Design Document

for

Fleet

1.0

Prepared by

Group :11 **Group Name: Null Pointers**

Habeeb Ramith Kumar	230432	hramith23@iitk.ac.in
Reddi Pallavi	230850	rpallavi23@iitk.ac.in
Pasala Bosu Akil Teja	230742	pbosuateja23@iitk.ac.in
Shashi Bhidodiya	230956	shashib23@iitk.ac.in
Manam Amara Gayathri	230624	mamara23@iitk.ac.in
Sai Prabhav	230060	addulasr23@iitk.ac.in
Koneti Karthik	230568	konetik23@iitk.ac.in
Poorvie Sadagopan	230759	poorvies23@iitk.ac.in
Pittala Sruthi	230751	sruthip23@iitk.ac.in
Jyothika Seru	230946	serujy23@iitk.ac.in

Course: CS253

Mentor TA: Souvik Mukherjee

Date: 07/02/2025

Contents II

Revisions III

1 Context Design 1

1.1 Context model 1
1.2 human Interface Design 2

2 Architecture design 12

3 object-oriented design 14

3.1 Use case diagram 14
3.2 Class diagram 17
3.3 sequence diagram 18
3.4 State diagram 22

4 Project Plan 28

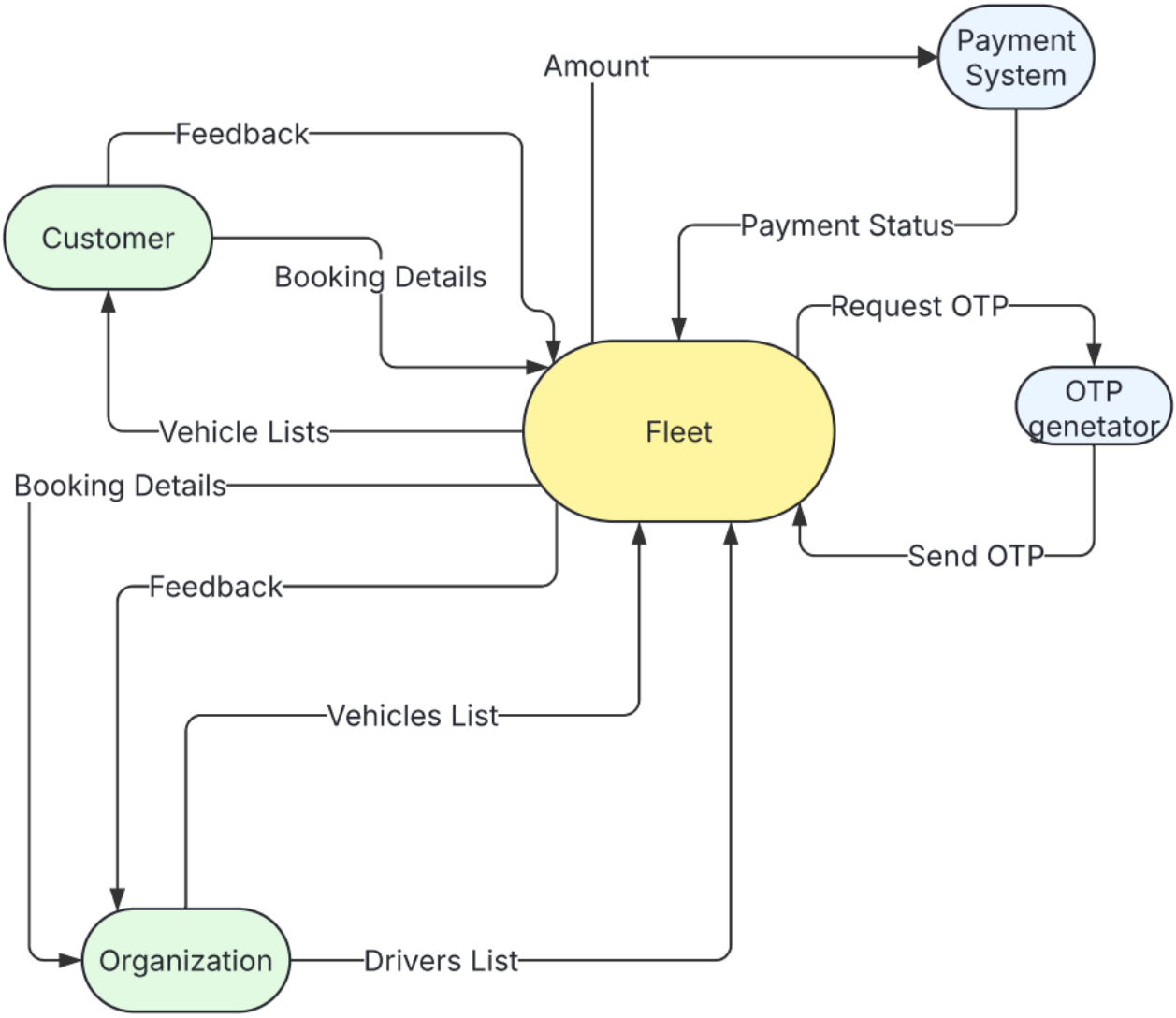
Appendix A - Group Log 29

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Habeeb Ramith Kumar Reddi Pallavi Pasala Bosu Akil Teja Shashi Bhidodiya Manam Amara Gayathri Saiprabhav Koneti Karthik Poorvie Sadagopan Pittala Sruti Jyothika Seru	First version of Design Document	07/02/25

