

**LAPORAN UJIAN AKHIR SEMESTER  
PEMROGRAMAN BERORIENTASI OBJEK  
“SmartPark”**

Dosen Pengampu: Taufik Ridwan, S.T., M.T



**Disusun Oleh:**

Kelompok 6

Dhafa Fikriansyah Putra (2310631250012)

M. Dwiki Husni Farhan (2310631250064)

Yogi Cahyono (2310631250039)

Zalfa Astiad Ryada (2310631250080)

**KELAS 4B  
PROGRAM STUDI SISTEM INFORMASI  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS SINGAPERBANGSA KARAWANG  
TAHUN 2025**

## **KATA PENGANTAR**

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan laporan proyek akhir yang berjudul " SmartPark " ini dengan baik dan tepat waktu. Laporan ini disusun dalam rangka memenuhi tugas akhir pada mata kuliah Pemrograman Berorientasi Objek, Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang.

Penyusunan laporan ini tidak terlepas dari dukungan, bantuan, dan bimbingan dari berbagai pihak. Oleh karena itu, kami mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Taufik Ridwan, S.T., M.T, selaku dosen pengampu mata kuliah, yang telah membimbing dan memberikan arahan selama proses pelaksanaan proyek ini.
2. Seluruh dosen dan staf pengajar di Fakultas Ilmu Komputer UNSIKA atas ilmu dan wawasan yang telah diberikan.
3. Teman-teman seperjuangan dalam satu kelompok dan kelas atas kerjasama, dukungan, serta motivasi yang sangat berarti.

Penulis menyadari bahwa masih terdapat beberapa kekurangan dan keterbatasan dalam penyusunan laporan ini. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun dari para pembaca demi penyempurnaan laporan ini.

Akhir kata penulis berharap semoga laporan ini dapat bermanfaat bagi dirinya dan pembaca sekalian. Terima kasih banyak.

Karawang, 28 Mei 2025

Kelompok 6

## DAFTAR ISI

|  |     |
|--|-----|
| KATA PENGANTAR.....  | .ii |
| DAFTAR ISI.....  | iii |
| BAB I PENDAHULUAN .....                                      | .2  |
| 1.1    Latar Belakang .....                                  | .2  |
| 1.2    Rumusan Masalah .....                                 | .3  |
| 1.3    Tujuan.....   | .3  |
| BAB II PEMBAHASAN .....                                      | .4  |
| 2.1    Deskripsi .....                                       | .4  |
| 2.2    Fitur-Fitur .....                                     | .4  |
| 2.3    Konsep OOP dalam Kode Program .....                   | .6  |
| 2.4    Perancangan UML ( Unifield Modelling Language ) ..... | .9  |
| 2.4.1    Class Diagram .....                                 | .9  |
| 2.4.2    Use Case Diagram .....                              | .11 |
| 2.4.3    Activity Diagram .....                              | .12 |
| 2.4.4    Sequence Diagram.....                               | .19 |
| 2.5    Perancangan Database .....                            | .26 |
| 2.6    Implementasi Kode Program dan Pengujian .....         | .29 |
| BAB III PENUTUP .....  | .79 |
| 3.1    Kesimpulan .....                                      | .79 |

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Seiring berkembangnya zaman dan kemajuan teknologi informasi, kebutuhan akan sistem otomatis dalam berbagai aspek kehidupan semakin meningkat. Salah satu penerapan teknologi tersebut adalah dalam pengelolaan parkir kendaraan. Sistem parkir manual yang masih banyak digunakan saat ini sering kali menimbulkan berbagai permasalahan, seperti kesalahan pencatatan, antrean panjang, dan kebingungan dalam perhitungan tarif parkir. Kondisi ini tidak hanya menurunkan efisiensi kerja petugas parkir, tetapi juga mengurangi kepuasan pengguna layanan parkir.

Untuk menjawab permasalahan tersebut, SmartPark hadir sebagai solusi yang efektif. Sistem ini mampu mencatat data kendaraan secara real-time, menghitung tarif berdasarkan waktu masuk dan keluar secara otomatis, serta menyediakan laporan transaksi yang dapat digunakan sebagai data analitik dan audit. Selain itu, dengan antarmuka pengguna berbasis grafis (GUI), sistem ini dapat digunakan dengan mudah oleh petugas tanpa perlu keahlian teknis yang tinggi.

Universitas Singaperbangsa Karawang, sebagai institusi pendidikan yang mendukung penerapan teknologi informasi dalam kehidupan kampus, sangat cocok menjadi lokasi implementasi SmartPark. Dengan meningkatnya jumlah kendaraan mahasiswa, dosen, dan staf setiap tahunnya, sistem parkir yang efisien dan terintegrasi menjadi kebutuhan yang mendesak. Penerapan sistem ini tidak hanya akan memperlancar arus kendaraan dan meminimalkan human error, tetapi juga memberikan pengalaman digitalisasi nyata bagi civitas akademika.

Untuk itu, tim kami berinisiatif merancang dan mengembangkan SmartPark berbasis aplikasi desktop menggunakan bahasa pemrograman Java dan arsitektur MVC (Model-View-Controller). Dengan penambahan pola DAO (Data Access Object), pengelolaan data menjadi lebih terstruktur dan modular. Proyek ini juga dilengkapi dengan database relasional, perhitungan tarif otomatis, serta antarmuka pengguna yang intuitif.

## **1.2 Rumusan Masalah**

1. Bagaimana merancang sistem parkir berbasis Java yang modular, efisien, dan mudah dikembangkan?
2. Bagaimana sistem dapat mencatat transaksi masuk dan keluar kendaraan serta menghitung tarif otomatis?

## **1.3 Tujuan**

1. Membangun sistem parkir dengan pemisahan komponen MVC.
2. Menyediakan fitur login, entry parkir, exit parkir, laporan, dan perhitungan tarif otomatis.

## **BAB II**

### **PEMBAHASAN**

#### **2.1 Deskripsi**

SmartPark adalah aplikasi manajemen parkir berbasis desktop yang dibangun menggunakan Java Swing. Aplikasi ini menyediakan fitur-fitur utama seperti pencatatan kendaraan masuk dan keluar, penghitungan tarif otomatis, pengelolaan data user, laporan transaksi parkir, serta ekspor laporan ke format Excel dan cetak struk keluar kendaraan dalam bentuk PDF.

#### **2.2 Fitur-Fitur**

SmartPark ini memiliki sejumlah fitur utama yang dirancang untuk mendukung pengelolaan kendaraan yang efisien, cepat, dan minim kesalahan. Fitur-fitur ini dikembangkan dengan mempertimbangkan kebutuhan pengguna (user experience), efisiensi kerja petugas parkir, serta kemudahan dalam pengelolaan data transaksi. Berikut adalah penjelasan fitur-fitur utama dalam sistem:

1. Kendaraan Masuk & Keluar: Petugas parkir dapat mencatat kendaraan yang masuk dengan menginput nomor plat dan memilih jenis kendaraan. Sistem secara otomatis mencatat waktu masuk. Saat kendaraan keluar, petugas cukup menginput kembali plat nomor yang sama. Sistem akan mencocokkan data entry sebelumnya dan mencatat waktu keluar.
2. Penghitungan Tarif Otomatis: Setelah waktu keluar dimasukkan, sistem secara otomatis menghitung tarif parkir berdasarkan durasi parkir dan jenis kendaraan. Perhitungan dilakukan dengan memanfaatkan ParkingCalculator, yang membaca tarif per jam dari tabel parking\_rates dan mengkalkulasi biaya total secara real-time.
3. Cetak Struk PDF Otomatis saat Kendaraan Keluar: Setelah transaksi keluar selesai, sistem dapat menghasilkan struk parkir dalam format PDF secara otomatis. Struk berisi detail plat nomor, jenis kendaraan, waktu masuk dan keluar, durasi parkir, serta tarif yang dikenakan. Struk ini disimpan di direktori khusus /struk/.

4. Kelola Data User (Admin & Operator): Admin memiliki hak akses penuh untuk menambah, mengedit, dan menghapus akun user. Fitur ini penting untuk mengatur siapa saja yang dapat menggunakan sistem, termasuk membedakan antara peran admin dan operator.
5. Filter & Pencarian Data Transaksi: Fitur pencarian dan filter transaksi memudahkan pengguna melihat riwayat parkir. Pengguna dapat memfilter data berdasarkan tanggal, plat nomor, atau jenis kendaraan. Hal ini memudahkan dalam melakukan audit atau mencari transaksi tertentu secara cepat.
6. Export Laporan Excel untuk Transaksi & User: Sistem memungkinkan data transaksi dan data user diekspor ke dalam file Excel. Fitur ini sangat berguna untuk pembuatan laporan harian, mingguan, atau bulanan tanpa perlu input ulang data secara manual. File disimpan di dalam folder /laporan/ dan dapat digunakan sebagai backup maupun untuk keperluan pelaporan resmi.

Fitur-fitur tersebut menjadikan sistem parkir ini tidak hanya praktis digunakan tetapi juga sangat mendukung kebutuhan operasional parkir modern yang mengandalkan otomatisasi dan digitalisasi.

## 2.3 Konsep OOP dalam Kode Program

Object-Oriented Programming (OOP) atau Pemrograman Berbasis Objek adalah paradigma pemrograman yang berfokus pada penggunaan objek dan class untuk membangun solusi dari suatu sistem. Dalam sistem parkir otomatis ini, seluruh struktur program dikembangkan menggunakan pendekatan OOP yang mencakup: Class dan Object, Enkapsulasi, Pewarisan (Inheritance), Polimorfisme (Polymorphism), Exception Handling, Modularisasi, dan Event-Driven Programming. Berikut adalah implementasi masing-masing konsep dalam kode program:

**1. Class dan Object** Class digunakan sebagai blueprint atau cetakan untuk membuat objek. Dalam sistem ini, entitas utama direpresentasikan sebagai class:

- Vehicle.java → merepresentasikan data kendaraan
- User.java → menyimpan data pengguna
- ParkingTransaction.java → mencatat data transaksi parkir
- ParkingRate.java → menyimpan informasi tarif parkir

Contoh penggunaan object:

```
Vehicle motor = new Vehicle("B1234XYZ", "motor");
```

Kode di atas membuat objek kendaraan motor dengan data plat nomor dan tipe kendaraan.

**2. Enkapsulasi (Encapsulation)** Enkapsulasi diterapkan dengan menjadikan semua atribut class bersifat private, lalu menyediakan akses melalui getter dan setter:

```
private String plateNumber;  
public String getPlateNumber() {  
    return plateNumber;  
}  
public void setPlateNumber(String plateNumber) {  
    this.plateNumber = plateNumber;  
}
```

Dengan demikian, data hanya bisa dimodifikasi melalui method yang telah disediakan, menjaga integritas data.

**3. Pewarisan (Inheritance)** Inheritance memungkinkan sebuah class mewarisi properti dan method dari class lain. Dalam sistem ini:

- Kelas GUI seperti MainFrame, LoginView, dan lainnya mewarisi javax.swing.JFrame.

Contoh:

```
public class LoginView extends JFrame {  
    // kode implementasi tampilan login  
}
```

Pewarisan ini memungkinkan LoginView langsung menggunakan komponen GUI bawaan dari JFrame.

**4. Polimorfisme (Polymorphism)** Polimorfisme memungkinkan objek berperilaku berbeda tergantung konteks pemanggilan.

Contoh penerapan dalam listener:

```
loginButton.addActionListener(e -> handleLogin());
```

Metode handleLogin() bisa berbeda di setiap tampilan, meskipun dipicu oleh perintah yang sama yaitu klik tombol.

**5. Exception Handling** Sistem ini menangani kemungkinan kesalahan dengan try-catch block.

Contoh:

```
try (Connection conn = DatabaseConnection.getConnection()) {  
    // eksekusi query  
} catch (SQLException e) {  
    JOptionPane.showMessageDialog(null, "Koneksi gagal: " + e.getMessage());  
}
```

Kode di atas memastikan sistem tidak langsung crash saat koneksi database gagal.

**6. Modularisasi dan Struktur Paket** Sistem parkir ini dibagi ke dalam beberapa paket dengan tanggung jawab spesifik:

- models/ → kelas entitas data seperti Vehicle, User, dll.
- controllers/ → logika bisnis dan pengolahan input
- dao/ → Data Access Object, mengatur koneksi dan query database
- views/ → tampilan GUI pengguna
- utils/ → alat bantu seperti ParkingCalculator dan DateTimeUtil

Struktur ini memudahkan pemeliharaan dan pengembangan sistem dengan prinsip Single Responsibility.

**7. Event-Driven Programming (EDP)** GUI dalam sistem ini berbasis event-driven, artinya sistem merespon aksi pengguna secara langsung.

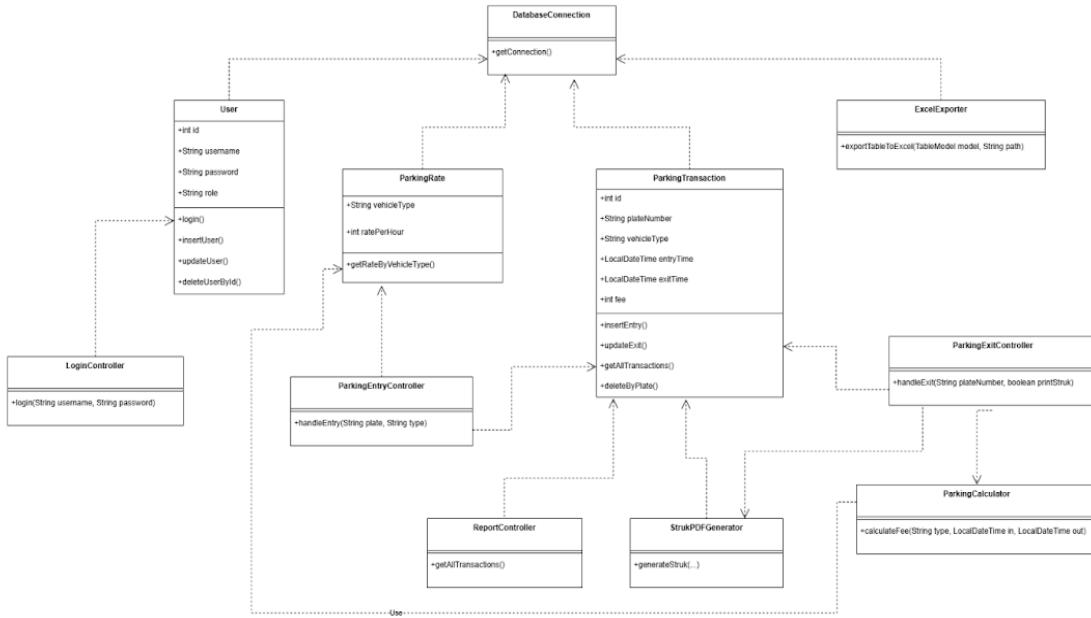
Contoh:

```
exitButton.addActionListener(e -> handleExit());
```

Ketika tombol exit diklik, sistem memanggil method handleExit() yang kemudian memproses transaksi keluar.

## 2.4 Perancangan UML ( Unifield Modelling Language )

### 2.4.1 Class Diagram



Pada inti sistem, terdapat kelas User yang merepresentasikan setiap pengguna sistem, baik itu petugas maupun admin. Kelas ini menyimpan identitas unik (id\_user), kredensial (username, password), dan peran (id\_role) pengguna. Perilaku utamanya mencakup login(), insertUser(), updateUser(), dan deleteUser() untuk manajemen akun. Kelas User ini kemungkinan besar akan berinteraksi dengan LoginController yang bertanggung jawab atas logika otentikasi.

Sistem parkir itu sendiri memiliki dua entitas penting yang mengelola transaksi parkir: ParkingTable dan ParkingTransaction. Kelas ParkingTable bertanggung jawab untuk mengelola data master terkait slot parkir dan konfigurasi dasar, seperti id\_slot, jenis\_slot, dan status\_slot. Kelas ini memiliki metode seperti addParkingSpace() dan deleteParkingSpace() untuk mengatur ketersediaan slot. Sementara itu, ParkingTransaction merekam setiap aktivitas parkir, mencatat id\_transaction, plat\_number, entry\_time, exit\_time, dan final\_fee. Kelas ini memiliki metode saveTransaction() untuk mencatat transaksi dan updateTransaction() untuk memperbarui detail saat kendaraan keluar.

Perhitungan biaya parkir ditangani oleh dua kelas terkait: ParkingCalculator dan

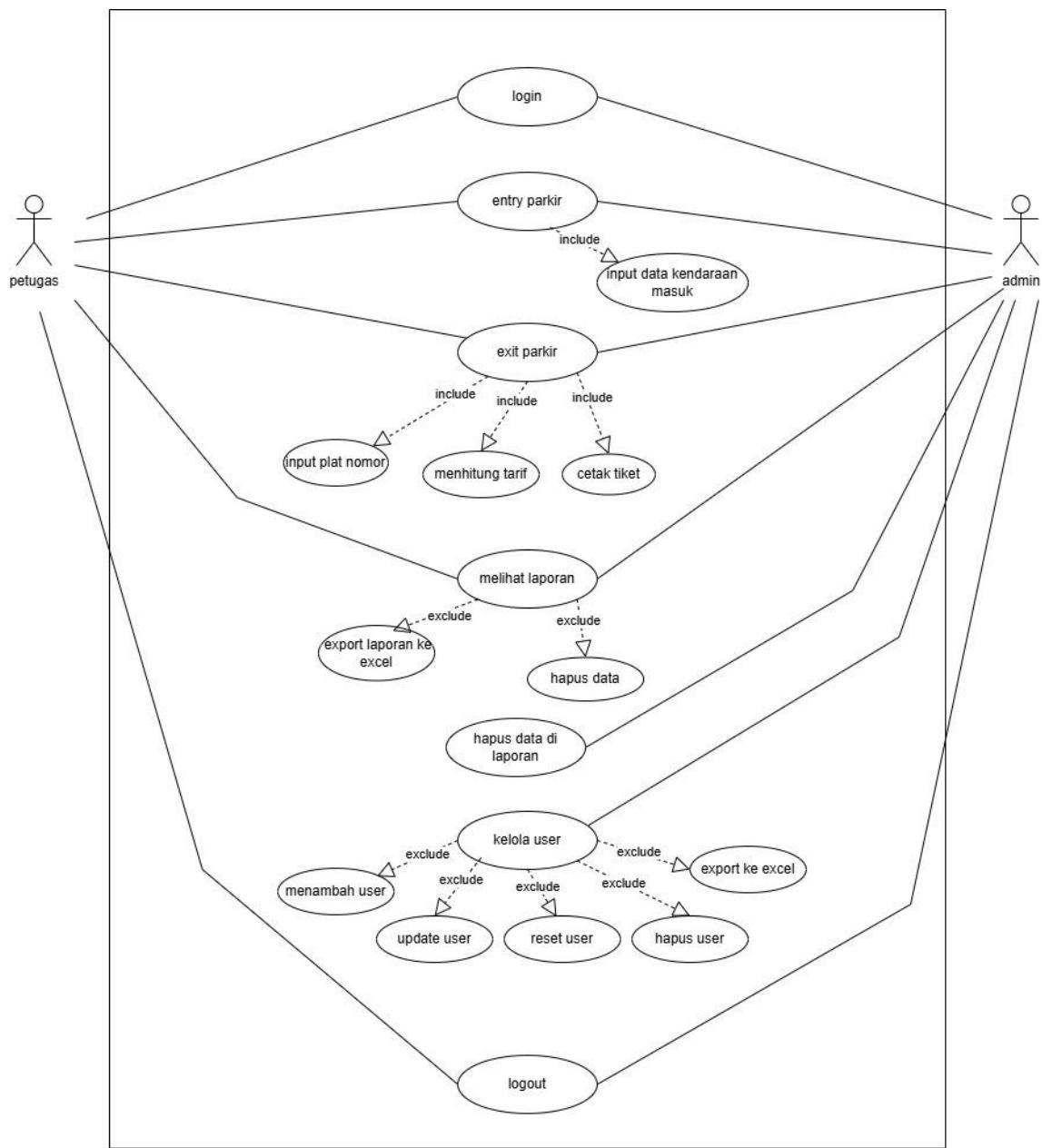
ParkingTariffCalculator. ParkingTariffCalculator berfokus pada manajemen tarif dasar, menyimpan id\_tarif, jenis\_kendaraan, dan tarif\_per\_jam, serta memiliki metode getTarif() untuk mengambil tarif yang sesuai. ParkingCalculator menggunakan tarif ini untuk calculateParkingFee(entryTime, exitTime, tarif) dan berinterelasi dengan ParkingTariffCalculator untuk mendapatkan data tarif.

Untuk interaksi dengan database, ada kelas DatabaseConnection yang menyediakan metode getConnection() untuk membangun koneksi ke database, memungkinkan berbagai kelas lain untuk menyimpan atau mengambil data. Beberapa kelas seperti User, ParkingTable, dan ParkingTransaction akan sangat bergantung pada DatabaseConnection ini.

Terdapat juga kelas ExcelConverter dengan metode exportDataToExcel(filename, sheetName, data) yang berfungsi untuk mengekspor data laporan ke format Excel, serta PrintController dengan metode getReceipt() dan generateInvoice() untuk menangani pencetakan struk dan pembuatan invoice.

Logika kontrol untuk fitur-fitur utama diimplementasikan melalui beberapa kelas kontroler. LoginController menangani login(username, password) untuk proses otentikasi. ParkingEntryController memiliki metode handleEntry(String platNomor) untuk mengelola proses kendaraan masuk. ParkingExitController dengan metode handleExit(String platNomor) bertanggung jawab atas alur kendaraan keluar. Terakhir, ReportController dengan getReports() mengelola pengambilan dan penyajian laporan.

## 2.4.2 Use Case Diagram



Use case diagram sistem parkir tersebut memiliki 2 aktor utama yang terdiri dari petugas dan admin. Keduanya memiliki akses untuk melakukan login ke dalam sistem. Setelah login, petugas dan admin dapat melakukan proses entry parkir, yang mencakup proses wajib berupa input data kendaraan masuk, yaitu nomor polisi dan jenis kendaraan. Saat kendaraan keluar, petugas dan admin menggunakan fitur exit parkir yang di dalamnya terdapat tiga proses penting, yaitu input plat nomor, menghitung tarif, dan mencetak tiket. Ketiganya merupakan proses wajib yang harus dilakukan saat kendaraan keluar.

Petugas dan admin juga dapat menggunakan fitur melihat laporan untuk memantau data

transaksi parkir. Namun, ada beberapa fitur tambahan yang hanya digunakan jika diperlukan, seperti export laporan ke Excel dan hapus data dari laporan. Khusus untuk menghapus data dari laporan, fitur tersebut hanya dapat dilakukan oleh admin saja. Selain itu, admin memiliki akses terhadap fitur kelola user untuk mengatur data pengguna sistem. Di dalam kelola user, terdapat beberapa proses tambahan seperti menambah user, update user, reset user, hapus user, dan export ke Excel, yang semuanya bersifat opsional dan tergantung kebutuhan.

Setelah melakukan tugas-tugasnya, baik petugas maupun admin dapat logout dari sistem untuk mengakhiri sesi kerja. Relasi include digunakan pada proses-proses yang wajib dijalankan, Sedangkan relasi exclude menunjukkan proses-proses yang bersifat opsional, digunakan jika diperlukan saja.

### 2.4.3 Activity Diagram

#### 2.4.3.1 Login Petugas

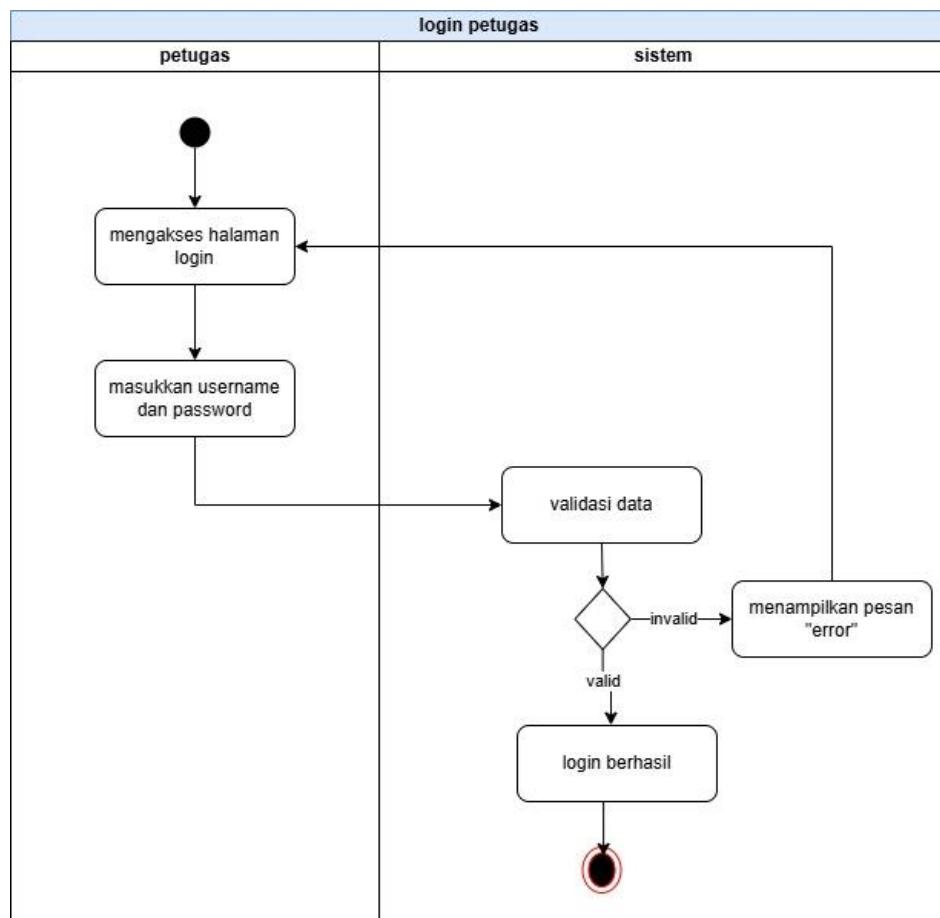


Diagram aktivitas ini menjelaskan alur yang dilakukan oleh petugas saat login ke dalam sistem parkir. Prosesnya dimulai ketika petugas membuka halaman login. Selanjutnya, petugas diminta untuk memasukkan username dan password.

Setelah data login dimasukkan, sistem akan memeriksa apakah username dan password tersebut valid atau tidak. Jika data tidak valid, sistem akan menampilkan pesan "error" kepada petugas. Namun, jika data valid, sistem akan memproses login dan memberikan akses ke petugas.

