

**LAPORAN PROYEK UJIAN AKHIR SEMESTER
PEMROGRAMAN BERORIENTASI OBJEK**

Dosen Pengampu :

Taufik Ridwan, S.T., M.T.



Disusun Oleh :

Kelompok 6 :

1. Adinda Khaerunnisa (2310631250082)
2. Kartika Sari (2310631250022)
3. Rizka Amaniah (2310631250076)
4. Siti Marsiah (2310631250033)

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
T.A 2025/2026**

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan Tugas Ujian Akhir Semester (UAS) Mata Kuliah Pemrograman Berbasis Objek (PBO) ini dengan baik dan tepat waktu.

Laporan ini berisi perancangan dan implementasi program yang mengaplikasikan konsep-konsep Pemrograman Berbasis Objek melalui pembuatan aplikasi manajemen layanan laundry yang terhubung dengan database untuk mengelola data pelanggan, layanan, transaksi, serta operasi baca, perbarui, dan hapus data. Program ini dirancang sesuai dengan ketentuan tugas, dengan menerapkan prinsip-prinsip OOP, penggunaan Graphic User Interface (GUI), serta integrasi dengan database guna mendukung pengelolaan data laundry secara efisien dan sistematis. Dalam laporan ini, kami menyusun langkah-langkah perancangan sistem menggunakan Unified Modeling Language (UML), termasuk pembuatan Class Diagram sebagai dasar desain sistem. Selain itu, laporan ini juga mencakup implementasi kode program serta proses pengujian untuk memastikan aplikasi dapat berjalan sesuai dengan kebutuhan.

Laporan ini merupakan hasil dari kerja sama dan dedikasi seluruh anggota Kelompok 6, yang telah berkontribusi secara maksimal dalam menyelesaikan tugas ini. Kami juga mengucapkan terima kasih kepada dosen pengampu Mata Kuliah Pemrograman Berbasis Objek atas bimbingan, arahan, serta materi yang telah membantu kami selama proses pengerjaan.

Kami menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kami dengan terbuka menerima kritik dan saran yang membangun untuk perbaikan ke depan. Semoga laporan ini dapat bermanfaat, khususnya dalam memahami dan menerapkan konsep Pemrograman Berbasis Objek dalam pengembangan aplikasi desktop yang terhubung dengan database. Demikian kata pengantar ini kami sampaikan. Terima kasih atas perhatian dan dukungan dari semua pihak yang telah membantu dalam proses penyusunan laporan ini.

Karawang, 22 Mei 2025

Kelompok 6

DAFTAR ISI

| | |
|--|-----|
| KATA PENGANTAR | ii |
| DAFTAR ISI..... | iii |
| BAB I..... | 1 |
| PENDAHULUAN | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Rumusan Masalah | 2 |
| 1.3 Tujuan..... | 2 |
| 1.4 Manfaat..... | 3 |
| BAB II..... | 4 |
| LANDASAN TEORI..... | 4 |
| 2.1 Java untuk GUI..... | 4 |
| 2.2 GUI dengan NetBeans..... | 5 |
| 2.3 Java Database Connectivity dengan MySQL..... | 6 |
| 2.4 Unified Modeling Language (UML)..... | 7 |
| 2.5 Entity Relationship Diagram..... | 8 |
| 2.6 Class Diagram | 9 |
| 2.7 Activity Diagram..... | 9 |
| 2.8 Use Case Diagram..... | 9 |
| BAB III | 11 |
| ISI..... | 11 |
| 3. 1 Fitur Aplikasi..... | 11 |
| 3.1.1. Login..... | 11 |
| 3.1.2 Pesanan | 11 |
| 3.1.3 Pembayaran | 12 |
| 3. 2 Konsep OOP di kode program | 14 |
| 3.3 Database | 17 |
| 3.4 UML..... | 19 |
| 3.5 Implementasi kode Program dan Pengujian..... | 29 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemrograman Berbasis Objek (PBO) merupakan pendekatan dalam pengembangan perangkat lunak yang memungkinkan pengelolaan data dan fungsionalitas dalam bentuk objek, sehingga membuat sistem menjadi lebih modular dan mudah dikembangkan. Konsep ini sangat cocok diterapkan dalam pengembangan aplikasi manajemen *laundry*, di mana berbagai entitas seperti pelanggan, layanan, transaksi, dan pembayaran harus dikelola secara terstruktur.

Laundry merupakan salah satu bentuk usaha jasa yang banyak diminati masyarakat, terutama di daerah perkotaan. Dalam pengelolaannya, banyak *laundry* masih menggunakan pencatatan manual yang berisiko menimbulkan kesalahan, seperti kehilangan data, pencatatan ganda, atau keterlambatan dalam memperbarui status layanan. Hal ini dapat menghambat efisiensi operasional dan memengaruhi kepuasan pelanggan.

Untuk mengatasi permasalahan tersebut, perlu dikembangkan aplikasi manajemen *laundry* berbasis Java yang dilengkapi dengan GUI dan terhubung dengan database. Aplikasi ini akan mempermudah pengelolaan data pelanggan, layanan (cuci kering, cuci lipat, dll), serta transaksi dan status *order*. Dengan sistem ini, proses operasional *laundry* dapat dilakukan secara lebih cepat, akurat, dan terdokumentasi dengan baik.

Aplikasi ini akan dibangun menggunakan bahasa pemrograman Java, yang memiliki keunggulan dalam pengembangan aplikasi desktop, serta kemudahan dalam koneksi dengan *database MySQL*. Sistem akan mengimplementasikan operasi CRUD (*Create, Read, Update, Delete*), memungkinkan pengguna untuk menambah, melihat, mengubah, dan menghapus data dengan lebih efektif. Selain itu, dengan penggunaan relasi antar tabel dalam *database*, aplikasi dapat menghasilkan laporan-laporan seperti riwayat transaksi, status layanan, hingga laporan pelanggan tetap.

Laporan ini akan membahas tahap-tahap perancangan sistem menggunakan UML, implementasi kode program menggunakan Java, serta pengujian fungsionalitas aplikasi agar sesuai dengan kebutuhan pengguna.

Dalam pengembangan aplikasi ini, prinsip-prinsip Pemrograman Berbasis Objek seperti enkapsulasi, pewarisan, dan *polimorfisme* akan diterapkan guna menghasilkan struktur kode yang terorganisir dan mudah untuk dikembangkan di masa mendatang. Dengan penggunaan kelas-kelas yang merepresentasikan entitas dalam sistem seperti Pelanggan, Layanan, Transaksi, dan Pembayaran, proses pengelolaan data akan menjadi lebih sistematis dan fleksibel. Penggunaan antarmuka grafis (GUI) juga akan memberikan pengalaman pengguna yang lebih baik, memudahkan staf *laundry* dalam mengoperasikan aplikasi tanpa harus memiliki latar belakang teknis. Dengan demikian, aplikasi ini diharapkan mampu menjadi solusi digital yang praktis, efisien, dan tepat guna bagi usaha laundry skala kecil hingga menengah dalam meningkatkan kualitas layanan dan manajemen operasional sehari-hari.

1.2 Rumusan Masalah

1. Bagaimana merancang dan mengimplementasikan aplikasi manajemen *laundry* berbasis Java untuk mengelola data pelanggan, layanan, dan transaksi secara terintegrasi?
2. Bagaimana aplikasi ini dapat menggantikan proses pencatatan manual yang rentan terhadap kesalahan dan keterlambatan?
3. Bagaimana penerapan konsep CRUD dapat meningkatkan efisiensi dan akurasi dalam pengelolaan data *laundry*?
4. Bagaimana aplikasi ini dapat membantu pemilik usaha *laundry* dalam memantau layanan dan menyusun laporan operasional dengan lebih mudah?

1.3 Tujuan

1. Merancang dan membangun aplikasi manajemen *laundry* berbasis Java yang terhubung dengan *database* dan GUI yang *user-friendly*.
2. Mengembangkan aplikasi yang mampu menggantikan sistem pencatatan manual dengan proses digital yang lebih akurat dan efisien.
3. Menerapkan konsep CRUD dalam aplikasi untuk memudahkan pengelolaan data pelanggan, layanan, dan transaksi.
4. Menghasilkan aplikasi yang dapat menyajikan laporan layanan dan transaksi dengan akurat, guna membantu proses evaluasi dan pengambilan keputusan oleh pengelola *laundry*.

1.4 Manfaat

1. Meningkatkan efisiensi operasional *laundry* melalui digitalisasi proses pengelolaan data dan transaksi.
2. Mengurangi risiko kesalahan pencatatan yang sering terjadi pada proses manual.
3. Memudahkan pemilik usaha dalam melihat laporan transaksi dan status layanan.
4. Meningkatkan kepuasan pelanggan melalui pelayanan yang lebih cepat dan terorganisir.
5. Mendukung pengembangan usaha *laundry* agar dapat bersaing di era digital.
6. Memberikan pengalaman praktis dalam implementasi OOP dan integrasi database dalam pengembangan aplikasi.

BAB II

LANDASAN TEORI

2.1 Java untuk GUI

Java adalah bahasa pemrograman yang banyak digunakan untuk pengembangan aplikasi, termasuk aplikasi desktop dengan *Graphical User Interface* (GUI). GUI merupakan antarmuka yang memanfaatkan elemen-elemen grafis seperti tombol, teks, dan gambar untuk memungkinkan pengguna berinteraksi dengan aplikasi. Java menyediakan pustaka untuk membangun GUI yang fleksibel dan dapat dijalankan di berbagai *platform*, berkat konsep *Write Once, Run Anywhere* (WORA). Salah satu pustaka GUI yang terkenal di Java adalah *Swing*, yang merupakan bagian dari *Java Foundation Classes* (JFC). *Swing* memungkinkan pengembang untuk membuat aplikasi desktop dengan komponen-komponen GUI seperti *JFrame* (untuk jendela utama), *JButton* (tombol), *JLabel* (label teks), dan *JTextField* (kolom input teks). Komponen-komponen ini dapat disesuaikan dan diatur sedemikian rupa untuk menciptakan antarmuka yang *user-friendly* dan konsisten di berbagai sistem operasi.

Salah satu aspek penting dalam pengembangan aplikasi GUI adalah *event handling*, yaitu menangani aksi pengguna seperti klik tombol atau input dari *keyboard*. Java menggunakan event listeners untuk menangani berbagai jenis peristiwa tersebut, seperti *ActionListener* untuk tombol, *MouseListener* untuk interaksi *mouse*, dan *KeyListener* untuk input keyboard. Pendekatan ini membuat aplikasi menjadi interaktif dan responsif terhadap aksi pengguna.

Java juga memungkinkan aplikasi GUI terhubung dengan database menggunakan JDBC (*Java Database Connectivity*), yang memudahkan aplikasi untuk mengelola data secara langsung dari antarmuka pengguna. Dengan menggunakan CRUD (*Create, Read, Update, Delete*), aplikasi dapat mengelola data produk atau transaksi secara efisien, sehingga mempercepat proses transaksi dan pengelolaan data. Dengan kemampuan untuk terintegrasi dengan berbagai jenis database seperti MySQL, PostgreSQL, dan SQLite, Java memberikan solusi yang aman dan terstruktur dalam pengelolaan data.

Secara keseluruhan, Java untuk GUI menawarkan banyak keuntungan, seperti portabilitas, kemudahan pemeliharaan, dan fleksibilitas dalam mengembangkan aplikasi berbasis antarmuka grafis. Melalui pustaka seperti *Swing* adalah dukungan untuk integrasi

database menggunakan JDBC, Java memungkinkan pengembang untuk membangun aplikasi desktop yang interaktif, responsif, dan mudah diakses oleh pengguna.

