Design Document

for

Chalo Kart

(Golf Cart Management)

Version 1.0

Prepared by

Group 13:		**Group Name: EE Got Latent**
Trijal Srivastava	**221214**	**trijals22@iitk.ac.in**
Snehasis Satapathy	**221070**	**ssnihasis22@iitk.ac.in**
Gautam Arora	**220405**	**garora22@iitk.ac.in**
Naman Gupta	**220686**	**namangupta22@iitk.ac**
Divyam Agarwal	**220376**	**divyamag22@iitk.ac.in**
Udbhav Agarwal	**221149**	**audbhav22@iitk.ac.in**
Deham Rajvanshi	**220335**	**dehamr22@iitk.ac.in**
Ayan Gupta	**220258**	**ayangupta22@iitk.ac.in**
Saksham Parihar	**220939**	**psaksham22@iitk.ac.in**
Dharvi Singhal	**220354**	**dharvis22@iitk.ac.in**

Course: CS253

INDEX

Revisions

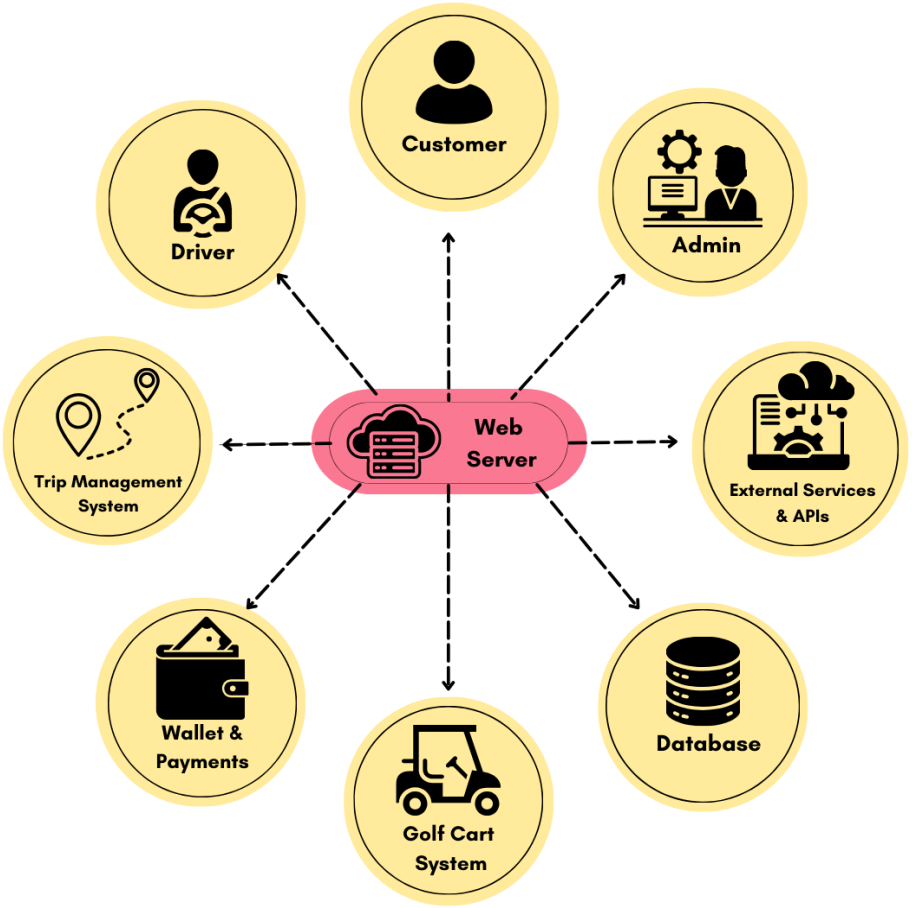
Version	Primary Author(s)	Description of Version	Date Completed
1.1	Trijal Srivastava Snehasis Satapathy Gautam Arora Naman Gupta Divyam Agarwal Udbhav Agarwal Deham Rajvanshi Ayan Gupta Saksham Parihar Dharvi Singha	The first version of the Software Design Document was once presented to the TA for proofreading.	7/02/2025

1 Context Design

1.1 Context Model

1.1.1 Overview of the System

The system is designed to facilitate managing and operating a **golf cart transportation service**. It includes functionalities for trip booking, driver assignments, route optimisation, and payment processing. The system integrates multiple entities such as **customers, drivers, admins, trips, wallets, and payments**, ensuring seamless operations and tracking.

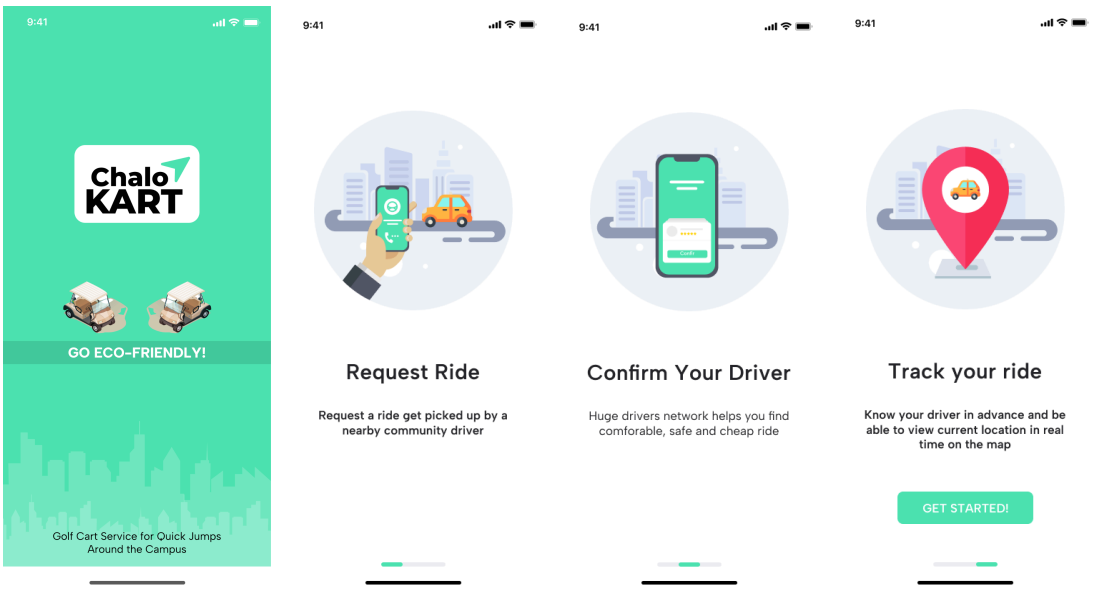


1.2 Human Interface Design

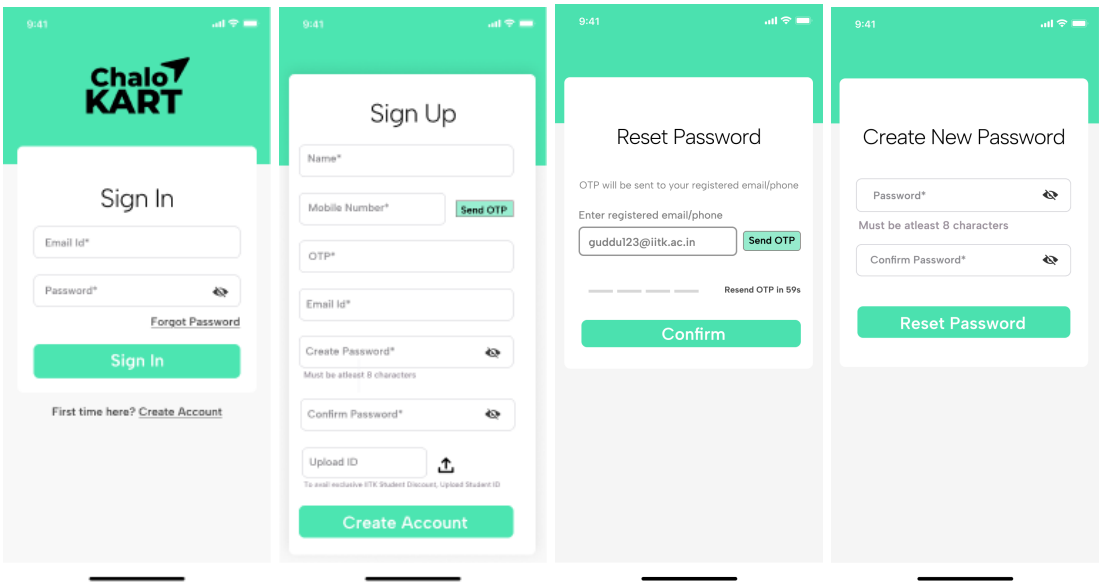
Users will engage with the system via the designed mobile application through a user-friendly interface on both the client and driver sides. It will be accessible on Android devices, ensuring a seamless interaction to facilitate the application services.

1.2.1 User Login, Registration and Onboarding

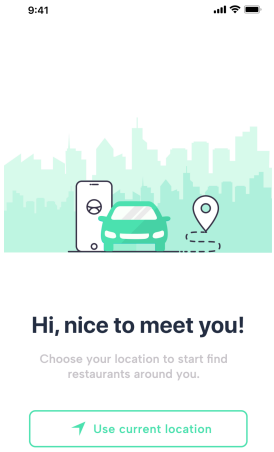
A neat loading page features the ‘Chalo Kart’ logo when the application is opened. It is followed by a short onboarding to better understand the app and its use.



To use the app, the user is required to sign in. If the user is new, there is an option to create a new account. A ‘forgot password’ button is also present in case the user loses his credentials and wants to retrieve them.

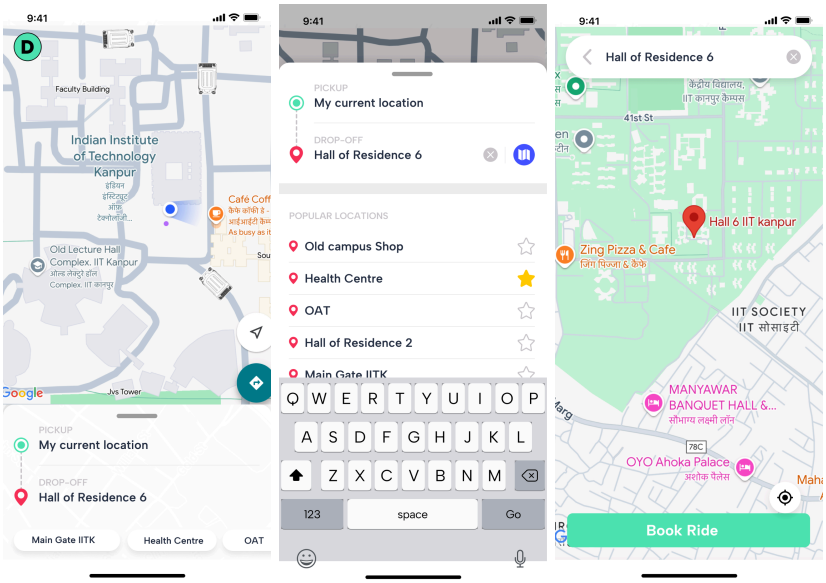


Lastly, after logging in, there is an option to set up a GPS location or give GPS access from the mobile settings.

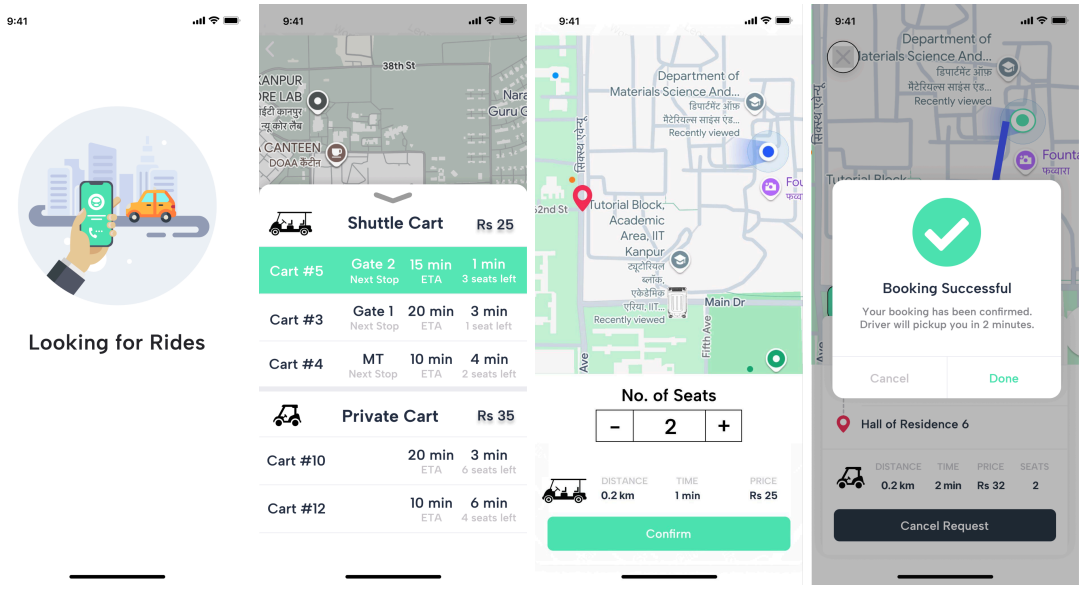


1.2.2 Home Page and Ride Booking

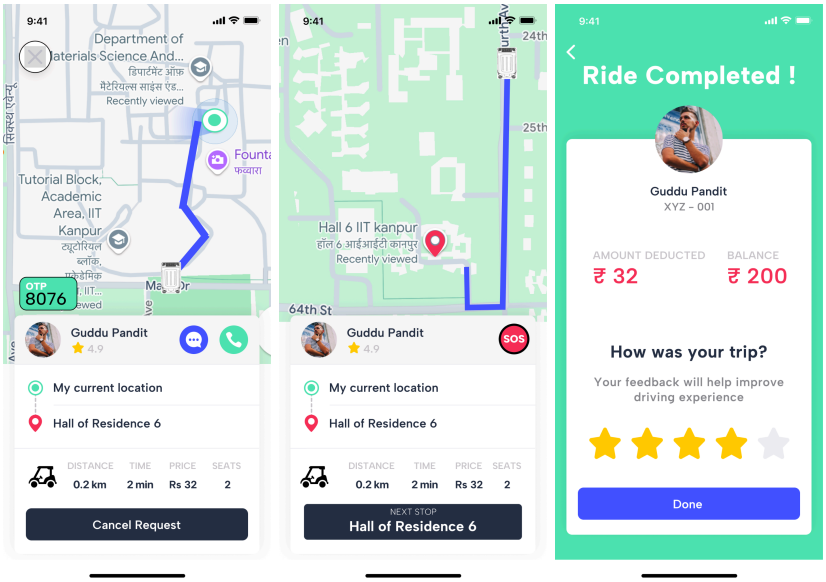
Once the application is set up, the user will see a map showing the user's location and an option to continue ride booking.