#### 2.4.3.2 Login Admin

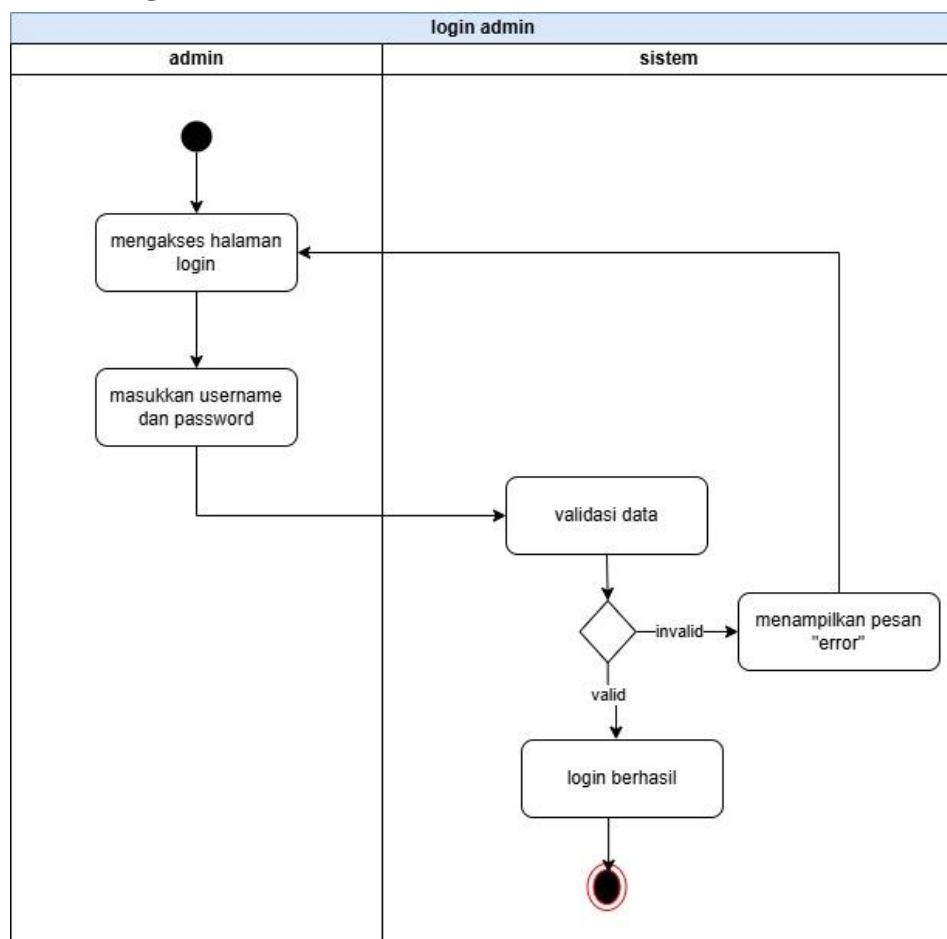
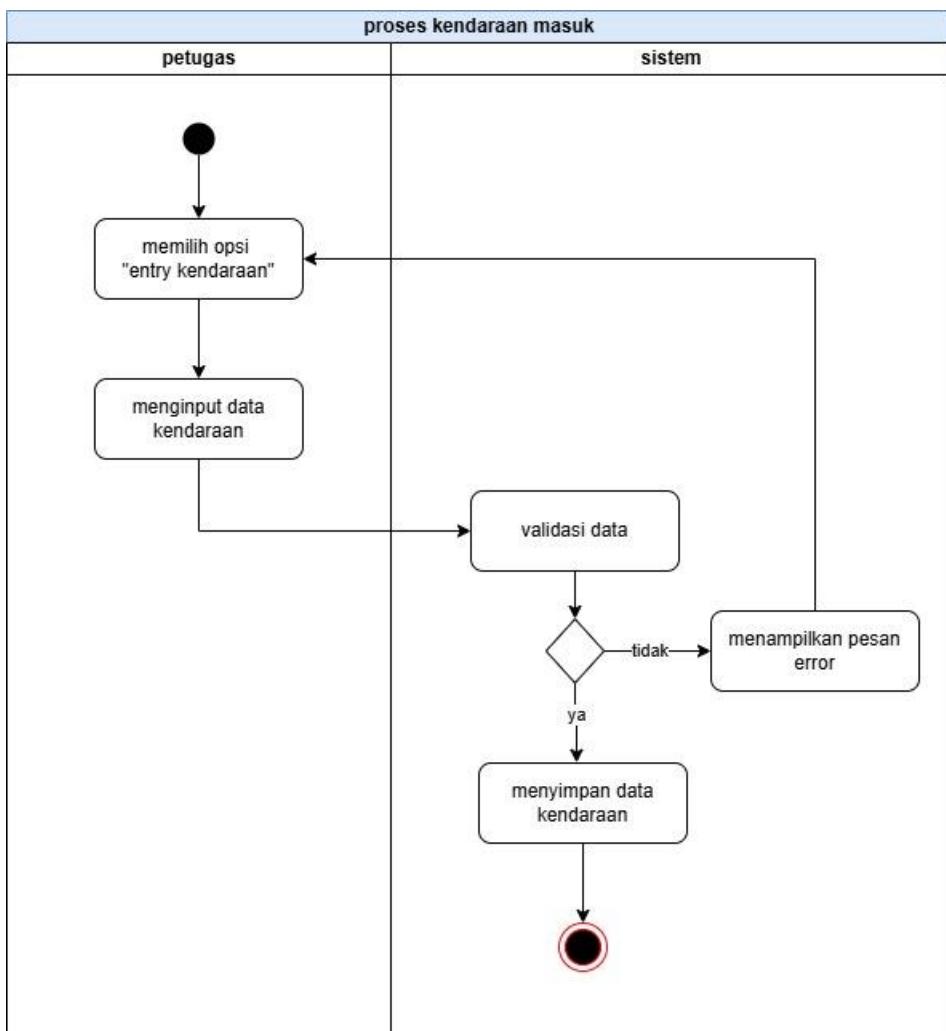


Diagram aktivitas ini menjelaskan alur yang dilakukan oleh admin saat login ke dalam sistem parkir. Prosesnya dimulai ketika admin membuka halaman login. Selanjutnya, admin diminta untuk memasukkan username dan password.

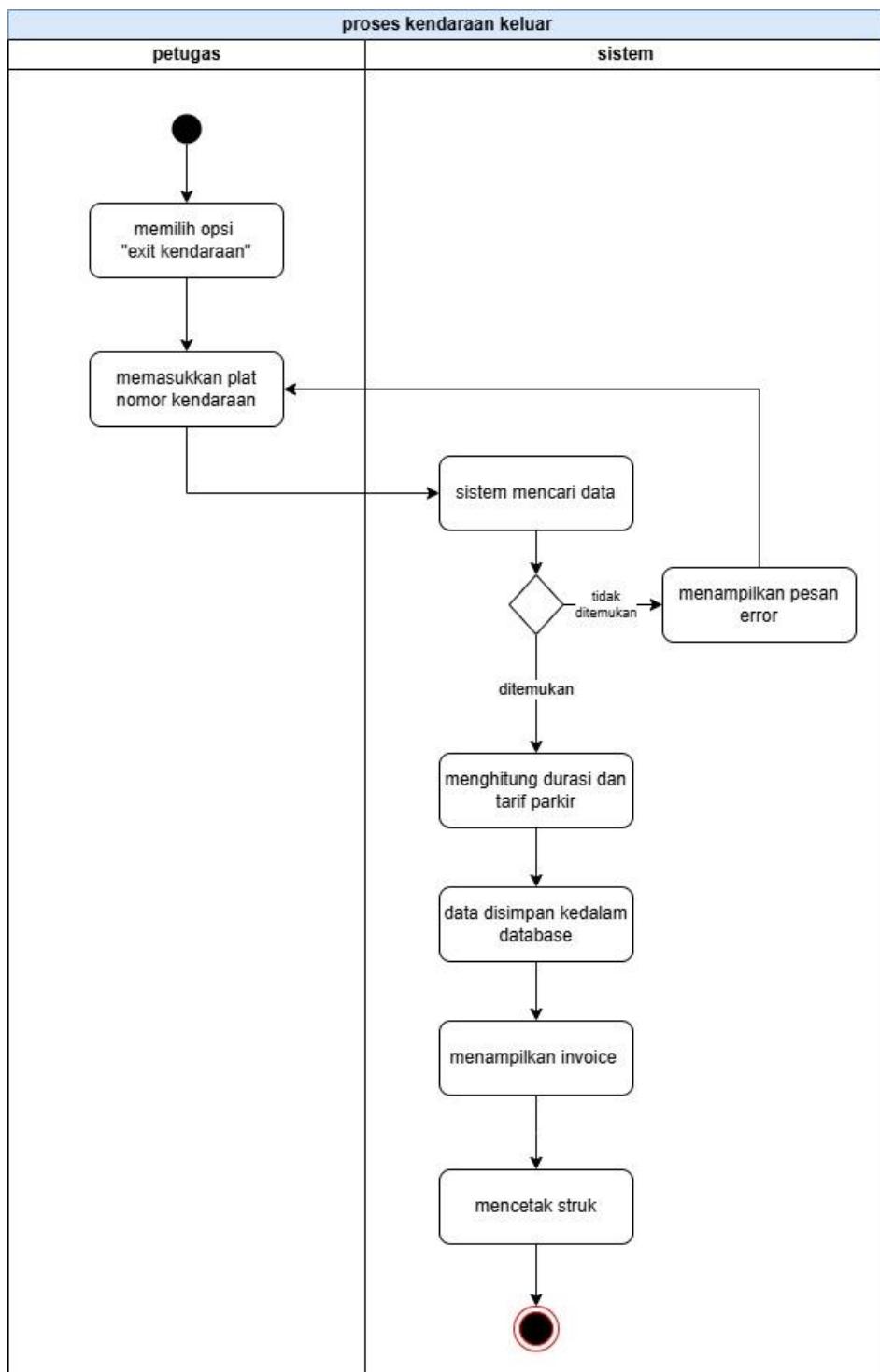
Setelah data login dimasukkan, sistem akan memeriksa apakah username dan password tersebut valid atau tidak. Jika data tidak valid, sistem akan menampilkan pesan "error" kepada admin. Namun, jika data valid, sistem akan memproses login dan memberikan akses ke admin.

#### 2.4.3.3 Kendaraan Masuk



Proses kendaraan masuk dimulai ketika petugas memilih opsi 'Entry Kendaraan' pada sistem parkir. Setelah opsi tersebut dipilih, petugas akan menginput data kendaraan yang masuk, seperti nomor plat, jenis kendaraan, atau informasi lain yang diperlukan. Data yang telah diinput kemudian divalidasi oleh sistem untuk memastikan kelengkapan dan kebenarannya. Jika data tidak valid, sistem akan menampilkan pesan error, dan petugas harus memperbaiki atau mengulangi input data. Sebaliknya, jika data valid, sistem akan menyimpan data kendaraan tersebut ke dalam database, menandakan bahwa proses entry kendaraan telah berhasil dilakukan.

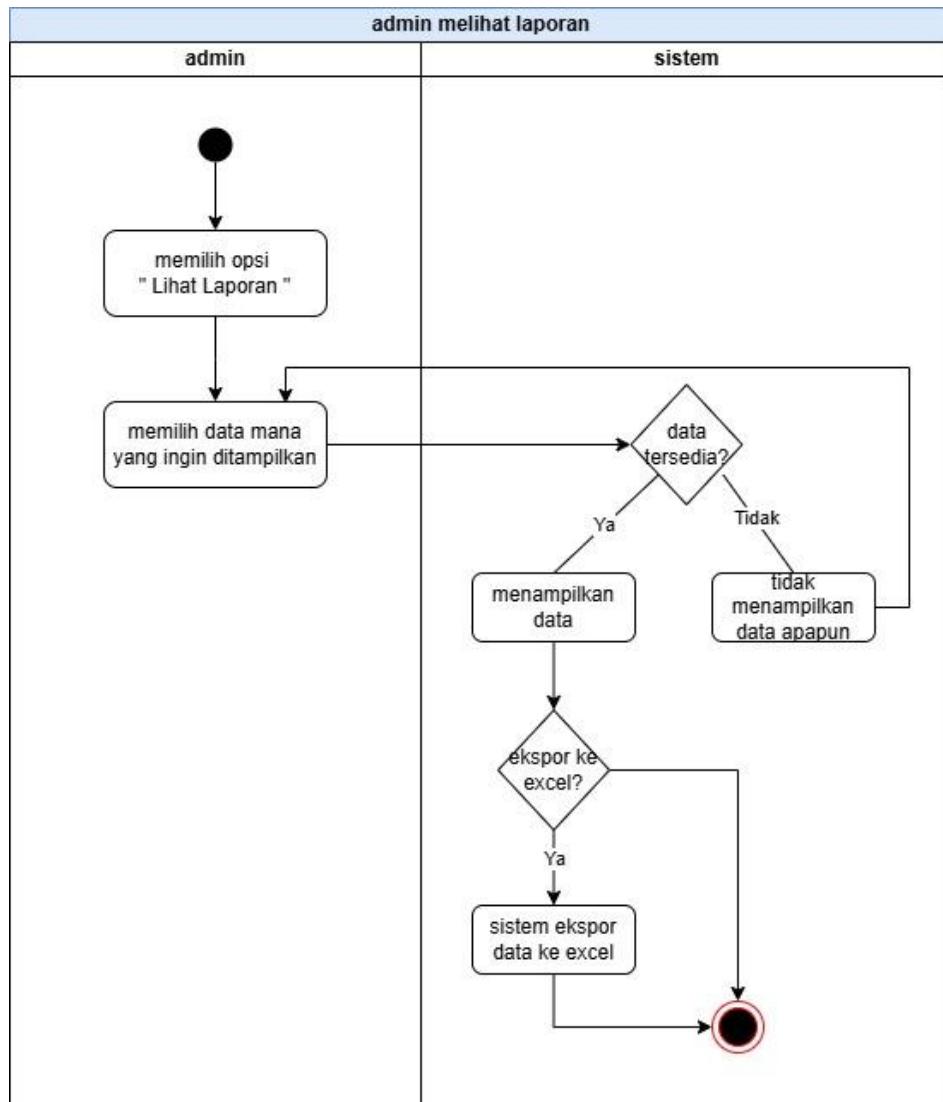
#### 2.4.3.4 Kendaraan Keluar



Proses kendaraan keluar diawali ketika petugas memilih opsi "Exit Kendaraan" pada sistem. Petugas kemudian memasukkan nomor plat kendaraan yang akan keluar. Sistem akan melakukan pencarian data kendaraan tersebut. Jika data tidak ditemukan, sistem akan menampilkan pesan error dan proses berhenti. Namun jika data ditemukan, sistem akan menghitung durasi parkir dan besaran tarif yang harus dibayarkan. Setelah itu, data

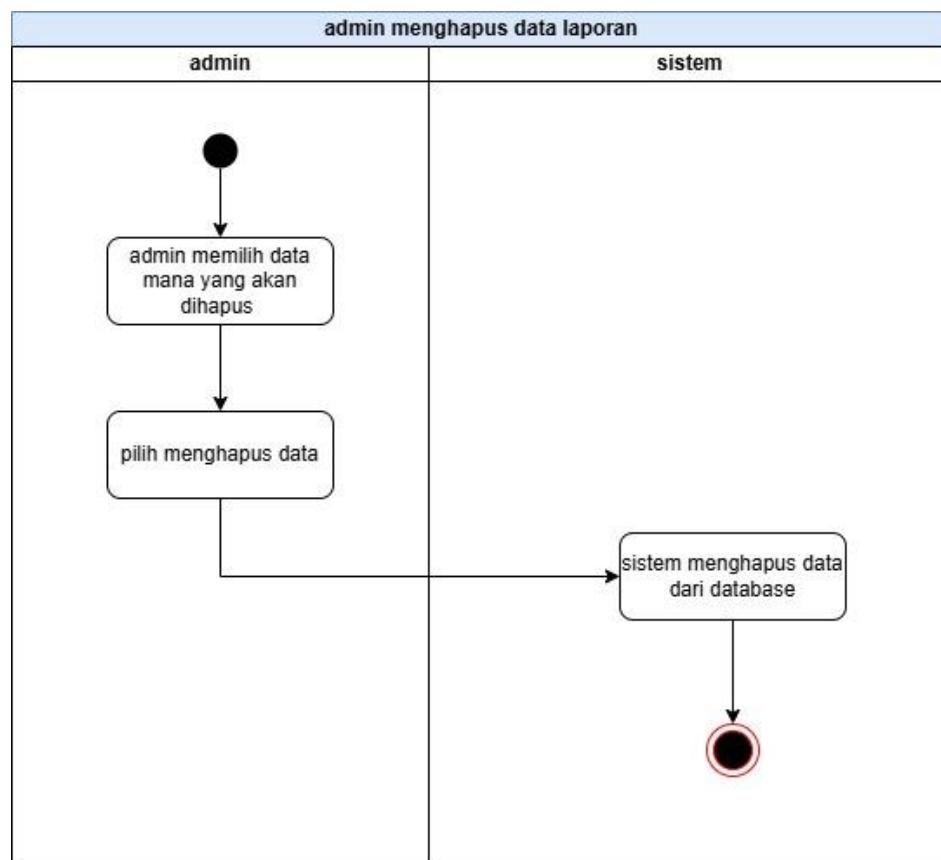
transaksi disimpan ke dalam database. Sistem kemudian menampilkan invoice pembayaran dan mencetak struk parkir sebagai bukti transaksi bagi pengendara.

#### 2.4.3.5 Melihat Laporan



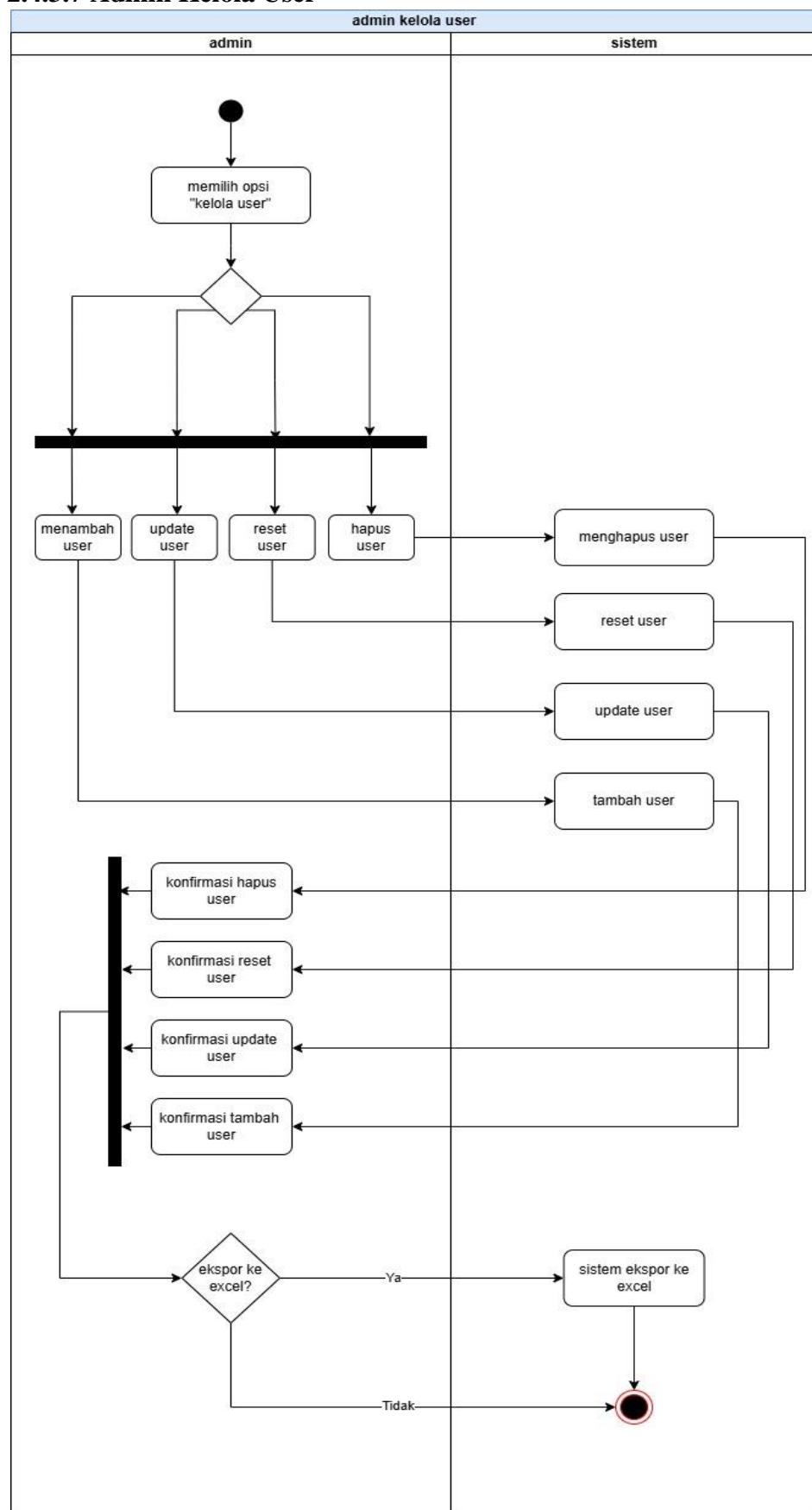
Admin dapat melihat berbagai laporan dengan memilih opsi "Lihat Laporan" pada sistem. Setelah memilih opsi tersebut, admin menentukan data laporan mana yang ingin ditampilkan, seperti laporan harian, mingguan, atau bulanan. Sistem kemudian akan menampilkan data laporan sesuai dengan pilihan admin. Fitur ini memungkinkan admin untuk memantau aktivitas parkir, Seperti pendapatan, dan data statistik lainnya yang diperlukan untuk evaluasi ataupun pengambilan keputusan.

#### 2.4.3.6 Menghapus Data Laporan



Proses penghapusan data laporan dimulai ketika admin mengakses sistem dan memilih data laporan tertentu yang ingin dihapus. Setelah menentukan data yang akan dihapus, admin memilih opsi penghapusan. Sistem kemudian akan memproses permintaan tersebut dengan menghapus data yang dimaksud dari database.

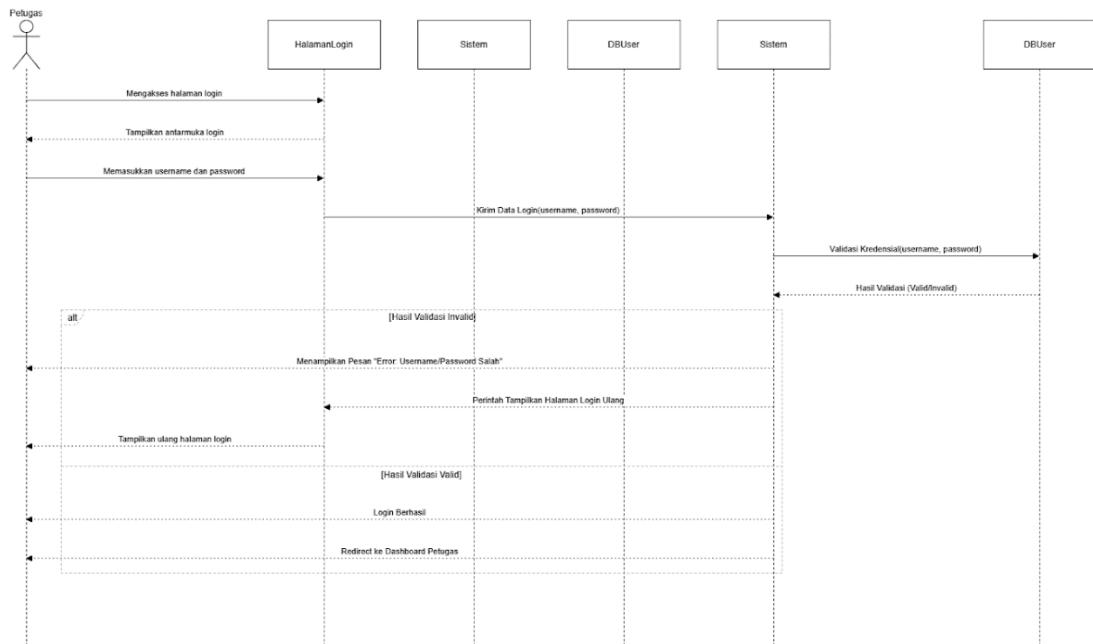
#### 2.4.3.7 Admin Kelola User



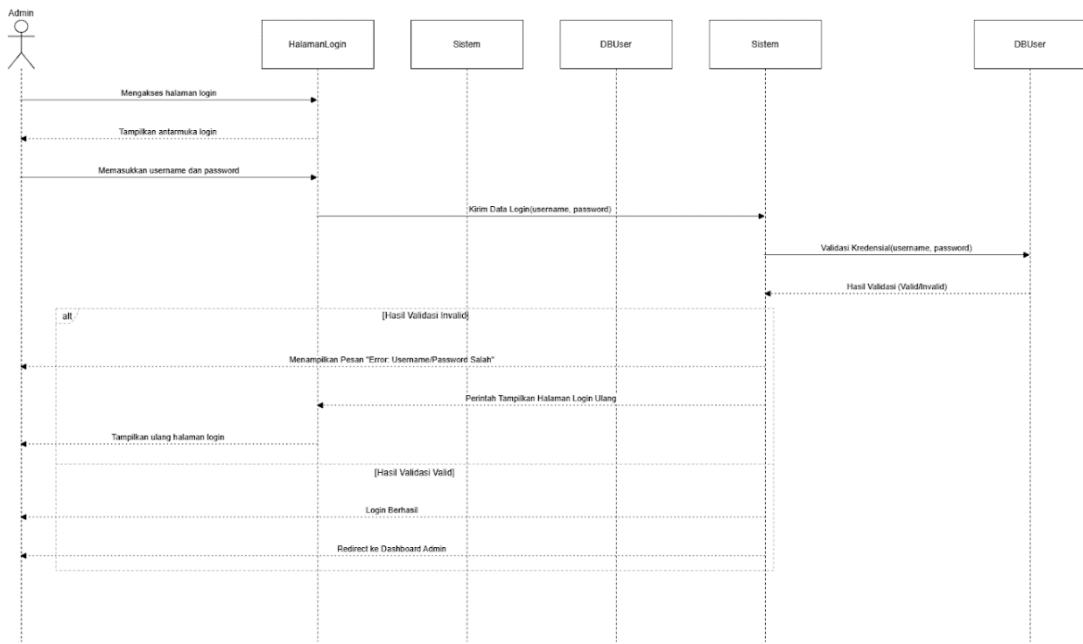
Admin dapat mengelola data pengguna melalui opsi "Kelola User" pada sistem. Terdapat empat pilihan utama yang tersedia: menambah user baru, mengupdate data user, mereset data user, atau menghapus user. Setiap tindakan memerlukan konfirmasi dari admin untuk memastikan tidak terjadi kesalahan operasi. Selain itu, sistem menyediakan opsi untuk mengekspor data user ke format Excel, baik setelah melakukan penambahan, pembaruan, reset, maupun penghapusan user.

#### 2.4.4 Sequence Diagram

##### 2.4.4.1 Login petugas



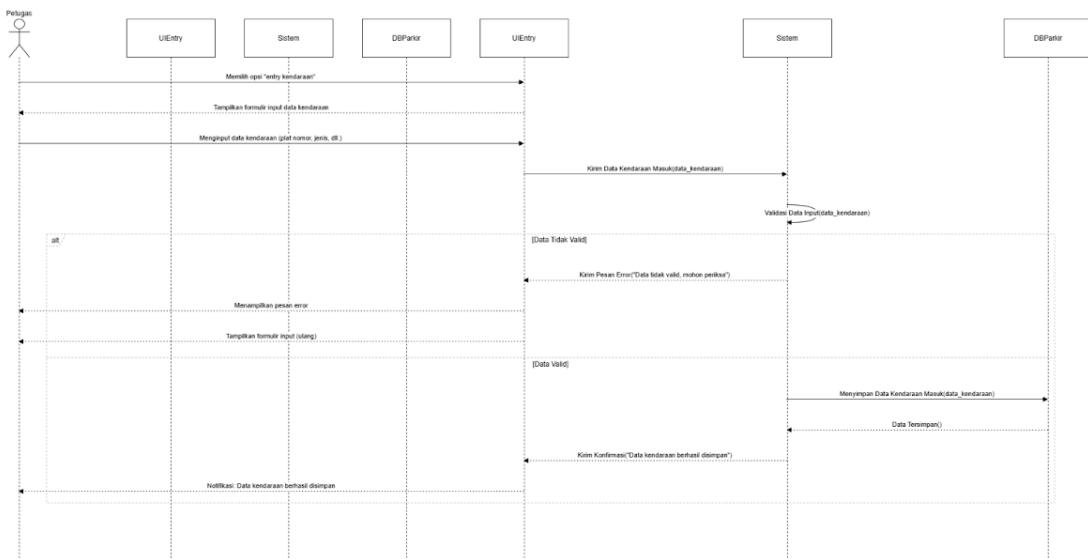
#### 2.4.4.2 Login Admin



Proses login merupakan gerbang awal bagi baik Petugas maupun Admin untuk mengakses fungsionalitas sistem. Alur dimulai ketika aktor (Petugas atau Admin) mencoba untuk mengakses sistem, yang kemudian akan menampilkan Halaman Login. Aktor kemudian memasukkan kredensial (username dan password) ke dalam halaman tersebut. Halaman Login bertanggung jawab untuk meneruskan kredensial ini ke Sistem Parkir.

