

**Palestine Technical University – Kadoorie**

**College of Engineering and Technology**

**Department of Computer Systems Engineering**

**Course name:**

**Software Engineering**

**Project title:**

**DESIGNING A USER-FRIENDLY CAR RENTAL SERVICE WEBSITE FOR CONVENIENT VEHICLE BOOKING IN PALESTINE**

**By:**

**Afnan Abo-Asal – 202111997 - Section 1**

**Layla Al-Saabna – 202110985 - Section 4**

**Rahaf Alawneh – 202112169 - Section 1**

**Supervisors:**

**Dr. Nael Salman**

**Dr. Osama Hamed**

**Tulkarm, Palestine**

**Spring Semester 2023-2024**

# **Abstract**

The process of renting a vehicle in Palestine often encounters various difficulties. This project aims to introduce a user-friendly website designed to streamline the booking process, enabling users to easily rent a vehicle through their devices. By merely selecting the rental duration, pick-up date, and desired vehicle, customers can complete their bookings in just a few clicks. The anticipated outcome of this website's development and launch is a notable increase in user engagement, attributed to its simplification of the car rental process, thereby offering a significantly more convenient solution for users.

# **Table of Contents**

**Abstract2**

**Introduction4**

Problem Statement4

Objectives4

Target Group4

**Preliminary Literature Review 5**

**Methodology and Requirements 6**

Software Development Method6

Functional Requirements6

Non-Functional Requirements10

**Software Diagrams 13**

ER Diagram13

Use Case Diagram14

Class Diagram15

Sequence Diagram16

Log In/Register Activity Diagram 17

Reservation Process Activity Diagram 18

Renting Experience Rating Activity Diagram 19

**User Interfaces 20**

**Implementation 22**

**Team Work 25**

# **Chapter 1: Introduction**

* 1. **Problem Statement**

The process of renting a vehicle in Palestine often encounters various difficulties, starting from making phone calls, going to companies, search for the desired vehicle, and many more steps until you finally make an agreement.

The **'car-rentals.ps'** platform is a Software as a Service (SaaS) solution designed to streamline the vehicle rental process in Palestine. It connects customers with a wide range of vehicle providers, facilitated by a comprehensive relational database. This database meticulously manages details from lessors, vehicles, and rental transactions to customer ratings, all through a user-friendly interface.

Our service transitions the car rental process to a fully online environment, covering every region across Palestine, we offer a comprehensive and accessible car rental service. **'car-rentals.ps'** stands out for its attention to detail in listing vehicles and for the ease it brings to the rental process.

For those wishing to list their personal vehicles, our platform simplifies this process, requiring just the necessary details about the car. Renters can easily select a vehicle, set rental dates, and complete the booking online. By focusing on efficiency and convenience, **'car-rentals.ps'** is dedicated to enhancing the car rental experience for both car owners and renters.

* 1. **Objectives**

The main goal of the project is to transfer the car rental process from its traditional way into an online platform by designing a user-friendly car rental service website in Palestine. Other objectives can be summarized as follows:

1. To offer an around-the-clock online platform enabling users to effortlessly search for, compare, and book vehicles.
2. To implement advanced filtering options, allowing users to refine their searches based on specific criteria such as car type, rental price, and additional features, thus facilitating a tailored rental experience.
3. To integrate a seamless booking and payment system that ensures a secure and efficient transaction process, coupled with immediate confirmation and digital documentation for users.
4. To aspire the contribution of the broader development of digital services in Palestine.
   1. **Target Group**

* Citizens who wish to rent a vehicle.
* Car Rental companies in Palestine.
* Individuals who wish to rent out their own vehicles.

# **Chapter 2:** **Preliminary Literature Review**

The Palestinian platform, “<https://www.shobiddak.com/ar>", serves as a bridge connecting users with vehicle rental owners. However, the site was not exclusively designed for vehicle rentals, leading to the oversight of several critical features. Most notably, it lacks the option for users to specify the exact rental period. Additionally, it fails to display the available inventory of vehicles for rent, resulting in a diminished selection as users apply more filters. Furthermore, the process requires direct communication with the vehicle owner, often necessitating a phone call (provided the call is answered) and possibly a physical visit to inspect the vehicle before finalizing the agreement.

On a brighter note, there exists a more specialized website, “[https://car-rental-jerusalem.com](https://car-rental-jerusalem.com/)", which offers a dedicated car rental service. However, its operations are limited to Jerusalem city, also permitting the rental duration to be only three days or longer.

# **Chapter 3:** **Methodology and Requirements**

* 1. **Software Development Method**

Software model used: Agile Development Method

Why Agile Method?

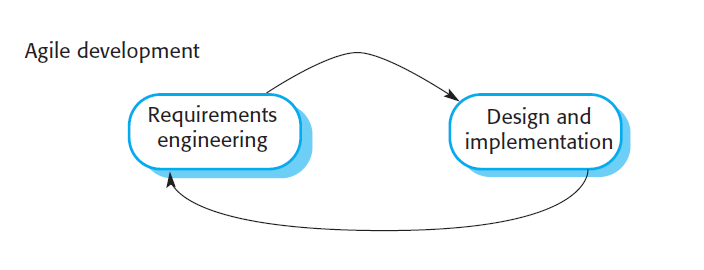
Agile methodologies provide a structured yet flexible framework that enhances project delivery, product quality, and customer satisfaction while fostering a collaborative and adaptive work environment.

Figure 1: Agile Development Method

* 1. **Requirements**
     1. **Functional Requirements**
* **User**
  + - * 1. Customer

**1. Customer Registration**

* **Description**: The system must allow a new customer to create an account.
* **Preconditions**: The customer is not already registered.
* **Postconditions**: The customer's information is stored in the system, and an account is created.
* **Steps**:
  1. The customer enters personal details (name, email, phone, address).
  2. The customer creates a username and password.
  3. The system validates the information.
  4. The system sends a confirmation email to the customer.
  5. The system stores the customer's information.

**2. Customer Login**

* **Description**: The system must allow a customer to log into their account.
* **Preconditions**: The customer is registered.
* **Postconditions**: The customer is authenticated and granted access to their account.
* **Steps**:
  1. The customer enters their username and password.
  2. The system verifies the credentials.
  3. The system grants access if the credentials are valid.

**3. Search Vehicles**

* **Description**: The system must allow customers to search for available vehicles.
* **Preconditions**: The customer is logged in.
* **Postconditions**: A list of available vehicles matching the search criteria is displayed.
* **Steps**:
  1. The customer enters search criteria (location, dates, vehicle type).
  2. The system retrieves available vehicles based on the criteria.
  3. The system displays the list of available vehicles.

**4. View Vehicle Details**

* **Description**: The system must allow customers to view detailed information about a specific vehicle.
* **Preconditions**: The customer has searched for vehicles.
* **Postconditions**: Detailed information of the selected vehicle is displayed.
* **Steps**:
  1. The customer selects a vehicle from the search results.
  2. The system retrieves details of the selected vehicle.
  3. The system displays the vehicle details (make, model, year, features, rate).

**5. Make a Reservation**

* **Description**: The system must allow customers to book a vehicle for rental.
* **Preconditions**: The customer has selected a vehicle.
* **Postconditions**: The reservation is created and confirmed.
* **Steps**:
  1. The customer enters rental details (start date, end date).
  2. The system checks availability for the selected period.
  3. The system creates the reservation.
  4. The system sends a confirmation email to the customer.

**6. Make Payment**

* **Description**: The system must allow customers to pay for their rental reservation.
* **Preconditions**: The reservation has been made.
* **Postconditions**: The payment is processed and confirmed.
* **Steps**:
  1. The customer enters payment details (credit card information).
  2. The system processes the payment through a payment gateway.
  3. The system updates the reservation status to 'Paid'.
  4. The system sends a payment confirmation email to the customer.

**7. View Reservation**

* **Description**: The system must allow customers to view their current and past reservations.
* **Preconditions**: The customer is logged in.
* **Postconditions**: A list of the customer's reservations is displayed.
* **Steps**:
  1. The customer navigates to the reservations section.
  2. The system retrieves the customer's reservations.
  3. The system displays the list of reservations with details.

