Software Requirements Specification

for

Chalo Kart
(Golf Cart Management)

Version 1.0

Prepared by

Group 13:		Group Name: EE Got Latent
Trijal Srivastava	221144	trijals22@iitk.ac.in
Snehasis Satapathy	221070	ssnishasis22@iitk.ac.in
Gautam Arora	220405	garora22@iitk.ac.in
Naman Gupta	220686	namangupta22@iitk.ac.in
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

Mentor TA: Mr. Vikrant Chauhan

Date: January 24, 2025

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 Singhal	The first version of the Software Requirement Specification (SRS) Document was once presented to the TA for proofreading.	24/01/25

1 Introduction

1.1 Product Scope

The recent inaugural golf cart service on campus has been a great initiative by the authorities. The issue of not finding any vehicle to travel from one place to another within the campus has been prevalent for a long time. The golf carts offer great relief in this regard, but the issue is that they are currently not managed, and it's very hard to trace their location. Also, the carts currently operate on some fixed route that is not known by everyone, and this causes a lot of inaccessibility issues to the commuters, making the carts of no use to the mast campus audience.
The **Chalo Kart** project presents a golf cart management system to transform IIT Kanpur's campus transportation. Our solution guarantees a smooth user experience for both residents and guests by easing out the golf cart reservation and administration process. Its two booking alternatives, Private Carts for custom routes according to the user and Shuttle Carts for fixed-route transportation, accommodate a range of user preferences while improving efficiency and convenience. The system's GPS monitoring, real-time updates, and user-friendly interface reduce operational snags and promote sustainable movement within the school environment.

1.2 Intended Audience and Document Overview

1.2.1 Intended Audience:

This document is intended for developers, project managers, testers, and users involved in designing, implementing, and maintaining the Chalo Kart system. It outlines the functional and non-functional requirements, interface details, and use cases.
For Developers: We have laid down the functional and non-functional requirements the developers will try to build.
For Testers: This document will test out all our software's functionalities.
For Project Managers: They will be able to understand the intent and the features the development team will be working on, which, in our case, will be the course instructors and the TAs.
For Users: they will be able to explore the application to its fullest once they go through the app and can appreciate the intricacies.

1.2.3 Document Overview:

Introduction:
This section provides some basic information and our vision that would be useful in reading the SRS and setting the context right. We have also included the document conventions, abbreviations, etc. Readers may skip the section if they are familiar with the basic terminologies.

Overall Description:

This section offers an overall view of the software system and its functionalities, assumptions, and dependencies. This will be a useful read for those seeking to familiarise themselves with the system at a glance. Readers are encouraged to read this part as it provides a good basis for understanding the next section of the SRS.

Specific Requirements:
This section contains detailed information about the software and its functionalities. We have also included various usercase using numerous tree diagrams. This proves indispensable for end-users, clients, and developers alike, serving as a roadmap during the development phase and a user manual for end-users.

Other Non-Functional Requirements:
Important non-functional requirements are expounded here. This is of special importance to the software developers.

1.3 Definitions, Acronyms and Abbreviations

SRS	*Software Requirements Specification*
UI	*User Interface*
API	*Application Programming Interface*
GPS	*Global Positioning System*
OTP	*One-Time Password*
MERN	*MongoDB, Express.js, React, Node.js*
ETA	*Estimated Time of Arrival*

1.4 Document Conventions

This document is drafted using **Google Docs** in **.docx** format. The following conventions are followed to ensure consistency and readability:

- **Font**: Arial
- **Font Size**: 12
- **Text Alignment**: Left-aligned
- **Line Spacing**: Single
- **Margins**: 1 inch on all sides
- **Section Titles**: Bold and numbered according to the hierarchy (e.g., 1, 1.1, 1.1.1).
- **Key Terms**: Italicized for emphasis when introduced.
- **Tables and Figures**: Labeled with captions and referenced in the text.

1.5 References and Acknowledgments

- We would like to acknowledge the help of our TA, Mr. Vikrant Chauhan, and our course instructor, Prof. Indranil Saha, for guiding us through the document and providing a template for the Software Requirements Specification document.
- We used Figma to craft visually compelling graphs, translating our ideas into a concise and impactful pictorial representation.
- The following book was consulted for background information and concepts related to Software Engineering.