Selanjutnya, Sistem Parkir akan melakukan validasi kredensial yang diterima dengan membandingkannya terhadap data yang tersimpan di Database Pengguna. Jika kredensial yang dimasukkan tidak valid, Sistem Parkir akan mengirimkan pesan error kembali ke Halaman Login, yang kemudian akan ditampilkan kepada aktor, dan aktor akan diminta untuk mencoba login kembali. Sebaliknya, jika kredensial valid, Sistem Parkir akan mengonfirmasi keberhasilan login dan mengarahkan aktor ke dashboard yang sesuai dengan peran mereka (Dashboard Petugas atau Dashboard Admin), menandai dimulainya sesi kerja dalam sistem.

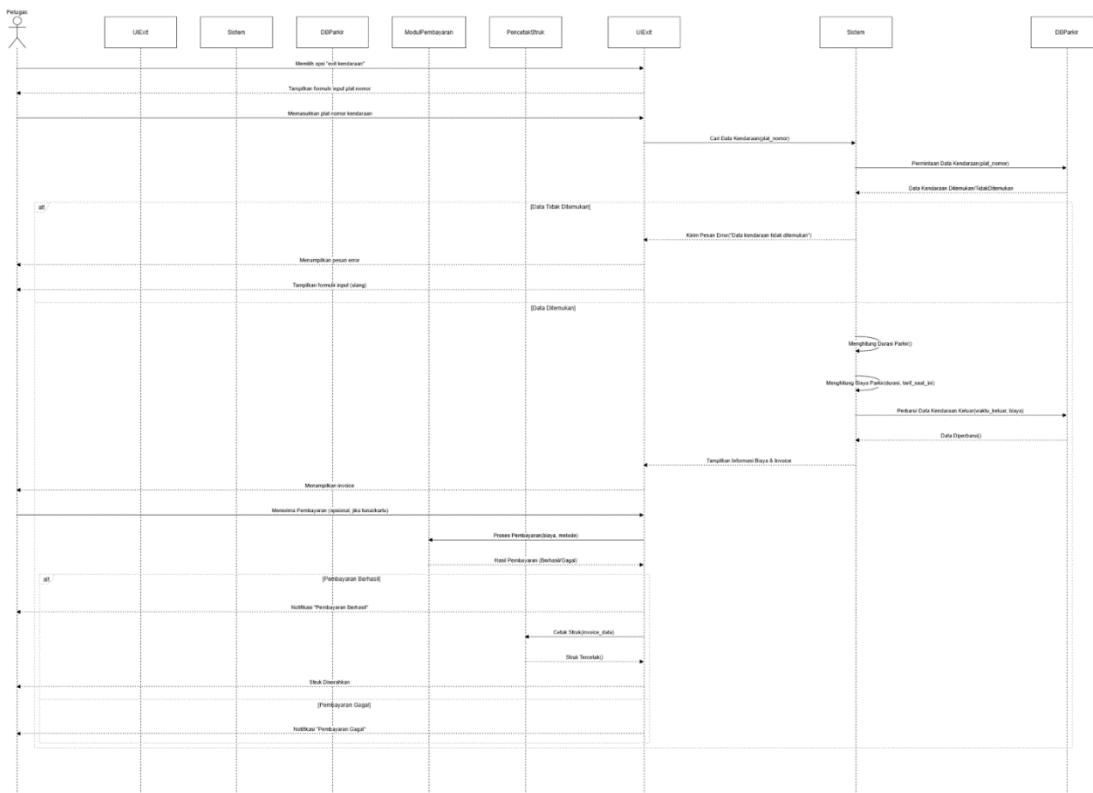
#### 2.4.4.3 Kendraan masuk



Proses pencatatan kendaraan yang masuk ke area parkir diinisiasi oleh Petugas. Ketika Petugas memilih opsi "entry kendaraan" pada antarmuka sistem, Antarmuka Entry Kendaraan akan merespons dengan menampilkan sebuah formulir input. Petugas kemudian bertugas untuk mengisi detail data kendaraan, seperti plat nomor dan jenis kendaraan, ke dalam formulir tersebut.

Setelah data diinput, Antarmuka Entry Kendaraan akan meneruskan data tersebut ke Sistem Parkir. Sistem Parkir akan segera melakukan validasi terhadap data yang diterima untuk memastikan kelengkapan dan keakuratannya. Apabila ditemukan bahwa data tidak valid, Sistem Parkir akan mengirimkan pesan kesalahan kembali ke antarmuka, yang kemudian akan ditampilkan kepada Petugas, dan Petugas harus mengulang proses input data. Namun, jika data terbukti valid, Sistem Parkir akan melanjutkan untuk menyimpan data kendaraan masuk tersebut ke Database Parkir. Setelah penyimpanan berhasil dikonfirmasi oleh database, Sistem Parkir akan memberikan notifikasi keberhasilan kepada Antarmuka Entry Kendaraan, yang pada gilirannya akan menampilkan pesan konfirmasi kepada Petugas, menandakan bahwa data kendaraan masuk telah berhasil dicatat.

#### 2.4.4.4 Kendaraan Keluar



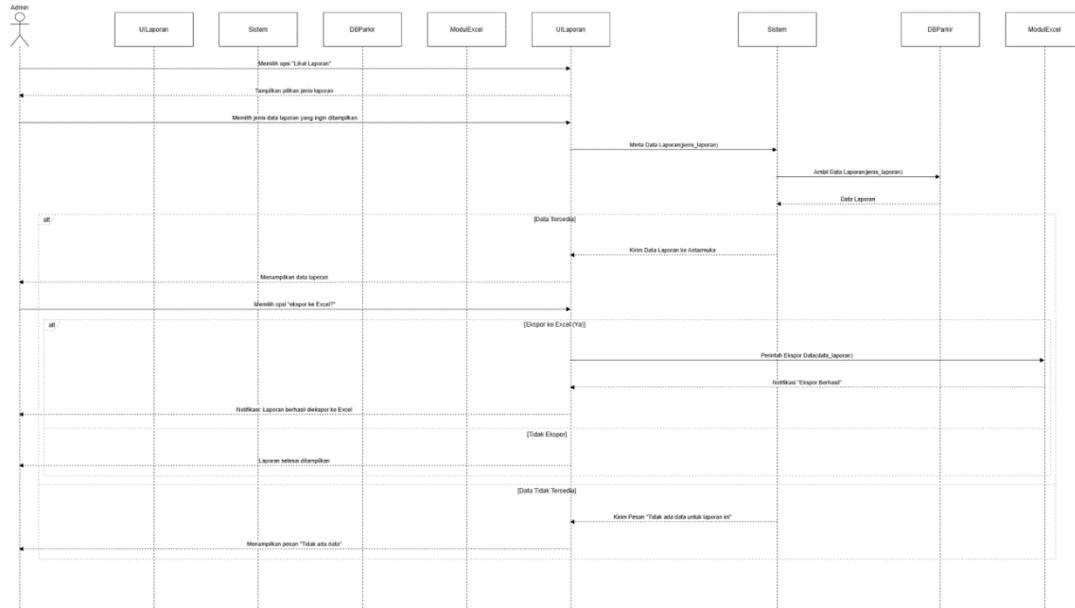
Proses kendaraan keluar merupakan alur kritis yang juga dioperasikan oleh Petugas. Dimulai ketika Petugas memilih opsi "exit kendaraan" pada antarmuka sistem. Antarmuka Exit Kendaraan kemudian akan meminta Petugas untuk memasukkan plat nomor kendaraan yang akan keluar. Setelah plat nomor diinput, Antarmuka Exit Kendaraan meneruskannya ke Sistem Parkir untuk proses pencarian data.

Sistem Parkir selanjutnya akan meminta Database Parkir untuk mencari data kendaraan berdasarkan plat nomor yang diberikan. Jika data kendaraan tidak ditemukan di database, Sistem Parkir akan mengirimkan pesan error ke antarmuka, yang kemudian akan ditampilkan kepada Petugas, dan Petugas harus mengulang pencarian atau memverifikasi plat nomor. Apabila data kendaraan ditemukan, Sistem Parkir akan secara otomatis menghitung durasi parkir dan total biaya yang harus dibayar berdasarkan tarif yang berlaku. Informasi ini, beserta waktu keluar, kemudian diperbarui di Database Parkir.

Setelah perhitungan dan pembaruan data selesai, Sistem Parkir akan mengirimkan detail biaya dan invoice ke Antarmuka Exit Kendaraan untuk ditampilkan kepada Petugas. Petugas akan menerima pembayaran dari pengendara (proses ini dapat melibatkan Modul Pembayaran untuk transaksi non-tunai). Setelah pembayaran diproses dan dikonfirmasi berhasil (jika gagal, pesan kesalahan akan ditampilkan), Antarmuka Exit Kendaraan akan

memerintahkan Pencetak Struk untuk mencetak bukti pembayaran. Setelah struk tercetak, Petugas menyerahkannya kepada pengendara, menandai selesainya proses keluar kendaraan.

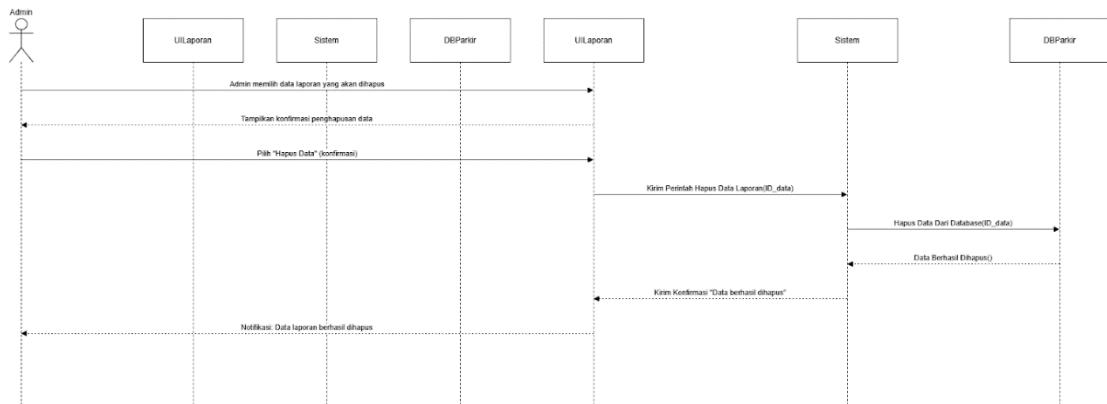
#### 2.4.4.5 Admin Melihat Laporan



Proses melihat laporan merupakan fungsionalitas penting bagi Admin untuk memantau aktivitas dan performa parkir. Proses ini dimulai ketika Admin memilih opsi "Lihat Laporan" melalui Antarmuka Laporan, yang kemudian akan menampilkan berbagai pilihan jenis laporan (misalnya, laporan harian, bulanan, atau pendapatan). Admin kemudian memilih jenis laporan spesifik yang ingin dilihat.

Antarmuka Laporan akan meneruskan permintaan ini ke Sistem Parkir, yang selanjutnya akan mengambil data yang relevan dari Database Parkir. Database Parkir akan mengembalikan data laporan tersebut ke Sistem Parkir. Jika data laporan tersedia, Sistem Parkir akan mengirimkannya kembali ke Antarmuka Laporan untuk ditampilkan kepada Admin. Pada tahap ini, Admin juga akan diberikan opsi untuk mengekspor laporan ke format Excel. Jika Admin memilih opsi eksport, Antarmuka Laporan akan memicu Modul Eksport Excel untuk melakukan proses tersebut, dan setelah berhasil, Admin akan menerima notifikasi. Namun, jika tidak ada data yang ditemukan untuk jenis laporan yang diminta, Sistem Parkir akan memberitahukan kepada Antarmuka Laporan bahwa tidak ada data untuk ditampilkan, dan pesan tersebut akan disampaikan kepada Admin.

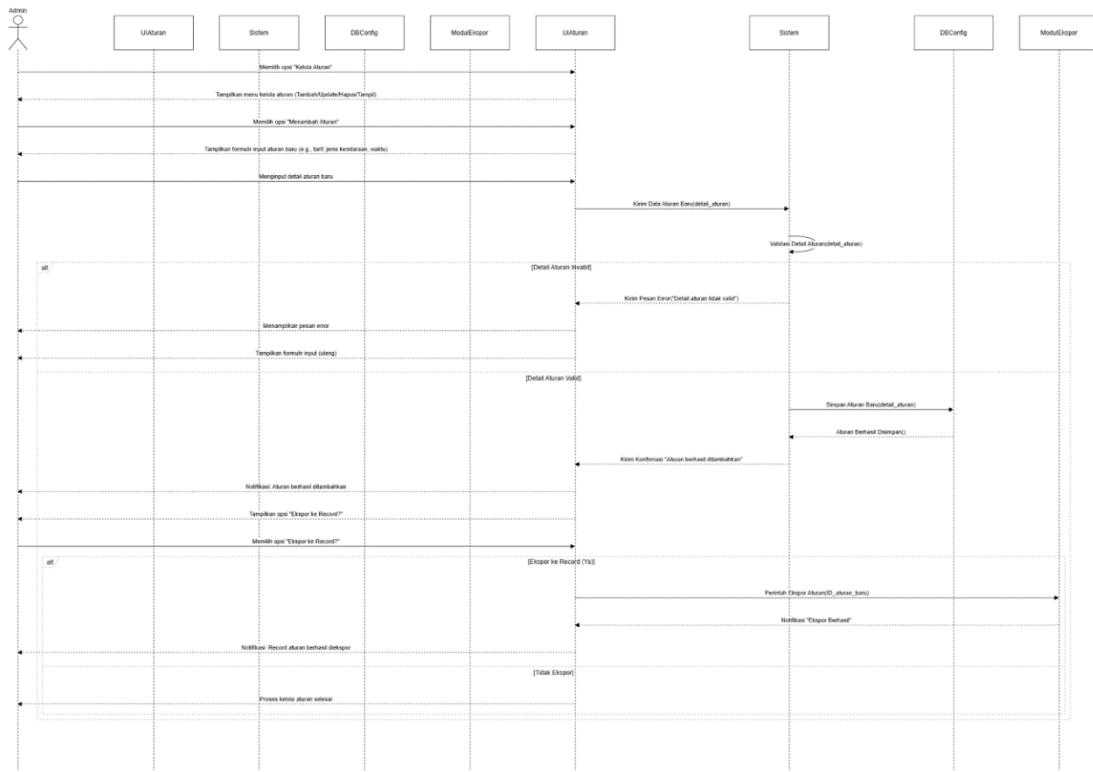
#### 2.4.4.6 Admin Menghapus Data Laporan



Proses penghapusan data laporan merupakan fungsi administratif yang hanya dapat dilakukan oleh Admin untuk menjaga kebersihan dan relevansi data. Alur dimulai ketika Admin mengidentifikasi dan memilih data laporan tertentu yang ingin dihapus melalui Antarmuka Laporan. Sebagai tindakan pencegahan, Antarmuka Laporan akan terlebih dahulu menampilkan permintaan konfirmasi kepada Admin untuk memastikan keputusan penghapusan.

Setelah Admin mengonfirmasi penghapusan, Antarmuka Laporan akan mengirimkan perintah penghapusan data tersebut ke Sistem Parkir. Sistem Parkir kemudian akan berinteraksi langsung dengan Database Parkir untuk menjalankan perintah penghapusan data yang telah dipilih. Setelah Database Parkir berhasil menghapus data, konfirmasi akan dikirimkan kembali ke Sistem Parkir. Kemudian, Sistem Parkir akan menginformasikan keberhasilan penghapusan data kepada Antarmuka Laporan, yang pada akhirnya akan menampilkan notifikasi kepada Admin bahwa data laporan telah berhasil dihapus dari sistem.

#### 2.4.4.7 Admin Kelola Aturan



Proses pengelolaan aturan, khususnya penambahan aturan baru, adalah tugas khusus Admin untuk mengonfigurasi operasional sistem parkir. Proses ini dimulai ketika Admin memilih opsi "Kelola Aturan" pada antarmuka sistem, yang kemudian akan menampilkan menu dengan pilihan seperti menambah, mengubah, menghapus, atau menampilkan aturan. Admin kemudian memilih sub-opsi "Menambah Aturan".

Setelah memilih sub-opsi tersebut, Antarmuka Kelola Aturan akan menampilkan formulir yang memungkinkan Admin untuk menginput detail aturan baru, seperti tarif parkir per jam, jenis kendaraan yang berlaku untuk tarif tersebut, atau durasi waktu tertentu. Data aturan yang telah diinput ini kemudian dikirimkan oleh Antarmuka Kelola Aturan ke Sistem Parkir.

Sistem Parkir akan melakukan validasi terhadap detail aturan yang diterima untuk memastikan format yang benar dan mencegah konflik dengan aturan yang sudah ada. Jika detail aturan dianggap tidak valid, Sistem Parkir akan mengirimkan pesan error kembali ke antarmuka, dan Admin harus mengulang proses input. Namun, jika aturan valid, Sistem Parkir akan melanjutkan untuk menyimpan aturan baru tersebut ke Database Konfigurasi. Setelah penyimpanan berhasil, Sistem Parkir akan mengirimkan konfirmasi keberhasilan ke Antarmuka Kelola Aturan, yang kemudian akan menampilkan notifikasi

kepada Admin. Pada tahap ini, Admin juga diberikan opsi untuk mengekspor catatan aturan tersebut ke record. Jika Admin memilih untuk mengekspor, Antarmuka Kelola Aturan akan berinteraksi dengan Modul Ekspor Record untuk menjalankan proses ekspor, dan setelahnya Admin akan menerima notifikasi bahwa record aturan berhasil diekspor.

## 2.5 Perancangan Database

The screenshot shows the phpMyAdmin interface for the 'parkir\_db' database. The 'Structure' tab is active, displaying the following table information:

| Table         | Action   | Rows | Type   | Collation          | Size     | Overhead |
|---------------|--|------|--------|--------------------|----------|----------|
| parking_rates | Browse  Structure  Search  Insert  Empty  Drop | 3    | InnoDB | utf8mb4_general_ci | 32.0 KiB | -        |
| transactions  | Browse  Structure  Search  Insert  Empty  Drop | 16   | InnoDB | utf8mb4_general_ci | 16.0 KiB | -        |
| users         | Browse  Structure  Search  Insert  Empty  Drop | 3    | InnoDB | utf8mb4_general_ci | 16.0 KiB | -        |
| 3 tables      | Sum  | 22   | InnoDB | utf8mb4_general_ci | 64.0 KiB | 0 B      |

Below the table list, there are buttons for 'Check all' and 'With selected:'. At the bottom left, there are links for 'Print' and 'Data dictionary'. On the right side, there is a 'Console' tab.

### • Parking Rates

The screenshot shows the phpMyAdmin interface for the 'parking\_rates' table. The 'Structure' tab is active, displaying the following column information:

| # | Name          | Type        | Collation          | Attributes | Null | Default | Comments       | Extra              | Action             |
|---|---------------|-------------|--------------------|------------|------|---------|----------------|--------------------|--------------------|
| 1 | id            | int(11)     | utf8mb4_general_ci |            | No   | None    | AUTO_INCREMENT | Change  Drop  More | Change  Drop  More |
| 2 | vehicle_type  | varchar(50) | utf8mb4_general_ci |            | No   | None    |                | Change  Drop  More | Change  Drop  More |
| 3 | rate_per_hour | int(11)     |                    |            | No   | None    |                | Change  Drop  More | Change  Drop  More |

Below the table structure, there are buttons for 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There are also links for 'Add to central columns' and 'Remove from central columns'. At the bottom, there is a search bar for 'Add' and 'column(s)' with a 'Go' button, and a link for 'Normalize'.

The 'Indexes' tab is also visible, showing the following index information:

| Action             | Keyname      | Type  | Unique | Packed | Column       | Cardinality | Collation | Null | Comment |
|--------------------|--------------|-------|--------|--------|--------------|-------------|-----------|------|---------|
| Edit  Rename  Drop | PRIMARY      | BTREE | Yes    | No     | id           | 3           | A         | No   |         |
| Edit  Rename  Drop | vehicle_type | BTREE | Yes    | No     | vehicle_type | 3           | A         | No   |         |

At the bottom, there is a link for 'Create an index on' followed by a 'Go' button, and a 'Console' tab.

Showing rows 0 - 2 (3 total, Query took 0.0003 seconds.)

SELECT \* FROM `parking\_rates`

id vehicle\_type rate\_per\_hour

|   |       |      |
|---|-------|------|
| 1 | Mobil | 3000 |
| 2 | Motor | 2000 |
| 3 | Truk  | 5000 |

Query results operations

Print Copy to clipboard Export Display chart Create view

## ● Transaction

Table structure

| # | Name         | Type        | Collation          | Attributes | Null | Default | Comments       | Extra | Action            |
|---|--------------|-------------|--------------------|------------|------|---------|----------------|-------|-------------------|
| 1 | id           | int(11)     | utf8mb4_general_ci |            | No   | None    | AUTO_INCREMENT |       | Change  Drop More |
| 2 | plate_number | varchar(20) | utf8mb4_general_ci |            | Yes  | NULL    |                |       | Change  Drop More |
| 3 | vehicle_type | varchar(20) | utf8mb4_general_ci |            | Yes  | NULL    |                |       | Change  Drop More |
| 4 | entry_time   | datetime    |                    |            | Yes  | NULL    |                |       | Change  Drop More |
| 5 | exit_time    | datetime    |                    |            | Yes  | NULL    |                |       | Change  Drop More |
| 6 | total_fee    | int(11)     |                    |            | Yes  | NULL    |                |       | Change  Drop More |

Add to central columns Remove from central columns

Print Propose table structure Track table Move columns Normalize

Add 1 column(s) after total\_fee Go

Indexes

| Action             | Keyname | Type  | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
|--------------------|---------|-------|--------|--------|--------|-------------|-----------|------|---------|
| Edit  Rename  Drop | PRIMARY | BTREE | Yes    | No     | id     | 17          | A         | No   |         |

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

id plate\_number vehicle\_type entry\_time exit\_time total\_fee

|    |            |       |                     |                     |       |
|----|------------|-------|---------------------|---------------------|-------|
| 2  | 365T8U83   | Mobil | 2025-05-27 22:17:45 | 2025-06-07 01:43:15 | 25000 |
| 4  | 12476Y73   | Motor | 2025-05-27 22:48:57 | 2025-05-27 22:49:50 | 3000  |
| 5  | 327654T78  | Motor | 2025-05-27 23:04:49 | 2025-05-27 23:06:01 | 3000  |
| 7  | 1253T8Y8   | Motor | 2025-05-28 19:46:50 | 2025-05-28 19:47:00 | 3000  |
| 9  | 237537T685 | Motor | 2025-05-31 22:31:01 | 2025-06-06 22:06:04 | 25000 |
| 12 | 123764T7   | Motor | 2025-06-06 20:18:03 | 2025-06-06 20:18:14 | 2000  |
| 13 | 1254SVU67  | Mobil | 2025-06-07 11:08:01 | 2025-06-07 11:08:08 | 3000  |
| 14 | 1254TU56   | Motor | 2025-06-07 11:08:52 | 2025-06-07 11:08:57 | 2000  |
| 15 | WRG23RT    | Motor | 2025-06-07 11:23:17 | 2025-06-07 11:23:27 | 2000  |
| 17 | 123TR456   | Mobil | 2025-06-07 11:24:38 | 2025-06-07 11:24:44 | 3000  |
| 18 | WRS56FRD   | Motor | 2025-06-07 11:39:07 | 2025-06-07 11:39:19 | 2000  |
| 24 | 234RT678   | Mobil | 2025-06-07 18:56:34 | 2025-06-07 18:56:41 | 3000  |
| 25 | 214YU70    | Motor | 2025-06-07 19:07:28 | 2025-06-07 19:11:34 | 2000  |
| 26 | 123ERT567  | Motor | 2025-06-07 20:20:40 | 2025-06-07 20:32:25 | 2000  |

- **Users**

Server: 127.0.0.1 » Database: parkir\_db » Table: users

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Tracking](#) [More](#)

[Table structure](#) [Relation view](#)

| # | Name                     | Type         | Collation          | Attributes | Null | Default | Comments | Extra          | Action   |
|---|--------------------------|--------------|--------------------|------------|------|---------|----------|----------------|--|
| 1 | <a href="#">id</a>       | int(11)      |                    |            | No   | None    |          | AUTO_INCREMENT | <a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a> |
| 2 | <a href="#">username</a> | varchar(50)  | utf8mb4_general_ci |            | No   | None    |          |                | <a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a> |
| 3 | <a href="#">password</a> | varchar(100) | utf8mb4_general_ci |            | No   | None    |          |                | <a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a> |
| 4 | <a href="#">role</a>     | varchar(20)  | utf8mb4_general_ci |            | Yes  | NULL    |          |                | <a href="#">Change</a> <a href="#">Drop</a> <a href="#">More</a> |

[Check all](#) *With selected:* [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#) [Spatial](#) [Fulltext](#)

[Add to central columns](#) [Remove from central columns](#)

---

[Print](#) [Propose table structure](#) [Track table](#) [Move columns](#) [Normalize](#)

[Add](#)  column(s) [after role](#) [Go](#)

[Indexes](#) [Create index](#)

| Action   | Keyname | Type  | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
|--|---------|-------|--------|--------|--------|-------------|-----------|------|---------|
| <a href="#">Edit</a> <a href="#">Rename</a> <a href="#">Drop</a> | PRIMARY | BTREE | Yes    | No     | id     | 4           | A         | No   |         |

[Create an index on](#)  columns [Go](#)

[Console](#)

Server: 127.0.0.1 » Database: parkir\_db » Table: users

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

Showing rows 0 - 2 (3 total, Query took 0.00003 seconds.)

SELECT \* FROM `users`

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

|                          | id | username | password  | role     |
|--------------------------|----|----------|-----------|----------|
| <input type="checkbox"/> | 1  | admin    | admin123  | admin    |
| <input type="checkbox"/> | 3  | dhafa    | dhafa123  | operator |
| <input type="checkbox"/> | 5  | farhan   | farhan123 | admin    |

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

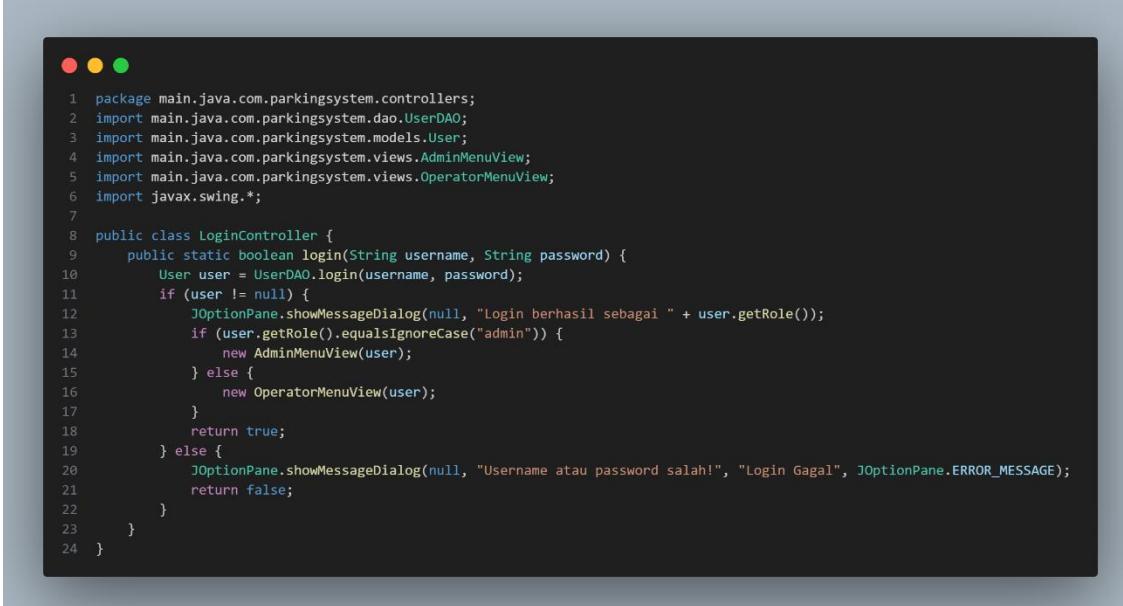
Query results operations

Print Copy to clipboard Export Display chart Create view

## 2.6 Implementasi Kode Program dan Pengujian

### 2.6.1 Controller

#### Logincontroller



```
1 package main.java.com.parkingsystem.controllers;
2 import main.java.com.parkingsystem.dao.UserDAO;
3 import main.java.com.parkingsystem.models.User;
4 import main.java.com.parkingsystem.views.AdminMenuView;
5 import main.java.com.parkingsystem.views.OperatorMenuView;
6 import javax.swing.*;
7
8 public class LoginController {
9     public static boolean login(String username, String password) {
10         User user = UserDAO.login(username, password);
11         if (user != null) {
12             JOptionPane.showMessageDialog(null, "Login berhasil sebagai " + user.getRole());
13             if (user.getRole().equalsIgnoreCase("admin")) {
14                 new AdminMenuView(user);
15             } else {
16                 new OperatorMenuView(user);
17             }
18             return true;
19         } else {
20             JOptionPane.showMessageDialog(null, "Username atau password salah!", "Login Gagal", JOptionPane.ERROR_MESSAGE);
21             return false;
22         }
23     }
24 }
```

Kode program tersebut digunakan untuk mengatur proses login dalam sebuah aplikasi sistem parkir. Di dalamnya terdapat sebuah kelas bernama LoginController yang memiliki satu fungsi utama, yaitu login. Fungsi ini bertugas untuk memeriksa apakah username dan password yang dimasukkan pengguna sudah benar atau belum.

