Table of Contents

1. Introduction……………………………………………………………………………………1
2. SystemOverview…………………………………………………………………………… 2
3. OOD Principles Applied…………………………………………………………………..…3
4. UML Diagrams……………………………………………………………………………………4
   1. Use Case Diagram…………………………………………………………………………5
   2. Class Diagram………………………………………………………………………………6
   3. Sequence diagram…………………………………………………………………………7
   4. Activity Diagram**……………………………………………………………..8**
5. Implementation Details………………………………………………………………………9

5.1 Key Classes and Their Roles………………………………………………………………10

**Object-Oriented Design (OOD) Documentation for Parking Management System**

**1. Introduction**

This document outlines the object-oriented design for the Parking Management System. The project integrates various functionalities including user authentication, vehicle management, parking slot booking, and payment processing—all implemented using object-oriented programming principles in Java. The design emphasizes the principles of abstraction, encapsulation, inheritance, and polymorphism to create a modular and maintainable system.

**2. System Overview**

The Parking Management System provides a console-based interface for managing parking facilities. The key functions include:

* **User Authentication:** Log in, registration, and password reset.
* **Vehicle Management:** Adding vehicles to the system.
* **Parking Slot Booking:** Reserving available parking slots.
* **Payment Processing:** Managing payments made by users.

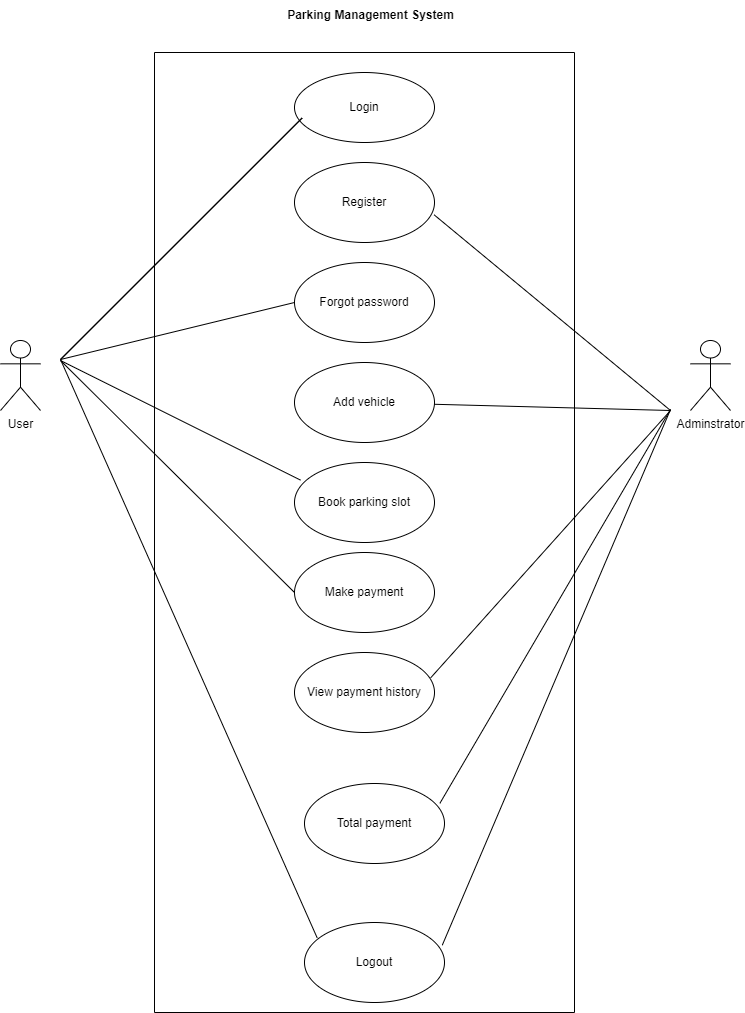
The system differentiates between **admin** and **regular user** roles, offering additional functionalities (such as viewing reports and registering new users) exclusively to administrators.

