Complex Engineering Problem

CAR RENTAL MANAGEMENT SYSTEM

NED University Of Engineering

OSAMA HASSAN (CS-24120)
ABDUL HADI (CS-24136)
MASLAMAH (CS-24150)

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.*

3. Project Flow & Class Diagram

System Workflow

1. User Interaction:

- New customers can register by entering personal details.
- Registered users log in to access the customer dashboard.
- Admins log in using an additional secure admin key.

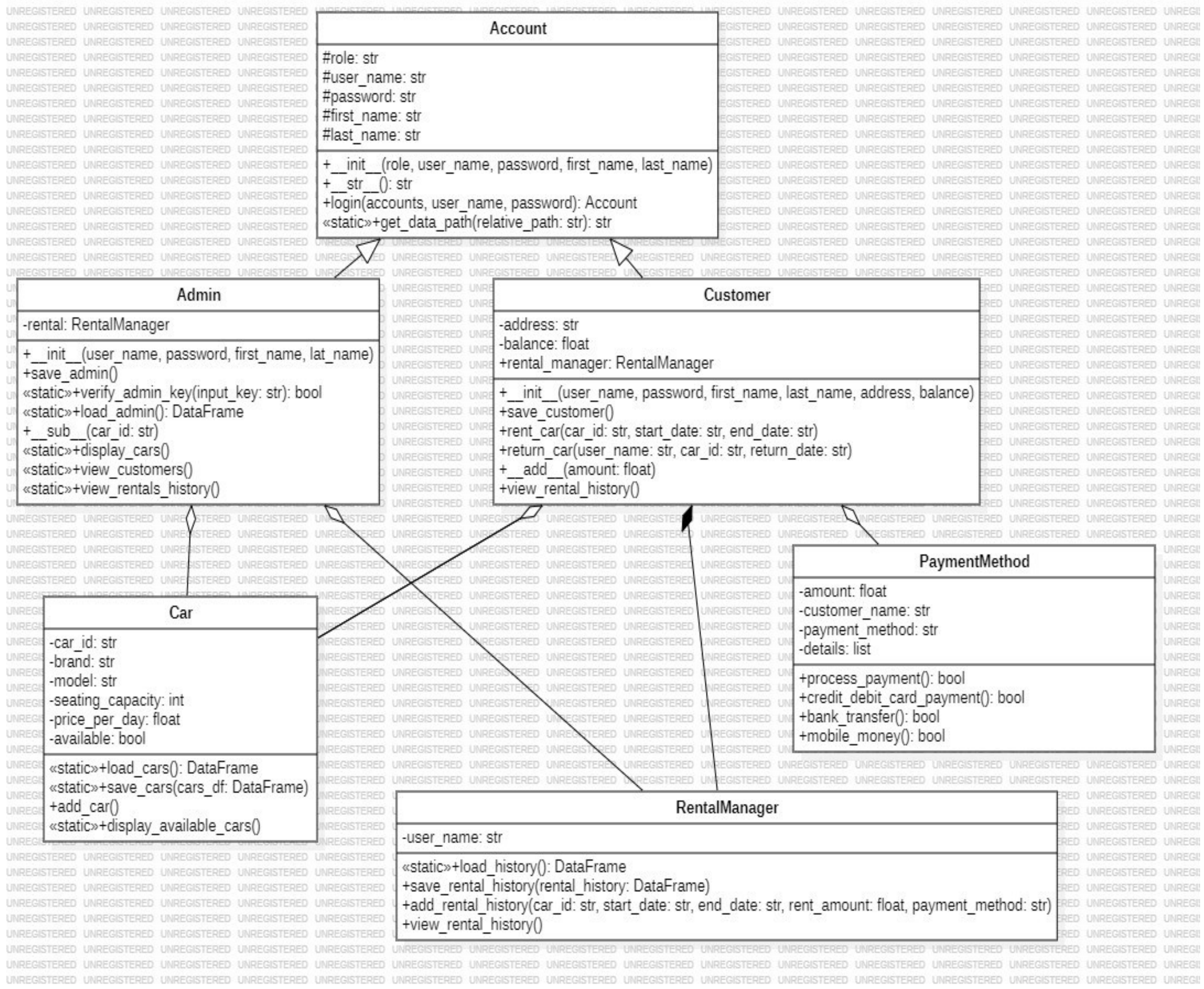
2. Customer Dashboard:

- View available cars (filter by brand, model, price, etc.).
- Rent a car and make payments directly from account balance.
- Return the car (record rental end date).
- View rental history and current rental status.
- Top-up balance using various payment options.

