

***Osama Hassan (cs-24120)***

***Abdul hadi (cs-24136)***

***Maslamah (cs-24150)***

OOP Complex Engineering Problem

Car rental management system

****

**#**

**1.Problem Description**

The Car Rental Management System is a Python-based object-oriented application developed to automate and manage the operations of a car rental service. The system enables:

***Customers*** to create accounts, log in securely, view available vehicles, rent a car based on their budget and preferences, make payments, return cars, and review rental history.

***Administrators*** to oversee the car inventory, add or remove cars from the system, monitor active rentals, and generate rental and customer reports.

The goal is to create a real-world simulation of a rental service using object-oriented programming techniques, ensuring a clean code structure, reusability, and maintainability. The application includes a user-friendly Streamlit interface, persistent data storage via JSON, and robust backend logic built with Python.

**2. Distinguishing Features & OOP Concepts Implemented**

**Key Features**

***Account Management:*** Users can sign up and manage profiles with personal and financial details.

***Role-Based Access:*** Admins and customers have separate functionalities and permissions.

***Car Availability Check:*** Only cars not currently rented out are shown as available.

***Rental System:*** Users can rent one car at a time, with start and end dates tracked.

***Secure Transactions:*** Users must have a sufficient balance for payment, deducted automatically.

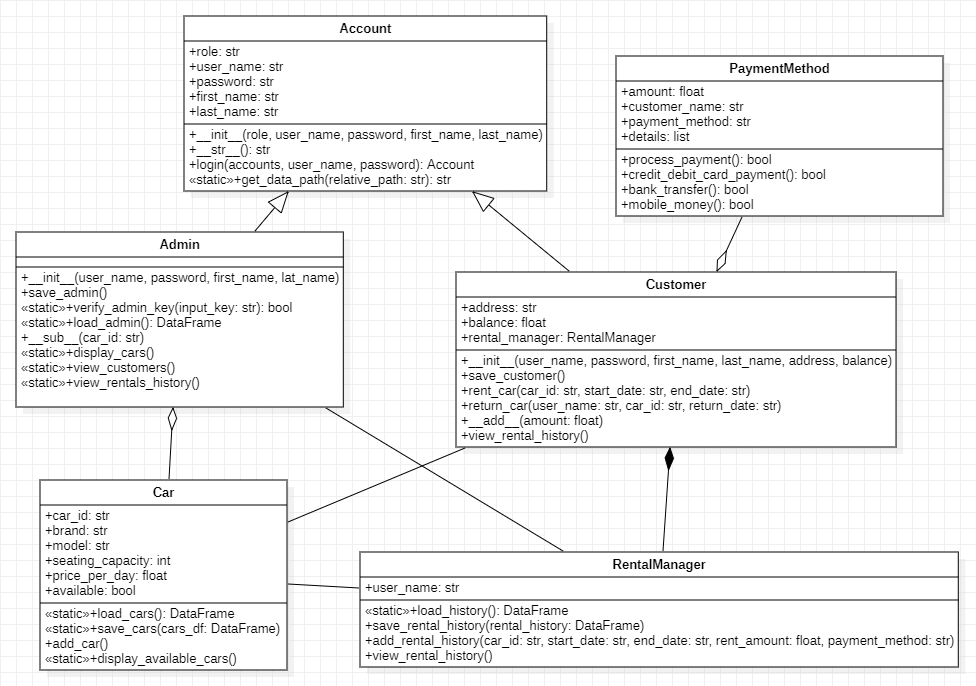
***Rental History:*** Tracks each customer's rental activities and transactions.