2.2 GUI dengan NetBeans

NetBeans adalah salah satu *Integrated Development Environment* (IDE) yang banyak digunakan untuk pengembangan aplikasi berbasis Java, terutama untuk membangun aplikasi *Graphical User Interface* (GUI). NetBeans menyediakan berbagai alat dan pustaka yang memungkinkan pengembang untuk membuat antarmuka pengguna yang interaktif, mudah digunakan, dan terintegrasi dengan berbagai fitur seperti *database*, *web services*, dan lain sebagainya. Dengan antarmuka yang sederhana dan mudah dipahami, NetBeans telah menjadi pilihan utama bagi banyak pengembang aplikasi Java.

Dalam pengembangan aplikasi GUI menggunakan NetBeans, pengembang dapat memanfaatkan *Swing* dan *JavaFX*, dua pustaka Java yang digunakan untuk membangun antarmuka grafis. *Swing* adalah pustaka GUI yang sudah ada sejak lama dalam Java dan merupakan bagian dari *Java Foundation Classes* (JFC). *Swing* menyediakan berbagai komponen GUI seperti *JFrame* (untuk jendela utama), *JButton* (tombol), *JLabel* (label teks), *JTextField* (kolom input teks), dan banyak lagi. Salah satu kelebihan menggunakan *Swing* adalah kemampuannya untuk membangun aplikasi GUI yang portabel, yaitu aplikasi yang dapat berjalan di berbagai platform (*Windows*, *Linux*, *macOS*) tanpa memerlukan perubahan kode..

NetBeans mempermudah pengembangan aplikasi GUI dengan menyediakan *GUI Builder* atau *Matisse*, alat visual yang memungkinkan pengembang untuk membangun antarmuka pengguna menggunakan pendekatan *drag-and-drop*. Dengan menggunakan *GUI Builder* di NetBeans, pengembang dapat dengan mudah menambahkan komponen seperti tombol, kolom teks, dan label ke dalam jendela aplikasi tanpa menulis kode secara manual. NetBeans secara otomatis menghasilkan kode Java yang diperlukan untuk menghubungkan komponen GUI dengan logika program, yang menghemat waktu dan usaha dalam pengembangan aplikasi.

Selain itu, NetBeans juga menyediakan kemudahan dalam mengintegrasikan aplikasi GUI dengan database menggunakan JDBC (*Java Database Connectivity*). Dengan JDBC, aplikasi GUI dapat melakukan operasi *CRUD* (*Create, Read, Update, Delete*) terhadap data yang ada di dalam database. Hal ini memungkinkan pengembang untuk membuat aplikasi

berbasis data yang dapat mengelola informasi dengan cara yang lebih efisien dan akurat. Dengan dukungan Swing dan JavaFX yang dapat terintegrasi dengan JDBC, NetBeans menjadi alat yang sangat *powerful* dalam pengembangan aplikasi berbasis Java yang membutuhkan antarmuka grafis yang interaktif dan pengelolaan data yang terstruktur.

Secara keseluruhan, NetBeans menyediakan lingkungan pengembangan yang komprehensif dan mudah digunakan untuk membangun aplikasi GUI berbasis Java. Dengan integrasi pustaka seperti *Swing* memudahkan dalam menghubungkan aplikasi dengan *database* menggunakan JDBC, NetBeans memungkinkan pengembang untuk membuat aplikasi yang kuat, responsif, dan interaktif, serta dapat berjalan dengan baik di berbagai platform.

2.3 Java Database Connectivity dengan MySQL

Java Database Connectivity (JDBC) adalah API (*Application Programming Interface*) yang memungkinkan aplikasi Java untuk terhubung dan berinteraksi dengan sistem manajemen basis data (DBMS) seperti MySQL. JDBC menyediakan antarmuka yang memungkinkan aplikasi Java melakukan berbagai operasi CRUD (*Create, Read, Update, Delete*) pada data yang ada di dalam database. Dengan menggunakan JDBC, pengembang aplikasi Java dapat mengelola data yang disimpan di database dengan cara yang terstruktur dan efisien. MySQL, yang merupakan salah satu DBMS yang paling banyak digunakan, sering menjadi pilihan untuk mengelola data dalam aplikasi berbasis Java karena kemudahan penggunaan, skalabilitas, dan kemampuannya untuk menangani berbagai jenis data dengan cepat dan aman.

Dalam JDBC, proses pertama yang dilakukan adalah membangun koneksi antara aplikasi Java dan *database*. Koneksi ini dibuat menggunakan objek *Connection*, yang diperlukan untuk menghubungkan aplikasi dengan *database*. Koneksi ini mengharuskan pengembang untuk menyediakan informasi seperti URL *database*, *username*, dan *password* agar aplikasi dapat mengakses data yang ada. Setelah koneksi berhasil, pengembang dapat menggunakan objek *Statement*, *PreparedStatement*, atau *CallableStatement* untuk mengeksekusi query SQL yang berisi perintah-perintah untuk mengambil atau memodifikasi data dalam database. *PreparedStatement* khususnya, digunakan untuk mencegah SQL injection dan mengoptimalkan performa query yang sering digunakan.

Setelah *query* dieksekusi, hasilnya akan disimpan dalam objek *ResultSet*, yang memungkinkan aplikasi Java untuk memproses data yang dikembalikan dari *database* dalam sebuah aplikasi apotek, *ResultSet* dapat digunakan untuk menampilkan data produk obat yang

tersedia di apotek, atau mengupdate stok obat berdasarkan transaksi yang baru saja dilakukan. Setelah operasi selesai, penting untuk menutup koneksi, *statement*, dan *result set* untuk melepaskan sumber daya yang digunakan, menghindari kebocoran memori, dan menjaga kinerja aplikasi.

Keuntungan utama dari penggunaan JDBC dengan MySQL adalah portabilitas dan fleksibilitas. Aplikasi Java yang menggunakan JDBC dapat dijalankan pada berbagai sistem operasi tanpa perlu menyesuaikan kode, berkat sifat *cross-platform* Java. Selain itu, penggunaan JDBC memungkinkan aplikasi untuk terhubung langsung dengan MySQL, memberikan pengelolaan data yang lebih baik dan mendalam. JDBC juga memungkinkan aplikasi Java untuk memanfaatkan fitur-fitur canggih dari MySQL, seperti transaksi, *foreign key*, dan *indeks*, yang dapat meningkatkan performa dan integritas data.

Dengan integrasi antara JDBC dan MySQL, pengembang dapat membuat aplikasi Java yang terhubung dengan basis data untuk menyimpan, memperbarui, dan mengelola data secara efisien. JDBC memastikan komunikasi yang aman dan efisien antara aplikasi dan *database*, yang penting untuk aplikasi-aplikasi berbasis data seperti *Point of Sales* (POS), sistem manajemen inventaris, atau aplikasi keuangan. JDBC mempermudah pengelolaan data yang lebih kompleks, memungkinkan aplikasi untuk bekerja lebih cepat dan lebih aman dalam memanipulasi data yang besar dan terstruktur.

2.4 Unified Modeling Language (UML)

Unified Modeling Language adalah bahasa standar yang digunakan untuk mendesain dan memodelkan sistem perangkat lunak. UML menyediakan notasi dan diagram yang digunakan untuk menggambarkan struktur dan perilaku sistem dalam berbagai fase pengembangan perangkat lunak. UML memfasilitasi komunikasi antara pengembang, pemangku kepentingan, dan anggota tim dalam proses desain dan implementasi. Dengan UML, pengembang dapat menggambarkan dengan jelas komponen-komponen yang ada dalam sistem, cara mereka berinteraksi, serta bagaimana sistem bekerja secara keseluruhan.

Tujuan utama penggunaan UML adalah untuk menyediakan visualisasi yang jelas dari sistem perangkat lunak, yang membantu pengembang dan pemangku kepentingan untuk memahami sistem secara lebih mendalam. UML juga digunakan untuk mengkomunikasikan ide-ide desain dengan lebih efektif antar anggota tim pengembang dan pemangku kepentingan, sehingga mengurangi kesalahpahaman. Selain itu, UML sangat berguna dalam

mendokumentasikan desain sistem secara jelas dan terstruktur, yang dapat digunakan sebagai referensi untuk pengembangan di masa depan atau pemeliharaan sistem. UML terdiri dari dua kategori utama Structural Diagrams (diagram struktural) dan Behavioral Diagrams (diagram perilaku). Setiap kategori ini memiliki diagram dengan tujuan tertentu untuk menggambarkan aspek berbeda dari sistem perangkat lunak.

2.5 Entity Relationship Diagram

ERD (Entity Relationship Diagram) atau diagram hubungan entitas adalah sebuah diagram yang digunakan untuk perancangan suatu *database* dan menunjukkan relasi atau hubungan antar objek atau entitas beserta atribut-atributnya secara detail. Dengan menggunakan *ERD*, sistem *database* yang sedang dibentuk dapat digambarkan dengan lebih terstruktur dan terlihat rapi. *ERD* sendiri sering digunakan untuk *debugging database* jika terjadi masalah pada *database*. Adapun komponen *ERD* ialah:

1. Entitas

Entitas merupakan sekumpulan objek yang dapat diidentifikasi secara unik dan berbeda satu dengan yang lainnya. Entitas ini biasanya digambarkan dengan lambang persegi panjang.

2. Atribut

Atribut setiap entitas pasti memiliki atribut yang berfungsi untuk menjelaskan atau mendeskripsikan karakteristik dari entitas tersebut.

3. Relasi

Dalam *ERD* adalah hubungan yang terjadi antara satu atau lebih entitas. Relasi sendiri sering disebut dengan proses. Komponen ini digambarkan dengan lambang belah ketupat. Terdapat tiga jenis relasi yang digunakan dalam *ERD* dan perlu kamu ketahui, berikut adalah jenisnya.

- a. *One to one*

One to one berarti setiap entitas hanya dapat memiliki relasi dengan satu entitas lain.

- b. *One to many*

One to many memiliki arti satu entitas dapat memiliki relasi dengan beberapa entitas, begitu pula sebaliknya.

- c. *Many to many*

Many to many memiliki arti setiap entitas yang ada dapat memiliki relasi dengan entitas lain, begitu pula sebaliknya.

2.6 Class Diagram

Class Diagram dalam UML (*Unified Modeling Language*) adalah salah satu diagram yang digunakan untuk menggambarkan struktur statis dari sistem perangkat lunak. Diagram ini menunjukkan kelas-kelas dalam sistem, yang berfungsi sebagai template atau *blueprint* untuk membuat objek-objek dalam program. Setiap kelas dalam diagram memiliki atribut-atribut yang menggambarkan data atau properti yang dimiliki kelas tersebut, serta metode-metode yang menggambarkan perilaku atau tindakan yang dapat dilakukan oleh objek kelas tersebut. *Class Diagram* juga menggambarkan relasi antar kelas, seperti asosiasi, pewarisan, agregasi, dan komposisi, yang menunjukkan bagaimana kelas-kelas dalam sistem saling berhubungan dan berinteraksi.

2.7 Activity Diagram

Activity Diagram adalah bagian penting dari *UML* yang menggambarkan aspek dinamis dari sistem. Logika prosedural, proses bisnis, dan aliran kerja suatu bisnis dapat dengan mudah dideskripsikan dalam *activity* diagram. Tujuan dari *activity* diagram adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukkan aliran pesan dari satu aktivitas ke aktivitas lain. Secara umum *Activity Diagram* digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktivitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran parallel, swim lane dsb.

2.8 Use Case Diagram

Use Case Diagram adalah jenis diagram dalam UML (*Unified Modeling Language*) yang digunakan untuk menggambarkan fungsionalitas sistem dari sudut pandang pengguna atau aktor eksternal. Diagram ini berfokus pada interaksi antara aktor (seperti pengguna atau sistem lain) dengan sistem, dan menggambarkan fungsi atau fitur yang disediakan oleh sistem untuk memenuhi kebutuhan aktor.

