketika pengguna mengetuk pilihan tersebut, sistem akan menampilkan *bottom sheet* yang berisi daftar bahasa yang tersedia.

Gambar 4.15 Tampilan *Bottom Sheet* Pemilihan Bahasa

Seperti ditunjukkan pada Gambar 4.15, *bottom sheet* memberikan dua pilihan, yaitu Bahasa Indonesia dan Bahasa Inggris. Saat pengguna memilih salah satu bahasa, aplikasi akan langsung memperbarui seluruh tampilan UI menggunakan bahasa tersebut tanpa perlu keluar dari aplikasi. Setelah pemilihan bahasa berhasil diproses, sistem memberikan umpan balik kepada pengguna dalam bentuk notifikasi di bagian bawah layar.

Gambar 4.16 Notifikasi Berhasil Mengubah Bahasa

Gambar 4.16 memperlihatkan bahwa sistem menampilkan notifikasi berisi pesan konfirmasi bahwa bahasa telah berhasil diubah. Selain pengaturan bahasa,

ikon *logout* juga tersedia di kiri atas halaman sebagai tombol cepat untuk *logout* dari sistem. Ikon ini konsisten dengan yang ditampilkan di *dashboard screen*, sehingga memudahkan pengguna untuk *logout* dari aplikasi kapan saja.

4.2.6 Implementasi Halaman Manajemen Staf (Khusus *Super Admin*)

Halaman Manajemen Staf disediakan secara khusus untuk pengguna dengan peran *Super Admin*. Halaman ini memungkinkan *Super Admin* untuk melihat daftar pengguna sistem, memfilter data berdasarkan *role*, serta mengaktifkan atau menonaktifkan akun secara langsung melalui interaksi berbasis tekan dan tahan.

Gambar 4.17 Halaman Manajemen Staf

Sebagaimana ditampilkan pada Gambar 4.17, halaman ini menampilkan daftar seluruh pengguna yang terdaftar dalam sistem, termasuk nama dan *email* masing-masing pengguna. Komponen utama halaman ini antara lain:

1. Kolom pencarian (*search bar*) di bagian atas, digunakan untuk mencari pengguna berdasarkan nama atau *email*. Ketika pengguna mengetik sesuatu, ikon tombol *clear* akan muncul secara otomatis untuk menghapus masukan pencarian dengan cepat sebagaimana terlihat pada Gambar 4.18.

Gambar 4.18 Fungsi Pencarian Aktif

2. Tombol *sortir* yang memungkinkan *Super Admin* mengurutkan daftar berdasarkan:
    a. Nama
    b. Email
    c. Role

Pengurutan dapat dilakukan secara naik (*ascending*) maupun turun (*descending*) seperti yang terlihat pada Gambar 4.19 dan Gambar 4.20 berikut.

Gambar 4.19 Sortir Naik Berdasarkan Nama

Gambar 4.20 Sortir Turun Berdasarkan Nama

Tarik untuk memperbarui (*pull to refresh*) dapat dilakukan kapan pun untuk memperbarui data dari server secara manual. Jika hasil pencarian tidak ditemukan atau data kosong, sistem akan menampilkan tampilan *empty state* dengan pesan yang informatif dan tombol untuk mencoba memuat ulang data seperti yang ditunjukkan pada Gambar 4.21.

Gambar 4.21 Tampilan Data Kosong (*Empty State*)

Untuk mengetahui informasi singkat mengenai pengguna, *Super Admin* dapat mengetuk sekali pada salah satu item pengguna. Info akan terlihat seperti pada Gambar 4.22 berikut.

Gambar 4.22 Tampilan Info Pengguna

Sistem juga mendukung interaksi berbasis *gestur*. Jika *Super Admin* menekan dan menahan (*long press*) salah satu item pengguna, maka sistem akan menampilkan dialog untuk mengaktifkan atau menonaktifkan role *Admin* atau staf pada akun tersebut, tergantung pada status saat ini seperti yang terlihat pada Gambar 4.23 dan Gambar 4.24 berikut.

Gambar 4.23 Mengaktifkan Akun Pegawai

Gambar 4.24 Menonaktifkan Akun Pegawai

Penggunaan interaksi *long press* ini bertujuan untuk menghindari salah pencet saat ada di halaman ini, serta memberikan kontrol yang cepat dan efisien kepada *Super Admin* dalam pengelolaan akses pengguna.

4.2.7 Implementasi Halaman Manajemen Layanan

Halaman manajemen layanan juga disediakan khusus untuk pengguna dengan peran *Super Admin*. Fitur ini memungkinkan pengelolaan data layanan *laundry* seperti nama layanan, harga per kilogram, dan deskripsi. *Super Admin* dapat melakukan aksi seperti menambah, memperbarui, menghapus, mengaktifkan, maupun menonaktifkan layanan melalui tampilan yang responsif dan mudah dipahami.

Tampilan awal halaman memperlihatkan daftar layanan *laundry* secara vertikal, lengkap dengan nama dan harga. Seperti yang ditunjukkan pada Gambar 4.25 di bawah ini, setiap item layanan dapat diklik untuk melihat detail lebih lanjut.

Gambar 4.25 Halaman Manajemen Layanan

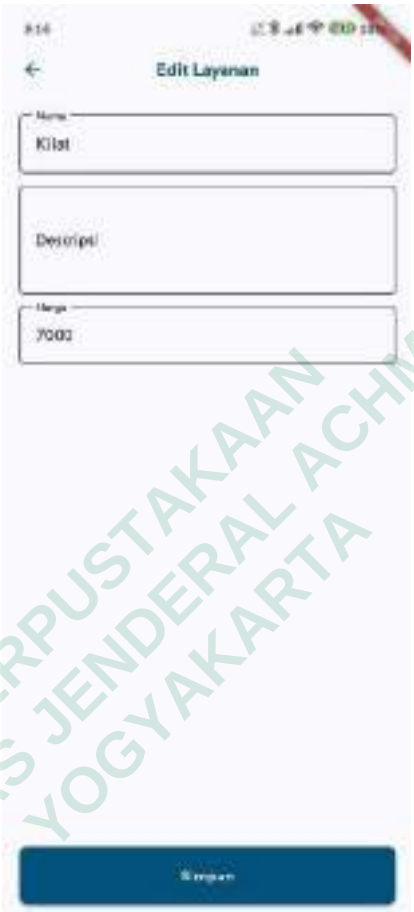
Untuk mempermudah pengelolaan, pengguna dapat mengurutkan data berdasarkan nama, harga, atau tanggal pembuatan melalui menu sortir. Menu tersebut ditampilkan pada Gambar 4.26 berikut ini.

Gambar 4.26 Menu Sortir Layanan

Jika pengguna mengetuk salah satu layanan, sistem akan menampilkan informasi detail sebagaimana terlihat pada Gambar 4.27 di bawah ini.

Gambar 4.27 Halaman Detail Layanan

Pada halaman detail ini, tersedia tiga tombol aksi yaitu ubah, nonaktifkan, dan hapus permanen layanan. Tombol ubah akan mengarahkan ke halaman edit layanan seperti pada Gambar 4.28. Apabila proses pembaruan berhasil, maka akan muncul notifikasi seperti yang terlihat pada Gambar 4.29.