When the pickup and drop locations are set and ‘book ride’ is clicked, the app will search for all available carts and show a list of the nearest ones on the next page. The booking is successful once the suitable cart and number of seats are chosen.



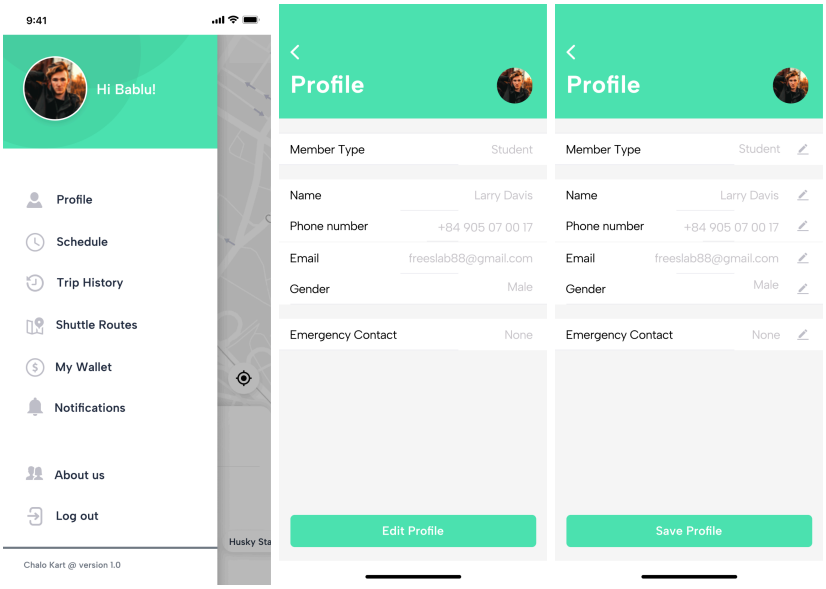
All the ride info, OTP, driver reaching time, and other details, are shown.



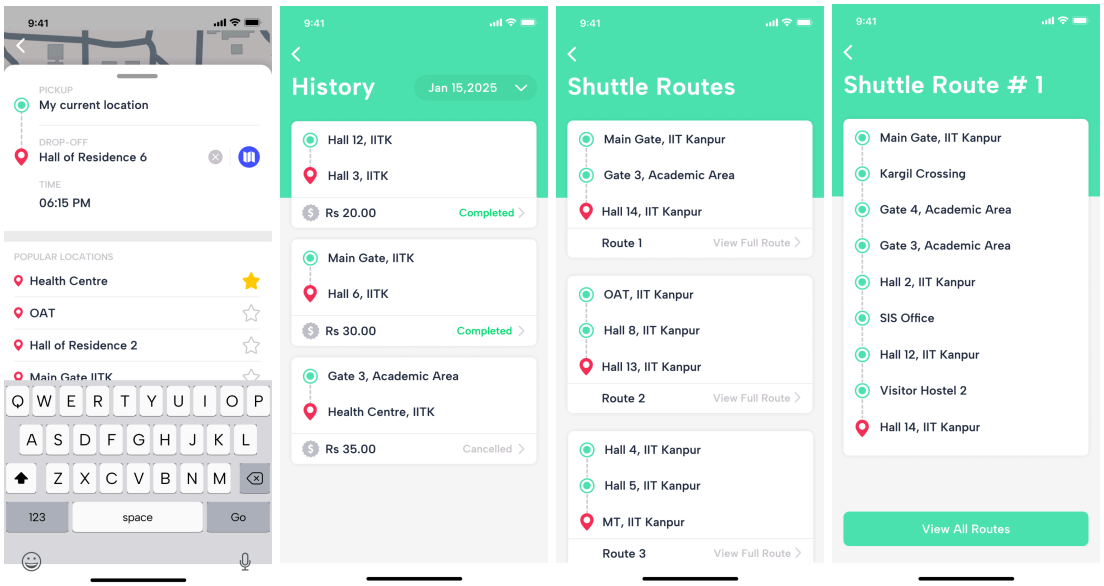
When the ride is completed, an appropriate amount is deducted from the user's wallet, and feedback is requested.

1.2.3 Sidebar Utilities

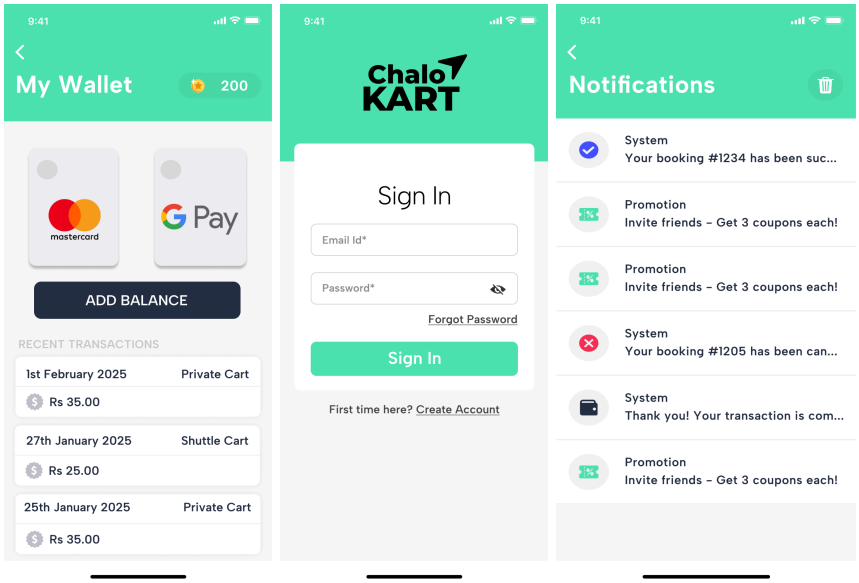
On the sidebar, ‘Profile’ is first on the list. It lists all the user info on the app and gives an option to update those.



An option to schedule the ride is also available, taking you to the book ride page and an additional ‘Time’ entry.



Trip History is shown for all the completed and cancelled rides. Shuttle Routes give a detailed plan of how the golf cart moves on each predefined route. ‘Notifications’ shows all the promotions, system messages, and ride information.

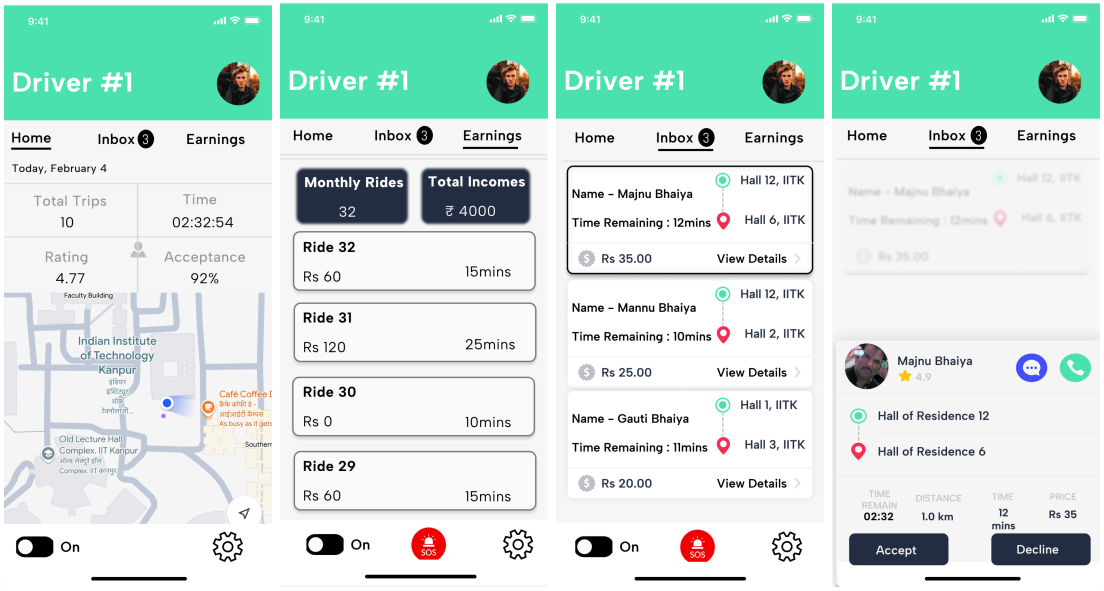


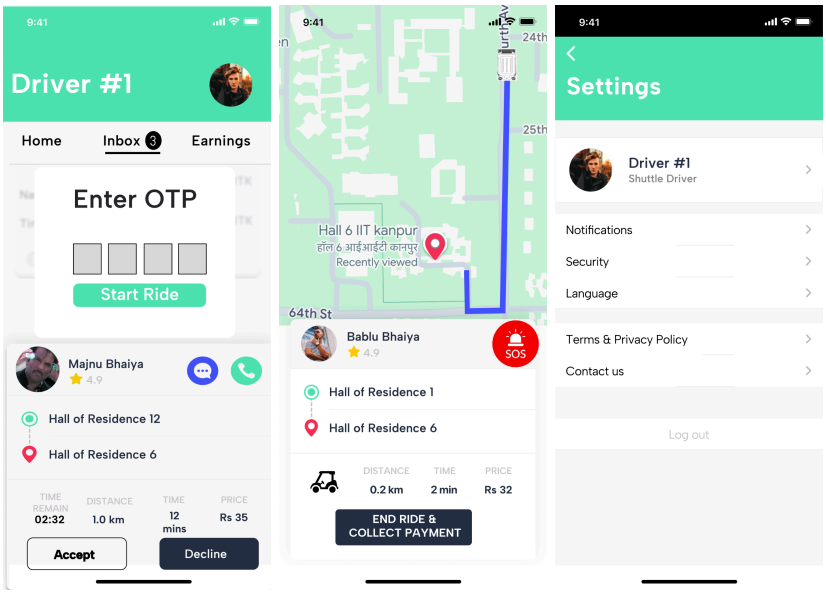
‘Wallet’ is a ‘Chalo Kart’ specific payment system. Users can top up their wallets using available payment methods. Once a ride is completed, the amount will be deducted from the user’s wallet. On clicking ‘Log out’ on the sidebar, you are redirected to the ‘signin’ page shown below.

1.2.4 Driver and Administrator Pages

The **driver dashboard** provides essential information, including the **number of trips completed, total driving time, earnings, and user ratings**.

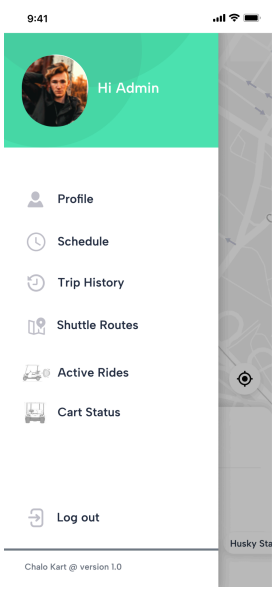
Additionally, the dashboard features real-time navigation to indicate the **pickup location for the next ride** and includes a secure interface for **entering the OTP** required to initiate the ride.

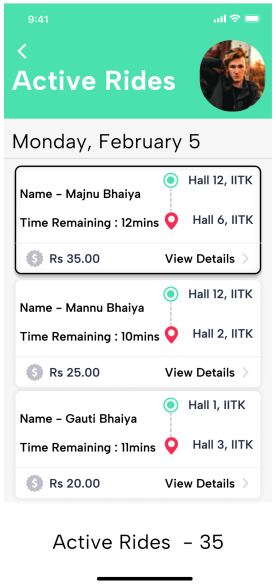




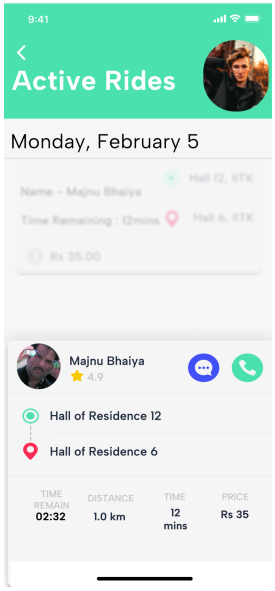
The **admin dashboard** provides comprehensive access to critical system data, including **cart status, trip history, and active rides**. Administrators have the ability to utilise the application as a standard user while also accessing additional functionalities through the **Cart Management System**.

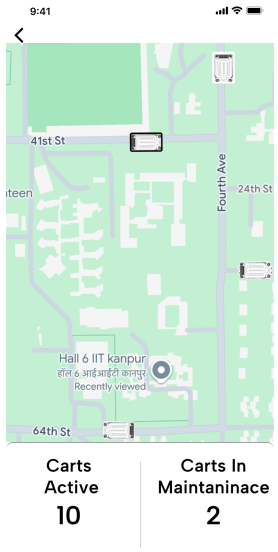
Admin privileges are linked to a **unique user ID**, ensuring that designated administrators can seamlessly switch between **user and administrative roles**, allowing them to manage operations efficiently while retaining the full capabilities of a regular user.



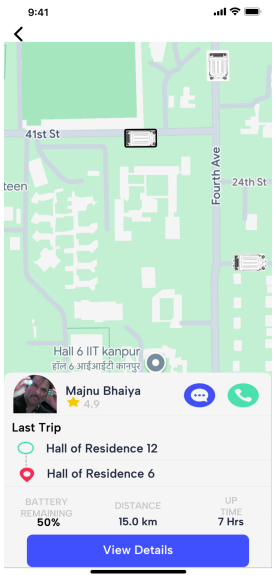


Active Rides - 35





Carts Active	Carts In Maintaninace
10	2



Current Status - Active

2 Architecture Design

Our software is designed using the **Model-View-Controller (MVC)** architecture, which helps maintain a clear separation between the data (**Model**), user interface (**View**), and business logic (**Controller**). Considering its complexity, we have broken down the app's functionality into smaller, more manageable components. Each major feature is further divided into individual tasks, ensuring **modularity** and **ease of maintenance**. This structured approach helps us to develop, test, and scale various parts of the application independently in a way that maintains a coherent system that is both efficient and easy to manage. MVC is also flexible and allows the application to expand and incorporate more features, if necessary, in the future.

