CS-3004: Software Design and Analysis

Final Deliverable - Documentation

Ayishah 22i-0957

Mohib Ullah Iftikhar 22i-1044

Maryam Farooq 22i-1217

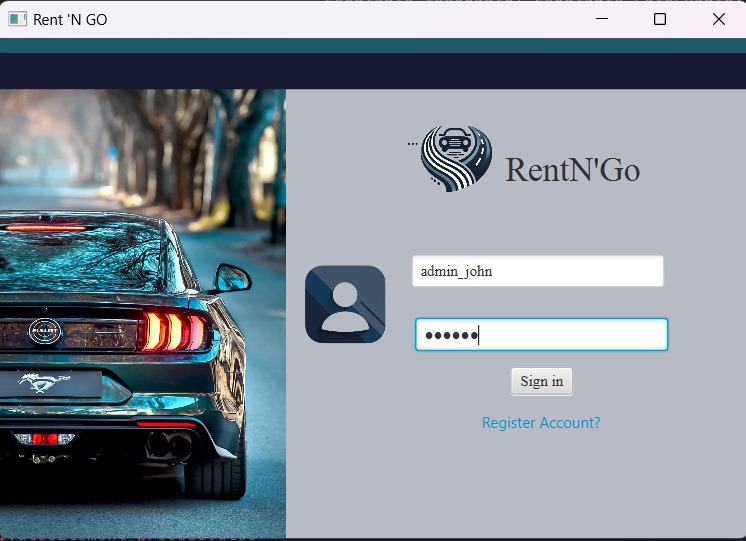
**Introduction:**

The RentN’Go Car Rental System is an application developed in Java which is meant to assist users who need a vehicle for recreational or other needs at an affordable fee. Due to the increased need for on-demand transportation, we also provide a more economical option when compared to taxi services by allowing users the liberty of hiring a car. Our vehicles are available in different classes, from low-priced to luxurious cars, and users can choose a specific class according to their preferred size, model, color, or make. The possibilities are also extended by providing a driver per request. Depending on the activity performed, the system has 2 roles: Admin and Customer, ensuring easy management as well as easy utilization of the application. Adverts, car bookings, driver allocation, and customer management account are all within the scope of duties of the admins, while the capabilities of the customers are making their bookings, changing their accounts, and making payments. RentN’Go will reshape the direction of car hirers by altering the viewpoint of car rental systems to offer more flexibility and efficiency to administrators.

**UI in JavaFX:**

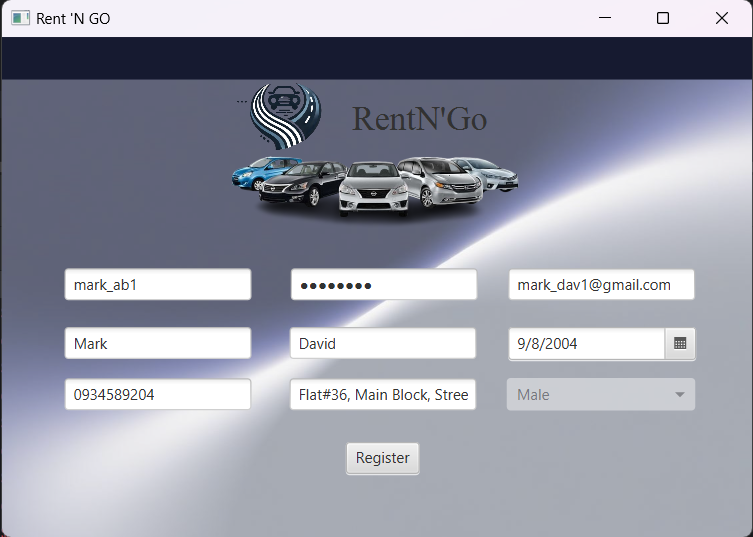
1. *LogIn Form:*

The Login Form has a Username and Password Field. The Password Field was chosen for the password to ensure security. The Register Account link redirects the User to the Registration form. The form interacts with the business logic layer to validate credentials against database and ensure that the admin is redirected to the admin forms and the customer is redirected to the customer form. For the design we added a car-themed background, our branding logo along with the name, and a user-friendly layout with a professional color scheme.



1. *Registration Form:*

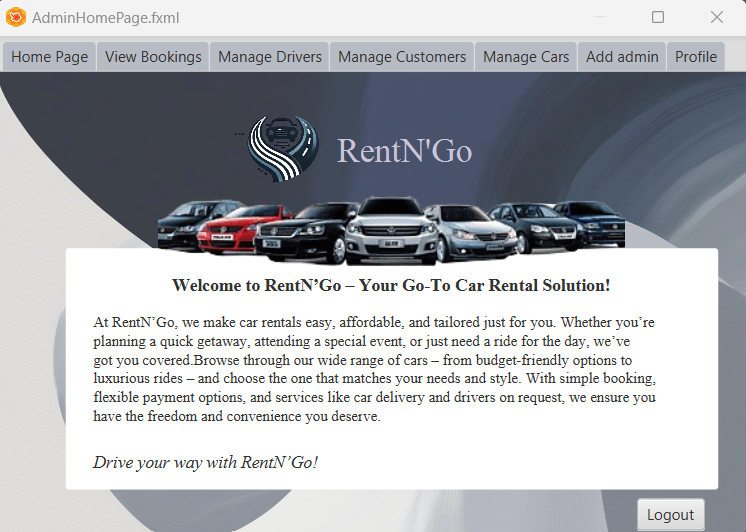
This form has fields for username, password, email, first/last name, dob, phone number, address and gender. DatePicker was used for dob and a ComboBox for gender to ensure ease of input for the user. After clicking the register Button, the user details are submitted to the business logic layer for validation and database storage and the user is redirected back to the Login page.



1. *Admin View Form:*

* Admin Homepage:

The RentN’Go admin homepage serves as a central hub for managing the entire car rental operation. Upon successful login, admins are presented with a user-friendly interface featuring a top menu bar with options for viewing bookings, managing drivers, customers, and cars, as well as adding new admins and accessing their own profile. The homepage also displays a welcome message and showcases a selected of available rental vehicles, providing a visual overview of the service offerings. Moreover, it also has a Logout buttons which redirects the admin back to LogIn form.



