Addis Ababa Institute of Technology

School of Information Technology and Engineering

Department of SiTE.

VROOMY -Software Design Specification

Team Members

Name	ID
BEKA BIRHANU ATOMSAU	UGR/3402/14
BETHEL DEREJE TEFERI	UGR/1397/14
DAWIT BELAY YIBELTAL	UGR/8622/14
LEMI DINKU GILO	UGR/3860/14
SAMRAWIT DAWIT MENGSTU	UGR/0221/14
SIMON DEREJE WOLDEAREGAY	UGR/0952/14

Advisor: Nuniyat Kifle

Date:

Table of Contents

List of Tables

List of figures

Definitions, Acronyms, Abbreviations

UML – Unified Modeling Language

SDS – Software Design Specifications

DB – Database

UI – User Interface

1. Introduction

1.1 Purpose

The purpose of this Software Design Specifications (SDS) document is to provide a comprehensive and detailed blueprint for the development and implementation of a Vroomy. This platform is envisioned to be a robust and user-friendly web application, facilitating seamless interactions between car owners and renters in a secure and efficient manner. The SDS will outline the structural and behavioral aspects of Study Pal through UML component diagrams, deployment diagrams, class diagrams, detailed class diagrams, and sequence diagrams.

1.2 General Overview

The Car Rental Platform is designed to provide a seamless and efficient solution for users to rent and manage vehicles. In the system context, our platform sits at the intersection of car owners and renters, facilitating a dynamic exchange of vehicles while ensuring a user-friendly experience.

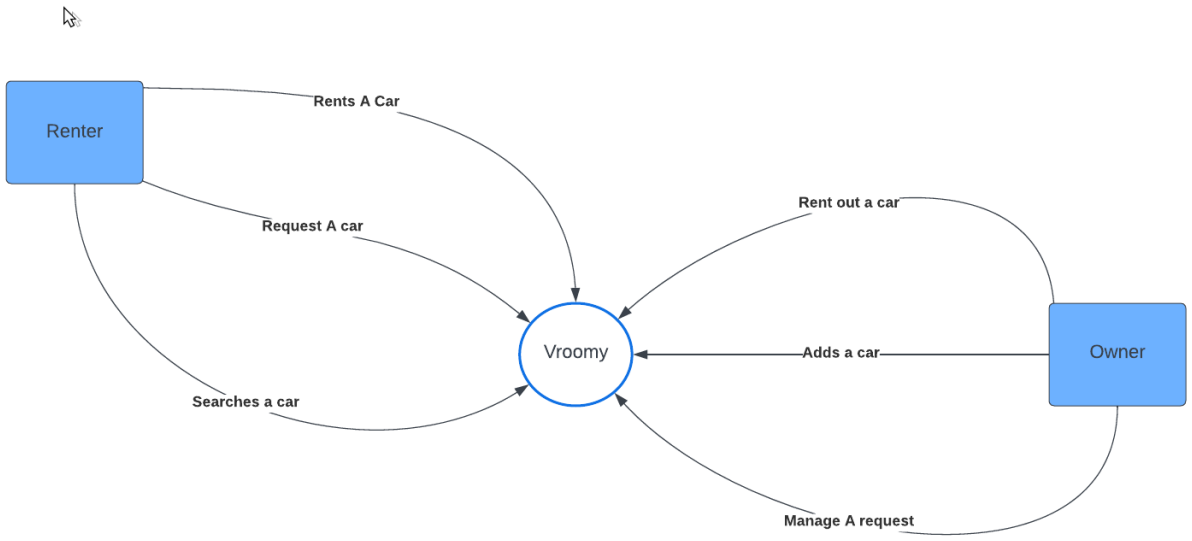
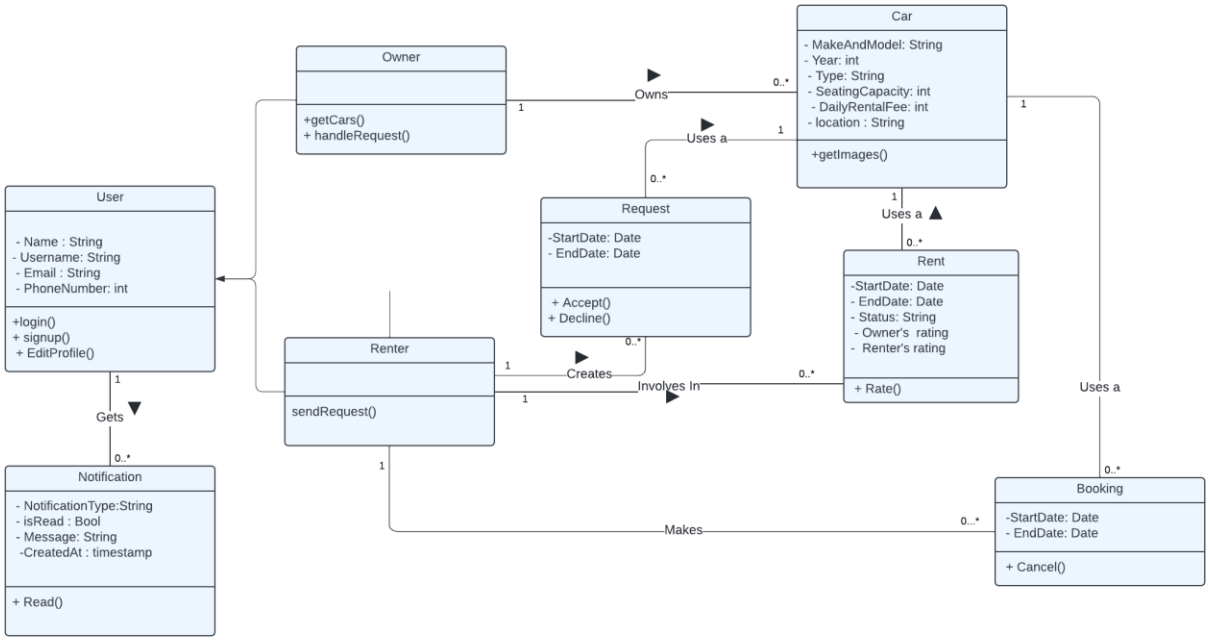


Figure 1: Context Diagram

Overview of System and Software Architectures:

The system architecture is grounded in a client-server model, where users interact with the platform through a web-based interface (client) that communicates with a centralized server hosting the application and database. This approach ensures accessibility, scalability, and centralized data management.

Design Goals:



1. **Clear Separation of Concerns:** Maintain a distinct separation between the Presentation, Business Logic, and Data Access Layers, ensuring clarity, ease of maintenance, and flexibility in development.
2. **Scalability and Maintainability:** Leverage the three-layer architecture to facilitate scalability, allowing for the efficient addition of features and adaptations to changing business requirements. The modular design enhances maintainability and ease of updates.
3. **Efficient Data Handling:** Optimize data management through the Data Access Layer, ensuring swift and reliable storage and retrieval of information.
4. **Consistent User Interface:** Implement a consistent and intuitive user interface across the Presentation Layer, fostering a positive user experience and ease of navigation.
5. **Enhanced Security:** Implement security measures across layers to safeguard user data and financial transactions, aligning with industry standards and best practices.