Pertama, data username dan password yang dikirim akan dicek ke database melalui kelas UserDAO. Jika data cocok, maka akan dikembalikan data pengguna (dalam bentuk objek User). Jika cocok, berarti login berhasil. Aplikasi kemudian akan menampilkan pesan bahwa login berhasil, dan mengecek peran (role) dari pengguna tersebut, apakah dia seorang admin atau operator. Kalau perannya admin, maka aplikasi akan membuka tampilan menu khusus admin. Tapi kalau bukan, akan diarahkan ke menu operator.

Sebaliknya, kalau username atau password salah (artinya data tidak ditemukan di database), maka aplikasi akan menampilkan pesan kesalahan yang menyatakan bahwa login gagal.

## Parking entry controller

```
● ● ●  
1 package main.java.com.parkingsystem.controllers;  
2 import main.java.com.parkingsystem.dao.TransactionDAO;  
3 import main.java.com.parkingsystem.models.ParkingTransaction;  
4 import javax.swing.*;  
5 import java.time.LocalDateTime;  
6  
7 public class ParkingEntryController {  
8     public static void handleEntry(String plate, String type) {  
9         LocalDateTime now = LocalDateTime.now();  
10        ParkingTransaction trx = new ParkingTransaction(plate, type, now);  
11        boolean result = TransactionDAO.insertEntry(trx);  
12        if (result) {  
13            JOptionPane.showMessageDialog(null, "Data parkir berhasil disimpan!");  
14        } else {  
15            JOptionPane.showMessageDialog(null, "Gagal menyimpan data parkir.");  
16        }  
17    }  
18 }
```

Kode program tersebut digunakan untuk mencatat kendaraan yang baru saja masuk ke area parkir. Di dalamnya terdapat kelas bernama `ParkingEntryController`, dan memiliki satu fungsi utama bernama `handleEntry`. Fungsi ini dijalankan saat ada kendaraan yang masuk dan menerima dua data penting: nomor plat kendaraan (`plate`) dan jenis kendaraan (`type`), misalnya mobil atau motor.

Begitu fungsi ini dipanggil, sistem akan mencatat waktu saat kendaraan masuk menggunakan `LocalDateTime.now()`, yaitu waktu saat itu juga. Kemudian, data plat, jenis kendaraan, dan waktu masuk tersebut dikemas menjadi satu objek bernama `ParkingTransaction`. Setelah itu, objek ini dikirim ke database lewat `TransactionDAO.insertEntry(trx)` untuk disimpan.

Kalau proses penyimpanan data berhasil, maka akan muncul pesan ke pengguna bahwa data parkir berhasil disimpan. Tapi kalau gagal, maka akan muncul pesan kesalahan bahwa data tidak bisa disimpan.

## Parkingexitcontroller

```
1 package main.java.com.parkingsystem.controllers;
2
3 import main.java.com.parkingsystem.dao.DatabaseConnection;
4 import main.java.com.parkingsystem.dao.TransactionDAO;
5 import main.java.com.parkingsystem.utils.ParkingCalculator;
6 import main.java.com.parkingsystem.utils.StrukPDFGenerator;
7
8 import javax.swing.*;
9 import java.awt.*;
10 import java.sql.*;
11 import java.time.LocalDateTime;
12 import java.time.temporal.ChronoUnit;
13
14 public class ParkingExitController {
15
16     public static void handleExit(String plateNumber, Component parentComponent) {
17         try (Connection conn = DatabaseConnection.getConnection()) {
18             String query = "SELECT entry_time, vehicle_type FROM transactions WHERE plate_number = ? AND exit_time IS NULL";
19             try (PreparedStatement stmt = conn.prepareStatement(query)) {
20                 stmt.setString(1, plateNumber);
21                 try (ResultSet rs = stmt.executeQuery()) {
22                     if (rs.next()) {
23                         Timestamp entryTS = rs.getTimestamp("entry_time");
24                         String vehicleType = rs.getString("vehicle_type");
25                         LocalDateTime entryTime = entryTS.toLocalDateTime();
26                         LocalDateTime exitTime = LocalDateTime.now();
27
28                         long duration = ChronoUnit.HOURS.between(entryTime, exitTime);
29                         if (duration <= 0) duration = 1;
30
31                         int rate = ParkingCalculator.calculateFee(vehicleType, entryTime, exitTime);
32                         boolean updated = TransactionDAO.updateExit(plateNumber, exitTime, rate);
33
34                         if (updated) {
35                             String message = "Kendaraan berhasil keluar.\n" +
36                                 "Durasi: " + duration + " jam\n" +
37                                 "Biaya: Rp " + String.format("%,d", rate).replace(',', '.');
38
39                         if (vehicleType.equalsIgnoreCase("Motor") && rate == 2500)
40                             message += "\nBiaya dibatasi maksimal Rp 25.000";
41                         else if (vehicleType.equalsIgnoreCase("Mobil") && rate == 30000)
42                             message += "\nBiaya dibatasi maksimal Rp 30.000";
43
44                             JOptionPane.showMessageDialog(parentComponent, message);
45
46                         // Cetak struk
47                         StrukPDFGenerator.generateStruk(plateNumber, vehicleType, entryTime, exitTime, rate);
48
49                         // Tutup jendela jika memungkinkan
50                         Window window = SwingUtilities.getWindowAncestor(parentComponent);
51                         if (window != null) window.dispose();
52                     } else {
53                         JOptionPane.showMessageDialog(parentComponent, "Gagal memproses keluar kendaraan.");
54                     }
55                 } else {
56                     JOptionPane.showMessageDialog(parentComponent, "Plat tidak ditemukan atau kendaraan sudah keluar.");
57                 }
58             }
59         }
60     } catch (SQLException e) {
61         JOptionPane.showMessageDialog(parentComponent, "Kesalahan database:\n" + e.getMessage());
62     } catch (Exception e) {
63         JOptionPane.showMessageDialog(parentComponent, "Kesalahan tak terduga:\n" + e.getMessage());
64     }
65 }
66 }
```

Kode program tersebut digunakan untuk menangani proses keluarnya kendaraan dari area parkir. Didalamnya terdapat kelas bernama ParkingExitController, dan fungsi utamanya adalah handleExit, yang akan dijalankan ketika seorang petugas memasukkan nomor plat kendaraan untuk proses keluar. Fungsi ini menerima dua parameter: nomor plat

kendaraan (plateNumber) dan komponen tampilan (parentComponent) yang digunakan untuk menampilkan pesan dialog ke pengguna.

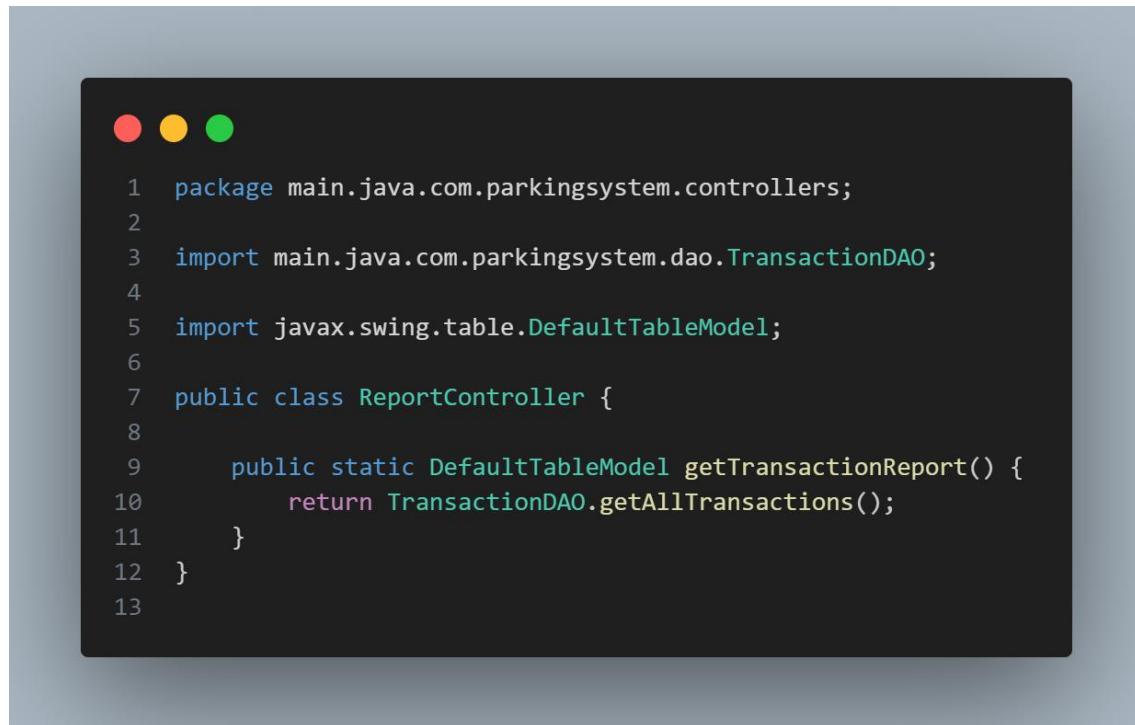
Pertama, program mencoba membuat koneksi ke database menggunakan DatabaseConnection.getConnection(). Setelah terkoneksi, sistem mencari data transaksi parkir berdasarkan nomor plat kendaraan yang belum memiliki waktu keluar (exit\_time IS NULL). Jika data ditemukan, program akan mengambil waktu masuk kendaraan (entry\_time) dan jenis kendaraannya (vehicle\_type). Lalu, waktu keluar akan dicatat sebagai waktu saat ini.

Setelah itu, program menghitung berapa lama kendaraan berada di dalam parkiran dengan menghitung selisih jam antara waktu masuk dan keluar. Jika hasilnya nol atau negatif (misalnya karena waktu masuk dan keluar terlalu dekat), maka secara otomatis durasi diatur menjadi minimal 1 jam. Kemudian, biaya parkir dihitung menggunakan bantuan kelas ParkingCalculator, yang menentukan tarif berdasarkan jenis kendaraan dan lamanya parkir.

Selanjutnya, sistem akan mencoba memperbarui data transaksi parkir di database dengan waktu keluar dan tarif yang sudah dihitung. Jika proses pembaruan berhasil, maka akan ditampilkan pesan sukses yang berisi informasi durasi parkir, biaya yang harus dibayar, serta catatan jika tarif sudah mencapai batas maksimal (misalnya maksimal Rp 25.000 untuk motor dan Rp 30.000 untuk mobil).

Setelah itu, sistem secara otomatis mencetak struk parkir dalam bentuk file PDF menggunakan StrukPDFGenerator.generateStruk, yang berisi semua detail parkir. Terakhir, jendela aplikasi yang terkait akan ditutup secara otomatis agar proses tampak selesai dengan rapi. Namun, jika data plat tidak ditemukan atau kendaraan sudah keluar sebelumnya, maka akan ditampilkan pesan error.

## Reportcontroller



```
1 package main.java.com.parkingsystem.controllers;
2
3 import main.java.com.parkingsystem.dao.TransactionDAO;
4
5 import javax.swing.table.DefaultTableModel;
6
7 public class ReportController {
8
9     public static DefaultTableModel getTransactionReport() {
10         return TransactionDAO.getAllTransactions();
11     }
12 }
13
```

Kode program tersebut digunakan untuk menampilkan laporan transaksi parkir. Didalamnya terdapat kelas bernama ReportController, dan hanya memiliki satu fungsi utama yaitu `getTransactionReport`. Fungsi ini bersifat static, artinya bisa langsung dipanggil tanpa harus membuat objek dari kelas ReportController.

Ketika fungsi `getTransactionReport` dipanggil, sistem akan mengambil seluruh data transaksi parkir yang tersimpan di database dengan memanggil `TransactionDAO.getAllTransactions()`. Hasilnya dikembalikan dalam bentuk `DefaultTableModel`, yaitu format data yang cocok untuk langsung ditampilkan dalam tabel di antarmuka pengguna (seperti di tabel Swing).

## 2.6.2 Dao

### Databaseconnection

```
● ● ●  
1 package main.java.com.parkingsystem.dao;  
2  
3 import java.sql.Connection;  
4 import java.sql.DriverManager;  
5 import java.sql.SQLException;  
6  
7 public class DatabaseConnection {  
8     private static final String URL = "jdbc:mysql://localhost:3306/parkir_db";  
9     private static final String USER = "root";  
10    private static final String PASSWORD = "";  
11  
12    public static Connection getConnection() throws SQLException {  
13        return DriverManager.getConnection(URL, USER, PASSWORD);  
14    }  
15 }
```

Kode program ini digunakan untuk mengatur koneksi ke database MySQL dalam sistem parkir, dipakai setiap kali aplikasi membutuhkan akses ke database untuk membaca atau menyimpan data seperti riwayat parkir, tarif, atau data pengguna.

Dalamnya terdapat kelas bernama DatabaseConnection yang berisi informasi penting untuk terhubung ke database, yaitu alamat database (URL), nama pengguna (USER), dan kata sandi (PASSWORD). Data tersebut disimpan sebagai variabel private static final agar tidak bisa diubah dari luar kelas secara sembarangan.

Dalamnya terdapat sebuah method bernama getConnection() yang berfungsi untuk membuka koneksi ke database menggunakan informasi yang sudah didefinisikan. Method ini akan dipanggil oleh bagian-bagian lain dari aplikasi yang memerlukan akses ke database, seperti saat login, menyimpan data kendaraan masuk, atau mengambil laporan parkir.

Agar aman dan terstruktur, kode ini tidak langsung menampilkan error, melainkan melemparkan exception (throws SQLException) kepada kelas yang memanggilnya, sehingga penanganan error bisa dilakukan di tempat lain, seperti pada kelas TestDB.

## Parkingrate

```
1 package main.java.com.parkingsystem.dao;
2
3 import java.sql.*;
4 import main.java.com.parkingsystem.models.ParkingRate;
5
6 public class ParkingRateDAO {
7
8     public static ParkingRate getRateByVehicleType(String type) {
9         try {
10             Connection conn = DatabaseConnection.getConnection();
11             String query = "SELECT * FROM parking_rates WHERE vehicle_type = ?";
12             PreparedStatement stmt = conn.prepareStatement(query);
13             stmt.setString(1, type);
14             ResultSet rs = stmt.executeQuery();
15
16             if (rs.next()) {
17                 String vehicleType = rs.getString("vehicle_type");
18                 int ratePerHour = rs.getInt("rate_per_hour");
19                 rs.close();
20                 stmt.close();
21                 conn.close();
22                 return new ParkingRate(vehicleType, ratePerHour);
23             }
24
25             rs.close();
26             stmt.close();
27             conn.close();
28         } catch (SQLException e) {
29             System.out.println("Gagal ambil tarif: " + e.getMessage());
30         }
31
32         return new ParkingRate(type, 3000);
33     }
34 }
35
```

Kode program ini digunakan untuk mengambil data tarif parkir berdasarkan jenis kendaraan dari database, biasanya dipakai saat sistem ingin mengetahui berapa biaya yang harus dibayar oleh pengguna berdasarkan tipe kendaraan seperti "motor" atau "mobil".

Dalamnya terdapat kelas bernama `ParkingRateDAO` yang berfungsi sebagai data access object, yaitu bagian dari aplikasi yang bertugas berkomunikasi langsung dengan database. Kelas ini memiliki sebuah method bernama `getRateByVehicleType()` yang menerima parameter `type` (misalnya: "mobil" atau "motor") dan akan mencari data tarif yang sesuai di dalam tabel `parking\_rates`.

Saat method dipanggil, ia akan membuka koneksi ke database menggunakan

`DatabaseConnection.getConnection()`, lalu menjalankan perintah SQL untuk mengambil data dengan query `SELECT \* FROM parking\_rates WHERE vehicle\_type = ?`. Nilai `?` diisi dengan jenis kendaraan menggunakan `setString(1, type)`.

Jika data ditemukan (`rs.next()`), maka program membaca nilai dari kolom `vehicle\_type` dan `rate\_per\_hour`, kemudian membuat dan mengembalikan objek `ParkingRate` dengan data tersebut. Setelah selesai, semua sumber daya seperti `ResultSet`, `PreparedStatement`, dan `Connection` ditutup agar tidak membuang memori atau koneksi.

Namun jika terjadi kesalahan saat mengambil data atau tidak ada data ditemukan, maka program akan menampilkan pesan error dan mengembalikan objek `ParkingRate` dengan nilai default, yaitu tarif sebesar 3000 per ja sebagai fallback.

## Transaction

```
1 package main.java.com.parkingsystem.dao;
2
3 import main.java.com.parkingsystem.models.ParkingTransaction;
4
5 import java.sql.*;
6 import java.time.LocalDateTime;
7
8 import javax.swing.table.DefaultTableModel;
9
10 public class TransactionDAO {
11
12     public static boolean insertEntry(ParkingTransaction trx) {
13         boolean success = false;
14
15         try {
16             Connection conn = DatabaseConnection.getConnection();
17             if (conn == null) return false;
18
19             String query = "INSERT INTO transactions (plate_number, vehicle_type, entry_time) VALUES (?, ?, ?)";
20             PreparedStatement stmt = conn.prepareStatement(query);
21
22             stmt.setString(1, trx.getPlateNumber());
23             stmt.setString(2, trx.getVehicleType());
24             stmt.setTimestamp(3, Timestamp.valueOf(trx.getEntryTime()));
25
26             int rows = stmt.executeUpdate();
27             success = (rows > 0);
28
29             stmt.close();
30             conn.close();
31         } catch (SQLException e) {
32             System.out.println("Insert error: " + e.getMessage());
33         }
34
35         return success;
36     }
37
38     public static boolean updateExit(String plateNumber, LocalDateTime exitTime, int fee) {
39         boolean success = false;
40
41         try {
42             Connection conn = DatabaseConnection.getConnection();
43             if (conn == null) return false;
44
45             String query = "UPDATE transactions SET exit_time = ?, total_fee = ? WHERE plate_number = ? AND exit_time IS NULL";
46             PreparedStatement stmt = conn.prepareStatement(query);
47             stmt.setTimestamp(1, Timestamp.valueOf(exitTime));
48             stmt.setInt(2, fee);
49             stmt.setString(3, plateNumber);
50
51             int rows = stmt.executeUpdate();
52             success = (rows > 0);
53
54             stmt.close();
55             conn.close();
56         } catch (SQLException e) {
57             System.out.println("Update error: " + e.getMessage());
58         }
59
60         return success;
61     }
62
63     public static DefaultTableModel getAllTransactions() {
64         String[] columns = {"No", "Plat Nomor", "Jenis", "Masuk", "Keluar", "Biaya"};
65         DefaultTableModel model = new DefaultTableModel(null, columns);
66
67         try {
68             Connection conn = DatabaseConnection.getConnection();
69             String query = "SELECT * FROM transactions ORDER BY id DESC";
70             PreparedStatement stmt = conn.prepareStatement(query);
71             ResultSet rs = stmt.executeQuery();
72
73             int no = 1;
74             while (rs.next()) {
75                 String plate = rs.getString("plate_number");
76                 String type = rs.getString("vehicle_type");
77                 String masuk = rs.getString("entry_time");
78                 String keluar = rs.getString("exit_time");
79                 String fee = rs.getString("total_fee");
80
81                 Object[] row = {no++, plate, type, masuk, keluar, fee};
82                 model.addRow(row);
83             }
84
85             rs.close();
86             stmt.close();
87             conn.close();
88         } catch (SQLException e) {
89             System.out.println("Load transaksi error: " + e.getMessage());
90         }
91         return model;
92     }
93
94     public static boolean deleteByPlate(String plate) {
95         try {
96             Connection conn = DatabaseConnection.getConnection();
97             String query = "DELETE FROM transactions WHERE plate_number = ?";
98             PreparedStatement stmt = conn.prepareStatement(query);
99             stmt.setString(1, plate);
100            int result = stmt.executeUpdate();
101            stmt.close();
102            conn.close();
103            return result > 0;
104        } catch (SQLException e) {
105            System.out.println("Hapus gagal: " + e.getMessage());
106            return false;
107        }
108    }
109}
```

Kode program ini digunakan untuk mengelola data transaksi parkir , seperti mencatat kendaraan yang masuk, mengupdate waktu keluar dan biaya parkir, menampilkan seluruh riwayat transaksi, serta menghapus data transaksi jika dibutuhkan. Kelas ini dipakai saat sistem parkir sedang menjalankan operasi utamanya, seperti proses kendaraan masuk, keluar, atau melihat laporan.

Didalamnya terdapat kelas bernama TransactionDAO yang merupakan singkatan dari Data Access Object . Didalamnya terdapat beberapa method yaitu:

1. InsertEntry(ParkingTransaction trx)

Digunakan untuk menyimpan data kendaraan yang baru masuk ke dalam database. Ia menerima sebuah objek ParkingTransaction yang berisi informasi plat nomor, jenis kendaraan, dan waktu masuk. Data tersebut kemudian dimasukkan ke tabel transactions menggunakan perintah SQL INSERT. Jika berhasil, method akan mengembalikan nilai true.

2. UpdateExit(String plateNumber, LocalDateTime exitTime, int fee)

Digunakan ketika kendaraan keluar, untuk memperbarui data transaksi dengan waktu keluar dan biaya parkir. Method ini mencari transaksi yang belum selesai (yang exit\_time-nya masih kosong) berdasarkan plat nomor, lalu mengisi waktu keluar dan total biaya menggunakan perintah SQL UPDATE. Jika update berhasil dilakukan, maka method akan mengembalikan nilai true.

3. GetAllTransactions()

Digunakan untuk mengambil semua data transaksi dari database dan menampilkannya dalam bentuk tabel di antarmuka aplikasi (GUI). Ia membuat sebuah objek DefaultTableModel yang nanti bisa ditampilkan di JTable. Setiap baris data dari tabel transactions di database akan diubah menjadi baris pada model tabel ini.

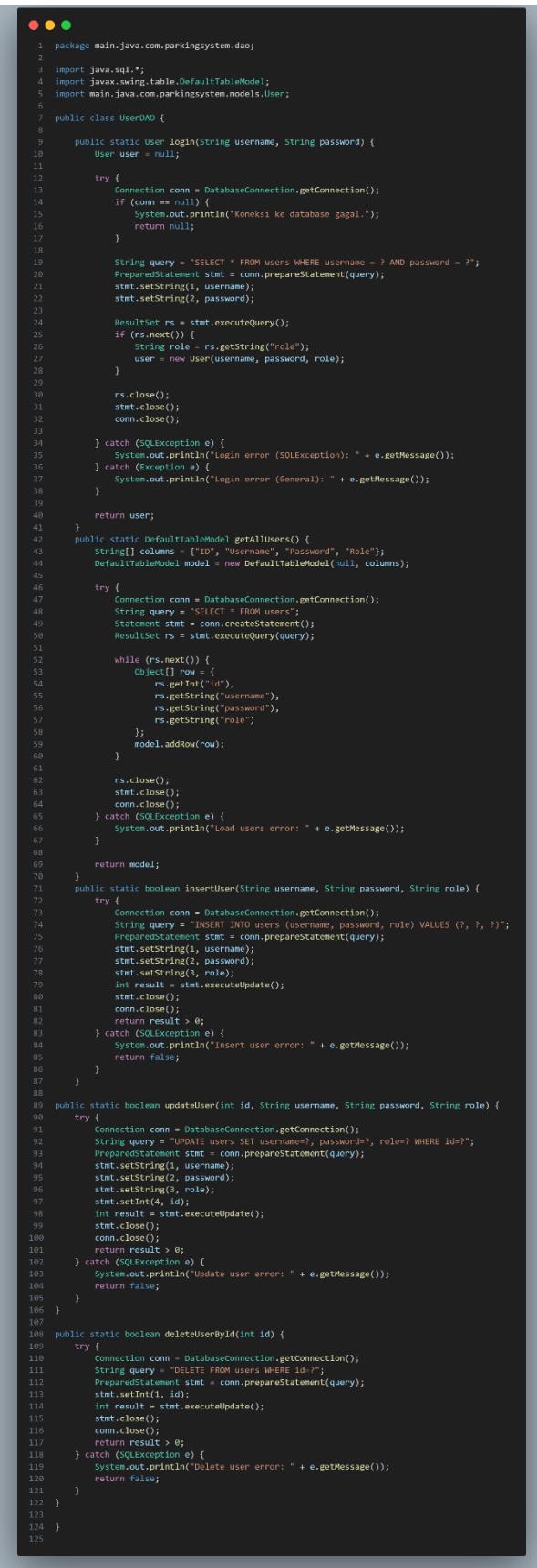
4. DeleteByPlate(String plate)

Digunakan untuk menghapus data transaksi berdasarkan plat nomor. Ini berguna misalnya jika ada data yang salah atau ingin dihapus secara manual. Method ini akan mengembalikan nilai true jika penghapusan berhasil.

Seluruh method di atas bekerja dengan membuka koneksi ke database menggunakan kelas DatabaseConnection, lalu menjalankan perintah SQL sesuai kebutuhan. Setelah selesai, semua sumber daya seperti ResultSet, PreparedStatement, dan Connection ditutup agar tidak membuang sumber daya sistem.

Setiap method juga sudah dilengkapi penanganan kesalahan (try-catch) untuk menangani error yang mungkin terjadi saat berinteraksi dengan database, sehingga aplikasi tidak langsung crash dan bisa memberi informasi error yang jelas.