**8. Cancel Reservation**

* **Description**: The system must allow customers to cancel a reservation.
* **Preconditions**: The reservation exists and is within a cancellable period.
* **Postconditions**: The reservation is cancelled.
* **Steps**:
  1. The customer selects a reservation to cancel.
  2. The system checks if the reservation can be cancelled.
  3. The system updates the reservation status to 'Cancelled'.
  4. The system sends a cancellation confirmation email to the customer.

**9. Rate Renting Experience**

* **Description**: The system must allow customers to rate their renting experience.
* **Preconditions**: The rental period is complete.
* **Postconditions**: The rating and comments are stored in the system.
* **Steps**:
  1. The customer selects a completed reservation.
  2. The customer enters a rating and comments.
  3. The system stores the rating and comments.
     + - 1. Lessor

**1. Lessor (Admin) Login**

* **Description**: The system must allow a lessor (admin) to log into the admin interface.
* **Preconditions**: The lessor has an admin account.
* **Postconditions**: The lessor is authenticated and granted access to the admin interface.
* **Steps**:
  1. The lessor enters their admin credentials.
  2. The system verifies the credentials.
  3. The system grants access to the admin interface.

**2. Add Vehicle**

* **Description**: The system must allow the lessor to add new vehicles to the system.
* **Preconditions**: The lessor is logged in.
* **Postconditions**: The vehicle details are added to the system.
* **Steps**:
  1. The lessor enters vehicle details (make, model, year, features, rate).
  2. The system validates the information.
  3. The system adds the vehicle to the list of available vehicles.

**3. Remove Vehicle**

* **Description**: The system must allow the lessor to remove vehicles from the system.
* **Preconditions**: The lessor is logged in.
* **Postconditions**: The vehicle is removed from the list of available vehicles.
* **Steps**:
  1. The lessor selects a vehicle to remove.
  2. The system checks if the vehicle can be removed.
  3. The system removes the vehicle from the list.

**4. Update Vehicle Details**

* **Description**: The system must allow the lessor to update details of existing vehicles.
* **Preconditions**: The lessor is logged in.
* **Postconditions**: The vehicle details are updated in the system.
* **Steps**:
  1. The lessor selects a vehicle to update.
  2. The lessor updates the vehicle details.
  3. The system validates the updated information.
  4. The system updates the vehicle details.

**5. View All Reservations**

* **Description**: The system must allow the lessor to view all customer reservations.
* **Preconditions**: The lessor is logged in.
* **Postconditions**: A list of all reservations is displayed.
* **Steps**:
  1. The lessor navigates to the reservations section.
  2. The system retrieves all reservations.
  3. The system displays the list of reservations with details.

**6. Generate Reports**

* **Description**: The system must allow the lessor to generate various reports (e.g., sales, rentals).
* **Preconditions**: The lessor is logged in.
* **Postconditions**: The requested report is generated and displayed.
* **Steps**:
  1. The lessor selects the type of report to generate.
  2. The system gathers the necessary data.
  3. The system generates the report.
  4. The system displays the report.
* **System**

**1. Send Confirmation Emails**

* **Description**: The system must send confirmation emails to customers after certain actions.
* **Triggers**: Actions such as registration, reservation, payment, and cancellation.
* **Steps**:
  1. The system sends an email after a customer registers.
  2. The system sends an email after a reservation is made.
  3. The system sends an email after a payment is made.
  4. The system sends an email after a reservation is cancelled.

**2. Process Payments**

* **Description**: The system must handle the processing of payments through an external payment gateway.
* **Triggers**: When a customer makes a payment.
* **Steps**:
  1. The system processes the payment details.
  2. The system communicates with the external payment gateway.
  3. The system updates the payment status based on the gateway response.

**3. Verify Password**

* **Description**: The system must verify the customer's password during the login and registration processes.
* **Preconditions**: The customer is entering a password for login or registration.
* **Postconditions**: The password is verified according to the system's security policies.
* **Steps**:
  1. The customer enters a password during login or registration.
  2. The system checks the password against the stored hashed password (for login).
  3. The system ensures the password meets complexity requirements (for registration).
  4. The system grants access or allows account creation if the password is valid.
  5. If the password is invalid, the system displays an error message (see Display Error Messages).

**4. Display Error Messages**

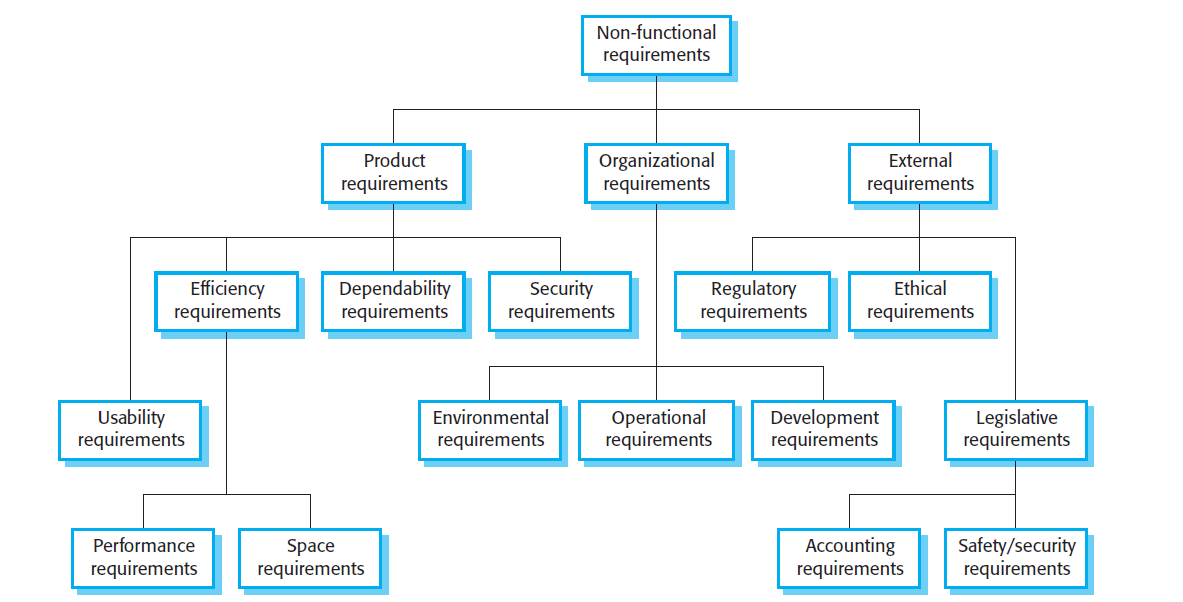
* **Description**: The system must display appropriate error messages to customers and lessors when an action cannot be completed.
* **Preconditions**: An error condition occurs (e.g., incorrect login credentials, invalid input).
* **Postconditions**: The customer or lessor is informed about the error and the required corrective action.
* **Steps**:
  1. The customer or lessor performs an action.
  2. The system detects an error condition (e.g., invalid input, system issue).
  3. The system generates a relevant error message.
  4. The system displays the error message to the user with details on how to correct it.
     1. **Non-Functional Requirements**

Figure 2: Types of non-Functional Requirements

**1. Performance**

* **Description**: The system must perform efficiently to handle multiple users and transactions simultaneously.
* **Details**:
  + The system should support at least 100 concurrent users without performance degradation.
  + Page load time should be less than 2 seconds for 95% of user interactions.
  + The system should process reservation and payment transactions within 3 seconds.

**2. Scalability**

* **Description**: The system must be scalable to accommodate growth in the number of users and transactions.
* **Details**:
  + The system architecture should support horizontal scaling to add more servers as needed.
  + Database design should support partitioning and replication to handle increased data volume.

**3. Reliability**

* **Description**: The system must be reliable and available to users at all times.
* **Details**:
  + The system should have an uptime of 99.9% (no more than 8.76 hours of downtime per year).
  + The system should have mechanisms for automatic failover in case of server failure.
  + Regular data backups should be performed to prevent data loss.

**4. Security**

* **Description**: The system must ensure the security of user data and transactions.
* **Details**:
  + All user data should be encrypted in transit (using HTTPS) and at rest.
  + The system should implement strong password policies and use multi-factor authentication (MFA) for admin access.
  + The system should undergo regular security audits and vulnerability assessments.