* View Bookings

This form displays fields for key booking details such as booking ID, car ID, user ID, driver ID, issue date, return date, payment status, and total charges. Upon selecting the Booking Id from the ComboBox, the relevant data is displayed. Additionally, a "Delete Booking" button is provided for managing existing bookings as needed. This centralized view empowers admins to efficiently monitor and oversee the rental process.

* Manage Drivers
  + - Add Driver

The "Add Driver" section of the admin homepage is designed to facilitate the efficient addition of new drivers to the car rental system. It provides a user-friendly interface with fields for entering essential driver information, including first name, last name, email, date of birth, phone number, address, and gender. Additionally, a dropdown menu is provided for selecting the driver's gender. Once all the required information is entered, the admin can click the "Add" button to submit the new driver details to the system. This streamlined process ensures that new drivers can be added quickly and accurately, contributing to the smooth operation of the car rental service

* + - Update Driver

The "Update Driver" section of the admin homepage is designed to facilitate the efficient modification of existing driver information within the car rental system. It provides a user-friendly interface with fields for updating essential driver details, including first name, last name, email, date of birth, phone number, address, and gender. It originally displays the info of the driver selected from the ComboBox, once the necessary changes are made, the admin can click the "Update" button to submit the updated driver information to the system.

* + - View Driver

The "View Driver" section of the admin homepage provides a convenient way for admins to access and review detailed information about individual drivers within the car rental system. The interface features a ComboBox where admins can select the specific driver they wish to view. Once a driver is selected, the system displays their key information, including first name, last name, email, date of birth, phone number, address, and gender. Additionally, a "Delete" button is provided, allowing admins to remove driver records from the system if necessary. This feature empowers admins to maintain accurate and up-to-date driver information, ensuring efficient management of the rental service.

* Manage Customers
  + - Add Customer

The "Add Customer" section of the admin homepage is designed to facilitate the efficient addition of new customers to the car rental system. It provides a user-friendly interface with fields for entering essential customer information, including username, password, email, first name, last name, date of birth, phone number, address, and gender. Additionally, a ComboBox is provided for selecting the customer's gender. Once all the required information is entered, the admin can click the "Add" button to submit the new customer details to the system.

* + - View Customer

The "View Customer" section of the admin homepage provides a convenient way for admins to access and review detailed information about individual customers within the car rental system. The interface features a dropdown menu where admins can select the specific customer they wish to view. Once a customer is selected, the system displays their key information, including username, email, first name, last name, date of birth, phone number, address, and gender. Additionally, a "Delete" button is provided, allowing admins to remove customer records from the system if necessary. This feature empowers admins to maintain accurate and up-to-date customer information, ensuring efficient management of the rental service.

* Manage Cars
  + - Add Car

The "Add Car" section of the admin homepage is designed to facilitate the efficient addition of new cars to the car rental system. It provides a user-friendly interface with fields for entering essential car information, including name, model, color, availability, charges per day, and sunroof availability. Once all the required information is entered, the admin can click the "Add" button to submit the new car details to the system. This streamlined process ensures that new cars can be added quickly and accurately, contributing to the smooth operation of the car rental service.

* + - View Car

The "View Car" section of the admin homepage provides a convenient way for admins to access and review detailed information about individual cars within the car rental system. The interface features a dropdown menu where admins can select the specific car they wish to view. Once a car is selected, the system displays its key information, including name, model, color, availability, charges per day, and sunroof availability. Additionally, a "Delete" button is provided, allowing admins to remove car records from the system if necessary. This feature empowers admins to maintain accurate and up-to-date car information, ensuring efficient management of the rental service.

* Add Admin

The "Add Admin" section of the admin homepage is designed to facilitate the efficient addition of new administrators to the car rental system. It provides a user-friendly interface with fields for entering essential admin information, including username, password, email, first name, last name, date of birth, phone number, address, and gender. Additionally, a dropdown menu is provided for selecting the admin's gender. Once all the required information is entered, the admin can click the "Add" button to submit the new admin details to the system. This streamlined process ensures that new admins can be added quickly and accurately, contributing to the effective management of the car rental service.

* Profile

The "Profile" section of the admin homepage is designed to allow admins to view and update their own personal information within the car rental system. The interface displays a user icon representing the admin's profile and provides fields for viewing and editing their username, password, email, first name, last name, date of birth, phone number, address, and gender. Additionally, a dropdown menu is provided for selecting the admin's gender. Once the necessary changes are made, the admin can click the "Update" button to submit the updated profile information to the system. This feature empowers admins to maintain accurate and up-to-date personal information, ensuring efficient management of their own account within the system.

1. *Customer View Form:*

* Customer Homepage

The **Customer Homepage** serves as the central hub for RentN'Go customers. Upon successful login, customers are greeted with a user-friendly interface featuring a top menu bar with options for accessing their bookings, making payments, and viewing their profile. The homepage also displays a welcoming message and showcases a selection of available rental vehicles, providing a visual overview of the service offerings. Moreover, it also has a Logout buttons which redirects the customer back to LogIn form.

* Bookings
  + - View Bookings

The section displays key details of each booking, including the booking ID, car ID, driver ID, issue date, return date, total charges, and payment status. Customers can conveniently select a specific booking from the dropdown menu to view its information in detail. The "Return Car" button facilitates the return process, allowing customers to complete their rental and redirect them to the payment tab.

* + - Add Booking

The "Add Booking" section of the customer homepage provides a user-friendly interface for customers to initiate new car rentals. It allows them to select a car by clicking on the ‘Choose a Car’ button which opens a new form that displays all the available Cars based on the car Id selected from the ComboBox. After that the user can enter the car Id and choose a driver based on the driver's age and ID, specify the desired rental period (issue date and return date), and view the total charges associated with the booking. Once all the necessary information is entered, the customer can click the "Confirm" button to finalize the booking process. This streamlined approach empowers customers to easily and efficiently book cars that meet their specific needs.