Dalam *Use Case Diagram*, terdapat dua elemen utama aktor dan *use case*. Aktor adalah entitas yang berinteraksi dengan sistem, seperti pengguna atau sistem lain yang memulai

interaksi. Use case adalah representasi dari suatu fungsionalitas atau layanan yang disediakan oleh sistem, seperti "*login*", "tambah data", "cetak laporan", dan lain-lain.

Relasi antar aktor dan *use case* digambarkan dengan garis yang menunjukkan interaksi antara aktor dan fungsi yang mereka gunakan. Selain itu, diagram ini dapat menunjukkan hubungan tambahan seperti *extend* (perluasan) dan *include* (penyertaan), yang menunjukkan variasi atau kondisi khusus dalam menjalankan use case tertentu.

Secara umum, *Use Case Diagram* digunakan untuk menggambarkan apa yang dilakukan sistem dan siapa yang akan menggunakannya, memberikan gambaran tinggi tentang sistem tanpa memerlukan detail teknis. Diagram ini sangat berguna dalam tahap analisis sistem untuk memahami kebutuhan pengguna dan mendefinisikan fungsionalitas dasar yang harus dimiliki oleh sistem

BAB III

ISI

3. 1 Fitur Aplikasi

3.1.1. Login

Program *login* dalam aplikasi *Misslaundy* merupakan bagian penting yang berfungsi sebagai pintu awal autentikasi pengguna sebelum mengakses sistem manajemen *laundry*. Aplikasi ini dibuat menggunakan bahasa pemrograman Java dengan antarmuka grafis berbasis *Swing*. Ketika program dijalankan, pengguna diminta untuk memasukkan *username* dan *password* melalui *field* yang telah disediakan. Komponen GUI yang digunakan meliputi *TextField* untuk *input username*, *JPasswordField* untuk *input password* (agar karakter tidak terlihat), dan dua buah tombol yaitu LOGIN dan RESET. Tombol LOGIN akan memicu *proses verifikasi* dengan mencocokkan data *input* pengguna ke dalam *database* melalui *query SQL* terhadap tabel *tbl_login*. Jika *username* ditemukan dan *password* cocok, sistem akan menampilkan pesan "*Login Berhasil*" dan membuka halaman utama atau *Home* dari aplikasi. Namun, jika *username* tidak ditemukan, maka akan muncul pesan "*Username Salah*", dan jika *password* tidak sesuai, maka sistem akan memberikan peringatan "*Password Salah*". Selain tombol *login*, tombol RESET juga disediakan untuk menghapus isi dari kedua *field input*, memudahkan pengguna untuk mengulang input jika terjadi kesalahan.

Secara keseluruhan, login ini berperan penting dalam mengatur akses pengguna dan menjadi langkah awal sebelum pengguna dapat mengoperasikan seluruh *fitur* dari sistem *Misslaundy*.

3.1.2 Pesanan

Fitur pesanan ini berfungsi sebagai formulir dan manajemen *input* transaksi *laundry*, mulai dari data pelanggan, layanan yang dipilih, hingga harga dan penyimpanan datanya. *Fitur* pesanan yang ditampilkan mencakup alur pemesanan jasa laundry. Berikut adalah penjelasan dari fitur-fitur utama pada halaman tersebut:

1. Form Input Pemesanan

Terletak di bagian atas kiri dan kanan layar.

Input Kiri:

- a) No Antrian: Nomor urut pelanggan (dapat digunakan untuk identifikasi pemesanan).
- b) Jenis Pewangi (*Dropdown*): Pelanggan dapat memilih jenis pewangi (misalnya: Cair, Bubuk).
- c) Jumlah Pakaian (/ KG): Berat cucian yang ingin dicuci dalam satuan kilogram.

Input Kanan:

- a) Nama: Nama pelanggan.
- b) No. Telp: Nomor telepon pelanggan untuk keperluan kontak.
- c) Alamat: Alamat pelanggan.

2. Harga dan Perhitungan

- a) HITUNG: Tombol untuk menghitung harga berdasarkan jenis pewangi dan jumlah pakaian.
- b) HARGA : Menampilkan hasil perhitungan harga.
- c) SIMPAN: Menyimpan data pesanan ke dalam database atau list.
- d) HAPUS: (Tombol tidak aktif jika belum ada yang dipilih) digunakan untuk menghapus pesanan tertentu.
- e) RESET: Menghapus semua data input agar bisa mengisi ulang dari awal.
- f) KEMBALI: Kembali ke halaman sebelumnya atau dashboard utama.

3. Tabel Daftar Pesanan

- a) Tabel ini menampilkan daftar semua pesanan yang telah masuk.
- b) Kolom mencakup: Nomor, Nama, No. Telepon, Alamat, Jenis Pewangi, Jumlah, Harga, Tanggal.

4. *Pop-Up* Konfirmasi

Setelah menekan tombol "SIMPAN", muncul popup "Berhasil" sebagai notifikasi bahwa pesanan telah tersimpan.

3.1.3 Pembayaran

Fitur Ambil/SELESAI membantu proses akhir transaksi *laundry*, yaitu saat pelanggan mengambil cucian. Dengan sistem auto-fill, kalkulasi otomatis, dan notifikasi berhasil, fitur ini berguna untuk memastikan pencatatan yang rapi.

1. Form Pengambilan

Terletak di bagian atas halaman

Input Kiri:

- a) No Antrian: Nomor urut yang digunakan untuk mencari dan memanggil data transaksi pelanggan sebelumnya.
- b) Nama: Diisi otomatis setelah No Antrian dimasukkan.
- c) No Telepon: Nomor kontak pelanggan, juga terisi otomatis.
- d) Alamat: Alamat pelanggan (juga otomatis dari data transaksi sebelumnya).

Input Tengah:

- a) Jenis Pewangi: Jenis pewangi yang digunakan saat laundry (Cair, Bubuk, dll).
- b) Jumlah: Berat atau jumlah pakaian yang dilaundry (biasanya dalam kg).
- c) Harga: Harga yang dihitung berdasarkan jumlah x jenis pewangi.
- d) Tanggal: Tanggal transaksi pengambilan.

Input Kanan:

- a) Total: Total biaya yang harus dibayar.
- b) Bayar: Jumlah uang yang dibayarkan pelanggan.
- c) Kembalian: Sistem menghitung otomatis jika bayar > total.

2. Tombol Utama

- a) SELESAI: Tombol ini menyimpan data pengambilan ke sistem dan menandai bahwa pesanan telah diambil pelanggan.
 - a. Saat ditekan, sistem menampilkan pop-up “Berhasil” sebagai notifikasi.
 - b. Data pelanggan ditambahkan ke daftar transaksi selesai.
- b) KEMBALI: Digunakan untuk kembali ke halaman utama/dashboard.
- c) REPORT: Digunakan untuk mencetak atau melihat laporan semua transaksi yang telah selesai.

3. Tabel Riwayat Transaksi

Terdapat dua tabel:

- a. Tabel pengambilan hari ini menampilkan semua pesanan yang sudah diambil berdasarkan No Antrian, Nama, Jenis Pewangi, Jumlah, Harga, Tanggal, dll.

- b. Tabel riwayat lengkap mencakup semua transaksi yang tersimpan di sistem.

4. Auto-Fill Berdasarkan No Antrian

Saat No Antrian dimasukkan:

- a. Sistem secara otomatis mengisi data pelanggan dan rincian cucian.
- b. Memudahkan proses pengambilan tanpa input manual ulang.
- c. Mencegah kesalahan input dan mempercepat pelayanan.

3. 2 Konsep OOP di kode program

Object-Oriented Programming (OOP) adalah paradigma pemrograman yang mengorganisasi program berdasarkan objek. Dalam OOP, objek merupakan representasi dari entitas nyata atau konsep logis yang memiliki data (atribut) dan perilaku (metode). Paradigma ini memungkinkan pengembangan perangkat lunak yang modular, fleksibel, dan mudah dipelihara karena memfasilitasi prinsip-prinsip desain seperti enkapsulasi, abstraksi, pewarisan, dan polimorfisme. Pada aplikasi Java Laundry ini, konsep OOP diterapkan dengan cukup konsisten melalui struktur kelas yang terpisah dan memiliki tanggung jawab masing-masing. Empat kelas utama yang dianalisis adalah `cucian.java`, `home.java`, `login.java`, dan `pelanggan.java`.

1. Kelas `cucian.java`

Kelas `cucian` merupakan inti dari logika bisnis dalam aplikasi ini. Kelas ini bertanggung jawab untuk menyimpan dan memproses data terkait layanan laundry, seperti berat cucian (`berat`), harga per kilogram (`harga`), dan total biaya (`total`). Seluruh atribut tersebut dideklarasikan sebagai `private`, sehingga tidak dapat diakses secara langsung dari luar kelas. Sebagai gantinya, akses ke atribut-atribut tersebut difasilitasi melalui metode `getter` dan `setter`.

Prinsip enkapsulasi sangat dominan dalam kelas ini. Dengan membatasi akses langsung ke atribut dan mengontrolnya melalui metode publik, kelas ini menjaga integritas data dan mencegah potensi kesalahan yang muncul akibat manipulasi langsung. Selain itu, kelas ini menerapkan abstraksi dengan menyembunyikan detail perhitungan total biaya dari pengguna atau kelas lain. Perhitungan total dilakukan melalui metode `setTotal()`, yang mengalikan berat dengan harga, dan hasilnya disimpan

sebagai total. Hal ini memungkinkan pengguna kelas untuk cukup memanggil metode tersebut tanpa mengetahui cara kerja perhitungannya secara rinci.

Dari sisi desain, kelas ini juga mendukung prinsip single responsibility, karena hanya fokus pada satu hal: menangani data cucian. Dengan memisahkan tanggung jawab ini dari input pengguna atau tampilan antarmuka, program menjadi lebih modular dan mudah dikembangkan. Jika di masa depan metode perhitungan berubah (misalnya menambahkan diskon atau biaya tambahan), perubahan hanya perlu dilakukan pada kelas ini tanpa mengubah kelas lain.

2. Kelas home.java

Kelas home berfungsi sebagai halaman utama aplikasi setelah pengguna berhasil login. Kelas ini menampilkan beberapa tombol atau menu navigasi yang dapat mengarahkan pengguna ke fitur-fitur utama seperti pengolahan data pelanggan, proses cucian, dan logout. Secara struktur, kelas ini berkaitan erat dengan tampilan antarmuka pengguna (UI), dan tidak memuat logika bisnis secara langsung.

Penerapan OOP pada kelas ini terutama terlihat pada abstraksi dan pemisahan tanggung jawab. Kelas ini tidak menangani perhitungan atau penyimpanan data, melainkan hanya mengelola tampilan dan navigasi aplikasi. Hal ini sesuai dengan prinsip single responsibility, di mana satu kelas hanya memiliki satu alasan untuk berubah, yaitu jika ada perubahan pada tampilan atau layout aplikasi. Dengan pendekatan ini, kelas home dapat diubah atau diganti tanpa mempengaruhi bagian logika program yang lain, sehingga sistem menjadi lebih terstruktur dan fleksibel.

Selain itu, kelas ini juga menunjukkan bagaimana objek-objek lain (seperti objek dari kelas login atau cucian) dapat digunakan untuk mengakses data atau menjalankan metode. Ini mencerminkan interaksi antar objek, yang merupakan prinsip fundamental dalam OOP.

3. Kelas login.java

Kelas login menangani proses autentikasi pengguna, yaitu verifikasi apakah username dan password yang dimasukkan sesuai dengan data yang diizinkan untuk mengakses aplikasi. Di dalam kelas ini, logika yang digunakan mencakup pengambilan

input dari pengguna, validasi data, dan memberikan tanggapan terhadap hasil validasi tersebut.