***Admin Tools:*** Admins can manage the fleet and generate comprehensive reports.

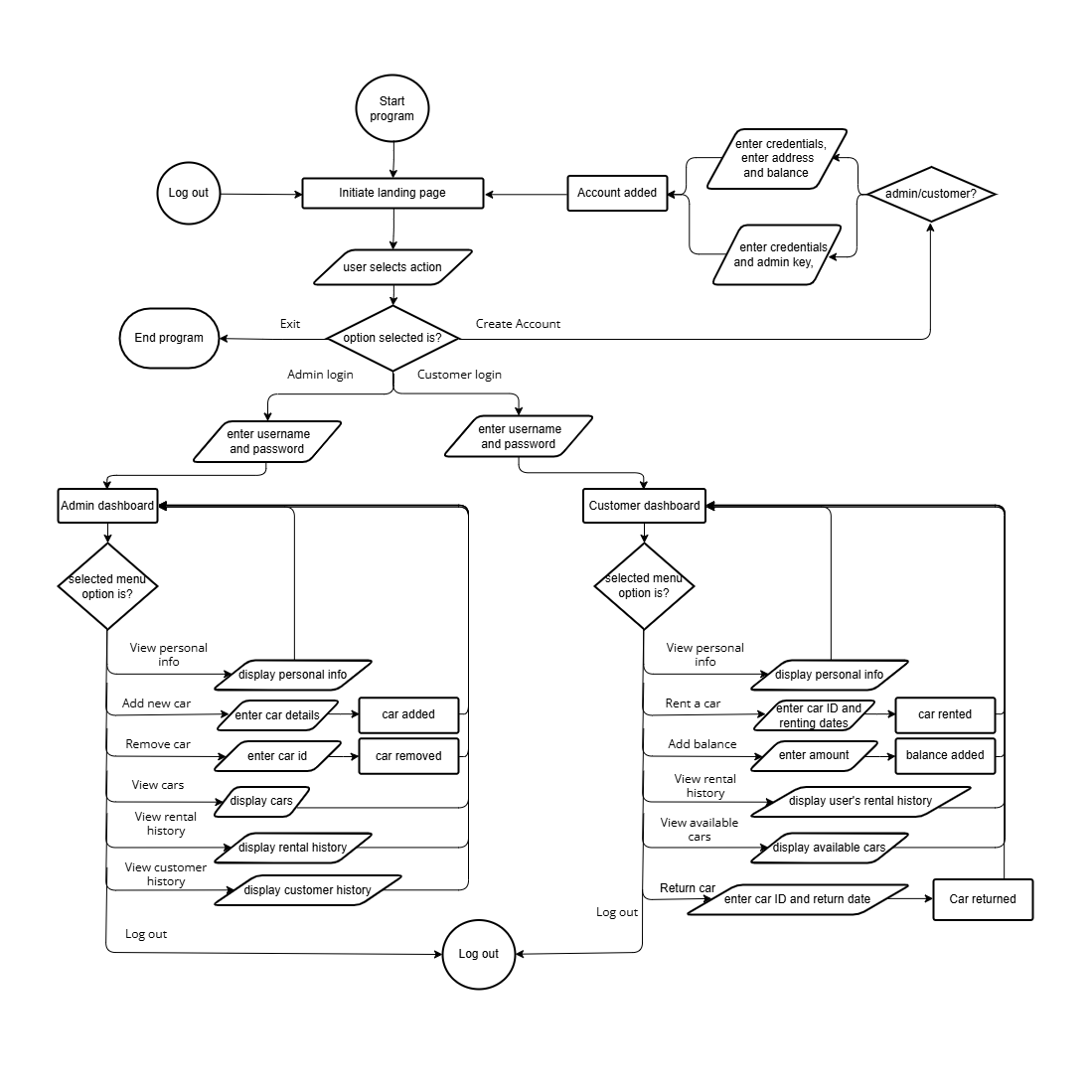
***Data Persistence:*** All operational data is saved in structured JSON files for ease of access and updates.

**OOP Concepts Implemented**

|  |  |
| --- | --- |
| **Object-Oriented Concept** | **Application** |
| *Inheritance* | *Admin and Customer inherit from the base Account class, sharing common methods such as login and profile handling.* |
| *Association*  *(Aggregation or composition)* | *RentalManager uses instances of Customer and Car classes to manage and track rentals.* |
| *Method Overriding* | *\_str\_() is overridden in Customer and Admin to customize object descriptions.* |
| *Operator Overloading* | *Implemented \_add\_ to top-up balance for a customer, and \_sub\_ to allow admin to remove cars.* |
| *Exception Handling* | *Handles file read/write errors, login failures, invalid payments, and car unavailability gracefully.* |

**Class Diagram**

***3. Project Flow & Class Diagram***

****System Workflow**

**4. Most Challenging Parts**

***Payment Logic & Synchronization:***

* Designing a payment system that simulates real-time deduction and checks for sufficient balance.
* Ensuring consistency in concurrent rentals and handling edge cases such as duplicate bookings.