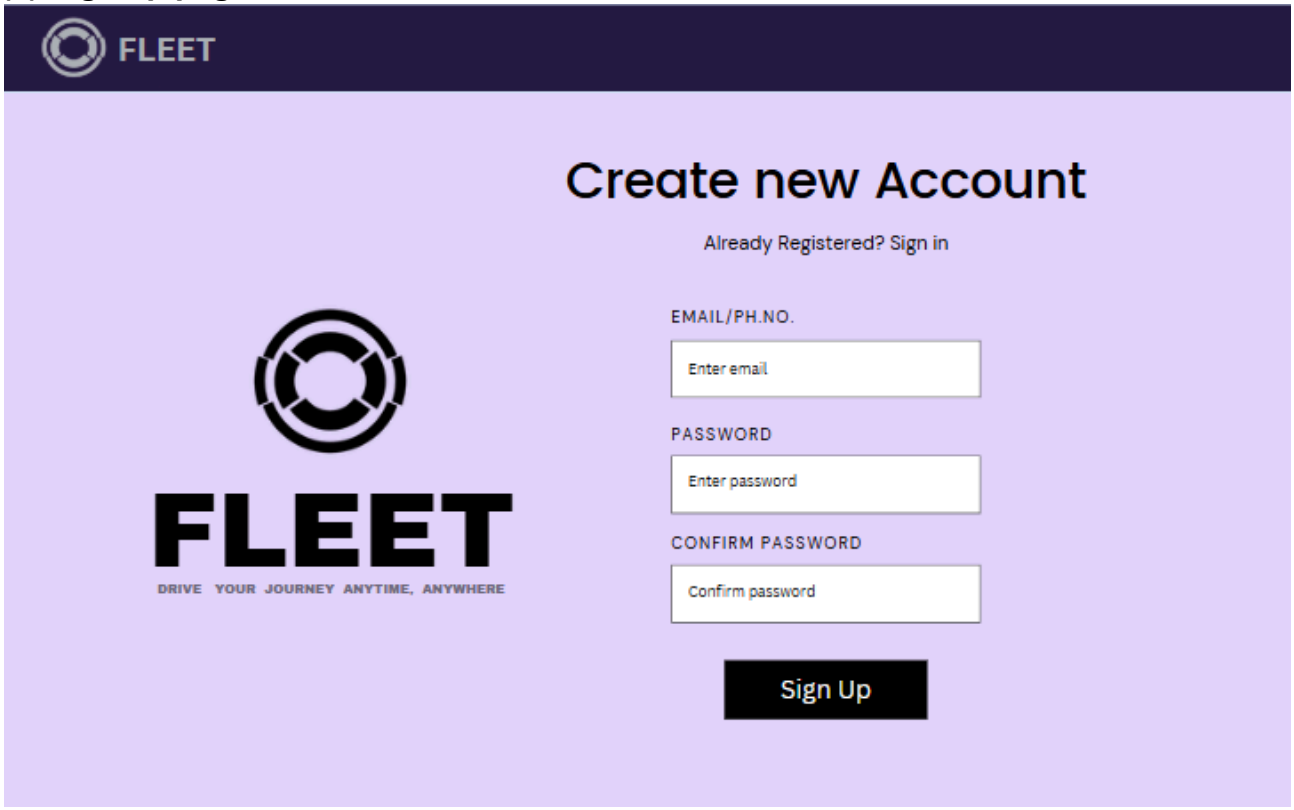
1 Context Design

1.1 Context Model

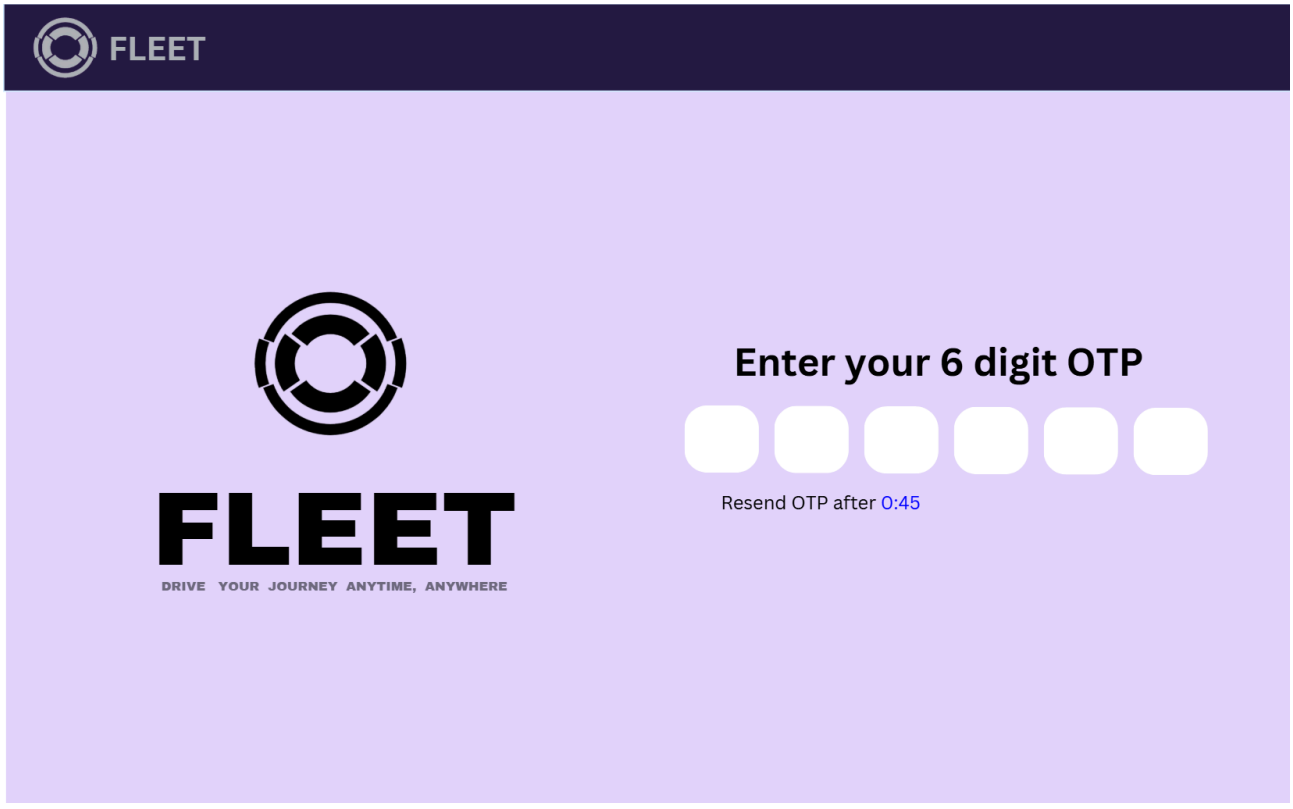


1.2 Human Interface Design

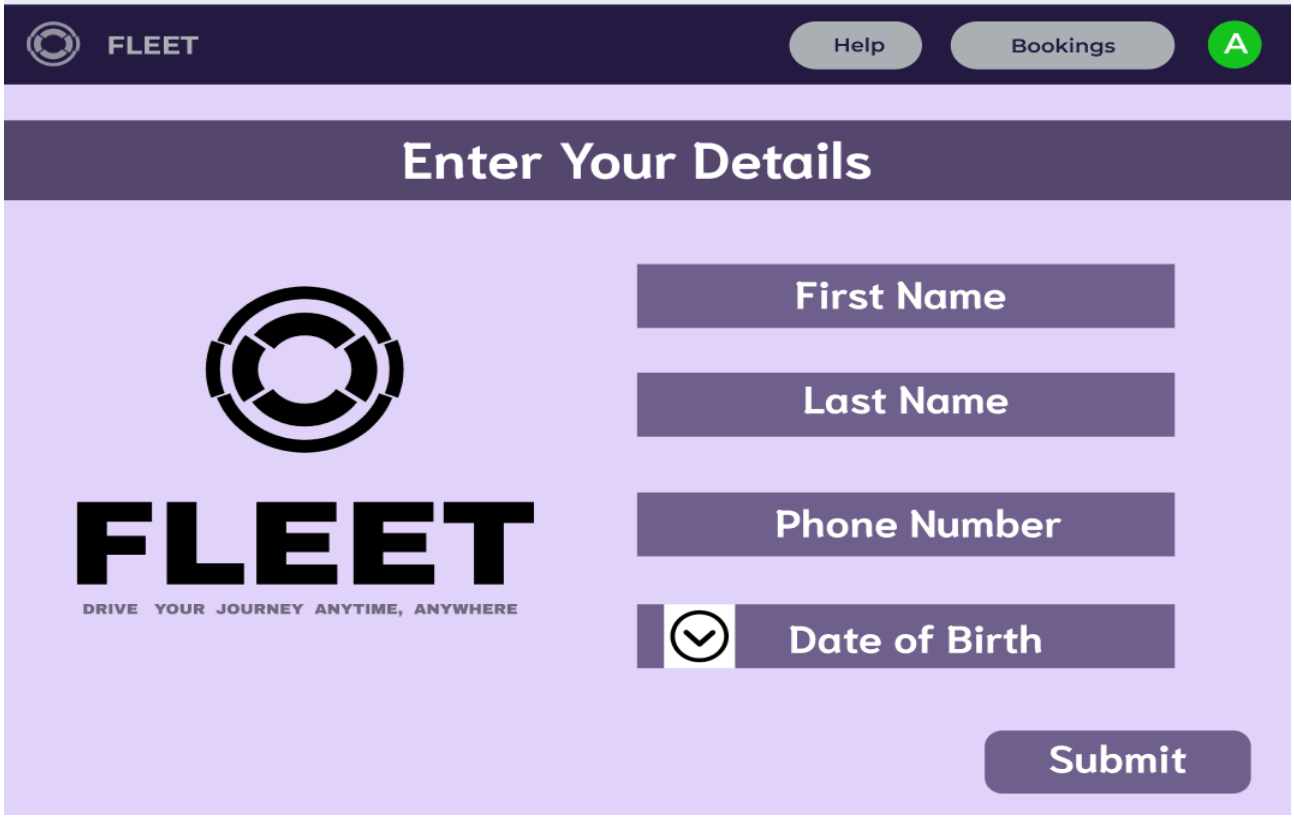
(a) Sign up page



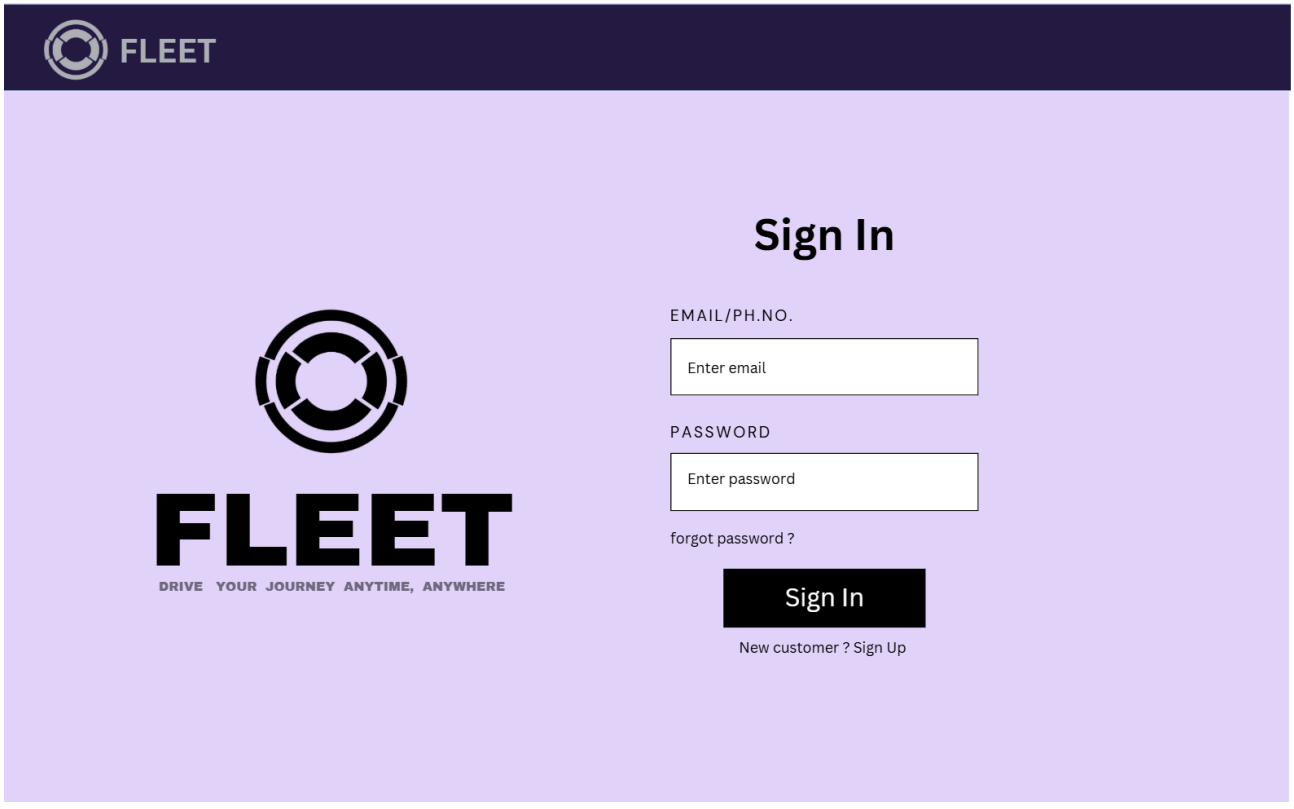
(b) OTP for sign up



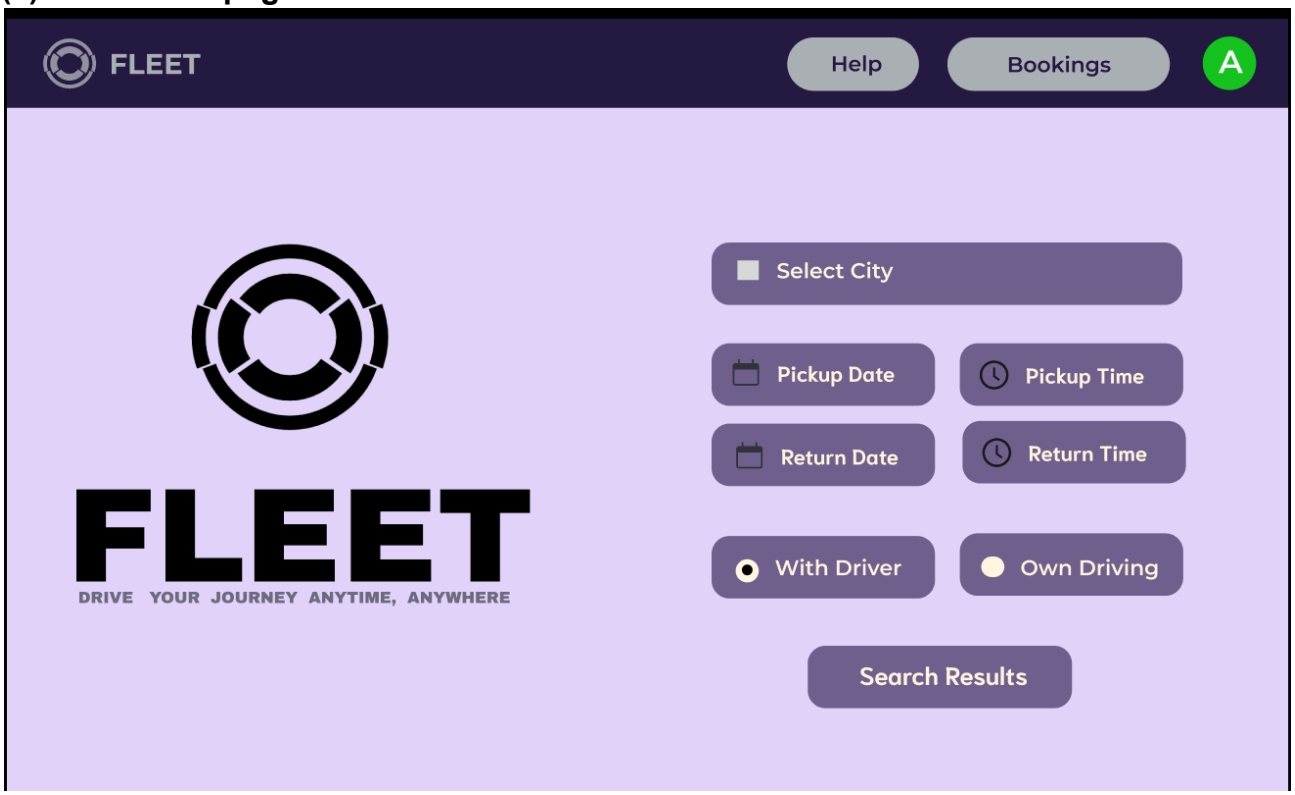
(c) Enter your details page



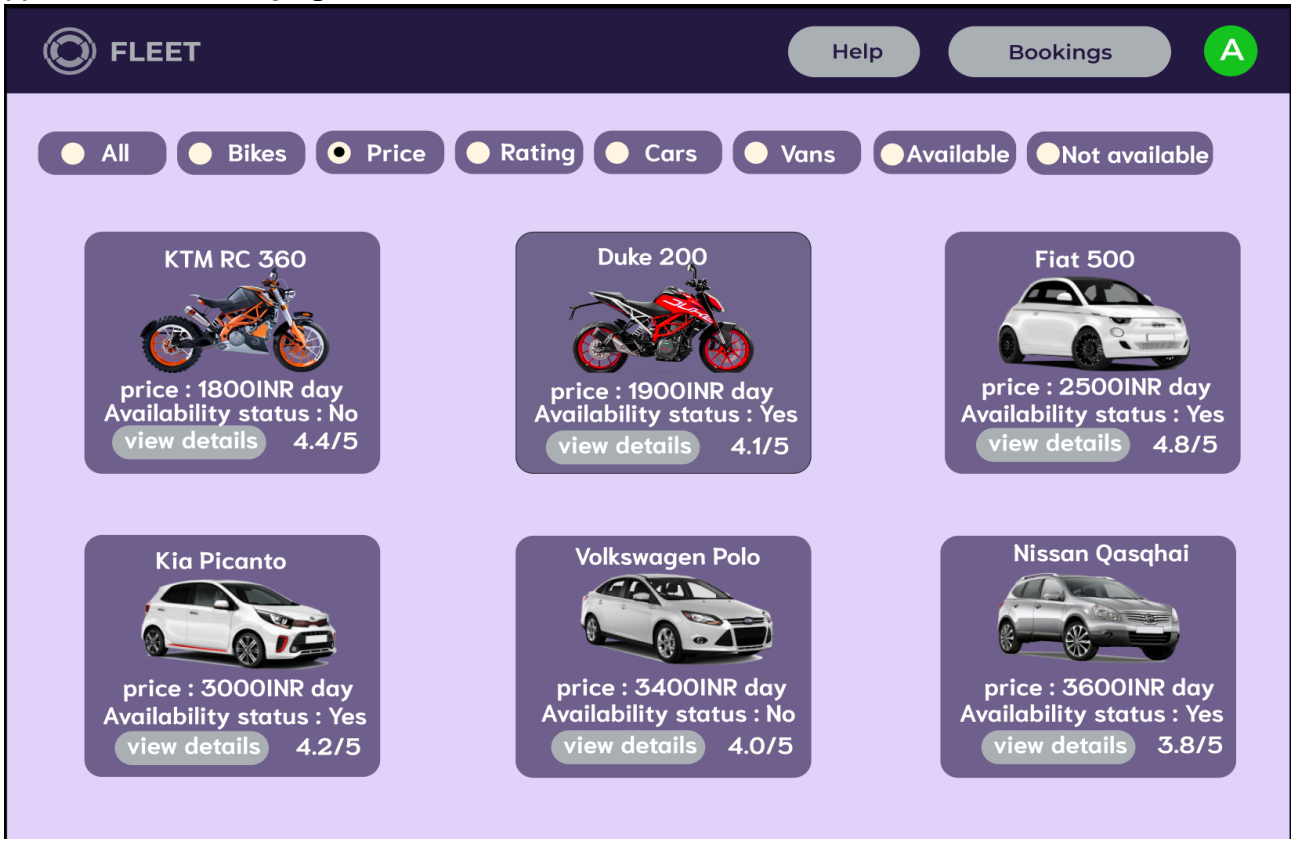
(d) Sign in page



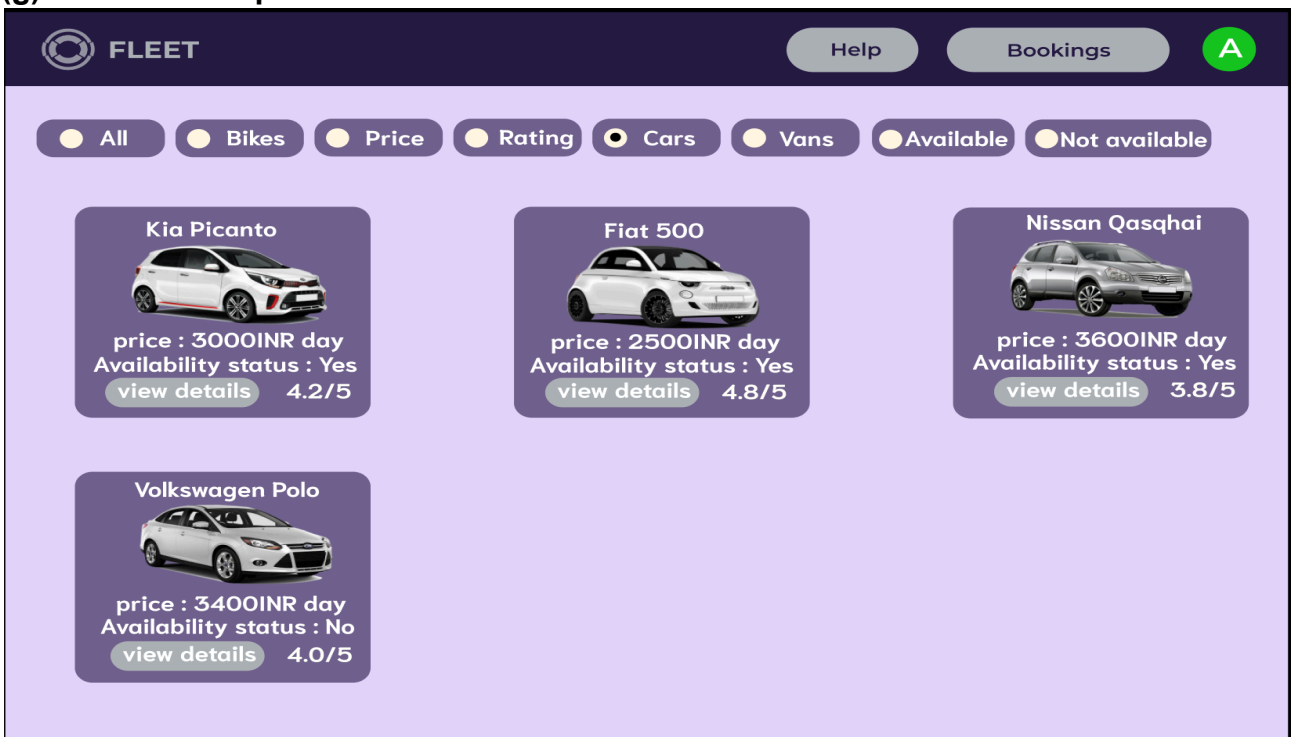
(e) User home page



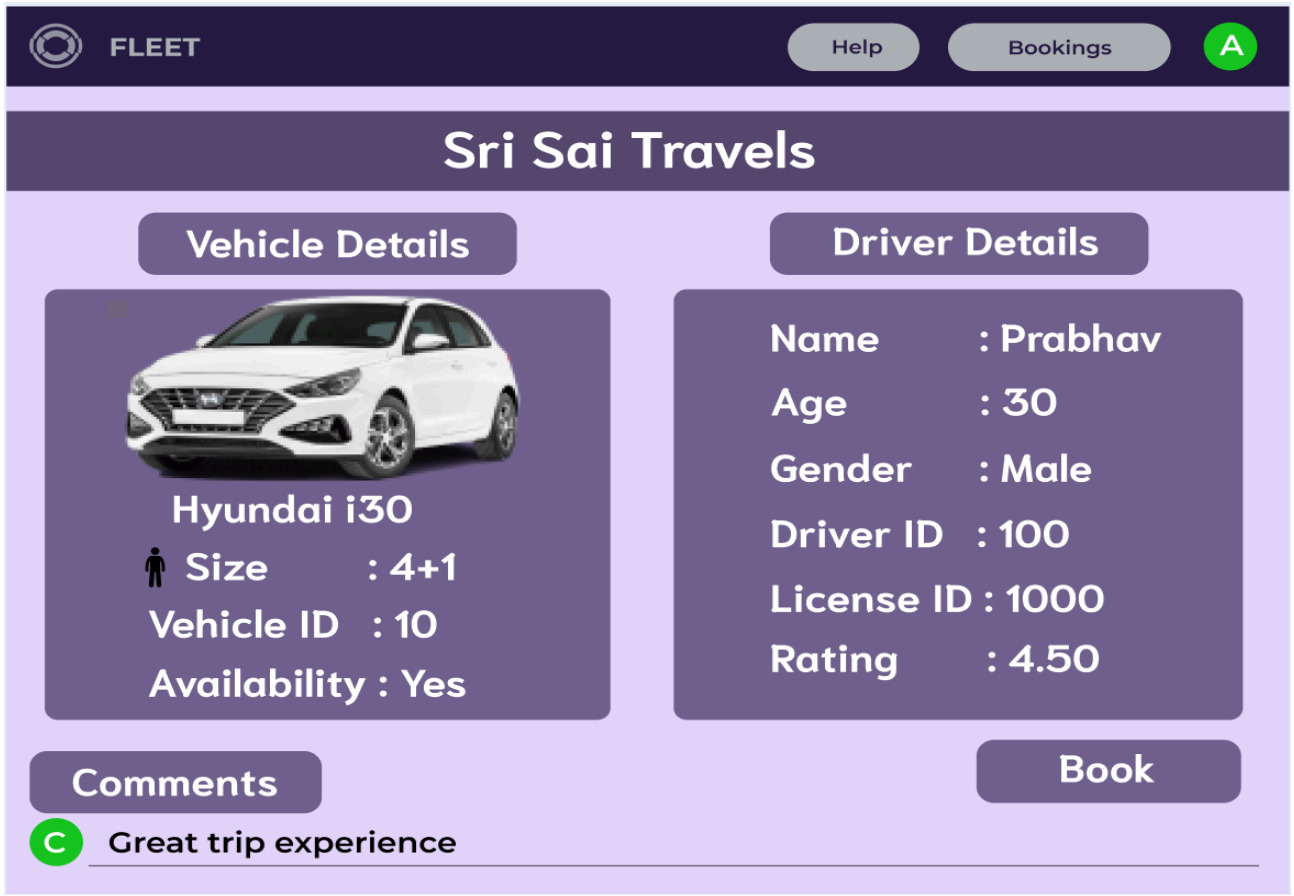
(f) Search results page



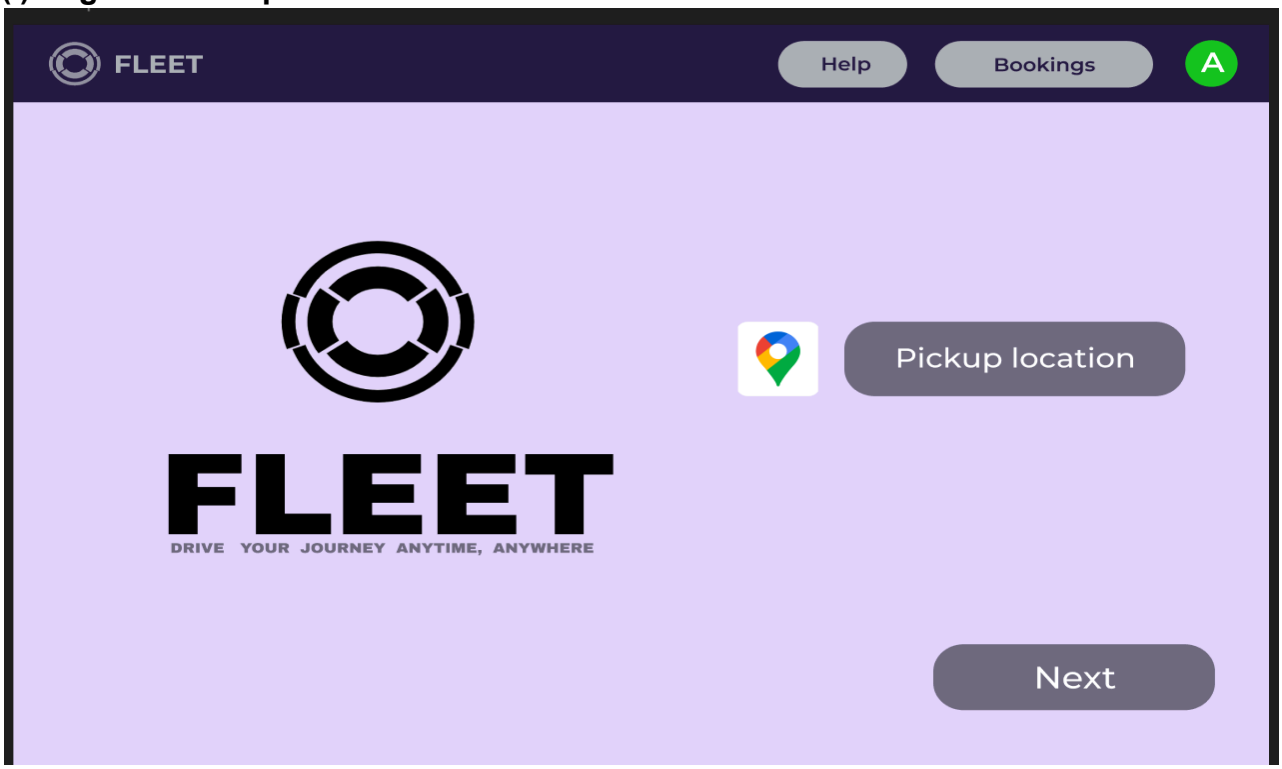
(g) Use of filter option



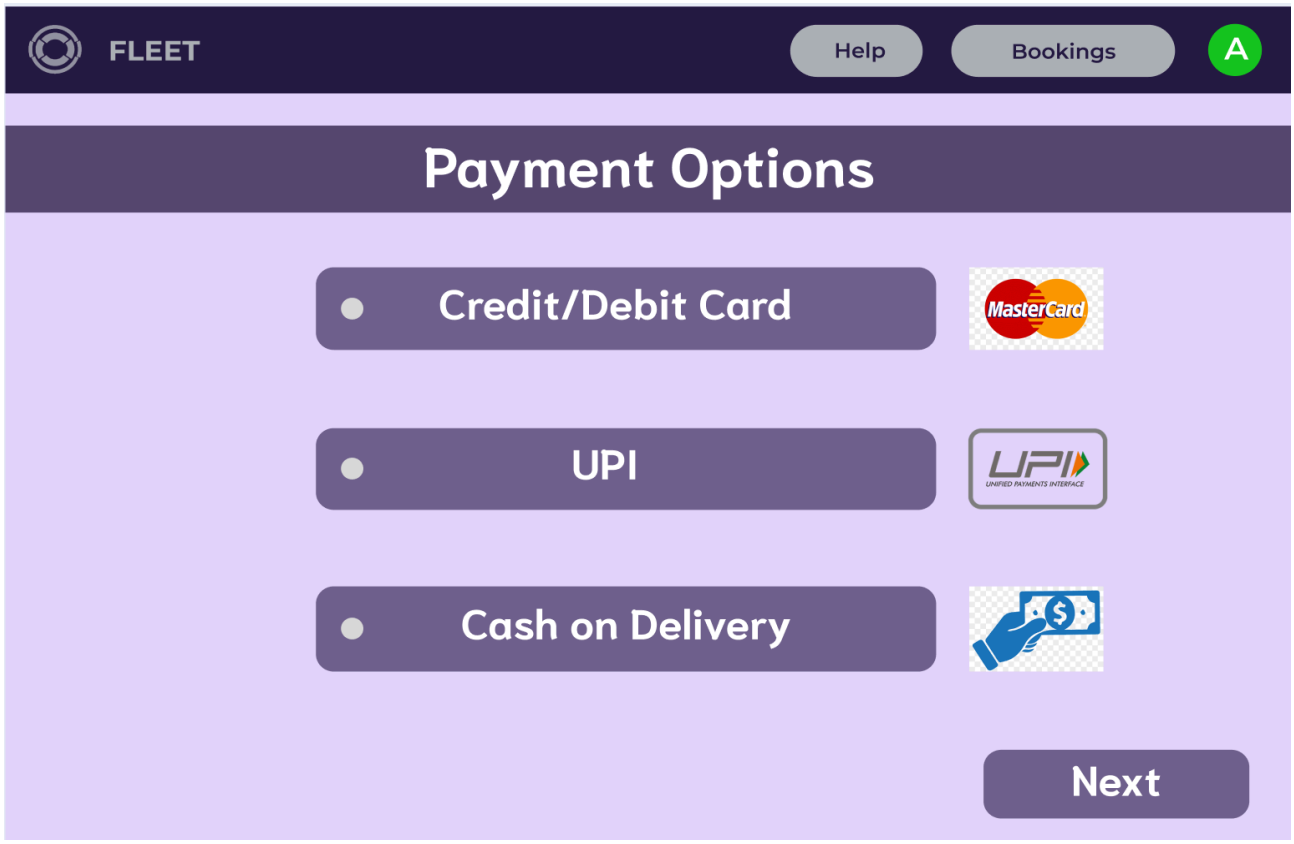
(h) View details page



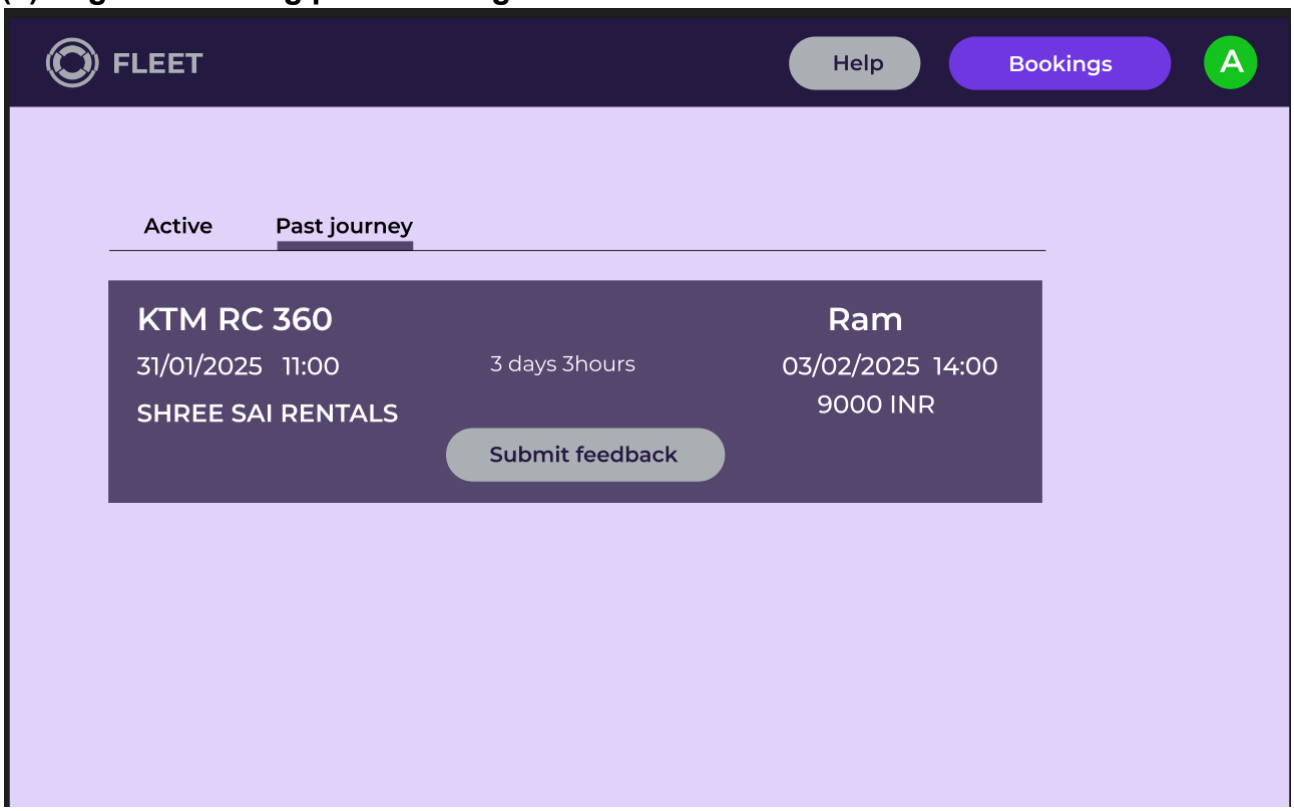
(i) Page for Pickup address



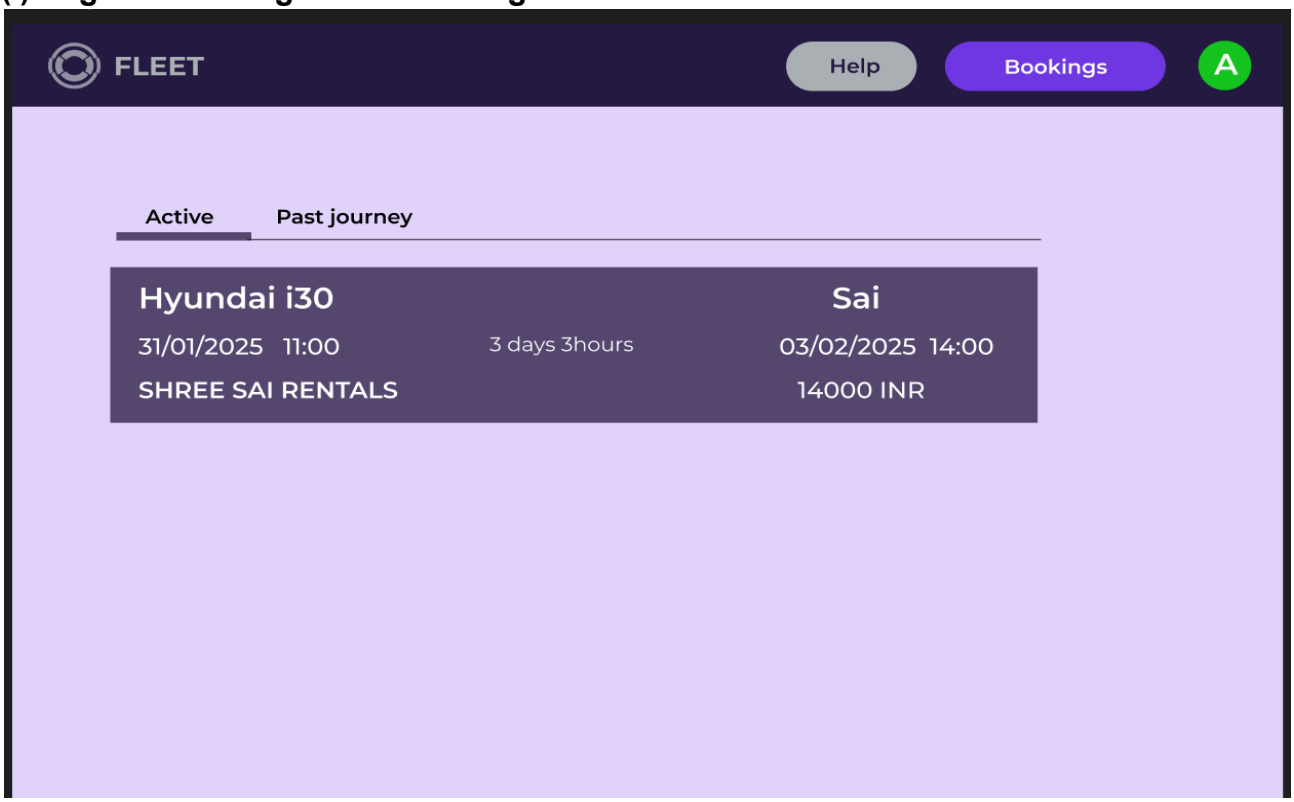
(j) Page for payment options



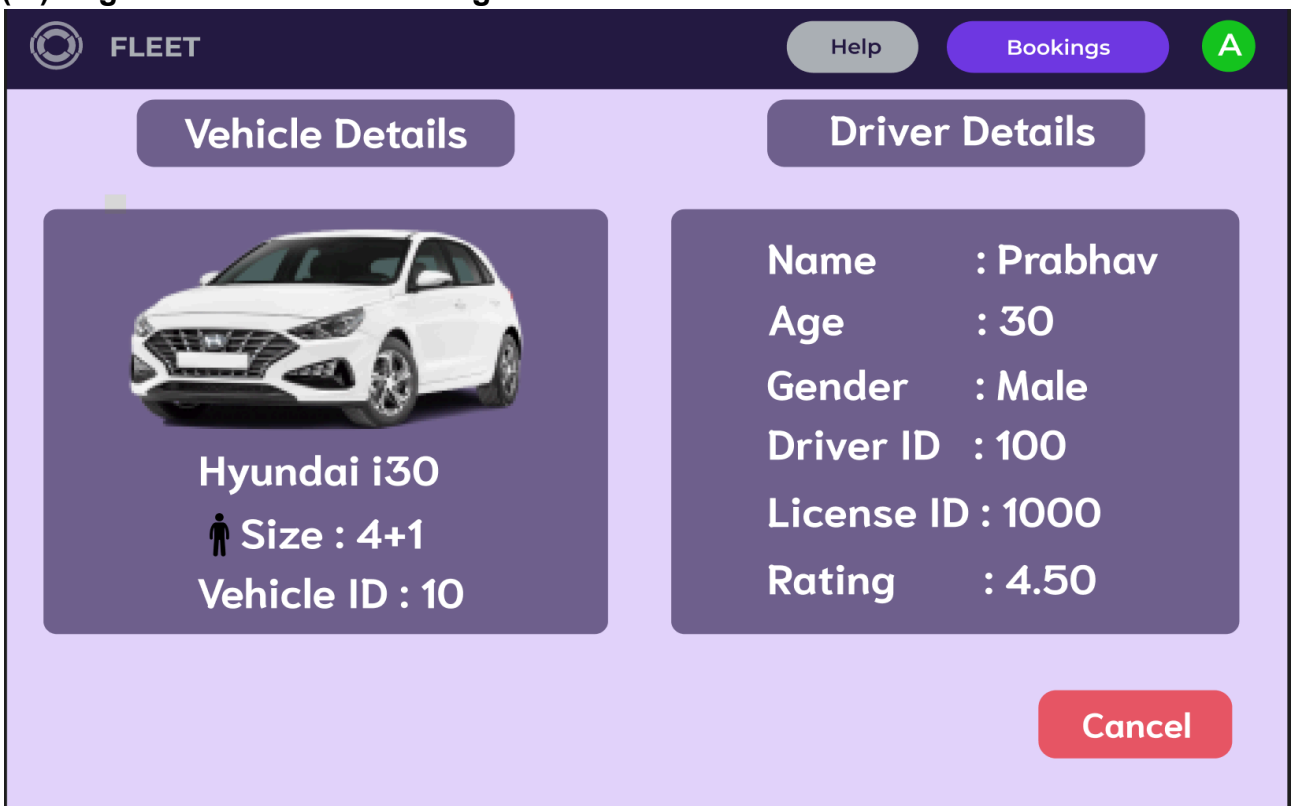
(k) Page for viewing past bookings



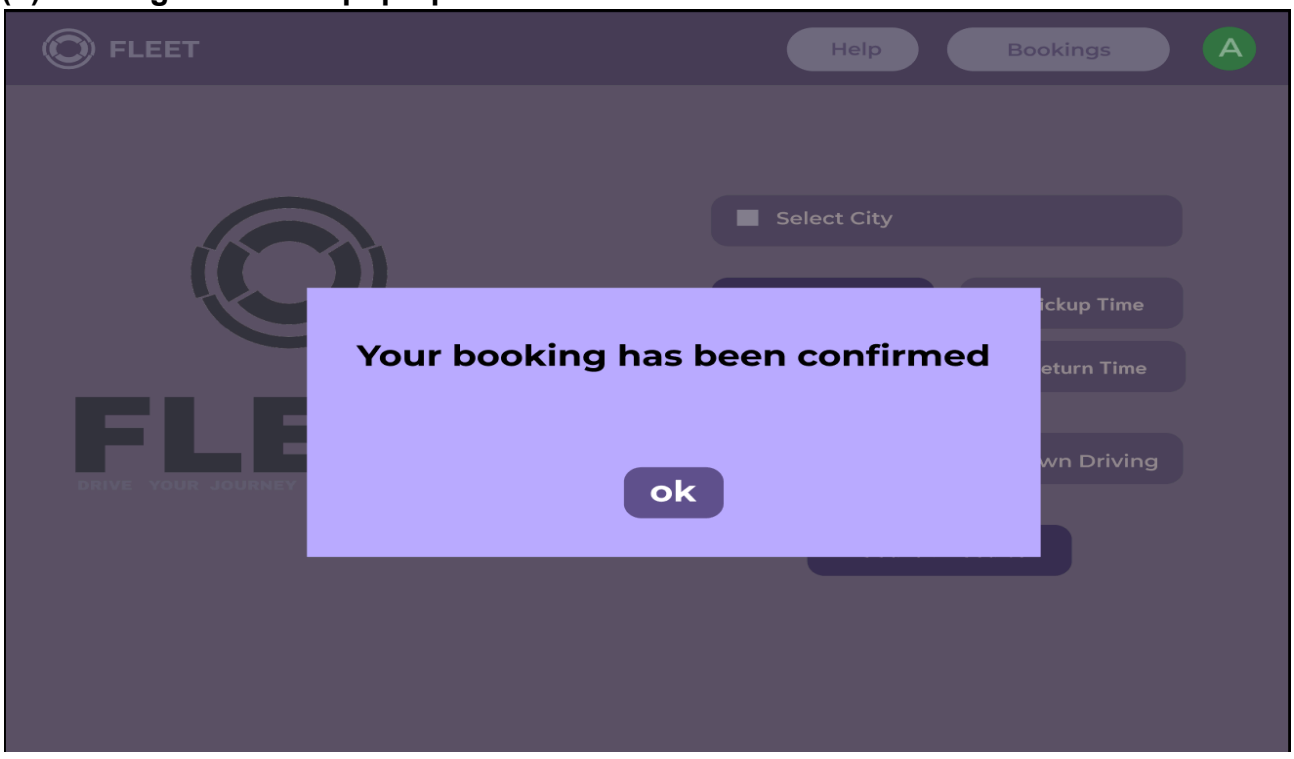
(l) Page for viewing active bookings



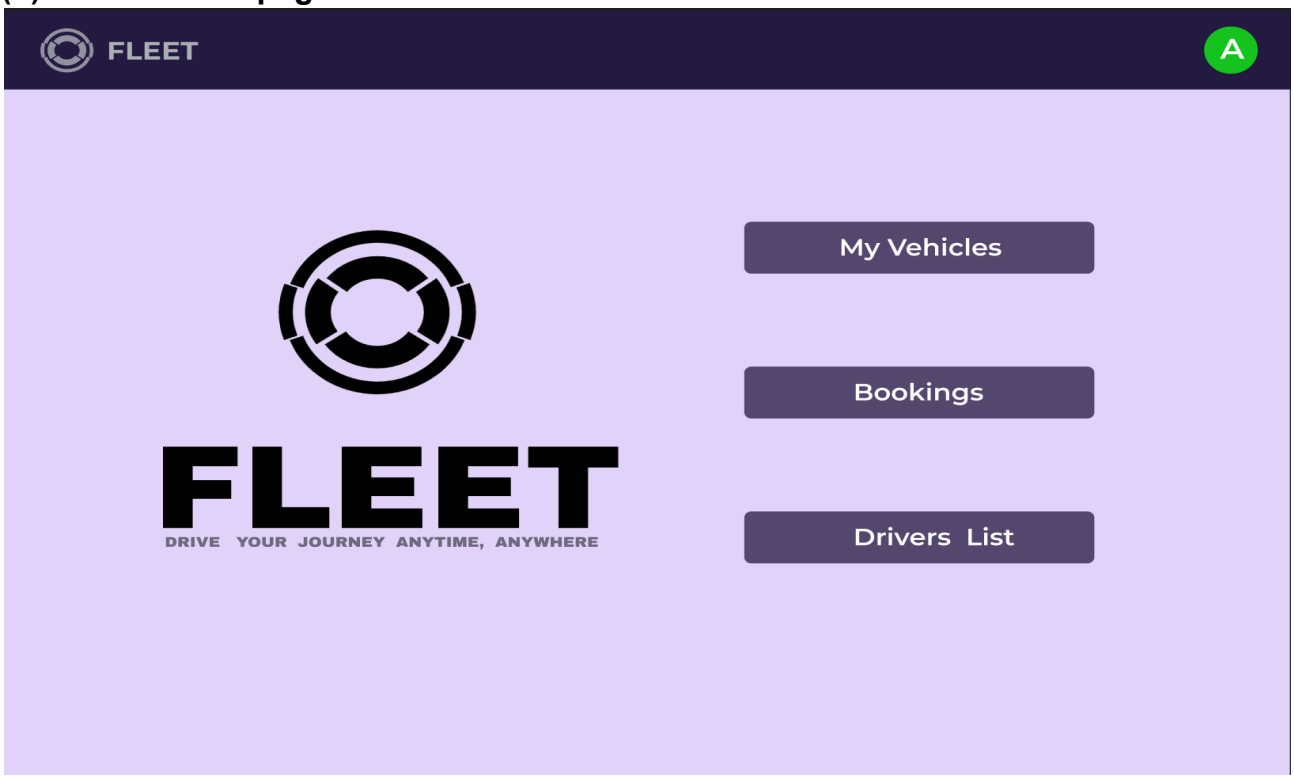
(m) Page to view Active booking details



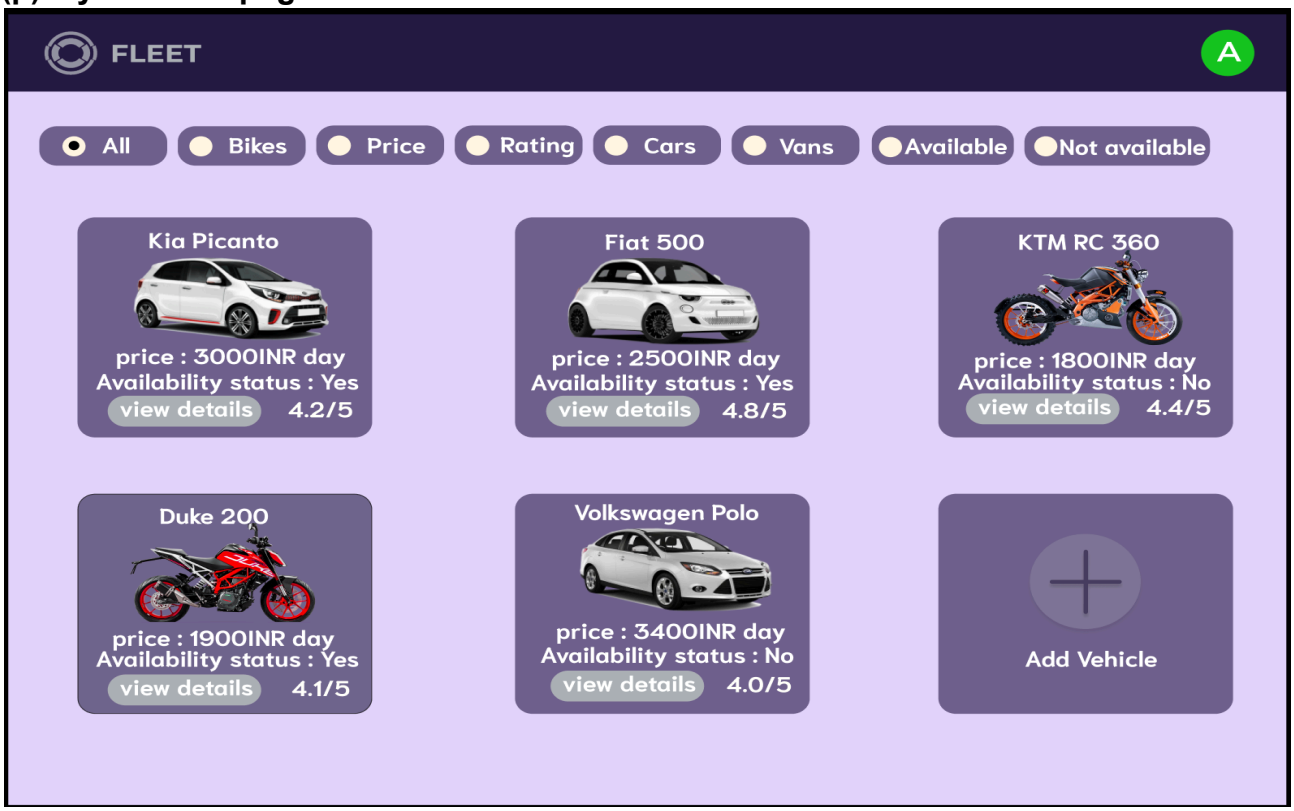
(n)Booking confirmed pop-up