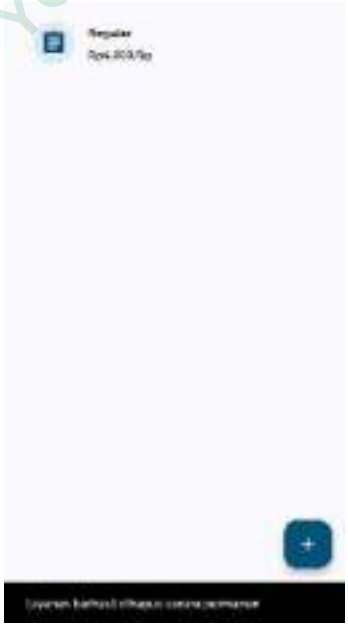


Gambar 4.28 Halaman Edit Layanan

Gambar 4.29 Berhasil Ubah Layanan

Jika pengguna memilih untuk menghapus layanan, maka sistem akan memunculkan dialog konfirmasi seperti pada Gambar 4.30, dan jika berhasil maka akan ada notifikasi seperti yang terlihat pada Gambar 4.31.

Gambar 4.30 Dialog Konfirmasi Hapus Layanan



Gambar 4.31 Berhasil Hapus Layanan

Begitu pula jika pengguna memilih menonaktifkan layanan, sistem akan memberikan konfirmasi terlebih dahulu sebagaimana ditunjukkan pada Gambar 4.32, lalu menampilkan hasilnya seperti pada Gambar 4.33.

Gambar 4.32 Dialog Konfirmasi Nonaktifkan Layanan

Gambar 4.33 Berhasil Nonaktifkan Layanan

Super Admin juga dapat menambahkan layanan baru melalui tombol tambah layanan. Halaman tambah layanan ditunjukkan pada Gambar 4.34 di bawah ini.

Gambar 4.34 Halaman Tambah Layanan

Kolom wajib yang harus diisi adalah nama layanan dan harga. Kolom deskripsi bersifat opsional. Jika kolom dikosongkan, maka sistem akan menampilkan validasi sebagaimana terlihat pada Gambar 4.35. Jika pengguna akan mengisi kolom harga maka akan tampil keyboard khusus numerik seperti yang ada pada Gambar 4.36.

Gambar 4.35 Validasi Kolom Wajib

Gambar 4.36 Input Harga Hanya Angka

Jika semua masukan sudah valid, maka sistem akan memproses penambahan layanan dan memberikan notifikasi seperti pada Gambar 4.37.

Gambar 4.37 Berhasil Tambah Layanan

Jika layanan sebelumnya dinonaktifkan, *Super Admin* dapat mengaktifkan kembali layanan tersebut dengan cara tekan dan tahan (*long press*) pada item layanan di daftar. Sistem akan memunculkan dialog konfirmasi seperti pada Gambar 4.38, dan menampilkan notifikasi keberhasilan seperti pada Gambar 4.39.

Gambar 4.38 Dialog Konfirmasi Aktifkan Layanan

Gambar 4.39 Berhasil Aktifkan Layanan

4.2.8 Implementasi Halaman Manajemen Pelanggan

Halaman Manajemen Pelanggan digunakan untuk mencatat dan mengelola data pelanggan yang menggunakan layanan *laundry*. Fitur ini dapat diakses oleh pengguna dengan peran *Admin* maupun *Super Admin*, dengan hak akses yang berbeda. *Admin* hanya dapat menonaktifkan pelanggan, sedangkan *Super Admin* memiliki akses tambahan untuk menghapus pelanggan secara permanen dari sistem.

Ketika pengguna membuka halaman ini, sistem akan menampilkan daftar pelanggan secara vertikal, lengkap dengan nama dan informasi kontak. Seperti yang ditunjukkan pada Gambar 4.41 di bawah ini, tampilan daftar disusun secara ringkas dan responsif, memudahkan pengguna dalam menjelajahi data pelanggan.

Gambar 4.40 Halaman Manajemen Pelanggan

Jika pelanggan memiliki nomor telepon yang tersimpan di sistem, maka akan muncul tombol khusus di sisi kanan item daftar yang dapat digunakan untuk

langsung menghubungi pelanggan melalui WhatsApp. Fitur ini ditampilkan lebih jelas pada Gambar 4.41, yang menunjukkan posisi tombol *redirect* di setiap baris pelanggan.

Gambar 4.41 Tombol *Redirect* WhatsApp

Pengguna juga dapat menyortir daftar pelanggan berdasarkan nama atau tanggal pembuatan akun menggunakan menu sortir seperti terlihat pada Gambar 4.42 berikut ini.

Gambar 4.42 Menu Sortir Pelanggan

Untuk menambahkan pelanggan baru, pengguna dapat mengakses halaman tambah pelanggan seperti yang diperlihatkan pada Gambar 4.43. Kolom yang wajib diisi adalah kolom nama, sedangkan kolom nomor telepon dan deskripsi bersifat opsional.

Gambar 4.43 Halaman Tambah Pelanggan

Jika kolom wajib tidak diisi, sistem akan memberikan peringatan sebagaimana ditunjukkan pada Gambar 4.44. Kemudian untuk mengisi kolom nomor telepon, keyboard yang tampil hanya keyboard numerik saja, seperti yang terlihat pada Gambar 4.45.

Gambar 4.44 Validasi Kolom Wajib

Gambar 4.45 Input Nomor Telepon Hanya Angka

Bagian jenis kelamin juga bisa diganti dengan beberapa pilihan yang ada, kemudian setelah dipilih maka tampilan jenis kelamin akan berubah. Tampilan pilihan jenis kelamin serta tampilan setelah diganti berturut-turut ditunjukkan pada Gambar 4.46, dan Gambar 4.47.

Gambar 4.46 Menu Pilihan Jenis Kelamin

Gambar 4.47 Berhasil Ubah Jenis Kelamin

Setelah kolom diisi dengan benar dan dikirim, aplikasi akan memberikan notifikasi bahwa pelanggan berhasil ditambahkan, sebagaimana terlihat pada Gambar 4.48.

Gambar 4.48 Berhasil Tambah Pelanggan

Pengguna juga dapat memperbarui data pelanggan melalui tampilan kolom yang serupa, seperti pada Gambar 4.49, dan akan memperoleh notifikasi keberhasilan yang ditunjukkan oleh Gambar 4.50.

Gambar 4.49 Halaman Edit Pelanggan

Gambar 4.50 Berhasil Ubah Pelanggan

Untuk melihat informasi lengkap mengenai pelanggan, pengguna cukup mengetuk salah satu item pada daftar. Kemudian sistem akan menampilkan detail pelanggan, sebagaimana ditunjukkan pada Gambar 4.51 dan Gambar 4.52.

Gambar 4.51 Detail Pelanggan Bagian Atas

Gambar 4.52 Detail Pelanggan Bagian Bawah

Di dalam halaman detail ini, jika pelanggan memiliki nomor telepon yang valid, maka akan ditampilkan tombol untuk *redirect* ke WhatsApp sebagaimana terlihat pada Gambar 4.53.