3. Admin Dashboard:

- Add new cars to the system with all relevant details.
- Remove existing cars by car ID or name.
- Generate and print reports on:
- ➢ Current rentals.
- ➢ All customers and their rental status.
- ➢ Reserved or returned cars.

Class Diagram


Account
#role: str
#user_name: str
#password: str
#first_name: str
#last_name: str
+__init__(role, user_name, password, first_name, last_name)
+__str__(): str
+login(accounts, user_name, password): Account
«static»+get_data_path(relative_path: str): str
Admin
-rental: RentalManager
+__init__(user_name, password, first_name, lat_name)
+save_admin()
«static»+verify_admin_key(input_key: str): bool
«static»+load_admin(): DataFrame
+__sub__(car_id: str)
«static»+display_cars()
«static»+view_customers()
«static»+view_rentals_history()
Customer
-address: str
-balance: float
+rental_manager: RentalManager
+__init__(user_name, password, first_name, last_name, address, balance)
+save_customer()
+rent_car(car_id: str, start_date: str, end_date: str)
+return_car(user_name: str, car_id: str, return_date: str)
+__add__(amount: float)
+view_rental_history()
PaymentMethod
-amount: float
-customer_name: str
-payment_method: str
-details: list
+process_payment(): bool
+credit_debit_card_payment(): bool
+bank_transfer(): bool
+mobile_money(): bool
Car
-car_id: str
-brand: str
-model: str
-seating_capacity: int
-price_per_day: float
-available: bool
«static»+load_cars(): DataFrame
«static»+save_cars(cars_df: DataFrame)
+add_car()
«static»+display_available_cars()
RentalManager
-user_name: str
«static»+load_history(): DataFrame
+save_rental_history(rental_history: DataFrame)
+add_rental_history(car_id: str, start_date: str, end_date: str, rent_amount: float, payment_method: str)
+view_rental_history()

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.