Model(Backend/Data Layer)

The **Model** serves as the data-centric component of the golf cart booking app, handling the storage and management of essential data. It directly interfaces with the **MySQL** database, ensuring smooth data retrieval and updates. This layer maintains structured information, including **users, drivers, rides, golf carts, payments, and shuttle routes.**

Since Django is the backend framework, it facilitates database interactions through its ORM (Object-Relational Mapping), ensuring data integrity and efficient querying. When a user books a ride, the model updates the ride details, cart availability, and payment status in real-time. The **wallet system** is implemented in this layer, where users can load money via **Razorpay's Free API**, track transaction history, and use the stored balance for seamless payments.

View(Presentation Layer)

The **View** represents the front-facing component of the app, which is responsible for rendering data and handling user interactions. It was developed using **Flutter** to provide a smooth cross-platform experience.

- Users can access profile management, login/signup, live tracking, ride booking/cancellation, payments, ride history, and shuttle route details.

- Drivers have access to ride acceptance/decline, live tracking, and ride history, ensuring efficient management of bookings.

- Administrators manage operations directly through the application by logging in as standard users, with elevated access privileges linked to their unique user ID. This access enables them to monitor active rides, review ride history, track cart status, and analyse detailed booking records, ensuring efficient system oversight.

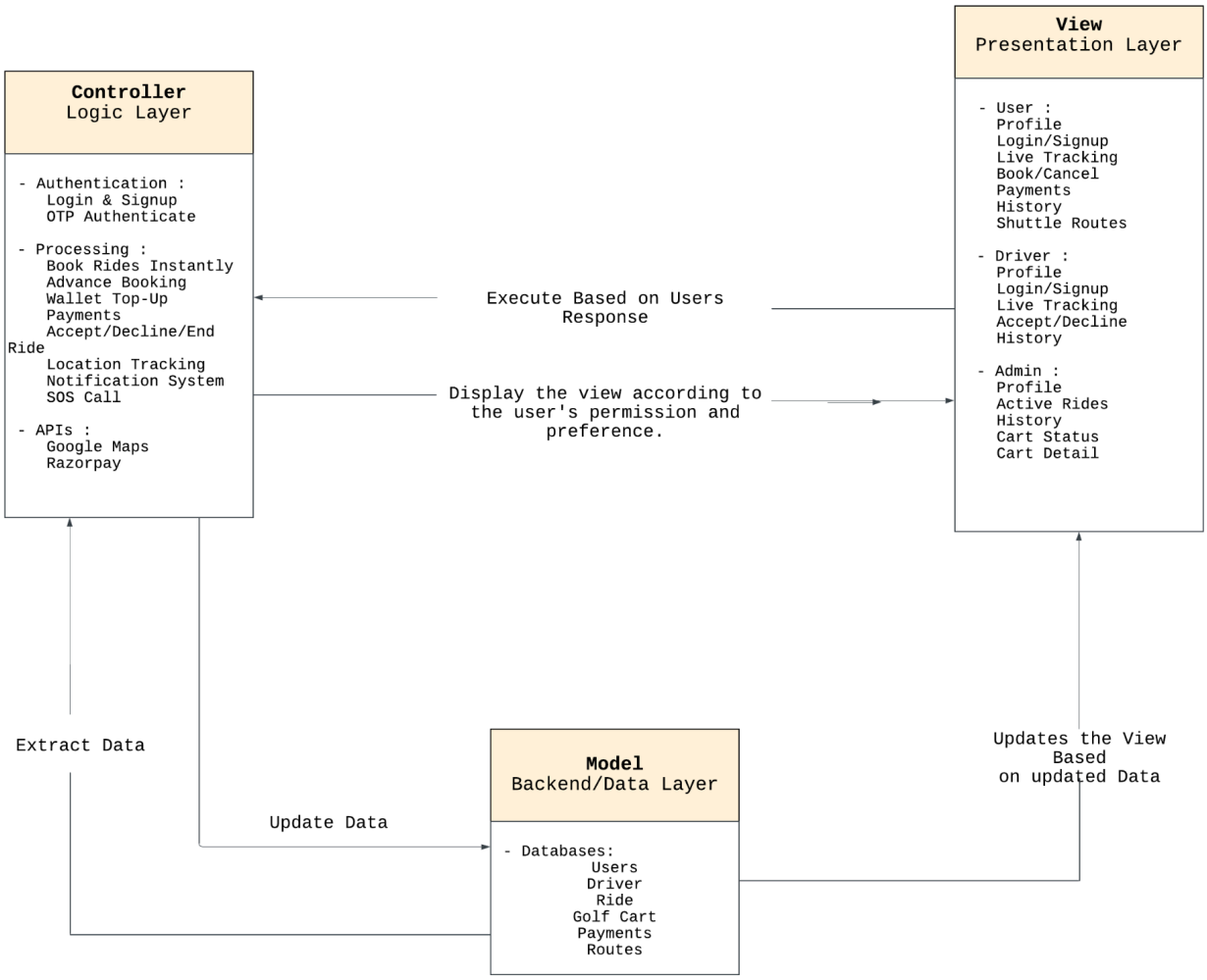
The View dynamically updates based on user actions and permissions, ensuring a **seamless** and **interactive** user experience.

Controller(Logic Layer)

The **Controller** acts as the logic hub, orchestrating data processing and application flow. Built using **Django in Python**, it handles user authentication, business logic, and API integrations.
Key responsibilities include:

- **Authentication:** Managing user login, signup, and OTP-based authentication.
- **Ride Processing:** Enabling instant ride booking, advance bookings, wallet top-ups, payments, ride status updates, location tracking, notifications, and SOS calls.
- **API Integrations:** Google Maps API is incorporated for navigation and real-time tracking, while Razorpay API handles wallet transactions and in-app payments.

By coordinating the **Model** and **View**, the Controller ensures that every user request is processed efficiently, delivering accurate data and actions in real-time.

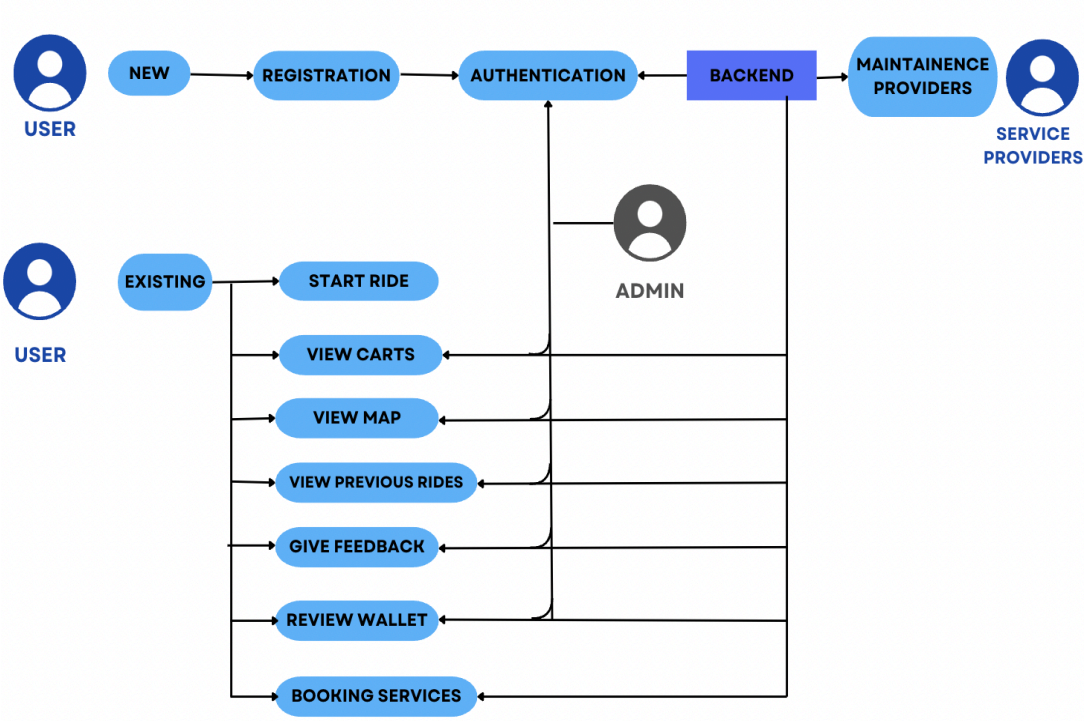


We structured our golf cart booking app using the **Model-View-Controller (MVC)** architecture to manage complex data interactions effectively. This design streamlines key operations such as **handling ride history, user management, bookings, and admin functionalities**, ensuring a smooth and organised workflow. One of the biggest advantages of MVC is its adaptability—future enhancements can be incorporated seamlessly without disrupting existing features. By following this architecture, we create a **scalable and maintainable system** that optimally serves the needs of **users, drivers, and administrators** while keeping the app efficient and easy to expand.