  Roger S. Pressman, Software Engineering: A Practitioner's Perspective, McGraw Hills Publications, Seventh Edition, 2010

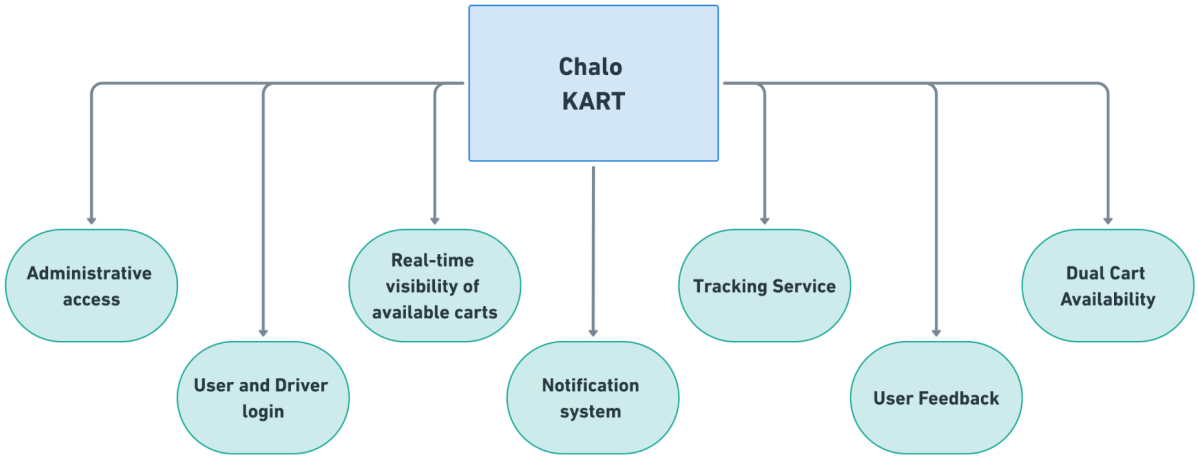
2 Overall Description

2.1 Product Overview

Our product, Chalo Kart, enhances the transportation experience on the IIT Kanpur campus by providing a convenient and effective golf cart management system. The product's primary purpose is to make it easy to schedule, track, and manage golf carts, a popular means of transportation on campus. The product replaces manual or semi-digital processes and eliminates inefficiencies, ensuring a modern, user-friendly, and robust solution. Therefore, it would be an independent and scalable solution designed to fit seamlessly into the campus's existing transportation ecosystem.

Chalo Kart would further enhance the booking system by introducing a dual golf cart booking system. This system caters to diverse user needs with two distinct cart options: **Shuttle Carts,** ideal for budget-friendly, high-capacity transportation along fixed routes with pre-defined hub spots and **Private Carts,** which are tailored for users seeking more personalised travel routes. Users specify their starting and ending locations, and the algorithm determines the shortest path between them.

Chalo Kart uses smart route planning, GPS tracking, and an easy-to-use website to make booking and travelling simple and hassle-free. By offering two types of carts, the system makes sure resources are used efficiently while meeting users' different needs and budgets.



2.2 Product Functionality

- Provide **Dual Cart Booking** Option
    - Shuttle Carts (10-seaters) with fixed routes and pre-defined hub spots.
    - Private Carts (5-seaters) with flexible routes based on user-defined start and end locations.
- Includes **administrative access** with a secure login for managing cart operations and maintenance schedules.
- Displays real-time cart availability and nearby hub spots using **GPS integration** and Google Maps API.
- Offers a **notification system** to inform users about cart arrivals, ride updates, and other announcements.
- Allows users to submit feedback on rides for service improvement.
- Provides access to detailed analytics, including ride history, payment records, and usage statistics.

2.3 Design and Implementation Constraints

- To provide real-time information about cart locations, ETAs, and availability at hubs, implementing efficient communication protocols between the server, GPS-enabled carts, and user devices is essential.
- The system must handle simultaneous booking requests for both shuttle and custom carts, ensuring smooth operations during peak usage times
- The system must be capable of scaling as the number of carts, users, and booking requests increases over time.
- Reliable internet connectivity is critical for real-time tracking, updates, and user-server interactions.