**3. OOD Principles Applied**

* **Abstraction:** The system hides implementation details behind well-defined interfaces. For example, the ParkingManager class abstracts the process of adding vehicles and booking slots.
* **Encapsulation:** Each class encapsulates its attributes and behaviors. Private attributes are accessed or modified via public methods.
* **Inheritance:** Common behavior (e.g., user authentication and registration) is centralized in classes such as UserManager and reused as needed.
* **Polymorphism:** Although the current implementation is console-based, the design allows for future enhancements (e.g., using different types of payment processing) by leveraging polymorphic behavior.

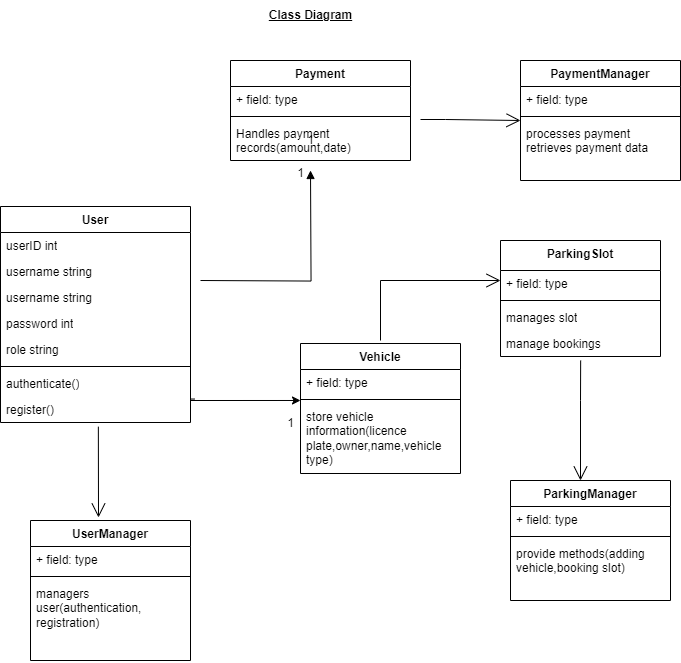
**4. UML Diagrams**

**4.1 Use Case Diagram**

****

* **Actors:**
  + **Admin:** Manages user registration, views payment history, and total payments.
  + **User:** Logs in, adds vehicles, books parking slots, makes payments, and logs out.
* **Key Use Cases:**
  + User Login
  + Register User (Admin Only)
  + Add Vehicle
  + Book Parking Slot
  + Make Payment
  + View Payment History (Admin Only)
  + View Total Payments (Admin Only)
  + Logout