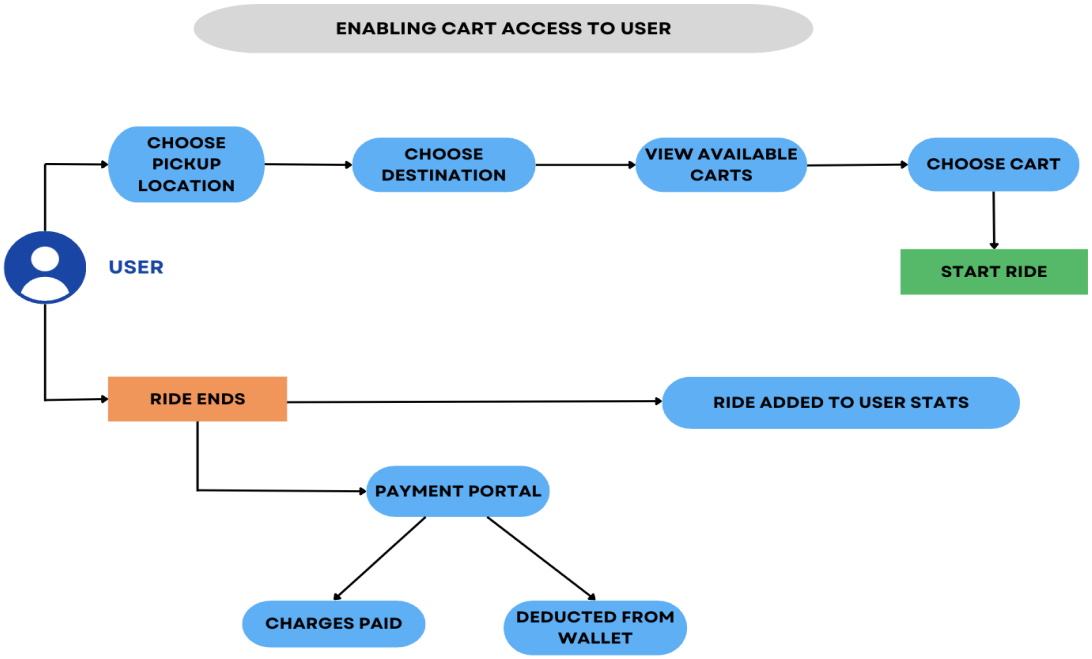
3 Object Oriented Design

3.1 Use Case Diagrams

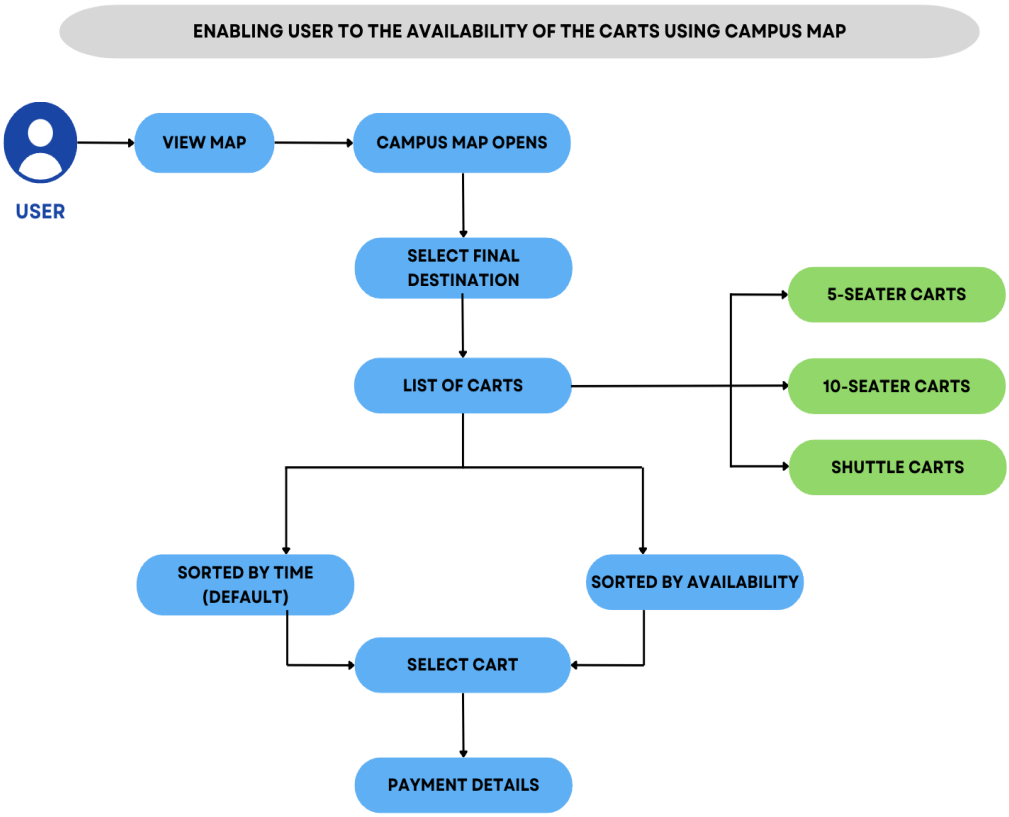
3.1.1 Use Case Model



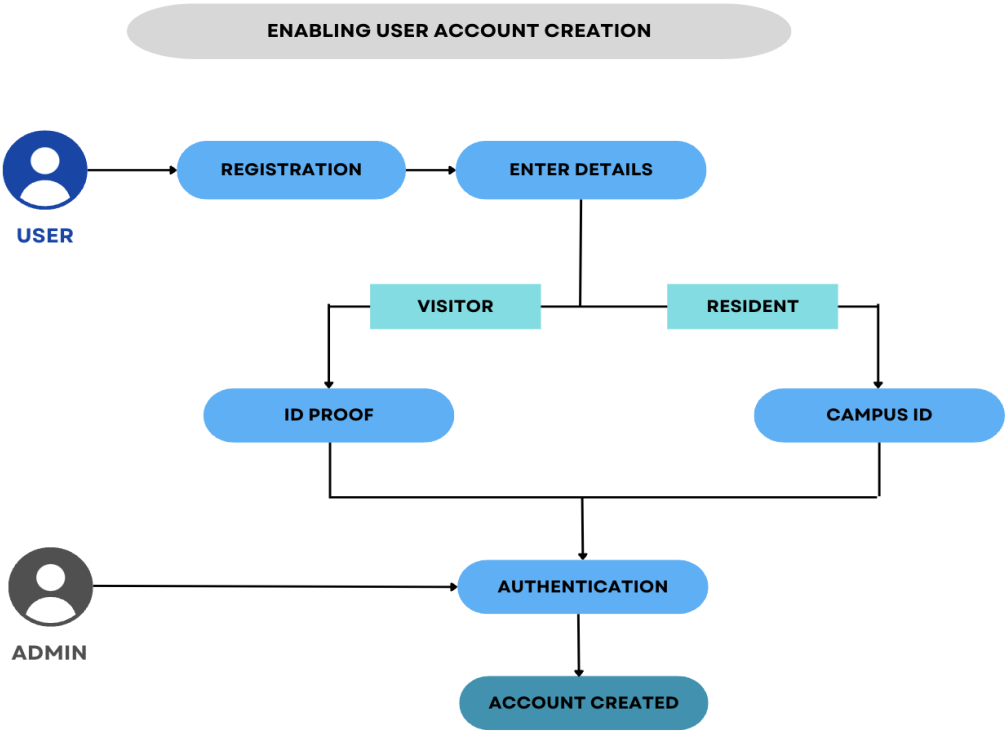
3.1.2 Use Case Diagram 2



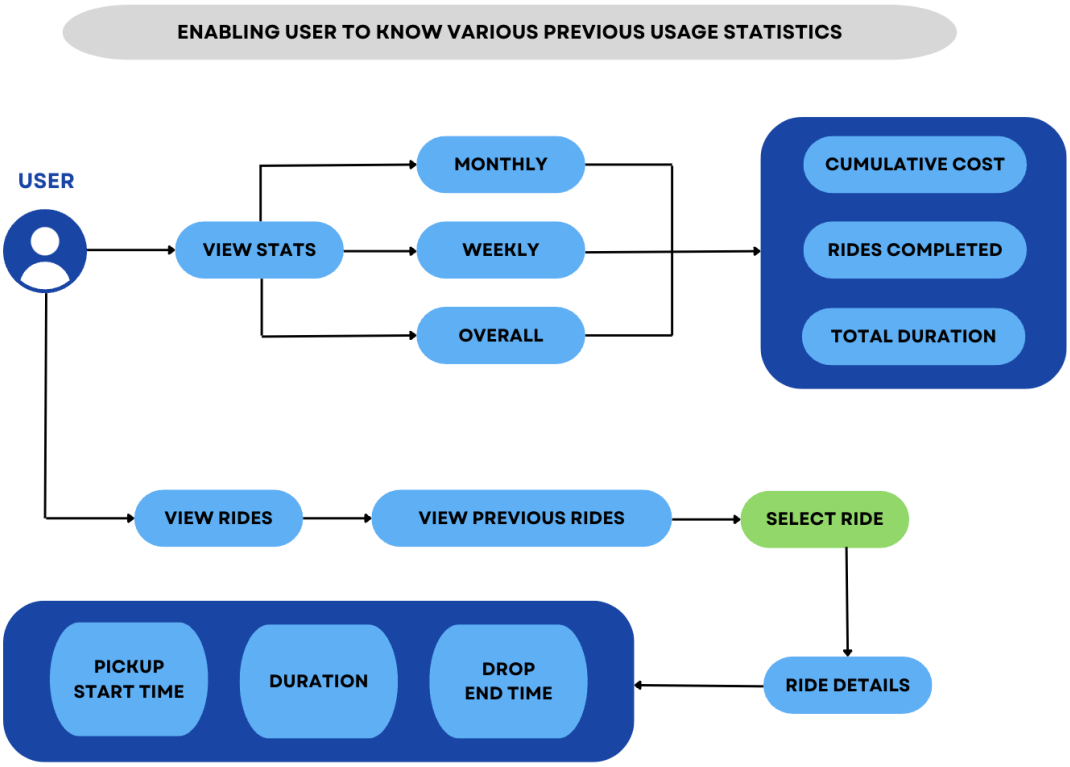
3.1.3 Use Case Diagram 3



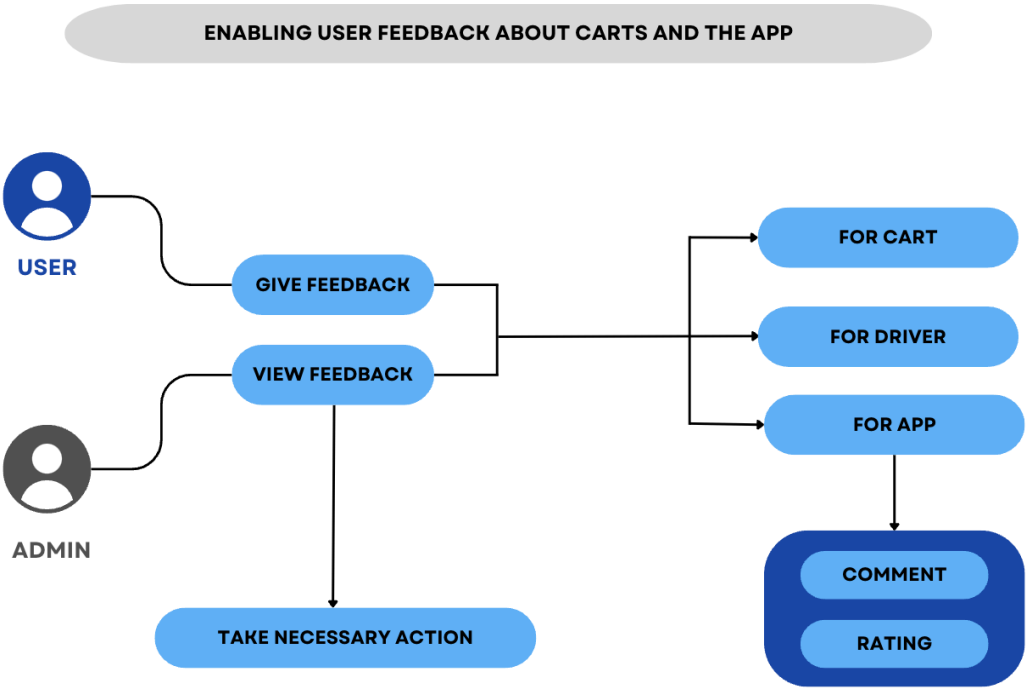
3.1.4 Use Case Diagram 4



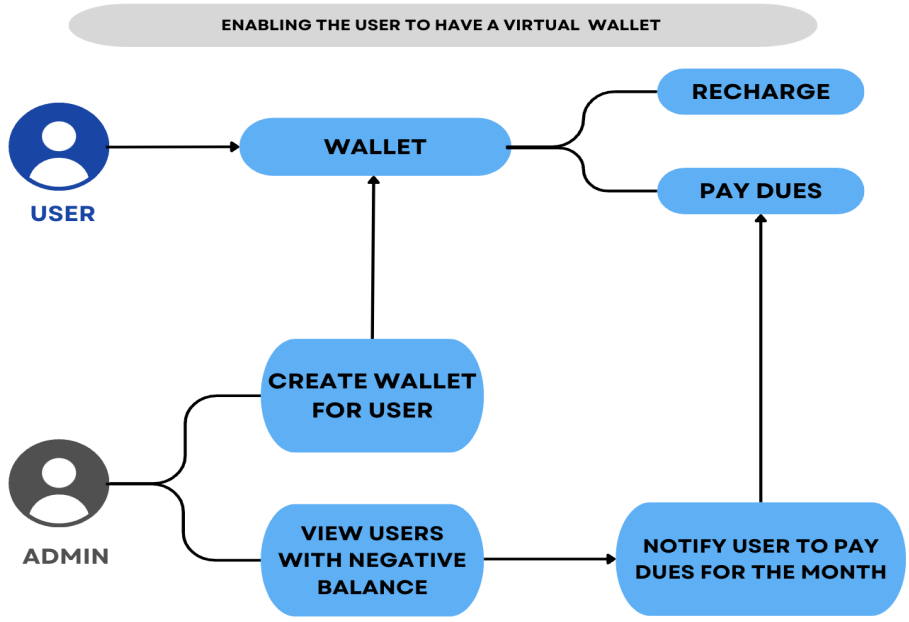
3.1.5 Use Case Diagram 5



3.1.6 Use Case Diagram 6

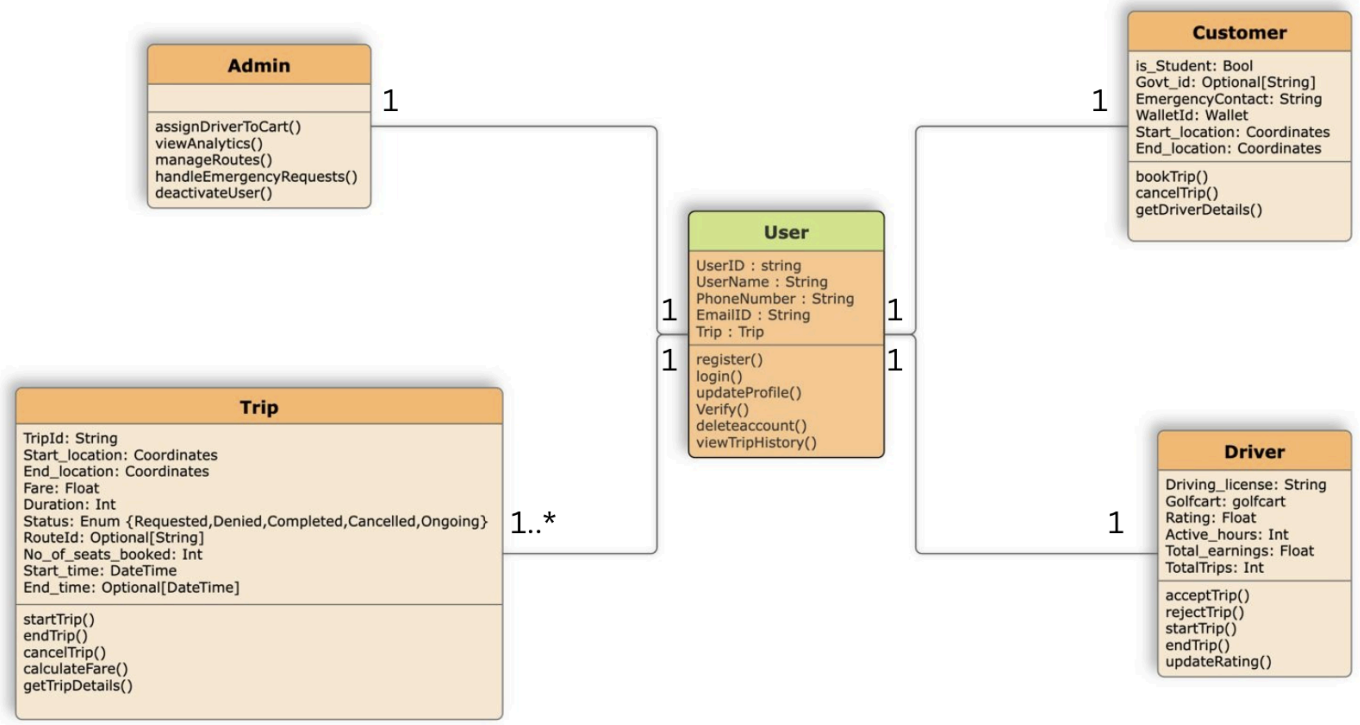


3.1.7 Use Case Diagram 7

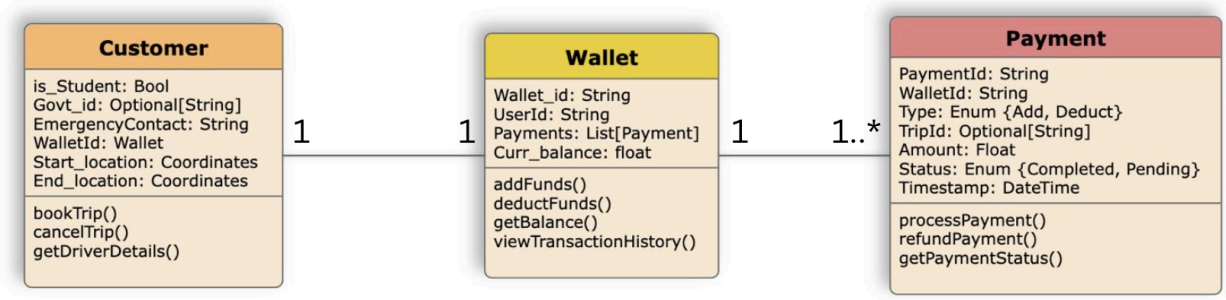


3.2 Class Diagrams