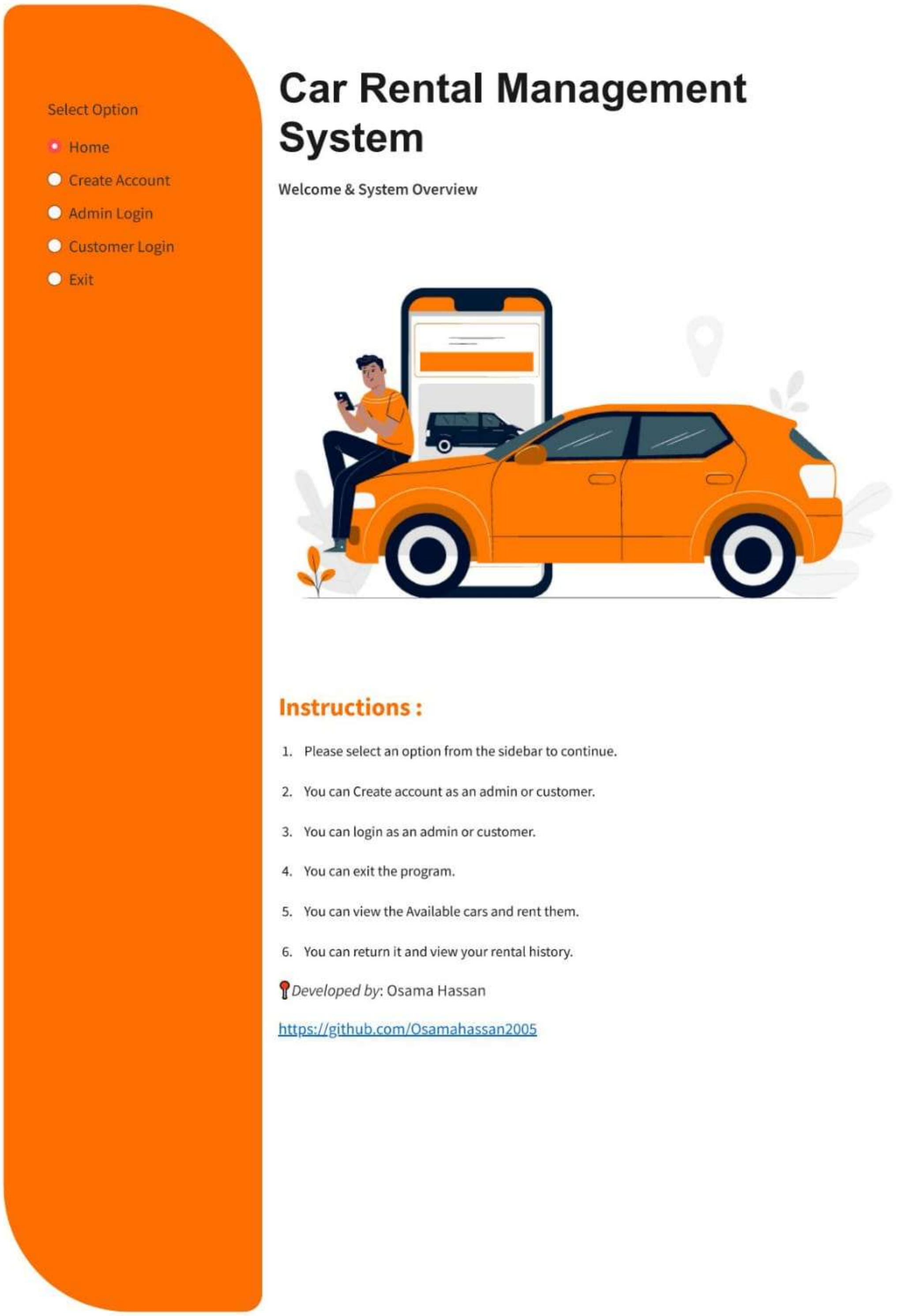
Conclusion

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