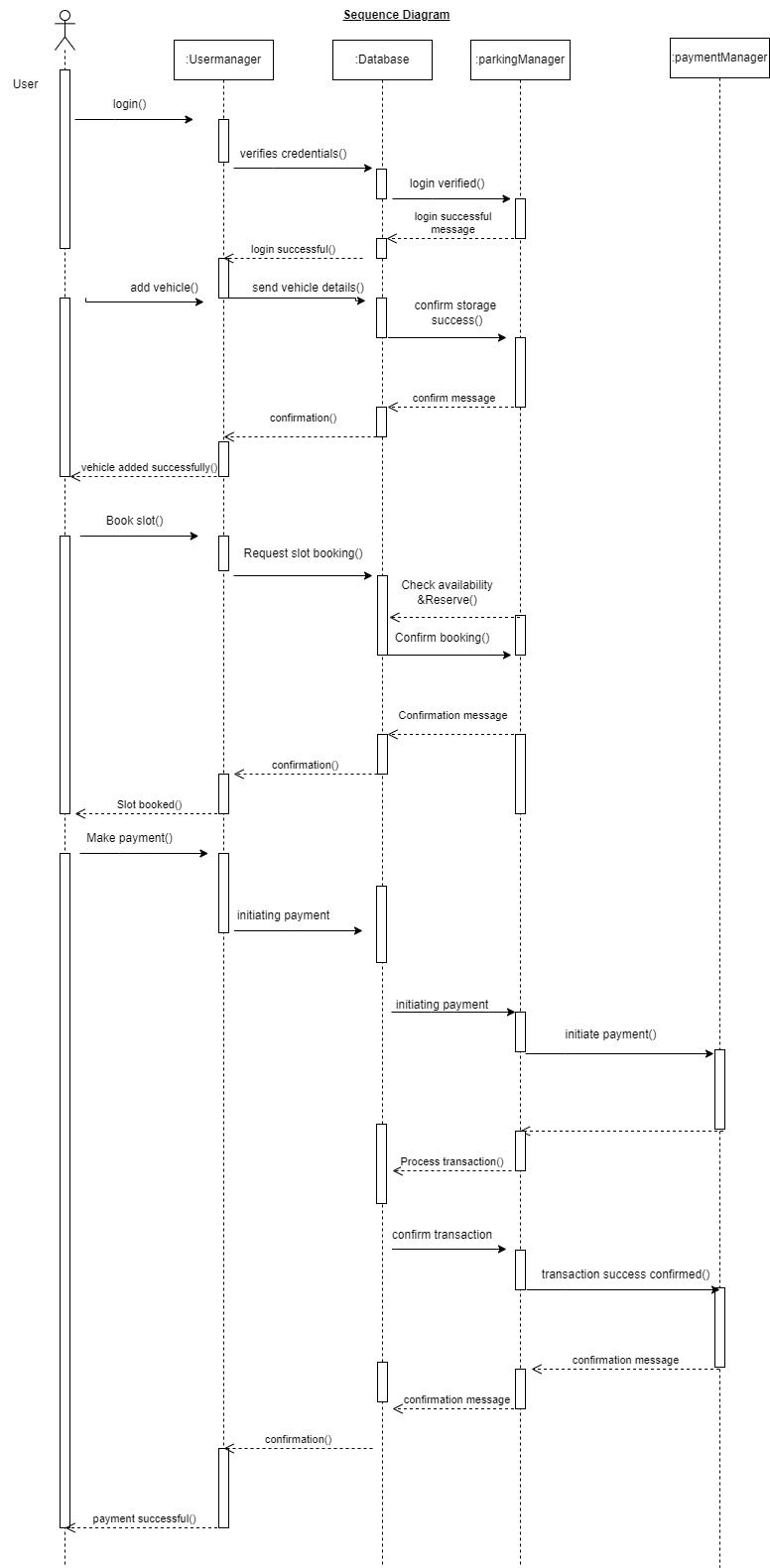
**4.2 Class Diagram**

****

The class diagram includes the following key classes:

* **User:** Contains attributes like userID, username, password, and role; methods include authenticate() and register().
* **ParkingSlot:** Manages slot details such as slotID and status (occupied/free), with methods to book and release a slot.
* **Vehicle:** Represents vehicle information including licensePlate, ownerName, and vehicleType.
* **Payment:** Handles payment records including paymentID, amount, and timestamp, with methods for processing payments.
* **ParkingManager:** Provides functionality for adding vehicles and booking parking slots.
* **UserManager:** Manages user authentication and registration using MySQL integration.
* **PaymentManager:** Processes payments and retrieves payment history and totals.