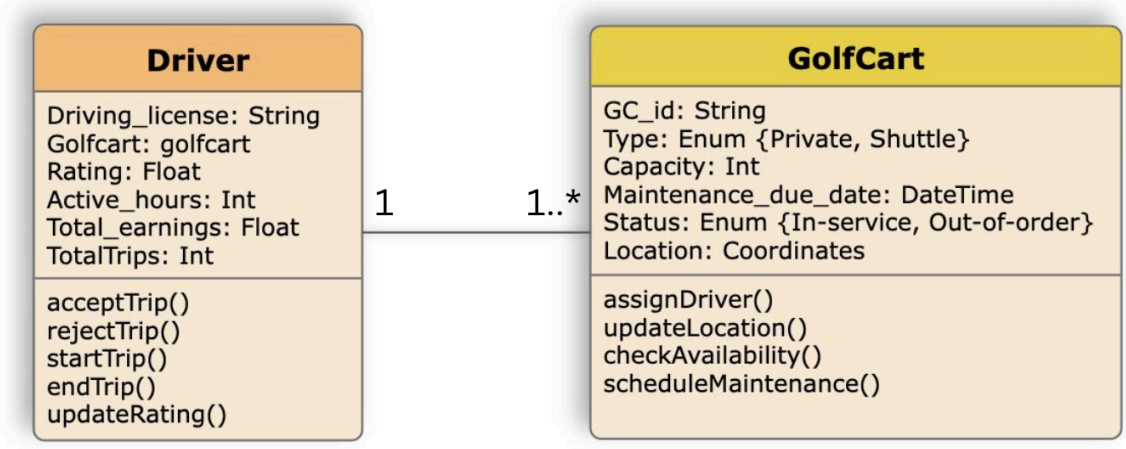
3.2.1 Case Diagram 1



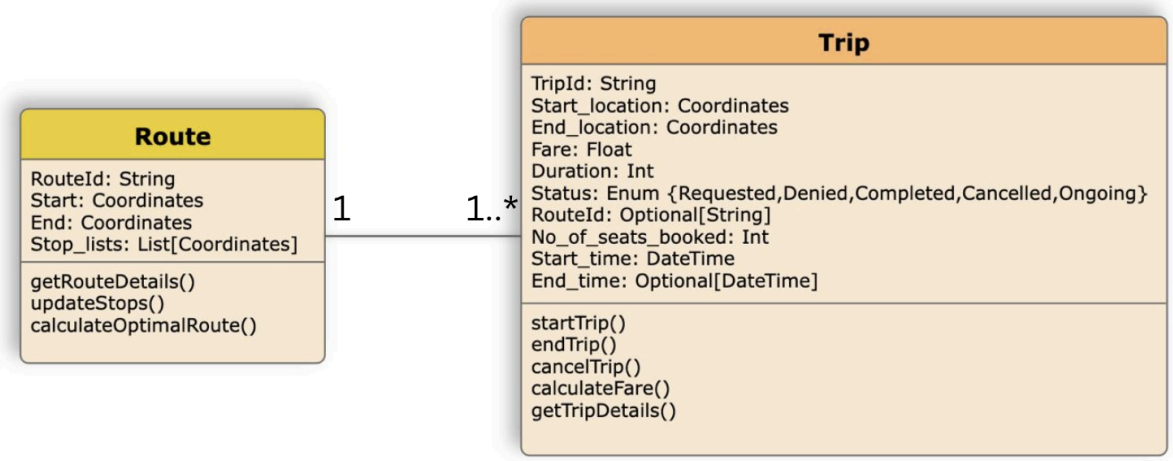
3.2.2 Case Diagram 2



3.2.3 Case Diagram 3

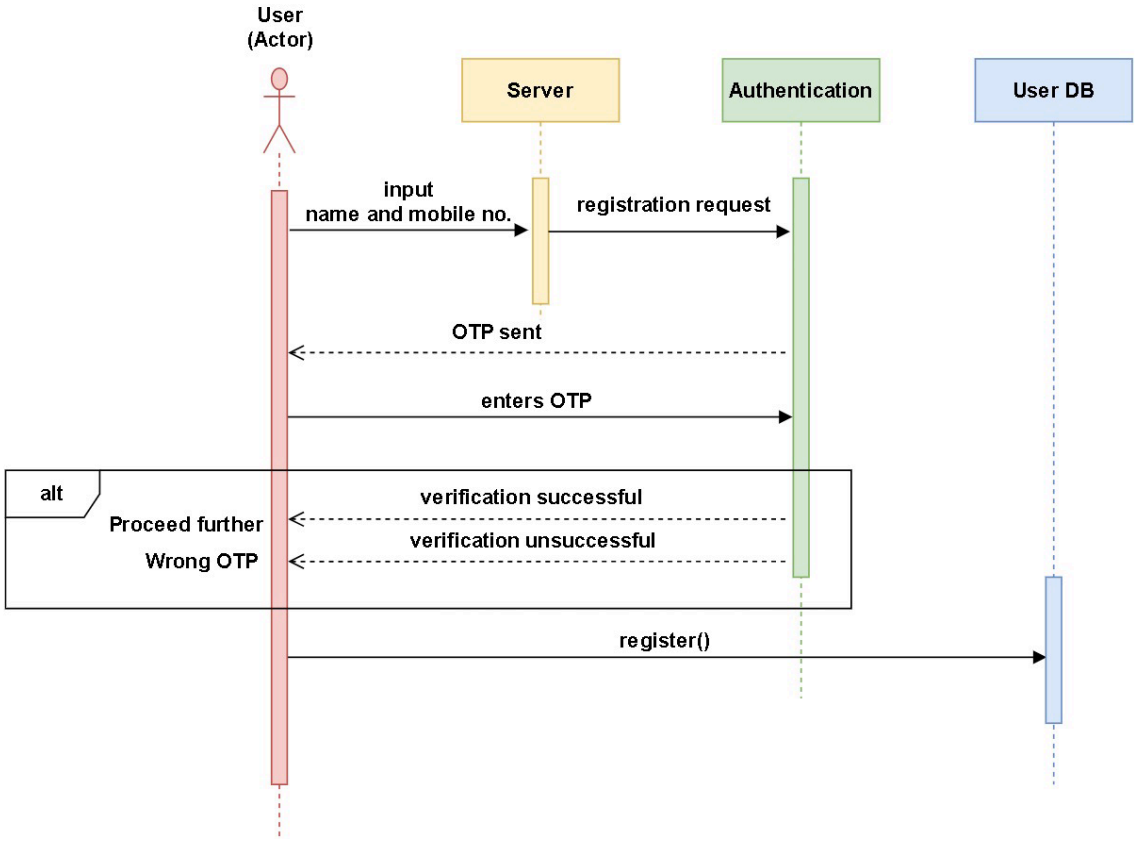


3.2.4 Case Diagram 4

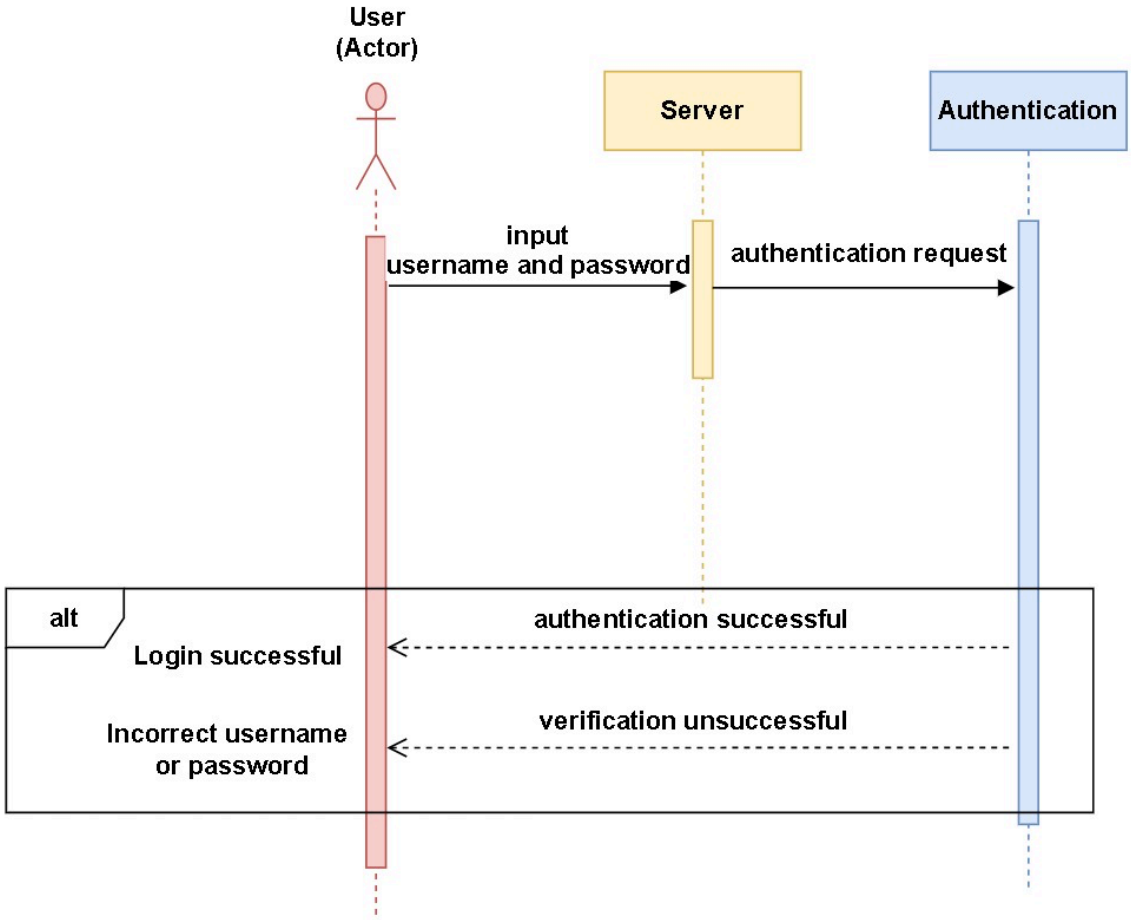


3.3 Sequence Diagrams

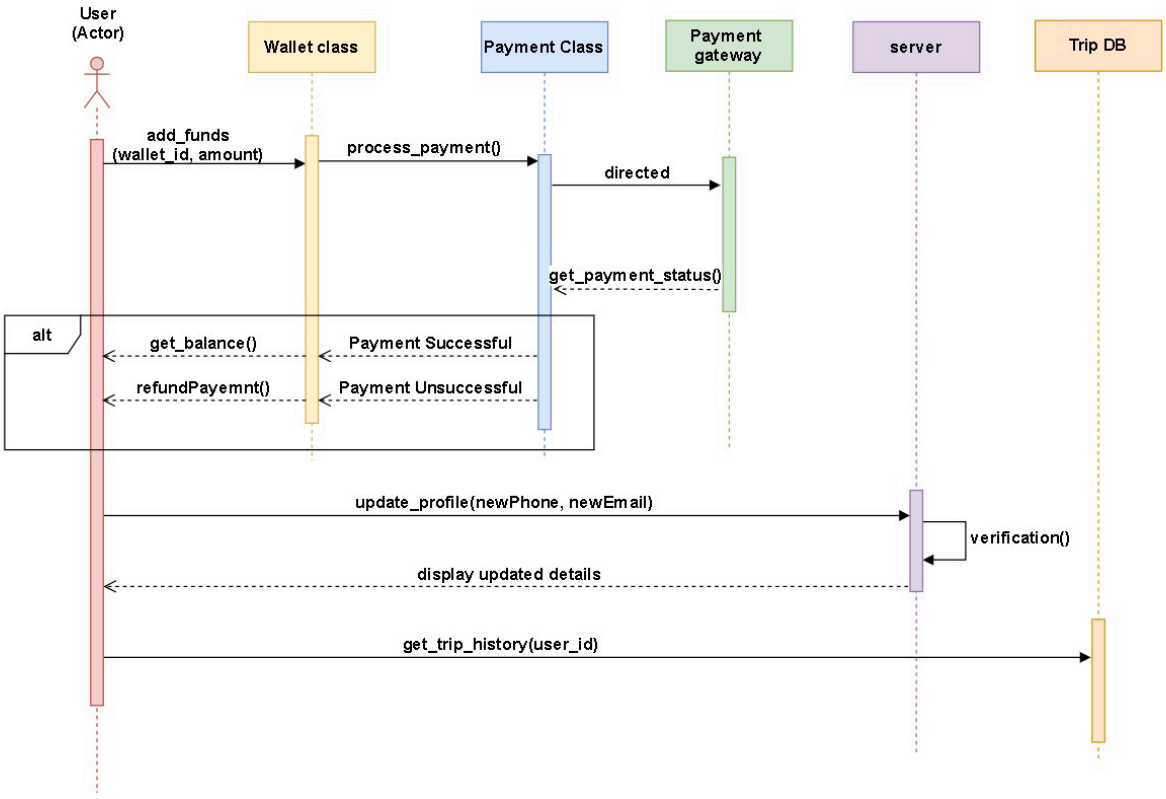
3.3.1 Sign Up



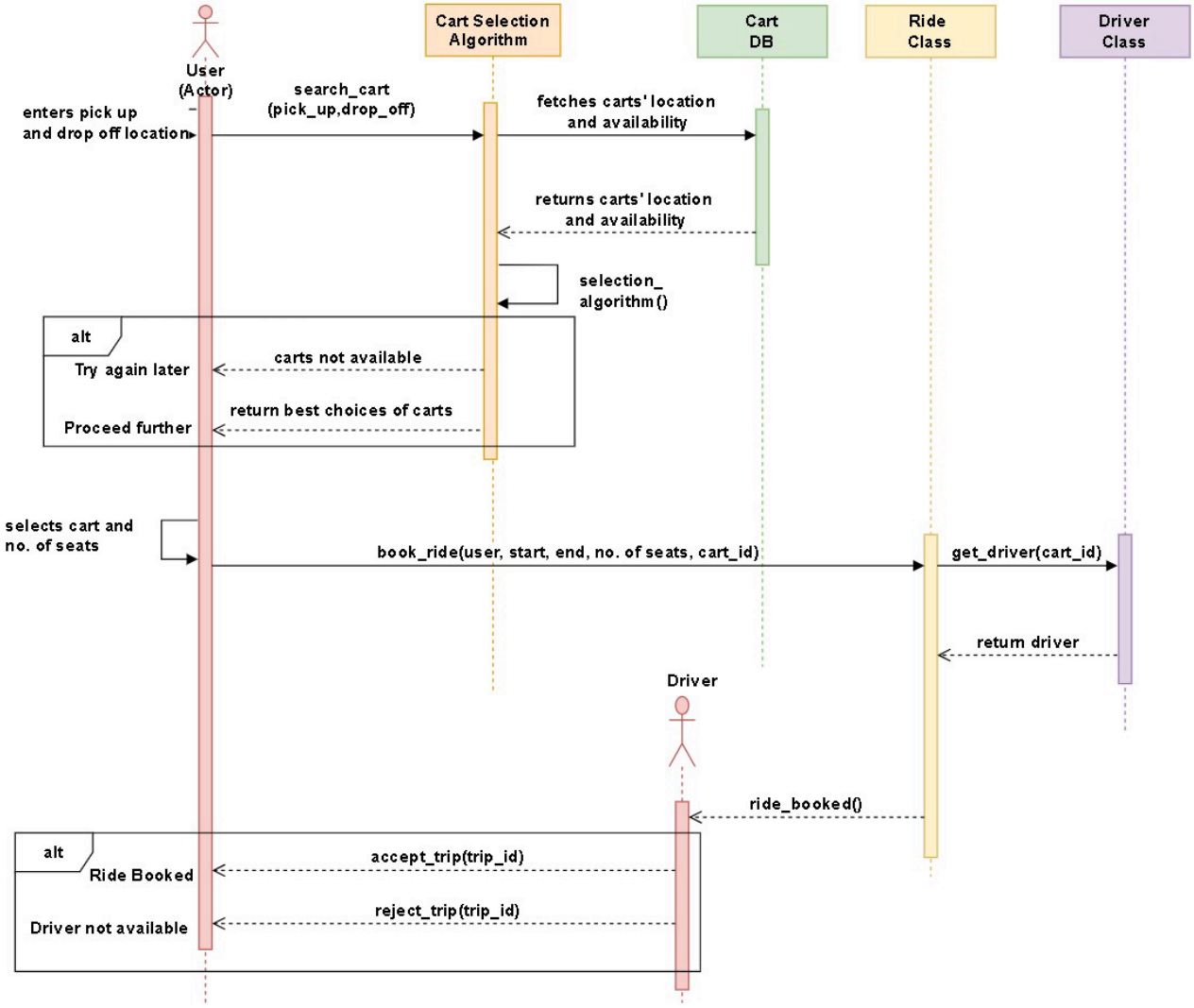
3.3.2 Log in


User
(Actor)
Server
Authentication
input
username and password
authentication request
alt
Login successful
authentication successful
Incorrect username
or password
verification unsuccessful