Gambar 4.53 Tombol *Redirect* ke WhatsApp

Tindakan terhadap data pelanggan juga dapat dilakukan dari halaman detail. *Admin* dapat menonaktifkan akun pelanggan jika diperlukan, sementara *Super Admin* memiliki opsi tambahan untuk menghapus data pelanggan secara permanen dari sistem.

Selain informasi pribadi, halaman detail pelanggan juga menampilkan tombol untuk mengetahui riwayat transaksi pelanggan yang tersimpan di sistem. Contoh tampilan riwayat tersebut ditunjukkan pada Gambar 4.54. Jika pelanggan belum pernah melakukan transaksi, maka sistem akan menampilkan status kosong seperti terlihat pada Gambar 4.55.

Gambar 4.54 Riwayat Transaksi Pelanggan

Gambar 4.55 Riwayat Transaksi Kosong

4.2.9 Implementasi Halaman Manajemen Transaksi

Halaman manajemen transaksi menjadi salah satu fitur terpenting dalam aplikasi karena berfungsi sebagai pusat pencatatan dan pengelolaan proses laundry dari awal hingga selesai. Halaman ini menampilkan daftar transaksi dalam tampilan tab yang terbagi berdasarkan status, yaitu tidak aktif, semua, aktif, sedang diproses, siap diambil, dan sudah diambil. Untuk tab bawaan ketika pertama kali mengakses halaman transaksi ini yaitu tab aktif. Seperti yang ditunjukkan pada Gambar 4.56 di bawah ini.

Gambar 4.56 Halaman Manajemen Transaksi

Setiap tab akan menampilkan data sesuai kategorinya dan ditampilkan secara vertikal. Pengguna juga dapat menyortir daftar transaksi berdasarkan berbagai kriteria, seperti nama pelanggan, layanan, jumlah, status pembayaran, serta tanggal-tanggal penting seperti mulai, selesai, dan dibuat. Fitur sortir ini digambarkan pada Gambar 4.57.

Gambar 4.57 Menu Sortir Transaksi

Ketika pengguna mengetuk salah satu transaksi, sistem akan menampilkan halaman detail transaksi. Contohnya dapat dilihat pada Gambar 4.58, yang memperlihatkan informasi lengkap seperti nomor transaksi, pelanggan, layanan, berat, total biaya, dan nama pegawai yang menangani.

Gambar 4.58 Halaman Detail Transaksi Bagian Atas

Di bagian atas halaman ini terdapat dua label status yaitu status transaksi (misalnya: "Sedang Diproses") dan status pembayaran (misalnya: "Belum Dibayar"). Pengguna dapat mengetuk label tersebut untuk membuka *bottom sheet* pilihan status. Pemilihan status dilakukan dengan satu ketukan, dan sistem akan memperbarui status secara otomatis. Ilustrasi perubahan status ini ditunjukkan secara bertahap pada Gambar 4.59 dan Gambar 4.60.

Gambar 4.59 Ubah Status Transaksi

Gambar 4.60 Ubah Status Pembayaran

Halaman detail transaksi juga menyediakan tiga tombol aksi di bagian bawah, yakni: ubah, hapus dan hapus permanen. Tombol "Ubah" membuka halaman edit transaksi, sementara "Hapus" akan menonaktifkan transaksi (*soft delete*). Tombol "Hapus Permanen" hanya tersedia bagi *Super Admin* untuk benar-benar menghapus data transaksi dari sistem. Tampilan aksi penghapusan serta tampilan setelah transaksi dihapus dapat dilihat pada Gambar 4.61 dan Gambar 4.62.

Gambar 4.61 Dialog Konfirmasi Hapus Transaksi

Gambar 4.62 Transaksi yang Sudah Dihapus

Untuk menambah transaksi baru, pengguna akan diarahkan ke halaman tambah transaksi seperti terlihat pada Gambar 4.63. Halaman ini terdiri atas beberapa langkah, mulai dari memilih pelanggan (Gambar 4.64), memilih layanan (Gambar 4.65), memilih status pembayaran dan status transaksi awal (Gambar 4.66 dan Gambar 4.67), serta mengisi berat cucian dan tanggal transaksi (Gambar 4.68, Gambar 4.69, dan Gambar 4.70). Validasi akan ditampilkan jika kolom wajib tidak diisi, sebagaimana pada Gambar 4.71.

Gambar 4.63 Halaman Tambah Transaksi

Gambar 4.64 Pilih Pelanggan

Gambar 4.65 Pilih Layanan

Gambar 4.66 Pilih Status Transaksi

Gambar 4.67 Pilih Status Pembayaran

Gambar 4.68 Pilih Tanggal Mulai

Gambar 4.69 Pilih Tanggal Selesai

Gambar 4.70 Input Berat Cucian

Gambar 4.71 Validasi Kolom Kosong

Jika belum ada transaksi yang tercatat, sistem akan menampilkan tampilan kosong seperti yang ditunjukkan pada Gambar 4.72.

Gambar 4.72 Tampilan Transaksi Kosong

Selain itu, sistem juga menyediakan fitur cetak nota transaksi melalui printer Bluetooth. Pengguna dapat mencetak nota dari halaman detail transaksi dengan menekan ikon printer di pojok kanan atas. Sebelum mencetak, sistem akan memeriksa izin akses perangkat. Jika belum diizinkan, sistem akan menampilkan

permintaan izin seperti pada Gambar 4.73 mengenai lokasi dan Gambar 4.73 mengenai *device nearby*.

Gambar 4.73 Izin Lokasi

Gambar 4.74 Izin Perangkat Sekitar

Setelah izin diberikan, tampilan akan seperti yang ada di Gambar 4.75, pengguna dapat membuka halaman pengaturan printer melalui tombol yang ada di kanan atas lalu menuju halaman pengaturan printer seperti pada Gambar 4.76. Di sini, pengguna dapat memilih printer, menyambungkannya, dan melakukan tes cetak nota.

Gambar 4.75 Halaman Cetak Nota

Gambar 4.76 Halaman Pengaturan Printer

Gambar 4.77 Printer Terhubung

Untuk mencoba koneksi ke printer yang sudah tersambung. Pengguna bisa mencoba tes cetak nota dengan klik tombol “Cetak Uji” yang hasilnya bisa dilihat pada Gambar 4.78.

Gambar 4.78 Hasil Tes Cetak Nota

Setelah tes cetak nota berhasil, pengguna bisa kembali ke halaman sebelumnya untuk melihat *preview* nota transaksi yang ditunjukkan pada Gambar 4.79, dan jika berhasil mencetak nota maka sistem akan memberikan notifikasi seperti pada Gambar 4.80.

Gambar 4.79 *Preview* Nota Transaksi

Gambar 4.80 Berhasil Cetak Nota

Hasil akhir dari nota cetak juga ditampilkan oleh sistem sebagai referensi visual seperti ditunjukkan pada Gambar 4.81.