## Userdao



```
1 package main.java.com.parkingsystem.dao;
2
3 import java.sql.*;
4 import javax.swing.table.DefaultTableModel;
5 import main.java.com.parkingsystem.models.User;
6
7 public class UserDAO {
8
9     public static User login(String username, String password) {
10         User user = null;
11
12         try {
13             Connection conn = DatabaseConnection.getConnection();
14             if (conn == null) {
15                 System.out.println("Koneksi ke database gagal.");
16                 return null;
17             }
18
19             String query = "SELECT * FROM users WHERE username = ? AND password = ?";
20             PreparedStatement stat = conn.prepareStatement(query);
21             stat.setString(1, username);
22             stat.setString(2, password);
23
24             ResultSet rs = stat.executeQuery();
25             if (rs.next()) {
26                 String role = rs.getString("role");
27                 user = new User(username, password, role);
28             }
29
30             rs.close();
31             stat.close();
32             conn.close();
33
34         } catch (SQLException e) {
35             System.out.println("Login error (SQLException): " + e.getMessage());
36         } catch (Exception e) {
37             System.out.println("Login error (General): " + e.getMessage());
38         }
39
40         return user;
41     }
42     public static DefaultTableModel getAllUsers() {
43         String[] columns = {"ID", "Username", "Password", "Role"};
44         DefaultTableModel model = new DefaultTableModel(null, columns);
45
46         try {
47             Connection conn = DatabaseConnection.getConnection();
48             String query = "SELECT * FROM users";
49             Statement stat = conn.createStatement();
50             ResultSet rs = stat.executeQuery(query);
51
52             while (rs.next()) {
53                 Object[] row = {
54                     rs.getInt("id"),
55                     rs.getString("username"),
56                     rs.getString("password"),
57                     rs.getString("role")
58                 };
59                 model.addRow(row);
60             }
61
62             rs.close();
63             stat.close();
64             conn.close();
65         } catch (SQLException e) {
66             System.out.println("Load users error: " + e.getMessage());
67         }
68
69         return model;
70     }
71     public static boolean insertUser(String username, String password, String role) {
72         try {
73             Connection conn = DatabaseConnection.getConnection();
74             String query = "INSERT INTO users (username, password, role) VALUES (?, ?, ?)";
75             PreparedStatement stat = conn.prepareStatement(query);
76             stat.setString(1, username);
77             stat.setString(2, password);
78             stat.setString(3, role);
79             int result = stat.executeUpdate();
80             stat.close();
81             conn.close();
82             return result > 0;
83         } catch (SQLException e) {
84             System.out.println("Insert user error: " + e.getMessage());
85             return false;
86         }
87     }
88     public static boolean updateUser(int id, String username, String password, String role) {
89         try {
90             Connection conn = DatabaseConnection.getConnection();
91             String query = "UPDATE users SET username=?, password=?, role=? WHERE id=?";
92             PreparedStatement stat = conn.prepareStatement(query);
93             stat.setString(1, username);
94             stat.setString(2, password);
95             stat.setString(3, role);
96             stat.setInt(4, id);
97             int result = stat.executeUpdate();
98             stat.close();
99             conn.close();
100            return result > 0;
101        } catch (SQLException e) {
102            System.out.println("Update user error: " + e.getMessage());
103            return false;
104        }
105    }
106    public static boolean deleteUserById(int id) {
107        try {
108            Connection conn = DatabaseConnection.getConnection();
109            String query = "DELETE FROM users WHERE id=?";
110            PreparedStatement stat = conn.prepareStatement(query);
111            stat.setInt(1, id);
112            int result = stat.executeUpdate();
113            stat.close();
114            conn.close();
115            return result > 0;
116        } catch (SQLException e) {
117            System.out.println("Delete user error: " + e.getMessage());
118            return false;
119        }
120    }
121 }
122 }
123 }
124 }
```

Kode program ini digunakan untuk mengelola data pengguna (user) dalam sistem parkir, biasanya dipakai saat admin atau petugas ingin login, melihat daftar user, menambahkan, mengedit, atau menghapus akun pengguna.

Dalamnya terdapat kelas bernama UserDAO yang berfungsi sebagai Data Access Object , yaitu bagian dari aplikasi yang bertanggung jawab untuk berinteraksi langsung dengan database. Didalamnya terdapat beberapa method yaitu:

1. `Login(String username, String password)`

Digunakan ketika seorang pengguna ingin masuk ke aplikasi. Method ini mencari data user di tabel users berdasarkan username dan password yang dimasukkan. Jika cocok, maka akan dibuat objek User baru dengan informasi tersebut (termasuk role, seperti "Admin" atau "Petugas") dan dikembalikan sebagai hasil login. Jika tidak ditemukan, maka akan mengembalikan nilai null.

2. `GetAllUsers()`

Digunakan untuk mengambil seluruh data pengguna dari database dan menampilkannya dalam bentuk tabel di antarmuka GUI. Ia membuat sebuah objek DefaultTableModel yang nanti bisa ditampilkan di komponen JTable. Setiap baris dari tabel users di database akan menjadi baris data pada model tabel ini.

`insertUser(String username, String password, String role)`

Digunakan untuk menambahkan data pengguna baru ke dalam database. Method ini menerima input berupa username, password, dan role, lalu menyimpannya ke tabel users menggunakan perintah SQL INSERT. Jika berhasil, method akan mengembalikan nilai true.

3. `UpdateUser(int id, String username, String password, String role)`

Digunakan untuk memperbarui data pengguna yang sudah ada berdasarkan ID-nya. Method ini mengubah informasi seperti username, password, atau role sesuai dengan data baru yang dimasukkan. Jika proses update berhasil dilakukan, maka method akan mengembalikan nilai true.

4. `DeleteUserById(int id)`

Digunakan untuk menghapus data pengguna berdasarkan ID-nya dari database. Ini berguna jika ada akun yang tidak aktif atau ingin dihapus oleh admin. Jika penghapusan berhasil, method akan mengembalikan nilai true.

Setiap method bekerja dengan membuka koneksi ke database menggunakan kelas DatabaseConnection, menjalankan query SQL sesuai fungsinya, lalu menutup seperti ResultSet, PreparedStatement, dan Connection agar tidak membuang memori atau koneksi.

Selain itu, semua operasi database juga telah dilengkapi dengan penanganan kesalahan (try-catch) untuk menangani error yang mungkin terjadi, sehingga aplikasi tetap stabil dan memberikan informasi error yang jelas.

### 2.6.3 Models

#### Parkingrate

```
1 package main.java.com.parkingsystem.models;
2
3 public class ParkingRate {
4     private String vehicleType;
5     private int ratePerHour;
6
7     public ParkingRate(String vehicleType, int ratePerHour) {
8         this.vehicleType = vehicleType;
9         this.ratePerHour = ratePerHour;
10    }
11
12    public String getVehicleType() {
13        return vehicleType;
14    }
15
16    public int getRatePerHour() {
17        return ratePerHour;
18    }
19 }
20
```

Kode program tersebut digunakan untuk menyimpan informasi tarif parkir berdasarkan jenis kendaraan dan dipakai ketika sistem perlu mengetahui berapa biaya yang harus dibayar pengguna parkir. Didalamnya terdapat kelas bernama ParkingRate dan berisi dua variabel utama: vehicleType untuk menyimpan jenis kendaraan (seperti "Motor" atau

"Mobil"), dan ratePerHour untuk menyimpan besaran tarif per jam dalam bentuk angka. Pertama, program memiliki sebuah konstruktor yang menerima dua parameter saat membuat objek baru (jenis kendaraan dan tarif per jamnya). Data ini kemudian disimpan dalam objek untuk bisa diakses nanti. Untuk mengambil informasi yang sudah disimpan, kelas ini menyediakan dua method sederhana: getVehicleType() yang mengembalikan jenis kendaraan dalam bentuk teks, dan getRatePerHour() yang mengembalikan angka tarif parkir per jam.

Kelas ini biasanya digunakan bersama komponen lain dalam sistem parkir. Misalnya, ketika kendaraan hendak keluar dari parkiran, sistem akan mencari objek ParkingRate yang sesuai dengan jenis kendaraan tersebut, lalu menghitung total biaya berdasarkan lama parkir dan tarif per jam yang tersimpan. Karena modelnya sederhana dan jelas, memudahkan pengelolaan tarif parkir yang mungkin berbeda-beda untuk setiap jenis kendaraan.

## Parkingtransaction

```
1 package main.java.com.parkingsystem.models;
2
3 import java.time.LocalDateTime;
4
5 public class ParkingTransaction {
6     private String plateNumber;
7     private String vehicleType;
8     private LocalDateTime entryTime;
9
10    public ParkingTransaction(String plateNumber, String vehicleType, LocalDateTime entryTime) {
11        this.plateNumber = plateNumber;
12        this.vehicleType = vehicleType;
13        this.entryTime = entryTime;
14    }
15
16    public String getPlateNumber() {
17        return plateNumber;
18    }
19
20    public String getVehicleType() {
21        return vehicleType;
22    }
23
24    public LocalDateTime getEntryTime() {
25        return entryTime;
26    }
27}
28}
```

Kode program tersebut digunakan untuk mencatat transaksi parkir kendaraan, dipakai ketika ada kendaraan baru masuk ke area parkir. Didalamnya terdapat kelas bernama ParkingTransaction dan berisi tiga variabel yaitu plateNumber untuk mencatat nomor plat kendaraan, vehicleType untuk menentukan jenis kendaraan, entryTime yang mencatat tanggal dan waktu tepat saat kendaraan masuk.

Ketika membuat transaksi parkir baru, sistem akan memanggil konstruktor kelas ini dengan ketiga variabel tersebut. Data ini kemudian disimpan dalam objek untuk keperluan pencatatan. Untuk mengakses data yang sudah disimpan, kelas ini menyediakan tiga method yaitu getPlateNumber() untuk mengetahui nomor plat kendaraan, getVehicleType() untuk melihat jenis kendaraan, getEntryTime() untuk mengecek waktu masuk kendaraan.

## User

```
1 package main.java.com.parkingsystem.models;
2
3 public class User {
4     private String username;
5     private String password;
6     private String role;
7
8     public User(String username, String password, String role) {
9         this.username = username;
10        this.password = password;
11        this.role = role;
12    }
13
14    public String getUsername() {
15        return username;
16    }
17
18    public String getPassword() {
19        return password;
20    }
21
22    public String getRole() {
23        return role;
24    }
25
26 }
27
```

Kode program ini digunakan untuk mengelola data pengguna sistem parkir, dipakai ketika ada admin atau petugas yang perlu login ke aplikasi. Didalamnya terdapat kelas bernama User dan berisi tiga variabel yaitu username untuk menyimpan nama pengguna, password untuk menyimpan kata sandi pengguna, role untuk menentukan hak akses pengguna (seperti "Admin" atau "Petugas").

Saat membuat user baru, sistem akan memanggil konstruktor kelas ini dengan ketiga variabel tersebut. Data ini kemudian disimpan dalam objek User. Untuk mengakses data user, kelas tersebut mepunyai tiga method yaitu getUsername() untuk mengetahui nama pengguna, getPassword() untuk password, getRole() untuk mengecek jenis hak akses pengguna.

Data user yang disimpan dalam objek ini bersifat tetap dan tidak bisa diubah langsung dari luar kelas. Ini menjaga keamanan sistem agar tidak ada yang bisa sembarangan mengubah data pengguna.

## Vehicle

```
● ● ●  
1 package main.java.com.parkingsystem.models;  
2  
3 public class Vehicle {  
4     private String plateNumber;  
5     private String vehicleType;  
6  
7     public Vehicle(String plateNumber, String vehicleType) {  
8         this.plateNumber = plateNumber;  
9         this.vehicleType = vehicleType;  
10    }  
11  
12    public String getPlateNumber() {  
13        return plateNumber;  
14    }  
15  
16    public String getVehicleType() {  
17        return vehicleType;  
18    }  
19}  
20
```

Kode program ini digunakan untuk menyimpan informasi kendaraan yang parkir, dipakai setiap kali ada mobil atau motor masuk ke tempat parkir. Di dalamnya terdapat kelas bernama Vehicle dan berisi dua variabel yaitu plateNumber untuk mencatat nomor plat kendaraan, vehicleType untuk menentukan jenis kendaraan. Ketika ada kendaraan baru masuk, sistem akan membuat objek Vehicle dengan mengisi nomor plat dan jenis kendaraannya. Data ini kemudian disimpan dan bisa diambil kembali saat dibutuhkan menggunakan getPlateNumber() untuk melihat nomor plat kendaraan, getVehicleType() untuk mengetahui jenis kendaraan.

#### 2.6.4 Utils

##### Datetimeutil



```
1 package main.java.com.parkingsystem.utils;
2
3 import java.time.LocalDateTime;
4 import java.time.format.DateTimeFormatter;
5
6 public class DateUtil {
7
8     private static final DateTimeFormatter FORMATTER = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");
9
10    public static String format(LocalDateTime dateTime) {
11        return dateTime != null ? dateTime.format(FORMATTER) : "-";
12    }
13
14    public static LocalDateTime parse(String dateTimeStr) {
15        return LocalDateTime.parse(dateTimeStr, FORMATTER);
16    }
17}
18
```

Kode program tersebut digunakan untuk membantu dalam mengatur tampilan dan pengolahan waktu dan tanggal di aplikasi sistem parkir. Di dalamnya terdapat kelas bernama DateUtil, dan isinya adalah fungsi-fungsi yang berhubungan dengan format tanggal dan waktu. Setelah import terdapat format tanggal yang sudah ditentukan, yaitu "dd-MM-yyyy HH:mm" (contohnya: 08-06-2025 14:30). Format ini digunakan supaya semua tanggal di aplikasi punya tampilan yang konsisten dan mudah dibaca.

Fungsi pertama, format, digunakan untuk mengubah data waktu (dalam bentuk LocalDateTime) menjadi teks yang rapi sesuai format tadi. Kalau datanya kosong atau null, maka akan ditampilkan tanda "-". Sedangkan fungsi kedua, parse, digunakan untuk kebalikannya, yaitu mengubah teks tanggal (dalam format yang sudah ditentukan)

menjadi data waktu (LocalDateTime). Ini berguna untuk mengambil data dari file atau inputan pengguna dan ingin mengolahnya kembali sebagai waktu asli/real time.

## Excelexporter

```
1 package main.java.com.parkingsystem.utils;
2
3 import org.apache.poi.ss.usermodel.*;
4 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
5
6 import javax.swing.*;
7 import javax.swing.table.TableModel;
8 import java.io.FileOutputStream;
9 import java.io.IOException;
10
11 public class ExcelExporter {
12
13     public static void exportTableToExcel(TableModel model, String filePath) {
14         try (Workbook workbook = new XSSFWorkbook()) {
15             Sheet sheet = workbook.createSheet("Laporan Parkir");
16
17             Font headerFont = workbook.createFont();
18             headerFont.setBold(true);
19             headerFont.setFontHeightInPoints((short) 12);
20             headerFont.setColor(IndexedColors.WHITE.getIndex());
21
22             CellStyle headerCellStyle = workbook.createCellStyle();
23             headerCellStyle.setFillForegroundColor(IndexedColors.DARK_BLUE.getIndex());
24             headerCellStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);
25             headerCellStyle.setFont(headerFont);
26             headerCellStyle.setAlignment(HorizontalAlignment.CENTER);
27
28             Row headerRow = sheet.createRow(0);
29             for (int col = 0; col < model.getColumnCount(); col++) {
30                 Cell cell = headerRow.createCell(col);
31                 cell.setCellValue(model.getColumnName(col));
32                 cell.setCellStyle(headerCellStyle);
33             }
34
35             for (int rowIdx = 0; rowIdx < model.getRowCount(); rowIdx++) {
36                 Row row = sheet.createRow(rowIdx + 1);
37                 for (int col = 0; col < model.getColumnCount(); col++) {
38                     Object value = model.getValueAt(rowIdx, col);
39                     row.createCell(col).setCellValue(value == null ? "" : value.toString());
40                 }
41             }
42
43             for (int i = 0; i < model.getColumnCount(); i++) {
44                 sheet.autoSizeColumn(i);
45             }
46
47             try (FileOutputStream fileOut = new FileOutputStream(filePath)) {
48                 workbook.write(fileOut);
49             }
50
51             JOptionPane.showMessageDialog(null, "Laporan berhasil dieksport ke Excel.");
52         } catch (IOException e) {
53             JOptionPane.showMessageDialog(null, "Gagal mengeksport ke Excel: " + e.getMessage());
54         }
55     }
56 }
57 }
```

Kode program tersebut digunakan untuk mengekspor data dari tabel ke dalam file Excel, dipakai saat pengguna ingin menyimpan laporan parkir dalam format .xlsx. Kelas ini bernama ExcelExporter dan berisi satu metode bernama exportTableToExcel yang menerima dua parameter: model, yaitu data dari tabel, dan filePath, yaitu lokasi atau nama file Excel yang akan dibuat.

Pertama, program membuat objek workbook baru dengan bantuan library Apache POI, yang memang khusus digunakan untuk membuat dan mengolah file Excel. Kemudian, dibuat satu sheet bernama "Laporan Parkir". Di awal sheet, program membuat baris pertama sebagai header (judul kolom), lalu memberi gaya pada header agar terlihat lebih rapi. Setelah header dibuat, program melanjutkan dengan mengisi data baris demi baris berdasarkan isi dari tabel. Setiap sel diisi dengan data yang sesuai dari TableModel, dan jika ada nilai yang kosong (null), maka akan diganti dengan string kosong supaya tidak error. Agar tampilan Excel-nya lebih enak dilihat, semua kolom juga diatur otomatis lebarnya sesuai isi.

Setelah semua data selesai dimasukkan, program menyimpan file Excel ke lokasi yang ditentukan melalui filePath menggunakan FileOutputStream. Jika proses berhasil, akan muncul pesan popup yang memberi tahu bahwa laporan berhasil diekspor. Tetapi kalau terjadi kesalahan maka akan muncul pesan error.

## Parkingcalculator

```
1 package main.java.com.parkingsystem.utils;
2
3 import main.java.com.parkingsystem.dao.ParkingRateDAO;
4 import main.java.com.parkingsystem.models.ParkingRate;
5
6 import java.time.LocalDateTime;
7 import java.time.temporal.ChronoUnit;
8
9 public class ParkingCalculator {
10     public static int calculateFee(String vehicleType, LocalDateTime entry, LocalDateTime exit) {
11         long duration = ChronoUnit.HOURS.between(entry, exit);
12         if (duration <= 0) duration = 1;
13
14         int total;
15
16         if (vehicleType.equalsIgnoreCase("Motor")) {
17             total = 2000 + (int) ((duration - 1) * 2000);
18             if (total > 25000) total = 25000;
19         } else if (vehicleType.equalsIgnoreCase("Mobil")) {
20             total = 3000 + (int) ((duration - 1) * 3000);
21             if (total > 30000) total = 30000;
22         } else {
23             // fallback untuk jenis kendaraan lain, pakai tarif per jam dari DB
24             ParkingRate rate = ParkingRateDAO.getRateByVehicleType(vehicleType);
25             total = (int) duration * rate.getRatePerHour();
26         }
27
28         return total;
29     }
30 }
31
```

Kode program tersebut digunakan untuk menghitung biaya parkir berdasarkan jenis kendaraan dan lamanya parkir. Didalamnya terdapat kelas bernama ParkingCalculator, dan di dalamnya ada satu fungsi bernama calculateFee yang akan dipanggil setiap kali sistem ingin tahu berapa biaya yang harus dibayar oleh kendaraan saat keluar dari parkiran.

Fungsi ini menerima tiga data: jenis kendaraan (vehicleType), waktu masuk (entry), dan waktu keluar (exit). Pertama, sistem menghitung durasi parkir dalam satuan jam, dengan menggunakan ChronoUnit.HOURS.between. Kalau hasilnya nol atau negatif (misalnya kendaraan hanya parkir beberapa menit), maka durasi dianggap minimal 1 jam agar tetap dikenakan biaya.

Selanjutnya, biaya dihitung berdasarkan jenis kendaraannya. Kalau jenisnya motor, maka dikenakan tarif Rp2.000 untuk jam pertama, lalu Rp2.000 lagi untuk setiap jam berikutnya. Tapi kalau totalnya melebihi Rp25.000, maka tarifnya dibatasi hanya sampai Rp25.000. Untuk mobil, perhitungan serupa, hanya saja tarifnya Rp3.000 per jam, dan maksimal dibatasi Rp30.000.

Kalau jenis kendaraan bukan motor atau mobil (misalnya kendaraan lain seperti truk atau jenis baru yang ditambahkan), maka tarifnya tidak dihitung secara manual, tapi diambil dari database melalui `ParkingRateDAO.getRateByVehicleType`, lalu dikalikan dengan jumlah jam parkir. Jadi sistem tetap fleksibel dan bisa menyesuaikan dengan tarif yang disimpan di database.

## Strukpdfgenerator

```
1 package main.java.com.parkingsystem.utils;
2
3 import com.itextpdf.text.*;
4 import com.itextpdf.text.pdf.PdfWriter;
5
6 import javax.swing.*;
7 import java.awt.Desktop;
8 import java.io.File;
9 import java.io.FileOutputStream;
10 import java.time.LocalDateTime;
11 import java.time.format.DateTimeFormatter;
12
13 public class StrukPDFGenerator {
14
15     public static void generateStruk(String plate, String type, LocalDateTime entry, LocalDateTime exit, int fee) {
16         try {
17             // Format waktu
18             DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");
19
20             // Buat folder "struk" jika belum ada
21             File folder = new File("struk");
22             if (!folder.exists()) {
23                 folder.mkdirs();
24             }
25
26             // Nama file dan path
27             String fileName = "Struk_" + plate + "_" + System.currentTimeMillis() + ".pdf";
28             File outputFile = new File(folder, fileName);
29
30             // Siapkan PDF Document
31             Document document = new Document();
32             PdfWriter.getInstance(document, new FileOutputStream(outputFile));
33             document.open();
34
35             // Logo di atas (gunakan path lokal logo.png dari resources)
36             String logoPath = "src/main/resources/logo.png"; // atau sesuaikan path-mu
37             File logoFile = new File(logoPath);
38             if (logoFile.exists()) {
39                 Image logo = Image.getInstance(logoPath);
40                 logo.scaleAbsolute(80, 80); // Ukuran logo
41                 logo.setAlignment(Element.ALIGN_CENTER);
42                 document.add(logo);
43             }
44
45             // Judul
46             Font titleFont = new Font(Font.FontFamily.HELVETICA, 18, Font.BOLD);
47             Paragraph title = new Paragraph("SMARTPARK - STRUK PARKIR", titleFont);
48             title.setAlignment(Element.ALIGN_CENTER);
49             title.setSpacingAfter(20);
50             document.add(title);
51
52             // Isi data
53             document.add(new Paragraph("Plat Nomor : " + plate));
54             document.add(new Paragraph("Jenis Kendaraan: " + type));
55             document.add(new Paragraph("Waktu Masuk : " + entry.format(formatter)));
56             document.add(new Paragraph("Waktu Keluar : " + exit.format(formatter)));
57             document.add(new Paragraph("Durasi Parkir : " + getDuration(entry, exit) + " jam"));
58             document.add(new Paragraph("Biaya Parkir : Rp " + fee));
59             document.add(new Paragraph(""));
60
61             // Info tarif
62             if (type.equalsIgnoreCase("Motor")) {
63                 document.add(new Paragraph("* Tarif: Rp 2.000 per jam"));
64                 document.add(new Paragraph("* Maksimal bayar: Rp 25.000"));
65             } else if (type.equalsIgnoreCase("Mobil")) {
66                 document.add(new Paragraph("* Tarif: Rp 3.000 per jam"));
67                 document.add(new Paragraph("* Maksimal bayar: Rp 30.000"));
68             } else {
69                 document.add(new Paragraph("* Tarif mengikuti sistem database"));
70             }
71
72             // Footer
73             Paragraph footer = new Paragraph("\nTerima kasih telah menggunakan SmartPark!", new Font(Font.FontFamily.HELVETICA, 10, Font.ITALIC));
74             footer.setAlignment(Element.ALIGN_CENTER);
75             footer.setSpacingBefore(20);
76             document.add(footer);
77
78             document.close();
79
80             // Buka file otomatis setelah disimpan
81             if (Desktop.isDesktopSupported()) {
82                 Desktop.getDesktop().open(outputFile);
83             }
84
85             System.out.println("Struk berhasil disimpan di: " + outputFile.getAbsolutePath());
86
87         } catch (Exception e) {
88             JOptionPane.showMessageDialog(null, "Gagal mencetak struk: " + e.getMessage());
89             e.printStackTrace();
90         }
91     }
92
93     private static long getDuration(LocalDateTime entry, LocalDateTime exit) {
94         long durasi = java.time.temporal.ChronoUnit.HOURS.between(entry, exit);
95         return durasi <= 0 ? 1 : durasi;
96     }
97 }
98 }
```

Kode program tersebut digunakan untuk membuat struk parkir dalam format PDF, dipakai saat pengguna ingin mencetak bukti pembayaran parkir. Didalamnya terdapat kelas bernama StrukPDFGenerator dan berisi satu metode utama bernama generateStruk yang menerima beberapa parameter, yaitu nomor plat kendaraan, jenis kendaraan, waktu masuk, waktu keluar, serta biaya parkir.

Pertama, program memformat waktu masuk dan keluar menggunakan DateTimeFormatter agar lebih mudah dibaca. Kemudian, ia membuat folder bernama "struk" jika belum ada, sebagai tempat menyimpan file PDF. Nama file PDF-nya terdiri dari kata "Struk", nomor plat kendaraan, dan timestamp untuk memastikan setiap struk unik dan tidak tertimpa.

Setelah itu, program membuat dokumen PDF menggunakan library iTextPDF. Jika ada file logo di folder resources, logo akan ditampilkan di bagian atas struk dengan ukuran yang sudah disesuaikan. Judul "SMARTPARK - STRUK PARKIR" ditulis dengan font tebal dan ukuran besar di tengah halaman. Di bawahnya, semua detail transaksi parkir ditampilkan, seperti nomor plat, jenis kendaraan, waktu masuk dan keluar, durasi parkir (dihitung dalam jam), serta total biaya.

Program juga menambahkan informasi tarif parkir berdasarkan jenis kendaraan. Misalnya, untuk motor tarifnya Rp 2.000 per jam (maksimal Rp 25.000), sedangkan mobil Rp 3.000 per jam (maksimal Rp 30.000). Di bagian bawah, ada pesan "Terima kasih telah menggunakan SmartPark!" sebagai penutup.

Setelah struk selesai dibuat, program akan mencoba membukanya secara otomatis menggunakan Desktop.getDesktop().open() jika sistem mendukung. Jika berhasil, struk akan langsung terbuka di aplikasi PDF default pengguna. Namun, jika terjadi kesalahan, program akan menampilkan pesan error melalui JOptionPane agar pengguna tahu apa yang salah.

Fungsi getDuration digunakan untuk menghitung durasi parkir dalam jam. Jika waktu keluar lebih awal dari waktu masuk (misalnya karena kesalahan input), durasi dianggap 1 jam untuk menghindari nilai negatif.