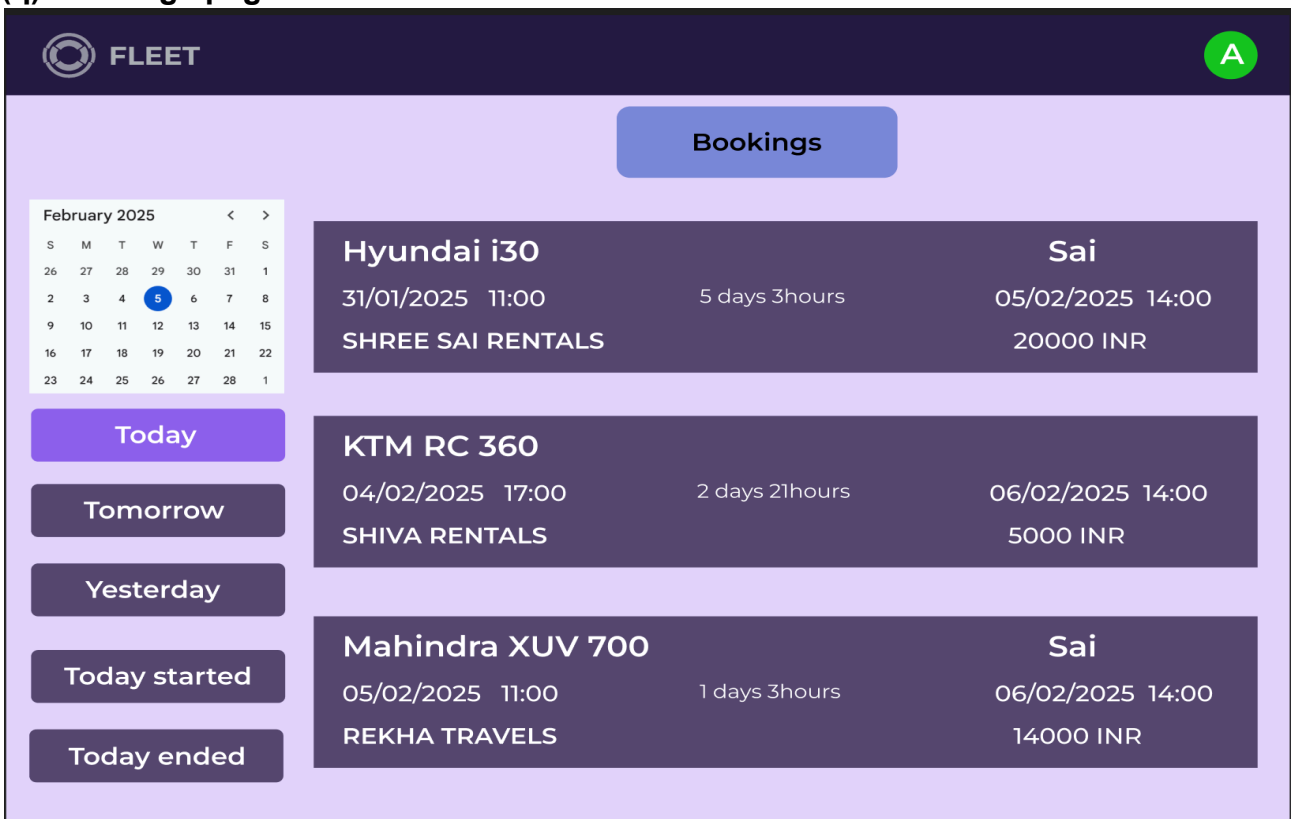
(o) Admin Home page



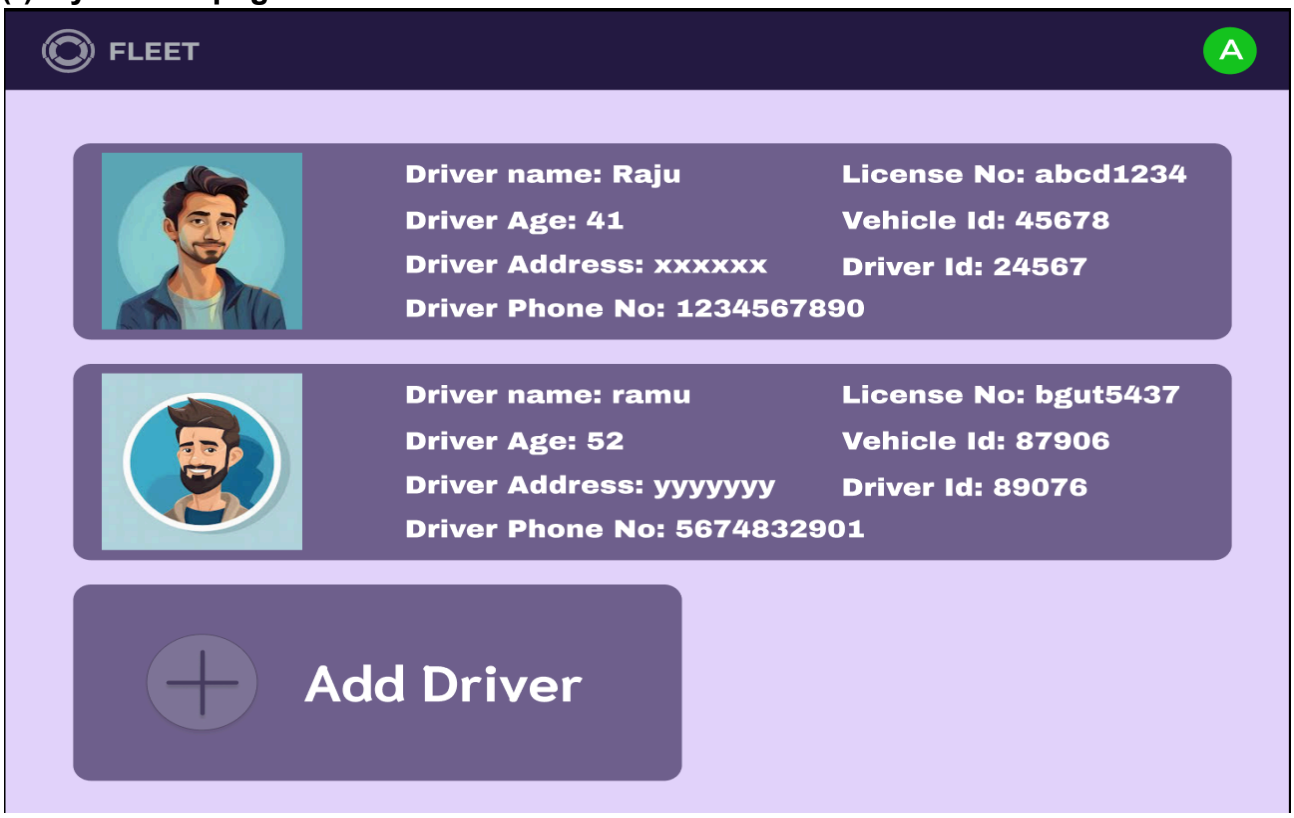
(p) My vehicles page for Admin



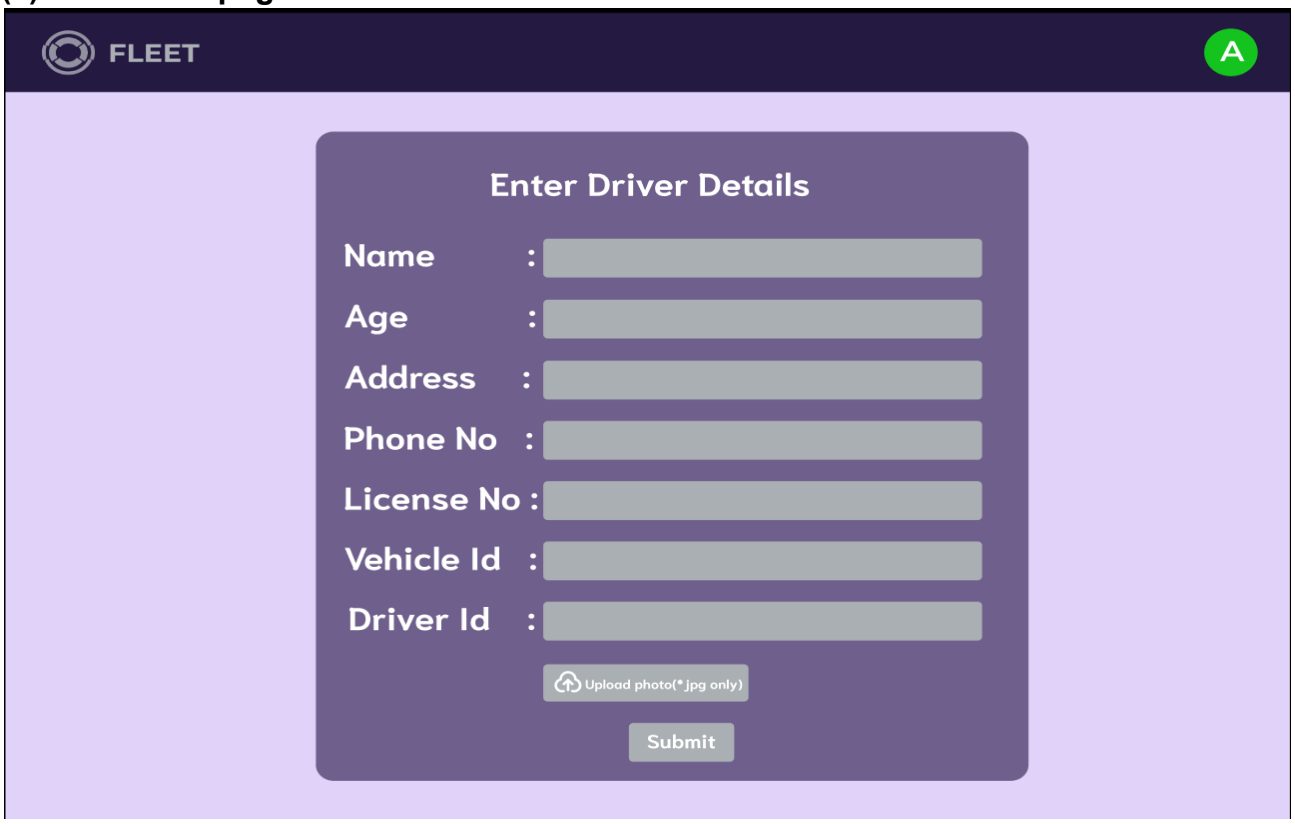
(q) Bookings page for Admin



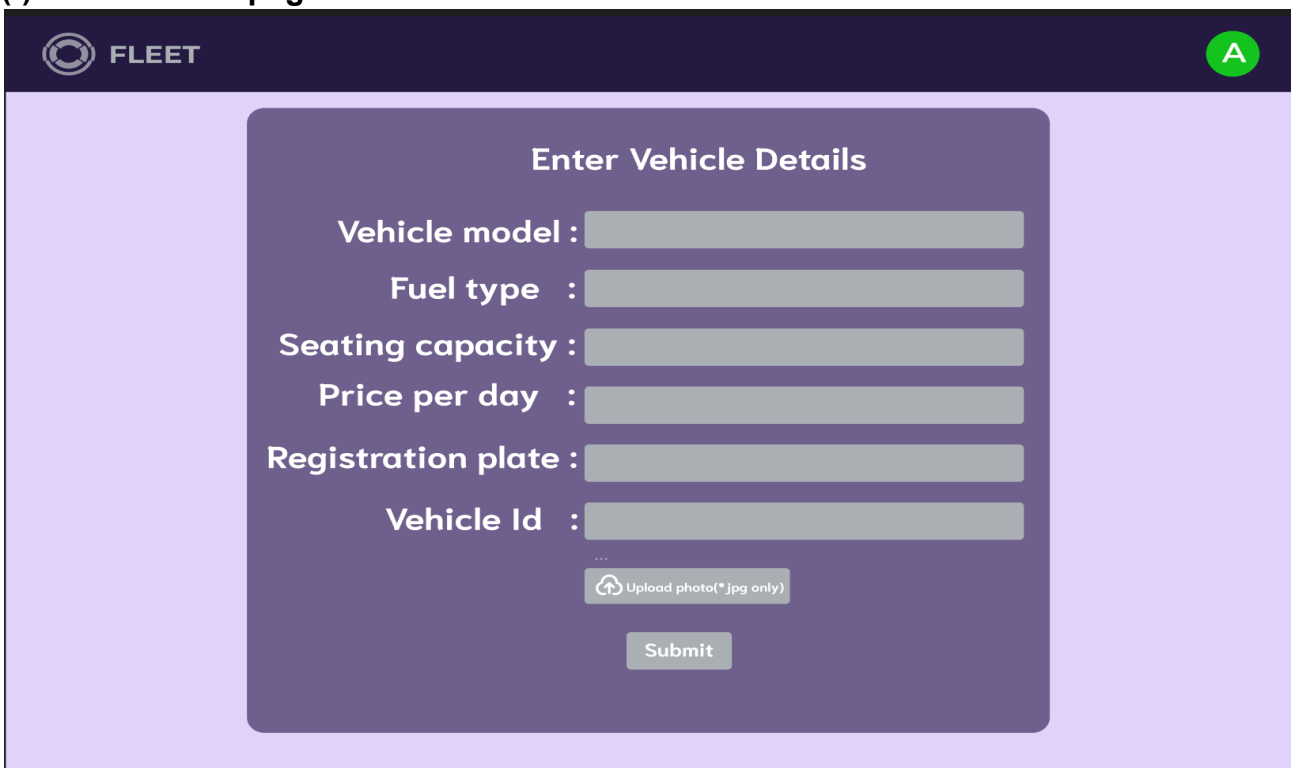
(r) My drivers page for Admin



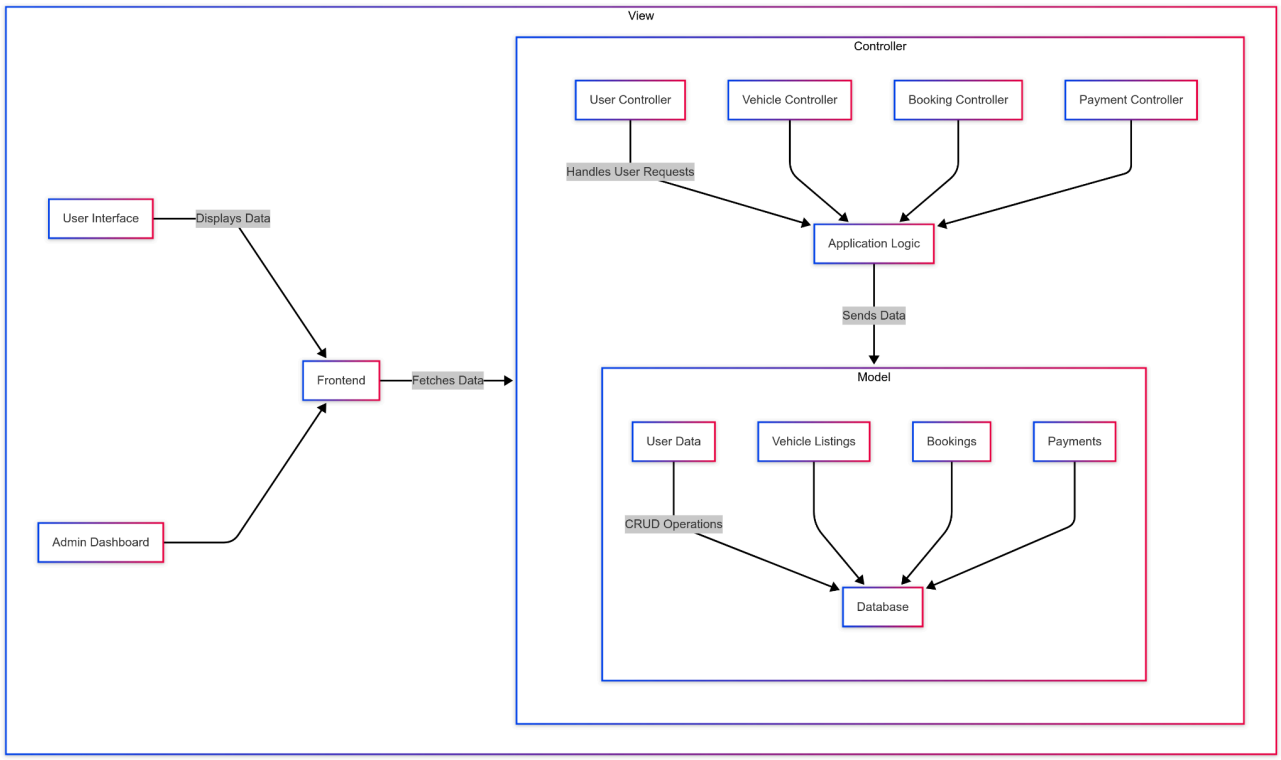
(s) Add driver page for Admin



(t) Add vehicles page for Admin



2 Architecture Design



Fleet is a web-based application implementing a **Model-View-Controller model**.

Why Are We Using MVC?

We chose Model-View-Controller (MVC) for Fleet because it provides a structured and modular way to develop the application. By separating concerns between the Model (business logic & database management), View (user interface & presentation), and Controller (handles user inputs and requests), we ensure that the system remains scalable, maintainable, and easy to modify. This will help us as we integrate new features, such as improved search filters or AI-based recommendations.

Advantages of Using MVC in Fleet

1. **Modularity & Maintainability** – Code is organized in separate layers, making it easier to debug, update, and scale.