* + - Update Booking

The "Update Booking" section of the customer homepage provides a user-friendly interface for customers to modify existing car rental bookings. It allows them to enter the booking ID and specify the desired return date. Once the necessary information is entered, the customer can click the "Confirm" button to update the booking. This feature empowers customers to make changes to their bookings as needed, ensuring flexibility and convenience in their rental experience.

* Payment

The "Payment" section of the customer homepage provides a user-friendly interface for customers to make payments for their car rental bookings. It allows them to enter the booking ID, select the preferred mode of payment from a dropdown menu, and specify the amount to be paid. Once all the necessary information is entered, the customer can click the "Confirm" button to initiate the payment process. This streamlined approach ensures a convenient and efficient payment experience for customers.

* Profile

The “Profile" section of the Customer homepage is designed to allow customers to view and update their own personal information within the car rental system. The interface displays a user icon representing the customer’s profile and provides fields for viewing and editing their username, password, email, first name, last name, date of birth, phone number, address, and gender. Additionally, a dropdown menu is provided for selecting the customer’s gender. Once the necessary changes are made, the admin can click the "Update" button to submit the updated profile information to the system. This feature empowers admins to maintain accurate and up-to-date personal information, ensuring efficient management of their own account within the system.

**Business Logic Layer (OOP Principles):**

1. Encapsulation:

In the ‘Driver’, ‘User’, ‘Payment’, and related classes, each class has private attributes and public getter and setter methods. This encapsulates the data and ensures that it can only be accessed or modified through the public methods, providing a controlled and secure way to interact with the data.

1. Inheritance:

The ‘Admin’ and ‘Customer’ class inherit from the ‘User’ class. These subclasses extend the general ‘User’ class, and while they inherit the properties and methods of the ‘User’ class, they also have specific behaviors and attributes (‘type’) that differentiate them.

1. Abstraction:

The ‘UserManagement’ and the rest management classes abstract away the details of how users and other classes are managed. The user doesn’t need to know how the data is stored in the database or how drivers are added/removed from the list; they only interact with the higher-level methods. The ‘addDriver()’ method allows admin to add a driver, but the underlying logic of how drivers are stored and managed in the database is hidden from the user.

**Database:**

MySQL is used for data storage, with tables for Users, Cars, Bookings, and Payments. The CRUD operations are handled by methods addBooking(), getCarDetails(), etc.

The persistence handler interacts with the database to store and retrieve data, ensuring data integrity through ‘Database.java’. The UI collects user input and sends requests to the business logic layer, the business logic layer communicates with the persistence handles to interact with the database and perform necessary operations. Data is stored in MySQL and retrieved for UI updates using the PROCs.

**Gang of Four (GoF) patterns:**

1. The ***CarRentalManagement*** class uses the Façade Design Pattern from the GoF Patterns to provide a simplified interface to a set of related subsystems while hiding their complexity. The class consolidates and abstracts operations from multiple subsystems like our Management classes and Database. This makes it easier for our clients to interact with the system as they only need to work with the CarRentalManagement Class instead of directly interacting with the individual subsystems. Instead of directly calling ‘userManager.addUser’, the client calls addUser on CarRentalManagement. Moreover, the internal workings of the individual management classes are hidden from the client, the client does not need to know about how DataBase is used internally or the relationships between these subsystems.
2. The ***UserManagement*** class uses the Factory Method Pattern. The ‘addUser’ method employs a factory-like approach by instantiating different types of users based on the type parameter.
3. The ***BookingManagement*** class uses the Factory Method Pattern. The ‘addBooking’ method creates and initializes a Booking object with all required parameters before adding it to the system. It encapsulates the creation logic and ensures that every booking is initialized correctly.

**GRASP (General Responsibility Assignment Software Patterns) Principles:**

1. **CarRentalManagement**

|  |  |
| --- | --- |
| Controller | It manages system-level requests (e.g., ‘addUser’, ‘updateCar’) and delegates them to appropriate subsystems like ‘UserManagement’ and ‘CarManagement’. |
| High Cohesion | It groups responsibilities logically by separating functionalities into subsystems (e.g., users, cars, bookings), reducing duplication and ensuring maintainability. |
| Low Coupling | It minimizes dependencies on subsystem implementations by interacting with subsystems through well-defined interfaces, making it resilient to changes in subsystem details. |
| Pure Fabrication | It is a fabricated class for handling system-level responsibilities. It mediates between the client and subsystems, improving reusability and decoupling domain logic from the UI. |
| Information Expert | It delegates responsibilities to the management classes, ensuring proper encapsulation (e.g., BookingManagment handles bookings because it has access to booking data). |
| Creator | It creates subsystem instances. |

1. **UserManagement**

|  |  |
| --- | --- |
| Controller | It processes requests like ‘addUser’ and ‘searchUserbyID’ by delegating to the appropriate methods or classes. |
| High Cohesion | It groups all user-related responsibilities within UserManagement. |
| Low Coupling | It interacts with ‘DataBase’ without depending on its implementation as the database logic is isolated within DataBase, protecting UserManagement from direct impact. |
| Information Expert | It manages ‘userList’ and handles user-related operations like adding, updating and deleting users. |
| Creator | Creates ‘Customer’ and ‘Admin’ instances when adding users. |

1. **DriverManagement**

|  |  |
| --- | --- |
| Controller | Manages operations such as creating, updating, searching, and deleting drivers. |
| High Cohesion | All methods contribute to managing driver data, ensuring clarity and maintainability. |
| Low Coupling | It delegates database operations to DB, focusing only on driver management. |
| Information Expert | Possesses the knowledge to search, update, and filter drivers using ‘driverList’ and database. |
| Creator | Responsible for creating ‘Driver’ objects. |