**4.3 Sequence Diagram**

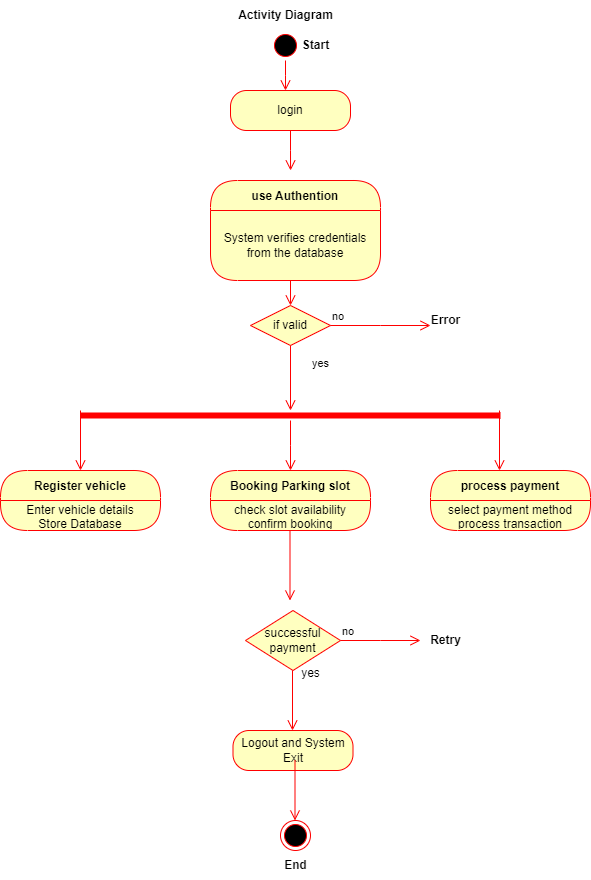
****

The sequence diagram illustrates interactions between objects:

* **User** initiates login by providing credentials.
* **UserManager** verifies credentials with the database and returns the user role.
* Upon successful login, the **User** selects options (e.g., adding a vehicle, booking a slot, processing a payment).
* The corresponding managers (ParkingManager or PaymentManager) interact with the database to execute the requested operations.

**4.4 Activity Diagram**

The activity diagram outlines the system’s workflow:

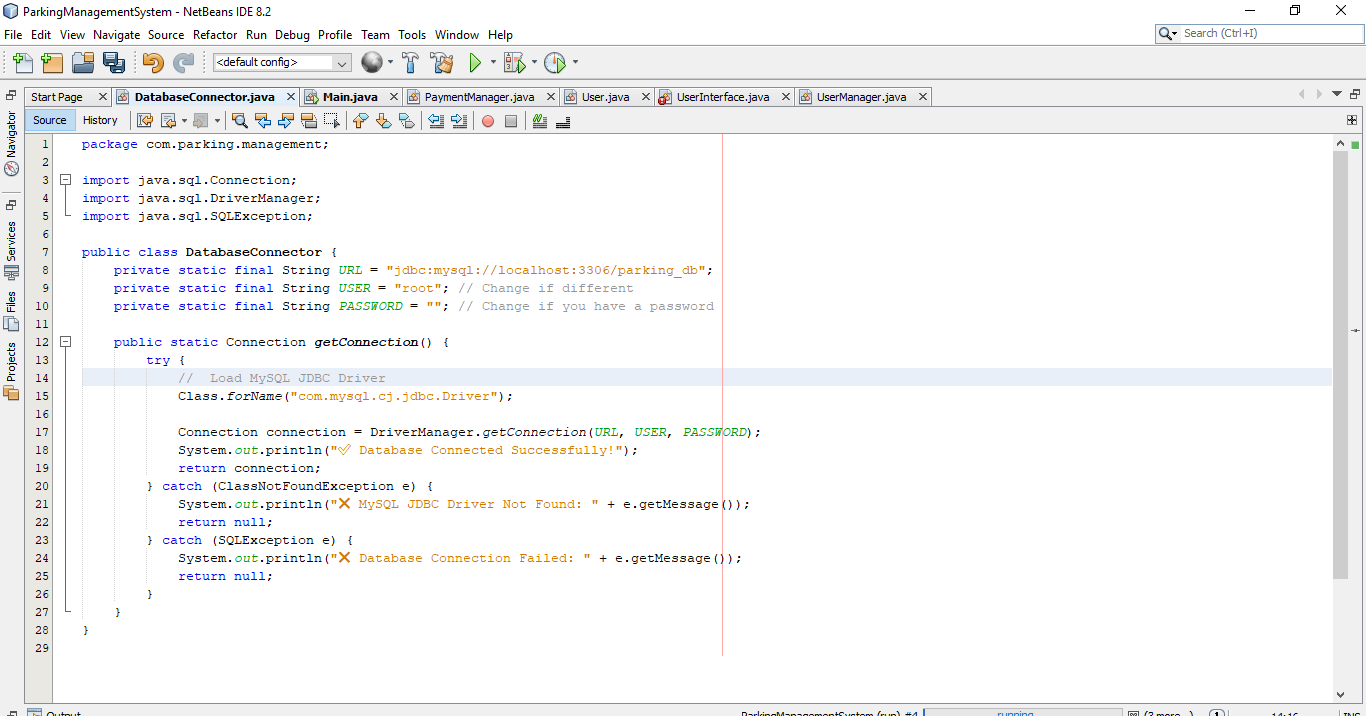


1. **Login:** The user enters credentials, and the system verifies them.
2. **Menu Selection:** Based on the user's role, the system displays appropriate options.
3. **Operation Execution:** The user selects an operation (e.g., add vehicle, book slot, make payment).
   * For vehicle addition: Input vehicle details and store them.
   * For slot booking: Choose a slot and update its status.
   * For payment: Input payment details, and process the payment.
4. **Logout:** The user logs out, and the session ends.