## 2.6.5 Views

### AdminMenuView



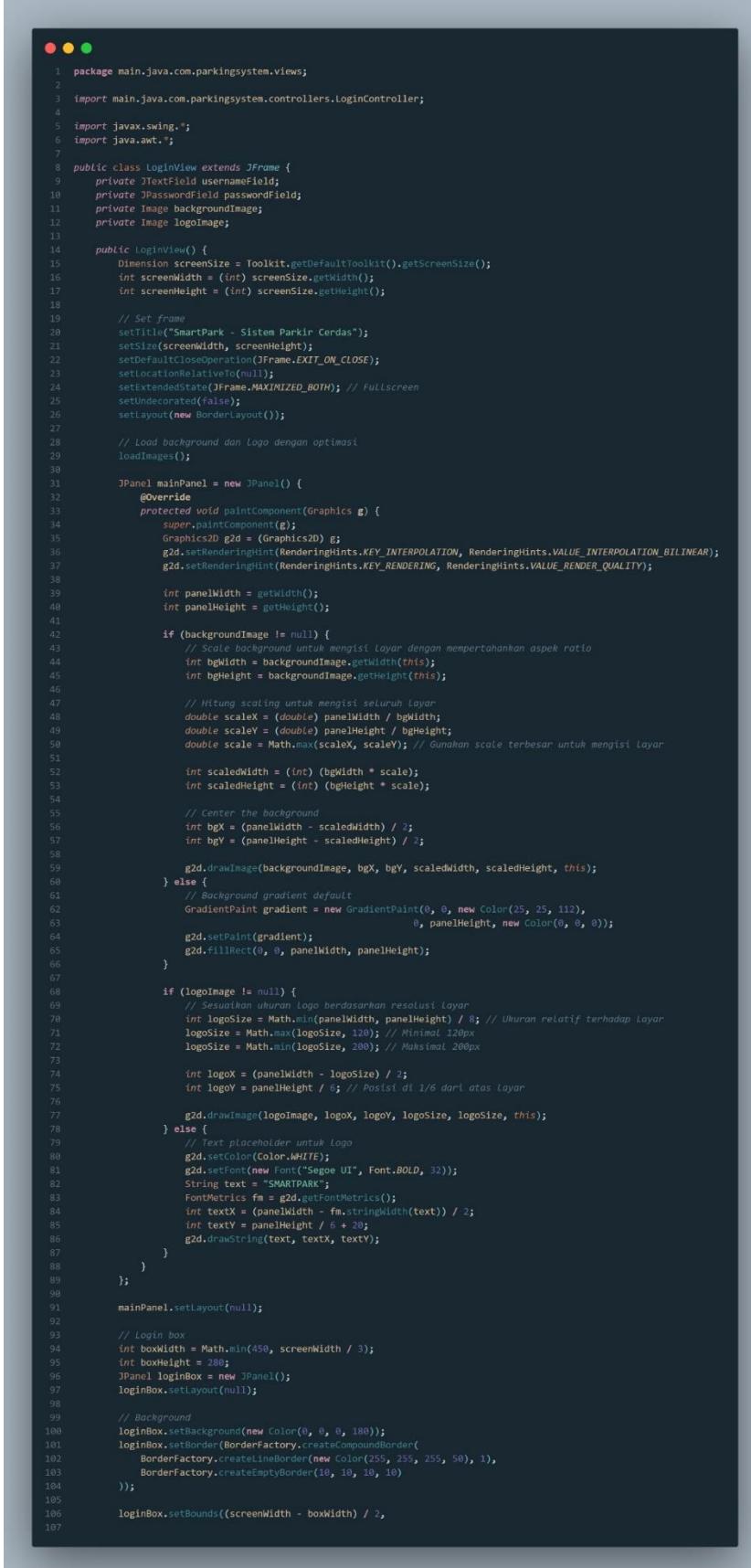
```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.models.User;
4 import javax.swing.*;
5 import java.awt.*;
6
7 public class AdminMenuView extends JFrame {
8     private Image backgroundImage;
9     private Image logoImage;
10
11    public AdminMenuView(User user) {
12        setTitle("SmartPark - Admin Dashboard");
13        setExtendedState(JFrame.MAXIMIZED_BOTH);
14        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15        setLayout(new BorderLayout());
16
17        loadImages();
18
19        JPanel contentPanel = new JPanel() {
20            @Override
21            protected void paintComponent(Graphics g) {
22                super.paintComponent(g);
23                Graphics2D g2d = (Graphics2D) g;
24                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
25                g2d.setRenderingHint(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
26
27                int panelWidth = getWidth();
28                int panelHeight = getHeight();
29
30                if (backgroundImage != null) {
31                    int bgWidth = backgroundImage.getWidth(this);
32                    int bgHeight = backgroundImage.getHeight(this);
33
34                    double scaleX = (double) panelWidth / bgWidth;
35                    double scaleY = (double) panelHeight / bgHeight;
36                    double scale = Math.max(scaleX, scaleY);
37
38                    int scaledWidth = (int) (bgWidth * scale);
39                    int scaledHeight = (int) (bgHeight * scale);
40
41                    int bgX = (panelWidth - scaledWidth) / 2;
42                    int bgY = (panelHeight - scaledHeight) / 2;
43
44                    g2d.drawImage(backgroundImage, bgX, bgY, scaledWidth, scaledHeight, this);
45                } else {
46                    GradientPaint gradient = new GradientPaint(0, 0, new Color(25, 25, 112),
47                        0, panelHeight, new Color(0, 0, 0));
48                    g2d.setPaint(gradient);
49                    g2d.fillRect(0, 0, panelWidth, panelHeight);
50                }
51            }
52        };
53        contentPanel.setLayout(null);
54
55        JLabel logoLabel = new JLabel();
56        if (logoImage != null) {
57            ImageIcon logoIcon = new ImageIcon(logoImage.getScaledInstance(100, 100, Image.SCALE_SMOOTH));
58            logoLabel.setIcon(logoIcon);
59        } else {
```

```
1 logoLabel.setText("SP");
2         logoLabel.setFont(new Font("Segoe UI", Font.BOLD, 36));
3         logoLabel.setForeground(Color.WHITE);
4         logoLabel.setHorizontalAlignment(SwingConstants.CENTER);
5         logoLabel.setVerticalAlignment(SwingConstants.CENTER);
6         logoLabel.setOpaque(true);
7         logoLabel.setBackground(new Color(30, 136, 229));
8     }
9     logoLabel.setBounds(20, 20, 100, 100);
10    contentPanel.add(logoLabel);
11
12    JLabel lblWelcome = new JLabel("Halo, " + user.getUsername() + " (Admin)");
13    lblWelcome.setFont(new Font("Segoe UI", Font.BOLD, 24));
14    lblWelcome.setForeground(Color.WHITE);
15    lblWelcome.setBounds(140, 40, 600, 30);
16    contentPanel.add(lblWelcome);
17
18    JButton logoutBtn = new JButton("Logout");
19    logoutBtn.setFont(new Font("Segoe UI", Font.PLAIN, 14));
20    logoutBtn.setBackground(new Color(220, 53, 69));
21    logoutBtn.setForeground(Color.WHITE);
22    logoutBtn.setFocusPainted(false);
23    logoutBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
24    logoutBtn.setBorder(BorderFactory.createEmptyBorder(8, 15, 8, 15));
25    logoutBtn.setBounds(140, 80, 80, 30);
26    logoutBtn.addActionListener(e -> {
27        dispose();
28        new LoginView();
29    });
30    logoutBtn.addMouseListener(new java.awt.event.MouseAdapter() {
31        public void mouseEntered(java.awt.event.MouseEvent evt) {
32            logoutBtn.setBackground(new Color(200, 35, 51));
33        }
34        public void mouseExited(java.awt.event.MouseEvent evt) {
35            logoutBtn.setBackground(new Color(220, 53, 69));
36        }
37    });
38    contentPanel.add(logoutBtn);
39
40    JPanel buttonPanel = new JPanel();
41    buttonPanel.setOpaque(false);
42    buttonPanel.setLayout(new GridLayout(2, 2, 20, 20));
43
44    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
45    int buttonPanelWidth = 500;
46    int buttonPanelHeight = 250;
47    int buttonPanelX = (screenSize.width - buttonPanelWidth) / 2;
48    int buttonPanelY = (screenSize.height - buttonPanelHeight) / 2;
49
50    buttonPanel.setBounds(buttonPanelX, buttonPanelY, buttonPanelWidth, buttonPanelHeight);
51
52    buttonPanel.add(createMenuButton("Entry Kendaraan", () -> new ParkingEntryView()));
53    buttonPanel.add(createMenuButton("Exit Kendaraan", () -> new ParkingExitView()));
54    buttonPanel.add(createMenuButton("Lihat Laporan", () -> new ReportView(true)));
55    buttonPanel.add(createMenuButton("Kelola User", () -> new UserView()));
56
57    contentPanel.add(buttonPanel);
58    add(contentPanel);
59
60    setVisible(true);
61 }
```

```
 1  private JButton createMenuButton(String text, Runnable action) {
 2      JButton btn = new JButton(text);
 3      btn.setFont(new Font("Segoe UI", Font.BOLD, 16));
 4      btn.setBackground(new Color(30, 136, 229));
 5      btn.setForeground(Color.WHITE);
 6      btn.setFocusPainted(false);
 7      btn.setCursor(new Cursor(Cursor.HAND_CURSOR));
 8      btn.setBorder(BorderFactory.createEmptyBorder(15, 20, 15, 20));
 9      btn.setPreferredSize(new Dimension(200, 100));
10
11      btn.addActionListener(e -> {
12          try {
13              action.run();
14          } catch (Exception ex) {
15              JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
16              ex.printStackTrace();
17          }
18      });
19
20      btn.addMouseListener(new java.awt.event.MouseAdapter() {
21          public void mouseEntered(java.awt.event.MouseEvent evt) {
22              btn.setBackground(new Color(25, 118, 210));
23          }
24          public void mouseExited(java.awt.event.MouseEvent evt) {
25              btn.setBackground(new Color(30, 136, 229));
26          }
27      });
28      return btn;
29  }
30
31  private void loadImages() {
32      backgroundImage = loadImageFromPaths(new String[]{
33          "src/main/resources/bg.jpg", "resources/bg.jpg", "bg.jpg"
34      });
35      logoImage = loadImageFromPaths(new String[]{
36          "src/main/resources/logo.png", "resources/logo.png", "logo.png"
37      });
38  }
39
40  private Image loadImageFromPaths(String[] paths) {
41      for (String path : paths) {
42          ImageIcon icon = new ImageIcon(path);
43          if (icon.getIconWidth() > 0 && icon.getIconHeight() > 0) {
44              return icon.getImage();
45          }
46      }
47      return null;
48  }
49 }
50 }
```

Kelas AdminMenuView berfungsi sebagai dasbor utama untuk pengguna dengan peran administrator. Tampilan ini didesain agar selalu memenuhi layar penuh (maksimal), dilengkapi dengan gambar latar belakang dan logo aplikasi untuk estetika yang menarik. Di sisi kiri atas layar, terdapat logo aplikasi dan teks sapaan yang menampilkan username administrator yang sedang login bersama dengan peran pengguna. Terdapat juga tombol "Logout" yang berfungsi untuk mengakhiri sesi pengguna dan mengarahkan kembali ke halaman login. Bagian utama dari AdminMenuView adalah sebuah panel berisi empat tombol menu utama yang tersusun rapi dalam format grid. Tombol-tombol ini adalah: "Entry Kendaraan", "Exit Kendaraan", "Lihat Laporan", dan "Kelola User". Setiap tombol ini telah dikonfigurasi dengan event listener yang akan membuka tampilan (View) yang sesuai ketika diklik, seperti ParkingEntryView, ParkingExitView, ReportView, atau UserView. Desain ini memberikan akses cepat dan terorganisir bagi administrator untuk menjalankan berbagai fungsi pengelolaan sistem parkir.

## LoginView



```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.controllers.LoginController;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 public class LoginView extends JFrame {
9     private JTextField usernameField;
10    private JPasswordField passwordField;
11    private Image backgroundImage;
12    private Image logoImage;
13
14    public LoginView() {
15        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
16        int screenWidth = (int) screenSize.getWidth();
17        int screenHeight = (int) screenSize.getHeight();
18
19        // Set frame
20        setTitle("SmartPark - Sistem Parkir Cerdas");
21        setSize(screenWidth, screenHeight);
22        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23        setLocationRelativeTo(null);
24        setExtendedState(JFrame.MAXIMIZED_BOTH); // Fullscreen
25        setUndecorated(false);
26       .setLayout(new BorderLayout());
27
28        // Load background dan Logo dengan optimasi
29        loadImages();
30
31        JPanel mainPanel = new JPanel() {
32            @Override
33            protected void paintComponent(Graphics g) {
34                super.paintComponent(g);
35                Graphics2D g2d = (Graphics2D) g;
36                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
37                g2d.setRenderingHint(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
38
39                int panelWidth = getWidth();
40                int panelHeight = getHeight();
41
42                if (backgroundImage != null) {
43                    // Scale background untuk mengisi seluruh layar dengan mempertahankan aspek ratio
44                    int bgWidth = backgroundImage.getWidth(this);
45                    int bgHeight = backgroundImage.getHeight(this);
46
47                    // Hitung scaling untuk mengisi seluruh layar
48                    double scaleX = (double) panelWidth / bgWidth;
49                    double scaleY = (double) panelHeight / bgHeight;
50                    double scale = Math.max(scaleX, scaleY); // Gunakan scale terbesar untuk mengisi layar
51
52                    int scaledWidth = (int) (bgWidth * scale);
53                    int scaledHeight = (int) (bgHeight * scale);
54
55                    // Center the background
56                    int bgX = (panelWidth - scaledWidth) / 2;
57                    int bgY = (panelHeight - scaledHeight) / 2;
58
59                    g2d.drawImage(backgroundImage, bgX, bgY, scaledWidth, scaledHeight, this);
60                } else {
61                    // Background gradient default
62                    GradientPaint gradient = new GradientPaint(0, 0, new Color(25, 25, 112),
63                                                    0, panelHeight, new Color(0, 0, 0));
64
65                    g2d.setPaint(gradient);
66                    g2d.fillRect(0, 0, panelWidth, panelHeight);
67                }
68
69                if (logoImage != null) {
70                    // Sesuaikan ukuran logo berdasarkan resolusi layar
71                    int logoSize = Math.min(panelWidth, panelHeight) / 8; // Ukuran relatif terhadap layar
72                    logoSize = Math.max(logoSize, 120); // Minimal 120px
73                    logoSize = Math.min(logoSize, 200); // Maksimal 200px
74
75                    int logoX = (panelWidth - logoSize) / 2;
76                    int logoY = panelHeight / 6; // Posisi di 1/6 dari atas layar
77
78                    g2d.drawImage(logoImage, logoX, logoY, logoSize, logoSize, this);
79                } else {
80                    // Text placeholder untuk logo
81                    g2d.setColor(Color.WHITE);
82                    g2d.setFont(new Font("Segoe UI", Font.BOLD, 32));
83                    String text = "SMARTPARK";
84                    FontMetrics fm = g2d.getFontMetrics();
85                    int textX = (panelWidth - fm.stringWidth(text)) / 2;
86                    int textY = panelHeight / 6 + 20;
87                    g2d.drawString(text, textX, textY);
88                }
89            }
90        };
91        mainPanel.setLayout(null);
92
93        // Login box
94        int boxWidth = Math.min(450, screenWidth / 3);
95        int boxHeight = 280;
96        JPanel loginBox = new JPanel();
97        loginBox.setLayout(null);
98
99        // Background
100        loginBox.setBackground(new Color(0, 0, 0, 180));
101        loginBox.setBorder(BorderFactory.createCompoundBorder(
102            BorderFactory.createLineBorder(new Color(255, 255, 255, 50), 1),
103            BorderFactory.createEmptyBorder(10, 10, 10, 10)
104        ));
105
106        loginBox.setBounds((screenWidth - boxWidth) / 2,
```

```

1 // Title
2 jLabel1.setTitle = new JLabel("Login ke SmartPark", SwingConstants.CENTER);
3 jLabel1.setFont(new Font("Segoe UI", Font.BOLD, 20));
4 jLabel1.setForeground(Color.WHITE);
5 jLabel1.setOpaque(true);
6 jLabel1.setBounds(0, 20, boxWidth, 30);
7 loginBox.add(jLabel1);
8
9 // Username
10 JLabel userLabel = new JLabel("Username:");
11 userLabel.setForeground(Color.WHITE);
12 userLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
13 userLabel.setBounds(50, 70, 80, 25);
14 loginBox.add(userLabel);
15
16 JTextField usernameField = new JTextField();
17 usernameField.setBounds(50, 90, boxWidth - 100, 30);
18 usernameField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
19 usernameField.setBorder(BorderFactory.createCompoundBorder(
20     BorderFactory.createTitledBorder(new Color(100, 100, 100), 1),
21     BorderFactory.createEmptyBorder(1, 10, 5, 10)));
22 loginBox.add(usernameField);
23
24 // Password
25 JLabel passLabel = new JLabel("Password:");
26 passLabel.setForeground(Color.WHITE);
27 passLabel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
28 passLabel.setBounds(50, 120, 80, 25);
29 loginBox.add(passLabel);
30
31 JPasswordField passwordField = new JPasswordField();
32 passwordField.setBounds(50, 140, boxWidth - 100, 30);
33 passwordField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
34 passwordField.setBorder(BorderFactory.createCompoundBorder(
35     BorderFactory.createTitledBorder(new Color(100, 100, 100), 1),
36     BorderFactory.createEmptyBorder(1, 10, 5, 10)));
37 loginBox.add(passwordField);
38
39 // Login button
40 JButton loginBtn = new JButton("LOGIN");
41 loginBtn.setBounds(0, 210, boxWidth - 100, 35);
42 loginBtn.setBackground(new Color(10, 130, 220));
43 loginBtn.setForeground(Color.WHITE);
44 loginBtn.setRolloverEnabled(false);
45 loginBtn.setFont(new Font("Segoe UI", Font.BOLD, 10));
46 loginBtn.setBorder(BorderFactory.createEmptyBorder());
47 loginBtn.setCursor(new Cursor(Cursor.IANO_CURSOR));
48 loginBox.add(loginBtn);
49
50 // Hover effect
51 loginBtn.addMouseListener(new java.awt.event.MouseAdapter() {
52     @Override
53     public void mouseEntered(java.awt.event.MouseEvent evt) {
54         loginBtn.setBackground(new Color(10, 110, 210));
55     }
56
57     @Override
58     public void mouseExited(java.awt.event.MouseEvent evt) {
59         loginBtn.setBackground(new Color(10, 130, 220));
60     }
61 });
62
63 loginBtn.addActionListener(e -> {
64     String user = usernameField.getText().trim();
65     String pass = new String(passwordField.getPassword());
66
67     if (user.isEmpty() || pass.isEmpty()) {
68         JOptionPane.showMessageDialog(this,
69             "Username dan Password tidak boleh kosong!",
70             "Error",
71             JOptionPane.ERROR_MESSAGE);
72         return;
73     }
74
75     LoginController.login(user, pass);
76 });
77
78 // Enter key untuk login
79 passwordField.addActionListener(e -> loginBtn.onClick());
80
81 loginBox.add(loginBtn);
82 mainPanel.add(loginBox);
83 add(mainPanel);
84
85 // Fokus pada username field saat startup
86 SwingUtilities.invokeLater(() -> usernameField.requestFocus());
87
88 setVisible(true);
89 }
90
91 private void loadImages() {
92     try {
93         // Path alternatif untuk mencari gambar
94         String[] bgPaths = {
95             "src/main/resources/bg.jpg",
96             "src/resources/bg.jpg",
97             "resources/bg.jpg",
98             "bg.jpg"
99         };
100
101         String[] logoPaths = {
102             "src/main/resources/logo.png",
103             "src/resources/logo.png",
104             "resources/logo.png",
105             "logo.png"
106         };
107
108         // Load background
109         backgroundImage = loadImageFromPaths(bgPaths, "Background");
110
111         // Load logo
112         logoImage = loadImageFromPaths(logoPaths, "Logo");
113
114     } catch (Exception e) {
115         System.out.println("Error loading images: " + e.getMessage());
116     }
117 }
118
119 private Image loadImageFromPaths(String[] paths, String imageType) {
120     for (String path : paths) {
121         try {
122             ImageIcon icon = new ImageIcon(path);
123             if (icon.getIconWidth() > 0 & icon.getIconHeight() > 0) {
124                 System.out.println(imageType + " loaded successfully from: " + path);
125                 System.out.println(imageType + " size: " + icon.getIconWidth() + "x" + icon.getIconHeight());
126                 return icon.getImage();
127             }
128         } catch (Exception e) {
129             // Lanjut ke path berikutnya
130         }
131     }
132     System.out.println(imageType + " not found in any of the specified paths");
133     return null;
134 }
135 }

```

LoginView adalah antarmuka utama yang memungkinkan pengguna untuk masuk ke dalam sistem SmartPark. Tampilan ini dirancang untuk fullscreen agar sesuai dengan berbagai ukuran layar, dengan latar belakang yang dapat diisi gambar atau gradient warna jika gambar tidak tersedia. Di bagian tengah layar, terdapat sebuah kotak login semi-transparan yang beradaptasi dengan ukuran layar, memberikan fokus visual yang jelas. Kotak login ini berisi field input untuk username dan password, serta sebuah tombol "LOGIN". Sebelum proses login dilakukan, sistem akan memvalidasi bahwa kedua field input tidak kosong. Setelah tombol login ditekan (atau pengguna menekan Enter pada field password), LoginView akan memanggil LoginController untuk memproses otentikasi. Jika login berhasil, tampilan akan beralih ke menu utama yang sesuai dengan peran pengguna (admin atau operator); jika gagal, pesan kesalahan akan ditampilkan. Penekanan visual pada logo aplikasi juga terintegrasi, dengan ukuran yang disesuaikan secara dinamis agar terlihat optimal di berbagai resolusi layar laptop.

## OperatorMenuView

```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.models.User;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 public class OperatorMenuView extends JFrame {
9     private Image backgroundImage;
10    private Image logoImage;
11
12    public OperatorMenuView(User user) {
13        setTitle("SmartPark - Operator Dashboard");
14        setExtendedState(JFrame.MAXIMIZED_BOTH);
15        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        setLayout(new BorderLayout());
17
18        loadImages();
19
20        JPanel contentPanel = new JPanel() {
21            @Override
22            protected void paintComponent(Graphics g) {
23                super.paintComponent(g);
24                Graphics2D g2d = (Graphics2D) g;
25                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
26                g2d.setRenderingHint(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
27
28                int panelWidth = getWidth();
29                int panelHeight = getHeight();
30
31                if (backgroundImage != null) {
32                    int bgWidth = backgroundImage.getWidth(this);
33                    int bgHeight = backgroundImage.getHeight(this);
34
35                    double scaleX = (double) panelWidth / bgWidth;
36                    double scaleY = (double) panelHeight / bgHeight;
37                    double scale = Math.max(scaleX, scaleY);
38
39                    int scaledWidth = (int) (bgWidth * scale);
40                    int scaledHeight = (int) (bgHeight * scale);
41
42                    int bgX = (panelWidth - scaledWidth) / 2;
43                    int bgY = (panelHeight - scaledHeight) / 2;
44
45                    g2d.drawImage(backgroundImage, bgX, bgY, scaledWidth, scaledHeight, this);
46                } else {
47                    GradientPaint gradient = new GradientPaint(0, 0, new Color(25, 25, 112),
48                        0, panelHeight, new Color(0, 0, 0));
49                    g2d.setPaint(gradient);
50                    g2d.fillRect(0, 0, panelWidth, panelHeight);
51                }
52            }
53        };
54        contentPanel.setLayout(null);
55
56        JLabel logoLabel = new JLabel();
57        if (logoImage != null) {
58            ImageIcon logoIcon = new ImageIcon(logoImage.getScaledInstance(100, 100, Image.SCALE_SMOOTH));
59            logoLabel.setIcon(logoIcon);
60        } else {
61            logoLabel.setText("SP");
62            logoLabel.setFont(new Font("Segoe UI", Font.BOLD, 36));
63            logoLabel.setForeground(Color.WHITE);
64            logoLabel.setHorizontalAlignment(SwingConstants.CENTER);
65            logoLabel.setVerticalAlignment(SwingConstants.CENTER);
66            logoLabel.setOpaque(true);
67            logoLabel.setBackground(new Color(30, 136, 229));
68        }
69        logoLabel.setBounds(20, 20, 100, 100);
70        contentPanel.add(logoLabel);
71
72        JLabel lblWelcome = new JLabel("Halo, " + user.getUsername() + " (Operator)");
73        lblWelcome.setFont(new Font("Segoe UI", Font.BOLD, 24));
74        lblWelcome.setForeground(Color.WHITE);
75        lblWelcome.setBounds(140, 40, 600, 30);
76        contentPanel.add(lblWelcome);
```

```
1 JButton logoutBtn = new JButton("Logout");
2 logoutBtn.setFont(new Font("Segoe UI", Font.PLAIN, 14));
3 logoutBtn.setBackground(new Color(220, 53, 69));
4 logoutBtn.setForeground(Color.WHITE);
5 logoutBtn.setFocusPainted(false);
6 logoutBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
7 logoutBtn.setBorder(BorderFactory.createEmptyBorder(8, 15, 8, 15));
8 logoutBtn.setBounds(148, 80, 80, 30);
9 logoutBtn.addActionListener(e -> {
10     dispose();
11     new LoginView();
12 });
13 logoutBtn.addMouseListener(new java.awt.event.MouseAdapter() {
14     public void mouseEntered(java.awt.event.MouseEvent evt) {
15         logoutBtn.setBackground(new Color(200, 35, 51));
16     }
17     public void mouseExited(java.awt.event.MouseEvent evt) {
18         logoutBtn.setBackground(new Color(220, 53, 69));
19     }
20 });
21 contentPanel.add(logoutBtn);
22
23 JPanel buttonPanel = new JPanel();
24 buttonPanel.setOpaque(false);
25 buttonPanel.setLayout(new GridLayout(1, 3, 40, 20));
26
27 Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
28 int buttonPanelWidth = 880;
29 int buttonPanelHeight = 200;
30 int buttonPanelX = (screenSize.width - buttonPanelWidth) / 2;
31 int buttonPanelY = (screenSize.height - buttonPanelHeight) / 2;
32
33 buttonPanel.setBounds(buttonPanelX, buttonPanelY, buttonPanelWidth, buttonPanelHeight);
34
35 buttonPanel.add(createMenuButton("Entry Kendaraan", () -> new ParkingEntryView()));
36 buttonPanel.add(createMenuButton("Exit Kendaraan", () -> new ParkingExitView()));
37 buttonPanel.add(createMenuButton("Lihat Laporan", () -> new ReportView(false)));
38
39 contentPanel.add(buttonPanel);
40 add(contentPanel);
41
42 setVisible(true);
43 }
44
45 private JButton createMenuButton(String text, Runnable action) {
46     JButton btn = new JButton(text);
47     btn.setFont(new Font("Segoe UI", Font.BOLD, 15));
48     btn.setBackground(new Color(30, 136, 229));
49     btn.setForeground(Color.WHITE);
50     btn.setFocusPainted(false);
51     btn.setCursor(new Cursor(Cursor.HAND_CURSOR));
52     btn.setBorder(BorderFactory.createEmptyBorder(10, 25, 10, 25));
53     btn.setPreferredSize(new Dimension(250, 60));
54
55     btn.addActionListener(e -> {
56         try {
57             action.run();
58         } catch (Exception ex) {
59             JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
60             ex.printStackTrace();
61         }
62     });
63
64     btn.addMouseListener(new java.awt.event.MouseAdapter() {
65         public void mouseEntered(java.awt.event.MouseEvent evt) {
66             btn.setBackground(new Color(25, 115, 210));
67         }
68         public void mouseExited(java.awt.event.MouseEvent evt) {
69             btn.setBackground(new Color(30, 136, 229));
70         }
71     });
72     return btn;
73 }
74
75 private void loadImages() {
76     backgroundImage = loadImageFromPaths(new String[]{
77         "src/main/resources/bg.jpg", "resources/bg.jpg", "bg.jpg"
78     });
79     logoImage = loadImageFromPaths(new String[]{
80         "src/main/resources/logo.png", "resources/logo.png", "logo.png"
81     });
82 }
83
84 private Image loadImageFromPaths(String[] paths) {
85     for (String path : paths) {
86         ImageIcon icon = new ImageIcon(path);
87         if (icon.getIconWidth() > 0 && icon.getIconHeight() > 0) {
88             return icon.getImage();
89         }
90     }
91     return null;
92 }
93 }
```

Serupa dengan AdminMenuView, kelas OperatorMenuView berfungsi sebagai dasbor utama namun khusus untuk pengguna dengan peran operator. Tampilan ini juga memaksimalkan penggunaan layar penuh dan menampilkan gambar latar belakang serta logo aplikasi. Di pojok kiri atas, terdapat logo aplikasi dan sapaan yang menampilkan username operator yang sedang login beserta perannya. Tombol "Logout" juga tersedia untuk mengakhiri sesi dan kembali ke halaman login. Perbedaan utama terletak pada pilihan menu yang tersedia; OperatorMenuView hanya menyediakan tiga tombol menu utama, yaitu "Entry Kendaraan", "Exit Kendaraan", dan "Lihat Laporan". Tombol-tombol ini tersusun dalam tata letak grid yang lebih sederhana, memfokuskan operator pada tugas-tugas inti terkait operasional parkir sehari-hari tanpa akses ke fungsi pengelolaan pengguna. Setiap tombol juga memiliki event listener yang akan membuka tampilan yang relevan, seperti ParkingEntryView, ParkingExitView, atau ReportView, memastikan operator memiliki akses langsung ke fungsionalitas yang mereka butuhkan.