2. **Scalability** – New features like payment integrations, user reviews can be added without modifying the entire codebase.
3. **Parallel Development** – We can work on different parts of the software simultaneously (e.g., frontend team works on UI while backend team manages database operations).
4. **Reusability** – Components like the booking system, authentication, and payment processing can be reused in different parts of the application.

Disadvantages of Using MVC in Fleet

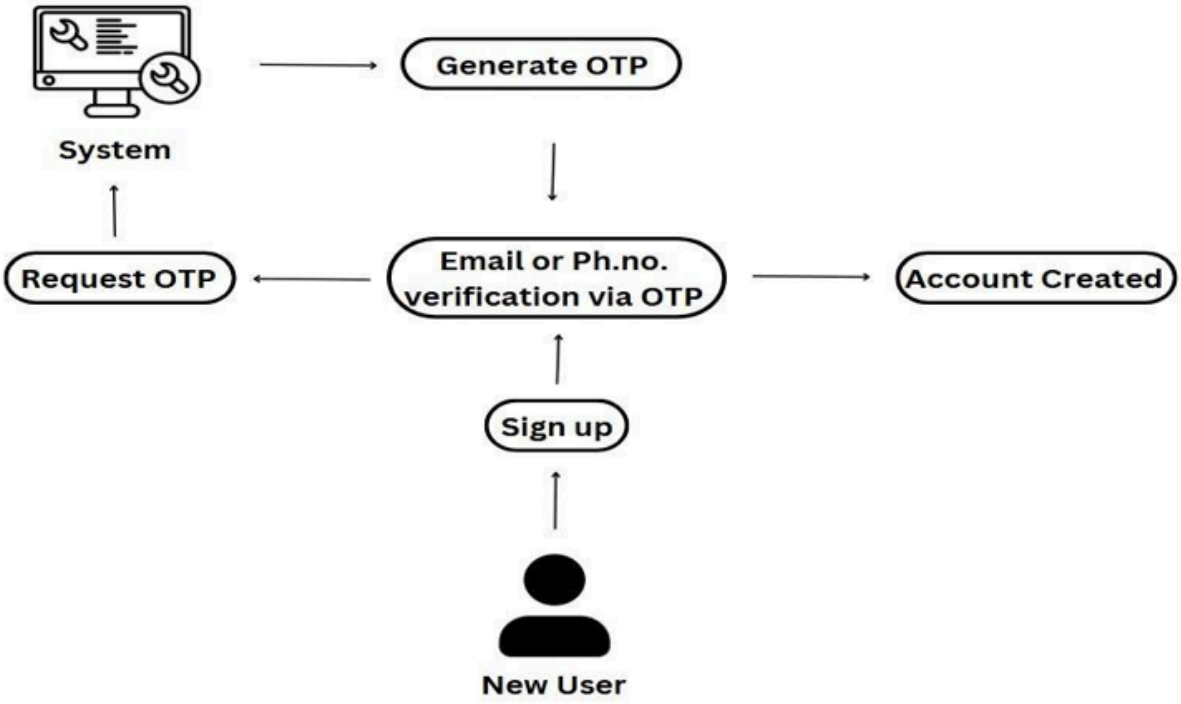
1. For a relatively small application, MVC might introduce unnecessary overhead.
2. While MVC separates concerns, poorly designed controllers might become too dependent on Views, making modifications more difficult.

Non Functional Requirement	Will MVC Meet It?	Why?
Performance	Yes	MVC improves performance by organizing database queries in the Model, preventing inefficient data retrieval in Views. Using caching techniques within the Model layer can further optimize speed.
Security	Yes	MVC enforces separation of concerns, meaning security logic is handled in the Model, reducing exposure to frontend attacks.