Dari sudut pandang OOP, kelas ini mencerminkan prinsip abstraksi, di mana proses autentikasi disembunyikan dari kelas lain. Kelas ini juga menunjukkan modularitas, karena seluruh logika login dipisahkan dari kelas lain seperti home atau pelanggan. Artinya, apabila sistem login ingin diperluas, misalnya menambahkan peran pengguna atau enkripsi password, perubahan dapat difokuskan pada kelas ini saja tanpa mengganggu struktur kelas lain.

Kelas login juga berpotensi untuk dikembangkan lebih lanjut menggunakan konsep pewarisan. Misalnya, jika aplikasi ditingkatkan untuk mendukung peran pengguna seperti “admin” dan “kasir”, maka kelas login dasar dapat diwarisi oleh kelas-kelas turunan yang memiliki metode login khusus sesuai peran masing-masing.

4. Kelas pelanggan.java

Kelas pelanggan merepresentasikan data pelanggan secara lengkap, termasuk nama, alamat, dan nomor telepon. Atribut-atribut tersebut juga dideklarasikan sebagai private dan diakses melalui metode getter dan setter. Ini merupakan contoh penerapan enkapsulasi yang sangat baik, karena data pelanggan merupakan informasi sensitif yang tidak boleh diubah sembarangan.

Kelas ini juga menunjukkan penerapan abstraksi, karena hanya menangani data pelanggan tanpa mencampurkannya dengan logika tampilan atau perhitungan cucian. Kelas ini sangat ideal untuk digunakan dalam arsitektur Model-View-Controller (MVC), di mana kelas pelanggan bertindak sebagai model (penyimpan data), dan dapat berinteraksi dengan view (home) dan controller (login, MenuUtama, atau model) sesuai dengan alur program.

Selain itu, kelas ini memiliki tanggung jawab yang sangat spesifik, sehingga memudahkan pengembangan, pengujian, serta perawatan kode. Jika suatu saat dibutuhkan penambahan informasi seperti email atau status langganan, maka hal tersebut cukup ditambahkan di kelas ini tanpa mengubah kelas lainnya.

5. Kelas pembayaran.java

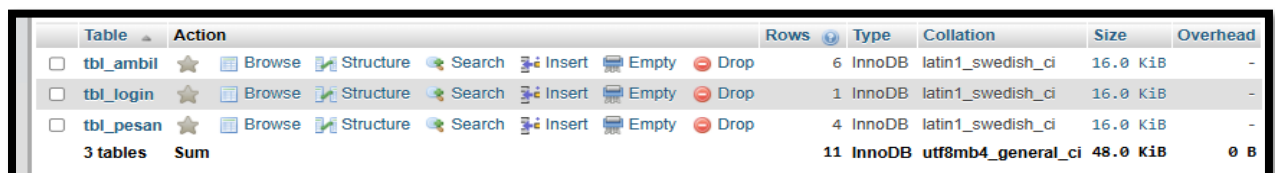
Kelas pembayaran memiliki fungsi utama dalam mengelola proses pembayaran atas layanan laundry yang dilakukan pelanggan. Dalam konteks aplikasi ini, kelas pembayaran berinteraksi dengan data cucian untuk menentukan total pembayaran, jumlah uang yang diberikan pelanggan, dan kembalian. Atribut-atribut seperti total, bayar, dan kembalian dideklarasikan secara private, yang menunjukkan penerapan prinsip enkapsulasi secara konsisten. Seperti pada kelas lain, akses terhadap atribut dilakukan menggunakan metode getter dan setter.

Kelas ini menampung metode utama seperti `setKembalian()`, yaitu perhitungan logika dasar yang mengurangi nilai bayar dari total untuk memperoleh kembalian. Penerapan abstraksi terlihat jelas karena perhitungan dilakukan di dalam metode tersebut dan pengguna kelas hanya perlu memanggil fungsi ini tanpa mengetahui detail implementasinya.

Selain itu, kelas pembayaran menjaga prinsip single responsibility, karena hanya bertanggung jawab atas satu aspek, yaitu proses pembayaran. Dengan begitu, pengelolaan data pelanggan, data cucian, dan tampilan UI dilakukan oleh kelas lain yang relevan. Jika terjadi perubahan logika bisnis, seperti penambahan fitur diskon atau pembayaran digital, perubahan hanya difokuskan pada kelas ini.

Dari perspektif OOP yang lebih lanjut, kelas ini juga berpotensi dikembangkan melalui pewarisan. Misalnya, bisa dibuat subkelas seperti `PembayaranTunai` atau `PembayaranDigital` yang masing-masing mewarisi atribut dan metode dasar dari pembayaran, namun memiliki metode baru atau logika tambahan sesuai kebutuhan jenis pembayaran.

3.3 Database



| Table | Action | Rows | Type | Collation | Size | Overhead |
|--------------------------------------|---|-----------|---------------|---------------------------|-----------------|------------|
| <input type="checkbox"/> tbl_ambil | ★ Browse Structure Search Insert Empty Drop | 6 | InnoDB | latin1_swedish_ci | 16.0 KiB | - |
| <input type="checkbox"/> tbl_login | ★ Browse Structure Search Insert Empty Drop | 1 | InnoDB | latin1_swedish_ci | 16.0 KiB | - |
| <input type="checkbox"/> tbl_pesanan | ★ Browse Structure Search Insert Empty Drop | 4 | InnoDB | latin1_swedish_ci | 16.0 KiB | - |
| 3 tables | Sum | 11 | InnoDB | utf8mb4_general_ci | 48.0 KiB | 0 B |

Gambar 3.3.1 Database db_laundry

Penjelasan :

Pada gambar 3.3.1 diatas menampilkan *database* dari db_laundry, yang dimana berisi tabel-tabel seperti tabel tbl_ambil, tbl_login, dan tbl_pesanan yang berisi data-data di dalamnya.

| no_antrian | nama | no_telp | alamat | j_pewangi | jumlah | harga | tanggal | total | bayar | kembali |
|------------|--------|---------------|-----------|-----------|--------|-------|------------|-------|--------|---------|
| 1 | dinday | 2310631250082 | telukgong | Sakura | 1 | 18000 | 2025-05-30 | 18000 | 20000 | 2000 |
| 2 | tika | 0202020202 | tambun | Bubuk | 2 | 23000 | 2025-05-30 | 23000 | 50000 | 27000 |
| 7 | okin | 10920393932 | pluit | Bubuk | 7 | 73000 | 2025-05-31 | 73000 | 100000 | 27000 |
| 8 | aqelaa | 198866567 | agz | Bubuk | 6 | 63000 | 2025-05-31 | 63000 | 200000 | 137000 |
| 10 | sisi | 0909090909 | krwng | Cair | 5 | 55000 | 2025-05-31 | 55000 | 100000 | 45000 |
| 12 | tikuy | 1010101010 | tambun | Cair | 2 | 25000 | 2025-06-01 | 25000 | 50000 | 25000 |

Gambar 3.3.2 Data tbl_ambil

Penjelasan :

Pada gambar 3.3.2 diatas menampilkan tabel tb_ambil tabel ini terhubung dengan tabel lain seperti tb_pesanan melalui relasi *foreign key*, dan merupakan tabel utama dalam membuat pesanan, status layanan, dan keuangan *laundry*.

| user | pass |
|--------|-------|
| dinduy | 12345 |

Gambar 3.3.3 Data tbl_login

Penjelasan :

Pada gambar 3.3.3 diatas menampilkan isi dari tabel tb_member dalam *database* db_laundryuas, yang berfungsi untuk menyimpan data pelanggan layanan *laundry*. Tabel ini memiliki beberapa atribut penting, yaitu id_member sebagai *primary key* untuk setiap pelanggan, nama yang mencatat nama pelanggan seperti dan alamat yang berisi informasi lokasi tempat tinggal pelanggan, jenis_kelamin menunjukkan jenis kelamin pelanggan, serta tlp yang menyimpan nomor telepon pelanggan.

| ← T → | | no_antrian | nama | no_telp | alamat | j_pewangi | jumlah | harga | tanggal |
|--------------------------|--|------------|---------|--------------|-----------|-----------|--------|-------|------------|
| <input type="checkbox"/> |  Edit  Copy  Delete | 1 | dinda | 0101010101 | Telukgong | cair | 5 | 55000 | 2025-05-30 |
| <input type="checkbox"/> |  Edit  Copy  Delete | 4 | siah | 0303030303 | Bekasi | Bubuk | 6 | 63000 | 2025-05-30 |
| <input type="checkbox"/> |  Edit  Copy  Delete | 5 | rizka | 0404040404 | cikarang | Cair | 1 | 15000 | 2025-05-30 |
| <input type="checkbox"/> |  Edit  Copy  Delete | 13 | Kartika | 085763451267 | Tambun | Cair | 2 | 25000 | 2025-06-08 |

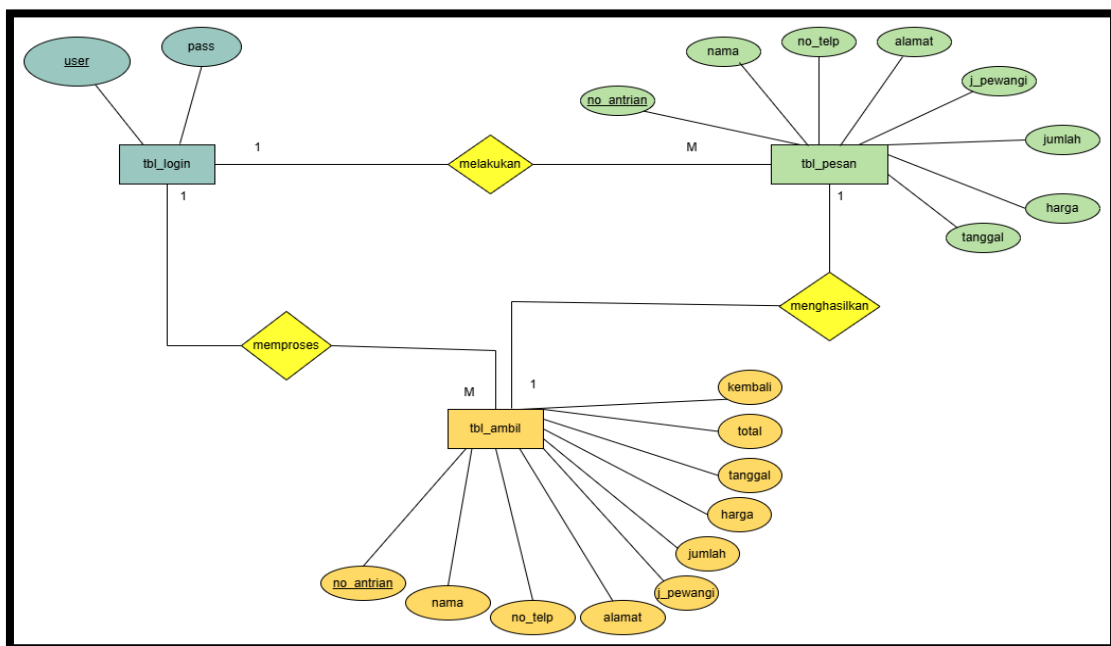
Gambar 3.3.4 Data Pesan

Penjelasan :

Pada gambar 3.3.4 di atas menampilkan isi dari tabel tb_paket Tabel ini memiliki beberapa atribut penting, yaitu id_paket *primary key* dari setiap paket layanan, jenis yang menunjukkan jenis pakaian atau barang yang dilaundry, nama_paket yang merupakan penamaan khusus dari paket layanan, serta harga yang mencantumkan tarif layanan untuk masing-masing paket. Atribut-atribut ini mengelompokkan dan mengatur harga berdasarkan jenis layanan yang dipilih oleh pelanggan, serta mempermudah proses transaksi dan laporan administrasi.