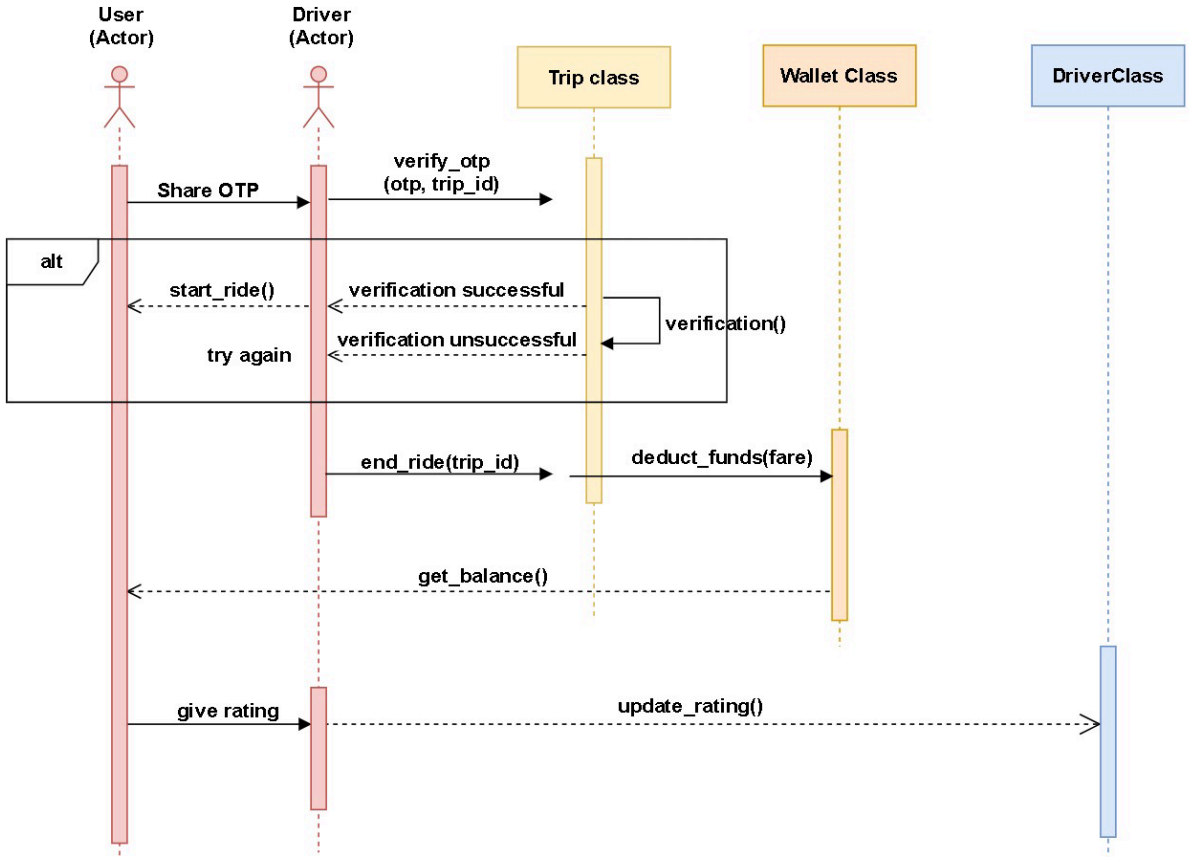
3.3.3 User


User
(Actor)
Wallet class
Payment Class
Payment
gateway
server
Trip DB
add_funds
(wallet_id, amount)
process_payment()
directed
get_payment_status()
alt
get_balance()
Payment Successful
refundPayemnt()
Payment Unsuccessful
update_profile(newPhone, newEmail)
verification()
display updated details
get_trip_history(user_id)

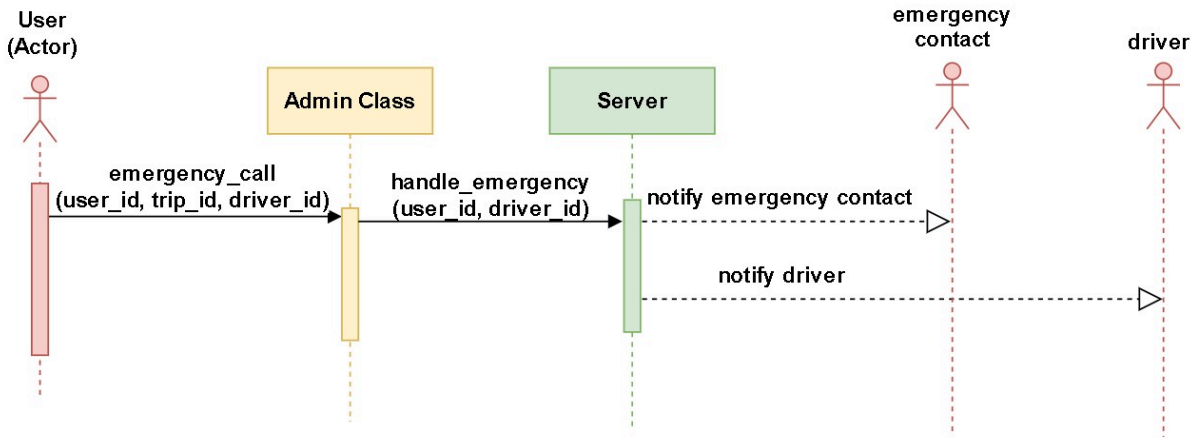
3.3.3 Ride Booking



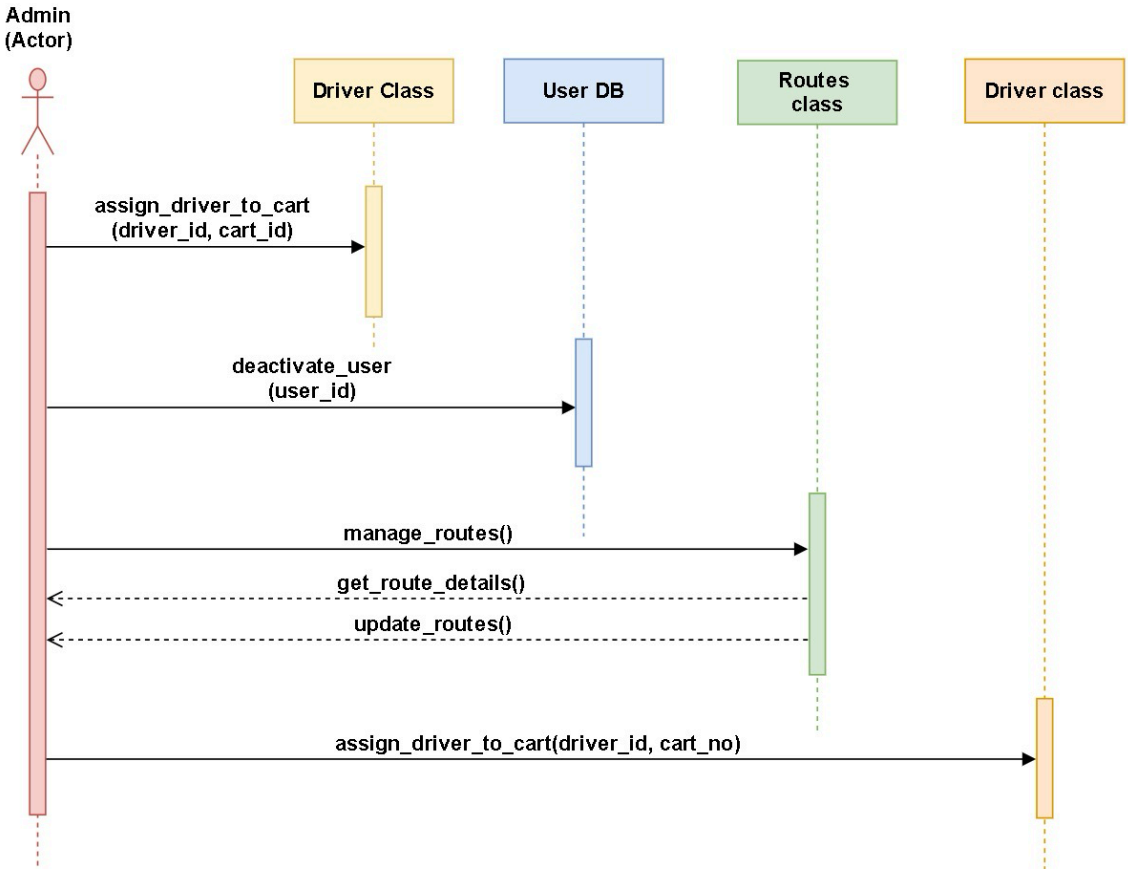
3.3.5 Ride Process and Payment



3.3.6 Emergency Situation

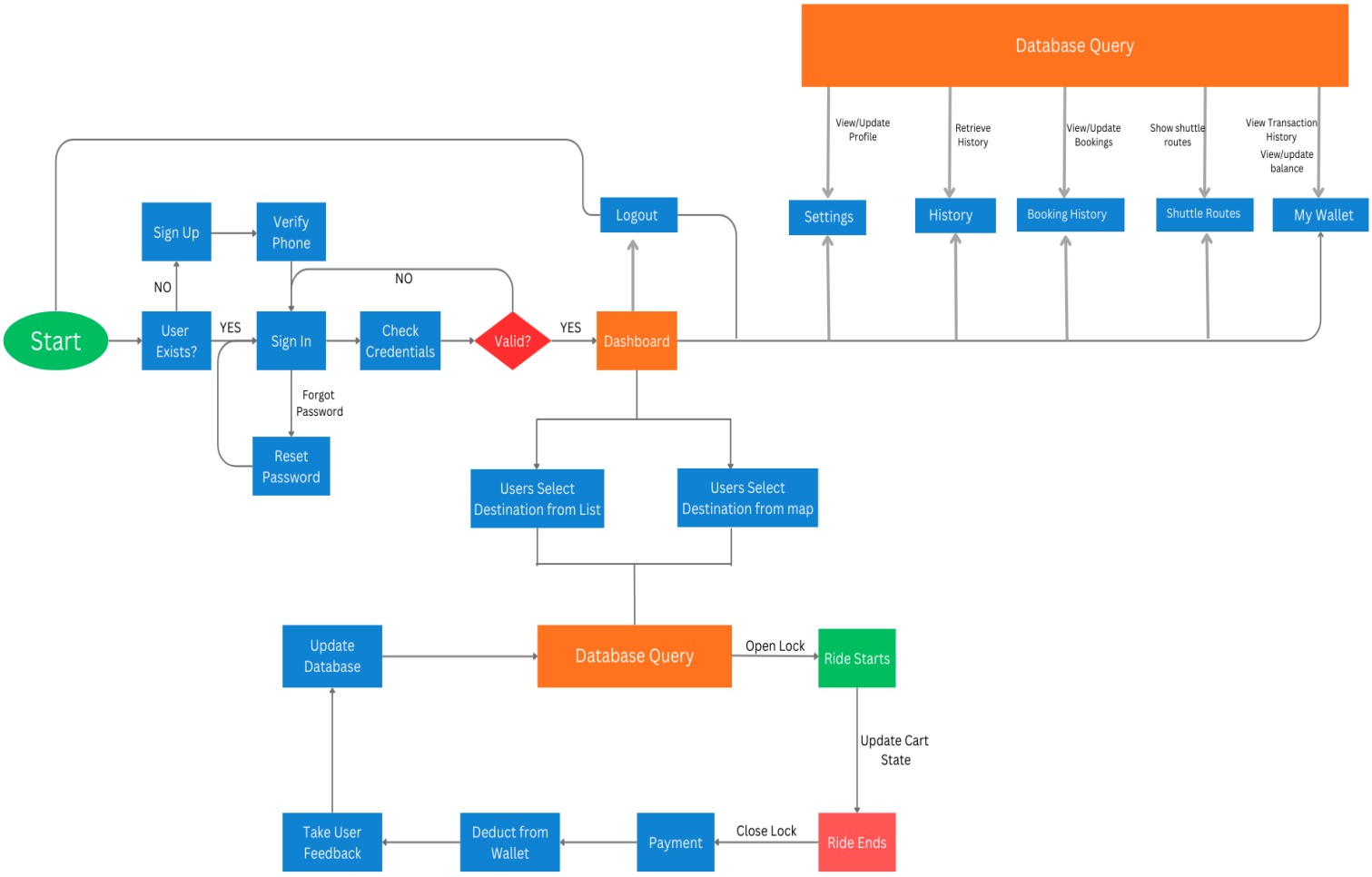


3.3.7 Admin


Admin
(Actor)
Driver Class
User DB
Routes
class
Driver class
assign_driver_to_cart
(driver_id, cart_id)
deactivate_user
(user_id)
manage_routes()
get_route_details()
update_routes()
assign_driver_to_cart(driver_id, cart_no)