Usability	Yes	Since Views are separate from business logic, developers can create clean, user-friendly interfaces without affecting backend logic.
Maintainability	Yes	The modular structure of MVC ensures that features can be updated or replaced without affecting the entire system.
Portability	Yes	The frontend (View) is independent, making it easier to design responsive UIs for different devices.
Interoperability	Yes	MVC allows easy integration of APIs within the Controller and Model, making it flexible for external services.
Reusability	Yes	Components like authentication, booking, and payment logic can be reused in different modules.

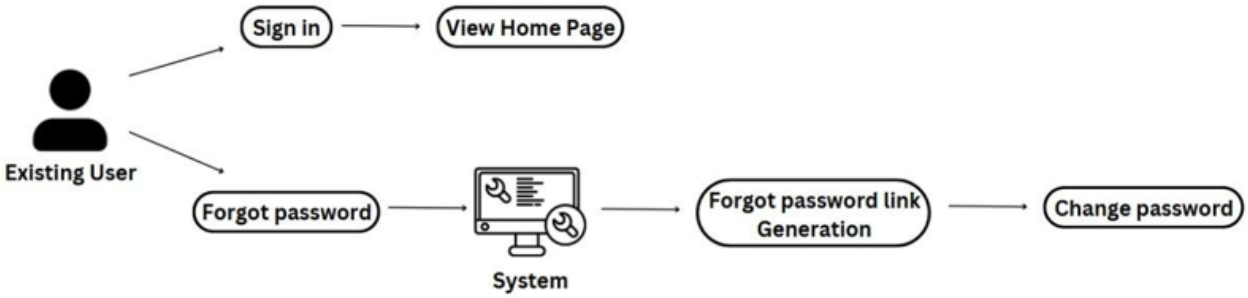
3 Object Oriented Design

3.1 Use CaseDiagrams

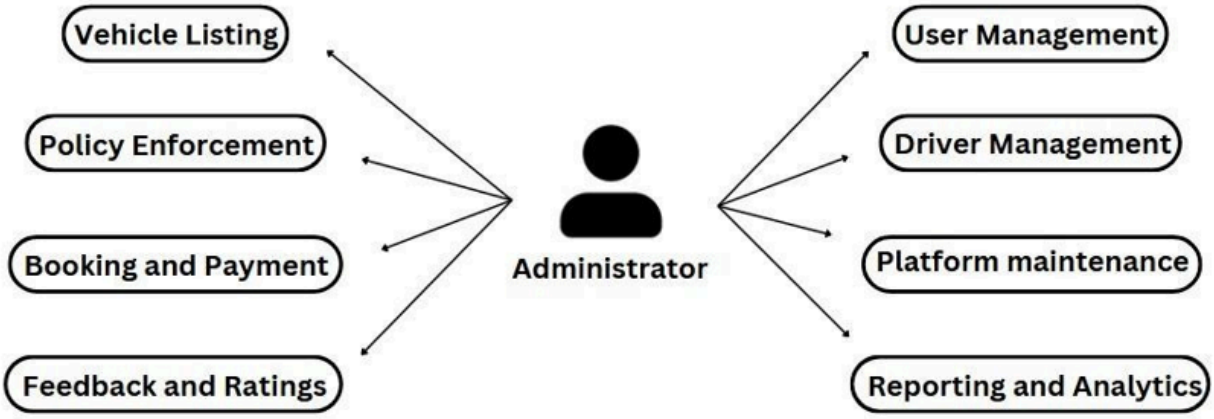
3.1.1 Use Case #1 (UC-1 User Sign Up)



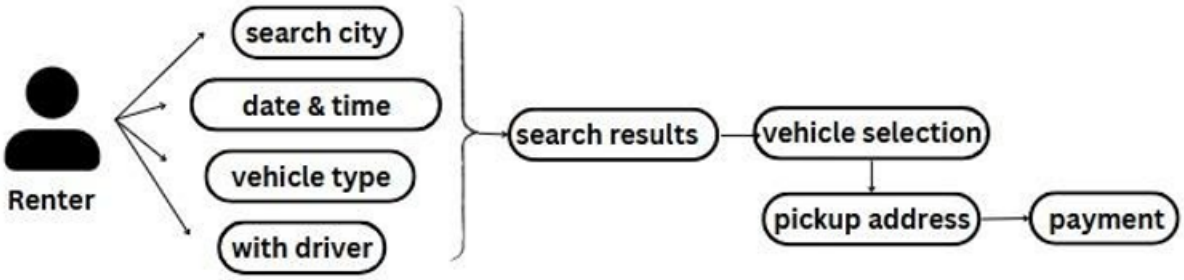
3.1.2 Use Case #2 (UC-2 User Sign In)



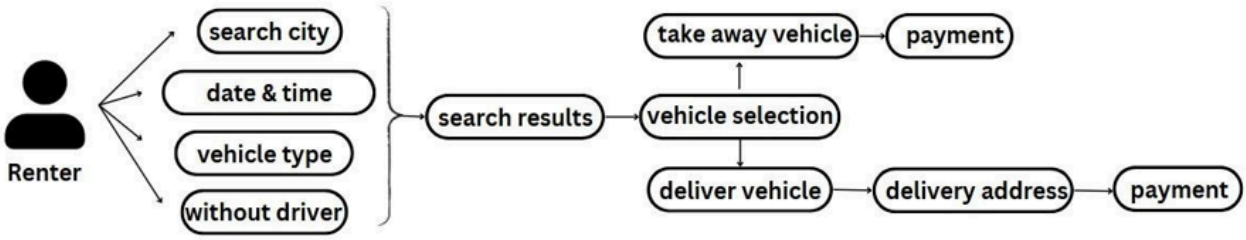
3.1.3 Use Case #3 (UC-3 Administrator)



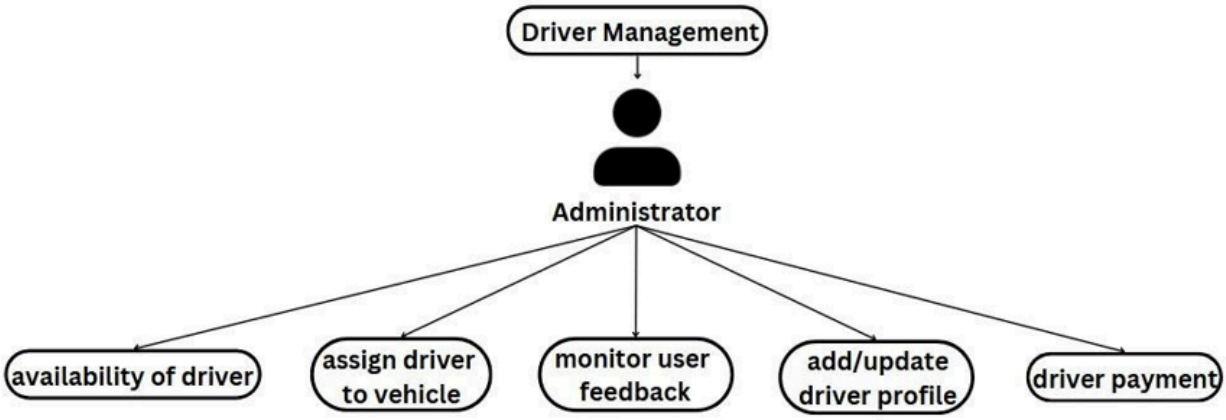
3.1.4 Use Case #4 (UC-4 Renter with driver)