3.4 UML

1. Entity Relationship Diagram

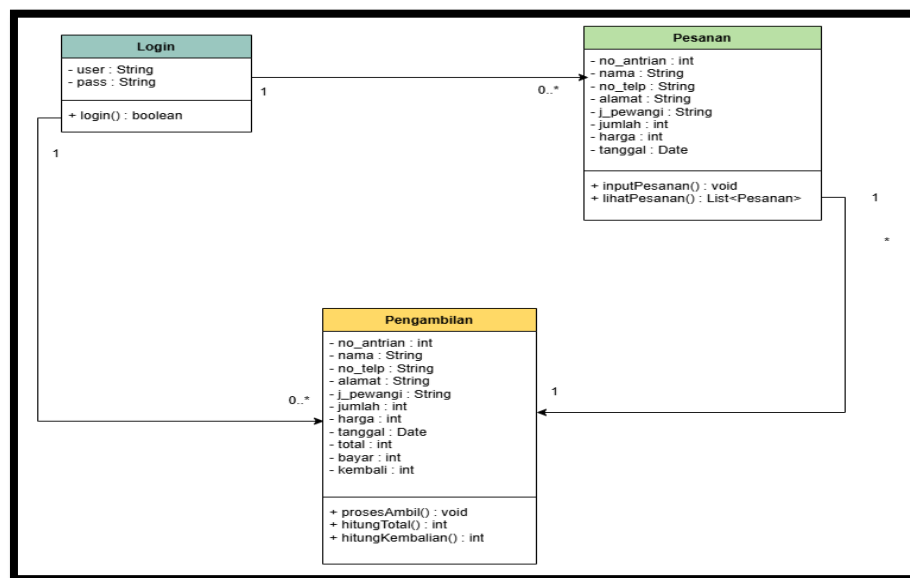


Gambar 3.4.1 Entity Relationship Diagram pada sistem Laundry

Penjelasan :

Pada gambar 3.4.1 *Entity Relationship Diagram (ERD)* yang menggambarkan hubungan antar tabel dalam basis data sistem *laundry*. Terdapat tiga entitas utama yaitu *tbl_login*, *tbl_pesan*, dan *tbl_ambil*. Entitas *tbl_login* memiliki atribut *user* dan *pass*, serta berelasi satu ke banyak (1:M) dengan tabel *tbl_pesan* melalui relasi "melakukan", yang berarti seorang pengguna dapat melakukan banyak pesanan. Selain itu, *tbl_login* juga berelasi ke *tbl_ambil* melalui relasi "memproses". Entitas *tbl_pesan* menyimpan informasi detail pesanan seperti *no_antrian*, *nama*, *no_telp*, *alamat*, *j_pewangi*, *jumlah*, *harga*, dan *tanggal*. Tabel ini berelasi satu ke banyak (1:M) dengan *tbl_ambil* melalui relasi "menghasilkan", di mana satu pesanan dapat menghasilkan banyak transaksi pengambilan. Entitas *tbl_ambil* mencatat informasi pengambilan pesanan, termasuk *no_antrian*, *nama*, *no_telp*, *alamat*, *j_pewangi*, *jumlah*, *harga*, *tanggal*, *total*, *bayar*, dan *kembali*.

2. Class Diagram



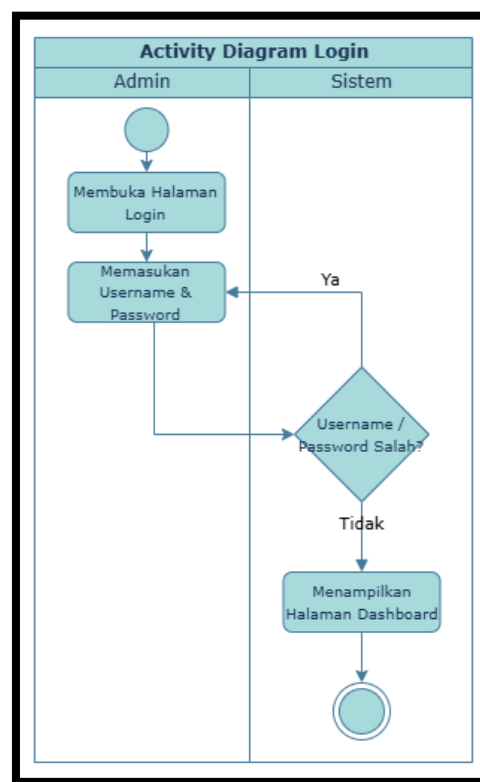
Gambar 3.4.2 *Class Diagram* pada sistem *Laundry*

Penjelasan :

Pada gambar 3.4.2 *Class diagram* pada sistem *laundry* menggambarkan struktur utama data dan hubungan antar objek dalam sistem. Terdapat tiga kelas utama, *class diagram* yang memodelkan struktur objek dalam sistem. Terdapat tiga kelas utama yaitu *Login*, *Pesanan*, dan *Pengambilan*. Kelas *Login* memiliki atribut *user* dan *pass*, serta method *login()* untuk memverifikasi pengguna. Kelas *Pesanan* menyimpan data pesanan dengan atribut seperti *no_antrian*, *nama*, *no_telp*, *alamat*, *j_pewangi*, *jumlah*, *harga*, dan *tanggal*. Kelas ini memiliki

method `inputPesanan()` untuk memasukkan data pesanan dan `lihatPesanan()` untuk menampilkan daftar pesanan. Kelas Pengambilan menyimpan informasi proses pengambilan dengan atribut yang serupa dengan kelas Pesanan, ditambah atribut keuangan seperti total, bayar, dan kembali. Kelas ini memiliki method `prosesAmbil()` untuk memproses pengambilan, `hitungTotal()` untuk menghitung total biaya, serta `hitungKembalian()` untuk menghitung uang kembalian. Relasi antar kelas menunjukkan bahwa satu Login dapat memiliki banyak Pesanan dan banyak Pengambilan, sementara satu Pesanan dapat dihubungkan dengan banyak Pengambilan. Relasi antar kelas menunjukkan bahwa satu pelanggan dapat membuat banyak pesanan dan memproses banyak pengambilan. Setiap pesanan juga dapat dihubungkan dengan satu proses pengambilan. Dengan struktur ini, sistem mampu mengelola data laundry secara terorganisir dan efisien.

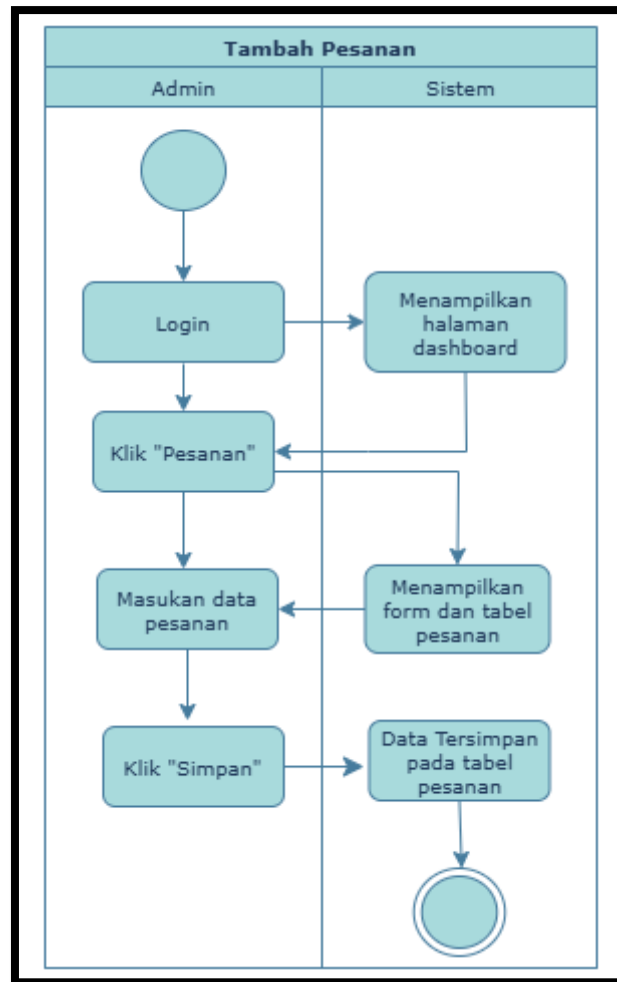
3. Activity Diagram



Gambar 3.3.3 Activity Diagram Login pada sistem *Laudry*

Penjelasan :

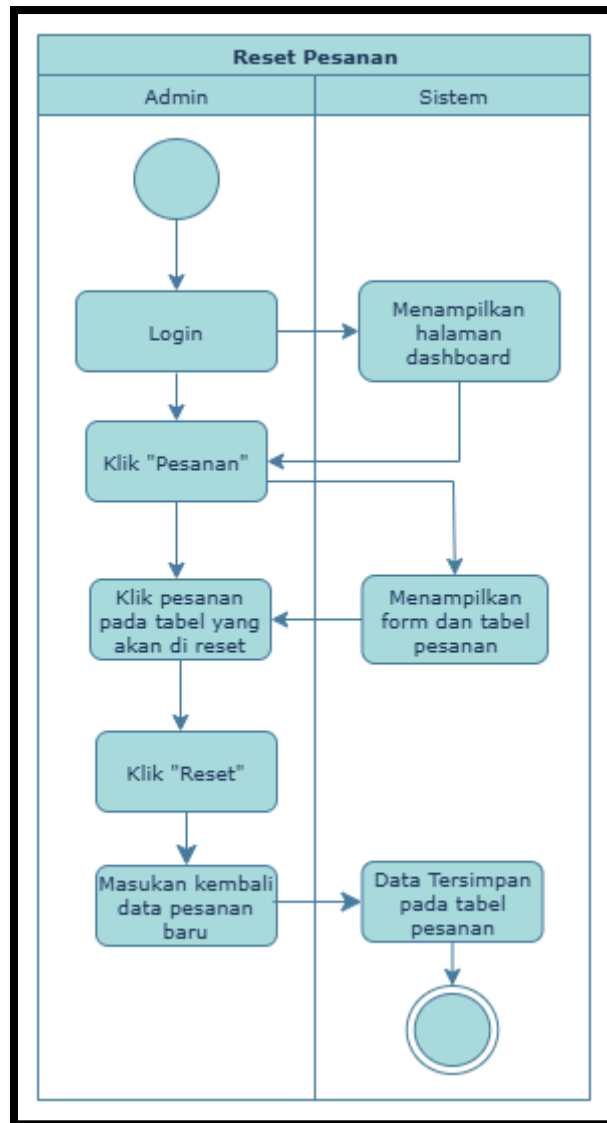
Pada gambar 3.3.3 *Activity Diagram Login*, proses diawali oleh admin yang membuka halaman login dan kemudian memasukkan username dan password. Sistem kemudian memverifikasi data yang dimasukkan. Jika username atau password salah, sistem akan meminta admin untuk memasukkan ulang data tersebut. Jika data benar, maka sistem akan menampilkan halaman dashboard yang menjadi halaman utama setelah berhasil login.