1.3 Development Methods & Contingencies

System and Software Design Methodology:

The system and software design for the Car Rental Platform employ an Object-Oriented Design (OOD) approach, utilizing Unified Modeling Language (UML) for modeling and documentation. Object-Oriented principles are leveraged to encapsulate entities and functionalities within modular and reusable objects, promoting a flexible and scalable design.

UML and Object-Oriented Design: UML diagrams, including class diagrams, sequence diagrams, and component diagrams, play a pivotal role in visualizing the system's structure and behavior. Object-Oriented Design facilitates the encapsulation of data and behavior within classes, promoting code reusability, and enhancing maintainability.

Considered Methods: During the design phase, several methodologies were considered, including Structured Design and Prototyping. While Structured Design provides a systematic approach to software design, Object-Oriented Design was favored for its ability to model real-world entities effectively and support adaptability to evolving requirements.

Contingencies and Workarounds:

The Car Rental Platform has been designed to leverage an external API for robust user authentications, enhancing security and ensuring a seamless user experience. However, recognizing the inherent risk associated with external dependencies, such as changes, downtime, or discontinuation of the chosen authentication service, contingency plans have been established.

Alternative Plans:

Recognizing the critical nature of user authentication, a comprehensive alternative plan involves the development of a customized authentication system for the Car Rental Platform.

2. System Architecture

2.1 Subsystem decomposition

Vroomy follows a **three-layer model**, comprising the **Presentation Layer**, **Business Logic Layer**, and **Data Access Layer**.

1. **The Presentation Layer** ⇒ manages user interfaces, interactions, and overall user experience.

2. **The Business Logic Layer** enforces business rules, processes data, and orchestrates interactions between the Presentation and Data Access Layers.
3. **The Data Access Layer** handles storage and retrieval of data, ensuring efficient database interactions.

The software architecture aligns with the principles of three-layer architecture, promoting separation of concerns. This modular approach enhances maintainability, scalability, and code reusability. The Presentation Layer encapsulates user interface components, the Business Logic Layer manages application logic, and the Data Access Layer handles database interactions.