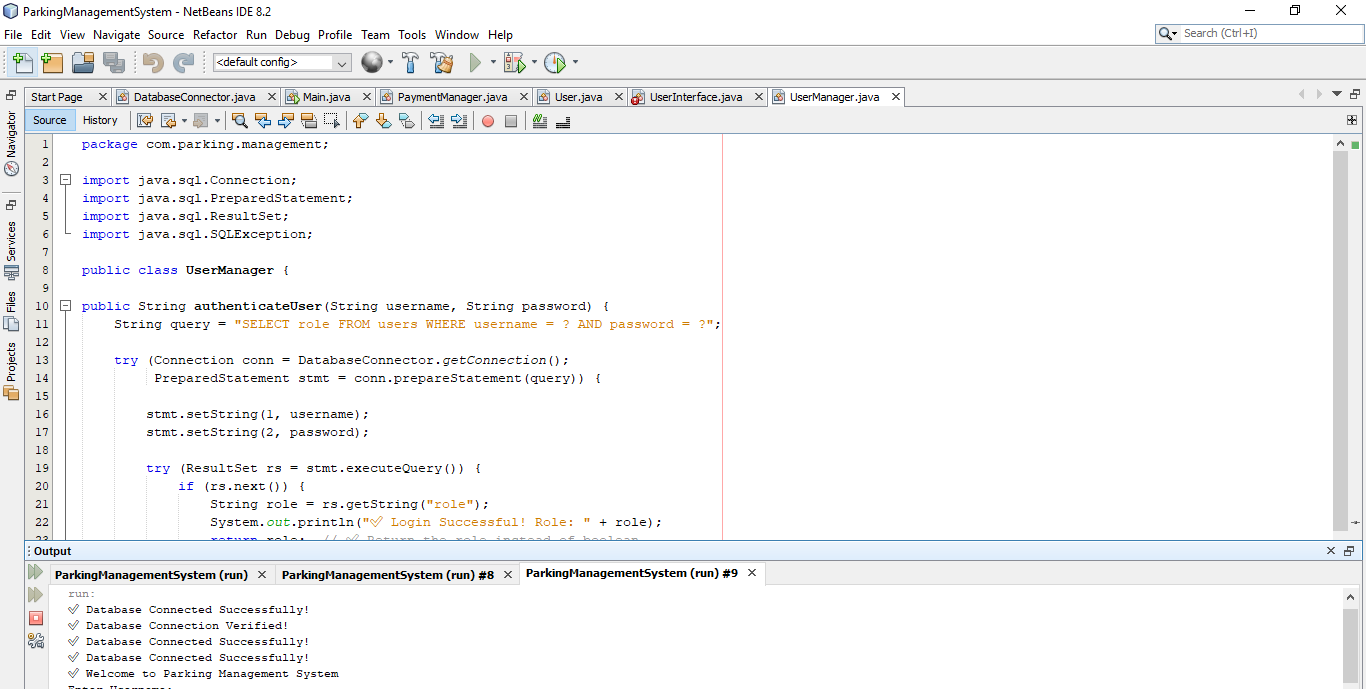
**5. Implementation Details**

**Key Classes and Their Roles**

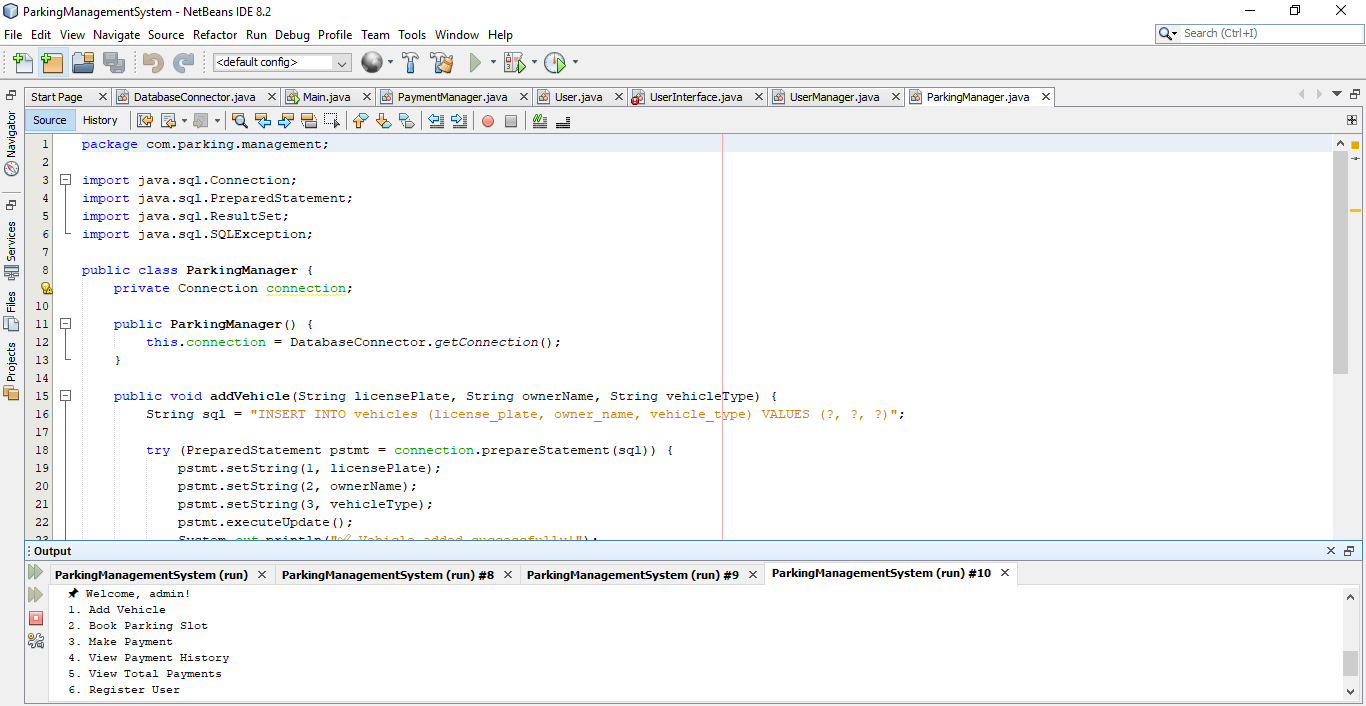
* **DatabaseConnector:** Provides a static method to establish a connection with the MySQL database.