**5. Usability**

* **Description**: The system must be user-friendly and easy to navigate.
* **Details**:
  + The user interface should be intuitive, with clear navigation and instructions.
  + The system should provide help and support features, such as FAQs and a user manual.
  + The system should support accessibility standards (e.g., WCAG) to ensure it is usable by people with disabilities.

**6. Maintainability**

* **Description**: The system must be maintainable and easy to update.
* **Details**:
  + The system should have modular code architecture to allow for easy updates and bug fixes.
  + Documentation should be provided for all system components to assist developers in maintenance tasks.
  + The system should support automated testing to ensure updates do not introduce new bugs.

**7. Compatibility**

* **Description**: The system must be compatible with various devices and browsers.
* **Details**:
  + The system should work on all major browsers (e.g., Chrome, Firefox, Safari, Edge).
  + The system should be responsive and function correctly on various devices, including desktops, tablets, and smartphones.

**8. Data Integrity**

* **Description**: The system must ensure the integrity and consistency of data.
* **Details**:
  + The system should implement transaction management to ensure data consistency.
  + Data validation checks should be performed to prevent incorrect data entry.
  + Regular audits should be conducted to ensure data accuracy and consistency.

**9. Availability**

* **Description**: The system must be available to users at all times with minimal downtime.
* **Details**:
  + Scheduled maintenance should be performed during off-peak hours and communicated to users in advance.
  + The system should have redundancy in place to ensure high availability.

**10. Portability**

* **Description**: The system should be portable and easily deployable in different environments.
* **Details**:
  + The system should be deployable on various cloud platforms (e.g., AWS, Azure, Google Cloud).
  + The system should support containerization (e.g., Docker) for easy deployment and scaling.

These non-functional requirements ensure that the car-rentals.ps system will not only function correctly but also perform efficiently, securely, and reliably while providing a good user experience.

# **Chapter 4: Software Diagrams**

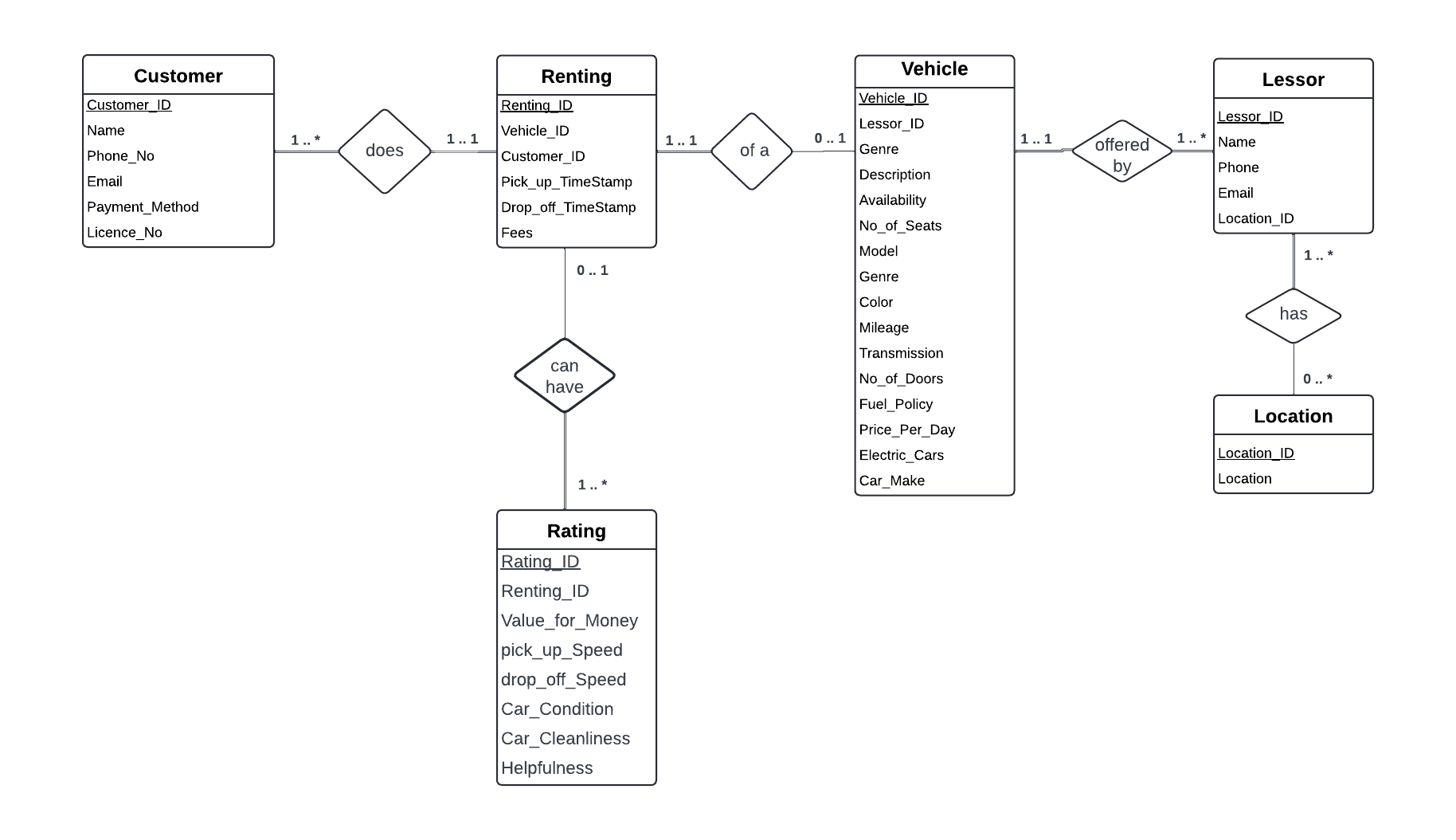
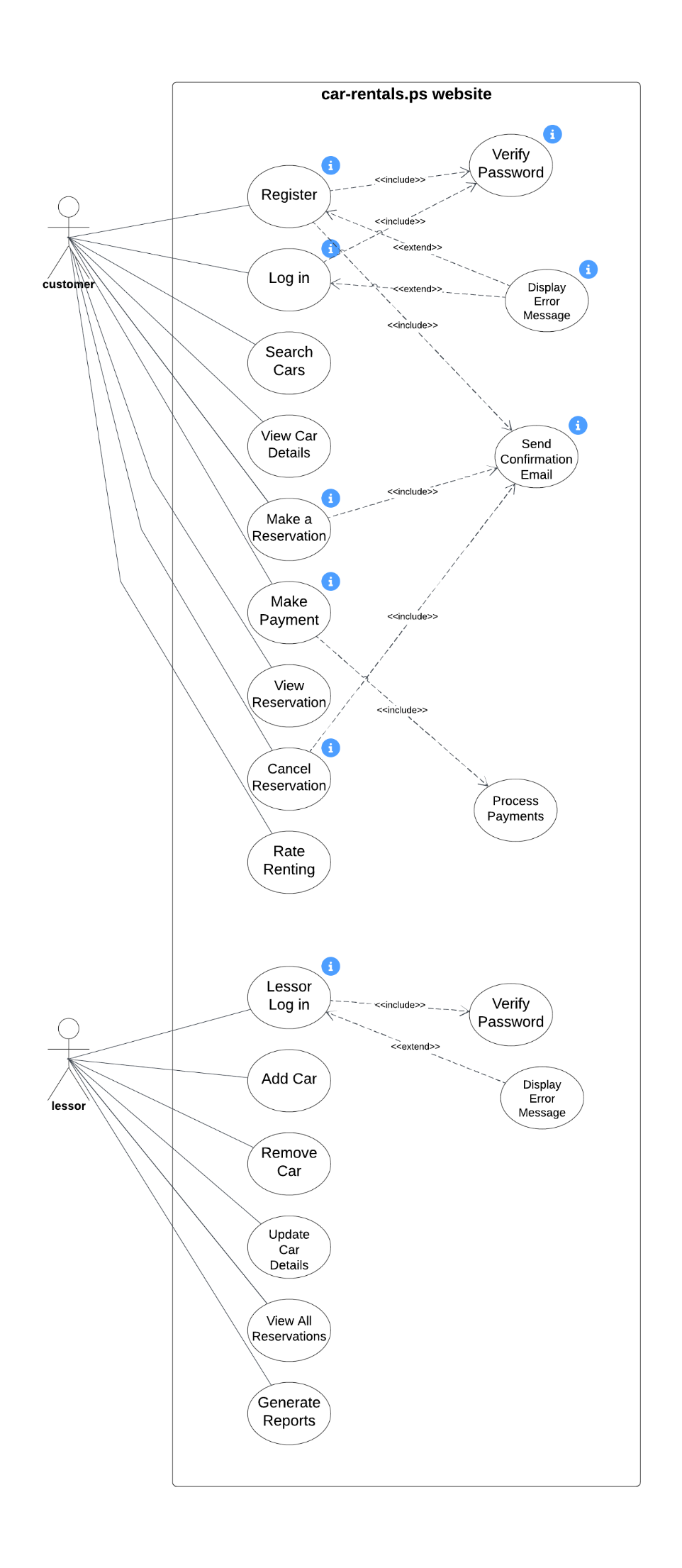
* 1.  **ER Diagram**