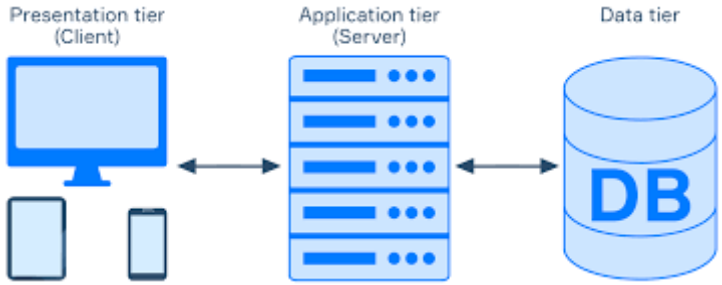


Figure 2. Layer 1 of component diagram

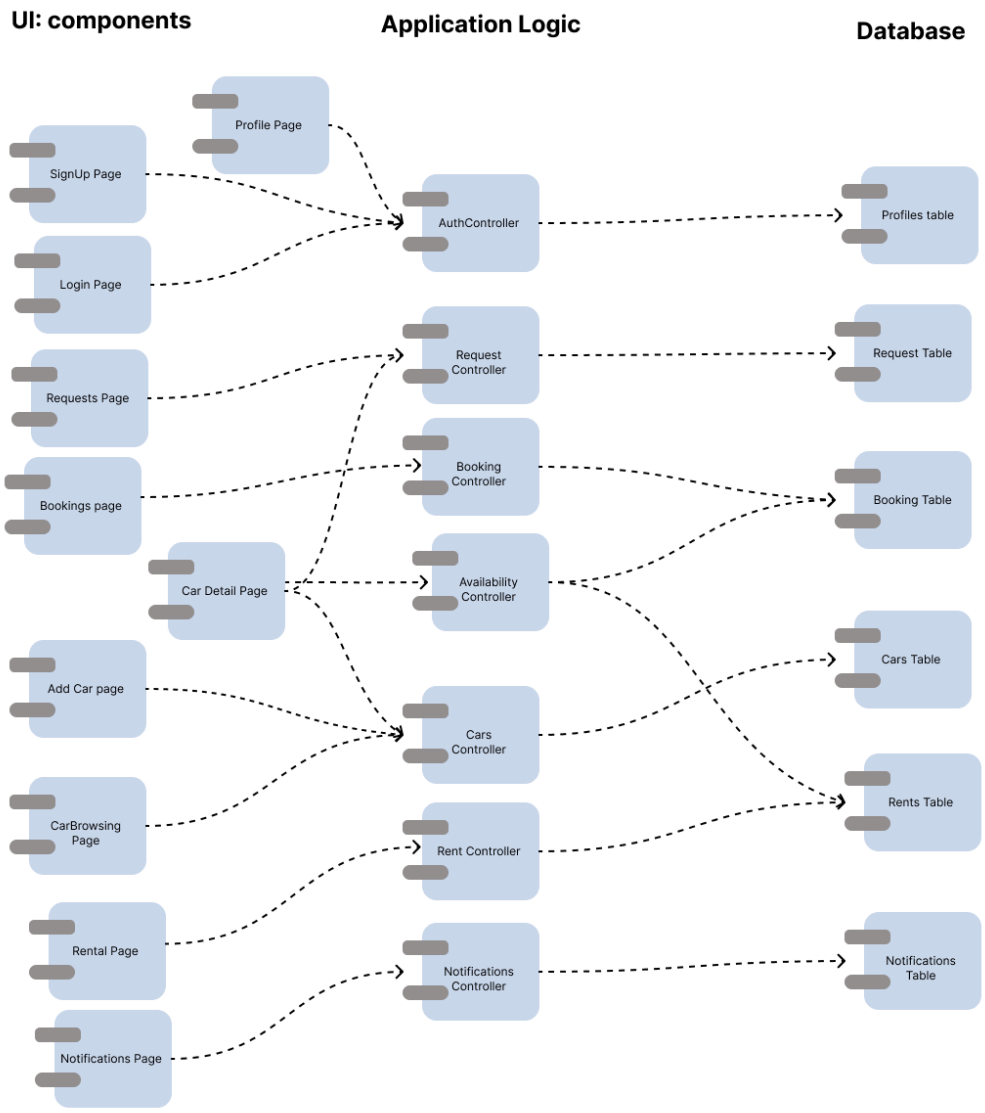


Figure 3: Layer 2 of component diagram

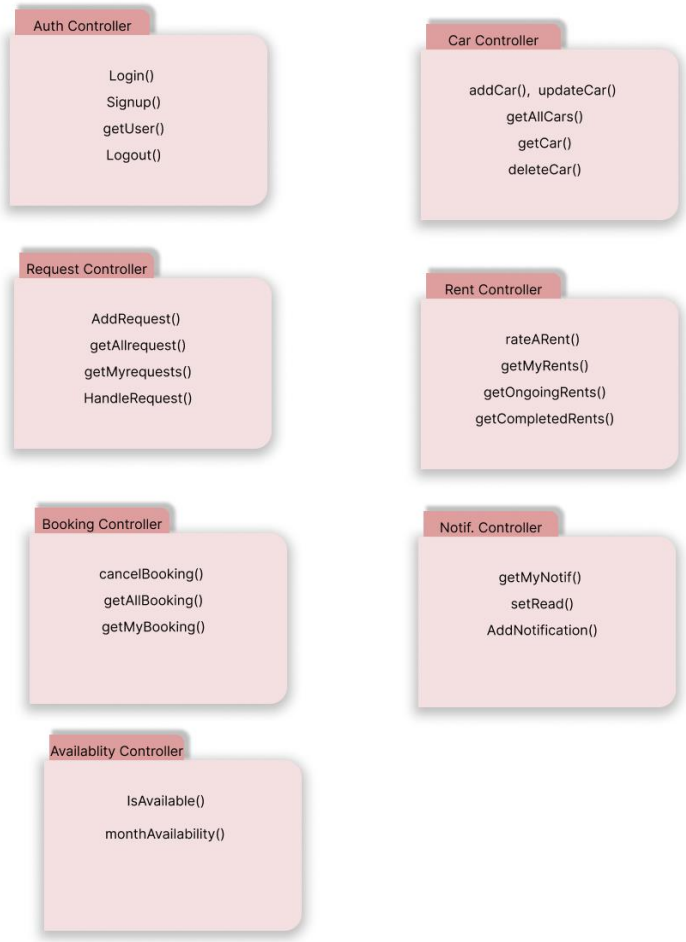


Figure 4: Layer 3 of component diagram

2.2 Hardware/software mapping

UML Deployment diagram

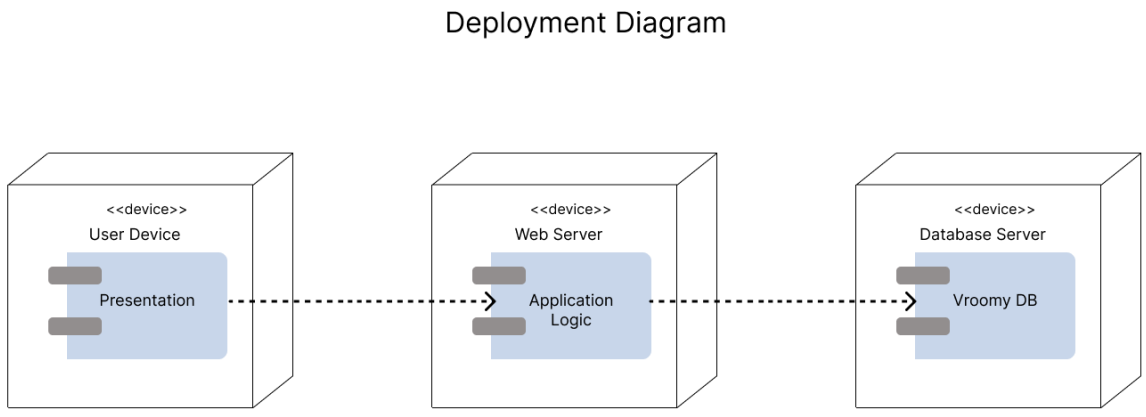