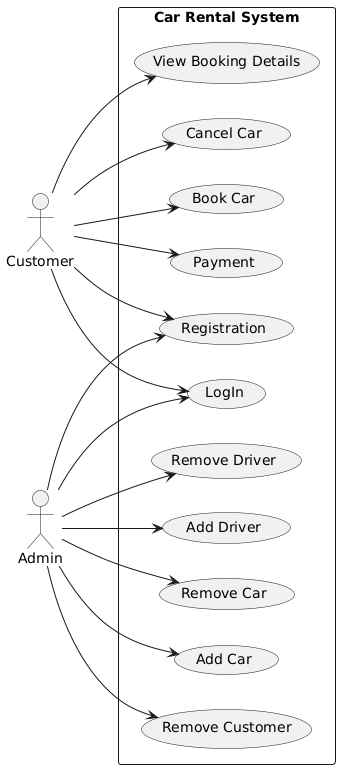
1. **CarManagement**

|  |  |
| --- | --- |
| Controller | Manages car related operations like add, delete, update and search. |
| High Cohesion | Groups all car-related responsibilities logically. |
| Low Coupling | Keeps minimal dependency on other components. |
| Information Expert | Serves as the expert for all operations on Car objects. |
| Creator | Responsible for creating Car objects. |

1. **BookingManagement**

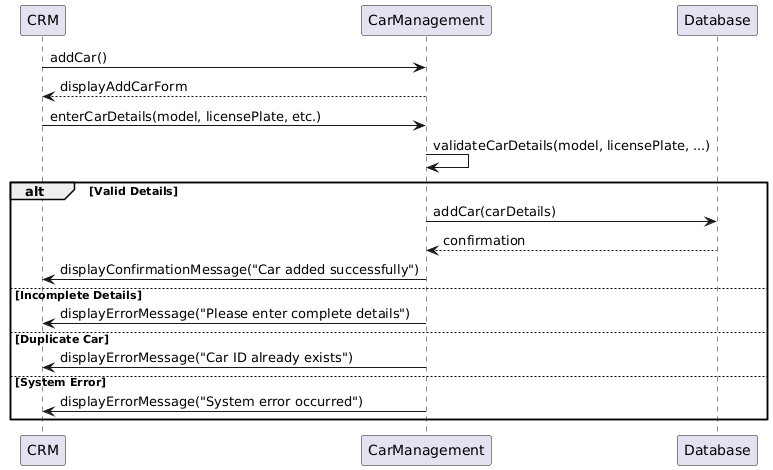
|  |  |
| --- | --- |
| Controller | Handles booking-related operations like adding, updating, and deleting bookings. |
| High Cohesion | Focuses exclusively on booking management. |
| Low Coupling | Encapsulates booing logic while delegating persistence tasks to DB. |
| Information Expert | Expert for managing booking-related data and operations. |
| Creator | Creates Booking objects using the addBooking method. |