Car Rental Management System

Welcome & System Overview

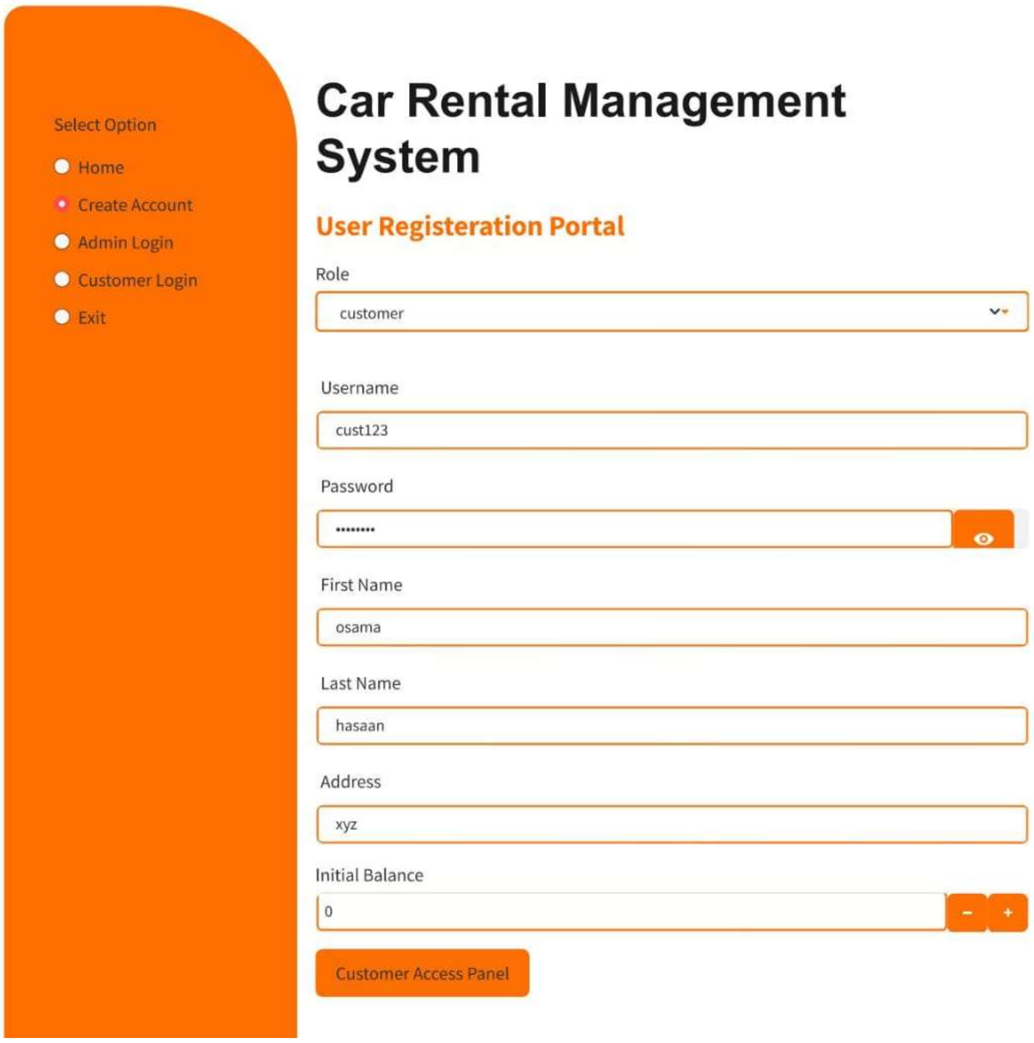
Instructions :

