OOP project

Vehicle Rental Management System

Group Members:

1. Kalkidan Birhabu UGR/1053/17
2. Kenean Engda UGR/3226/17
3. Lidya Demerw UGR/6153/17
4. Maedot Eskender UGR/9011/17
5. Nardos Nega UGR/8725/17

URL : https://github.com/lidyademerw/Vehicle-Rental-Management-System

Submission Date: January 30, 2026.

Vehicle Rental Management System

Problem Statement

Vehicle rental businesses often rely on manual record-keeping, which leads to errors such as double-booking, inconsistent pricing, and loss of rental history. This project, the Vehicle Rental Management System, provides a robust digital solution. It automates fleet tracking and cost calculation while ensuring data security through defined user roles and persistent storage.

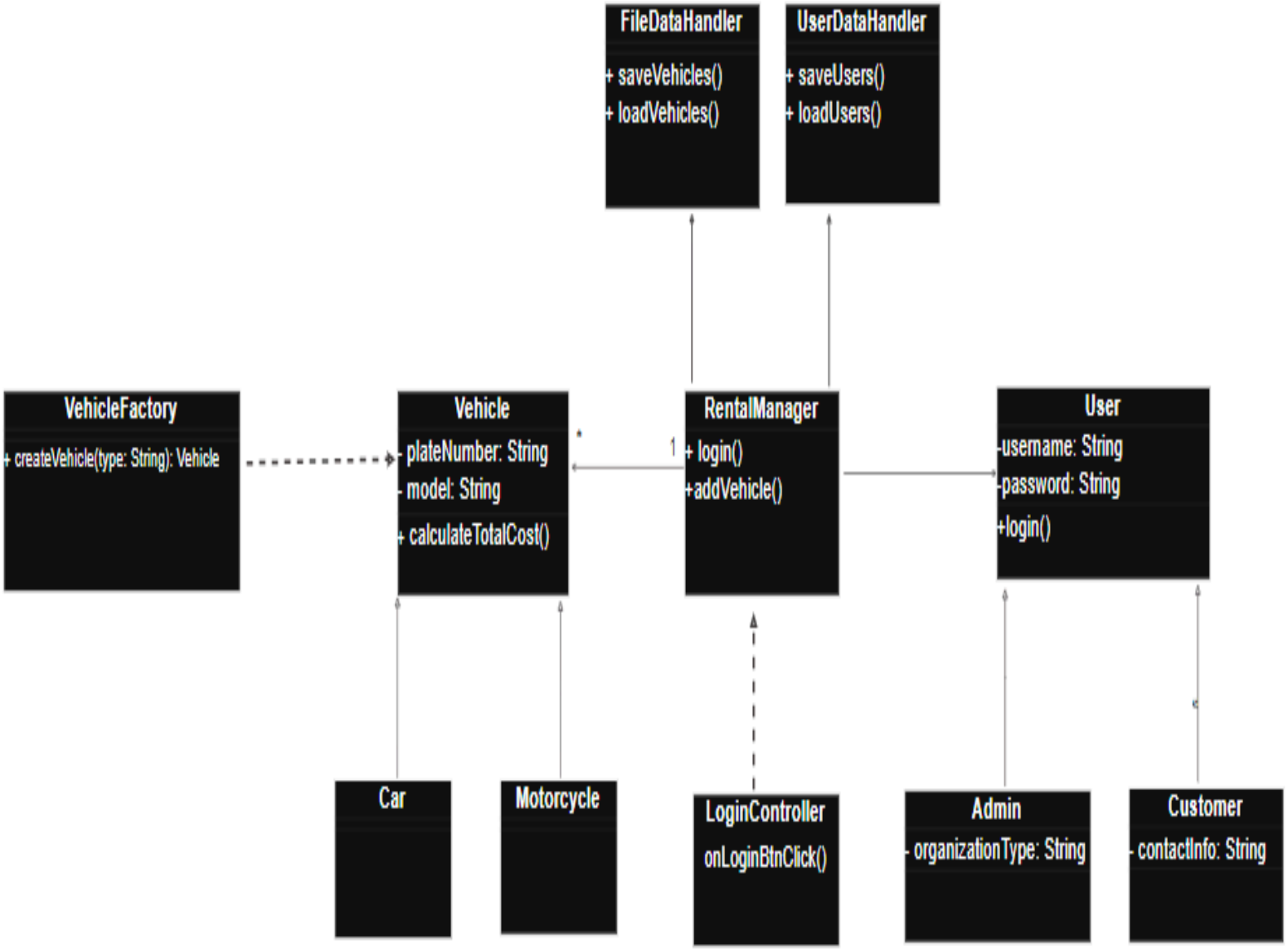
Explanation of OOP Concepts Used

Encapsulation: All class attributes (like plateNumber and password) are set to private. Access is controlled via public getters and setters to protect data integrity.
Abstraction: The Vehicle and User classes are declared as abstract. This ensures that common code is shared, but no "generic" vehicle or user can be created without being specific (e.g., a Car or a Customer).
Inheritance: We used the extends keyword so that Car inherits features from Vehicle. This reduces code duplication.
Polymorphism: The calculateTotalCost() method is defined in the parent class but behaves differently depending on the specific vehicle type.