Gambar 3.3.4 *Activity Diagram Tambah Pesanan* pada sistem *Laudry*

Penjelasan :

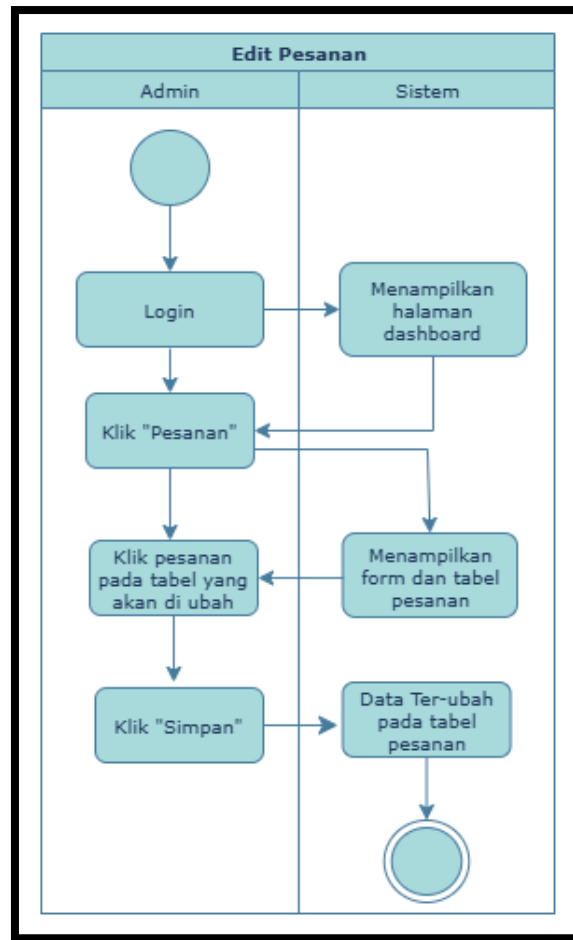
Pada gambar 3.3.4 *Activity Diagram Tambah Pesanan*, proses dimulai setelah admin melakukan login ke dalam sistem. Setelah berhasil masuk ke halaman dashboard, admin akan memilih menu "Pesanan". Sistem kemudian menampilkan form dan tabel pesanan. Admin dapat mengisi form dengan data pesanan yang ingin ditambahkan, lalu menekan tombol "Simpan". Sistem akan memproses penyimpanan data dan menampilkan data yang baru saja ditambahkan ke dalam tabel pesanan.



Gambar 3.3.5 Activity Diagram Edit Pesanan pada sistem *Laudry*

Penjelasan :

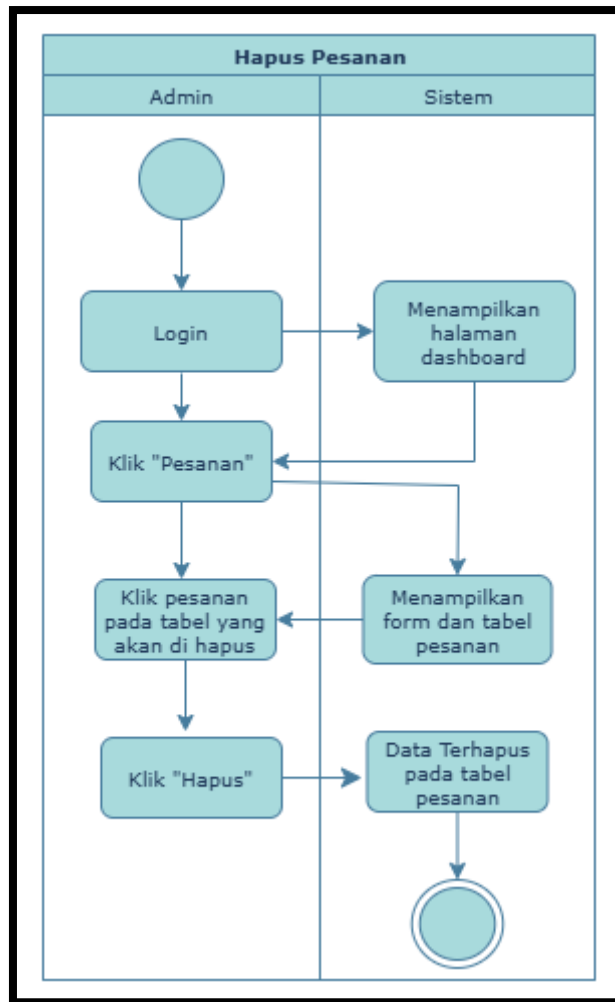
Pada gambar 3.3.5 Activity Diagram Edit Pesanan, prosesnya juga diawali dengan admin yang login ke dalam sistem hingga berhasil masuk ke halaman dashboard. Setelah itu, admin akan memilih menu "Pesanan" sehingga sistem menampilkan form dan tabel pesanan. Admin kemudian memilih salah satu data pesanan yang ingin diubah dari tabel tersebut. Setelah data yang dipilih ditampilkan dalam form, admin dapat mengedit informasi yang diperlukan dan kemudian menekan tombol "Simpan". Sistem kemudian akan memperbarui data yang telah diubah dan menampilkannya kembali di tabel pesanan.



Gambar 3.3.6 *Activity Diagram* Reset Pesanan pada sistem *Laudry*

Penjelasan :

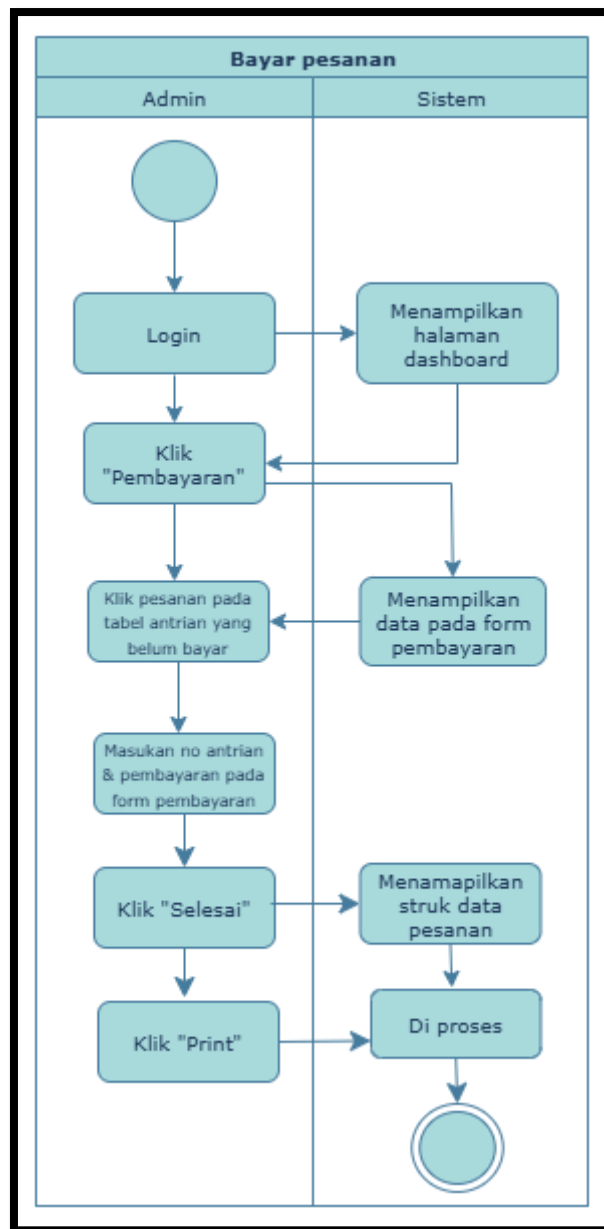
Pada gambar 3.3.6 *Activity Diagram* Reset Pesanan, admin memulai dengan login ke sistem. Setelah berhasil masuk ke halaman dashboard, admin memilih menu "Pesanan". Sistem kemudian menampilkan form dan tabel pesanan. Admin memilih data pesanan yang ingin di-reset dan menekan tombol "Reset". Setelah itu, admin dapat memasukkan kembali data pesanan baru, dan sistem akan menyimpan data yang diperbarui ke dalam tabel pesanan.



Gambar 3.3.7 Activity Diagram Hapus Pesanan pada sistem *Laudry*

Penjelasan :

Pada gambar 3.3.7 Activity Diagram Hapus Pesanan, admin login ke sistem, Setelah berhasil masuk ke halaman dashboard, memilih menu "Pesanan", kemudian memilih data pesanan yang ingin dihapus dari tabel. Setelah menekan tombol "Hapus", sistem akan menghapus data pesanan tersebut dari tabel.

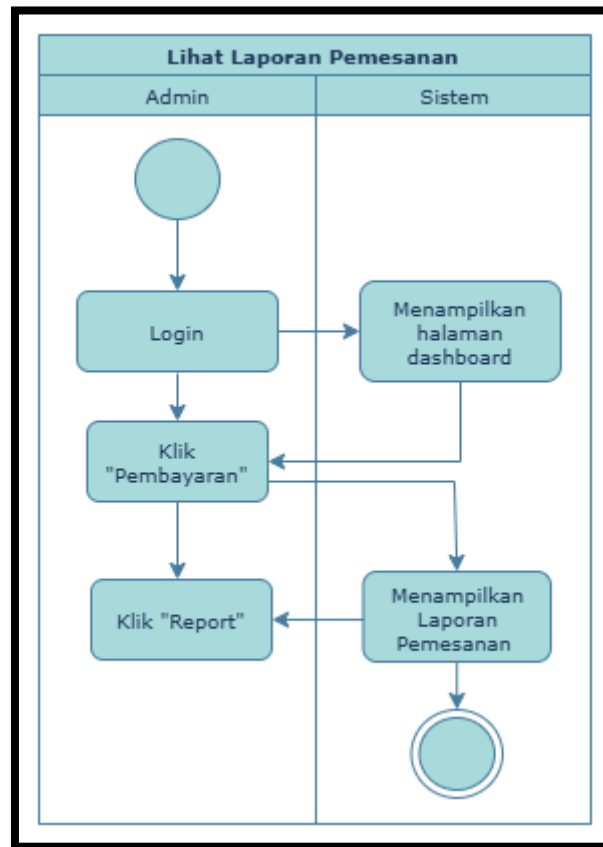


Gambar 3.3.8 Activity Diagram Bayar Pesanan pada sistem *Laudry*

Penjelasan :

Pada gambar 3.3.8 Activity Diagram Bayar Pesanan, dimulai dari admin yang login ke dalam sistem dan masuk ke halaman dashboard. Admin kemudian memilih menu "Pembayaran", dan sistem menampilkan tabel antrian pesanan yang belum dibayar. Admin memilih salah satu pesanan yang ingin diproses pembayarannya, lalu sistem menampilkan data pesanan tersebut dalam form pembayaran. Admin kemudian memasukkan nomor antrian dan jumlah pembayaran pada form yang tersedia. Setelah itu, admin mengklik tombol "Selesai", dan sistem akan menampilkan struk data pesanan sebagai bukti pembayaran. Terakhir, admin

dapat mengklik tombol "Print" untuk mencetak struk tersebut, sementara sistem memproses data pembayaran dan menyimpannya ke dalam sistem.