Figure 3: ER Diagram

[Click here to check the ER diagram closely](https://lucid.app/lucidchart/9f1bd0a7-113b-4e68-992c-aa58f51f3c64/edit?viewport_loc=-1125%2C-498%2C3481%2C1404%2C0_0&invitationId=inv_c75e2d48-49d5-496e-a2f0-097d51ad138d)

* 1.  **Use Case Diagram**

[Click here to check the Use Case Diagram closely](https://lucid.app/lucidchart/de245eea-7b19-479e-9f43-10535f82adb1/edit?viewport_loc=-1387%2C471%2C6063%2C2812%2C.Q4MUjXso07N&invitationId=inv_998d27fc-57b2-4860-bee5-c686206e69d1)

Figure 4: Use Case Diagram

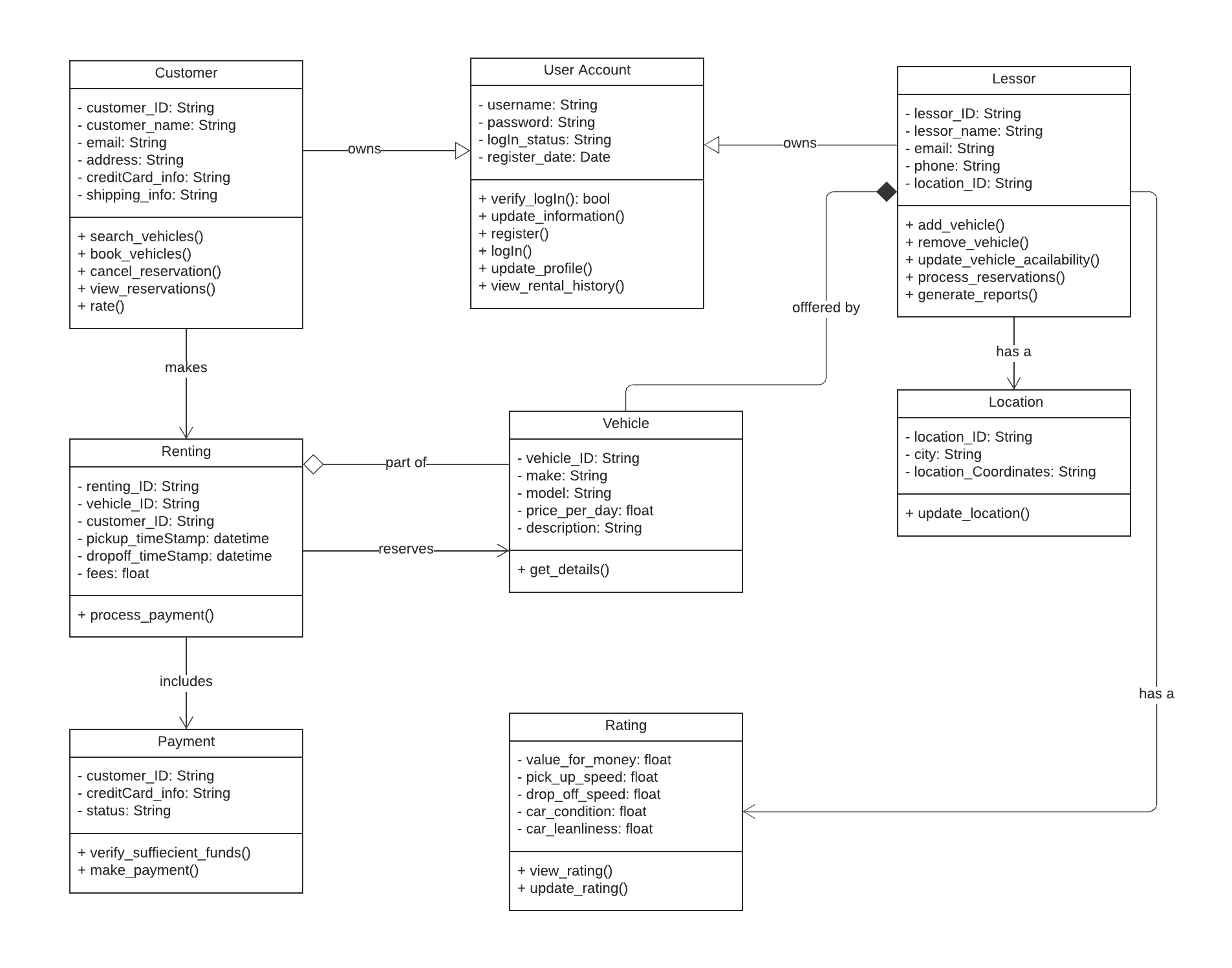
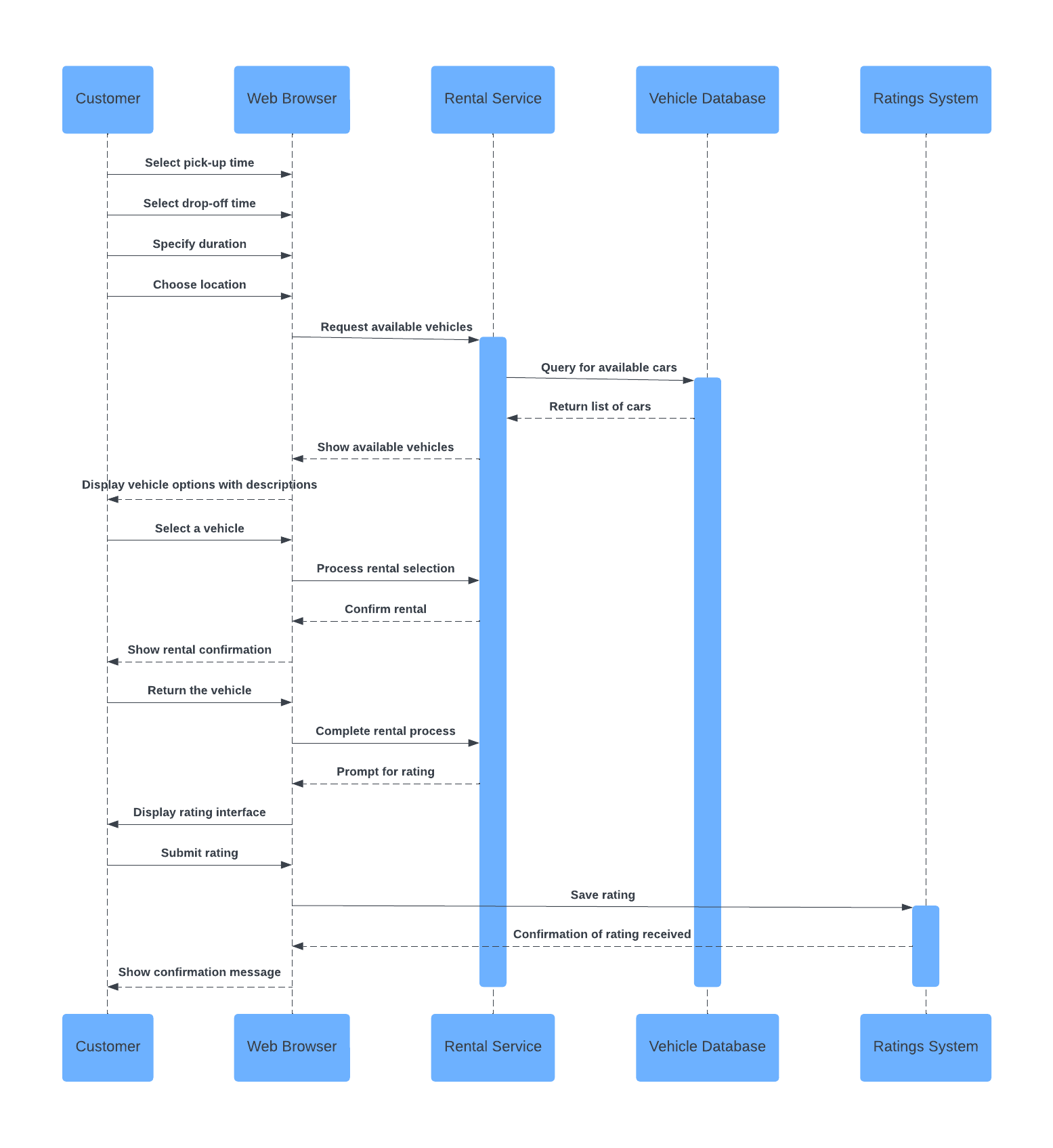
* 1.  **Class Diagram**