## ParkingEntryView

```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.controllers.ParkingEntryController;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 public class ParkingEntryView extends JFrame {
9     private Image backgroundImage;
10
11     public ParkingEntryView() {
12         setTitle("SmartPark - Entry Kendaraan");
13         setExtendedState(JFrame.MAXIMIZED_BOTH);
14         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
15         setLayout(null);
16
17         loadBackground();
18
19         JPanel mainPanel = new JPanel() {
20             @Override
21             protected void paintComponent(Graphics g) {
22                 super.paintComponent(g);
23                 Graphics2D g2d = (Graphics2D) g;
24                 g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
25
26                 int panelWidth = getWidth();
27                 int panelHeight = getHeight();
28
29                 if (backgroundImage != null) {
30                     int imgWidth = backgroundImage.getWidth(this);
31                     int imgHeight = backgroundImage.getHeight(this);
32
33                     double scalex = (double) panelWidth / imgWidth;
34                     double scaley = (double) panelHeight / imgHeight;
35                     double scale = Math.max(scalex, scaley);
36
37                     int scaledWidth = (int) (imgWidth * scale);
38                     int scaledHeight = (int) (imgHeight * scale);
39
40                     int x = (panelWidth - scaledWidth) / 2;
41                     int y = (panelHeight - scaledHeight) / 2;
42
43                     g2d.drawImage(backgroundImage, x, y, scaledWidth, scaledHeight, this);
44                 } else {
45                     g2d.setColor(Color.DARK_GRAY);
46                     g2d.fillRect(0, 0, panelWidth, panelHeight);
47                 }
48             }
49
50             mainPanel.setLayout(null);
51
52             int boxWidth = 400;
53             int boxHeight = 220;
54             int x = (Toolkit.getDefaultToolkit().getScreenSize().width - boxWidth) / 2;
55             int y = (Toolkit.getDefaultToolkit().getScreenSize().height - boxHeight) / 2;
56
57             JPanel formPanel = new JPanel();
58             formPanel.setLayout(null);
59             formPanel.setBounds(x, y, boxWidth, boxHeight);
60             formPanel.setBackground(new Color(0, 0, 0, 180));
61
62             JLabel title = new JLabel("Entry Kendaraan", SwingConstants.CENTER);
63             title.setFont(new Font("Segoe UI", Font.BOLD, 20));
64             title.setForeground(Color.WHITE);
65             title.setBounds(0, 60, boxWidth, 30);
66             formPanel.add(title);
67
68             JLabel lblPlate = new JLabel("Plat Nomor:");
69             lblPlate.setBounds(50, 60, 100, 25);
70             lblPlate.setForeground(Color.WHITE);
71             formPanel.add(lblPlate);
72
73             JTextField plateField = new JTextField();
74             plateField.setBounds(150, 60, 180, 25);
75             formPanel.add(plateField);
76
77             JLabel lblType = new JLabel("Jenis Kendaraan:");
78             lblType.setBounds(30, 100, 120, 25);
79             lblType.setForeground(Color.WHITE);
80             formPanel.add(lblType);
81
82             JComboBox<String> typeBox = new JComboBox<String>(new String[]{"Motor", "Mobil"});
83             typeBox.setBounds(150, 100, 180, 25);
84             formPanel.add(typeBox);
85
86             JButton btnSubmit = new JButton("Simpan");
87             btnSubmit.setBounds((boxWidth - 180) / 2, 150, 100, 30);
88             btnSubmit.setBackground(new Color(0, 136, 225));
89             btnSubmit.setForeground(Color.WHITE);
90             btnSubmit.setFocusPainted(false);
91             btnSubmit.addActionListener(e -> {
92                 String plate = plateField.getText().trim();
93                 String type = (String) typeBox.getSelectedItem();
94                 if (!plate.isEmpty()) {
95                     ParkingEntryController.handleEntry(plate, type);
96                     dispose();
97                 } else {
98                     JOptionPane.showMessageDialog(this, "Plat nomor tidak boleh kosong!");
99                 }
100            });
101            formPanel.add(btnSubmit);
102
103            mainPanel.add(formPanel);
104            setContentPane(mainPanel);
105            setVisible(true);
106            plateField.addActionListener(e -> btnSubmit.doClick());
107        }
108
109        private void loadBackground() {
110            String[] paths = {
111                "src/main/resources/bg.jpg", "resources/bg.jpg", "bg.jpg"
112            };
113            for (String path : paths) {
114                ImageIcon icon = new ImageIcon(path);
115                if (icon.getIconWidth() > 0 && icon.getIconHeight() > 0) {
116                    backgroundImage = icon.getImage();
117                    break;
118                }
119            }
120        }
121    }
122}
```

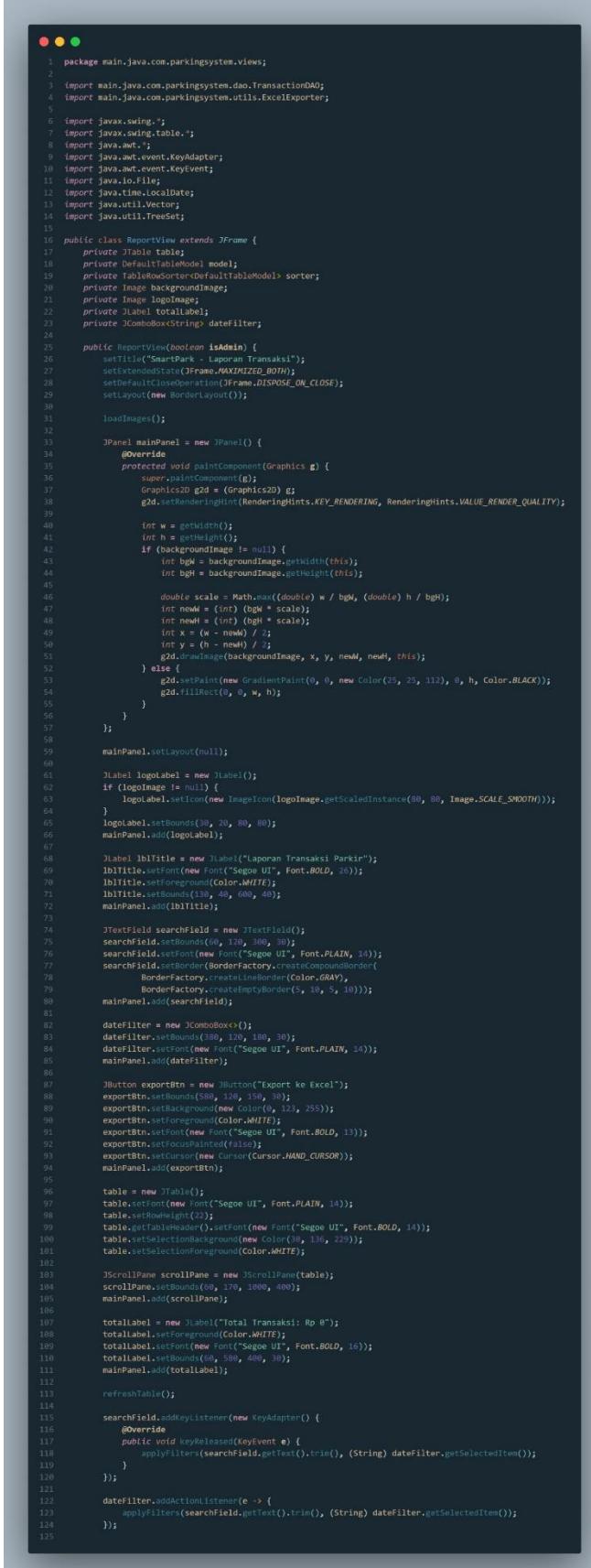
Kelas ParkingEntryView menyediakan antarmuka grafis yang intuitif untuk mencatat kendaraan yang masuk ke area parkir. Tampilan ini juga mengadopsi mode layar penuh dan dapat memuat gambar latar belakang. Di bagian tengah layar, terdapat sebuah panel formulir semi-transparan yang menjadi fokus utama. Formulir ini berisi field input untuk plat nomor kendaraan dan sebuah dropdown (combo box) untuk memilih jenis kendaraan (misalnya, "Motor" atau "Mobil"). Setelah pengguna memasukkan data yang diperlukan, tombol "Simpan" dapat diklik (atau pengguna menekan Enter pada field plat nomor). Sebelum data disimpan, sistem akan memvalidasi bahwa plat nomor tidak kosong. Jika validasi berhasil, ParkingEntryView akan memanggil ParkingEntryController untuk memproses penyimpanan transaksi masuk kendaraan. Setelah transaksi berhasil dicatat, tampilan ini akan ditutup secara otomatis, mengarahkan pengguna kembali ke menu sebelumnya. Desain ini memastikan proses pencatatan kendaraan masuk berjalan efisien dan minim kesalahan.

## ParkingExitView

```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.controllers.ParkingExitController;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 public class ParkingExitView extends JFrame {
9     private Image backgroundImage;
10    private Image logoImage;
11
12    public ParkingExitView() {
13        setTitle("Kendaraan Keluar - SmartPark");
14        setExtendedState(JFrame.MAXIMIZED_BOTH);
15        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
16        setLayout(new BorderLayout());
17
18        loadImages();
19
20        JPanel contentPanel = new JPanel() {
21            @Override
22            protected void paintComponent(Graphics g) {
23                super.paintComponent(g);
24                Graphics2D gd = (Graphics2D) g;
25                gd.setRenderingHint(RenderingHints.KEY_INTERPOLATION, RenderingHints.VALUE_INTERPOLATION_BILINEAR);
26                gd.setRenderingHint(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
27
28                int w = getWidth();
29                int h = getHeight();
30
31                if (backgroundImage != null) {
32                    int imgWidth = backgroundImage.getWidth(this);
33                    int imgH = backgroundImage.getHeight(this);
34                    double scale = Math.min((double) w / imgW, (double) h / imgH);
35                    int newW = (int) (imgW * scale);
36                    int newH = (int) (imgH * scale);
37                    int x = (w - newW) / 2;
38                    int y = (h - newH) / 2;
39
40                    gd2d.drawImage(backgroundImage, x, y, newW, newH, this);
41                } else {
42                    gd2d.setPaint(new GradientPaint(0, 0, new Color(25, 25, 112), 0, h, Color.BLACK));
43                    gd2d.fillRect(0, 0, w, h);
44                }
45            };
46        contentPanel.setLayout(null);
47
48        JLabel logoLabel = new JLabel();
49        if (logoImage != null) {
50            logoLabel.setIcon(new ImageIcon(logoImage.getScaledInstance(100, 100, Image.SCALE_SMOOTH)));
51        } else {
52            logoLabel.setText("SP");
53            logoLabel.setFont(new Font("Segoe UI", Font.BOLD, 36));
54            logoLabel.setOpaque(true);
55            logoLabel.setForeground(Color.WHITE);
56            logoLabel.setBackground(new Color(30, 136, 220));
57            logoLabel.setHorizontalTextPosition(SwingConstants.CENTER);
58        }
59        logoLabel.setBounds(30, 20, 100, 100);
60        contentPanel.add(logoLabel);
61
62        JLabel title = new JLabel("Proses Kendaraan Keluar", SwingConstants.CENTER);
63        title.setFont(new Font("Segoe UI", Font.BOLD, 24));
64        title.setForeground(Color.WHITE);
65        title.setBounds(100, 40, 600, 40);
66        contentPanel.add(title);
67
68        JPanel formPanel = new JPanel();
69        formPanel.setLayout(null);
70        formPanel.setLayout(null);
71        int formWidth = 400;
72        int formHeight = 160;
73        formPanel.setBounds(Toolkit.getDefaultToolkit().getScreenSize().width - formWidth) / 2,
74        (Toolkit.getDefaultToolkit().getScreenSize().height - formHeight) / 2,
75        formWidth, formHeight);
76
77        JLabel lbPlate = new JLabel("Plat Nomor:");
78        lbPlate.setFont(new Font("Segoe UI", Font.PLAIN, 18));
79        lbPlate.setForeground(Color.WHITE);
80        lbPlate.setBounds(20, 10, 120, 30);
81        formPanel.add(lbPlate);
82
83        JTextField tfPlate = new JTextField();
84        tfPlate.setFont(new Font("Segoe UI", Font.PLAIN, 16));
85        tfPlate.setBounds(150, 10, 200, 30);
86        formPanel.add(tfPlate);
87
88        JButton btnProses = new JButton("Proses Keluar");
89        btnProses.setFont(new Font("Segoe UI", Font.BOLD, 16));
90        btnProses.setBackground(new Color(30, 136, 220));
91        btnProses.setForeground(Color.WHITE);
92        btnProses.setFocusPainted(false);
93        btnProses.setCursor(new Cursor(Cursor.HAND_CURSOR));
94        btnProses.setBounds(150, 100, 200, 40);
95
96        btnProses.addActionListener(e -> {
97            String plate = tfPlate.getText().trim();
98            if (!plate.isEmpty()) {
99                ParkingExitController.handleExit(plate, btnProses);
100            } else {
101                JOptionPane.showMessageDialog(this, "Plat nomor tidak boleh kosong!", "Peringatan", JOptionPane.WARNING_MESSAGE);
102            }
103        });
104
105        tfPlate.addActionListener(e -> btnProses.doClick());
106
107        formPanel.add(btnProses);
108        contentPanel.add(formPanel);
109        add(contentPanel);
110        setVisible(true);
111    }
112
113    private void loadImages() {
114        backgroundImage = loadImageFromPaths(new String[]{
115            "src/main/resources/bg.jpg", "resources/bg.jpg", "bg.jpg"
116        });
117        logoImage = loadImageFromPaths(new String[]{
118            "src/main/resources/logo.png", "resources/logo.png", "logo.png"
119        });
120    }
121
122    private Image loadImageFromPaths(String[] paths) {
123        for (String path : paths) {
124            ImageIcon icon = new ImageIcon(path);
125            if (icon.getIconWidth() > 0 && icon.getIconHeight() > 0) {
126                return icon.getImage();
127            }
128        }
129        return null;
130    }
131 }
132 }
```

Kelas ParkingExitView berfungsi sebagai antarmuka grafis bagi pengguna untuk memproses keluarnya kendaraan dari area parkir. Antarmuka ini dirancang untuk memaksimalkan tampilan layar guna memberikan pengalaman visual yang optimal. Saat diinisialisasi, ParkingExitView akan memuat gambar latar belakang dan logo aplikasi untuk mempercantik tampilan, meskipun ia juga memiliki mekanisme fallback jika gambar tidak ditemukan. Komponen utama pada tampilan ini meliputi sebuah field teks untuk memasukkan plat nomor kendaraan dan sebuah tombol "Proses Keluar". Ketika pengguna menekan tombol tersebut atau menekan Enter setelah memasukkan plat nomor, sistem akan memanggil ParkingExitController untuk menangani logika keluarnya kendaraan, seperti menghitung durasi parkir, biaya, dan memperbarui status parkir. Desain ini memastikan bahwa proses keluar kendaraan menjadi cepat, efisien, dan user-friendly dengan validasi input sederhana untuk mencegah plat nomor kosong.

## ReportView



```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.dao.TransactionDAO;
4 import main.java.com.parkingsystem.utils.ExcelExporter;
5
6 import javax.swing.*;
7 import javax.swing.table.*;
8 import java.awt.*;
9 import java.awt.event.KeyAdapter;
10 import java.awt.event.KeyEvent;
11 import java.io.File;
12 import java.util.Date;
13 import java.util.Vector;
14 import java.util.TreeSet;
15
16 public class ReportView extends JFrame {
17     private JTable table;
18     private DefaultTableModel model;
19     private TableRowSorter<DefaultTableModel> sorter;
20     private Image backgroundImage;
21     private Image logoImage;
22     private JLabel totalLabel;
23     private JComboBox<String> dateFilter;
24
25     public ReportView(boolean isAdmin) {
26         setTitle("Sarana Parkir Laporan Transaksi");
27         setExtendedState(JFrame.MAXIMIZED_BOTH);
28         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
29         setLayout(new BorderLayout());
30
31         loadImages();
32
33         JPanel mainPanel = new JPanel() {
34             @Override
35             protected void paintComponent(Graphics g) {
36                 super.paintComponent(g);
37                 Graphics2D g2d = (Graphics2D) g;
38                 g2d.setRenderingHint(RenderingHints.KEY_RENDERING, RenderingHints.VALUE_RENDER_QUALITY);
39
40                 int w = getWidth();
41                 int h = getHeight();
42                 if (backgroundImage != null) {
43                     int bgW = backgroundImage.getWidth(this);
44                     int bgH = backgroundImage.getHeight(this);
45
46                     double scale = Math.max((double) w / bgW, (double) h / bgH);
47                     int newW = (int) (bgW * scale);
48                     int newH = (int) (bgH * scale);
49                     int x = (w - newW) / 2;
50                     int y = (h - newH) / 2;
51                     g2d.drawImage(backgroundImage, x, y, newW, newH, this);
52                 } else {
53                     g2d.setPaint(new GradientPaint(0, 0, new Color(25, 25, 112), 0, h, Color.BLACK));
54                     g2d.fillRect(0, 0, w, h);
55                 }
56             }
57         };
58
59         mainPanel.setLayout(null);
60
61         JLabel logoLabel = new JLabel();
62         if (logoImage != null) {
63             logoLabel.setIcon(new ImageIcon(logoImage.getScaledInstance(80, 80, Image.SCALE_SMOOTH)));
64         }
65         logoLabel.setBounds(30, 20, 80, 80);
66         mainPanel.add(logoLabel);
67
68         JLabel lblTitle = new JLabel("Laporan Transaksi Parkir");
69         lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 20));
70         lblTitle.setForeground(Color.WHITE);
71         lblTitle.setBounds(130, 40, 600, 40);
72         mainPanel.add(lblTitle);
73
74         JTextField searchField = new JTextField();
75         searchField.setBounds(90, 120, 300, 30);
76         searchField.setFont(new Font("Segoe UI", Font.PLAIN, 14));
77         searchField.setBorder(BorderFactory.createCompoundBorder(
78             BorderFactory.createLineBorder(Color.GRAY),
79             BorderFactory.createEmptyBorder(5, 10, 5, 10)));
80         mainPanel.add(searchField);
81
82         JButton exportBtn = new JButton("Export ke Excel");
83         exportBtn.setBounds(90, 120, 150, 30);
84         exportBtn.setBackground(new Color(0, 123, 255));
85         exportBtn.setForeground(Color.WHITE);
86         exportBtn.setFont(new Font("Segoe UI", Font.BOLD, 13));
87         exportBtn.setFocusable(false);
88         exportBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
89         mainPanel.add(exportBtn);
90
91         table = new JTable();
92         table.setFont(new Font("Segoe UI", Font.PLAIN, 14));
93         table.setRowHeight(22);
94         table.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 14));
95         table.setSelectionBackground(new Color(10, 136, 220));
96         table.setSelectionForeground(Color.WHITE);
97
98         JScrollPane scrollPane = new JScrollPane(table);
99         scrollPane.setBounds(90, 170, 1000, 400);
100        mainPanel.add(scrollPane);
101
102        totalLabel = new JLabel("Total Transaksi: Rp 0");
103        totalLabel.setForeground(Color.WHITE);
104        totalLabel.setFont(new Font("Segoe UI", Font.BOLD, 16));
105        totalLabel.setBounds(90, 580, 400, 30);
106        mainPanel.add(totalLabel);
107
108        refreshTable();
109
110        searchField.addKeyListener(new KeyAdapter() {
111            @Override
112            public void keyReleased(KeyEvent e) {
113                applyFilters(searchField.getText().trim(), (String) dateFilter.getSelectedItem());
114            }
115        });
116
117        dateFilter.addActionListener(e -> {
118            applyFilters(searchField.getText().trim(), (String) dateFilter.getSelectedItem());
119        });
120
121        dateFilter.addActionListener(e -> {
122            applyFilters(searchField.getText().trim(), (String) dateFilter.getSelectedItem());
123        });
124    }
125}
```

```

1  searchField.addKeyListener(new KeyAdapter() {
2      @Override
3      public void keyReleased(KeyEvent e) {
4          applyFilters(searchField.getText().trim(), (String) dateFilter.getSelectedItem());
5      }
6  });
7
8  dateFilter.addActionListener(e -> {
9      applyFilters(searchField.getText().trim(), (String) dateFilter.getSelectedItem());
10 });
11
12 exportBtn.addActionListener(e -> {
13     File dir = new File("Laporan");
14     if (!dir.exists()) dir.mkdirs();
15     File outfile = new File(dir, "Laporan_Parkir_" + LocalDate.now() + ".xlsx");
16     ExcelExporter.exportTableToExcel(model, outfile.getAbsolutePath());
17
18     JOptionPane.showMessageDialog(this,
19         "Laporan berhasil diexport ke file " + outfile.getAbsolutePath(),
20         "Export Berhasil", JOptionPane.INFORMATION_MESSAGE);
21 });
22
23 if (isAdmin) {
24     JButton btnDelete = new JButton("Hapus");
25     btnDelete.setRollover(true);
26     btnDelete.setBackground(new Color(230, 50, 60));
27     btnDelete.setForeground(Color.WHITE);
28     btnDelete.setFont(new Font("Segoe UI", Font.BOLD, 14));
29     btnDelete.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
30     btnDelete.addActionListener(e -> {
31         int row = table.getSelectedRow();
32         if (row >= 0) {
33             String plate = Table.getValueAt(row, 1).toString();
34             int confirm = JOptionPane.showConfirmDialog(this,
35                 "Yakin ingin menghapus data plat " + plate + "?",
36                 "Konfirmasi", JOptionPane.YES_NO_OPTION);
37             if (confirm == JOptionPane.YES_OPTION) {
38                 TransactionDAO.deletePlate(plate);
39                 refreshTable();
40             }
41         } else {
42             JOptionPane.showMessageDialog(this, "Pilih baris yang ingin dihapus.");
43         }
44     });
45     mainPanel.add(btnDelete);
46 }
47
48 add(mainPanel);
49 setvisible(true);
50 }
51 }
52 private void refreshTable() {
53     model = TransactionDAO.getAllTransactions();
54     table.setModel(model);
55     sorter = new TableRowSorter<Object>(model);
56     table.setRowSorter(sorter);
57     updateTotal();
58     populateTable();
59 }
60
61 private void applyFilters(String keyword, String date) {
62     RowFilter<DefaultTableModel, Object> combined = new RowFilter<Object>() {
63         @Override
64         public boolean include(RowFilter<Object> entry) extends DefaultableModel, ? extends Object> entry {
65             boolean matchesKeyword = keyword.isEmpty();
66             boolean matchesDate = (date == null || date.equals("Semua"));
67
68             for (int i = 0; i < entry.getValueCount(); i++) {
69                 if ((matchesKeyword && entry.getString(i).toLowerCase().contains(keyword.toLowerCase())))
70                     matchesKeyword = true;
71
72                 if ((matchesDate && entry.getString(i).startsWith(date))) {
73                     matchesDate = true;
74                 }
75             }
76             return matchesKeyword && matchesDate;
77         }
78     };
79     sorter.setRowFilter(combined);
80     updateTotal();
81 }
82
83 private void updateTotal() {
84     int total = 0;
85     for (int i = 0; i < sorter.getVisibleRowCount(); i++) {
86         int modelRow = sorter.convertRowIndexToModel(i);
87         Object valueObj = model.getValueAt(modelRow, 0);
88         if (valueObj != null) {
89             String value = valueObj.toString().replace("Rp ", "").replace(".", ",");
90             try {
91                 total += Integer.parseInt(value);
92             } catch (NumberFormatException ignored) {
93             }
94         }
95     }
96     totalLabel.setText("Total Transaksi: Rp " + String.format("%d", total).replace(',', '.'));
97 }
98
99 private void populateTable() {
100     dataFilter.removeAllItems();
101     dateFilter.removeAllItems();
102     dateFilter.addItem("Semua");
103     Vector<Object> data = model.getVector();
104     TreeSet<String> uniqueDates = new TreeSet<Object>();
105     for (Object rowObj : data) {
106         Vector<Object> row = (Vector<Object>) rowObj;
107         if (row.get(0) != null) {
108             String date = row.get(0).toString().split(" ")[0];
109             uniqueDates.add(date);
110         }
111     }
112     for (String d : uniqueDates) dateFilter.addItem(d);
113 }
114
115 private void loadImages() {
116     backgroundImage = loadImageFromPaths(new String[]{(
117         "src/main/resources/bg.jpg", "resources/bg.jpg", "bg.jpg"
118     )});
119     logoImage = loadImageFromPaths(new String[]{(
120         "src/main/Resources/logo.png", "Resources/logo.png", "Logo.png"
121     )});
122 }
123
124 private Image loadImageFromPaths(String[] paths) {
125     for (String path : paths) {
126         ImageIcon icon = new ImageIcon(path);
127         if (icon.getIconWidth() > 0 && icon.getIconHeight() > 0) {
128             return icon.getImage();
129         }
130     }
131     return null;
132 }
133 }
134

```

ReportView menyediakan fungsionalitas untuk menampilkan laporan transaksi parkir secara komprehensif. Tampilan ini juga menggunakan mode layar penuh dan dapat memuat gambar latar belakang serta logo. Fitur utamanya adalah sebuah tabel yang menampilkan detail setiap transaksi parkir, seperti plat nomor, jenis kendaraan, waktu masuk, waktu keluar, durasi, dan total biaya. Untuk memudahkan analisis data, ReportView dilengkapi dengan field pencarian yang memungkinkan pengguna mencari transaksi berdasarkan kata kunci tertentu, serta dropdown filter tanggal untuk menampilkan transaksi pada tanggal spesifik atau semua tanggal. Setiap perubahan pada filter ini akan secara dinamis memperbarui data di tabel dan secara otomatis menghitung total pendapatan dari transaksi yang ditampilkan. Selain itu, pengguna dapat mengekspor laporan ke format Excel untuk keperluan pencatatan atau pelaporan lebih lanjut. Bagi pengguna dengan hak akses admin, tersedia juga tombol "Hapus" yang memungkinkan penghapusan data transaksi tertentu setelah konfirmasi, memberikan kontrol penuh atas integritas data laporan.