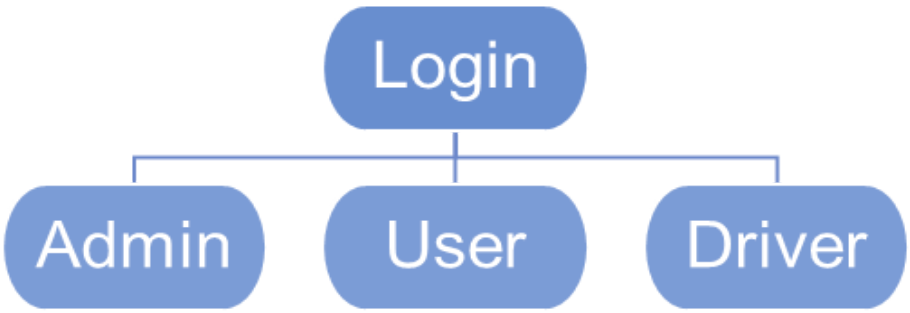
2.4 Assumptions and Dependencies

- The system depends on drivers to follow the shortest routes suggested by the algorithm and to adhere to scheduled pickups and drop-offs.
- For shuttle carts, the assumption is that predefined hub spots and their routes will remain fixed unless updated manually.
- The system assumes that user and driver devices (e.g., smartphones or tablets) will support the web application and can access location services for tracking and navigation.