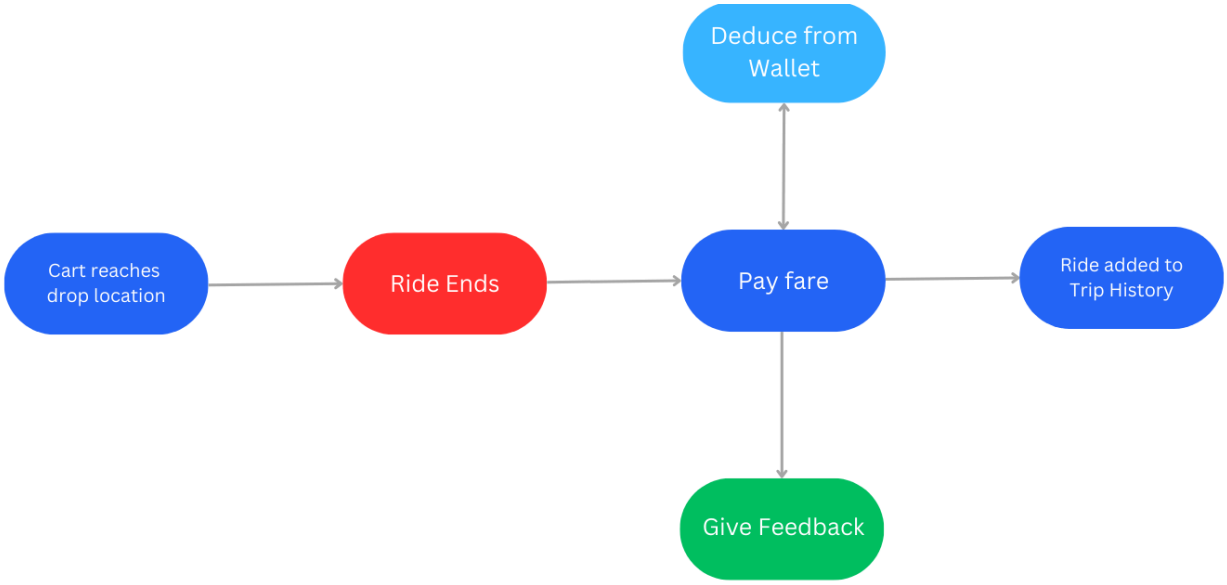
3.4 State Diagrams

3.4.1 Overall State Diagram

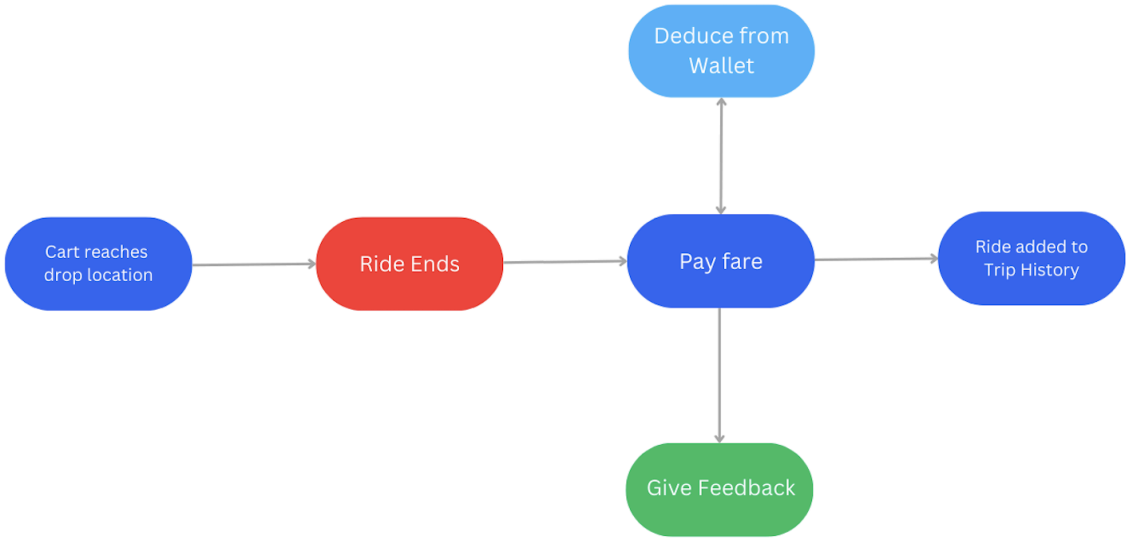


The image describes the high-level state diagram of the application from a user perspective. The user can be an existing user or a new user. The user can book a ride, view ride history, view wallet, view settings, view booking history, and give feedback.

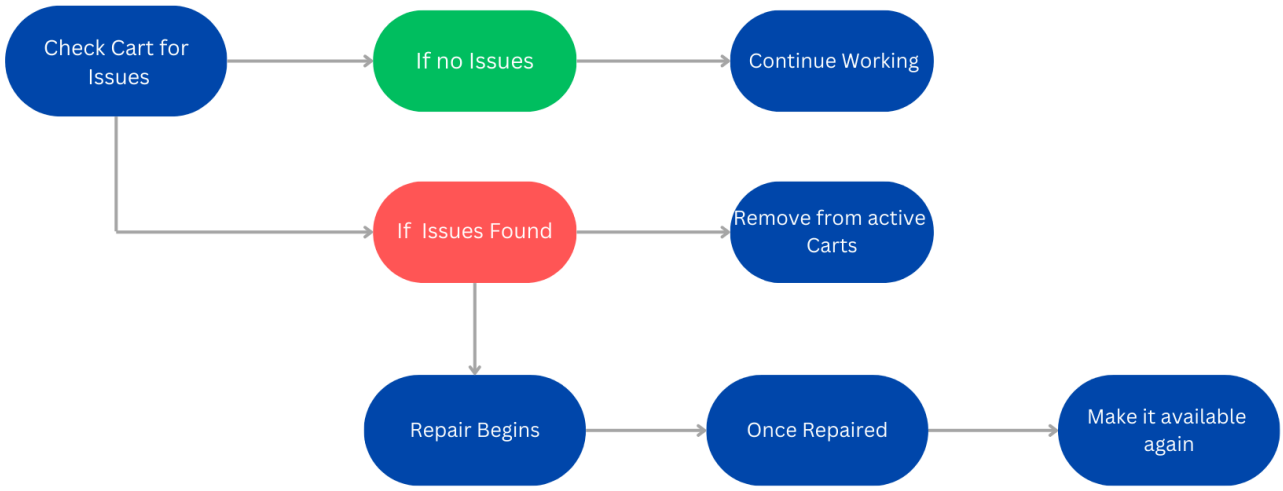
3.4.2 Ride Begin State Diagram



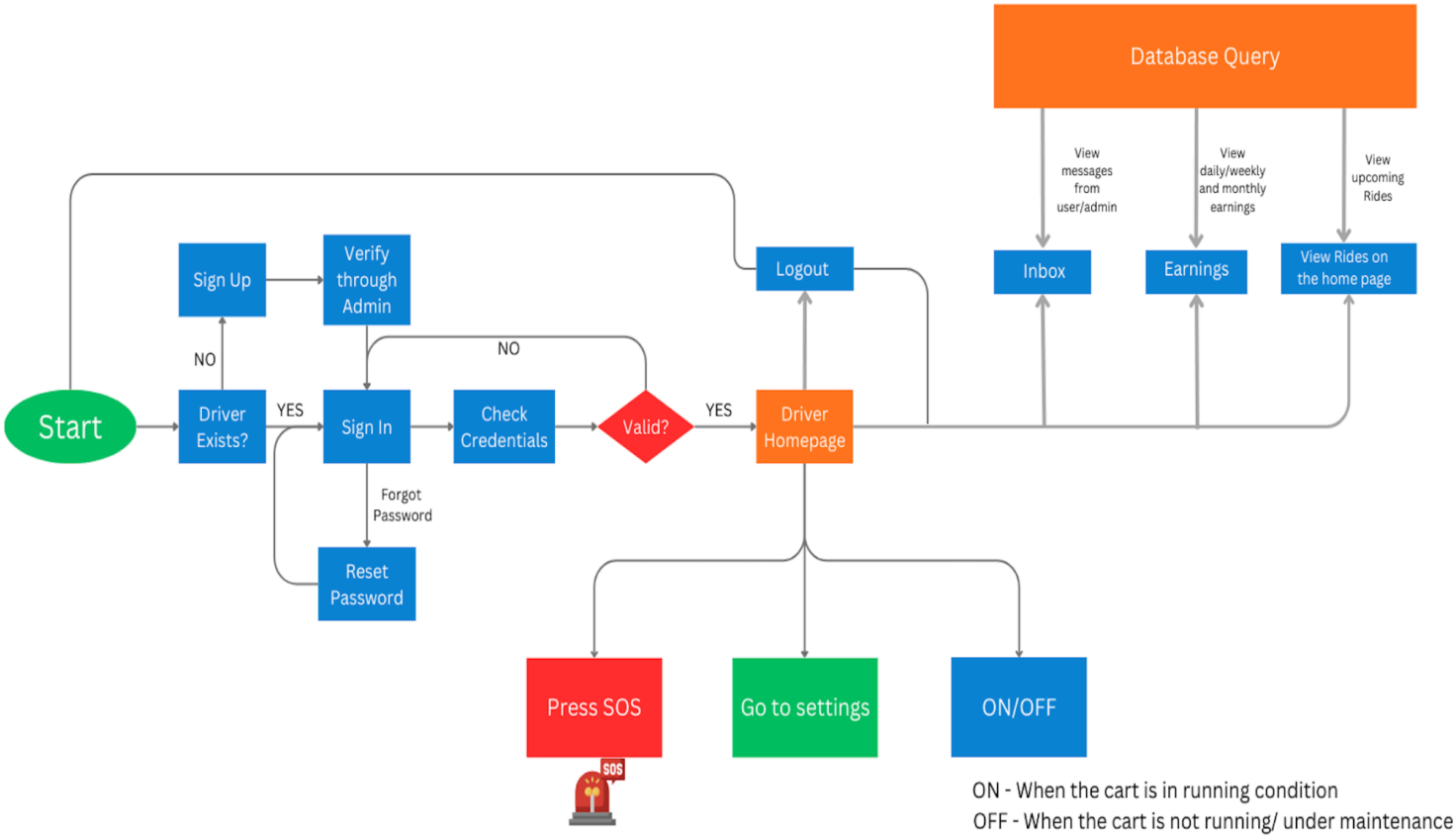
3.4.3 Ride Ends State Diagram



3.4.4 Maintenance State Diagrams

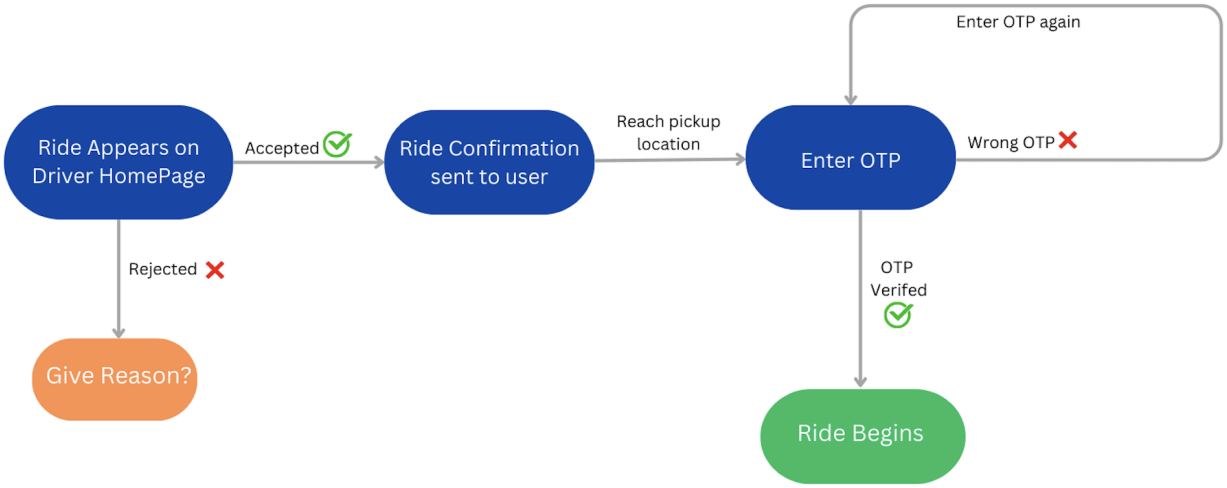


3.4.5 Driver State Diagram



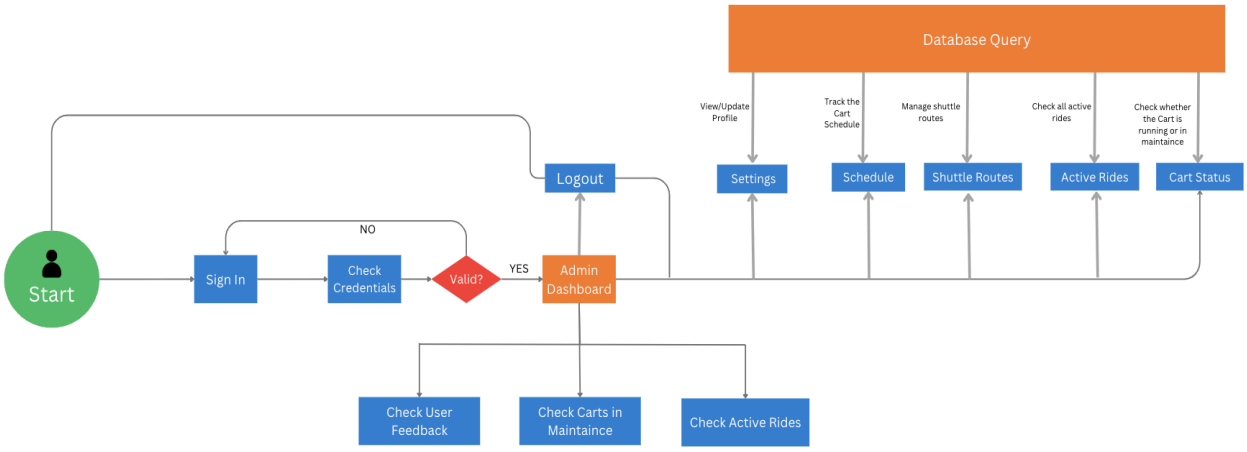
The above image describes the high-level state diagram of the application from a driver's perspective. The driver can either accept rides or reject them based on the availability and condition of the carts. The driver can see his total rides of the day/week/month, view ride history, view total money earned, view settings and put the cart to ON/OFF state depending upon whether the cart is running or in maintenance.