## UserView

```
1 package main.java.com.parkingsystem.views;
2
3 import main.java.com.parkingsystem.dao.UserDAO;
4 import main.java.com.parkingsystem.utils.ExcelExporter;
5
6 import javax.swing.*;
7 import javax.swing.table.DefaultTableModel;
8 import java.awt.*;
9 import java.io.File;
10 import java.time.LocalDate;
11
12 public class UserView extends JFrame {
13     private JTable table;
14     private JTextField tfUsername, tfPassword;
15     private JComboBox<String> cbRole;
16
17     public UserView() {
18         setTitle("SmartPark - Kelola User");
19         setExtendedState(JFrame.MAXIMIZED_BOTH);
20         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21         setLayout(null);
22
23         JLabel titleLabel = new JLabel("Manajemen User", SwingConstants.LEFT);
24         titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 26));
25         titleLabel.setBounds(40, 30, 400, 40);
26         add(titleLabel);
27
28         JButton exportBtn = new JButton("Export ke Excel");
29         exportBtn.setBounds(900, 40, 160, 30);
30         exportBtn.setFont(new Font("Segoe UI", Font.BOLD, 13));
31         exportBtn.setBackground(new Color(0, 123, 255));
32         exportBtn.setForeground(Color.WHITE);
33         exportBtn.setFocusPainted(false);
34         exportBtn.setCursor(new Cursor(Cursor.HAND_CURSOR));
35         exportBtn.addActionListener(e -> {
36             try {
37                 File dir = new File("laporan");
38                 i
```

```
1 JButton btnUpdate = createButton("Update", 400, 410, new Color(30, 130, 229));
2 btnUpdate.addActionListener(e -> {
3     int row = table.getSelectedRow();
4     if (row >= 0) {
5         String username = tfUsername.getText().trim();
6         String password = tfPassword.getText().trim();
7         if (username.isEmpty() || password.isEmpty()) {
8             JOptionPane.showMessageDialog(this, "Username dan password tidak boleh kosong.");
9             return;
10        }
11        int id = Integer.parseInt(table.getValueAt(row, 0).toString());
12        if (UserDAO.updateUser(id, username, password, cbRole.getSelectedItem().toString())) {
13            JOptionPane.showMessageDialog(this, "User berhasil diupdate.");
14            refreshTable();
15            resetForm();
16        }
17    }
18 });
19 add(btnUpdate);
20
21 JButton btnReset = createButton("Reset", 550, 370, new Color(30, 130, 229));
22 btnReset.addActionListener(e -> resetForm());
23 add(btnReset);
24
25 JButton btnDelete = createButton("Hapus", 550, 410, new Color(220, 53, 69));
26 btnDelete.addActionListener(e -> {
27     int row = table.getSelectedRow();
28     if (row >= 0) {
29         int id = Integer.parseInt(table.getValueAt(row, 0).toString());
30         int confirm = JOptionPane.showConfirmDialog(this, "Yakin ingin menghapus user ini?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
31         if (confirm == JOptionPane.YES_OPTION) {
32             if (UserDAO.deleteUserById(id)) {
33                 JOptionPane.showMessageDialog(this, "User berhasil dihapus.");
34                 refreshTable();
35                 resetForm();
36             }
37         }
38     }
39 });
40 add(btnDelete);
41
42 // Isi form ketika baris diklik
43 table.addMouseListener(new java.awt.event.MouseAdapter() {
44     public void mouseClicked(java.awt.event.MouseEvent evt) {
45         int row = table.getSelectedRow();
46         tfUsername.setText(table.getValueAt(row, 1).toString());
47         tfPassword.setText(table.getValueAt(row, 2).toString());
48         cbRole.setSelectedItem(table.getValueAt(row, 3).toString());
49     }
50 });
51
52 setVisible(true);
53 }
54
55 private void refreshTable() {
56     table.setModel(UserDAO.getAllUsers());
57 }
58
59 private void resetForm() {
60     tfUsername.setText("");
61     tfPassword.setText("");
62     cbRole.setSelectedIndex(0);
63 }
64
65 private JButton createButton(String text, int x, int y, Color bgColor) {
66     JButton btn = new JButton(text);
67     btn.setBounds(x, y, 120, 30);
68     btn.setFont(new Font("Segoe UI", Font.BOLD, 14));
69     btn.setBackground(bgColor);
70     btn.setForeground(Color.WHITE);
71     btn.setCursor(new Cursor(Cursor.HAND_CURSOR));
72     btn.setFocusPainted(false);
73     return btn;
74 }
75 }
```

Kelas UserView dirancang khusus untuk administrator guna mengelola akun pengguna dalam sistem. Tampilan ini menyajikan sebuah tabel yang merinci daftar pengguna, termasuk ID, username, password, dan role (admin atau operator). Administrator dapat mengekspor data pengguna ke format Excel untuk keperluan audit atau dokumentasi. Di bagian bawah tabel, terdapat formulir sederhana dengan field untuk username, password, dan dropdown untuk memilih role. Formulir ini mendukung operasi Tambah, Update, dan

Hapus pengguna. Ketika sebuah baris di tabel diklik, informasi pengguna yang dipilih akan secara otomatis terisi di formulir, memudahkan proses update. Validasi dasar diterapkan untuk memastikan username dan password tidak kosong saat menambahkan atau memperbarui pengguna. Fungsionalitas hapus juga dilengkapi dengan konfirmasi untuk mencegah penghapusan yang tidak disengaja. UserView menyediakan tool yang esensial bagi administrator untuk menjaga keamanan dan struktur akses dalam sistem parkir.

### 2.6.6 Main.java



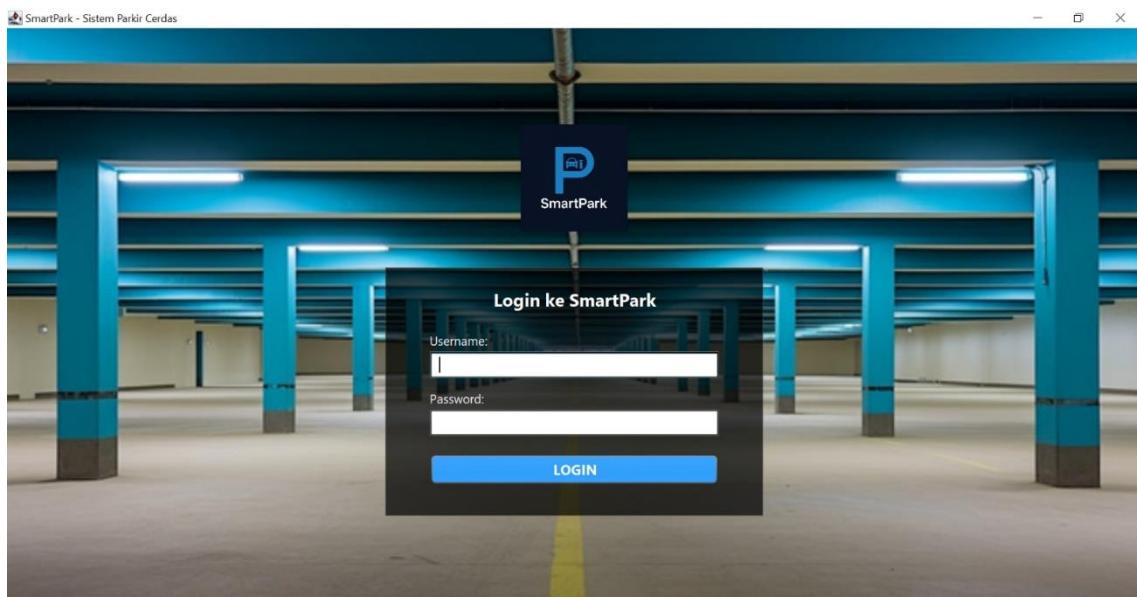
```
1 package main.java.com.parkingsystem;
2
3 import main.java.com.parkingsystem.views.LoginView;
4
5 import javax.swing.*;
6
7 public class Main {
8     public static void main(String[] args) {
9         // Set Look and Feel agar tampilannya modern (Opsional)
10        try {
11            UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
12        } catch (Exception e) {
13            System.out.println("Gagal atur tema: " + e.getMessage());
14        }
15        // Jalankan tampilan Login
16        SwingUtilities.invokeLater(() -> new LoginView());
17    }
18 }
19
```

Kode program ini digunakan untuk memulai aplikasi sistem parkir berbasis grafis, dipakai ketika pengguna pertama kali menjalankan program. Di dalamnya terdapat kelas utama bernama Main yang menjadi titik awal eksekusi aplikasi. Saat program dijalankan, kelas ini akan mencoba mengatur tampilan antarmuka agar terlihat lebih modern menggunakan tema "Nimbus". Jika gagal, aplikasi tetap bisa berjalan dengan tampilan default.

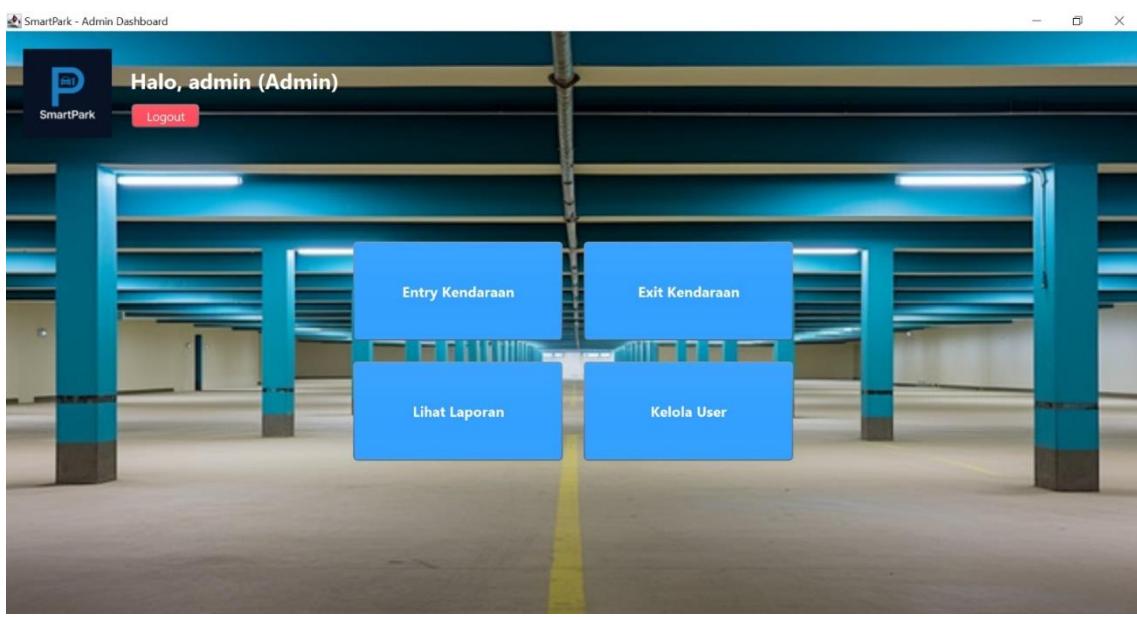
Selain itu, kode ini juga langsung menjalankan tampilan login dengan membuat objek dari kelas LoginView. Tampilan login ini nantinya akan menjadi halaman pertama yang dilihat oleh pengguna sebelum masuk ke bagian utama sistem parkir. Penggunaan SwingUtilities.invokeLater() memastikan bahwa antarmuka grafis dimuat secara aman dan tidak menyebabkan gangguan pada jalannya program.

## Pengujian

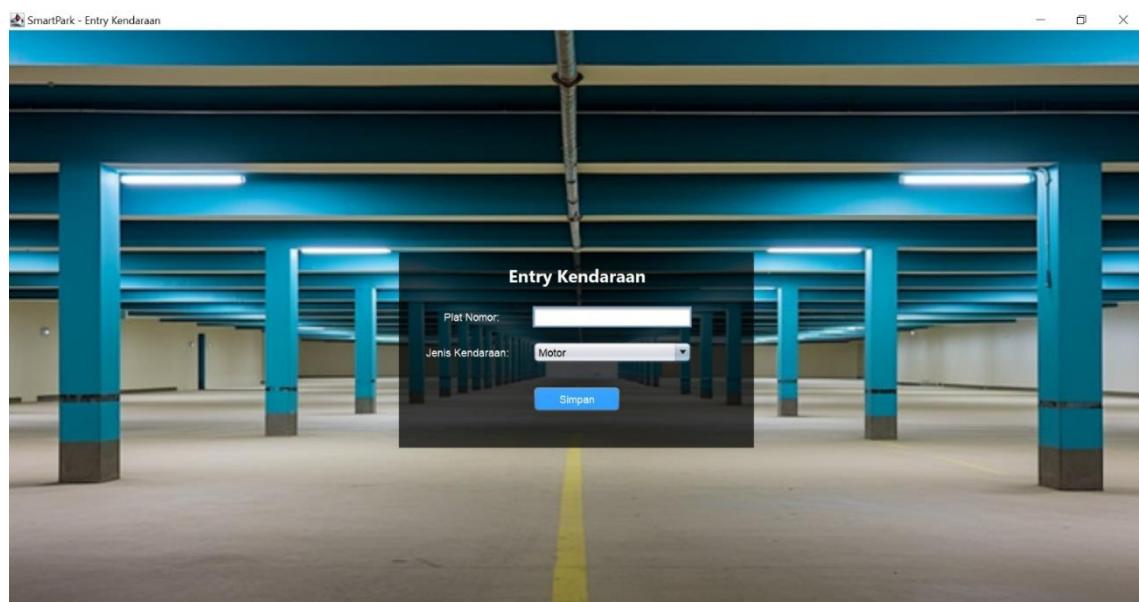
### Login



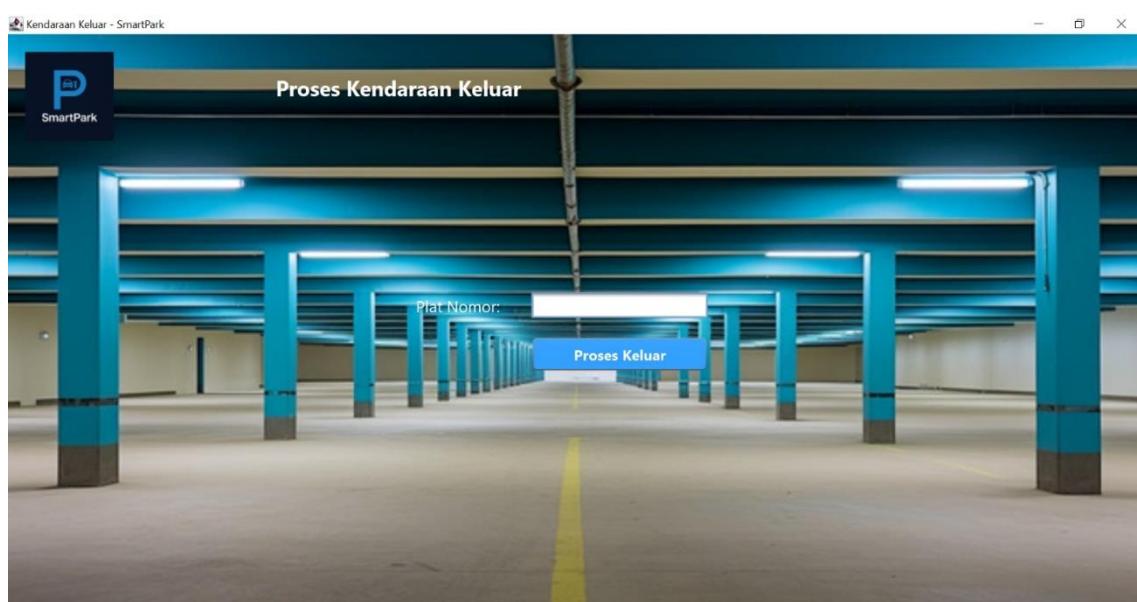
### Dashboard Admin



## Entry Kendaraan Admin



## Exit Kendaraan Admin



## Laporan Admin

The screenshot shows a window titled "Laporan Transaksi Parkir" (Parking Transaction Report). At the top, there is a search bar, a dropdown menu set to "Semua", and a blue "Export ke Excel" button. Below the header is a table with columns: No, Plat Nomor, Jenis, Masuk, Keluar, and Biaya. The table lists 16 transactions. At the bottom left, it says "Total Transaksi: Rp 85.000". On the right side, there is a red "Hapus" (Delete) button.

| No | Plat Nomor     | Jenis | Masuk               | Keluar              | Biaya |
|----|----------------|-------|---------------------|---------------------|-------|
| 1  | 1763542763UI78 | Mobil | 2025-06-07 22:19:01 | 2025-06-07 22:19:06 | 3000  |
| 2  | 123456TR78     | Motor | 2025-06-07 20:39:40 | 2025-06-07 20:39:47 | 2000  |
| 3  | 123ERT567      | Motor | 2025-06-07 20:20:40 | 2025-06-07 20:32:25 | 2000  |
| 4  | 214YU70        | Motor | 2025-06-07 19:07:28 | 2025-06-07 19:11:34 | 2000  |
| 5  | 234RT678       | Mobil | 2025-06-07 18:56:34 | 2025-06-07 18:56:41 | 3000  |
| 6  | WRS56RD        | Motor | 2025-06-07 11:39:07 | 2025-06-07 11:39:19 | 2000  |
| 7  | 123TR456       | Mobil | 2025-06-07 11:24:38 | 2025-06-07 11:24:44 | 3000  |
| 8  | WRG23RT        | Motor | 2025-06-07 11:23:17 | 2025-06-07 11:23:27 | 2000  |
| 9  | 1254TU56       | Motor | 2025-06-07 11:08:52 | 2025-06-07 11:08:57 | 2000  |
| 10 | 1254SVU67      | Mobil | 2025-06-07 11:08:01 | 2025-06-07 11:08:08 | 3000  |
| 11 | 23764T7        | Motor | 2025-06-06 20:18:03 | 2025-06-06 20:18:14 | 2000  |
| 12 | 237537T685     | Motor | 2025-05-31 22:31:01 | 2025-06-06 22:06:04 | 25000 |
| 13 | 1253T8Y8       | Motor | 2025-05-28 19:46:50 | 2025-05-28 19:47:00 | 3000  |
| 14 | 327654T78      | Motor | 2025-05-27 23:04:49 | 2025-05-27 23:06:01 | 3000  |
| 15 | 12476Y73       | Motor | 2025-05-27 22:48:57 | 2025-05-27 22:49:50 | 3000  |
| 16 | 36578U83       | Mobil | 2025-05-27 22:17:45 | 2025-06-07 01:43:15 | 25000 |

## Kelola User Admin

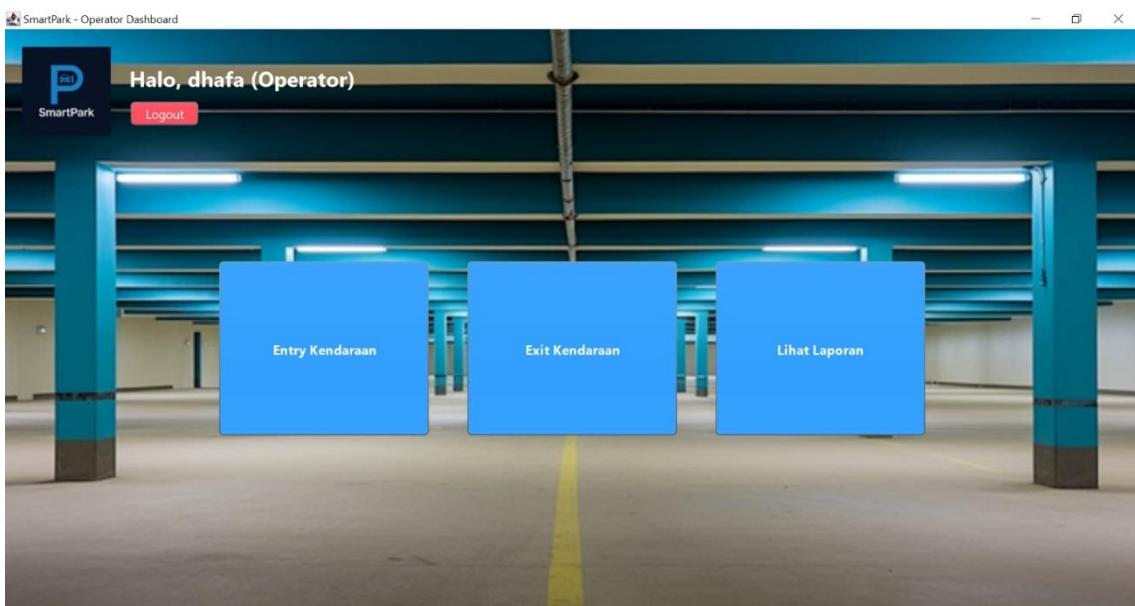
The screenshot shows a window titled "Manajemen User" (User Management). At the top right is a blue "Export ke Excel" button. Below the header is a table with columns: ID, Username, Password, and Role. The table lists three users. At the bottom, there are input fields for "Username", "Password", and "Role", along with "Tambah" (Add), "Reset", "Update", and "Hapus" (Delete) buttons.

| ID | Username | Password  | Role     |
|----|----------|-----------|----------|
| 1  | admin    | admin123  | admin    |
| 3  | dhafa    | dhafa123  | operator |
| 5  | farhan   | farhan123 | admin    |

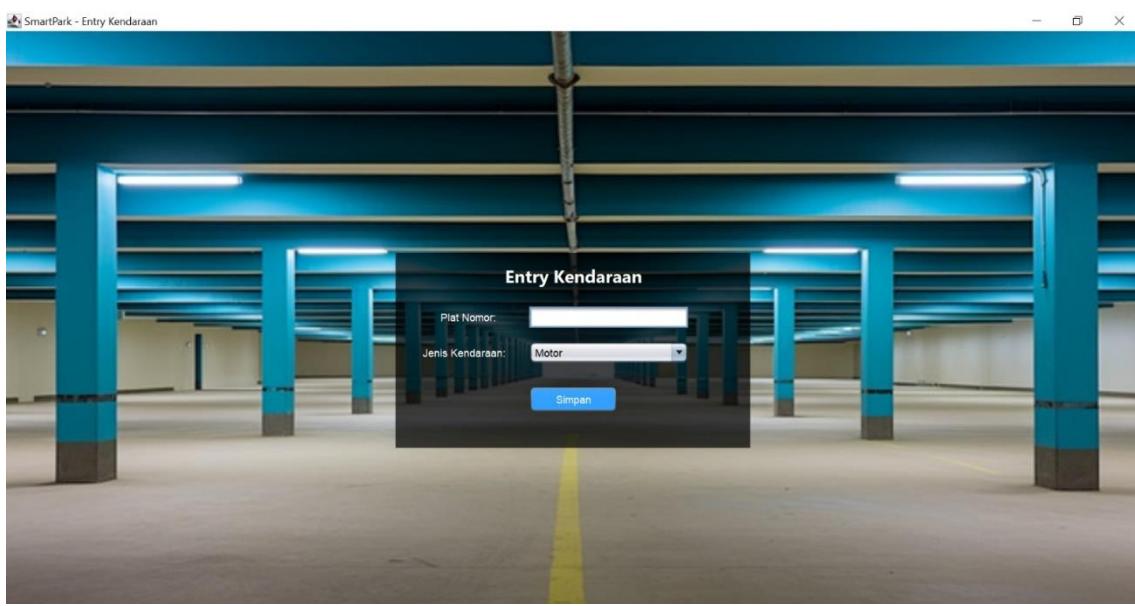
Username:   
Password:   
Role:

**Tambah** **Reset**  
**Update** **Hapus**

## Dashboard Operator



## Entry Kendaraan Operator



## Exit Kendaraan Operator



## Laporan Operator

A screenshot of the SmartPark software interface titled "Laporan Transaksi Parkir". It shows a view of a parking garage with blue support pillars. A table is displayed in the foreground, showing a list of transactions. At the bottom of the table, the text "Total Transaksi: Rp 85.000" is visible. The table has columns: No, Plat Nomor, Jenis, Masuk, Keluar, and Biaya. The data is as follows:

| No | Plat Nomor     | Jenis | Masuk               | Keluar              | Biaya |
|----|----------------|-------|---------------------|---------------------|-------|
| 1  | 1763542763UI78 | Mobil | 2025-06-07 22:19:01 | 2025-06-07 22:19:06 | 3000  |
| 2  | 123456TR78     | Motor | 2025-06-07 20:39:40 | 2025-06-07 20:39:47 | 2000  |
| 3  | 123ERT567      | Motor | 2025-06-07 20:20:40 | 2025-06-07 20:32:25 | 2000  |
| 4  | 214YU70        | Motor | 2025-06-07 19:07:28 | 2025-06-07 19:11:34 | 2000  |
| 5  | 234RT678       | Mobil | 2025-06-07 18:56:34 | 2025-06-07 18:56:41 | 3000  |
| 6  | WRS56RD        | Motor | 2025-06-07 11:39:07 | 2025-06-07 11:39:19 | 2000  |
| 7  | 123TR456       | Mobil | 2025-06-07 11:24:38 | 2025-06-07 11:24:44 | 3000  |
| 8  | WRG23RT        | Motor | 2025-06-07 11:23:17 | 2025-06-07 11:23:27 | 2000  |
| 9  | 1254TU56       | Motor | 2025-06-07 11:08:52 | 2025-06-07 11:08:57 | 2000  |
| 10 | 1254SVU67      | Mobil | 2025-06-07 11:08:01 | 2025-06-07 11:08:08 | 3000  |
| 11 | 23764T7        | Motor | 2025-06-06 20:18:03 | 2025-06-06 20:18:14 | 2000  |
| 12 | 237537T685     | Motor | 2025-05-31 22:31:01 | 2025-06-06 22:06:04 | 25000 |
| 13 | 1253T8Y8       | Motor | 2025-05-28 19:46:50 | 2025-05-28 19:47:00 | 3000  |
| 14 | 327654T78      | Motor | 2025-05-27 23:04:49 | 2025-05-27 23:06:01 | 3000  |
| 15 | 12476Y73       | Motor | 2025-05-27 22:48:57 | 2025-05-27 22:49:50 | 3000  |
| 16 | 365T8U83       | Mobil | 2025-05-27 22:17:45 | 2025-06-07 01:43:15 | 25000 |

## Tampilan Struk Motor



### SMARTPARK - STRUK PARKIR

Plat Nomor : 123ERT567  
Jenis Kendaraan: Motor  
Waktu Masuk : 07-06-2025 20:20  
Waktu Keluar : 07-06-2025 20:32  
Durasi Parkir : 1 jam  
Biaya Parkir : Rp 2000  
\* Tarif: Rp 2.000 per jam  
\* Maksimal bayar: Rp 25.000

*Terima kasih telah menggunakan SmartPark!*

---

## Tampilan Struk Mobil



### SMARTPARK - STRUK PARKIR

Plat Nomor : 123TR456  
Jenis Kendaraan: Mobil  
Waktu Masuk : 07-06-2025 11:24  
Waktu Keluar : 07-06-2025 11:24  
Durasi Parkir : 1 jam  
Biaya Parkir : Rp 3000  
\* Tarif: Rp 3.000 per jam  
\* Maksimal bayar: Rp 30.000

*Terima kasih telah menggunakan SmartPark!*

## **BAB III**

### **PENUTUP**

#### **3.1 Kesimpulan**

Berdasarkan hasil perancangan, pengembangan, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi SmartPark berhasil dikembangkan dengan menerapkan prinsip-prinsip Pemrograman Berorientasi Objek (OOP) secara utuh dan tepat. Penerapan konsep OOP seperti enkapsulasi, abstraksi, pewarisan, dan polimorfisme memberikan struktur kode yang modular, mudah dipelihara, dan fleksibel untuk dikembangkan di masa mendatang. Arsitektur sistem menggunakan pendekatan Model-View-Controller (MVC) serta Data Access Object (DAO) yang memisahkan antara tampilan, logika, dan akses data, sehingga meningkatkan keterbacaan dan efisiensi dalam pengelolaan program.

Aplikasi ini juga dilengkapi fitur-fitur utama yang berjalan sesuai dengan fungsinya, seperti pencatatan kendaraan masuk dan keluar, penghitungan tarif parkir otomatis berdasarkan jenis kendaraan dan durasi, pencetakan struk parkir dalam format PDF, pengelolaan data user (admin dan operator), serta kemampuan untuk mengekspor laporan ke format Excel. Penggunaan Java Swing sebagai antarmuka grafis mendukung interaksi pengguna secara mudah dan intuitif, bahkan bagi petugas parkir yang tidak memiliki latar belakang teknis.

Hasil pengujian menunjukkan bahwa aplikasi bekerja secara stabil dan dapat menangani berbagai skenario umum operasional parkir, termasuk validasi plat nomor, pengecekan tarif, dan penanganan error seperti koneksi database yang gagal. Dengan demikian, SmartPark layak diimplementasikan pada sistem parkir berskala kecil hingga menengah seperti kampus, perkantoran, atau area komersial. Sistem ini juga masih sangat terbuka untuk dikembangkan lebih lanjut, misalnya dengan menambahkan fitur integrasi RFID, kamera pembaca plat nomor, serta pembayaran digital.

Secara keseluruhan, proyek pengembangan aplikasi SmartPark telah berhasil menjawab rumusan masalah dan memenuhi tujuan yang telah ditetapkan pada awal perancangan. Proyek ini tidak hanya menghasilkan produk aplikasi yang bermanfaat

secara praktis, tetapi juga memberikan pengalaman nyata bagi tim pengembang dalam menerapkan teknologi dan prinsip rekayasa perangkat lunak modern.