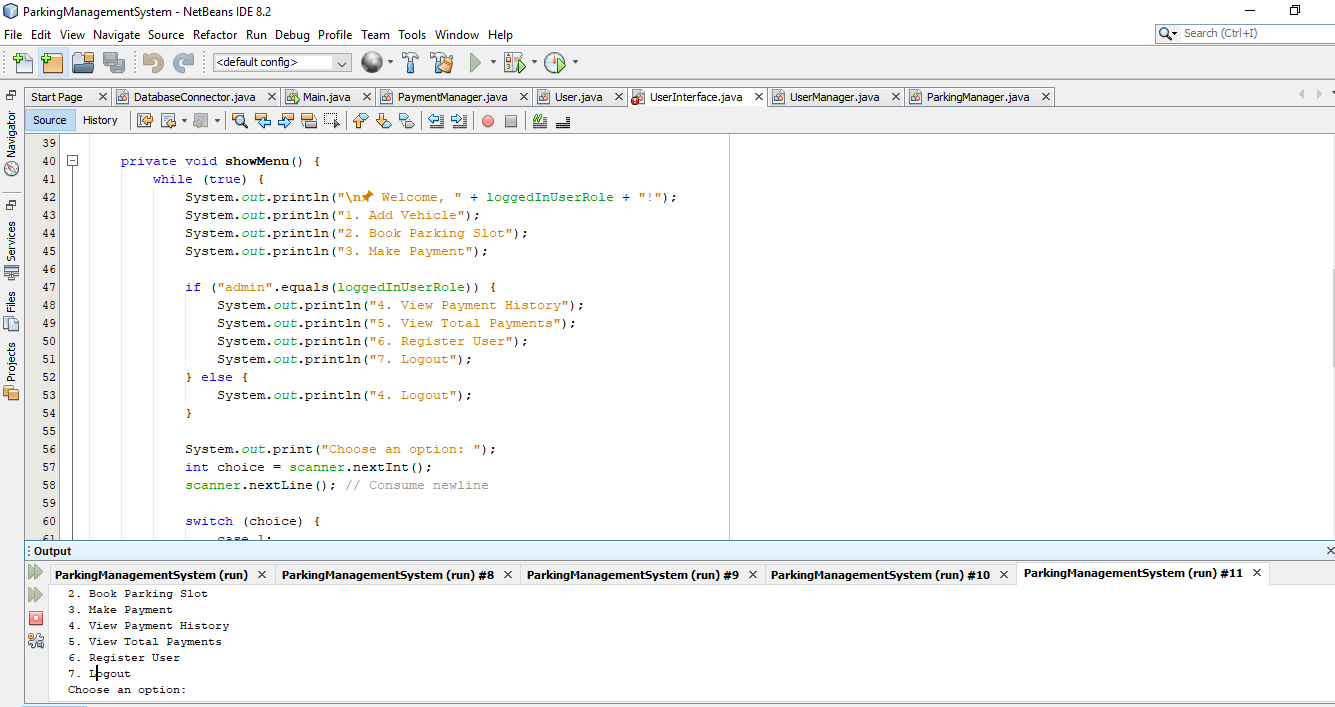


* **UserManager:** Contains methods for user authentication (authenticateUser) and registration (registerUser), interfacing with the users table in MySQL.



* **ParkingManager:** Implements functionalities such as adding a vehicle and booking a parking slot.



* **PaymentManager:** Handles payment processing, including storing payment records and retrieving payment history.
* **UserInterface:** Provides the console-based user interface, handling login, menu navigation, and invoking methods from the manager classes.
* 

**Code Organization**

The project is organized into a package named com.parking.management. Each class is responsible for a specific aspect of the system, ensuring modularity and easier maintenance.

**6. Conclusion**

The object-oriented design of the Parking Management System demonstrates how encapsulation, abstraction, inheritance, and polymorphism can be leveraged to create a modular and scalable system. The use of UML diagrams has helped clarify the system’s structure and interactions, forming a solid foundation for further development and future enhancements.