1. Please select an option from the sidebar to continue.
2. You can Create account as an admin or customer.
3. You can login as an admin or customer.
4. You can exit the program.
5. You can view the Available cars and rent them.
6. You can return it and view your rental history.

📍*Developed by*: Osama Hassan

https://github.com/Osamahassan2005

Test case 2: Create Account Section



Car Rental Management System

User Registeration Portal

Role: customer

Username: cust123

Password: ••••••••

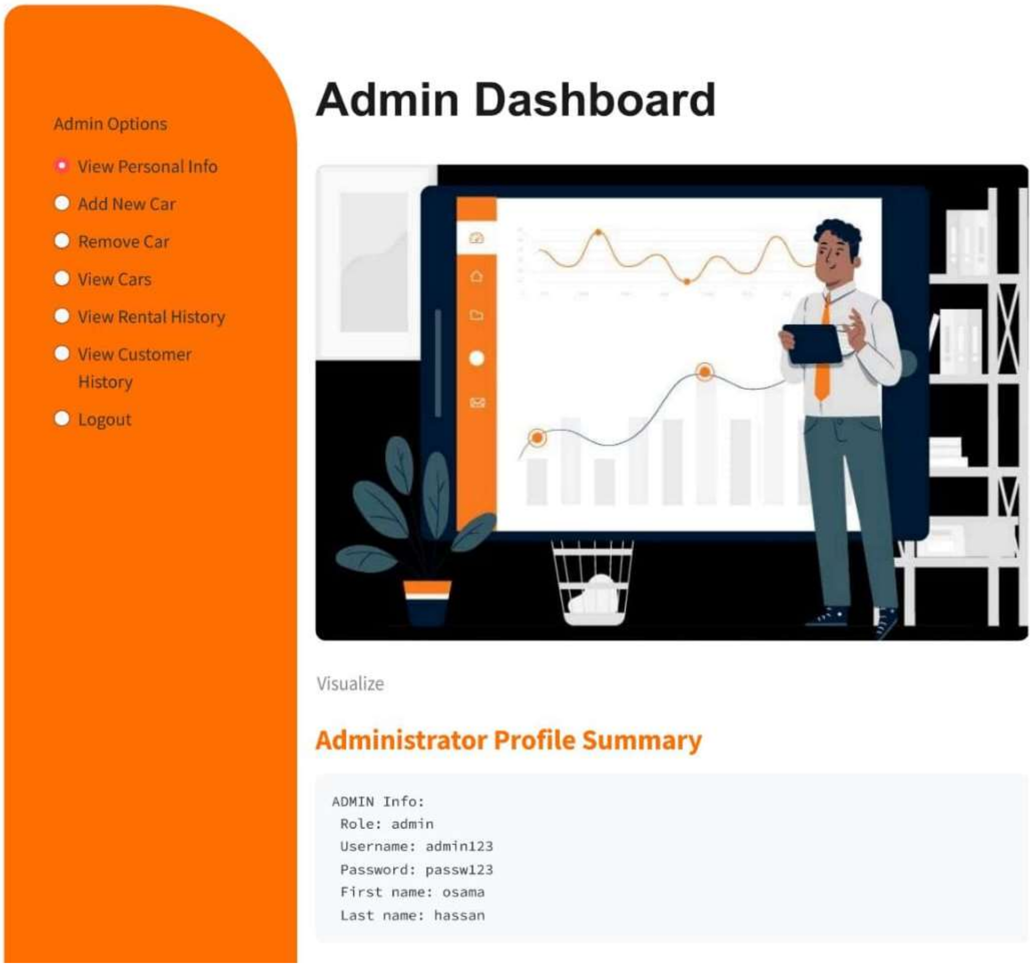
First Name: osama

Last Name: hasaan

Address: xyz

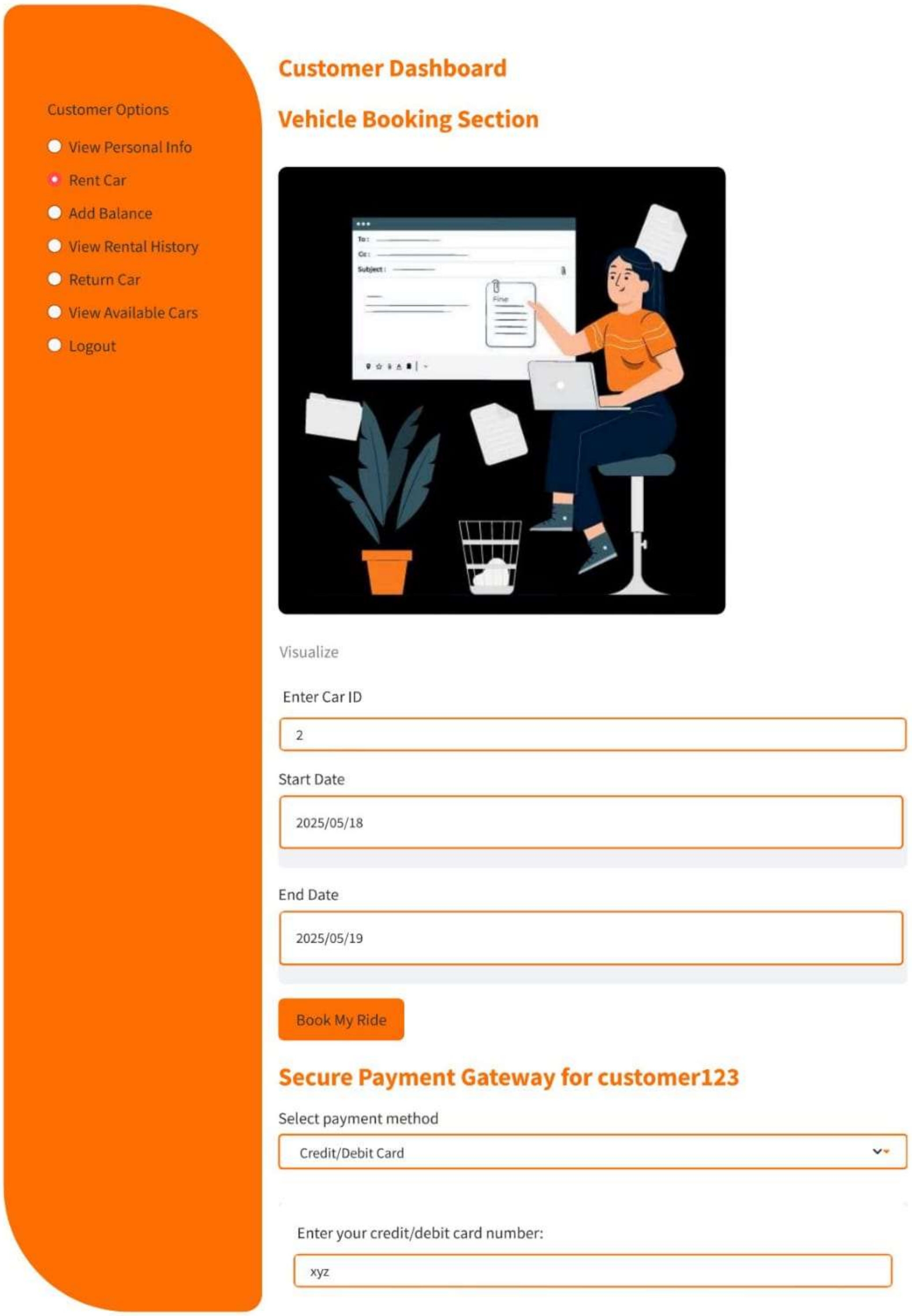
Initial Balance: 0

Test case 3: Admin Dashboard



```
ADMIN Info:
 Role: admin
 Username: admin123
 Password: passw123
 First name: osama
 Last name: hassan
```

Test case 4: Vehicle Booking Section



Customer Options

- View Personal Info
- [✓ Rent Car]
- Add Balance
- View Rental History
- Return Car
- View Available Cars
- Logout

Customer Dashboard

Vehicle Booking Section

Visualize

Enter Car ID

2

Start Date

2025/05/18

End Date

2025/05/19

Book My Ride

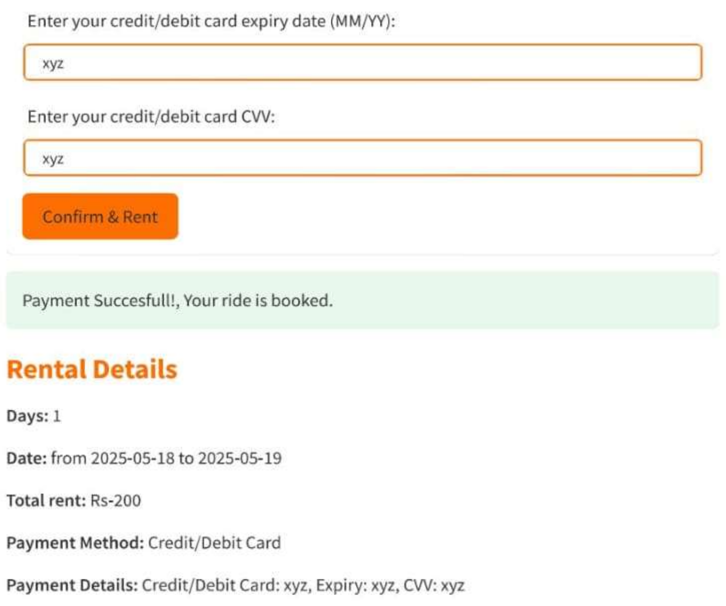
Secure Payment Gateway for customer123

Select payment method

Credit/Debit Card

Enter your credit/debit card number:

xyz


Enter your credit/debit card expiry date (MM/YY):

xyz

Enter your credit/debit card CVV:

xyz

Payment Succesfull!, Your ride is booked.