***Maintaining Data Integrity in JSON:***

* Carefully managing read/write operations to avoid overwriting or data loss, especially during rapid state changes like car returns.

***Streamlit Interface Navigation:***

* Maintaining clean session state management between admin and customer views using st.session\_state.

**5. New Things Learned in Python**

***Streamlit Web Framework:*** Learned how to build interactive web applications with dynamic content using widgets, forms, and session states.

***Operator Overloading:*** Learned how to define custom behaviors for built-in operators to make object interactions intuitive.

***Data Persistence with JSON:*** Gained experience in using JSON as a lightweight database alternative.

***Error Handling:*** Improved understanding of Python exceptions and built robust error-handling mechanisms to ensure a smooth user experience.

**6. Individual Contributions**

|  |  |
| --- | --- |
| **Team Member** | **Responsibilities** |
| *Osama Hassan* | *Streamlit UI development, payment logic, session state management.* |
| *Abdul Hadi* | *OOP architecture, admin/customer dashboards, class design.* |
| *Maslamah* | *Rental tracking system, JSON file handling, exception handling, balance operations.* |

**7. Future Expansions**

***Database Integration:*** Upgrade to a relational database like SQLite or PostgreSQL for faster and safer data operations.

***Mobile Application***: Build a mobile version of the system using Flutter or React Native.

***Car Recommendations:*** Use machine learning to recommend cars based on user history and preferences.

**8. References**

***1. Streamlit Documentation***

<https://docs.streamlit.io>

– Official guide for building web apps in Python using Streamlit.

***2. Pandas Documentation***

<https://pandas.pydata.org/docs>

– Reference for data manipulation and reading/writing JSON.

***3. Real Python – Object-Oriented Programming in Python***

<https://realpython.com/python3-object-oriented-programming>

– Detailed explanation of OOP principles used in Python.