Figure 5: Deployment Diagram

3. Object Model

3.1 Class Diagram

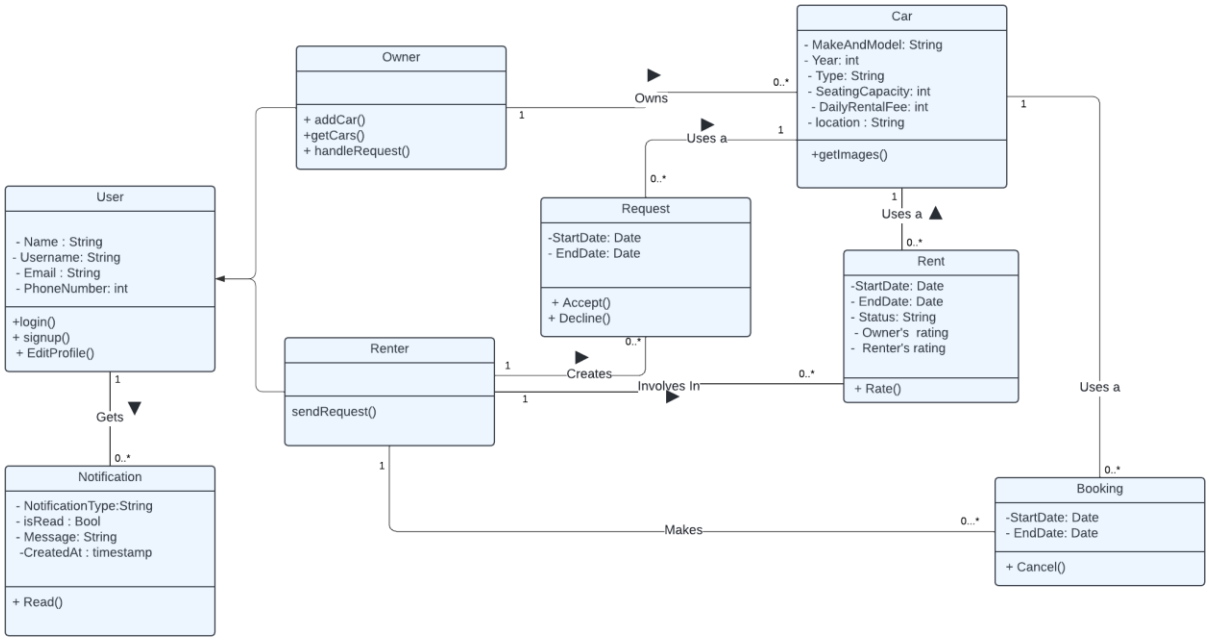


Figure 6: UML Class Diagram

3.2 Sequence Diagram

3.2.1 Search and Filter cars

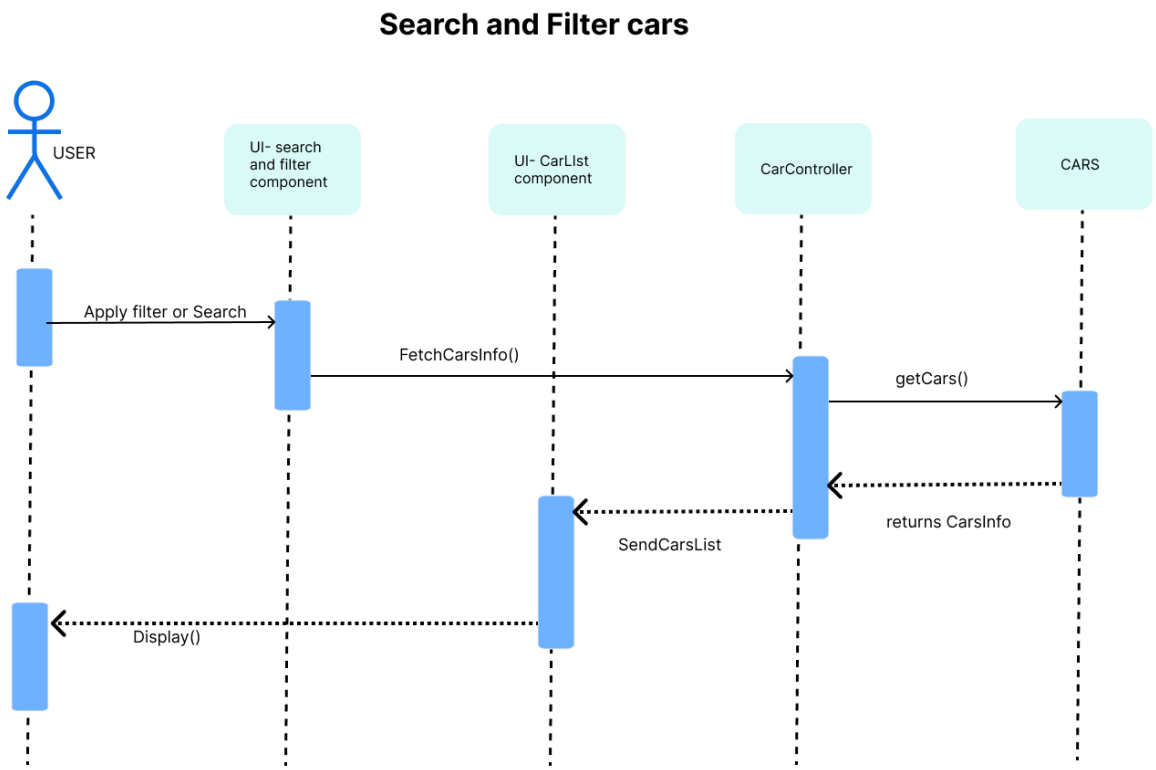


Figure 7: Search and Filter cars

3.2.2 Add or Edit car

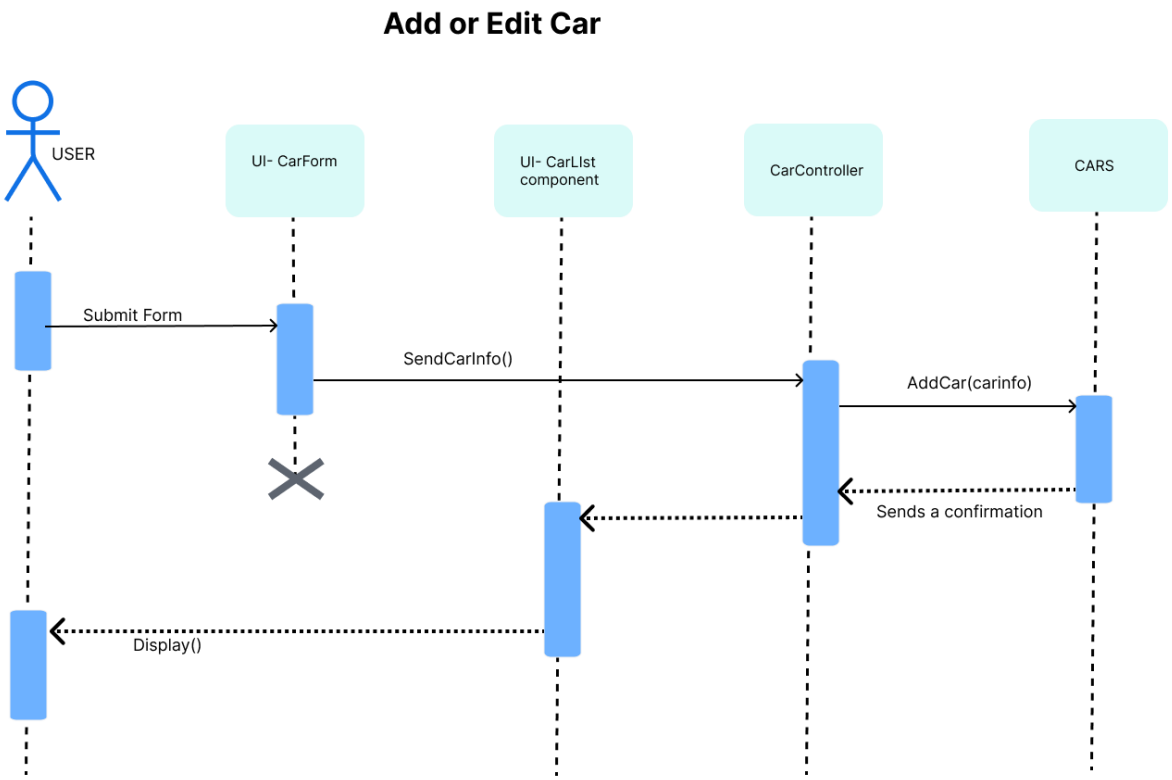


Figure 8: Add or Edit a car