Gambar 4.81 Hasil Cetak Nota

4.2.10 Implementasi Halaman Ekspor Laporan

Fitur Ekspor Laporan dirancang untuk memudahkan pemilik usaha dalam mendapatkan data transaksi dalam bentuk laporan keuangan yang dapat dicetak atau dibagikan. Laporan dapat dihasilkan dalam format PDF maupun Excel (XLSX), dan tersedia opsi periode harian, mingguan, dan bulanan.

Saat pengguna mengakses halaman ini, sistem akan menampilkan pilihan periode laporan di bagian atas, sebagaimana ditunjukkan pada Gambar 4.82 di bawah ini. Rentang tanggal dapat ditentukan secara manual menggunakan komponen kalender, dan format berkas dapat dipilih sesuai kebutuhan pengguna.

Gambar 4.82 Halaman Ekspor Laporan

Opsi periode laporan ditampilkan menggunakan *bottom sheet* dan dapat diubah sesuai kebutuhan. Ilustrasi pemilihan periode ditunjukkan pada Gambar 4.83, yang menunjukkan bahwa pengguna dapat memilih laporan harian, mingguan, atau bulanan.

Gambar 4.83 Menu Periode Laporan

Setelah pengguna memilih rentang waktu dan format berkas yang diinginkan, sistem akan menampilkan pratinjau informasi laporan termasuk jenis periode, tanggal, format berkas, dan estimasi jumlah data yang akan diekspor. Setelah itu, pengguna dapat menekan tombol "Ekspor Laporan" untuk memulai proses ekspor.

Jika pengguna memilih ekspor dalam format PDF, sistem akan menghasilkan berkas PDF yang dapat disimpan atau dibagikan seperti yang bisa dilihat pada Gambar 4.84.

Gambar 4.84 Berhasil Ekspor PDF

Gambar 4.85 PDF Berhasil Tersimpan

Gambar 4.86 PDF Dibagikan

Hasil dari berkas PDF yang telah di ekspor bisa dilihat pada Gambar 4.87 berikut.

Gambar 4.87 Hasil Ekspor PDF

Jika pengguna memilih format Excel, tampilan awalnya akan ditunjukkan seperti pada Gambar 4.88, lalu dilanjutkan dengan sukses ekspor yang ditunjukkan pada Gambar 4.89. Seperti di berkas PDF, hasil ekspor juga bisa di simpan maupun di bagikan. Hasil dari berkas ekspor Excel bisa dilihat di Gambar 4.90 berikut.

Gambar 4.88 Proses Ekspor Excel

Gambar 4.89 Berhasil Ekspor Excel

Gambar 4.90 Hasil Ekspor Excel

Fitur ekspor laporan memberikan fleksibilitas dan kemudahan bagi pengguna dalam mendapatkan dokumen pendukung operasional bisnis laundry. Data dapat digunakan untuk keperluan audit, pembukuan, atau diserahkan kepada pihak eksternal sesuai kebutuhan.

4.2.11 Implementasi Halaman Lacak Transaksi

Fitur lain yang tak kalah penting adalah lacak transaksi, yang dirancang untuk pengguna, khususnya untuk *User* biasa agar dapat memantau status laundry mereka secara mandiri. Halaman ini memungkinkan pengguna untuk memasukkan ID transaksi dan melihat detail serta status layanan secara langsung.

Seperti yang ditunjukkan pada Gambar 4.91 di bawah ini, halaman ini memiliki tampilan sederhana namun fungsional. Di bagian atas, terdapat kolom masukan untuk memasukkan ID transaksi, dilengkapi dengan ikon untuk

melakukan pindai kode QR jika ID tersedia dalam bentuk kode. Setelah pengguna menekan tombol “Cari Transaksi”, sistem akan melakukan pencarian dan menampilkan hasilnya.

Gambar 4.91 Halaman Lacak Transaksi

Sebelum menggunakan fitur pindai kode QR, sistem akan terlebih dahulu meminta izin akses ke kamera, yang ditunjukkan pada Gambar 4.92.

Gambar 4.92 Izin Akses Kamera

Jika pengguna melanjutkan proses pindai, tampilan pemindaian akan terlihat seperti pada Gambar 4.93. Setelah pemindaian berhasil, sistem secara otomatis mengisi kolom ID transaksi dengan hasil pindai seperti terlihat pada Gambar 4.94.

Gambar 4.93 Halaman Pindai Kode QR

Gambar 4.94 Berhasil Pindahi Kode QR

Jika pengguna membatalkan proses pindai, maka sistem akan kembali ke halaman awal dan menampilkan status seperti pada Gambar 4.95.

Gambar 4.95 Pindai Kode QR Dibatalkan

Setelah ID transaksi dimasukkan baik manual maupun hasil pindai kode QR, sistem akan melakukan pencarian dan menampilkan hasilnya. Jika transaksi ditemukan, detail dan status akan ditampilkan sebagaimana terlihat pada Gambar 4.96. Namun, jika ID tidak valid atau tidak ditemukan, sistem akan menampilkan notifikasi kegagalan seperti pada Gambar 4.97.

Gambar 4.96 Berhasil Melacak ID Transaksi

Gambar 4.97 Transaksi Tidak Ditemukan

Di bagian bawah layar, terdapat *floating action button* (FAB) yang dapat ditekan oleh pengguna untuk langsung menghubungi *Admin* melalui WhatsApp jika memerlukan bantuan lebih lanjut, seperti dijelaskan dalam panduan penggunaan yang juga tercantum di halaman ini.

Dengan adanya fitur Lacak Transaksi ini, pengguna tidak perlu menanyakan status *laundry* secara manual kepada staf dan pihak pengelola pun terbantu karena dapat mengurangi beban komunikasi secara langsung. Selain itu, dengan fitur ini staf bisa langsung memindai kode QR yang ada di nota *laundry* untuk mengetahui detail dari *laundry* yang lagi dikerjakan.

4.3 IMPLEMENTASI BASIS DATA PADA SUPABASE

Basis data dalam sistem ini dibangun menggunakan layanan Supabase dengan backend PostgreSQL, yang menyediakan fitur-fitur lengkap untuk autentikasi, otorisasi, serta penyimpanan data aplikasi. Desain basis data disesuaikan dengan kebutuhan sistem untuk mengelola pengguna berdasarkan peran (RBAC), layanan *laundry*, pelanggan, transaksi, serta pelaporan.

Struktur basis data terdiri atas skema *auth* yang disediakan oleh Supabase untuk autentikasi pengguna, dan skema *public* yang digunakan untuk menyimpan data aplikasi secara umum. Masing-masing tabel dirancang dengan memperhatikan keterkaitan antar entitas, kemudahan ekspansi, serta keamanan data melalui strategi seperti *soft delete* dan pencatatan waktu (*created_at, updated_at, dan deleted_at*).

Gambaran relasi tabel dalam basis data dapat dilihat pada Gambar 4.98.

Gambar 4.98 Struktur Basis Data di Supabase

4.3.1 Tabel *Auth.Users*

Tabel *auth.users* merupakan bagian dari skema *auth* bawaan dari Supabase yang digunakan untuk menyimpan data autentikasi pengguna. Data di tabel ini secara otomatis dibuat dan dikelola oleh Supabase ketika pengguna melakukan proses registrasi atau *login*.