Figure 5: Class Diagram

[Click here to check the Class diagram closely](https://lucid.app/lucidchart/ac1c2192-412f-418e-a291-8809a48ddac6/edit?viewport_loc=-77%2C553%2C2400%2C1113%2CHWEp-vi-RSFO&invitationId=inv_c38c0ff3-99c8-4944-8f0e-e0f7bf734aea)

* 1.  **Sequence Diagram**

[Click here to check the Sequence Diagram closely](https://lucid.app/lucidchart/32e81299-af19-41d1-8588-7965edf64817/edit?viewport_loc=-2646%2C59%2C4493%2C2084%2C0_0&invitationId=inv_25dbcdb5-0202-410f-83b7-e90f6f7ba3f9)

Figure 6: Sequence Diagram

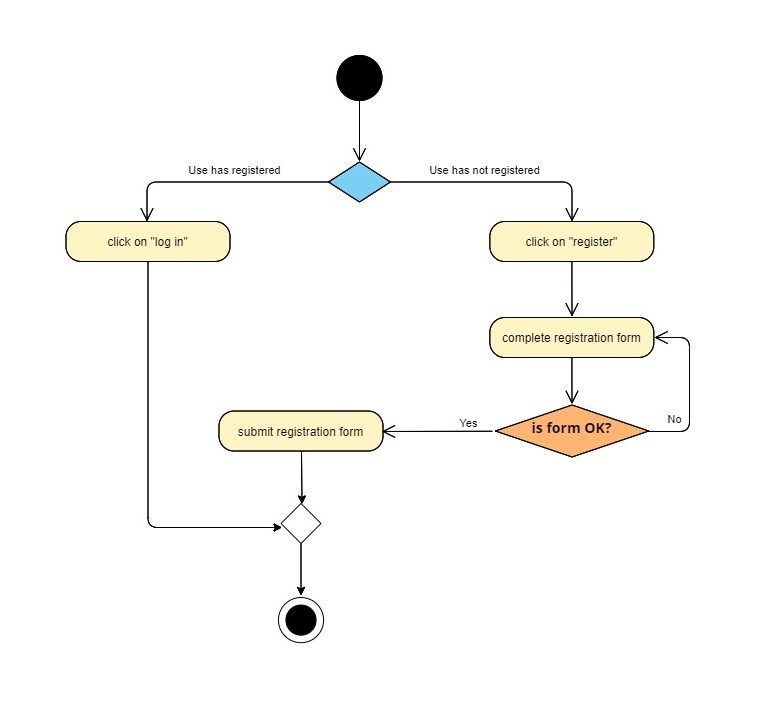
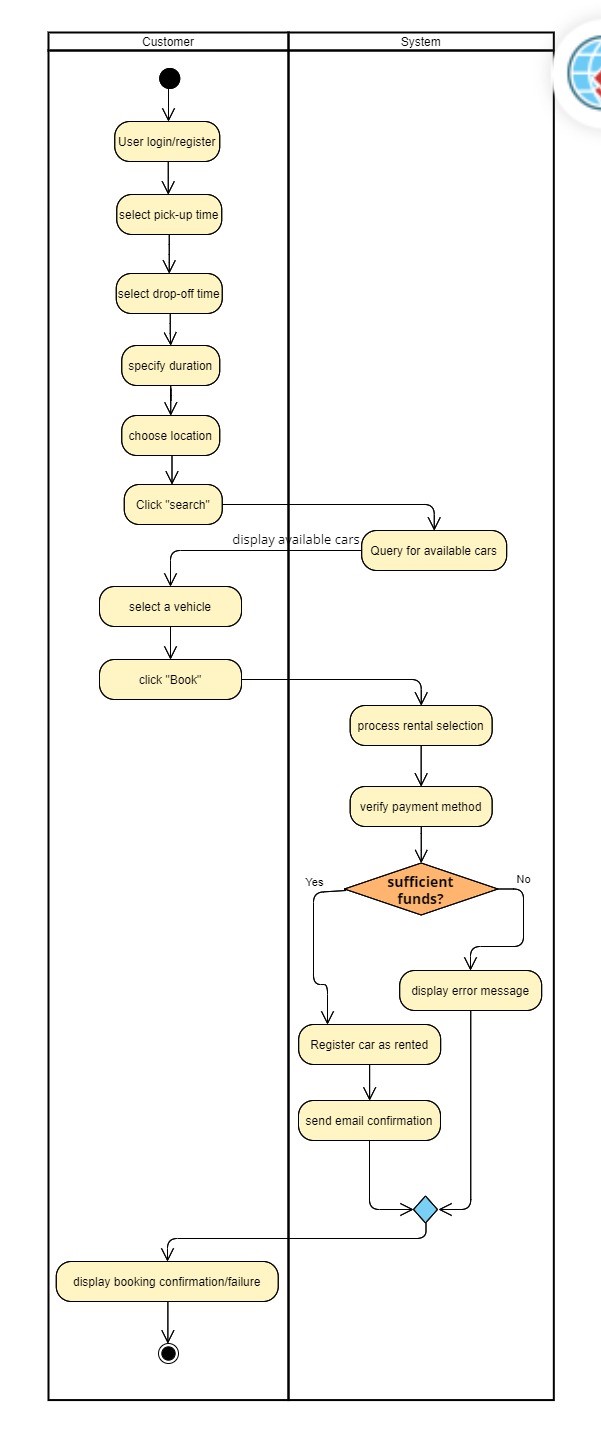
* 1. **Activity Diagrams**
     1. **** **Log In/Register Activity Diagram**

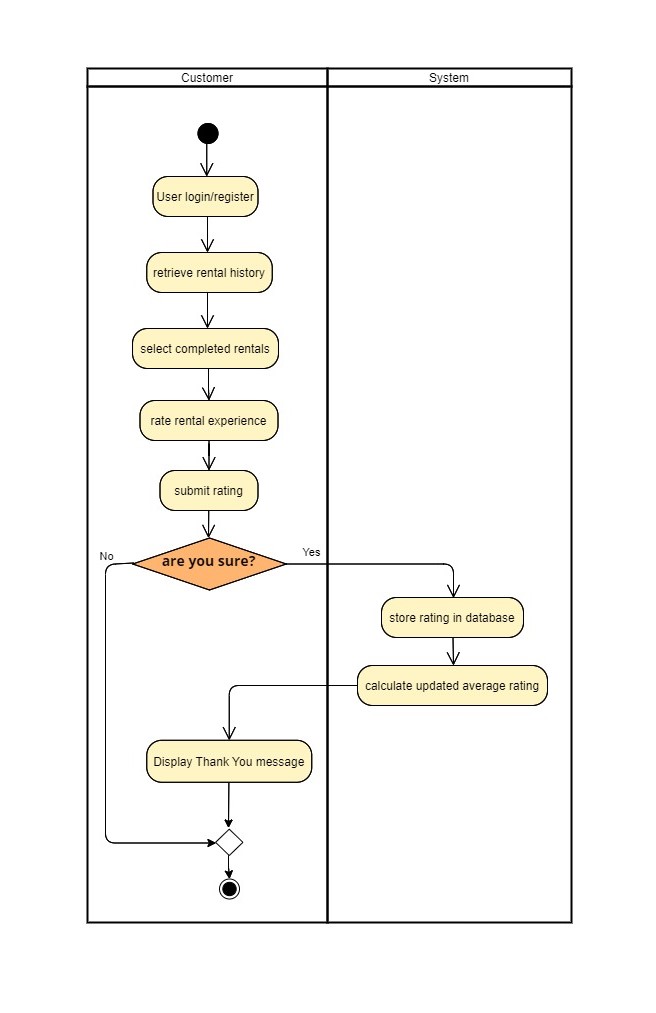
Figure 7: Log In/Register Activity Diagram