3.2.3 View Car Details

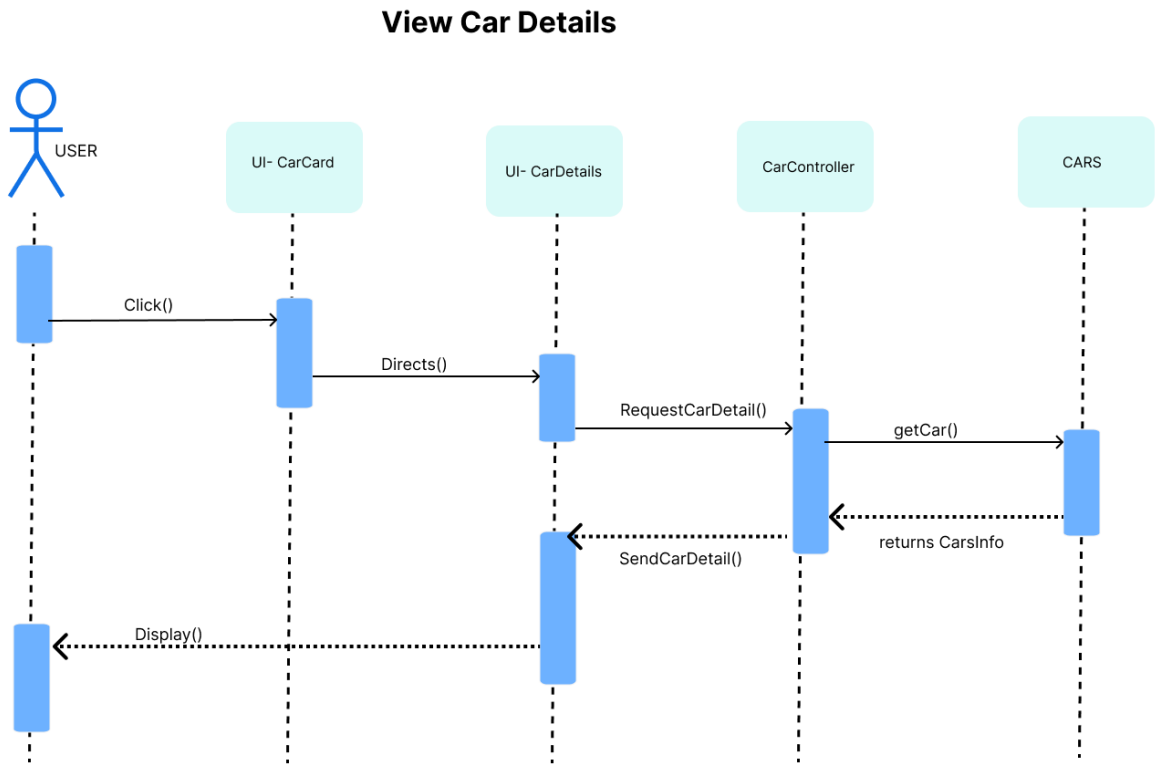


Figure 9: View car details

3.2.4 Send Request

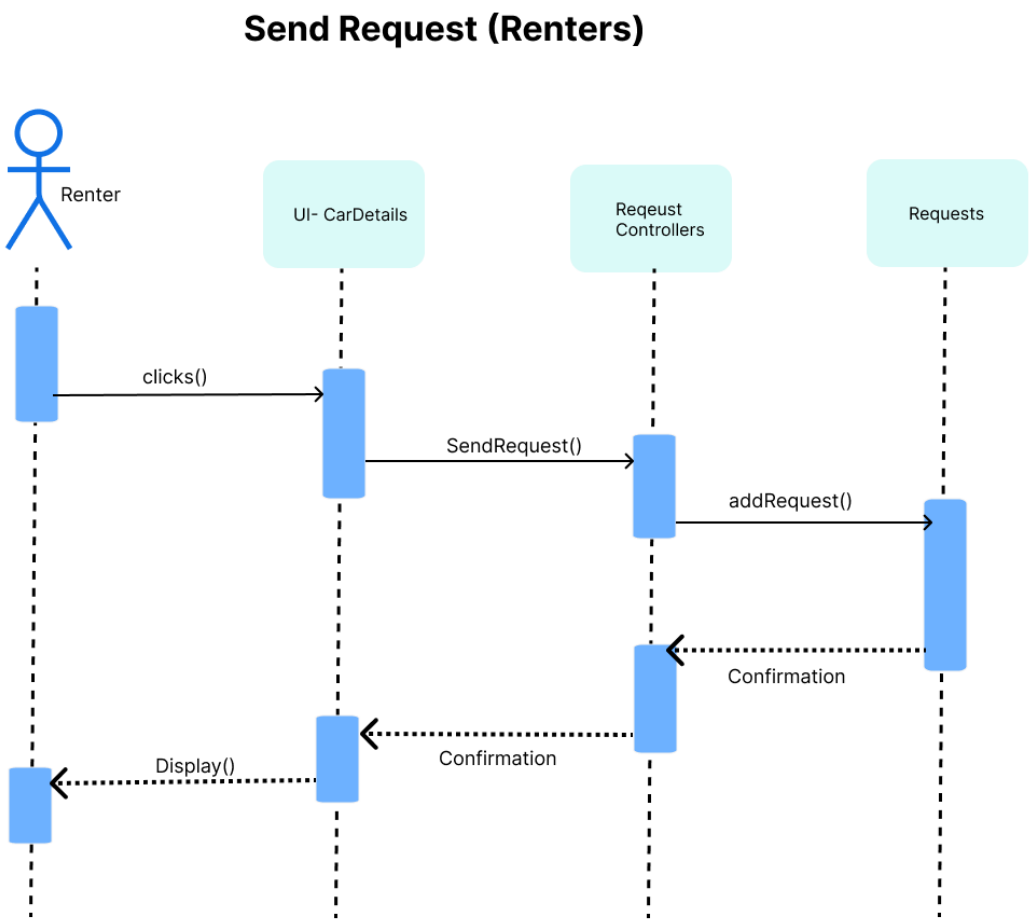


Figure 10: Send Request

3.2.5 Accept or decline request

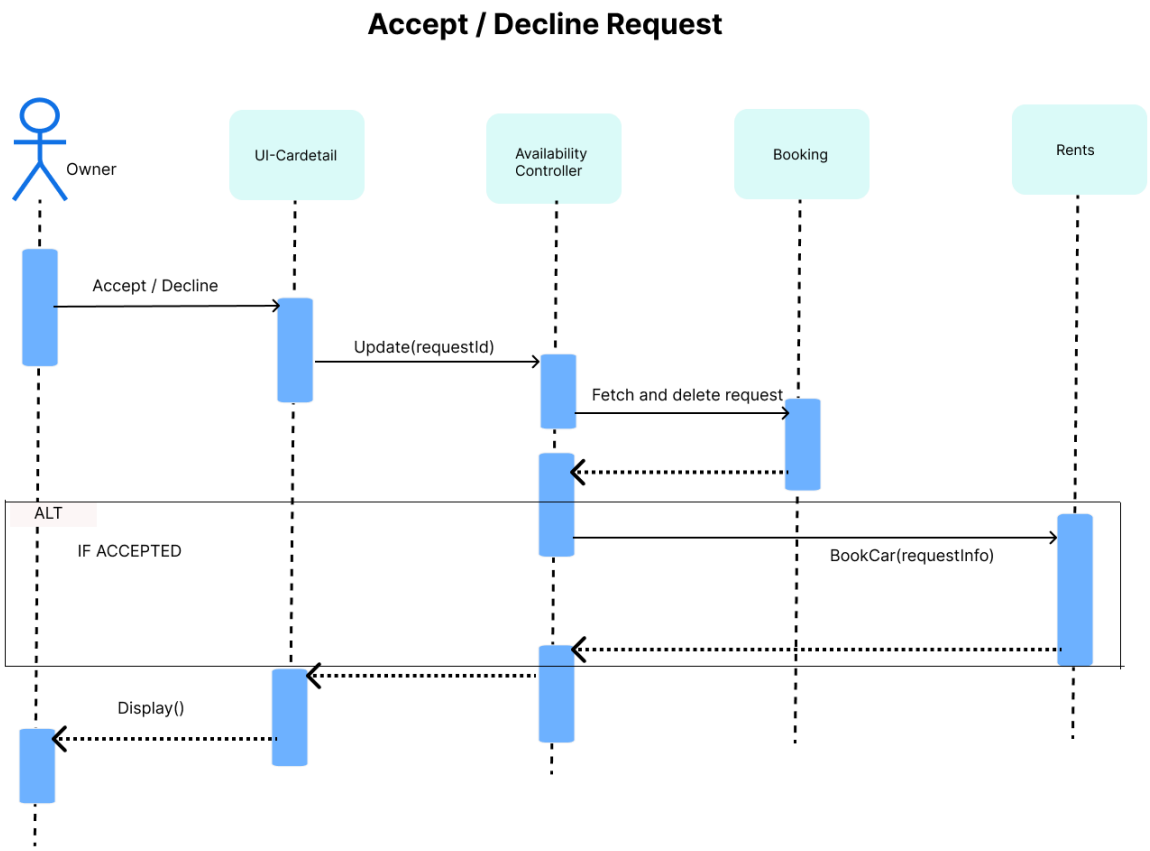


Figure 11: Accept or Decline request

3.2.6 Cancel Booking

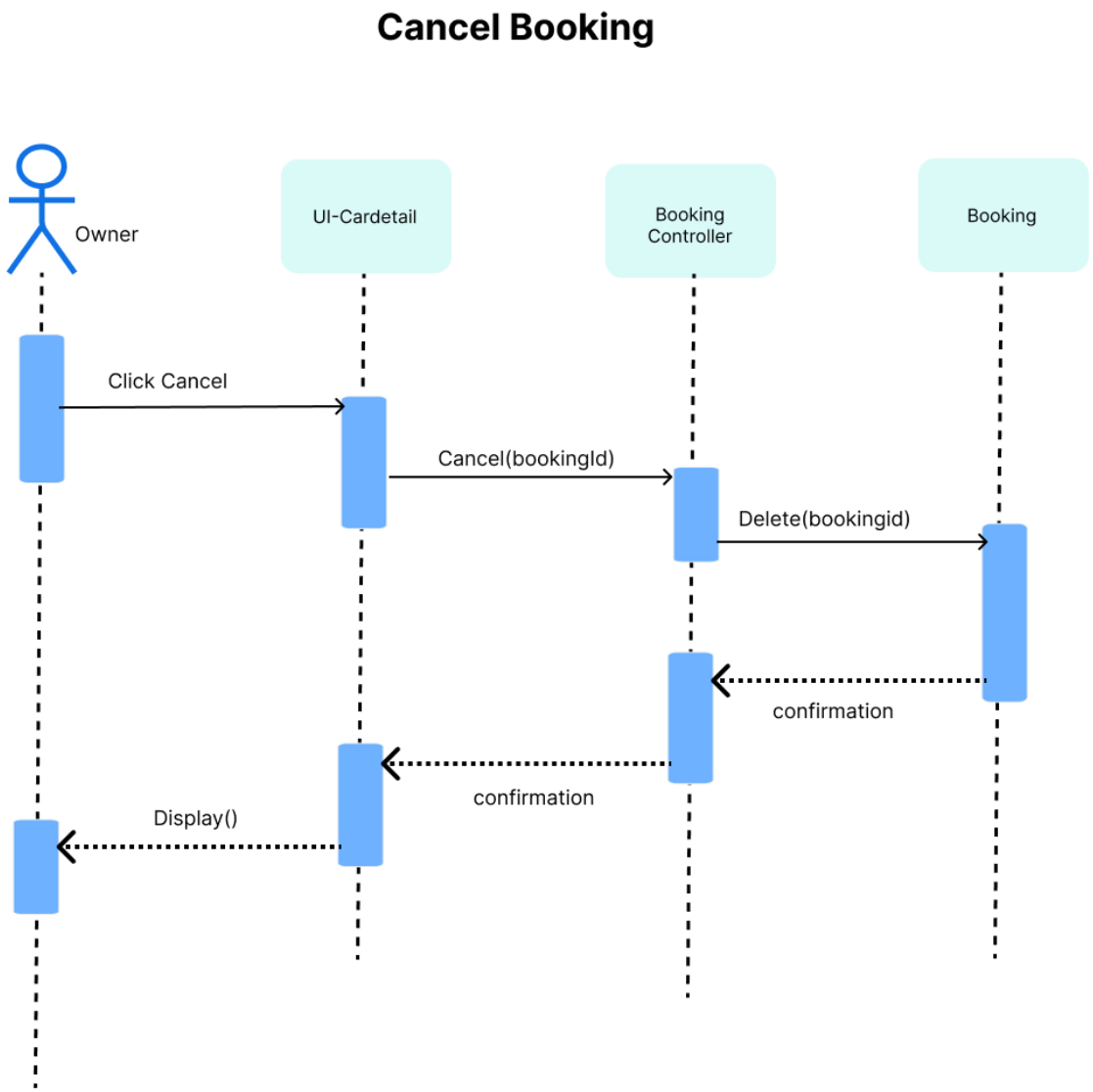