Rental Details

Days: 1

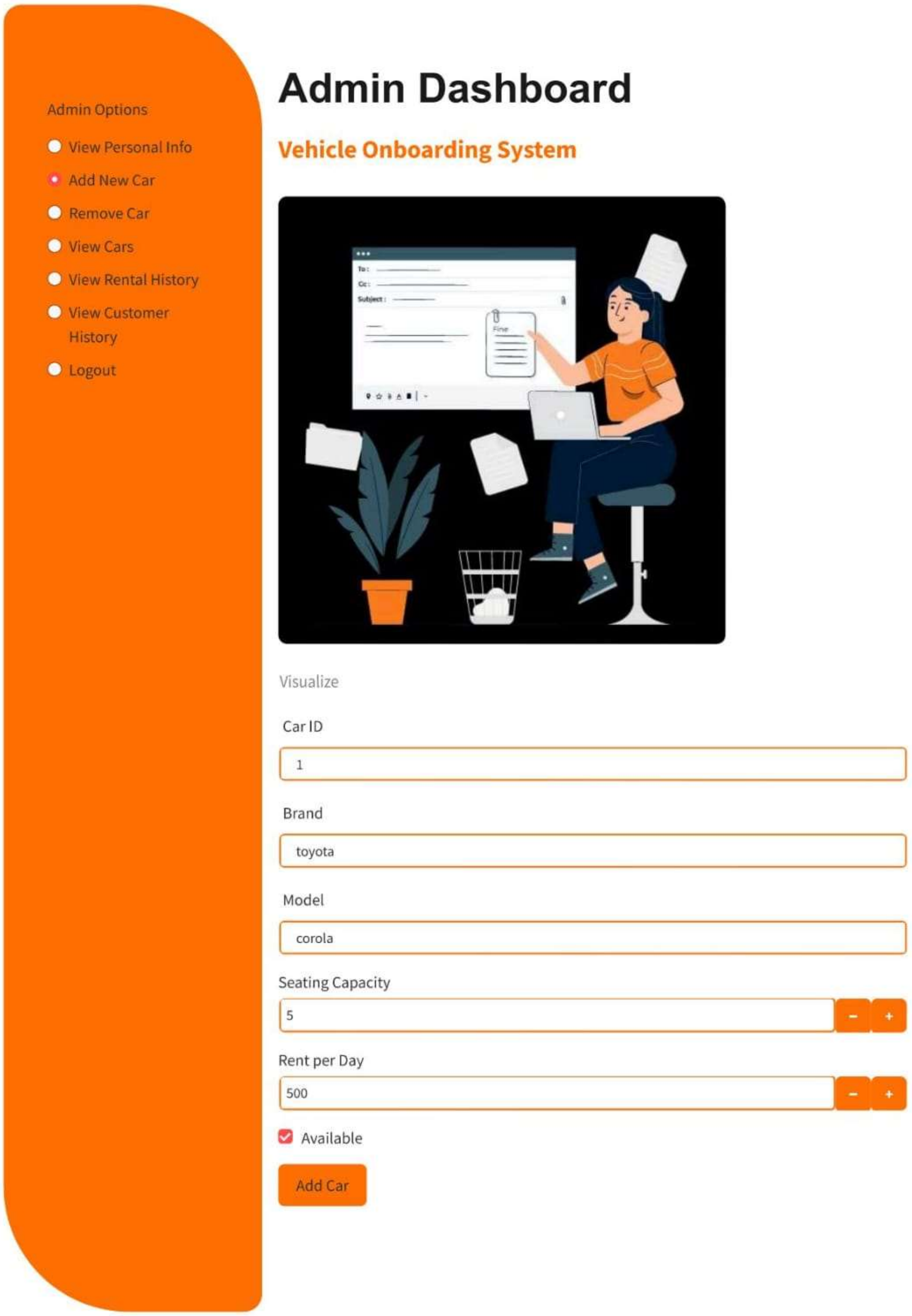
Date: from 2025-05-18 to 2025-05-19

Total rent: Rs-200

Payment Method: Credit/Debit Card

Payment Details: Credit/Debit Card: xyz, Expiry: xyz, CVV: xyz

Test case 5: Add new vehicle



Admin Dashboard

Vehicle Onboarding System

Visualize

Car ID: 1

Brand: toyota

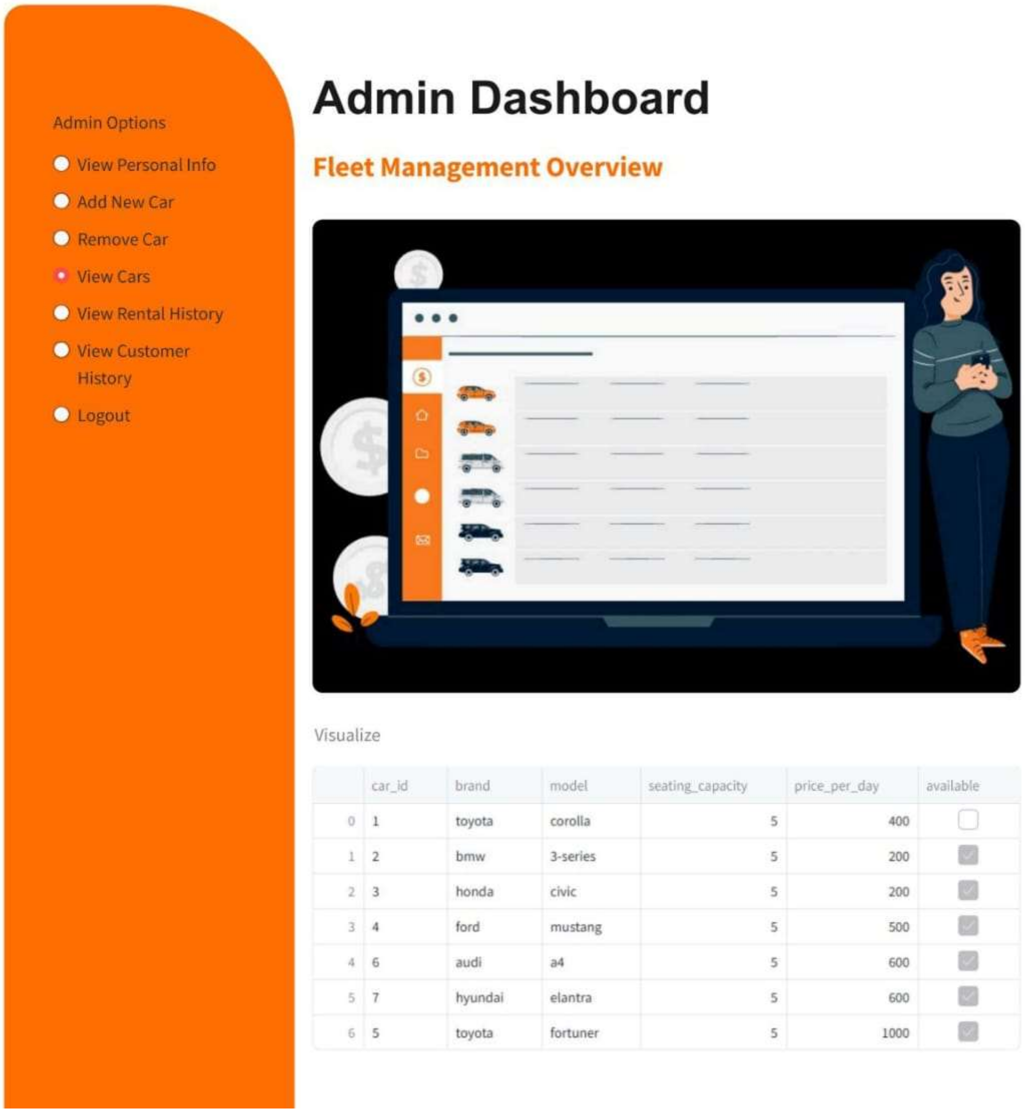
Model: corola

Seating Capacity: 5

Rent per Day: 500

[✓ Available]

Test case 6: View vehicle



Admin Options

- View Personal Info
- Add New Car
- Remove Car
- [✓ View Cars]
- View Rental History
- View Customer History
- Logout

Admin Dashboard

Fleet Management Overview

Visualize

	car_id	brand	model	seating_capacity	price_per_day	available
0	1	toyota	corolla	5	400	☐
1	2	bmw	3-series	5	200	✓
2	3	honda	civic	5	200	✓
3	4	ford	mustang	5	500	✓
4	6	audi	a4	5	600	✓
5	7	hyundai	elantra	5	600	✓
6	5	toyota	fortuner	5	1000	✓