**Use Case Diagram:**

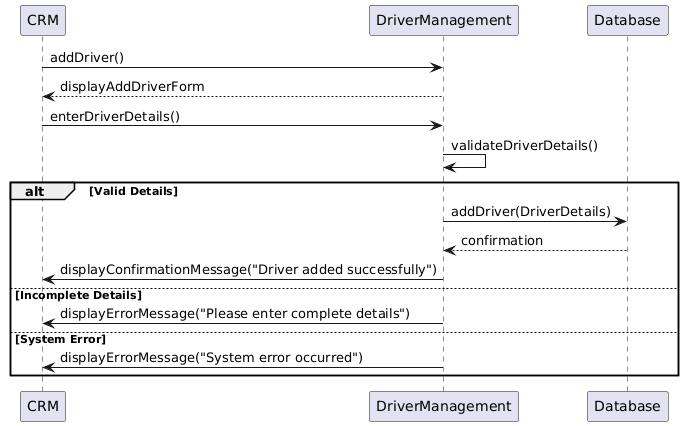


**System Diagrams (SD):**

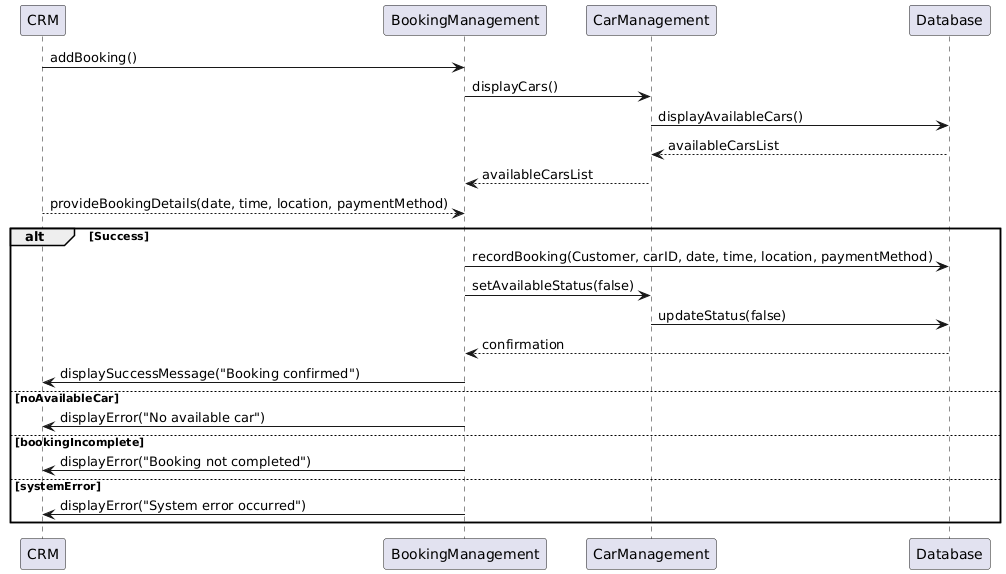
1. **Add Car:**



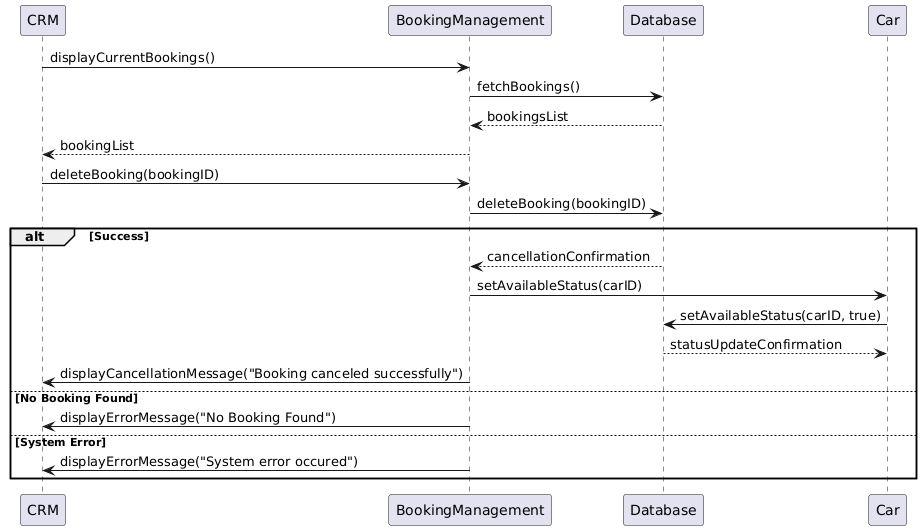
1. **Add Driver:**



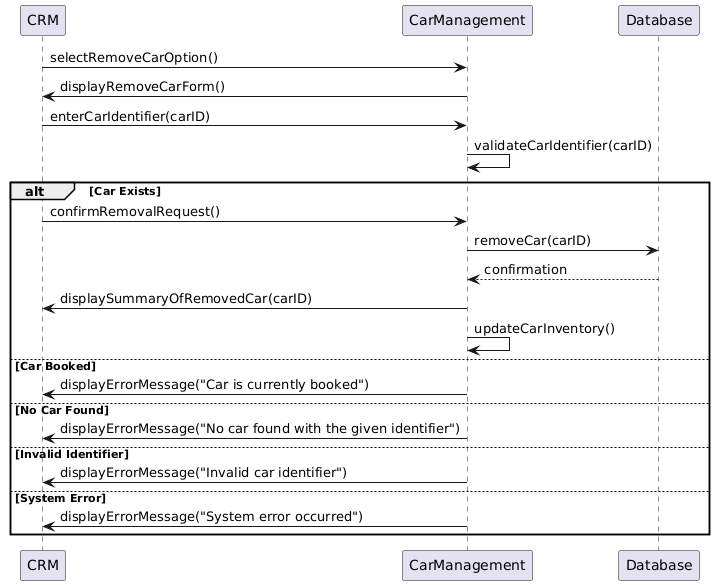
1. **Add Booking:**



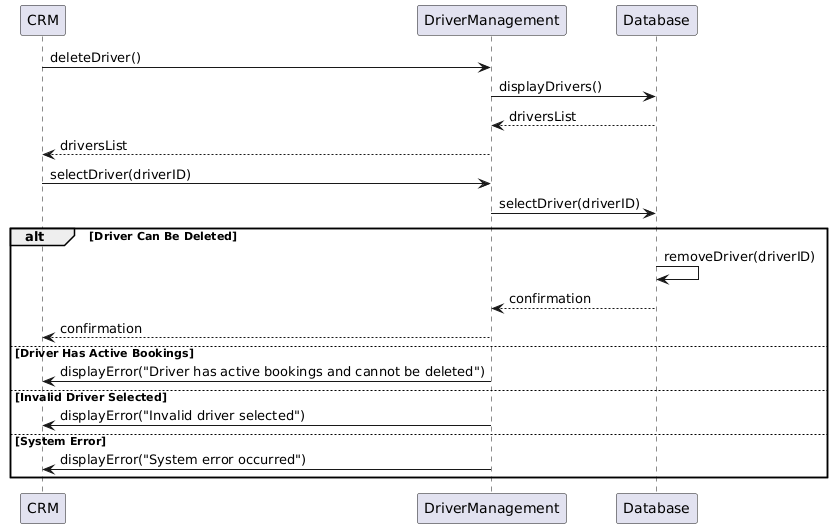
1. **Cancel Booking:**



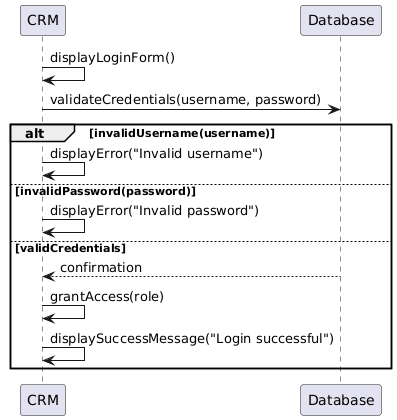
1. **Delete Car:**