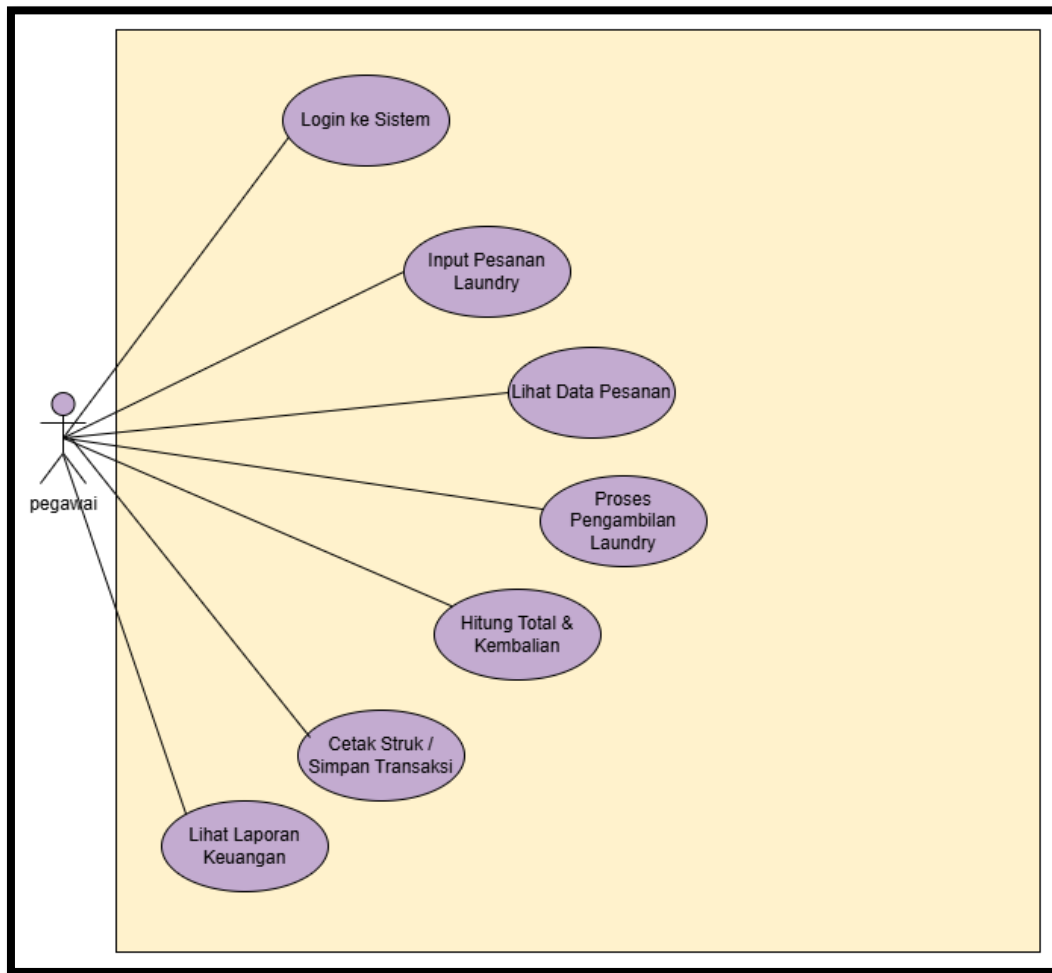


Gambar 3.3.9 Activity Diagram Lihat Laporan Pemesanan pada sistem *Laudry*

Penjelasan :

Pada gambar 3.3.9 Activity Diagram Lihat Laporan Pemesanan, diawali dengan admin yang melakukan login ke dalam sistem. Setelah berhasil masuk ke halaman dashboard, admin memilih menu "Pembayaran". Sistem kemudian menampilkan halaman pembayaran. Selanjutnya, admin menekan tombol "Report", dan sistem akan menampilkan laporan pemesanan yang berisi data pesanan.

4. Use Case Diagram



Gambar 3.4.1 *Use Case Diagram* pada sistem *Laundry*

Penjelasan :

Pada gambar 3.4.1 *Use Case Diagram* yang menunjukkan fungsi-fungsi utama yang dapat dilakukan oleh aktor dalam sistem, dalam hal ini pegawai. Pegawai dapat melakukan berbagai aktivitas, dimulai dari login ke sistem, input pesanan laundry, melihat data pesanan, memproses pengambilan laundry, menghitung total biaya dan kembalian, mencetak struk atau menyimpan transaksi, hingga melihat laporan keuangan. Diagram ini menggambarkan cakupan fungsi yang disediakan oleh sistem dan interaksi utama yang dapat dilakukan oleh pengguna.

3.5 Implementasi kode Program dan Pengujian



Gambar 3.5.1 Pengujian Kode Program Masukan *Username* dan *Password* dengan kondisi benar

Penjelasan :

Kelas *login* merupakan subclass dari *JFrame*, ini adalah jendela utama (frame) dari aplikasi *login*. *st*, *rs* digunakan untuk menjalankan *query* ke *database*. *con* menyimpan koneksi ke *database*, diambil dari class *koneksi.koneksiDB*. *Constructor* yang memanggil metode *initComponents()* untuk menyiapkan semua elemen GUI.

```
if (rs.next()) {  
    if (pass.getText().equals(rs.getString("pass"))) {  
        JOptionPane.showMessageDialog(rootPane, "Login Berhasil");  
        new home().setVisible(true);  
        this.dispose(); // Menutup form login jika diperlukan  
    } else {  
        JOptionPane.showMessageDialog(rootPane, "Password salah");  
    }  
} else {  
    JOptionPane.showMessageDialog(rootPane, "Username tidak ditemukan");  
}
```



```
}
```

kode program ini adalah tampilan kondisi cocok, tampil pesan "Login Berhasil" dan buka form baru home yang akan menampilkan kondisi seperti gambar 3.5.1.1 dengan kondisi benar.



Gambar 3.5.2 Masukan *Password* dengan kondisi yang salah

Penjelasan :

Pada gambar 3.5.2 dengan code program

```
} else {  
    JOptionPane.showMessageDialog(rootPane, "Password salah");  
    pass.setText("");  
    pass.requestFocus();  
}
```

adalah pengujian pada kondisi ketika salah memasukan *password* yang akan menampilkan *pop-up* "Password Salah"



Gambar 3.5.3 Masukan *Username* dengan kondisi yang salah

Penjelasan :

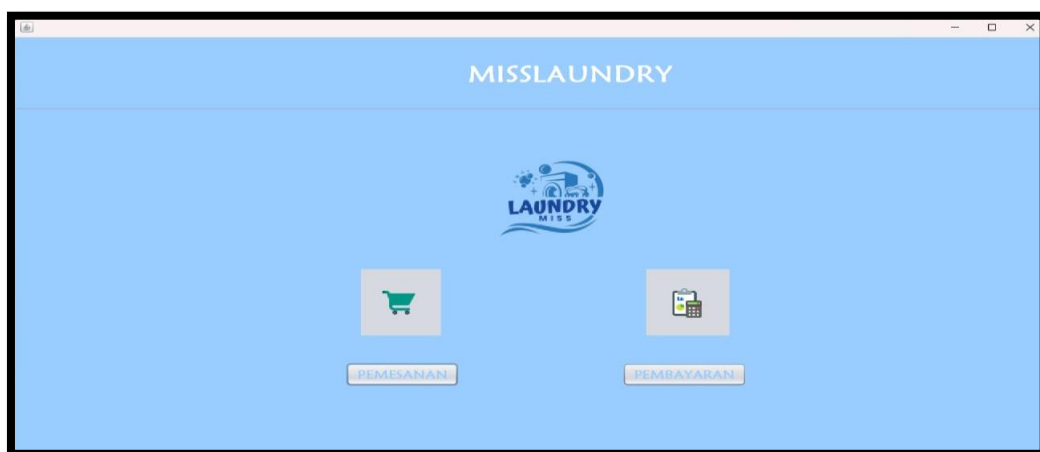
Pada gambar 3.5.3 dengan code program

```

} else {
    JOptionPane.showMessageDialog(rootPane, "Username salah");
    user.setText("");
    user.requestFocus();
}

```

adalah pengujian pada kondisi ketika salah memasukan *username* yang akan menampilkan *pop-up* "Username Salah"



Gambar 3.5.4 Menampilkan menu pesanan dan pembayaran

Penjelasan :

Pada gambar 3.5.4 Saat objek *home* dibuat, metode `initComponents()` akan dipanggil untuk membangun antarmuka grafis (GUI). `public home() { initComponents(); }` Metode ini otomatis dihasilkan oleh GUI builder . Kode program ini akan menampilkan halaman pesanan

```
private void pesanActionPerformed(java.awt.event.ActionEvent evt) {  
    pelanggan a2 = new pelanggan();  
    a2.setVisible(true);  
    this.setVisible(false);  
}
```

Dan ini akan menjalankan menu pembayaran,

```
private void ambilActionPerformed(java.awt.event.ActionEvent evt) {  
    pembayaran a3 = null;  
    try {  
        a3 = new pembayaran();  
    } catch (SQLException ex) {  
        Logger.getLogger(home.class.getName()).log(Level.SEVERE, null, ex);  
    }  
  
    if (a3 != null) {  
        a3.setVisible(true);  
        this.setVisible(false);  
    }  
}
```

dari masing-masing kode program tersebut akan masuk ke dalam *fitur* sesuai dengan yang di “*klik*” dan juga akan menyembunyikan *form home*.

| Nomor... | Nama | Nomor... | Alamat | Jenis... | Jumlah | Harga | Tanggal |
|----------|-------|-----------|----------|----------|--------|-------|-----------|
| 5 | rizka | 040404... | cikarang | Cair | 1 | 15000 | 2025-0... |
| 13 | tika | 089716... | Bekasi | Cair | 7 | 75000 | 2025-0... |
| 15 | sukma | 098765... | Bekasi | Cair | 6 | 65000 | 2025-0... |

Gambar 3.5.5 Menambahkan pesanan dan data pelanggan

Penjelasan :

Pada gambar 3.5.5 Form: "PESAN DISINI" Form ini terintegrasi langsung dengan backend Java dan basis data MySQL melalui berbagai komponen GUI yang dijalankan oleh kode program.

1. No Antrian (no_transaksi)

Field ini menampilkan nomor urut pelanggan secara otomatis setiap kali form dibuka. Nilainya diambil dari database menggunakan fungsi `get_antrian()`, yang membaca nomor antrian terbesar saat ini (`max(no_antrian)`) dari tabel `tbl_pesanan`, lalu menambahkannya untuk antrian baru.

2. Jenis Pewangi (j_pewangi)

Komponen `ComboBox` ini berisi pilihan jenis pewangi yang dapat dipilih oleh pelanggan, yaitu "Pilih", "Cair", dan "Bubuk". Pemilihan jenis pewangi akan memengaruhi harga total laundry, karena masing-masing pewangi memiliki harga tambahan tersendiri (misalnya: Cair = Rp5.000, Bubuk = Rp3.000). Pilihan ini diproses di bagian kode `hitungActionPerformed()` saat tombol "HITUNG" ditekan.

3. Jumlah Pakaian (jumlah)

Field ini diisi oleh pelanggan untuk menyatakan berapa banyak cucian yang ingin mereka laundry, dalam satuan kilogram (KG). Nilai ini juga akan dikalikan dengan tarif dasar (Rp10.000 per kg) pada proses perhitungan total harga.

4. Nama (nama)

Field teks ini digunakan untuk menginputkan nama pelanggan. Data ini penting sebagai identifikasi utama selain nomor antrian dan akan disimpan ke dalam *database* saat pesanan dikirim.

5. No. Telp (no_telp)

Digunakan untuk mencatat nomor telepon pelanggan. Informasi ini berguna untuk komunikasi jika pihak laundry perlu menghubungi pelanggan terkait pesanan mereka.

6. Alamat (alamat)

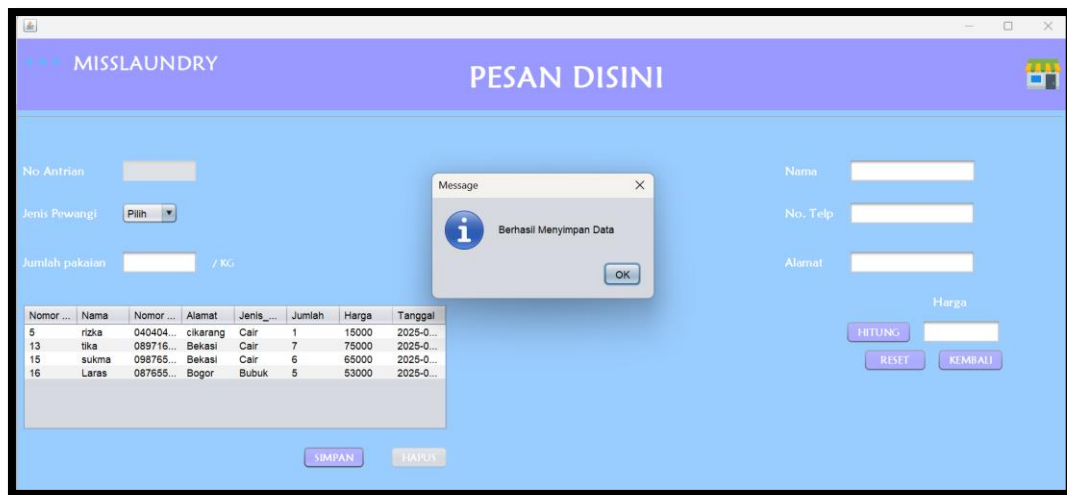
Field ini berfungsi untuk mencatat alamat pelanggan.