3 Specific Requirements

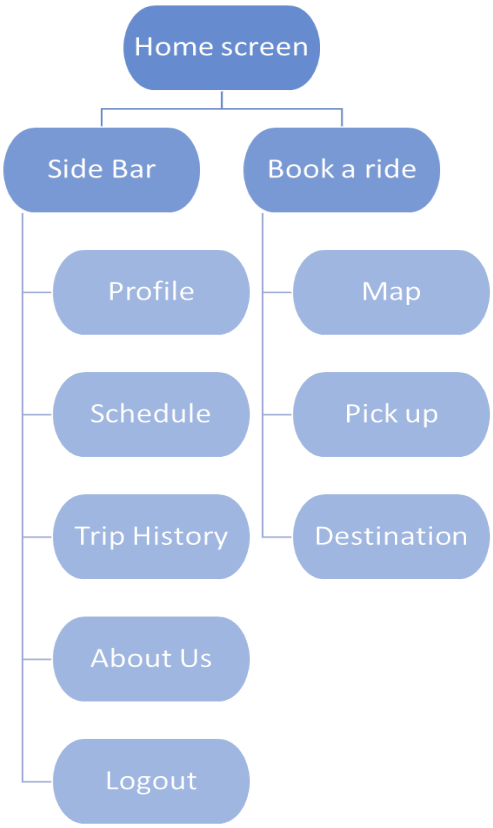
3.1 External Interface Requirements

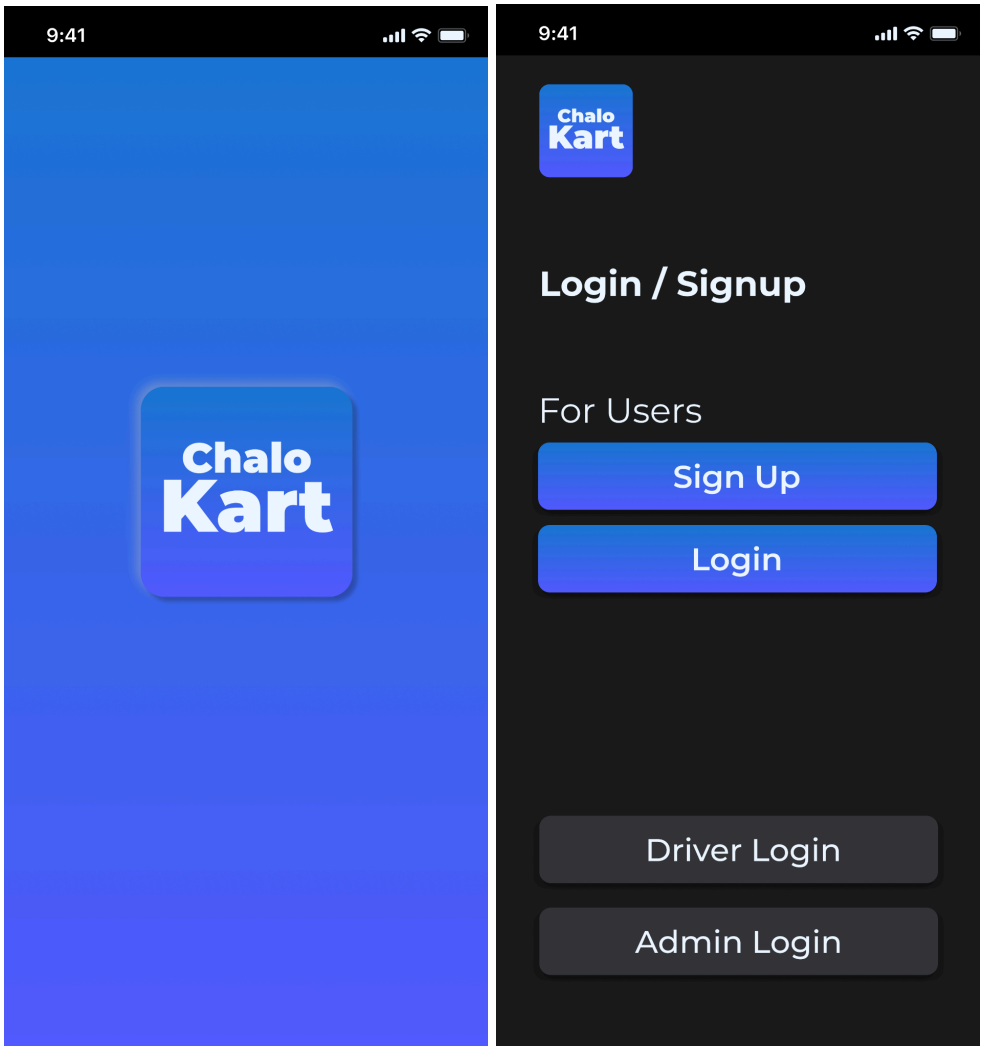
3.1.1 User Interfaces



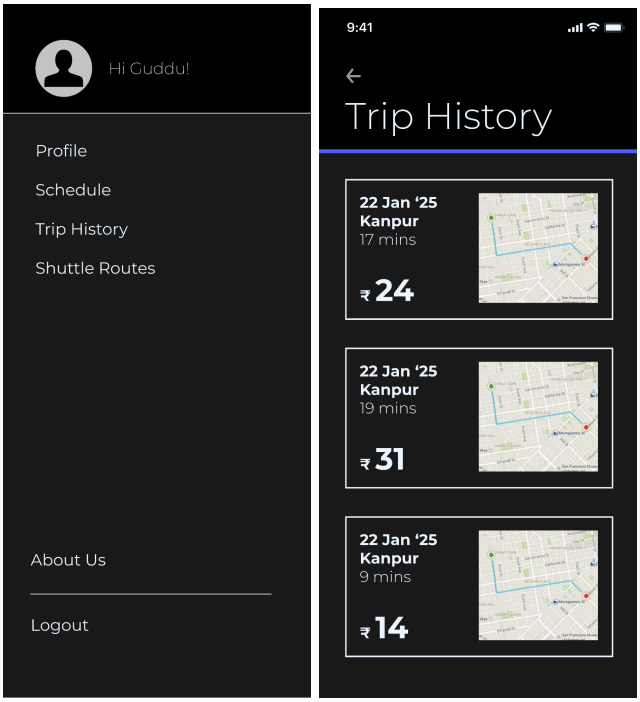
Our application offers a login interface for users and rivers and an admin login. New users must register on the website and authenticate their phone number and email.

To log in, users can use a password or login via OTP on their registered phone no. Driver and Admin login are 2 other options available, but there will be no option to sign up.