Figure 12 Cancel Booking

3.2.7 Check Availability

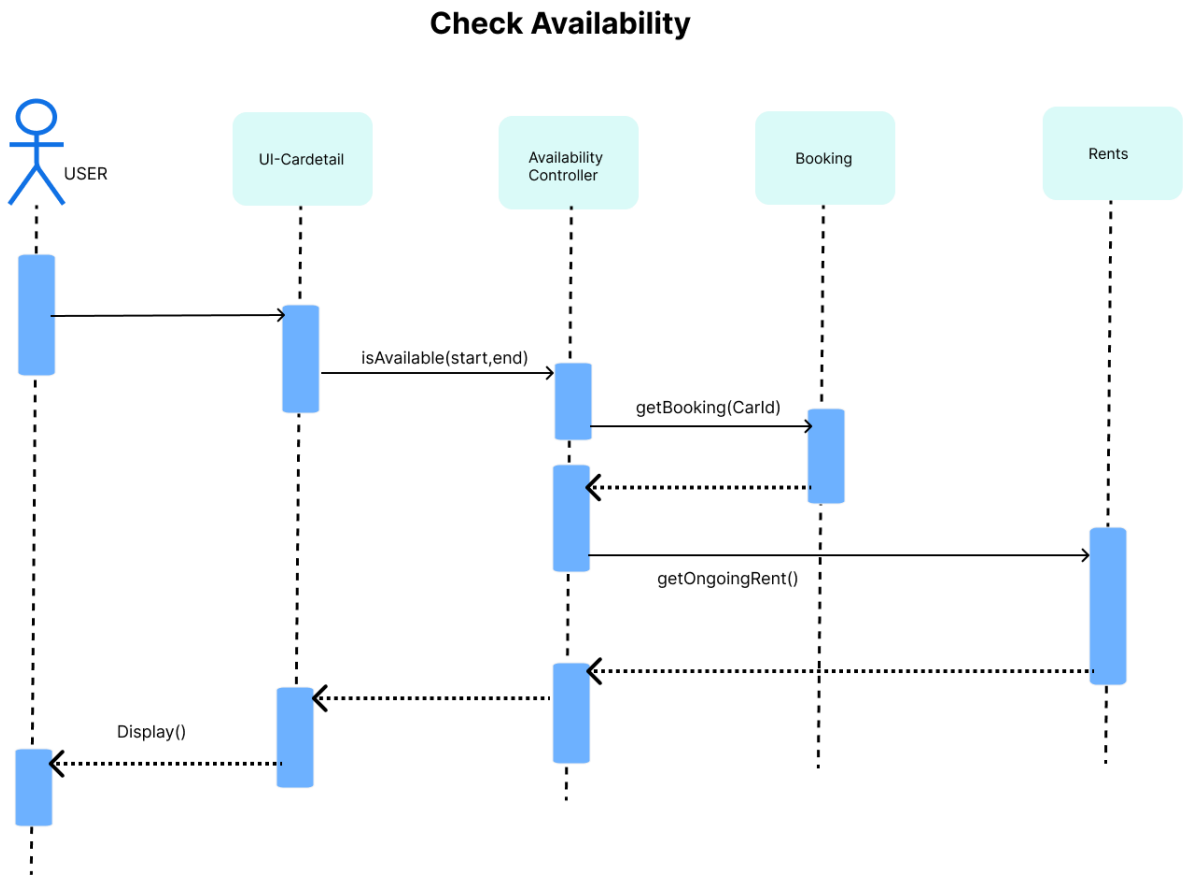


Figure 13: Check availability

3.2.8 Rating a rent

After a completion of a rent, both parties (renter and owner) will be prompted to rate their experience in that rental process. This review will indirectly affect the users rating.