[Click here to check the Activity Diagram closely](https://online.visual-paradigm.com/community/book/activity-diagram-car-rental-website-1rthxoe8gn)

* + 1. ** Reservation Process Activity Diagram**

[Click here to check the Activity Diagram closely](https://online.visual-paradigm.com/community/book/activity-diagram-car-rental-website-1rthxoe8gn)

Figure 8: Reservation Process Activity Diagram

* + 1. ** Renting Experience Rating Activity Diagram**

[Click here to check the Activity Diagram closely](https://online.visual-paradigm.com/community/book/activity-diagram-car-rental-website-1rthxoe8gn)

Figure 9: Renting Experience Rating Activity Diagram

# **Chapter 5: User Interfaces**

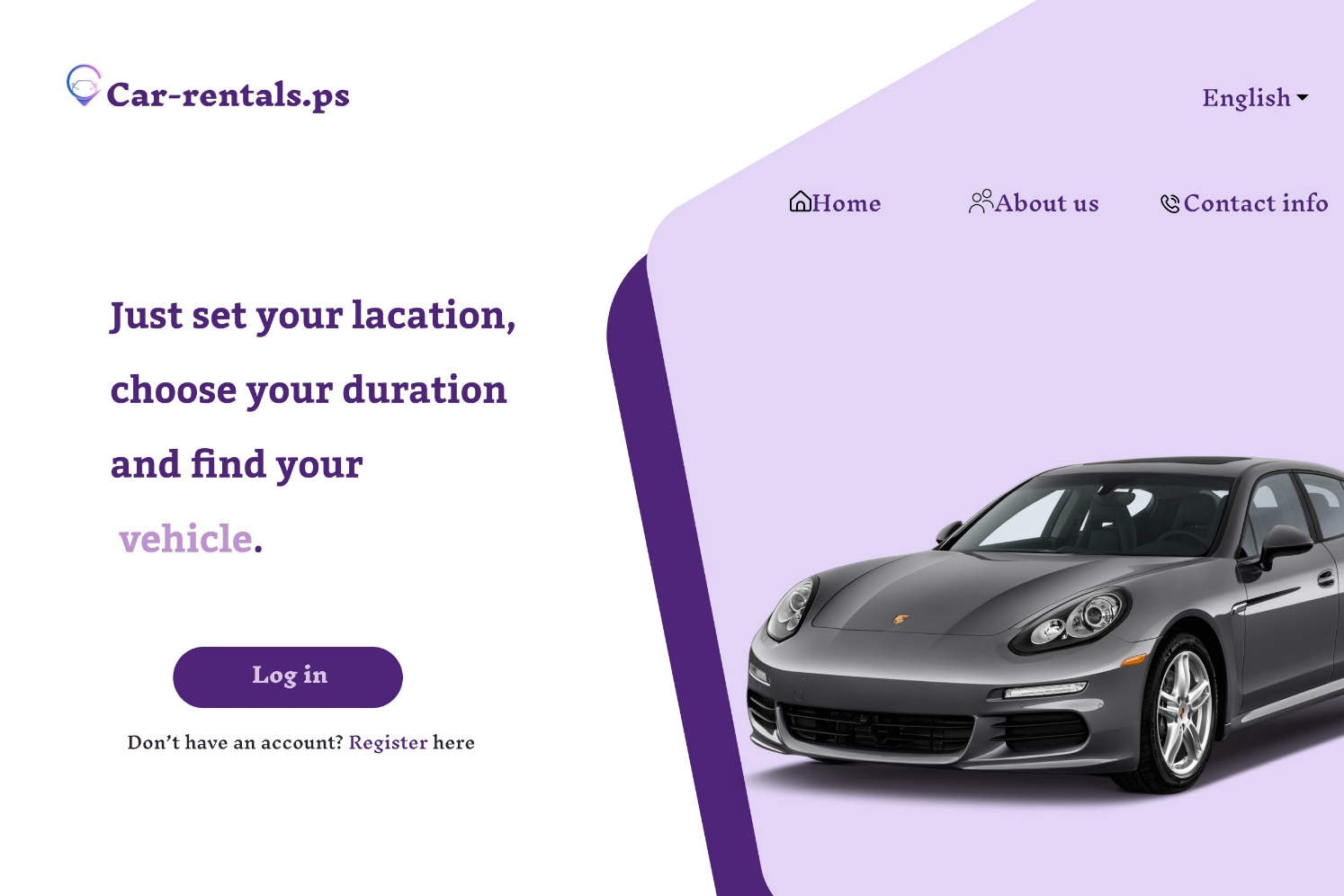
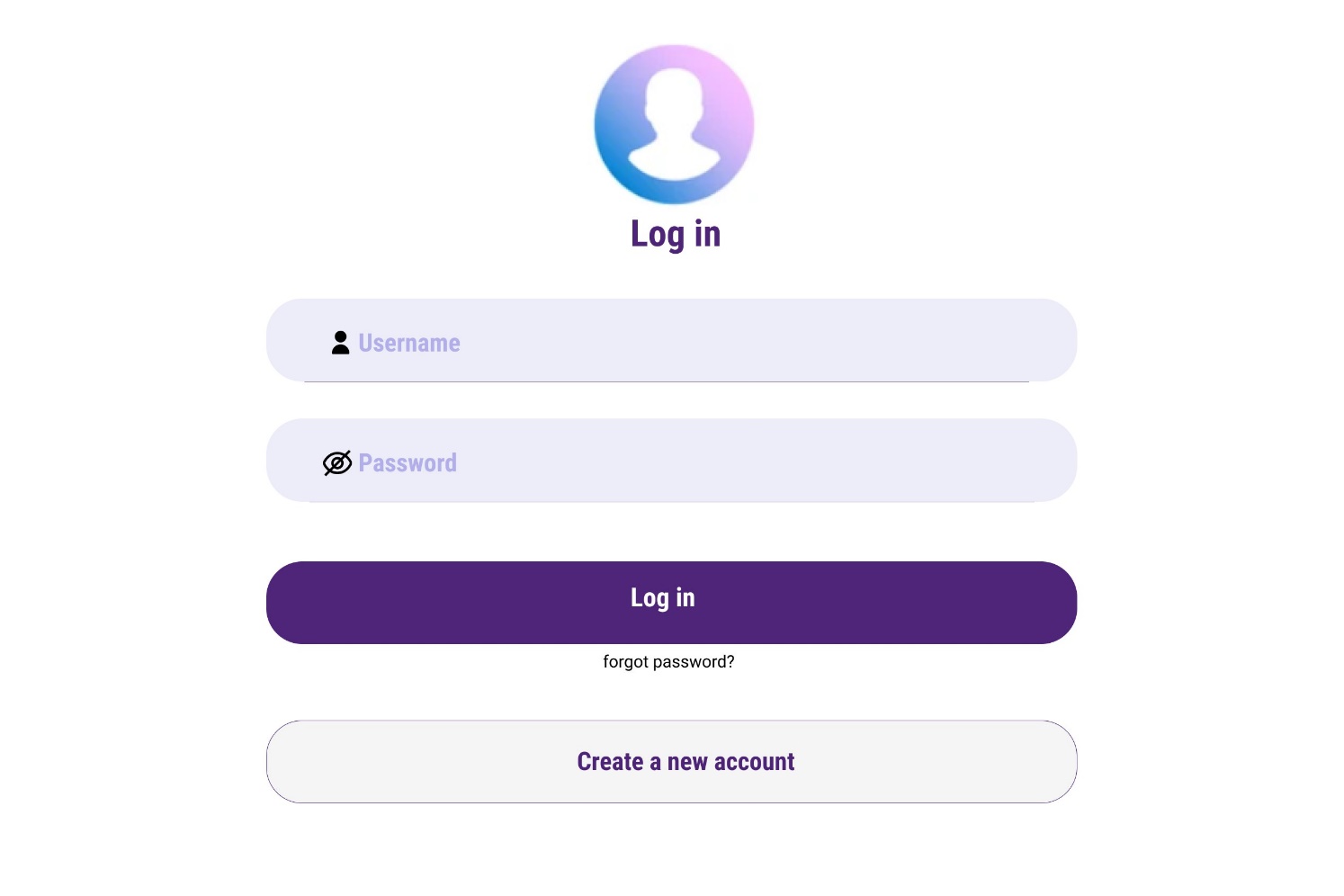
**Model 1:**