3.4.6 Driver Ride Acceptance State Diagram



3.4.7 Admin State Diagram

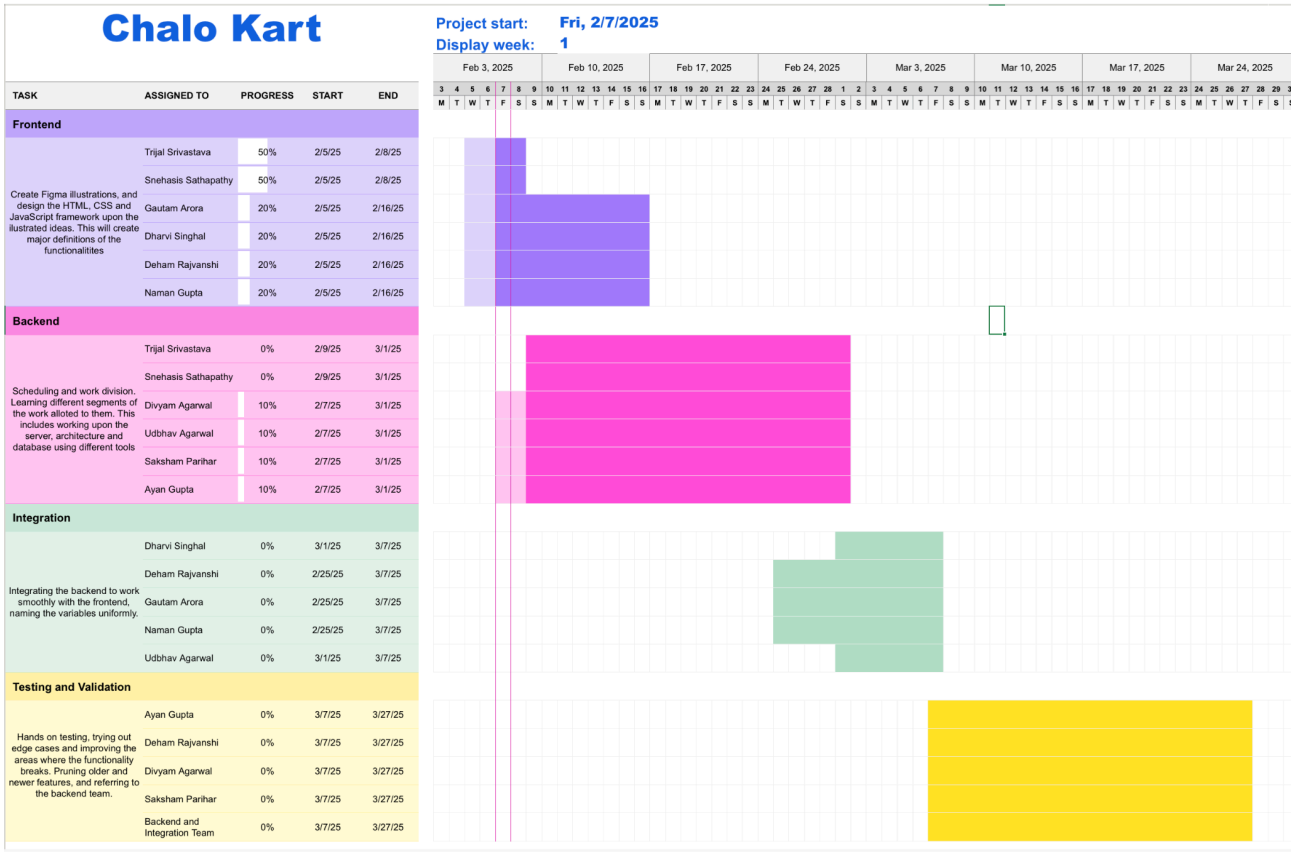
This state diagram represents the admin journey in a golf cart management system. The process starts with admin sign-in and credential verification. If authentication is successful, the admin accesses the dashboard, where they can monitor user feedback, check carts under maintenance, and track active rides. The system integrates with a database to manage cart schedules, shuttle routes, ride statuses, and conditions. Admins can also update their profiles, log out, or modify system settings. This structured workflow ensures efficient monitoring and management of the golf cart operations.



Project Plan

Project Planning

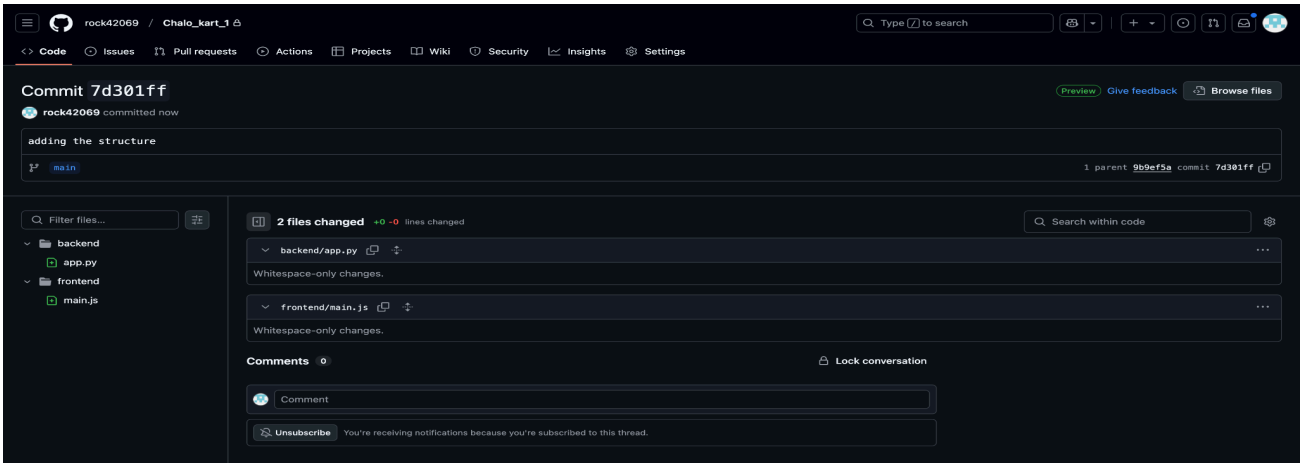
The proposed plan, with division and deadlines, are presented through this Gantt Chart



TASK	ASSIGNED TO	PROGRESS	START	END
Frontend				
Create Figma illustrations, and design the HTML, CSS and JavaScript framework upon the illustrated ideas. This will create major definitions of the functionalities	Trijal Srivastava	50%	2/5/25	2/8/25
	Snehasis Sathapathy	50%	2/5/25	2/8/25
	Gautam Arora	20%	2/5/25	2/16/25
	Dharvi Singhal	20%	2/5/25	2/16/25
	Deham Rajvanshi	20%	2/5/25	2/16/25
	Naman Gupta	20%	2/5/25	2/16/25
Backend				
Scheduling and work division. Learning different segments of the work alloted to them. This includes working upon the server, architecture and database using different tools	Trijal Srivastava	0%	2/9/25	3/1/25
	Snehasis Sathapathy	0%	2/9/25	3/1/25
	Divyam Agarwal	10%	2/7/25	3/1/25
	Udbhav Agarwal	10%	2/7/25	3/1/25
	Saksham Parihar	10%	2/7/25	3/1/25
	Ayan Gupta	10%	2/7/25	3/1/25
Integration				
Integrating the backend to work smoothly with the frontend, naming the variables uniformly.	Dharvi Singhal	0%	3/1/25	3/7/25
	Deham Rajvanshi	0%	2/25/25	3/7/25
	Gautam Arora	0%	2/25/25	3/7/25
	Naman Gupta	0%	2/25/25	3/7/25
	Udbhav Agarwal	0%	3/1/25	3/7/25
Testing and Validation				
Hands on testing, trying out edge cases and improving the areas where the functionality breaks. Pruning older and newer features, and referring to the backend team.	Ayan Gupta	0%	3/7/25	3/27/25
	Deham Rajvanshi	0%	3/7/25	3/27/25
	Divyam Agarwal	0%	3/7/25	3/27/25
	Saksham Parihar	0%	3/7/25	3/27/25
	Backend and Integration Team	0%	3/7/25	3/27/25

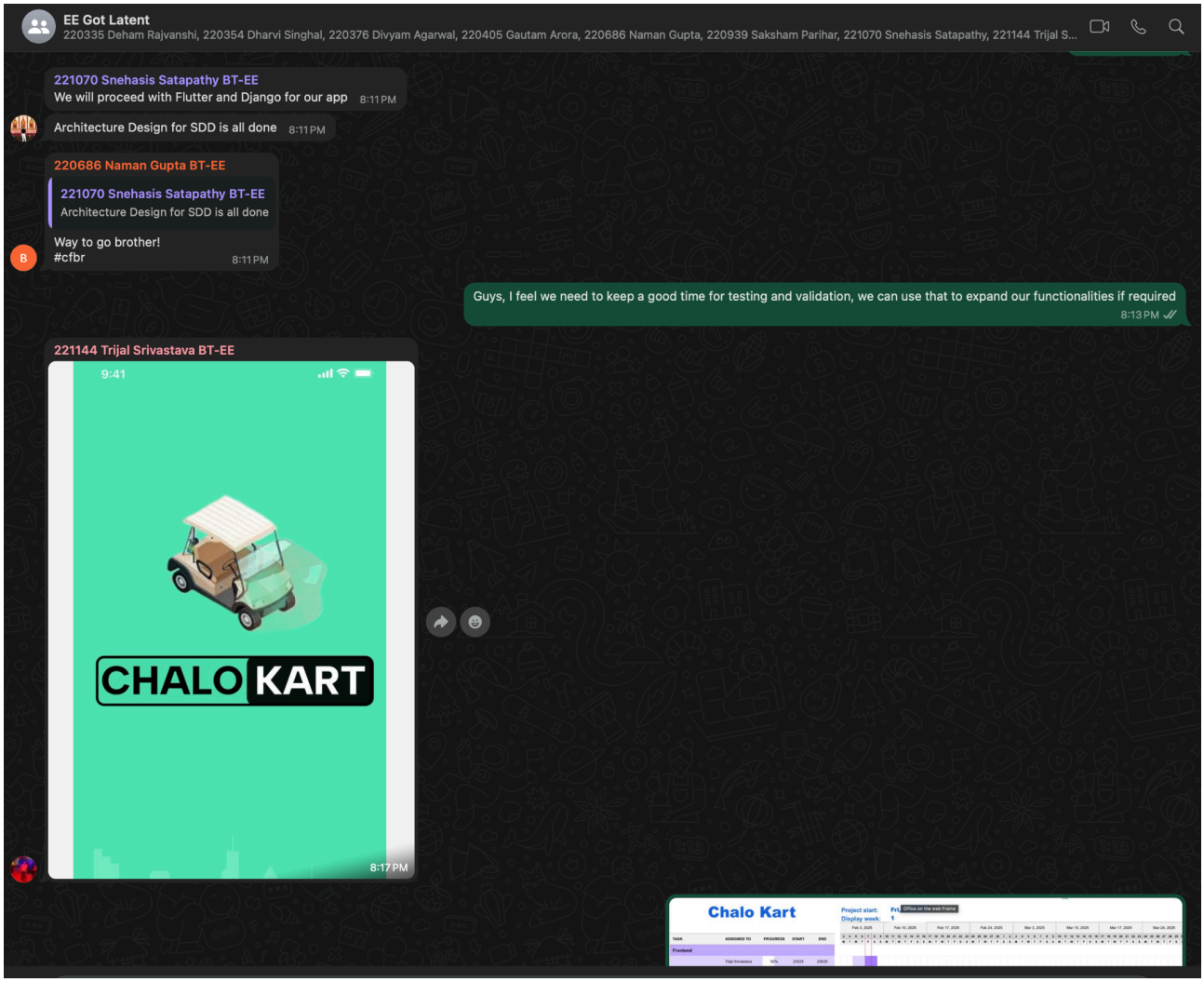
Code Collaboration

GitHub repository was made, and all of us were added as contributors



Communication

We had previously created a WhatsApp Group for all our formal discussions, share G-Meet links and other ideas.



Work Division

Team Member	Responsibility
Trijal Srivastava	Backend, Integration, UI/UX Design
Snehasis Satapathy	Backend, Integration, UI/UX Design
Gautam Arora	Frontend, Integration, UI/UX Design
Naman Gupta	Frontend, Integration, UI/UX Design
Divyam Agarwal	UI/UX Design, Database Management
Udbhav Agarwal	Backend, UI/UX Design, Testing
Deham Rajvanshi	Frontend, Testing
Ayan Gupta	Database Management, Documentation, Testing
Saksham Parihar	UI/UX Design, Backend, Testing
Dharvi Singhal	Frontend, Integration, Testing

Appendix A - Group Log

S.No.	Date	Time	Venue	Description
1	28/01/2025	21:30 - 22:30	Hall 12	Deliberate upon finer software details the number of UIs that will be needed. Referred to lectures to understand the different Architectures
2	31/01/2025	18:00 - 19:00	CCD	Went through the Design document and the deliverables. Divided amongst ourselves the different sections and a segment of the final code to learn in teams of 2.
3	02/02/2025	15:00 - 17:00	CCD	Since it was a Sunday, it was a light-hearted discussion, and we took an update from each sub-team, made finer adjustments to designs, and had some coffee together.
4	06/02/2025	22:00-23:00	Google Meet	Collected the deliverables of each sub-team, integrated the documents, and made formatting adjustments
5	07/02/2025	22:00 - 23:00	Google Meet	Proofreading of the Final Design Document and some changes were made to ensure it adheres to the rules of the Design Document.