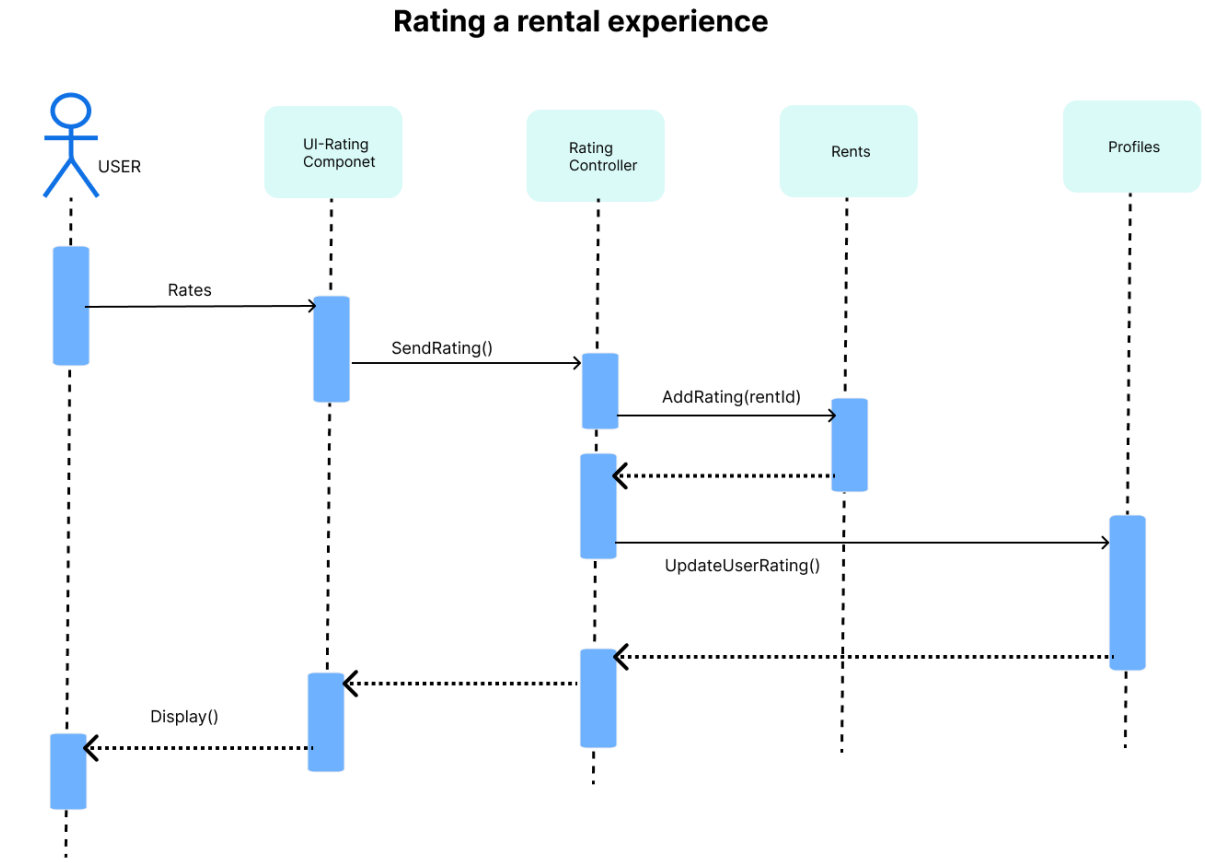


Figure 14: Rating a rental experience

3.2.9 Login

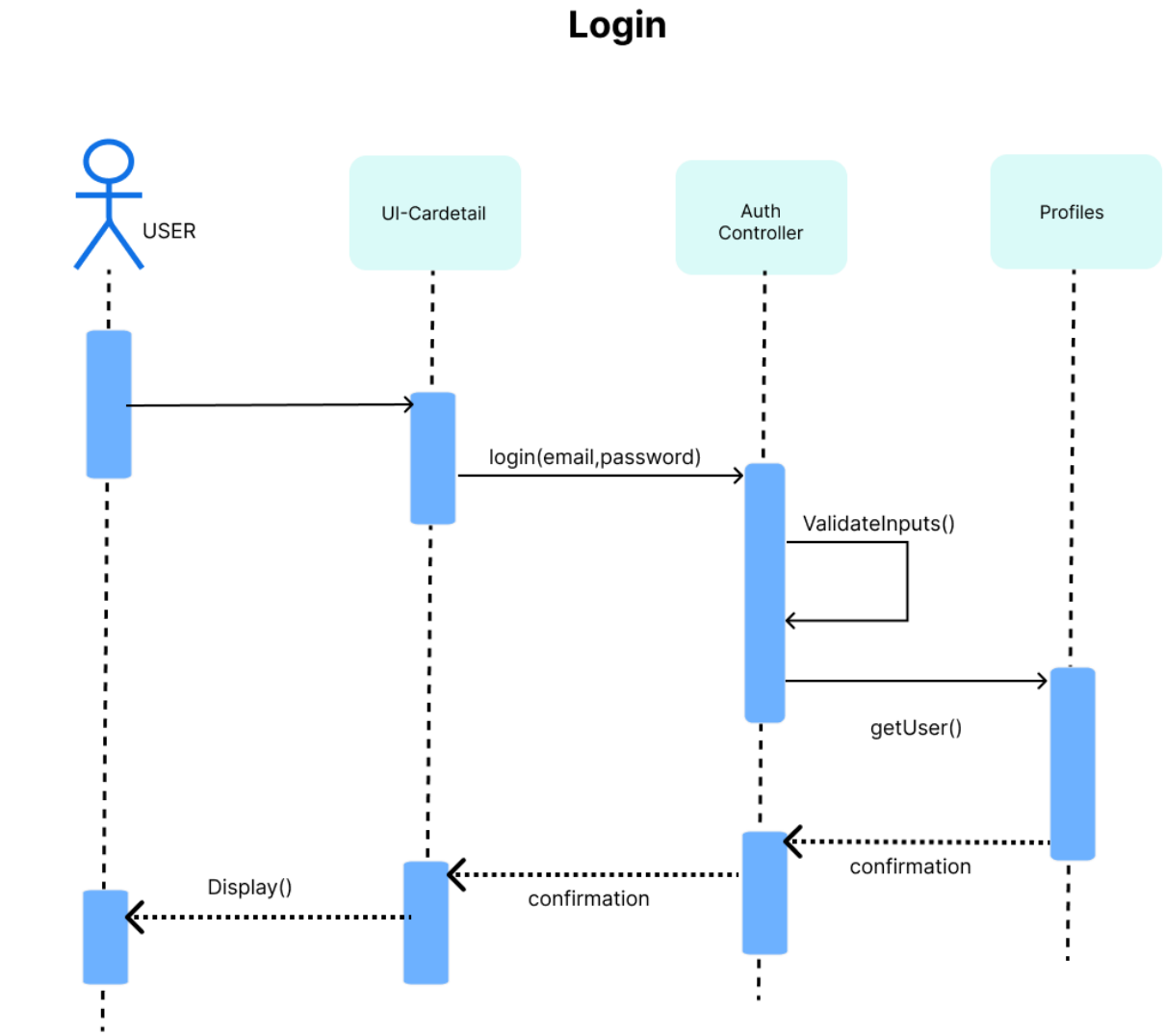


Figure 15: Login

3.3 State chart Diagram (optional element)

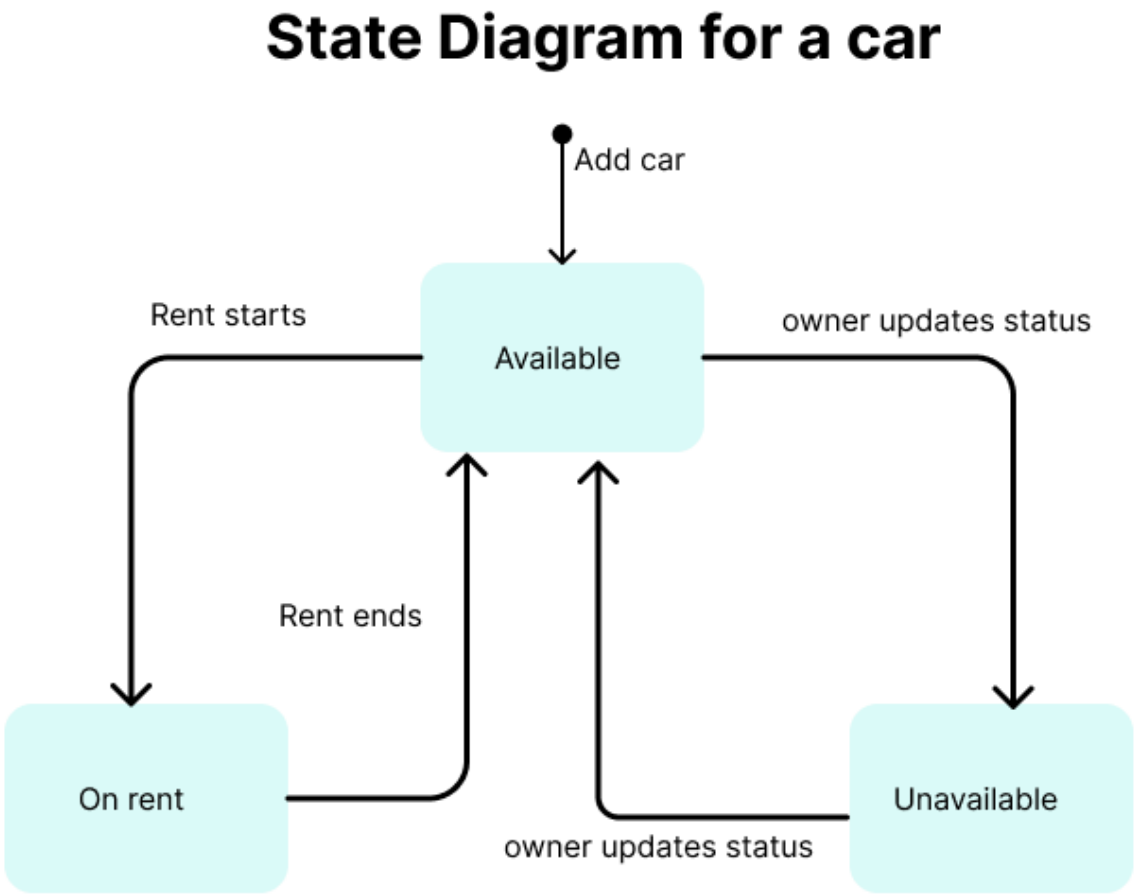


Figure 16 State diagram for a car

4. Detailed Design

NOTE: As the implementation is not performed using Object-Oriented Programming (OOP), most of the classes mentioned below are implemented as modules. Consequently, they won't possess attributes; instead, they will function as modules providing related operations.

4.1 AuthController

AuthController
+userLogin() +userSignUp() +editProfile() +getUserInfo()

Table 1: AuthController

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
userSignup()	Public	void	Email Name Username Phone no	The user must provide a valid and complete input	The user will be register in to the database.
userLogin()	Public	void	Email Password	The user already have an existing account	The user will be provided with session token.
EditProfile()	Public	void	-	The user has to be logged in.	The user Info will be updated
getUserInfo()	Public	object	-	The user has to be logged in.	The user Info will be returned

Table 2: AuthController Operation Description

4.2 Car Controller

CarController

+addCar() +UpdateCar() +getCarInfo() + getAllCars() +deleteCar ()

Table 3: Request Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addCar()	Public	void	CarInfo: object	The user must have an owner account	A car info will be added.
updateCar()	Public	void	Car id	The user must have an owner account	A car info will be updated.
getCarInfo()	Public	object	Car id	-	A detailed car info will be returned.
getAllCars()	Public	object	-	-	A list of cars will be returned
deleteCar()	Public	void	Car id	The user must have an owner account	A car info will be removed from the database.

Table 4: Car Controller Operation Description

4.3 Request Controller

RequestController
+AddRequest() +getReceivedRequest() +getSentRequests() +handleRequest() +cancelRequest()

Table 5: Request Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
addRequest()	Public	void	-	The user must have an renter account	A car info will be added.
getRecievedRequests()	Public	object	-	The user must have an owner account	Requests which are sent to show an interest on this owner's cars will be returned.
getSentRequests()	Public	object	-	The user must have an renter account	Every request sent by the renter will be returned
handleRequest()	Public	object	-	The user must have an owner account	The user Info will be returned
cancelReqeust()	Public	void	Req. id	The user must have an renter account	The request will be withdrawn.

Table 6: Request Controller Operation Description

4.4 Rent Controller

Rent Controller
+rateARent() +getMyRents() +getOngoingRents() +getCompletedRents()