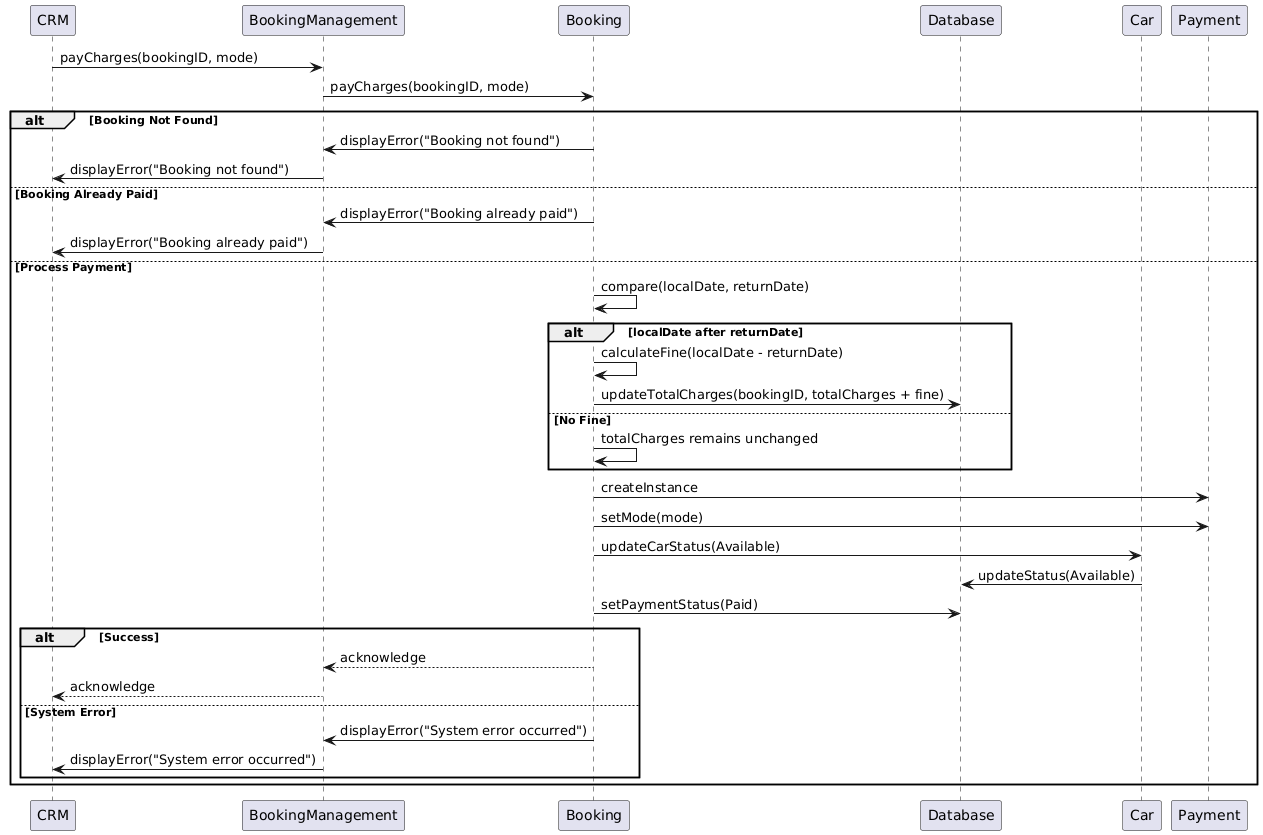
1. **Delete Driver:**



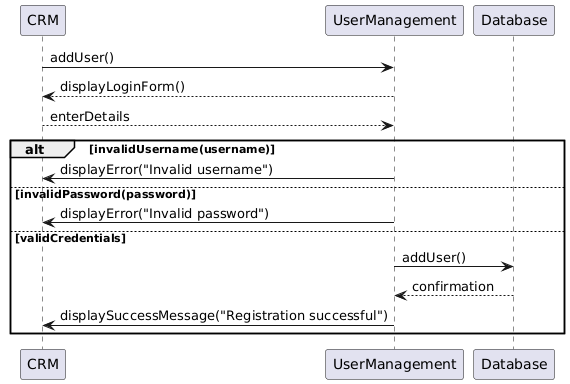
1. **Log In:**



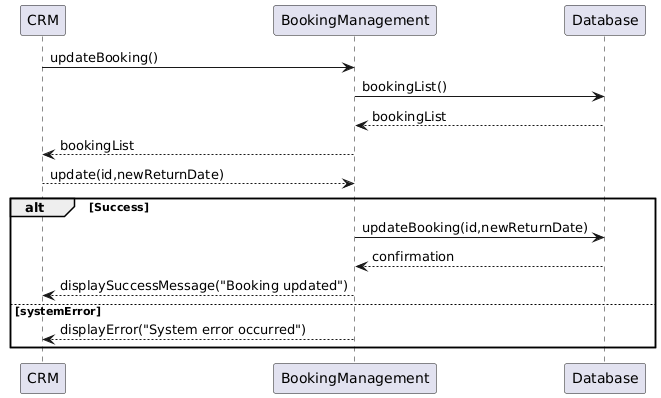
1. **Payment:**



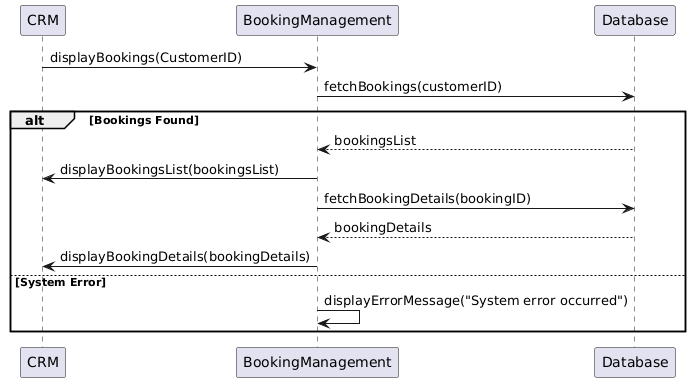
1. **Registration:**



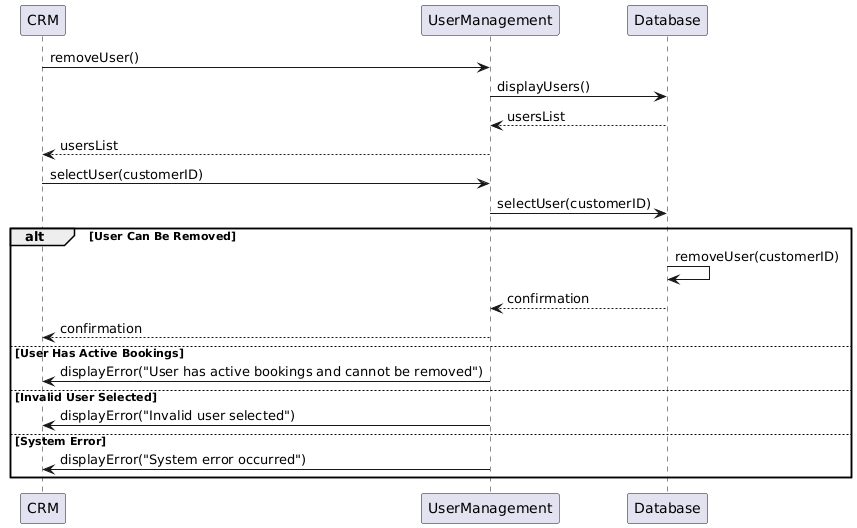
1. **Update Booking:**



1. **View Booking:**

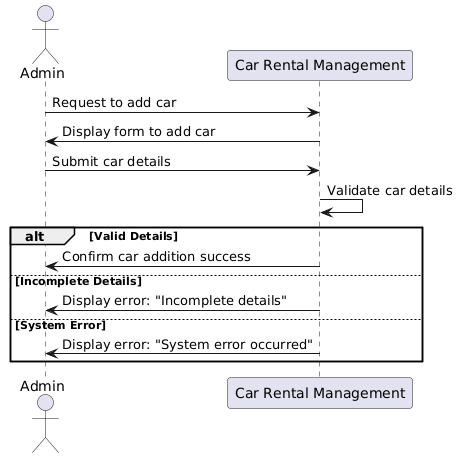


1. **Delete Customer:**