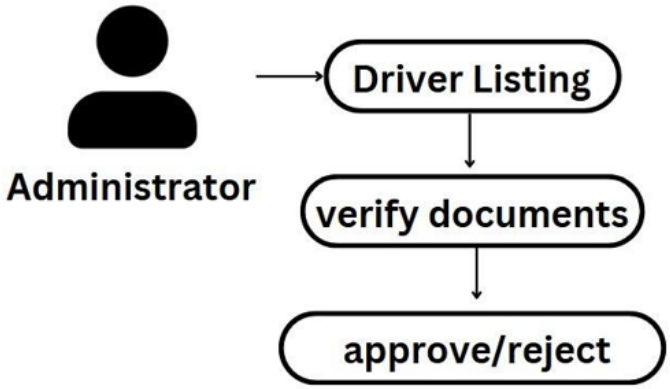
3.1.5 Use Case #5 (UC-5 Renter without driver)



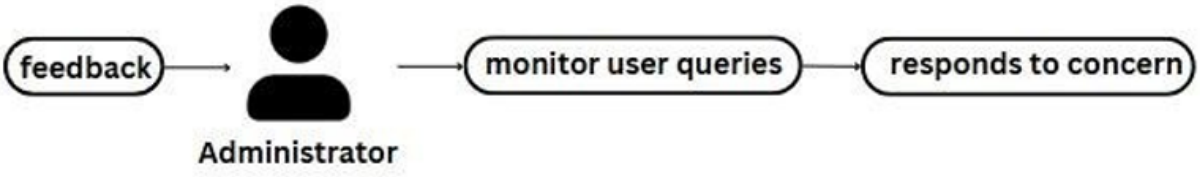
3.1.6 Use Case #6 (UC-6 Administration - Driver Management)



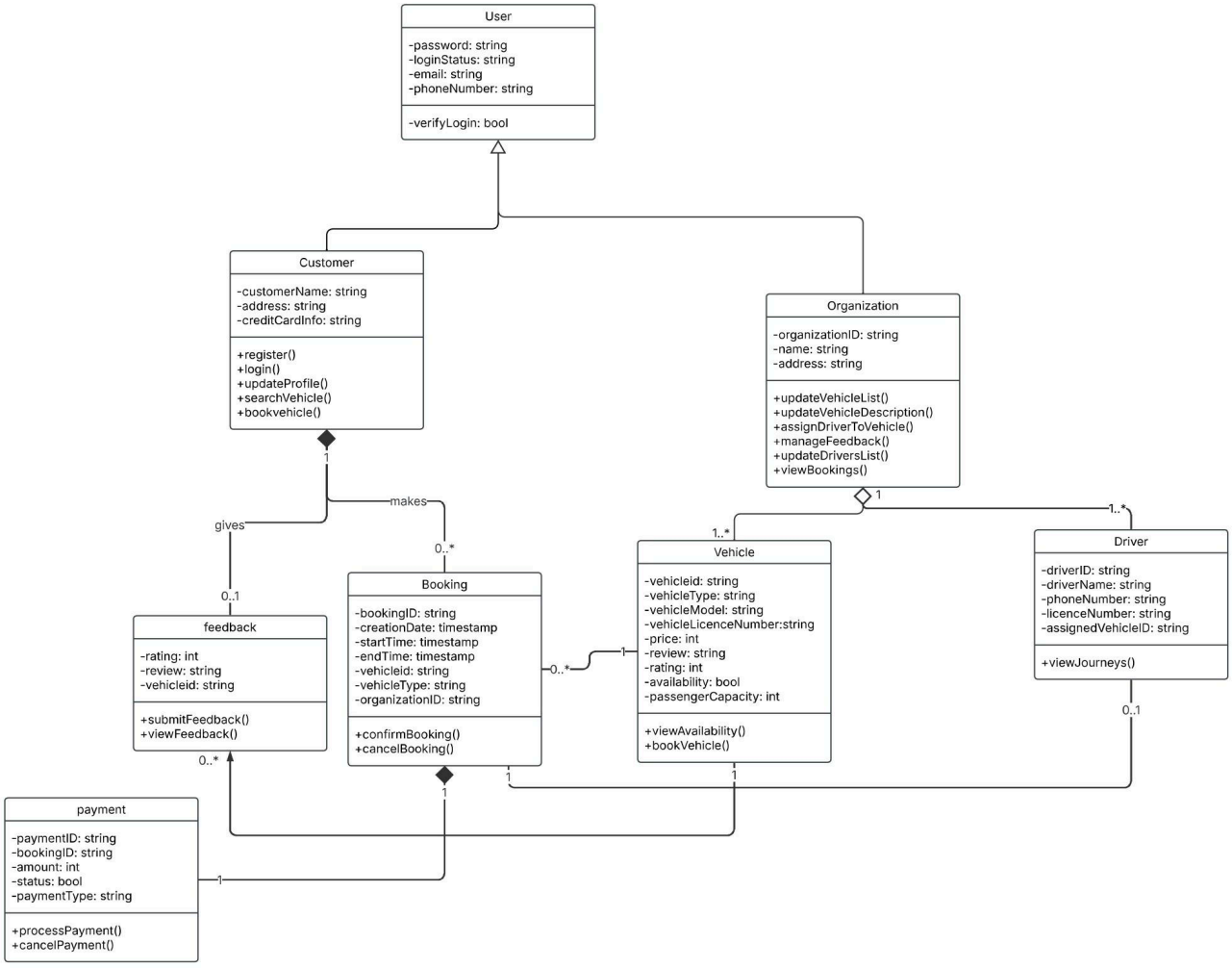
3.1.7 Use Case #7 (UC-7 Administrator - Driver Listing)