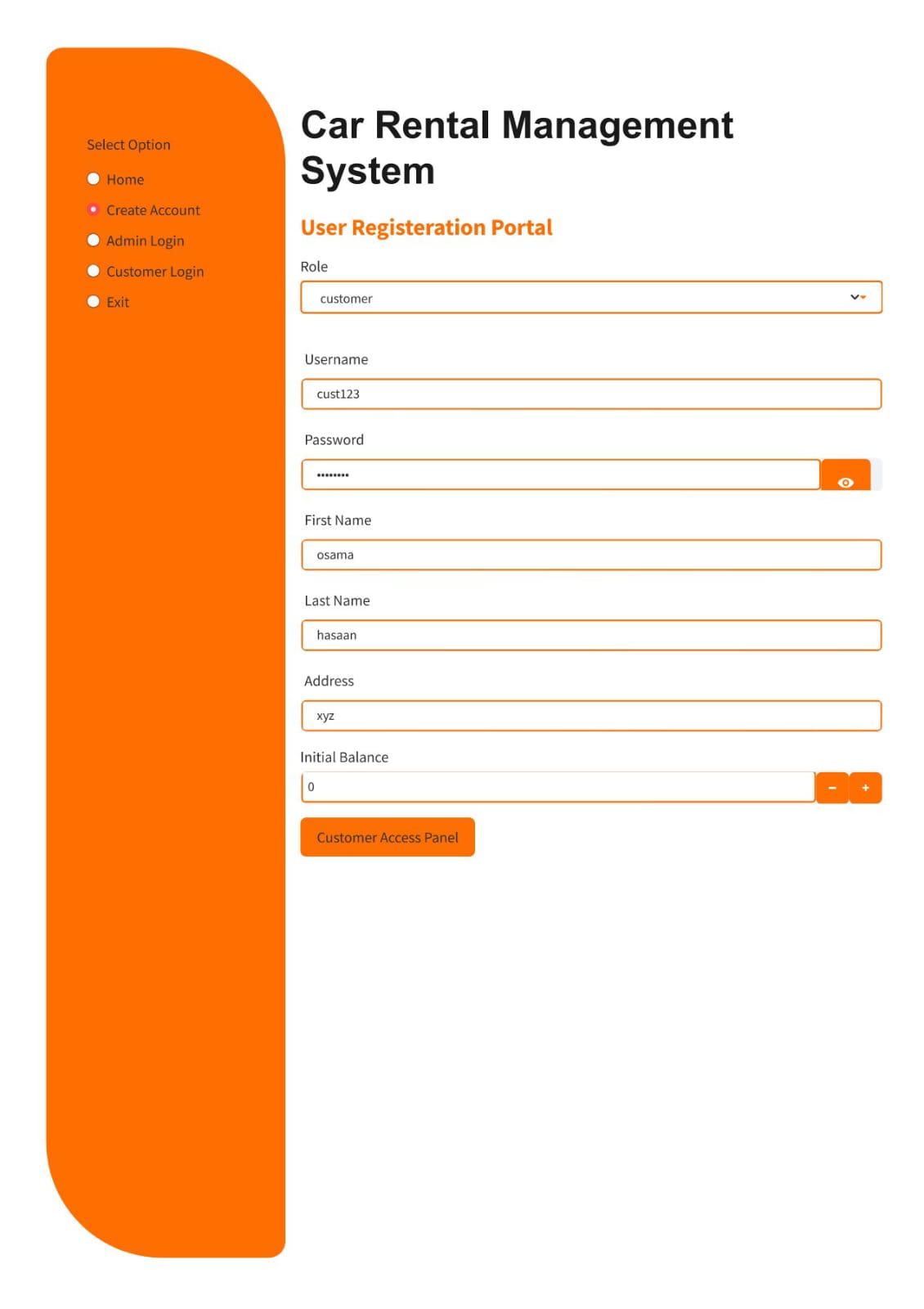
**Conclusion**

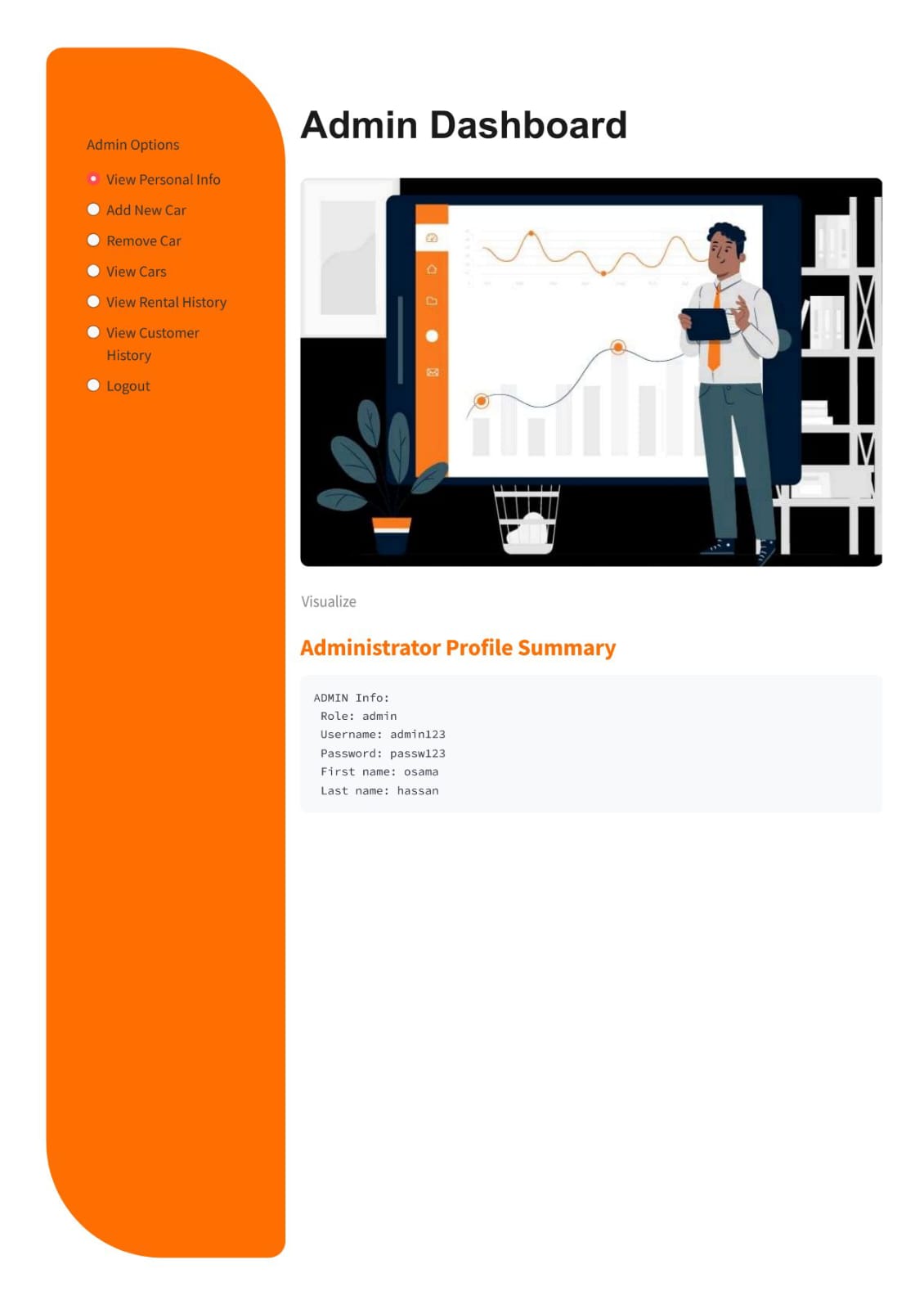
This project represents a comprehensive and real-world implementation of Object-Oriented Programming concepts in Python. From well-structured class design to user interactivity via Streamlit, the application demonstrates efficient coding practices and practical usage of design principles. The modular and extendable nature of the system makes it scalable and ready for integration with advanced features in the future.