7. Harga (tottal)

Harga akhir dicantumkan pada field ini dan dihitung secara otomatis ketika tombol "HITUNG" ditekan. Harga dihitung dengan rumus: $\text{Total} = (\text{Jumlah pakaian} \times 10.000) + \text{Harga Pewangi}$. Misalnya, jika pelanggan mencuci 5 KG pakaian dengan pewangi bubuk, maka harga total adalah: $(5 \times 10.000) + 3.000 = 53.000$ dengan code program:

```
private void hitungActionPerformed(java.awt.event.ActionEvent evt) {  
    int harga_pewangi = 0;  
    if (j_pewangi.getSelectedIndex() == 1) {  
        harga_pewangi = 5000;  
    } else if (j_pewangi.getSelectedIndex() == 2) {  
        harga_pewangi = 3000;  
    }  
    int berat = Integer.parseInt(jumlah.getText());  
    int total = (berat * 10000) + harga_pewangi;  
    tottal.setText(String.valueOf(total));  
}
```

Seluruh data yang telah diisi akan disimpan ke dalam tabel *tbl_pesanan* di database ketika pelanggan menekan tombol "SIMPAN", dan ditampilkan dalam tabel bawah form untuk monitoring antrian aktif.



Gambar 3.5.6 Mengecek data yang di tambahkan berhasil

Penjelasan :

Pada gambar 3.5.6 ketika tombol SIMPAN ditekan String sql = "no_antrian=NULL,nama='"+nama.getText()+"', ... "; model.simpan("tbl_pesanan", sql); data akan disimpan ke tabel tbl_pesanan. pesan akan muncul “Berhasil Menyimpan Data” Tabel Daftar Pemesanan tbl_beli akan menampilkan data dari tbl_pesanan java

Reset & Kembali Tombol RESET → memanggil method reset() untuk mengosongkan semua input. Tombol KEMBALI → kembali ke halaman home(): java SalinEdit new home().setVisible(true); dispose(); dari fitur pesanan kesimpulan alur :

1. User isi data pemesanan.
2. Tekan HITUNG → total harga dihitung.
3. Tekan SIMPAN → data disimpan ke tabel tbl_pesanan.
4. Tabel di bawah menampilkan pesanan aktif.
5. Lanjut ke halaman AMBIL DISINI untuk menyelesaikan pembayaran.

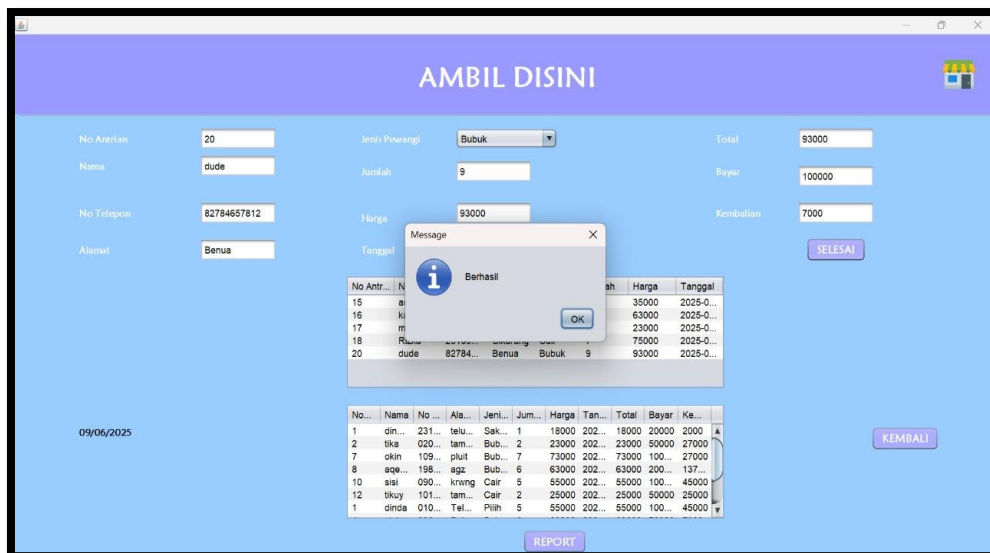
| No Antrian | Nama | No Tel | Alamat | Jenis P | Jumlah | Harga | Tanggal |
|------------|---------|----------|----------|---------|--------|-------|-----------|
| 15 | adinda | 23105 | Jekut | Calir | 3 | 35000 | 2025-0... |
| 16 | karlika | 23105 | Tambun | Bubuk | 6 | 63000 | 2025-0... |
| 17 | mansiah | 23105 | Bekasi | Bubuk | 2 | 23000 | 2025-0... |
| 18 | Rizka | 23105 | Cikarang | Calir | 7 | 75000 | 2025-0... |
| 20 | dude | 82784... | Benua | Bubuk | 9 | 93000 | 2025-0... |

| No | Nama | No | Ala | Jenl | Jum | Harga | Tan | Total | Bayar | Ke |
|----|--------|--------|---------|--------|-----|-------|--------|-------|--------|--------|
| 1 | din... | 231... | telu... | Sak... | 1 | 18000 | 202... | 18000 | 20000 | 2000 |
| 2 | tika | 020... | tam... | Bub... | 2 | 23000 | 202... | 23000 | 50000 | 27000 |
| 7 | okin | 109... | pluit | Bub... | 7 | 73000 | 202... | 73000 | 100... | 27000 |
| 8 | aqe... | 198... | agt | Bub... | 6 | 63000 | 202... | 63000 | 200... | 137... |
| 10 | slal | 090... | kneng | Calir | 6 | 65000 | 202... | 65000 | 100... | 45000 |
| 12 | sluy | 101... | tam... | Calir | 2 | 25000 | 202... | 25000 | 50000 | 25000 |
| 1 | dinda | 010... | Tel... | Pilih | 5 | 55000 | 202... | 55000 | 100... | 45000 |

Gambar 3.5.7 Melakukan pembayaran dengan nomer antrian

Penjelasan :

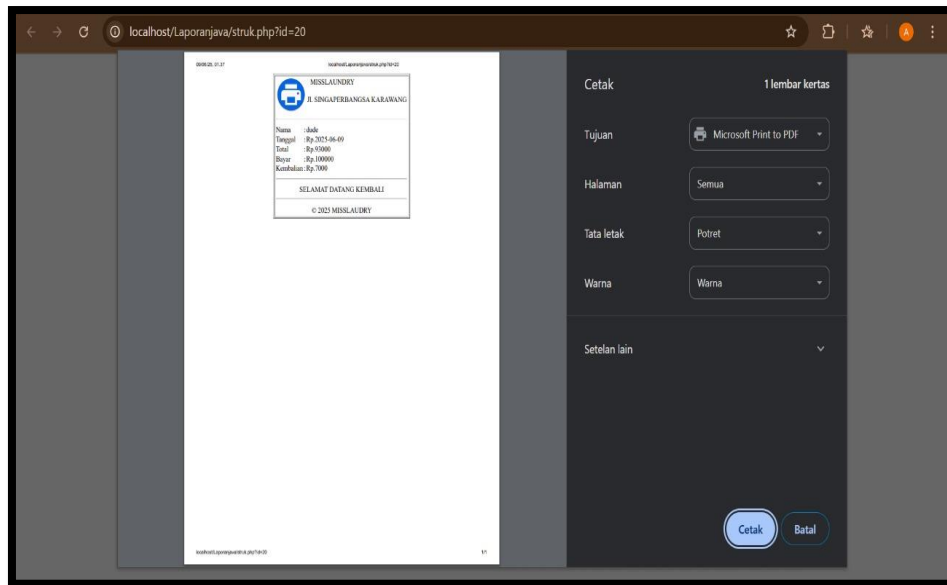
Pada gambar 3.5.7 *input* antrian form "PESAN DISINI". Pelanggan memasukkan no antrian, data pribadi, jenis pewangi, dan jumlah pakaian. Setelah menekan tombol SIMPAN, data akan disimpan ke tabel `tbl_pesanan` di database MySQL. Data ini ditampilkan dalam tabel1 pada form AMBIL DISINI. Ambil Pesanan form "AMBIL DISINI" yang menggunakan No Antrian sebagai *primary key* untuk mencari data pesanan pelanggan. Ketika pengguna mengetik nomor antrian (`no_transaksi`), event *listener* ini aktif. Transaksi Pembayaran pada kolom bayar dan kembalian diisi secara otomatis Saat tombol "SELESAI" ditekan data akan dipindahkan dari tabel `tbl_pesanan` ke `tbl_ambil`. data `no_antrian` digunakan untuk menghapus dari `tbl_pesanan` dan disimpan ke `tbl_ambil`. Dengan *code* program (`model.hapus("tbl_pesanan", "no_antrian", no_transaksi.getText())`) {if (`model.simpan("tbl_ambil", sql)`) {`tampildata()`; // refresh tabel1 `tampildata1()`; // refresh tabel2} yang akan membantu memindahkan *table* `tbl_pesanan` ke `tbl_ambil`.



Gambar 3.5.8 Melakukan penyelesaian pembayaran dan cetak *invoice*

Penjelasan :

Pada gambar 3.5.8 ketika *user* menekan Tombol SELESAI prosesnya cek apakah bayar cukup ($b \geq \text{total}$). Hapus dari tabel `tbl_pesanan` berdasarkan `no_antrian`. Simpan data ke tabel `tbl_ambil` `tbl_ambilStringsSql="no_antrian='"+no_transaksi.getText()+"',nama='"+nama.getText()+"',"; model.simpan("tbl_ambil", sql);` Tampilkan pesan "Berhasil" menggunakan `java JOptionPane.showMessageDialog(null, "Berhasil");` Setelah pesan "Berhasil", akan langsung mencetak *invoice* via *browser* aplikasi mencoba membuka URL: `http://localhost/Laporanjava/struk.php?id=20` lewat `javaSalinEditDesktop.getDesktop().browse(newURL("http://localhost/Laporanjava/struk.php?id="+ no_transaksi.getText()).toURI());` Selanjutnya tabel1 (daftar pesanan) di-*refresh*: data no antrian 20 dihapus dari sini. tabel2 (pesanan selesai) di-*refresh*: data no antrian 20 muncul di bawah ,yang akan otomatis terupdate



Gambar 3.5.9 Cetak *Invoice*

Penjelasan :

Pada gambar 3.5.9 adalah tampilan ketikan invoice berhasil di cetak dan menghasilkan input seperti gambar 3.5.9 yang padat di akses lewat mencetak *browser* URL: `http://localhost/Laporanjava/struk.php?id=20` lewat: `javaSalinEditDesktop.getDesktop().browse(newURL("http://localhost/Laporanjava/struk.php?id="+ no_transaksi.getText()).toURI());` dari fitur pembayaran kesimpulan alur :

1. Pelanggan pesan → Data masuk ke `tbl_pesanan`.
2. Operator cari berdasarkan No Antrian.
3. Operator isi data pembayaran → klik SELESAI.
4. Data dipindah ke `tbl_ambil` → pesanan dianggap selesai.
5. Bisa dicetak struk atau report (fitur REPORT membuka `struk.php / laporan`)