SOLID Principles Applied

Interface Segregation Principle (ISP): we have Bookable for customers and Manageable for admins, so customers aren't forced to see methods they don't use (like deleteVehicle())."
Open/Closed Principle (OCP): The system is designed so that new vehicle types (like Motorcycles) can be added without modifying the existing RentalManager code.
Single Responsibility (SRP): The FileDataHandler class is solely responsible for file operations, while the RentalManager handles business logic.
Liskov Substitution (LSP): Any method that expects a Vehicle can accept a Car without causing errors.
Dependency Inversion (DIP): High-level controllers depend on the RentalManager abstraction rather than directly communicating with the raw data file.
Multi-Role Access: Separate dashboards for Admins (Inventory Control) and Customers (Booking).
Fleet Management (CRUD): Admins can Create, Read, and Delete vehicle records in real-time.
Booking System: Customers can view only available vehicles and rent them with a single click.

Data Persistence: All data is saved to and loaded from a local vehicles.txt file using Java I/O.Input Validation: The system prevents errors by validating user inputs during login and data entry.



Hello!

Sign In / Register

Username

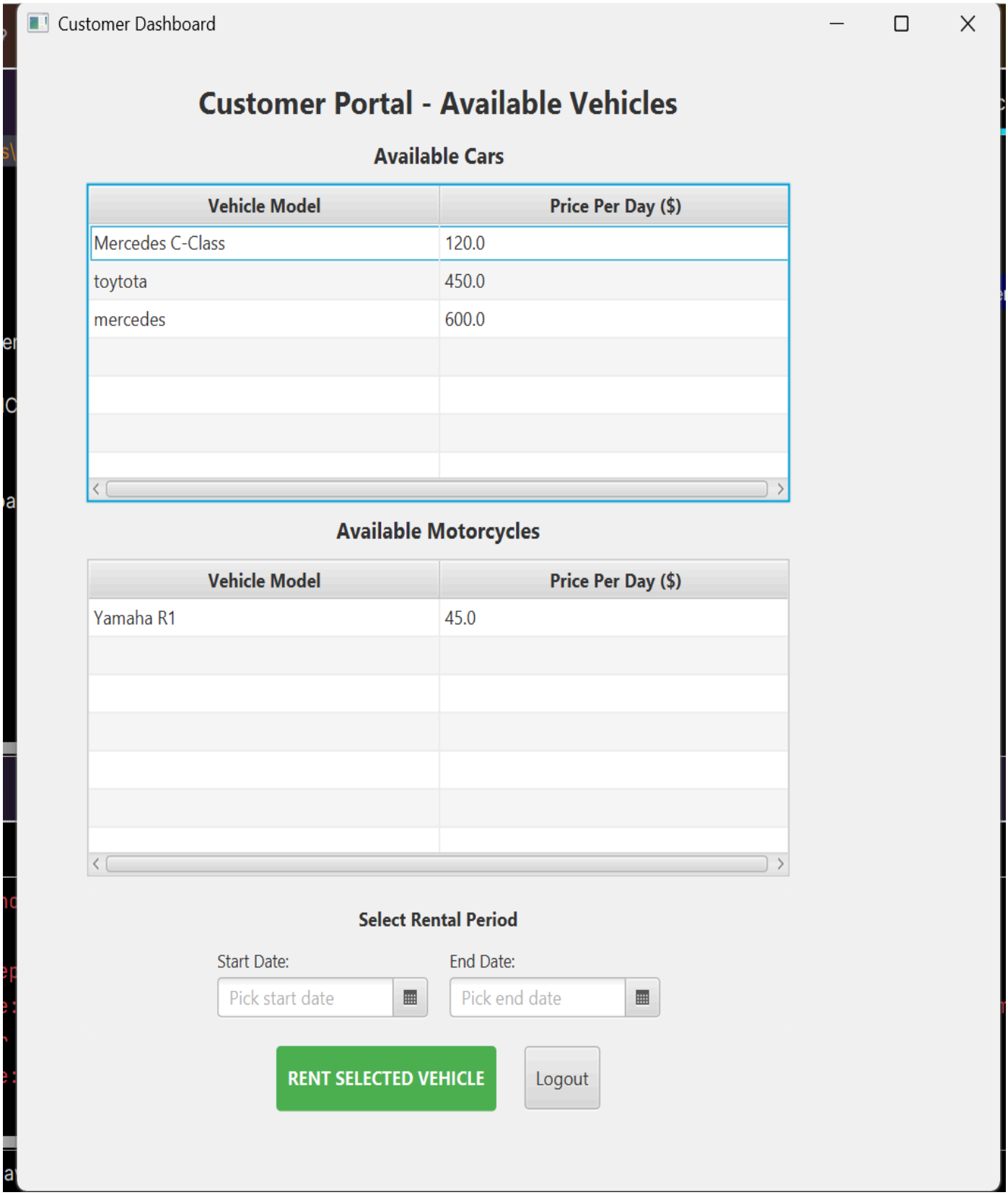
Password

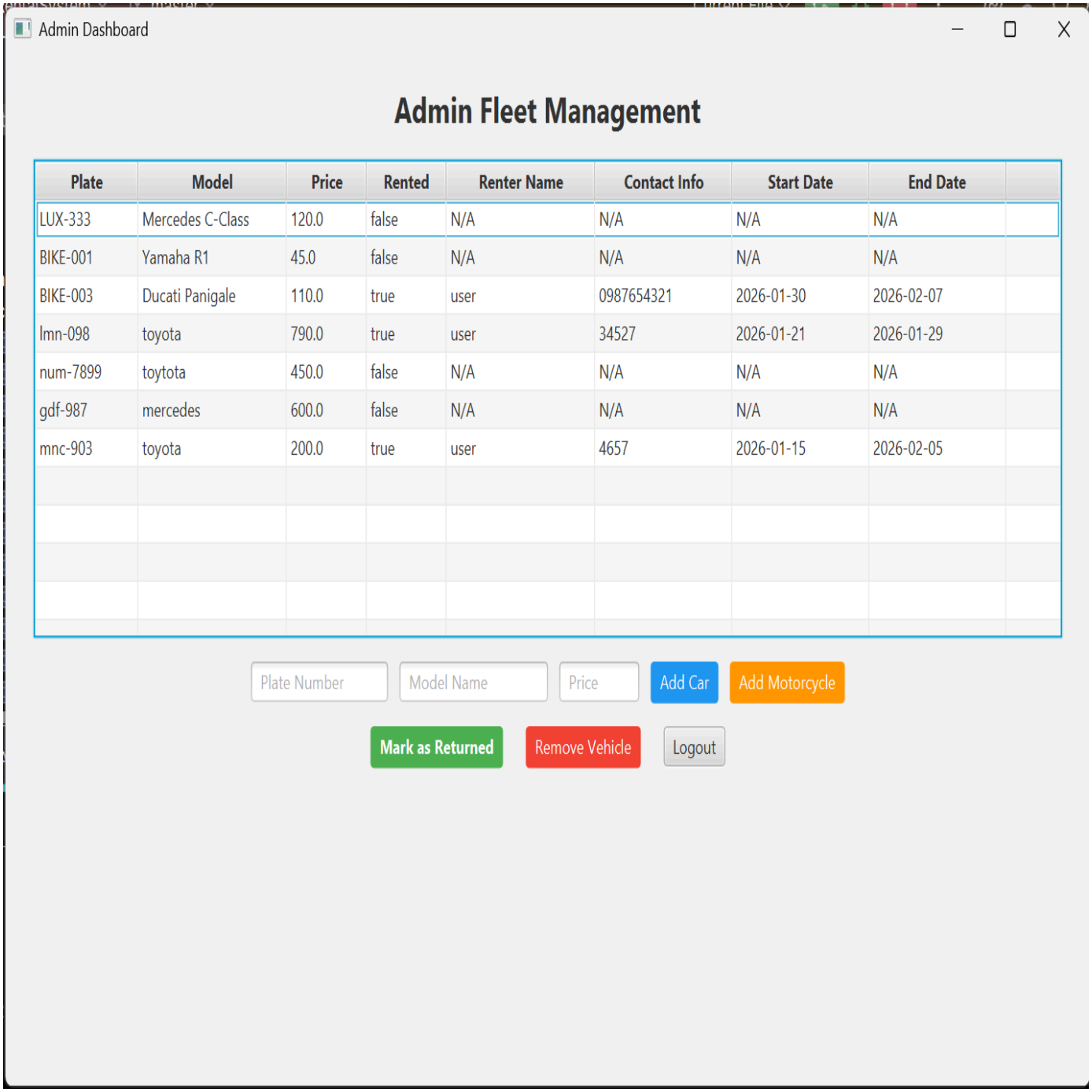
Contact Number (e.g., 050...)

Sign In

New here? Fill all fields and click below:

Sign Up Now


Customer Dashboard
Customer Portal - Available Vehicles
Available Cars
Vehicle Model
Price Per Day ($)
Mercedes C-Class
120.0
toytota
450.0
mercedes
600.0
Available Motorcycles
Vehicle Model
Price Per Day ($)
Yamaha R1
45.0
Select Rental Period
Start Date:
Pick start date
End Date:
Pick end date
RENT SELECTED VEHICLE
Logout



How to Run

Admin: admin / admin123
Customer: customer / user123