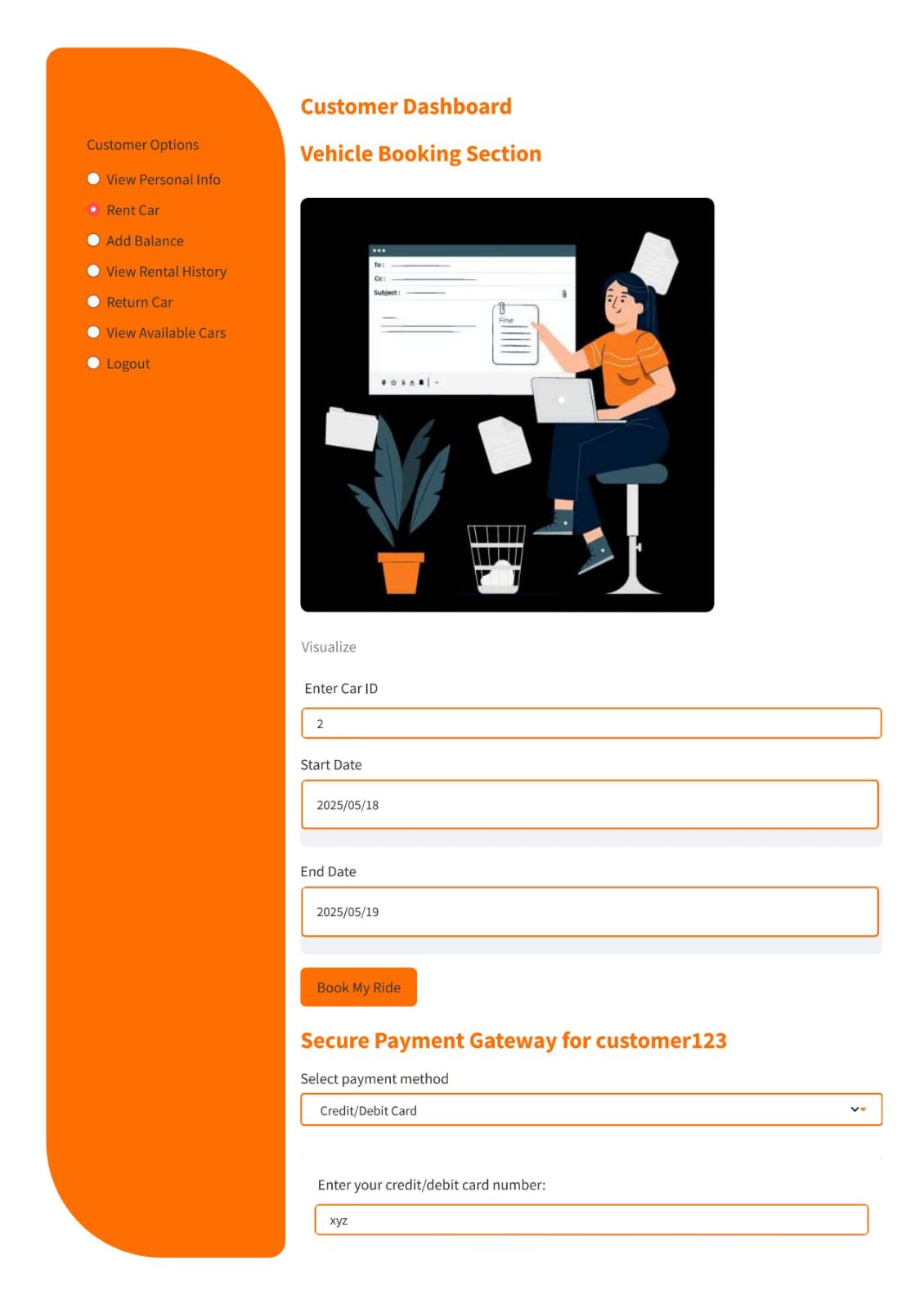
* ***GitHub Repository:*** <https://github.com/Osamahassan2005/Car-Rental-System>