Beberapa kolom utama pada tabel ini antara lain:

1. id: Merupakan UUID unik yang menjadi identitas utama pengguna.
2. *email*: Alamat *email* yang digunakan saat *login*.
3. *created_at*: Waktu ketika akun dibuat.

Tabel ini berperan sebagai referensi utama untuk autentikasi, dan menjadi kunci relasi pada tabel-tabel lain, seperti tabel *user_roles* dan tabel *public.users*. Karena *auth.users* tidak dapat dimodifikasi secara langsung, data tambahan pengguna seperti nama atau peran disimpan pada tabel di skema *public*.

Relasi dengan tabel lain:

1. Relasi *auth.users.id* digunakan sebagai *foreign key* pada tabel *user_roles* untuk mengelola peran.
2. Relasi juga digunakan pada tabel *public.users* untuk menyimpan profil tambahan.

Dengan memisahkan autentikasi (*auth.users*) dan informasi pengguna (*public.users*), sistem tetap menjaga keamanan data *login* dan fleksibilitas pengelolaan profil secara terpisah.

4.3.2 Tabel *Users*

Tabel *users* yang berada dalam skema *public* digunakan untuk menyimpan data profil pengguna yang lebih lengkap, termasuk nama, nomor telepon, gambar, serta informasi peran. Tabel ini menyimpan identitas pengguna dari *auth.users* melalui kolom *user_id*.

Kolom-kolom utama pada tabel ini antara lain:

1. *id*: *Primary key integer* (otomatis).
2. *user_id*: UUID yang merujuk ke *auth.users.id*, sebagai *foreign key*.
3. *name*: Nama lengkap pengguna.
4. *email*: Email pengguna (opsional, untuk kemudahan akses cepat).
5. *phone*: Nomor telepon pengguna.
6. *image_url*: Link foto profil pengguna.
7. *role_id*: Relasi ke sistem RBAC.
8. *created_at, updated_at, deleted_at*: Kolom *timestamp* untuk mencatat histori data.

Tabel ini berfungsi sebagai sumber data untuk:

1. Menampilkan nama pengguna pada halaman aplikasi.
2. Membedakan peran pengguna seperti *Admin*, *Super Admin*, dan *User* biasa.
3. Mengelola informasi kontak seperti nomor telepon dan gambar profil.

Relasi dengan tabel lain:

1. Kolom *user_id* memiliki relasi ke *auth.users*.*id*.
2. Kolom *role_id* dapat disesuaikan atau dikaitkan dengan logika otorisasi pada tabel RBAC seperti *user_roles* atau *role_permissions*.

Dengan desain seperti ini, informasi pengguna dapat dikelola dan ditampilkan secara fleksibel tanpa mempengaruhi sistem autentikasi inti yang dikelola oleh Supabase.

4.3.3 Tabel *User Roles*

Tabel *user_roles* berfungsi sebagai penghubung antara pengguna dengan perannya dalam sistem. Pendekatan ini memungkinkan sistem mengelola peran pengguna secara dinamis dan terpisah dari sistem autentikasi. Tabel ini juga memungkinkan fleksibilitas untuk mendukung lebih dari satu peran jika diperlukan di masa mendatang.

Kolom-kolom penting pada tabel ini meliputi:

1. id: *Primary key* dengan tipe *integer*.
2. *user_id*: UUID yang merupakan *foreign key* dan merujuk ke *auth.users*.*id*. Kolom ini menandai pengguna yang diberikan peran.
3. *role*: Merupakan enumerasi (enum) dari daftar peran yang tersedia dalam sistem, seperti super_*admin*, *admin*, dan *user*.

Relasi utama dalam tabel ini:

1. *user_id* merujuk langsung ke *auth.users.id* sebagai identitas autentikasi pengguna.

Tabel ini digunakan untuk mendukung *Role-Based Access Control* (RBAC) pada seluruh sistem aplikasi. Peran yang tersimpan akan menentukan hak akses pengguna ke berbagai fitur dalam aplikasi, seperti akses halaman manajemen pegawai (khusus *Super Admin*), kemampuan menghapus data permanen, dan lainnya.

4.3.4 Tabel *Role Permissions*

Untuk mengatur hak akses secara lebih efisien, sistem menggunakan tabel tambahan bernama *role_permissions*. Tabel ini menyimpan daftar izin (*permissions*) yang diberikan pada masing-masing peran (*role*). Dengan pendekatan ini, akses terhadap fitur tertentu tidak dikodekan secara statis, melainkan bisa dikontrol langsung dari database.

Kolom-kolom dalam tabel ini adalah:

1. id: *Primary key* bertipe *integer*.
2. *role*: Enumerasi dari peran (*app_role*) yang berlaku dalam sistem, sama seperti pada tabel *user_roles*.
3. *permission*: Enumerasi dari izin atau hak akses (*app_permission*) yang didefinisikan secara spesifik, seperti *users.select, users.insert, users.update, users.delete,* dsb.

Dengan struktur ini, sistem dapat melakukan validasi hak akses berdasarkan peran pengguna saat aplikasi berjalan. Sebagai contoh, sebelum menampilkan fitur "Hapus Permanen", sistem akan mengecek apakah peran pengguna memiliki *customer.delete* di *role_permissions*. Relasi logis yang terbentuk yaitu kolom *role* di tabel ini terhubung dengan kolom *role* yang ada pada tabel *user_roles*. Kemudian untuk *permission* digunakan sebagai referensi pada pemeriksaan izin di sisi *frontend* atau *backend*. Tabel *role_permissions* menjadikan sistem RBAC ini lebih modular, terukur, dan mudah diperluas di masa depan, tanpa harus mengubah struktur kode secara langsung.

4.3.5 Tabel *Customers*

Tabel *customers* menyimpan data pelanggan *laundry* yang menjadi entitas penting dalam setiap transaksi. Setiap transaksi *laundry* dikaitkan langsung dengan satu pelanggan dari tabel ini. Informasi pelanggan digunakan dalam proses masukan transaksi, pelacakan status *laundry* oleh pelanggan, serta komunikasi melalui WhatsApp.

Kolom-kolom utama dalam tabel ini meliputi:

1. id: *Primary key* bertipe *integer*, sebagai identitas unik pelanggan.
2. *name*: Nama lengkap pelanggan. Kolom ini wajib diisi saat pendaftaran.
3. *phone*: Nomor telepon pelanggan. Kolom ini digunakan untuk integrasi dengan WhatsApp.
4. *gender*: Data jenis kelamin pelanggan, menggunakan enumerasi *app_gender*.
5. *description*: Keterangan tambahan tentang pelanggan, bersifat opsional.
6. *created_at, updated_at, deleted_at*: Kolom *timestamp* untuk mencatat riwayat perubahan data.

Relasi penting dari tabel ini yaitu kolom *id* digunakan sebagai *foreign key* pada tabel *transactions* untuk mengaitkan transaksi dengan pelanggan.