Figure 11: Log In Page

Figure 10: Main Page

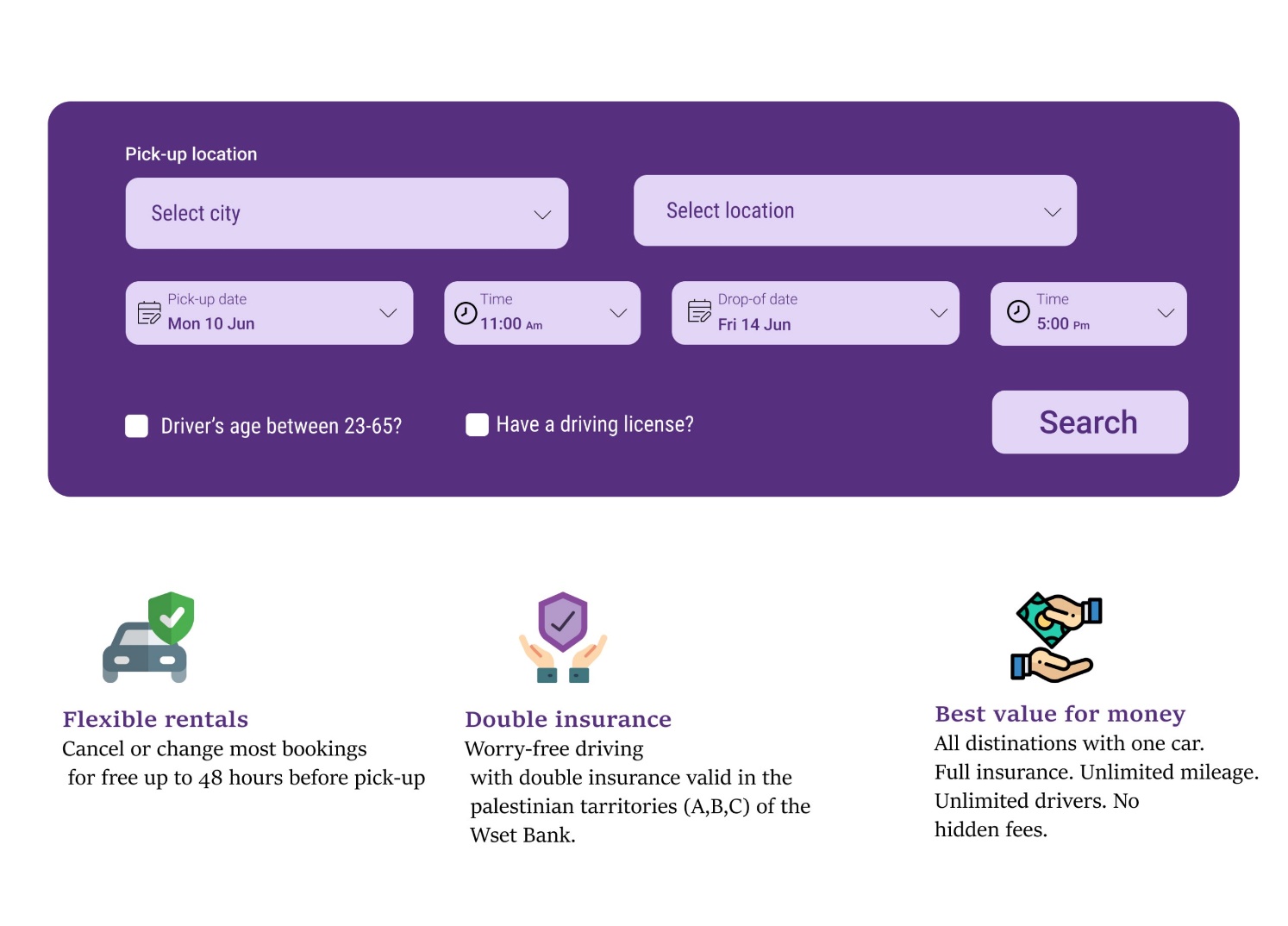


Figure 12: Filter Section

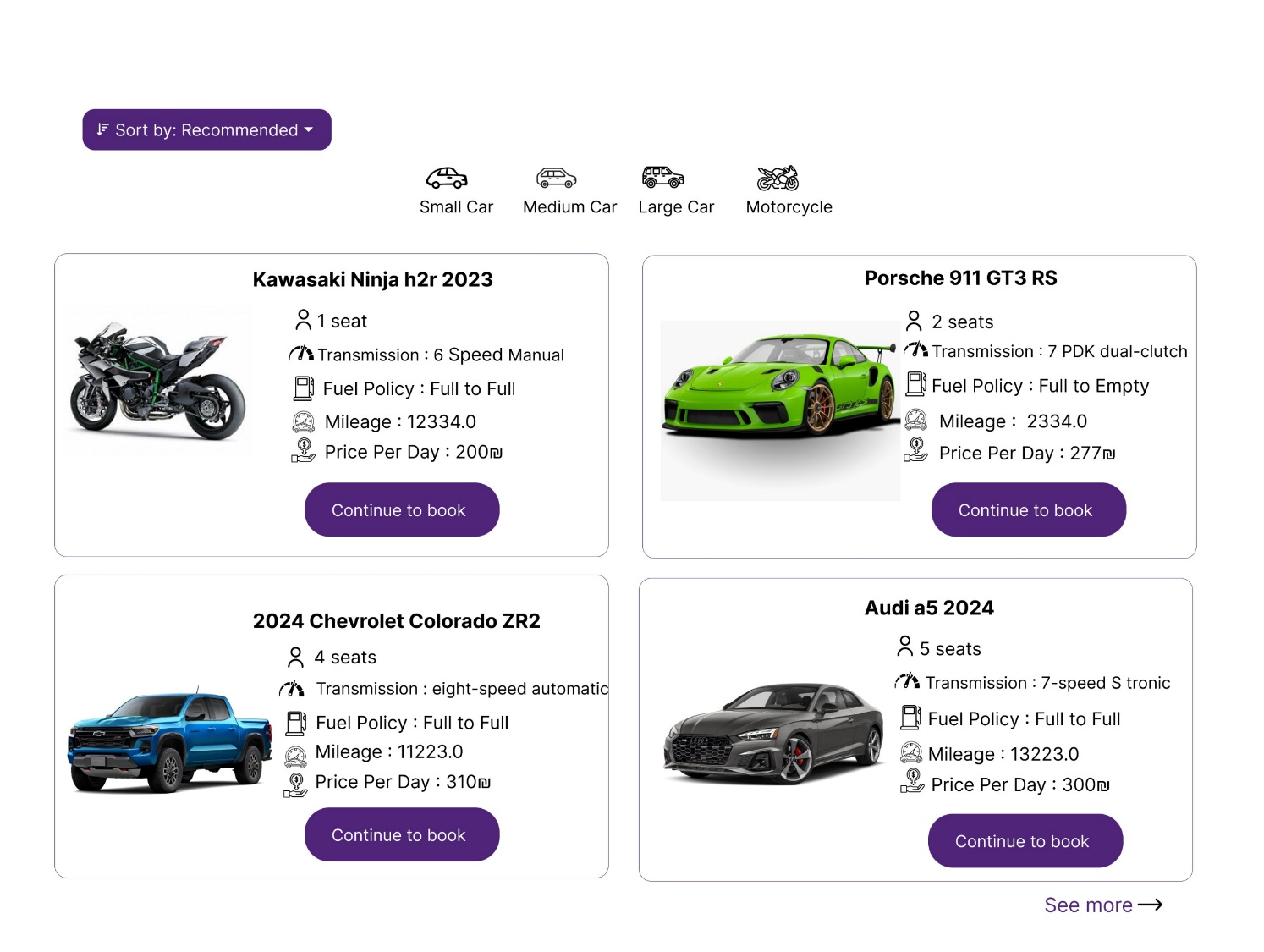


Figure 13: Available Vehicles List for Rent

**Model 2:**

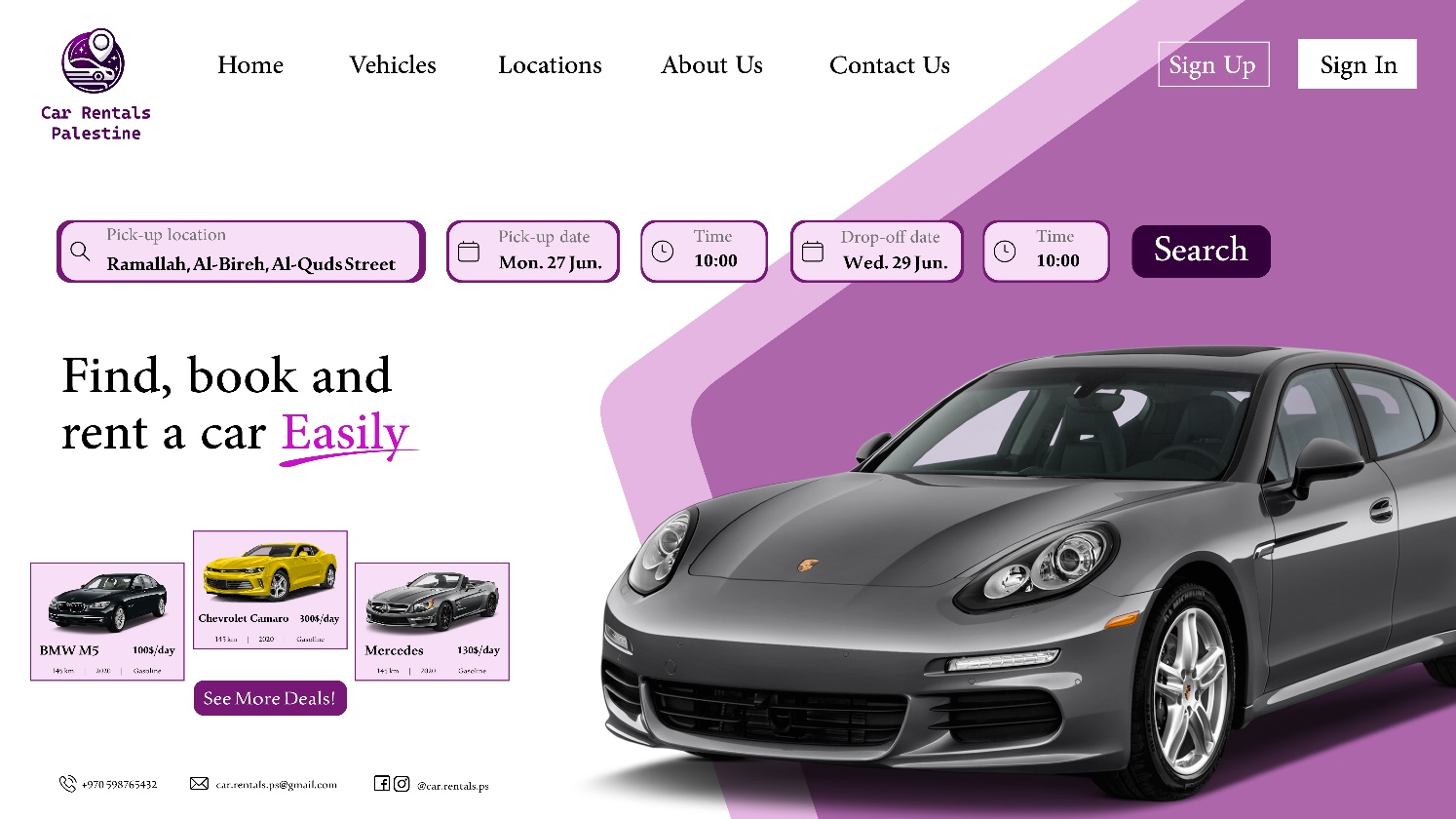
****

Figure 14: Main Page

# **Chapter 6: Implementation**

Here’s the link of the repository, you’ll find the code in the master branch:

<https://github.com/AfnanAbuasal/SW-EN-Project.git>

**Main Function:**

import datetime  
from typing import List, Dict  
  
# Sample data for cars available for rent  
car\_rentals = [  
    {  
        'car\_id': 1,  
        'car\_model': 'Toyota Camry',  
        'location': 'Tulkarm',  
        'availability': [  
            {'pick\_up\_date': '2024-06-01', 'drop\_off\_date': '2024-06-10'},  
            {'pick\_up\_date': '2024-06-15', 'drop\_off\_date': '2024-06-20'}  
        ]  
    },  
    {  
        'car\_id': 2,  
        'car\_model': 'Honda Accord',  
        'location': 'Jenin',  
        'availability': [  
            {'pick\_up\_date': '2024-06-05', 'drop\_off\_date': '2024-06-15'},  
            {'pick\_up\_date': '2024-06-18', 'drop\_off\_date': '2024-06-25'}  
        ]  
    },  
    {  
        'car\_id': 3,  
        'car\_model': 'Ford Mustang',  
        'location': 'Ramallah',  
        'availability': [  
            {'pick\_up\_date': '2024-06-10', 'drop\_off\_date': '2024-06-20'},  
            {'pick\_up\_date': '2024-06-22', 'drop\_off\_date': '2024-06-30'}  
        ]  
    },  
    {  
        'car\_id': 4,  
        'car\_model': 'Chevrolet Malibu',  
        'location': 'Nablus',  
        'availability': [  
            {'pick\_up\_date': '2024-06-01', 'drop\_off\_date': '2024-06-05'},  
            {'pick\_up\_date': '2024-06-08', 'drop\_off\_date': '2024-06-12'}  
        ]  
    }  
]  
  