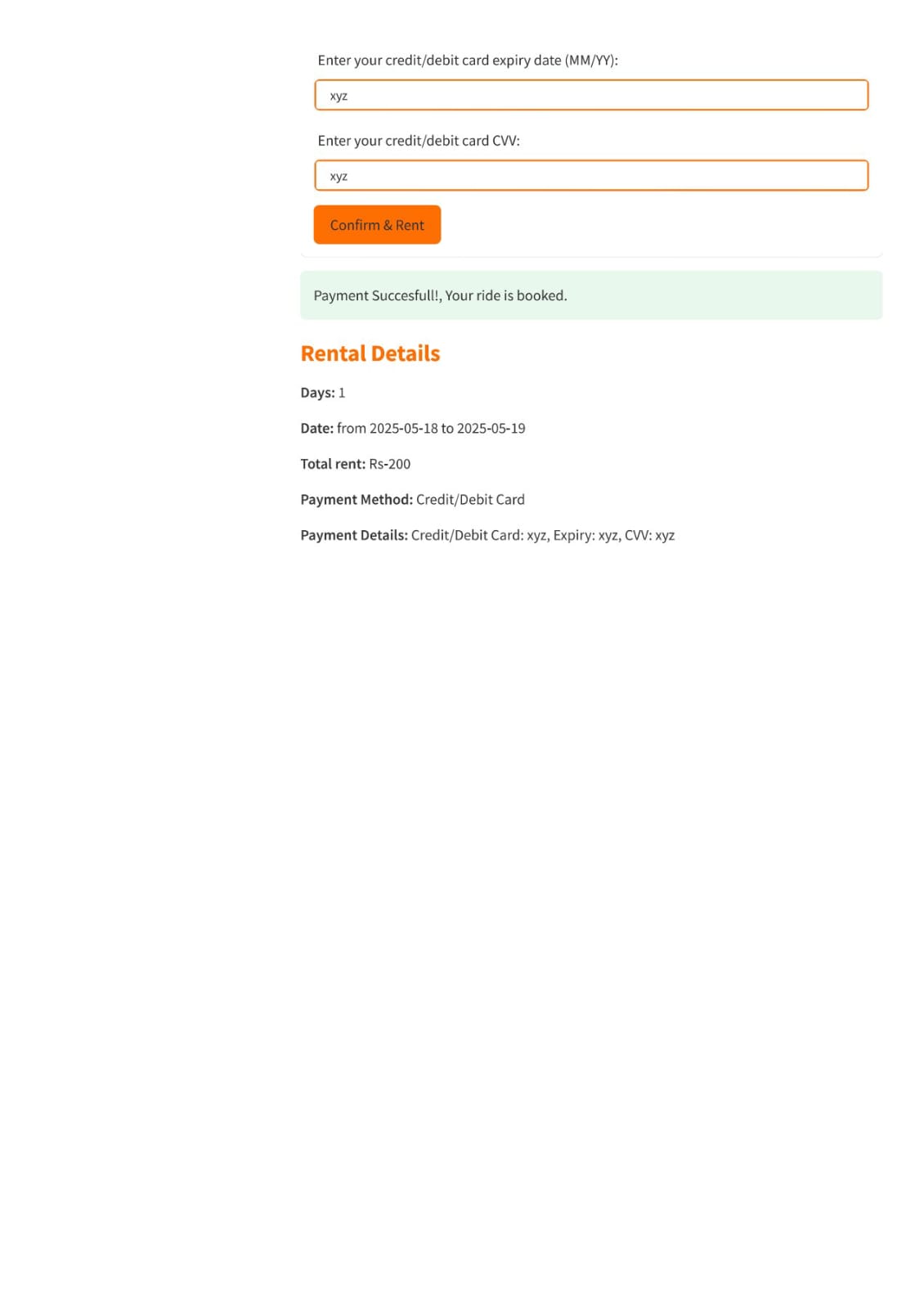
****Test Case Runs**

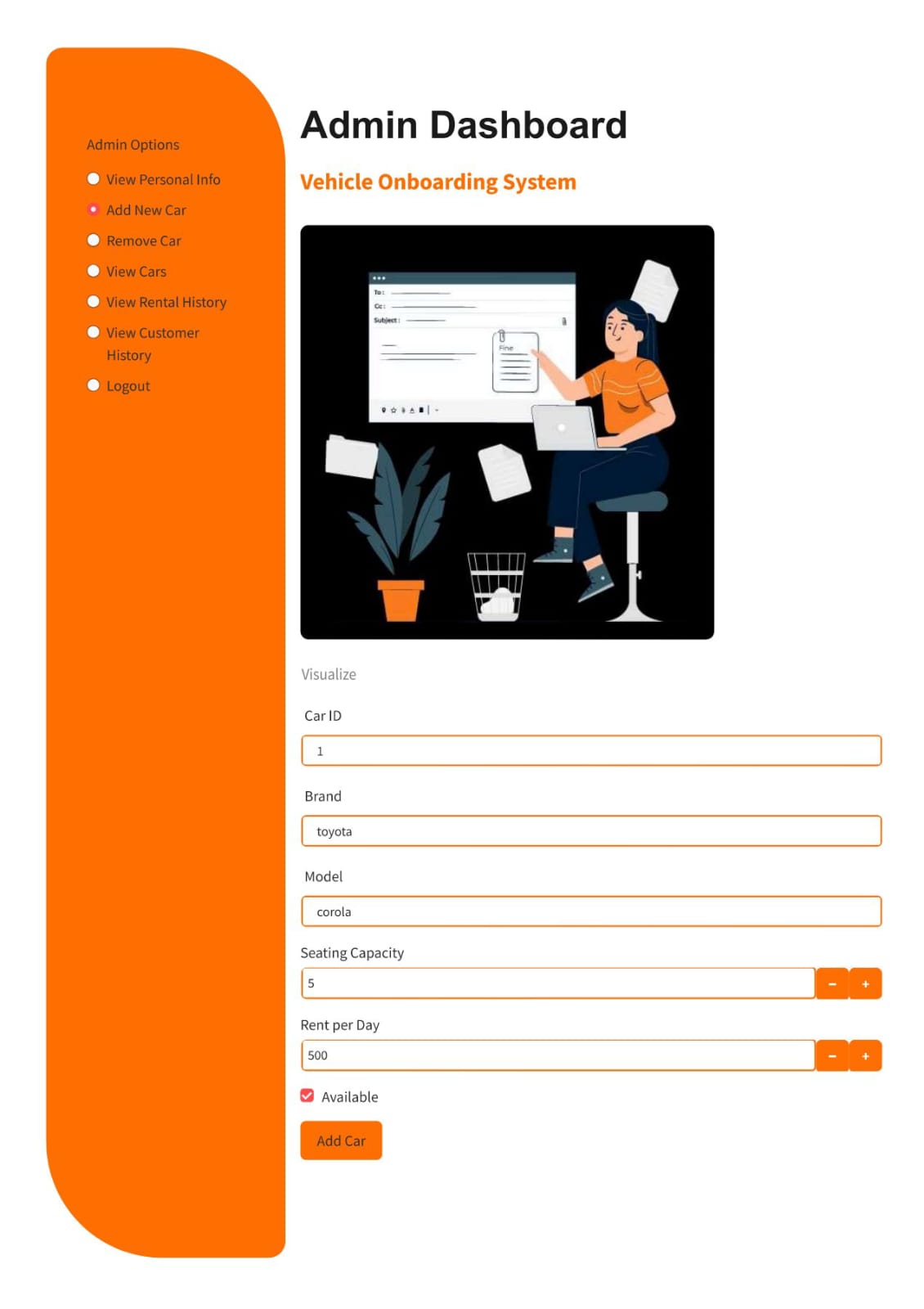
***Test case 1:*** Home Section

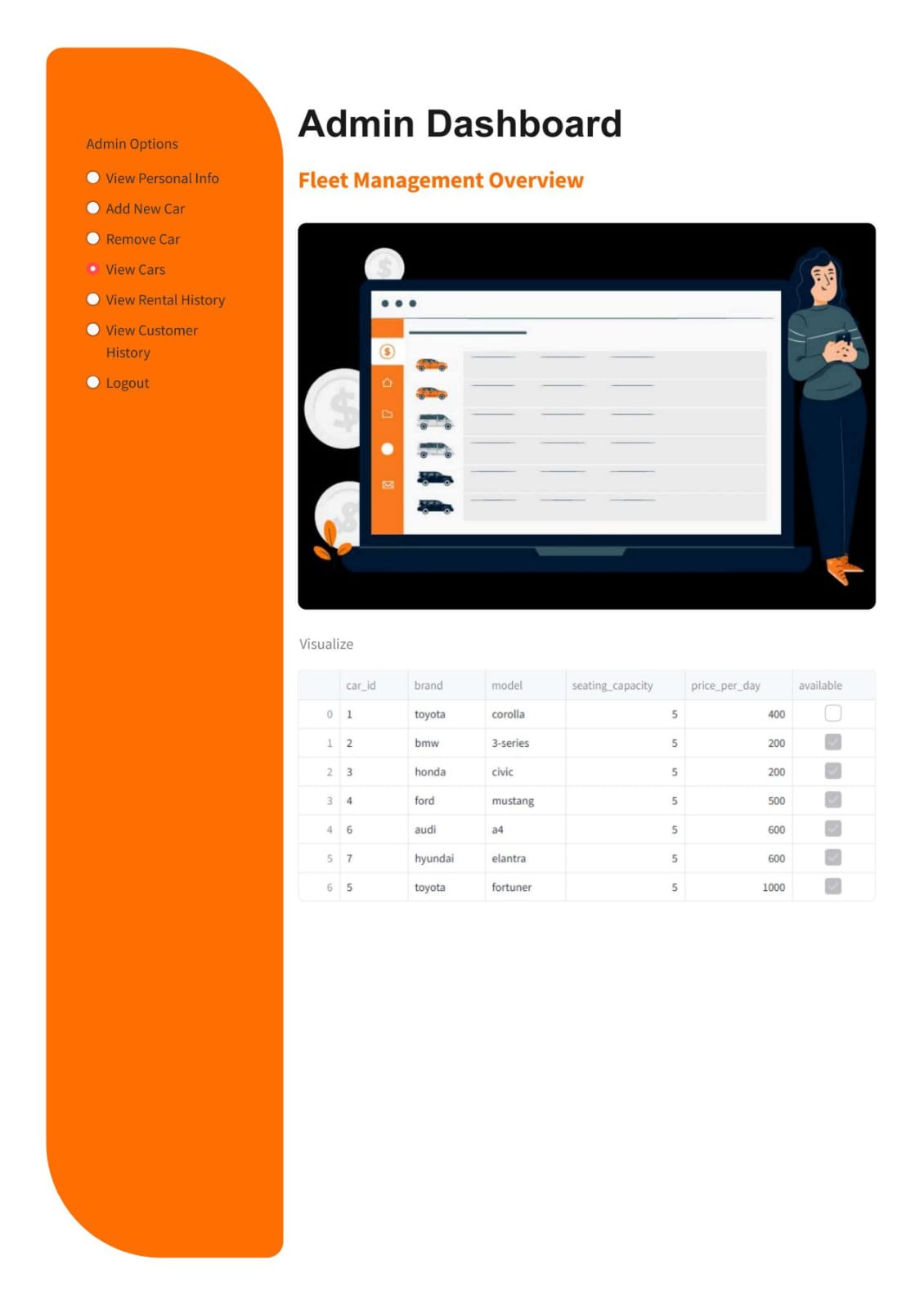
***Test case 2:*** Create Account Section

***Test case 3:*** Admin Dashboard

***Test case 4:*** Vehicle Booking Section



***Test case 5:*** Add new vehicle

***Test case 6:*** View vehicle