Tabel ini mendukung fungsionalitas seperti:

1. Menyimpan daftar pelanggan aktif.
2. Menampilkan riwayat transaksi berdasarkan pelanggan.
3. Menonaktifkan pelanggan tanpa menghapus data melalui kolom *deleted_at (soft delete).*
4. *Redirect* ke WhatsApp jika *phone* tersedia.

Dengan adanya kolom *created_at* dan *updated_at*, sistem dapat menyortir dan menampilkan pelanggan berdasarkan waktu, serta mendukung pelaporan dan pengarsipan data yang rapi.

4.3.6 Tabel *Services*

Tabel *services* digunakan untuk menyimpan daftar layanan *laundry* yang ditawarkan oleh bisnis. Data dari tabel ini digunakan dalam proses transaksi untuk menentukan jenis layanan yang digunakan, menghitung biaya, serta menampilkan deskripsi layanan.

Kolom-kolom utama pada tabel ini antara lain:

1. *id*: *Primary key* bertipe *integer*.

2. *name*: Nama layanan, misalnya “Cuci Kering”, “Setrika”, atau “Cuci & Setrika”.
3. *price*: Harga layanan (dalam satuan per kilogram).
4. *description*: Informasi tambahan atau syarat layanan, bersifat opsional.
5. *created_at, updated_at, deleted_at*: *Timestamp* yang mencatat histori data dan mendukung *soft delete.*

Relasi penting yang ada di tabel ini yaitu kolom *id* digunakan sebagai *foreign key* pada tabel *transactions* melalui kolom *service_id*. Layanan yang telah dibuat di tabel ini dapat:

1. Diaktifkan atau dinonaktifkan melalui aplikasi *(soft delete).*
2. Diurutkan berdasarkan nama, harga, atau waktu pembuatan.
3. Digunakan dalam proses penambahan dan pengeditan transaksi.

Dengan adanya kolom *deleted_at*, layanan yang sudah tidak digunakan tidak akan tampil dalam daftar aktif, tetapi tetap tersimpan untuk keperluan histori atau pelaporan.

4.3.7 Tabel *Transactions*

Tabel *transactions* merupakan tabel inti dalam sistem ini, yang mencatat seluruh proses layanan *laundry* dari awal hingga selesai. Setiap transaksi terhubung dengan satu pelanggan, satu jenis layanan, dan dicatat oleh satu staf (*Admin* atau *Super Admin*). Informasi dalam tabel ini menjadi dasar utama dalam fitur manajemen transaksi, laporan keuangan, pelacakan status *laundry*, hingga pencetakan nota.

Kolom-kolom utama dalam tabel ini meliputi:

1. *id*: *Primary key* bertipe teks yang disusun dalam format unik, misalnya TRX-250626-001. ID ini juga digunakan oleh pelanggan dalam fitur pelacakan transaksi.
2. *staff_id*: *Foreign key integer* yang merujuk ke *users.id*, menunjukkan siapa staf yang membuat transaksi.

3. *customer_id*: *Foreign key integer* yang merujuk ke *customers.id*, sebagai pemilik cucian.
4. *service_id*: *Foreign key integer* yang merujuk ke *services.id*, menentukan jenis layanan *laundry* yang dipilih.
5. *weight*: Berat cucian dalam satuan kilogram (*float*), menjadi acuan dalam penghitungan biaya.
6. *amount*: Total biaya yang dihitung berdasarkan harga layanan dikali berat cucian.
7. *description*: Catatan atau keterangan tambahan dari staf tentang cucian (opsional).
8. *transaction_status*: Status layanan *laundry* (*enum app_transaction_status*), misalnya: On *Progress, Ready for Pickup, Picked Up,* dan *Other*.
9. *payment_status*: Status pembayaran (*enum app_payment_status*), seperti: *Paid, Not Paid Yet,* dan *Other*.
10. *start_date*: Tanggal dimulainya transaksi, biasanya tanggal masuk cucian.
11. *end_date*: Perkiraan atau tanggal selesai *laundry.*
12. *paid_at*: *Timestamp* saat pembayaran dilakukan (jika status pembayaran diubah ke *Paid*).
13. *created_at, updated_at, deleted_at*: Kolom *timestamp* untuk pencatatan waktu pembuatan, perubahan, dan penghapusan (*soft delete*).

Relasi yang ada di tabel ini dengan tabel lain yaitu:

1. *staff_id* ke tabel *users* dengan kolom *id*
2. *customer_id* ke tabel *customers* dengan kolom *id*
3. *service_id* ke tabel *services* dengan kolom *id*

Tabel ini mendukung fitur-fitur berikut dalam aplikasi:

1. *Input* dan *update* transaksi, termasuk status dan pembayaran.
2. Filter dan sortir transaksi berdasarkan status, tanggal, atau nama.
3. Cetak nota transaksi yang diambil dari data tabel ini.

4. *Soft delete* untuk menonaktifkan transaksi tanpa menghapus data permanen, kecuali oleh *Super Admin*.
5. Histori transaksi pelanggan, yang diambil berdasarkan *customer_id.*

Selain itu, penggunaan *timestamp* (*created_at, updated_at, paid_at*) memungkinkan sistem membuat laporan harian, mingguan, dan bulanan, serta memberikan fleksibilitas dalam audit dan pelacakan aktivitas. Dengan skema ini, tabel *transactions* menjadi fondasi operasional utama dari sistem, menghubungkan seluruh entitas penting dan menjadi pusat data bagi fitur pelaporan, pelacakan, hingga cetak dokumen.

4.4 DESAIN SISTEM APLIKASI

Dalam pengembangan aplikasi manajemen laundry ini, digunakan pendekatan desain sistem yang berfokus pada modularitas, keterpisahan tanggung jawab (*separation of concerns*), dan skalabilitas kode. Dua pendekatan utama yang diterapkan adalah:

1. BLoC (Business Logic Component) sebagai arsitektur manajemen state di lapisan presentasi, dan
2. Clean Architecture sebagai kerangka kerja arsitektur keseluruhan yang memisahkan kode ke dalam tiga lapisan utama: presentation, domain, dan data.

Penggunaan dua pendekatan ini dipilih karena keduanya mendukung prinsip pengembangan perangkat lunak yang bersih, terstruktur, dan mudah diuji. BLoC membantu memisahkan logika bisnis dari antarmuka pengguna, sedangkan Clean Architecture memastikan bahwa setiap lapisan dalam sistem memiliki tanggung jawab yang spesifik dan tidak saling bergantung secara langsung.

Pada bagian ini, akan dijelaskan bagaimana kedua pendekatan tersebut diterapkan, dimulai dari implementasi BLoC pada fitur autentikasi, kemudian dilanjutkan dengan implementasi Clean Architecture dalam struktur kode aplikasi secara menyeluruh.

4.4.1 Penerapan BLoC

Untuk mengelola logika aplikasi secara terstruktur, aplikasi ini menggunakan BLoC sebagai arsitektur manajemen state di sisi presentasi layer. Pendekatan ini dipilih karena BLoC memisahkan antara logika bisnis dan tampilan (UI) sehingga kode lebih bersih, modular, dan mudah diuji.

Dengan BLoC, setiap interaksi pengguna seperti menekan tombol login diproses dalam bentuk *event*, kemudian menghasilkan perubahan status yang disebut dengan *state*. UI hanya perlu merespons perubahan *state* ini, tanpa mengetahui detail dari proses di belakangnya.