def filter\_car\_rentals(pick\_up\_date: str, drop\_off\_date: str, location: str) -> List[Dict]:  
    # Convert input dates to datetime objects for comparison  
    pick\_up\_date\_dt = datetime.datetime.strptime(pick\_up\_date, '%Y-%m-%d')  
    drop\_off\_date\_dt = datetime.datetime.strptime(drop\_off\_date, '%Y-%m-%d')  
  
    available\_cars = []  
  
    for car in car\_rentals:  
        if car['location'].lower() == location.lower():  
            for availability in car['availability']:  
                availability\_pick\_up\_date = datetime.datetime.strptime(availability['pick\_up\_date'], '%Y-%m-%d')  
                availability\_drop\_off\_date = datetime.datetime.strptime(availability['drop\_off\_date'], '%Y-%m-%d')  
  
                # Check if the requested period is within the availability period  
                if (availability\_pick\_up\_date <= pick\_up\_date\_dt <= availability\_drop\_off\_date) and \  
                   (availability\_pick\_up\_date <= drop\_off\_date\_dt <= availability\_drop\_off\_date):  
                    available\_cars.append(car)  
                    break  
  
    return available\_cars  
  
# Example usage  
pick\_up\_date = '2024-06-10'  
drop\_off\_date = '2024-06-20'  
location = 'Ramallah'  
  
filtered\_cars = filter\_car\_rentals(pick\_up\_date, drop\_off\_date, location)  
for car in filtered\_cars:  
    print(f"Car ID: {car['car\_id']}, Car Model: {car['car\_model']}, Location: {car['location']}")

# **Chapter 7: Team Work**

|  |  |
| --- | --- |
| **Team member** | **Tasks accomplished** |
| Afnan Abo-Asal | Constructed the software diagrams and SRS document |
| Layla Al-Saabna | Completed the whole implementation (Coding) |
| Rahaf Alawneh | Designed the User Interfaces |

Note: Most of the project’s work was accomplished during Zoom meetings for the team.