3.1.8 Use Case #8 (UC-8 Feedback)

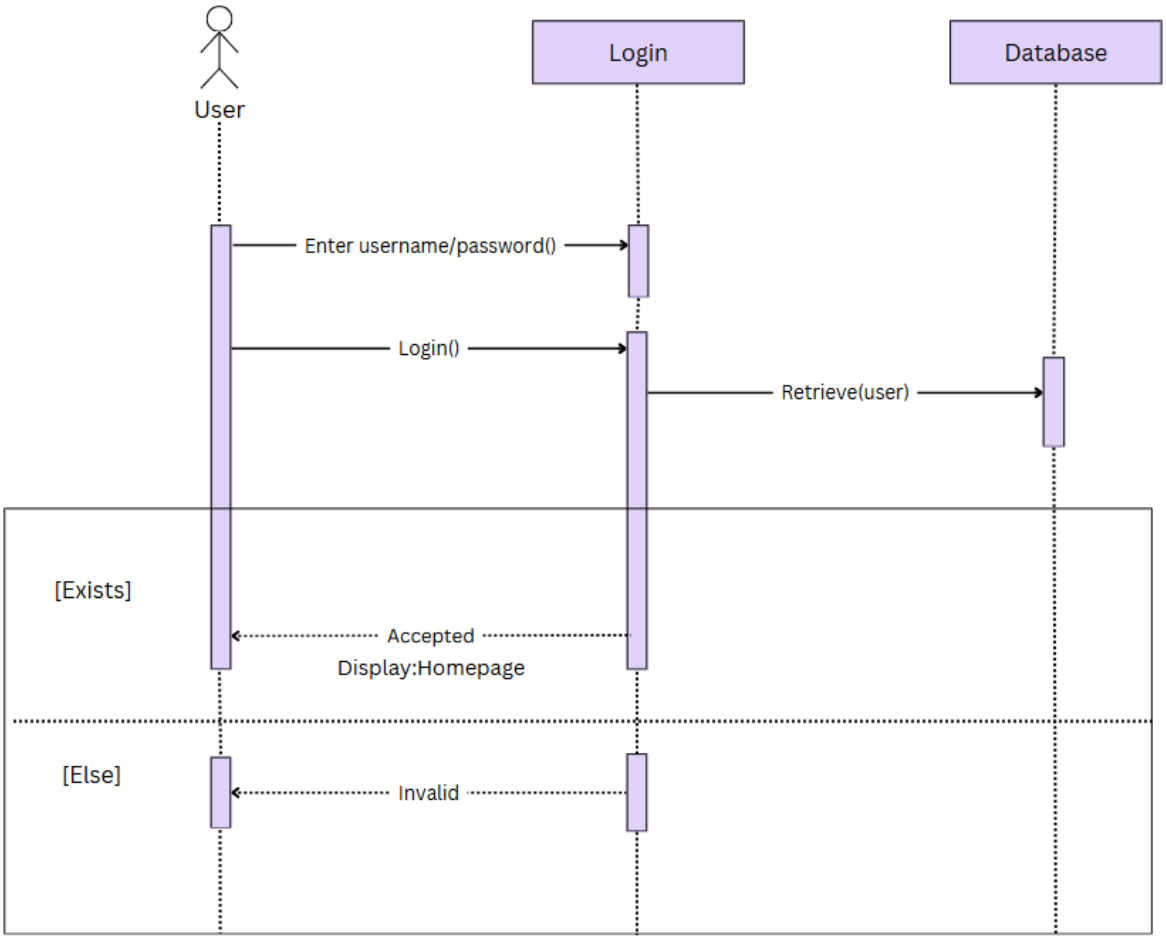


3.2 Class Diagrams

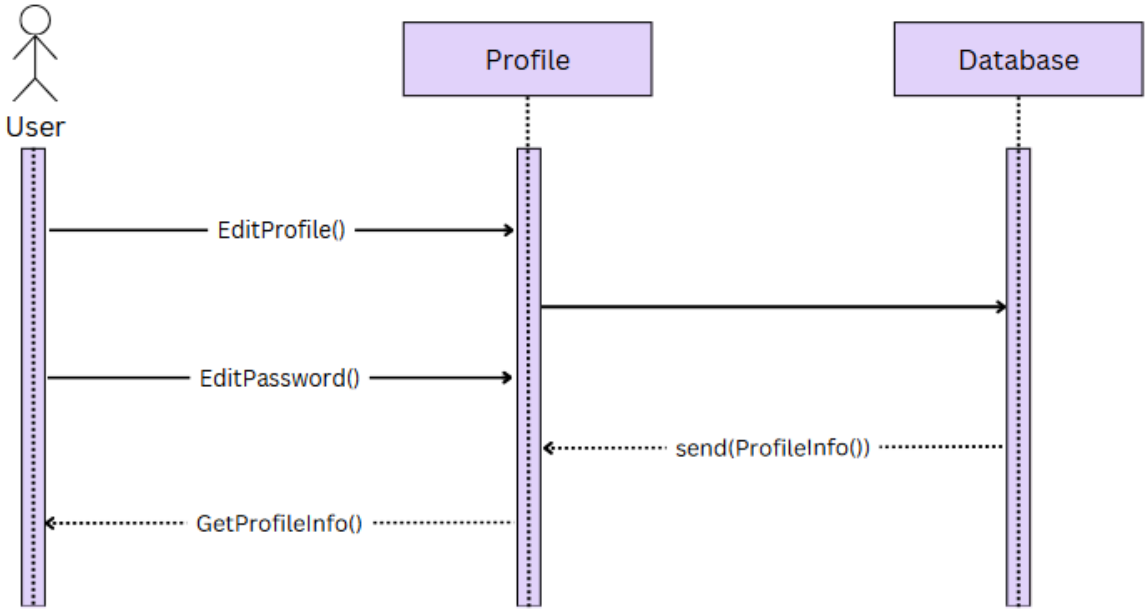


3.3 Sequence Diagrams

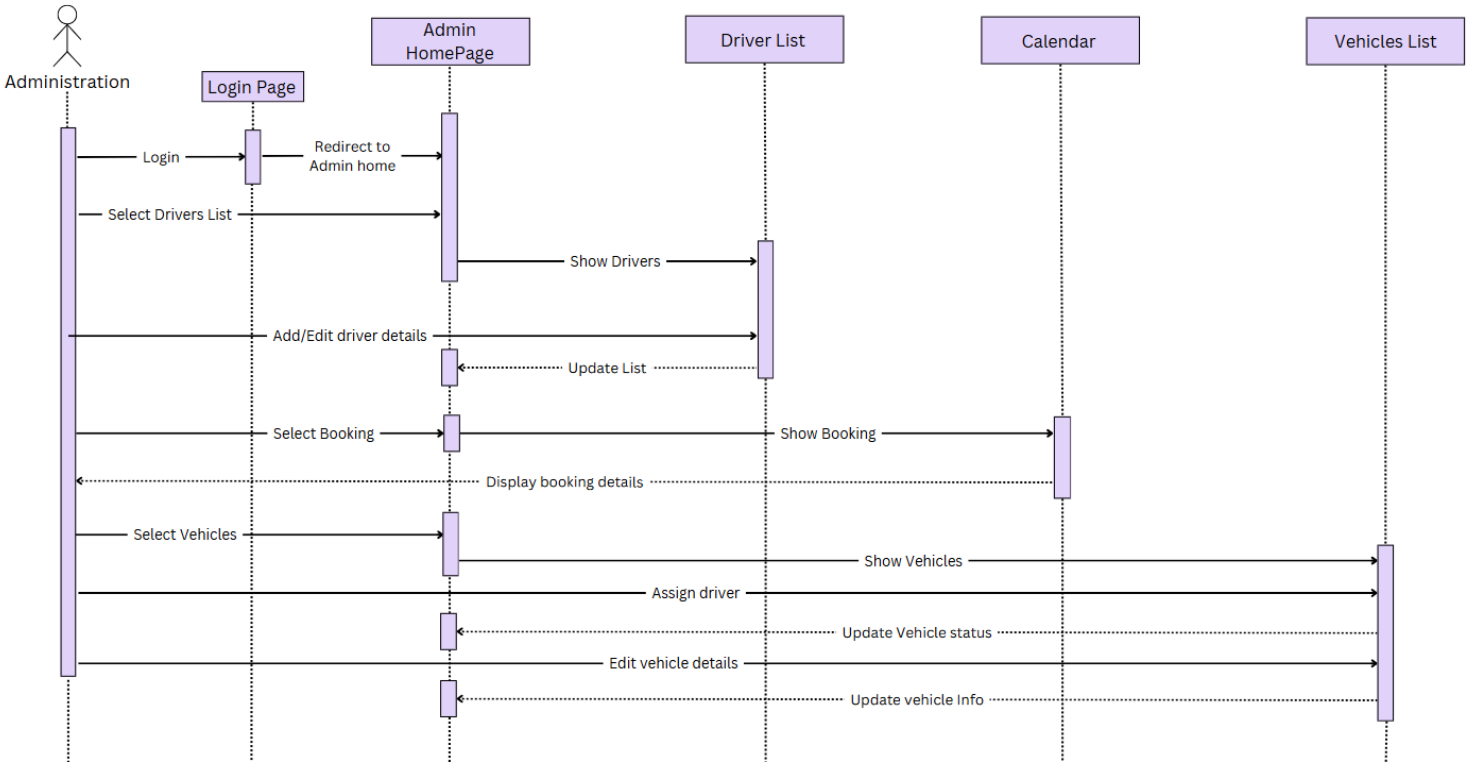
3.3.1 Authentication



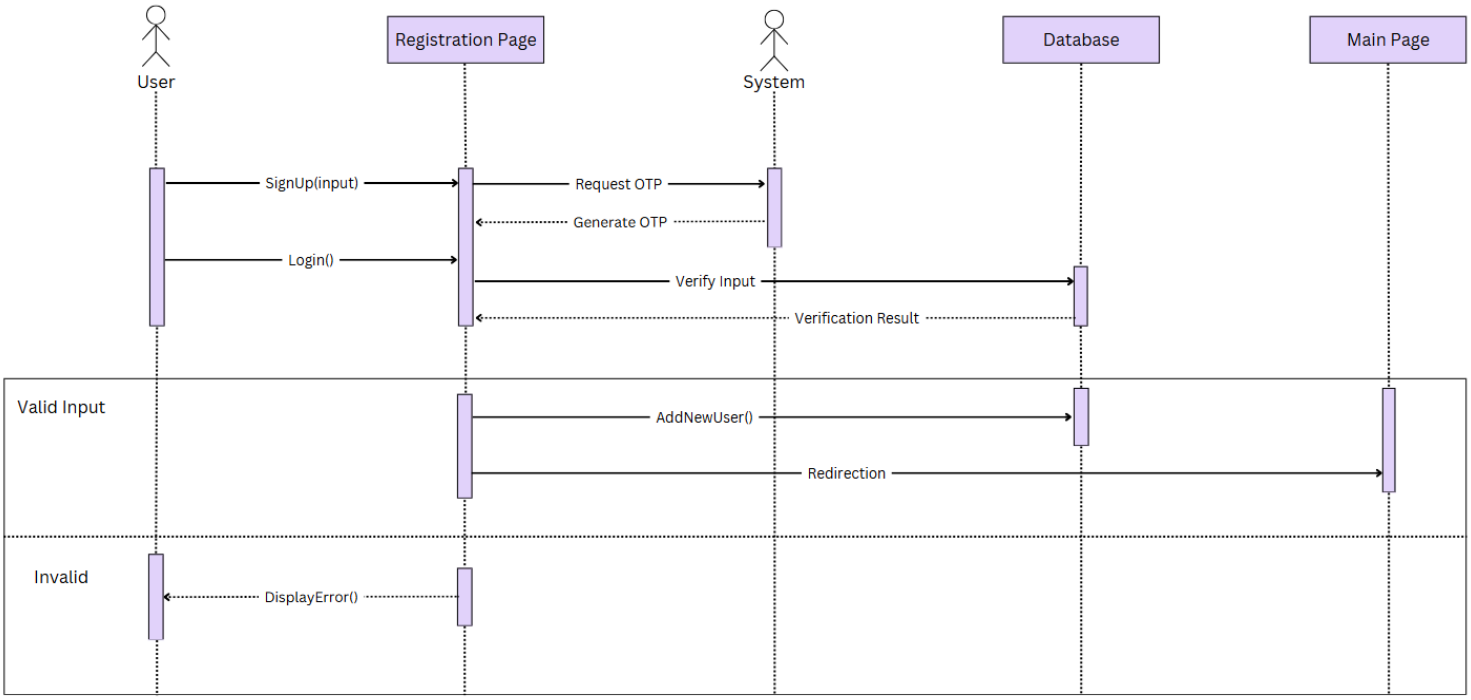
3.3.2 Edit Profile



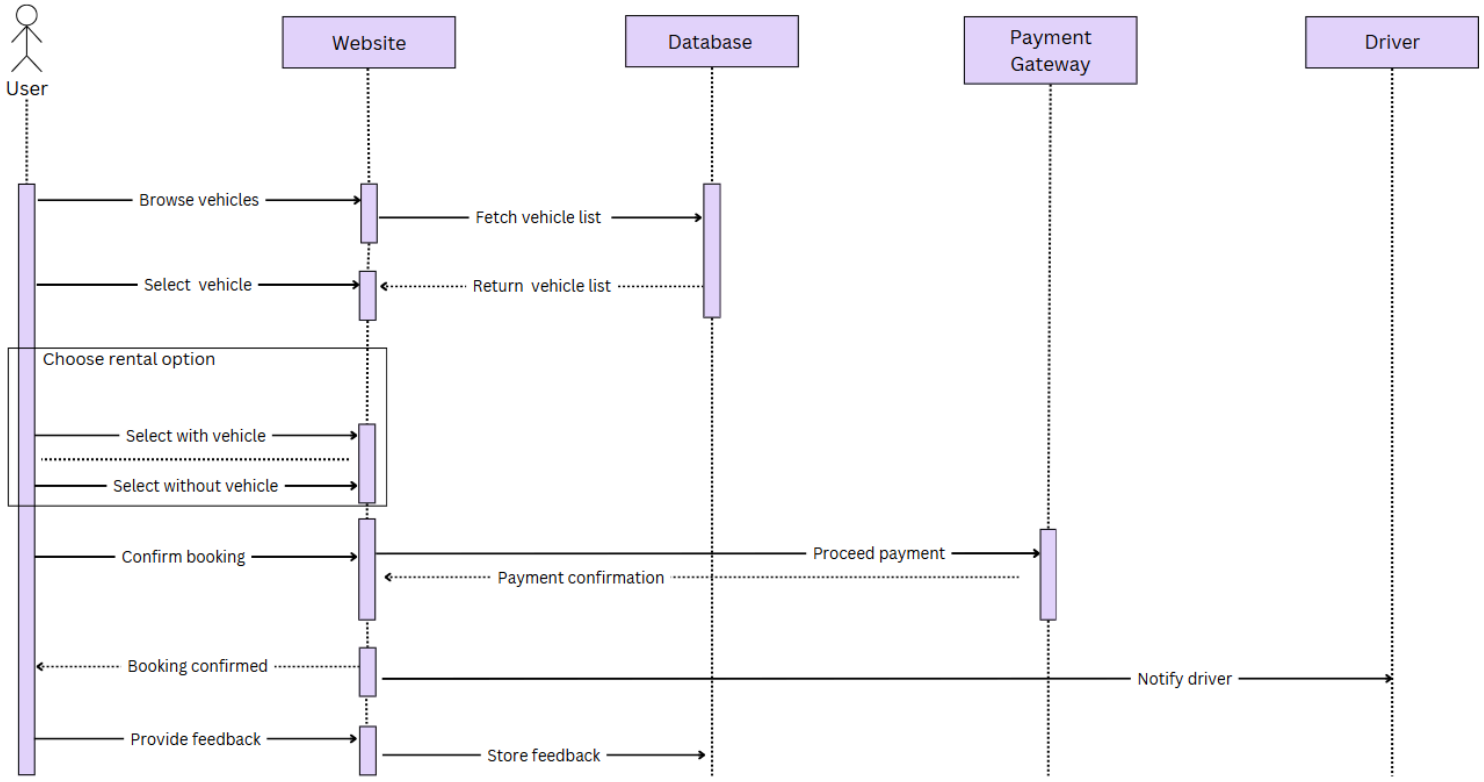
3.3.3 System - Admin Interaction



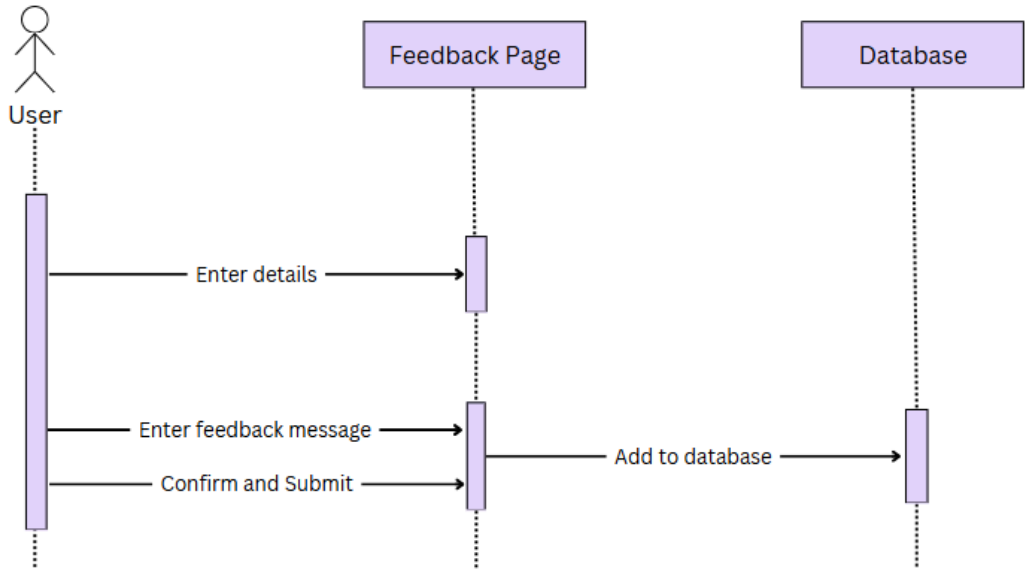
3.3.4 User Registration



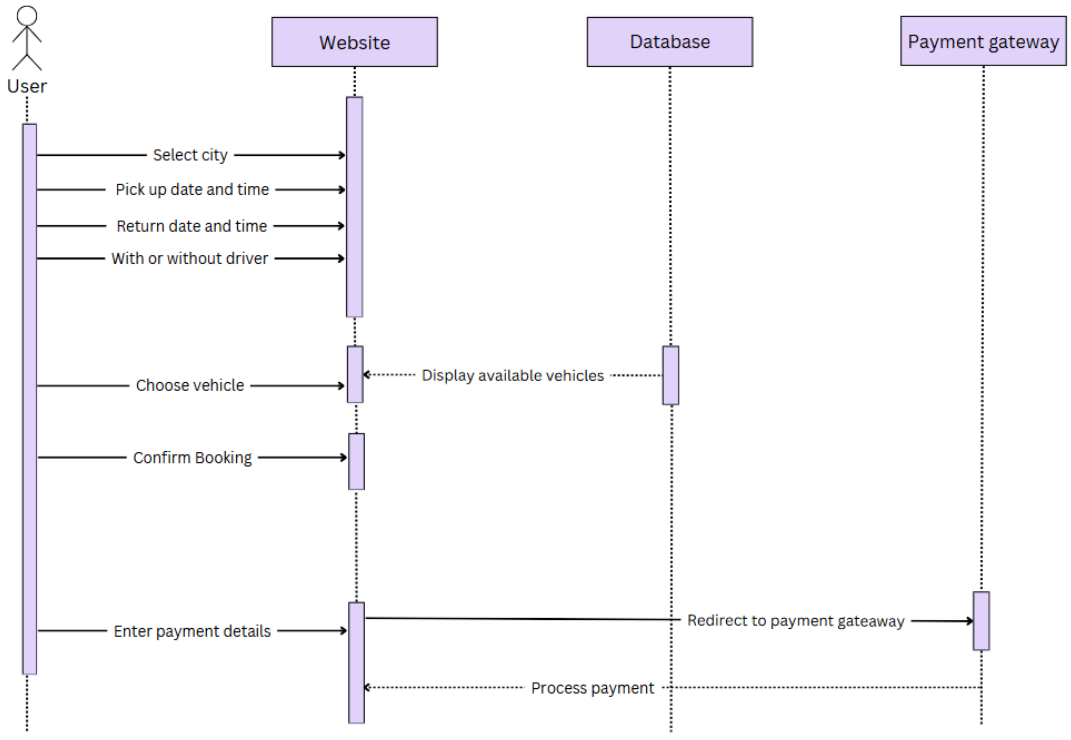
3.3.5 User - System Interaction



3.3.6 User Feedback

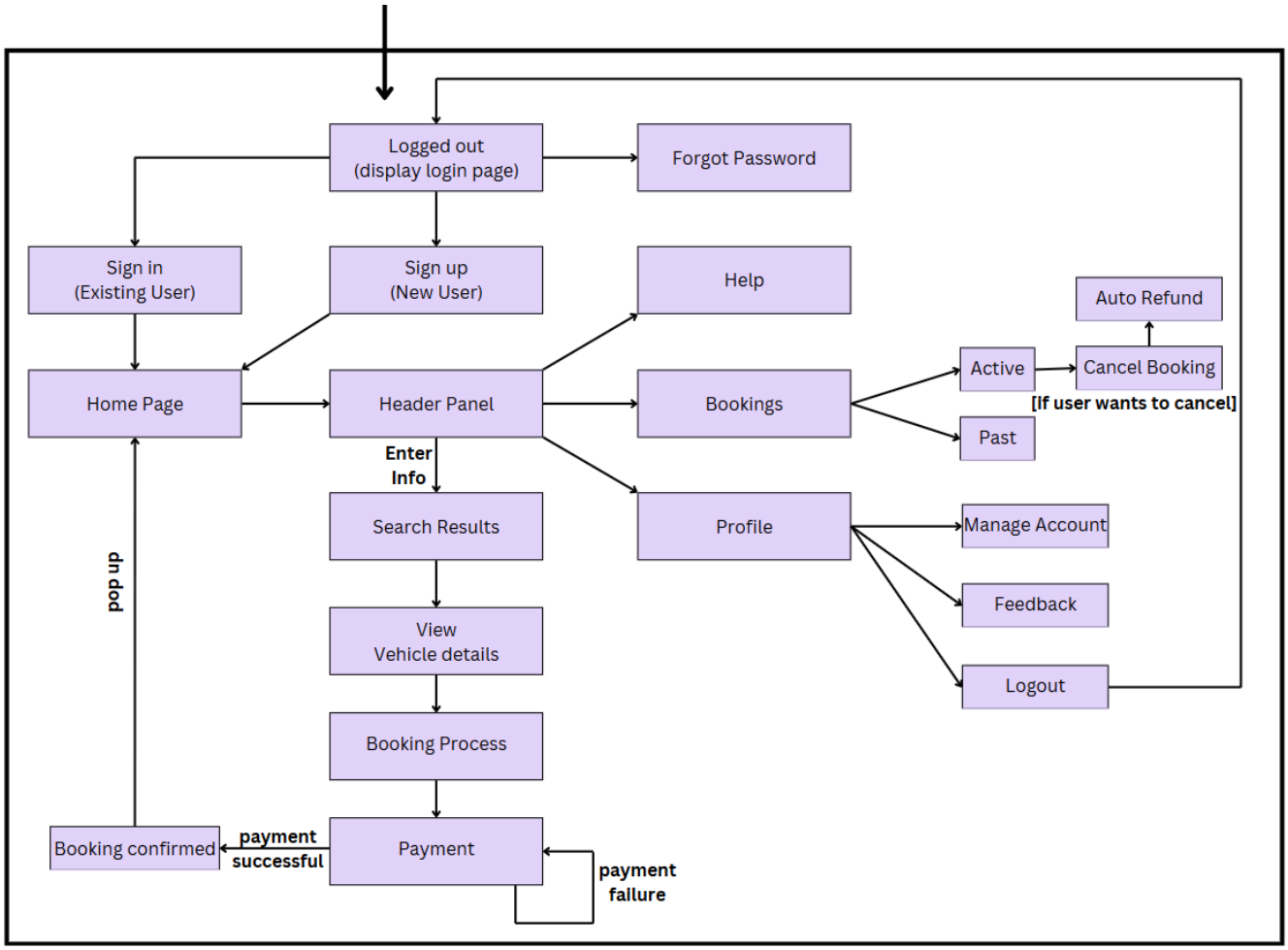


3.3.7 Payment Gateway

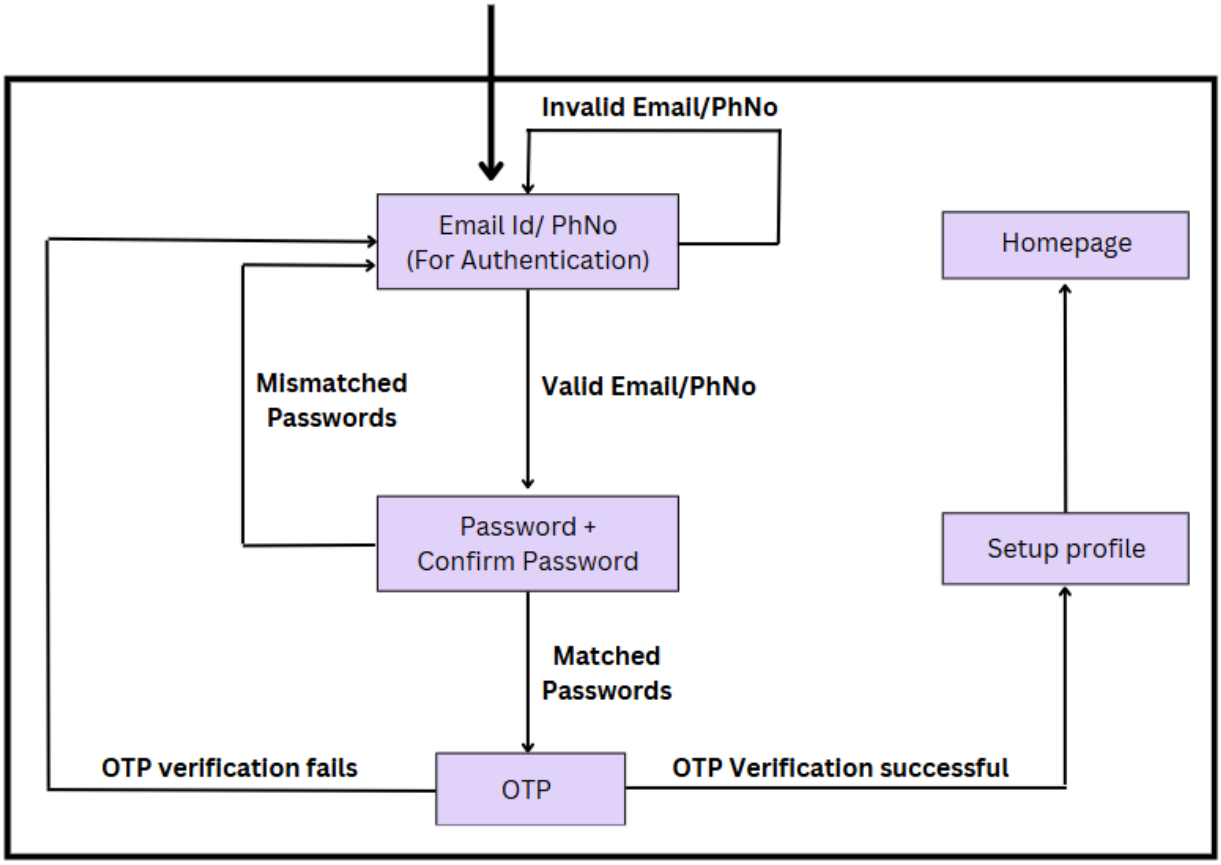


3.4 State Diagrams

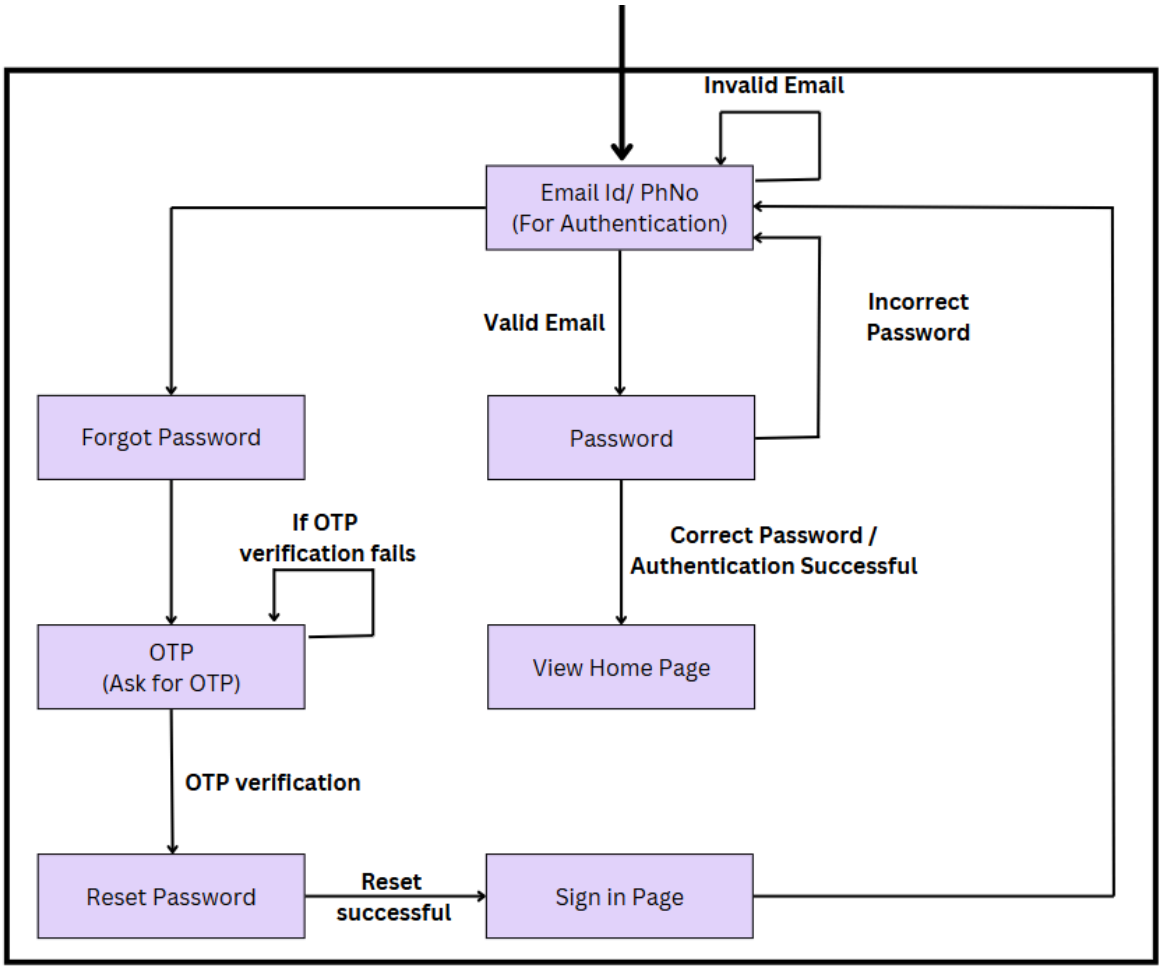
3.4.1 Overall State Diagram



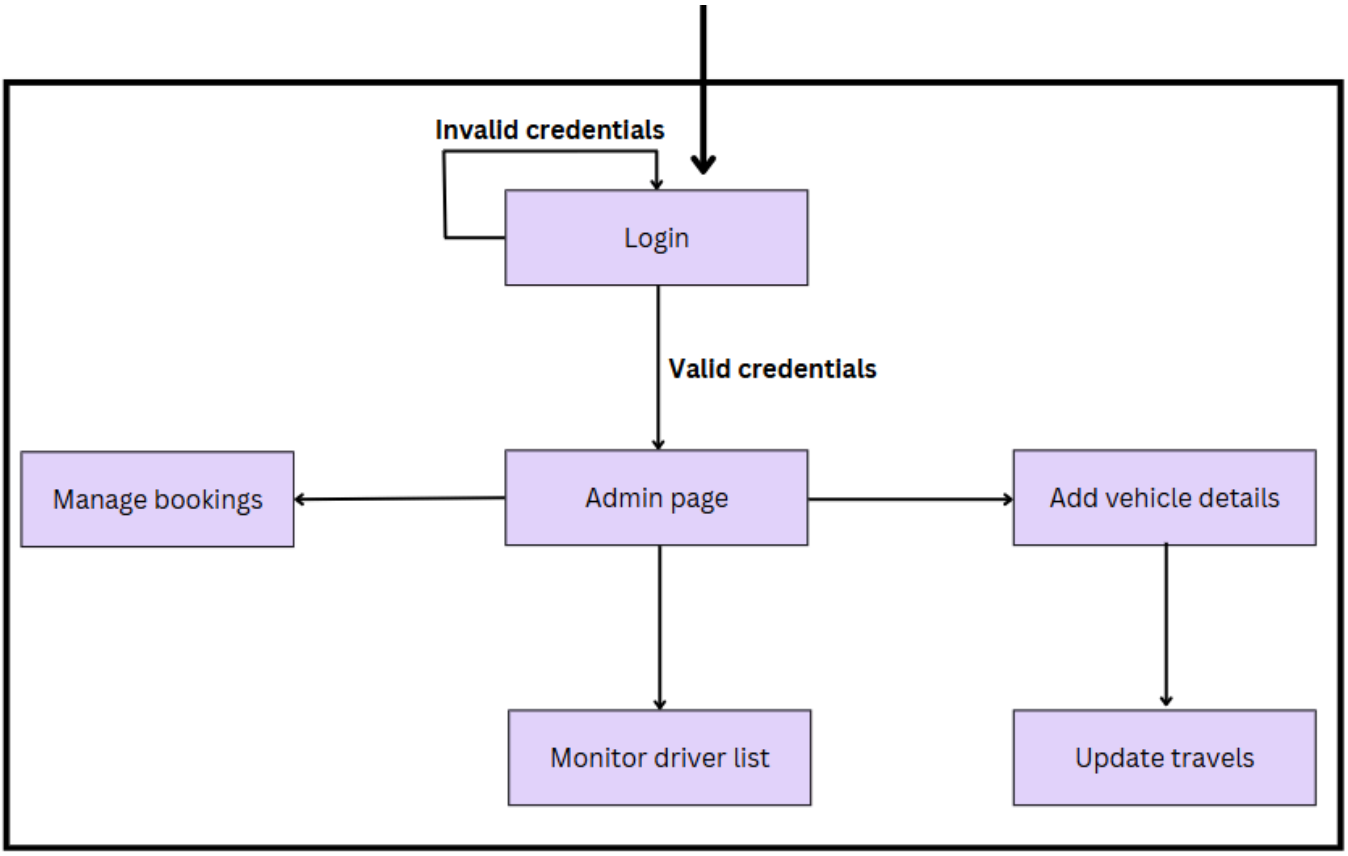
3.4.2 Sign Up (New User)