1. Pola Umum Penggunaan BLoC

   Implementasi BLoC terdiri dari tiga komponen utama, yaitu:

   a. *Event* sebagai aksi yang dilakukan pengguna atau sistem.
   b. *State* untuk menyimpan status aplikasi saat ini, misalnya sedang loading, sukses, maupun gagal.
   c. BLoC yaitu komponen utama yang mengelola alur *event* dan menghasilkan *state*.

Contoh penerapan BLoC pada fitur *login* dimulai dari *event* *AuthEventLogin*, yang menangkap data email dan kata sandi dari *form login*:

```
class AuthEventLogin extends AuthEvent {
  final String email;
  final String password;

  const AuthEventLogin({
    required this.email,
    required this.password,
  });

  @override
  List<Object> get props => [email, password];
}
```

2. Proses Alur Login Menggunakan BLoC

Ketika pengguna mengisi email dan kata sandi, lalu menekan tombol "Masuk", sistem akan mengeksekusi fungsi *_onLogin()* berikut:

```
void _onLogin() {
  if (_formKey.currentState?.validate() ?? false) {
    final email = _emailController.text;
    final password = _passwordController.text;

    _authBloc.add(AuthEventLogin(
      email: email,
      password: password,
    ));
  }
}
```

Fungsi ini bertugas menambahkan *event AuthEventLogin* ke dalam *AuthBloc* untuk kemudian diproses secara asinkron sesuai alur *login*.

```
on<AuthEventLogin>((event, emit) async {
  emit(AuthStateLoading());

  Either<Failure, Auth> result = await authLogin.call(
    event.email,
    event.password,
  );

  result.fold(
    (left) => emit(AuthStateFailure(message: left.message)),
    (right) => emit(AuthStateSuccessLogin(auth: right)),
  );
});
```

Selama proses berlangsung, *state* akan berubah menjadi *AuthStateLoading*, lalu berganti menjadi *AuthStateFailure* atau *AuthStateSuccessLogin* tergantung hasil dari proses *login*.

3. Representasi State di Aplikasi

Berikut adalah beberapa contoh state yang digunakan dalam proses *login*:

```
class AuthStateInitial extends AuthState {}

class AuthStateLoading extends AuthState {}

class AuthStateFailure extends AuthState {
  final String message;
  const AuthStateFailure({required this.message});
}

class AuthStateSuccessLogin extends AuthState {
  final Auth auth;
  const AuthStateSuccessLogin({required this.auth});
}
```

State inilah yang kemudian digunakan oleh UI untuk menampilkan notifikasi, mengarahkan halaman, atau memunculkan pesan kesalahan seperti yang terlihat pada kode berikut.

```
listener: (context, state) {
 if (state is AuthStateFailure) {
  if (state.message == 'invalid_credentials') {
   context.showSnackbar(context.appText.auth_login_error_message
);
  } else if (state.message == 'email_not_confirmed') {
   context.showSnackbar(context.appText.auth_login_error_email_n
ot_confirmed);
  } else {
   context.showSnackbar(state.message.toString());
  }
 } else if (state is AuthStateSuccessLogin) {
  context.showSnackbar(context.appText.auth_login_success_messag
e);
  pushReplacementHome(context: context);
 }
},
```

4. Keuntungan Penggunaan BLoC

   Dengan pendekatan ini, maka sistem akan:

   a. Memiliki alur logika yang jelas dari masukan pengguna hingga respons UI.
   b. Mudah untuk diuji unit secara terpisah (*event* dan *state*).
   c. Memungkinkan UI tetap *stateless* dan hanya bertugas sebagai representasi data.

Penerapan BLoC telah digunakan secara konsisten di seluruh fitur aplikasi, mulai dari login, register, hingga manajemen data. Namun dalam bagian ini, contoh fokus diberikan pada modul autentikasi login saja.

4.4.2 Penerapan *Clean Architecture*

Selain menggunakan BLoC untuk manajemen state di lapisan presentasi, aplikasi ini juga menerapkan pendekatan Clean Architecture sebagai kerangka kerja utama dalam pemisahan struktur kode. Clean Architecture membagi aplikasi ke dalam tiga lapisan inti, yaitu Presentation, Domain, dan Data, yang masing-masing memiliki tanggung jawab tersendiri dan tidak saling bergantung secara langsung.

Dengan pendekatan ini, setiap logika bisnis ditangani secara independen di domain layer, sedangkan data yang berasal dari API atau sumber eksternal lainnya dikelola di data layer. *Presentation layer* bertugas hanya sebagai penghubung dengan antarmuka pengguna melalui BLoC.

Untuk memberikan gambaran konkret, implementasi Clean Architecture dalam fitur *login* akan dijelaskan secara bertahap melalui struktur ketiga layer tersebut.

1. Domain Layer

Pada lapisan domain, logika bisnis aplikasi didefinisikan melalui *use case*. Untuk proses *login*, digunakan *use case* bernama *AuthLogin*, yang menerima email dan kata sandi sebagai parameter, dan melemparkan hasilnya berupa entitas *Auth* dalam bentuk *Either<Failure, Auth>*.

```
class AuthLogin {
  final AuthRepository authRepository;

  const AuthLogin({
    required this.authRepository,
  });

  Future<Either<Failure, Auth>> call(String email, String
password) async {
    return await authRepository.login(email, password);
  }
}
```

Use case ini hanya mengenal abstraksi *AuthRepository* dan tidak bergantung pada implementasi datanya. Hal ini memungkinkan unit testing dilakukan tanpa bergantung pada koneksi ke *backend*.

```
abstract class AuthRepository {
  Future<Either<Failure, Auth>> login(
    String email,
    String password,
  );
}
```

2. Data Layer

Implementasi dari *AuthRepository* dilakukan di data layer melalui kelas *AuthRepositoryImplementation*, yang bertanggung jawab menangani detail proses *login* dengan bantuan *datasource* misalnya dari Supabase.

```
class AuthRepositoryImplementation extends AuthRepository {
  final AuthRemoteDatasource authRemoteDatasource;

  AuthRepositoryImplementation({
    required this.authRemoteDatasource,
  });

  @override
  Future<Either<Failure, Auth>> login(
    String email,
    String password,
  ) async {
    try {
      final response = await authRemoteDatasource.login(email,
password);

      AuthManager.setCurrentUser(response);

      if (response.accessToken == null) {
        RoleManager.setUserRole('user');
      } else {
        final                    userRole                    =
response.accessToken!.getUserRoleFromJwt();
        RoleManager.setUserRole(userRole);
        await         authRemoteDatasource.saveAuth(response.id,
response.accessToken!);
      }

      return Right(response);
    } on ServerException catch (e) {
      return Left(Failure(message: e.message));
    } catch (e) {
      return Left(Failure(message: e.toString()));
    }
  }
}
```

Repositori ini bertugas untuk melakukan komunikasi ke server melalui *datasource*, mengelola *role* pengguna berdasarkan isi dari token JWT, dan menyimpan status *login* saat berhasil.