Table 7: Rent Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
rateArent()	Public	void	-	There is a completed rent which is unrated rate.	The user's rating will be updated indirectly
getMyRents()	Public	object	-	The user must have an owner account	All rents of the user will be returned
getOngoingRents ()	Public	object	-	The user must have an renter account	All ongoing rents of the user will be returned
getCompletedRents ()	Public	object	-	The user must have an owner account	All completed rents of the user will be returned

Table 8: Rent Operation Description

4.5 Booking Controller

Booking Controller
+cancelBooking() +getAllBooking() +getMyBooking()

Table 9: Booking Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
cancelBooking()	Public	void	Booking id	The user must have an owner account	The booking will be cancelled and the car will be available again for that time span.
getAllBooking()	Public	object	-	The user must have an owner account	Bookings for owner's cars will be returned.
getMyBooking()	Public	object	-	The user must have an renter account	Bookings set by the renter will be returned

Table 10: Booking controller operation Description

4.6 Availability Controller

Availability Controller
+ IsAvailable(start,end) + setUnavailable(start,end) + monthAvailability()

Table 11: Availability Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition

isAvailable()	Public	void	startDate: Date endDate: Date	The user must have an renter account	It will return the availability of the car for that time span.
setUnavailable()	Public	void	startDate: Date endDate: Date	The user must have an renter account	All requests for that time span will be declined.
monthAvailability()	Public	list	-month	The user must have an renter account	Bookings for owner's cars will be returned.

Table 12: Availability Controller Operation Description

4.7 Notification Controller

Notification Controller
+getMyNotif() +setRead() +AddNotification()

Table 13: Notification Controller

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
getMyNotifications()	Public	void	-	The user has to be logged in and at the notifications page.	It will return all notifications which are directed toward this user.
setRead()	Public	void	-notif. Id	The user has to be logged in and at the notifications page.	The notification will be updated as read.
AddNotification()	Public	void	-	The user has to be logged in.	A notification will be produced

Table 14: Notification Controller Operation Description

References

- Fowler, M. (1997). *UML Distilled: A brief guide to the standard object modeling language*.
- Sommerville, I. (n.d.). *Software Engineering* (9th ed.).
- freeCodeCamp. (n.d.). *UML Diagrams Full Course* [Video]. Youtube. Retrieved December 9, 2023, from https://www.youtube.com/watch?v=WnMQ8HlmeXc&t=1080s&pp=ygUDdW1s