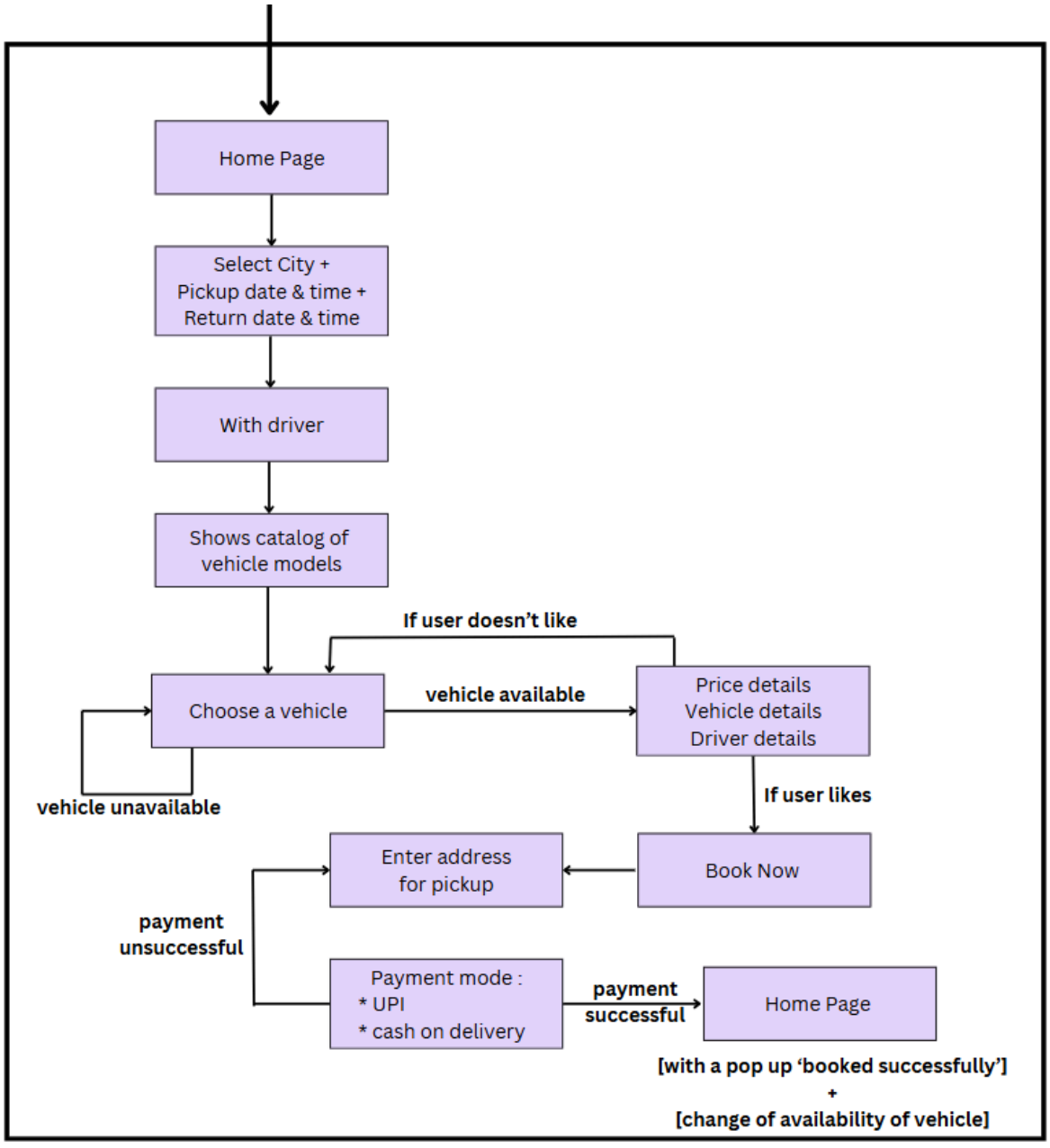
3.4.3 Sign In (Existing User)



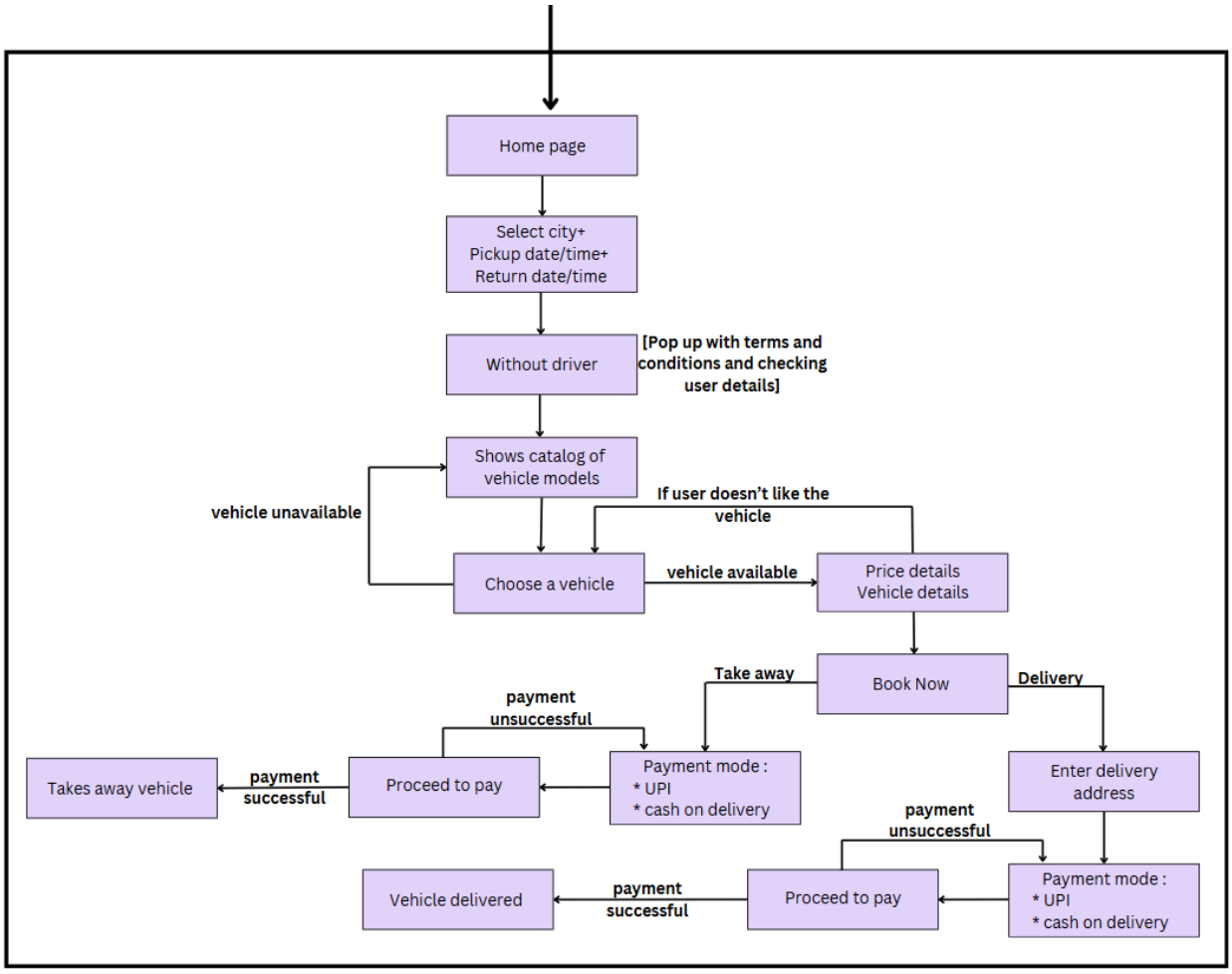
3.4.4 Administrator



3.4.5 User With driver

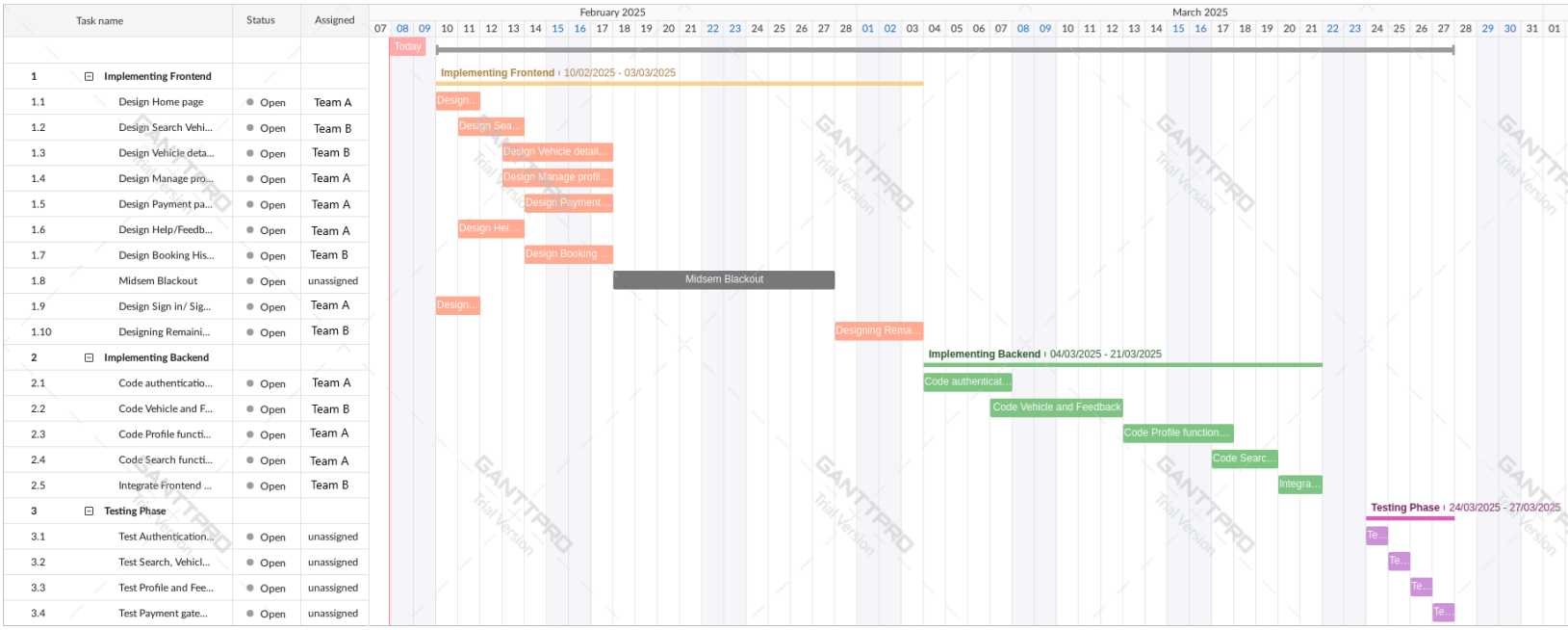


3.4.6 User Without driver



4 Project Plan

Fleet | **Fleet**



Frontend -

- Team A - Pallavi, Jyothika, Poorvie
- Team B - Prabhav, Shashi

Backend -

- Team A - Akhil, Karthik, Ramith
- Team B - Sruthi, Gayathri

Appendix A - Group Log

Meet Date	Topic Discussed	Duration
27/01/2025	We discussed prerequisites we need for the document	90 min
29/01/2025	We have divided work for the document	90 min
01/02/2025	We discussed doubts regarding our work	120 min
04/02/2025	We have divided work for project implementation	90 min
06/02/2025	We have combined our work and finalized our document	120 min
07/02/2025	Final submission	-