Datasource yang digunakan di sini adalah *AuthRemoteDatasource*, yang merupakan abstraksi untuk proses *login*.

```
abstract class AuthRemoteDatasource {
  Future<AuthModel> login(String email, String password);
}
```

Implementasi dari *datasource* berada di kelas *AuthRemoteDatasourceImplementation*, yang melakukan autentikasi ke Supabase menggunakan supabaseClient.

```
Class AuthRemoteDatasourceImplementation extends
AuthRemoteDatasource {
  final SupabaseClient supabaseClient;

  AuthRemoteDatasourceImplementation({
    required this.supabaseClient,
  });

  @override
  Future<AuthModel> login(String email, String password) async {
    try {
      final       AuthResponse       response       =       await
supabaseClient.auth.signInWithPassword(
        email: email,
        password: password,
      );

      return AuthModel(
        id: response.user!.id,
        accessToken: response.session!.accessToken,
        email: response.user!.email ?? '',
        name: response.user!.userMetadata!['name'] ?? '',
      );
    } on AuthException catch (e) {
      throw ServerException(message: e.code.toString());
    } catch (e) {
      throw ServerException(message: e.toString());
    }
  }
}
```

3. Auth Model

Model *AuthModel* digunakan sebagai *data transfer object* (DTO) untuk mentransformasikan hasil dari API yang menjadi entitas domain *Auth*. Model ini mewarisi entitas dan menambahkan kemampuan untuk *parsing* data dari format JSON.

```
class AuthModel extends Auth {
  const AuthModel({
    required super.accessToken,
    required super.id,
    required super.email,
    required super.name,
  });

  factory AuthModel.fromJson(Map<String, dynamic> json) {
    return AuthModel(
      accessToken: json['accessToken'],
      id: json['id'],
      email: json['email'],
      name: json['name'],
    );
  }
}
```

4. Alur Kerja Lengkap

Secara ringkas, alur *login* menggunakan Clean Architecture dapat dijelaskan sebagai berikut:

a. *Presentation layer* (melalui BLoC) mengirimkan *event login* ke *use case AuthLogin*.
b. *Use case* memanggil abstraksi repositori *AuthRepository*.
c. Implementasi repositori *(AuthRepositoryImplementation)* meneruskan permintaan ke *remote datasource*.
d. *Remote datasource (AuthRemoteDatasourceImplementation)* memanggil API Supabase dan mengembalikan data.
e. Hasil *login* diteruskan kembali ke *domain layer* sebagai entitas *Auth*, lalu diproses di BLoC.

Dengan struktur ini, sistem menjadi lebih terstruktur, mudah diuji, dan siap dikembangkan lebih lanjut tanpa perlu mengubah seluruh bagian sistem saat terjadi perubahan pada salah satu lapisan.

4.5 PEMBAHASAN HASIL EVALUASI SISTEM

Setelah aplikasi selesai dikembangkan dan diuji secara internal, dilakukan proses evaluasi terhadap pengguna langsung dari Laundry LB Fresh, yaitu pemilik dan staf yang menggunakan sistem dalam operasional sehari-hari. Evaluasi ini

bertujuan untuk mengetahui sejauh mana sistem yang dibangun sesuai dengan kebutuhan nyata, baik dari segi fitur, struktur data, maupun antarmuka pengguna.

Metode evaluasi dilakukan melalui penyebaran formulir *Google Form* dengan model penilaian skala 1–5, di mana skor 1 berarti "Sangat Tidak Setuju" dan skor 5 berarti "Sangat Setuju". Pertanyaan evaluasi dikelompokkan ke dalam tiga aspek utama yang sejalan dengan pertanyaan penelitian, yaitu: kesesuaian fitur, struktur dan keandalan data, dan desain antarmuka pengguna (UI).

4.5.1 Penilaian Kesesuaian Fitur Aplikasi

Dari hasil evaluasi terhadap fitur-fitur aplikasi, pihak Laundry LB Fresh memberikan respons positif terhadap fungsionalitas inti sistem. Fitur pencatatan transaksi digital dinilai memudahkan pencatatan dan pengelolaan cucian harian. Selain itu, kemampuan sistem untuk menampilkan status cucian seperti “Sedang Diproses”, “Siap Diambil”, dan “Sudah Diambil” sangat membantu staf dalam memantau alur kerja.

Fitur ekspor laporan ke dalam format PDF dan Excel juga mendapatkan skor tinggi karena sangat berguna dalam pemantauan pendapatan rutin. Sementara itu, kemampuan cetak nota dengan printer Bluetooth dianggap sebagai keunggulan aplikasi dalam pelayanan langsung ke pelanggan.

Tabel 4.1 berikut menunjukkan hasil rata-rata penilaian pengguna terhadap fitur-fitur aplikasi.

Tabel 4.1 Hasil Evaluasi Kesesuaian Fitur Aplikasi

No.	Pertanyaan	Skor Rata-rata
1.	Fitur-fitur aplikasi sesuai dengan kebutuhan operasional laundry	5
2.	Fitur pencatatan transaksi membantu pekerjaan	5
3.	Informasi status laundry mudah dipantau	4.7
4.	Fitur ekspor laporan membantu dalam melihat data keuangan	5
5.	Fitur cetak nota melalui printer sangat bermanfaat	5

4.5.2 Penilaian Struktur dan Keandalan Data

Dari sisi struktur data, sistem memperoleh penilaian baik dalam hal kecepatan akses, stabilitas penyimpanan, dan minimnya *error* saat pengguna berinteraksi dengan data. Pihak *laundry* menyatakan bahwa pencarian pelanggan dan layanan dapat dilakukan lumayan cepat, dan tidak mengalami kehilangan data atau transaksi yang tidak tersimpan.

Rincian skor evaluasi terhadap aspek ini dapat dilihat pada Tabel 4.2 berikut:

Tabel 4.2 Hasil Evaluasi Struktur dan Keandalan Data

No.	Pertanyaan	Skor Rata-rata
1.	Proses pencarian data pelanggan, layanan, dan transaksi terasa cepat	4.7
2.	Data yang disimpan tidak bermasalah	5
3.	Tidak ada kendala saat menyimpan atau mengedit data	5

4.5.3 Penilaian Tampilan dan Kemudahan Penggunaan

Antarmuka pengguna juga mendapatkan tanggapan positif. Aplikasi dinilai mudah dipahami dan digunakan bahkan oleh staf yang tidak memiliki latar belakang teknologi. Navigasi antar fitur juga berjalan lancar, dan elemen visual seperti ikon serta teks cukup membantu pengguna dalam memahami fungsi tiap tombol maupun kolom.

Tabel 4.3 berikut ini menyajikan skor penilaian dari aspek kemudahan dan kenyamanan penggunaan antarmuka aplikasi:

Tabel 4.3 Hasil Evaluasi Antarmuka Pengguna (UI)

No.	Pertanyaan	Skor Rata-rata
1.	Tampilan aplikasi mudah dipahami	5
2.	Alur penggunaan aplikasi tidak membingungkan	5
3.	Penggunaan aplikasi nyaman meskipun tanpa pelatihan khusus	5