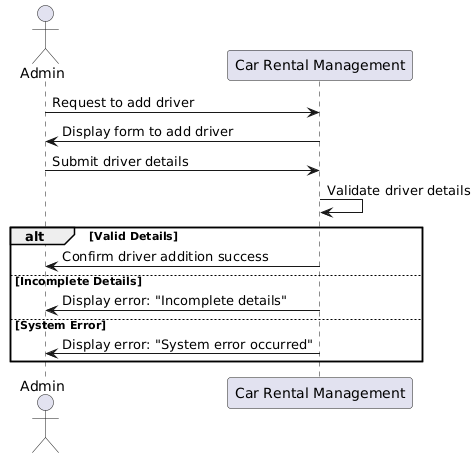


**System Sequence Diagrams(SSD):**

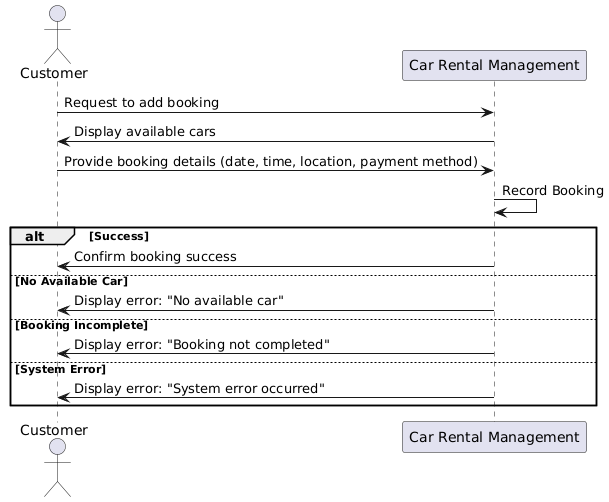
1. **Add Car:**



1. **Add Driver:**



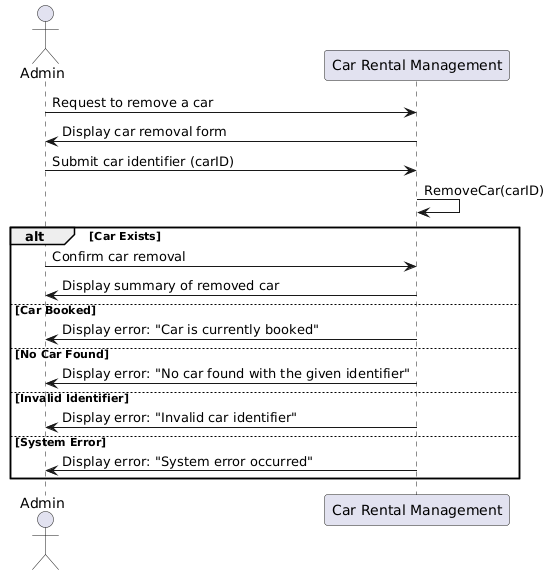
1. **Add Booking:**



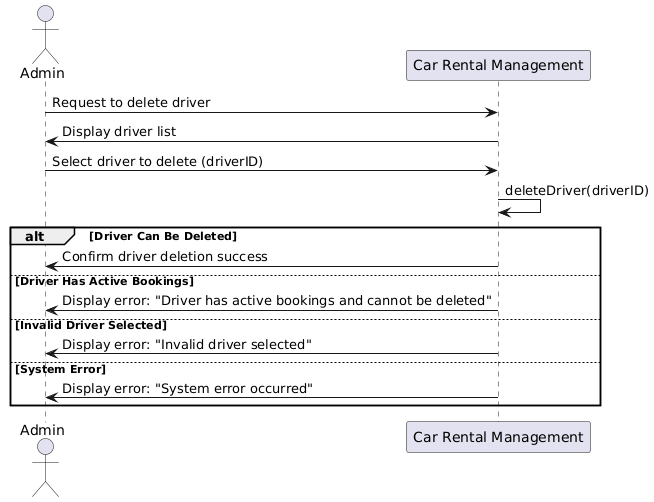
1. **Cancel Booking**



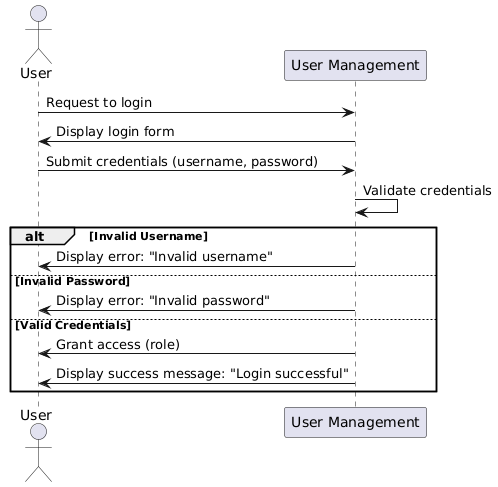
1. **Delete Car:**



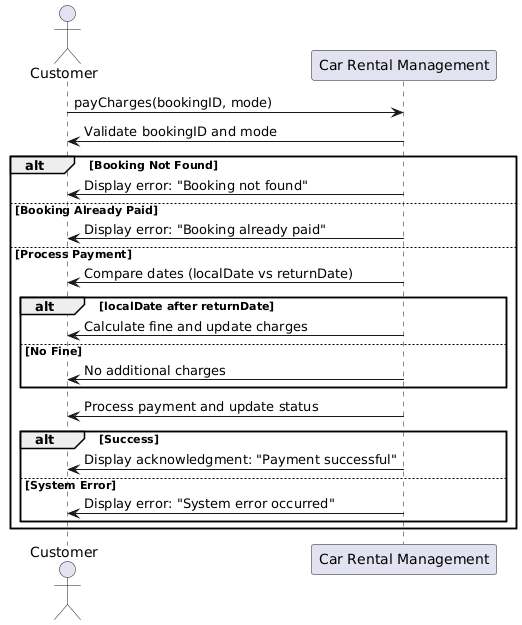
1. **Delete Driver:**



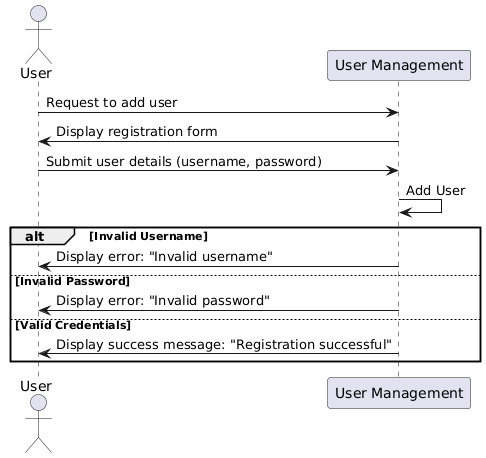
1. **Log In:**



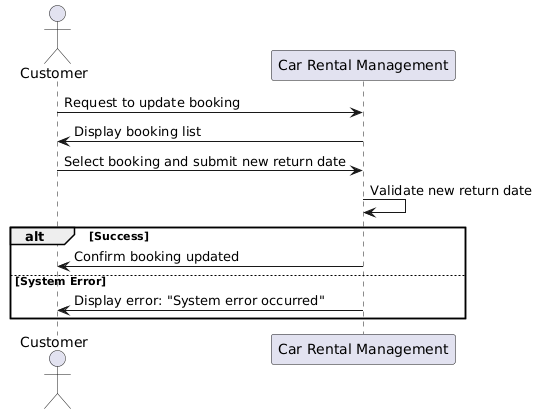
1. **Payment:**



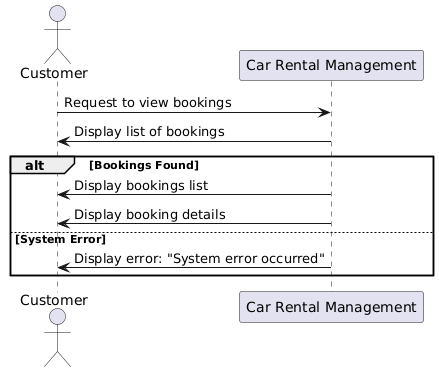
1. **Registration:**



1. **Update Booking:**



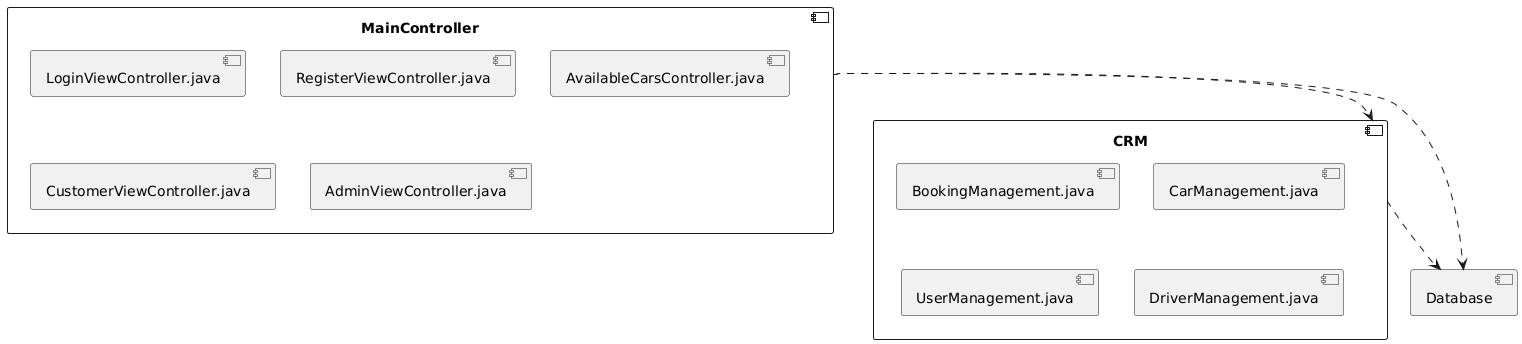
1. **View Booking:**



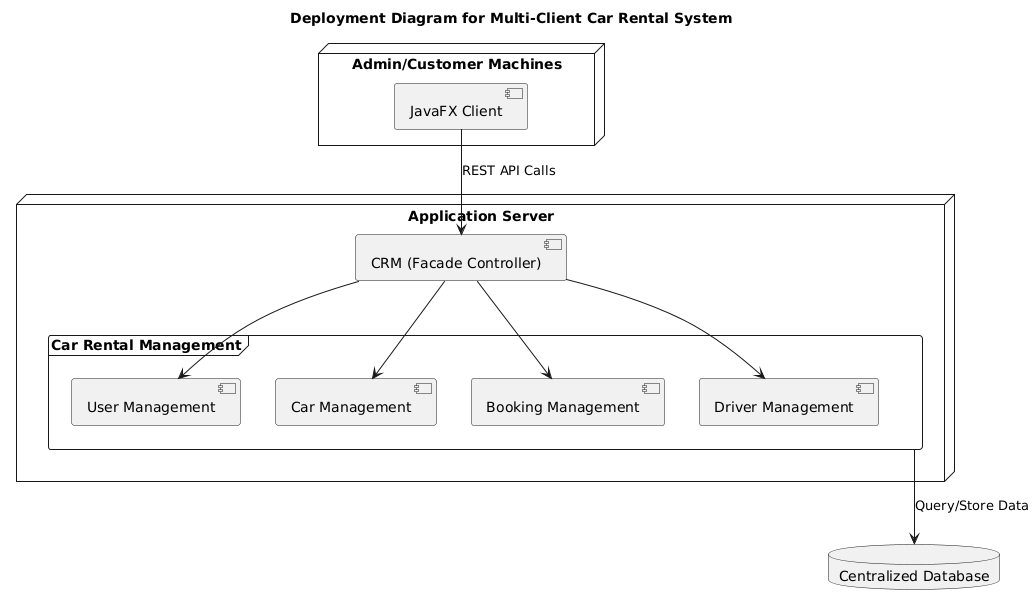
1. **Delete Customer:**



**Component Diagram:**



**Deployment Diagram:**



**Package Diagram:**