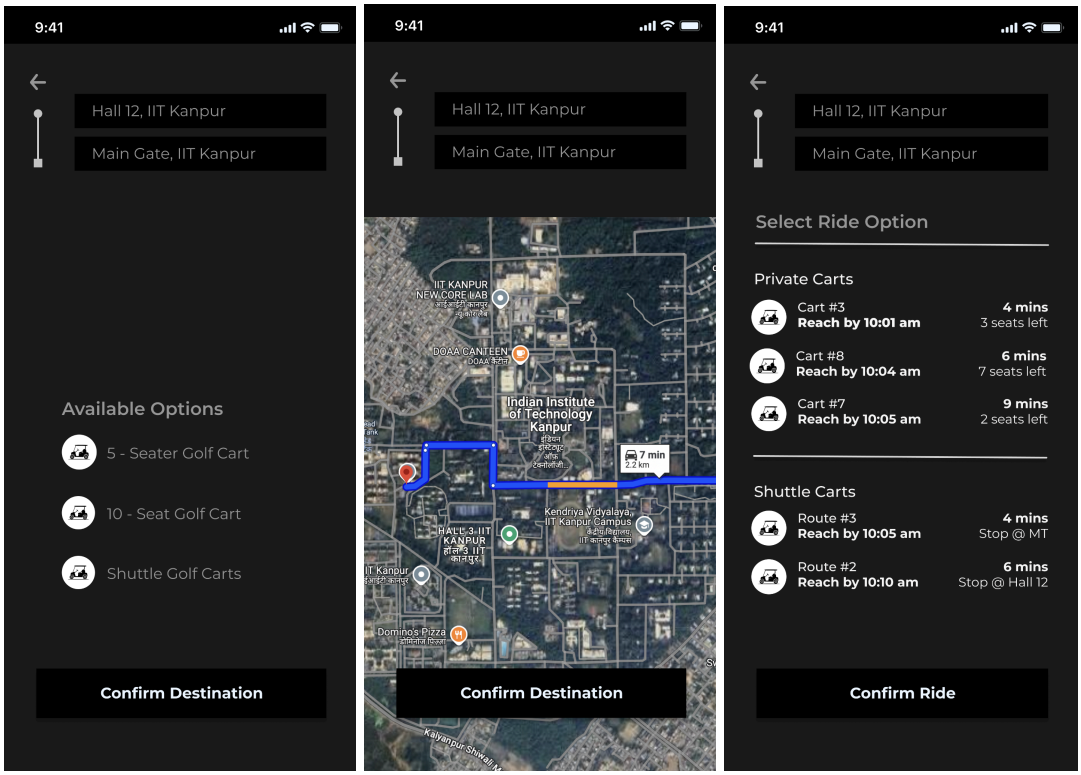




On the sidebar, we have options of Profile, Schedule, Trip History, Shuttle Routes, and an option to log out.



On the home screen, you will see a map and need to put in the start and end locations. Confirming the route will show all the possible cart options. It has 2 categories: Private Carts and Shuttle Carts. The estimated time of reaching the destination, seats left, golf cart number, etc., will be displayed.



3.1.2 Hardware Interfaces

- User devices: The software would be accessible on internet-enabled devices, including smartphones, tablets, and laptops, allowing users to book golf carts easily and conveniently.
- Driver devices: The drivers could use the software on internet-enabled devices like smartphones, tablets, and laptops, allowing them to accept the rides, use the map displayed to navigate to drop-off points, and mark the rides complete.

3.1.3 Software Interfaces

- The client-side components will function on all modern web browsers and mobile versions, providing responsive support like Apple Safari 7+, Google Chrome 44+, Microsoft Edge 90+, and Mozilla Firefox 40+.
- We will use multiple databases for storing the following:
    - User Database: Manages all user profiles and login credentials.
    - Driver Database: Maintains driver details and tracks performance and activity.
    - Ride Database: Captures real-time ride information for billing and analytics.
    - Golf Cart Database: Tracks operational readiness and maintenance needs of carts.
    - Payments Database: Ensures secure and transparent transaction records.
    - Stops Database: Provides static locations for route planning and optimisation.
    - Booking Database: Handles bookings for better scheduling and resource allocation.

  We shall be using MongoDB as our DBMS.

3.2 Functional Requirements

3.2.1 Administrative access using username and password

- The admin will be able to assign drivers to the carts and look at the analytics of drivers currently operating and the routes on which they are working.
- The admin can look at the emergency requests made by the users travelling in the cart.
- The admin will also be able to see the live location of the golf carts using the driver's location and will be able to track and enforce the routes.
- The admin can look at the analytics, such as total rides, total distance covered, etc.
- The admin can also deactivate and change or update the users' information.

3.2.2 Provision for users to create an account

- The system allows users to create their profiles by providing essential information, passwords, IITK campus email for campus residents, and visitor contact numbers.
- After successful authentication, the system allows users to update their personal information, reset their passwords, and avail themselves of ride services.

3.2.3 Provision for driver login

- Upon login to the system, the driver can see the cart he is assigned to and the basic analytics of earnings and rides he has done.
- The driver will be able to see the cart requests being made and will be able to accept and reject them.
- He will also see the destination points, the fixed routes and schedules for the day ahead.

3.2.4 Real-time visibility of available carts

- The user can see the live locations of all the carts on the maps and will be able to see the fixed routes that are there.
- The system will get updated in real-time based on the location of carts.

3.2.5 Booking of rides

- The user selects the pickup and drop-off points by entering the location's name or by choosing on the map.
- The user will have the option of choosing the number of seats to book.
- The system selects the top 5 carts based on the availability of seats, ETA, cost, and cart type (shuttle or non-shuttle), instilling confidence in its capabilities.
- An OTP will be generated, and the user will share it with the cart driver to get the ride started.

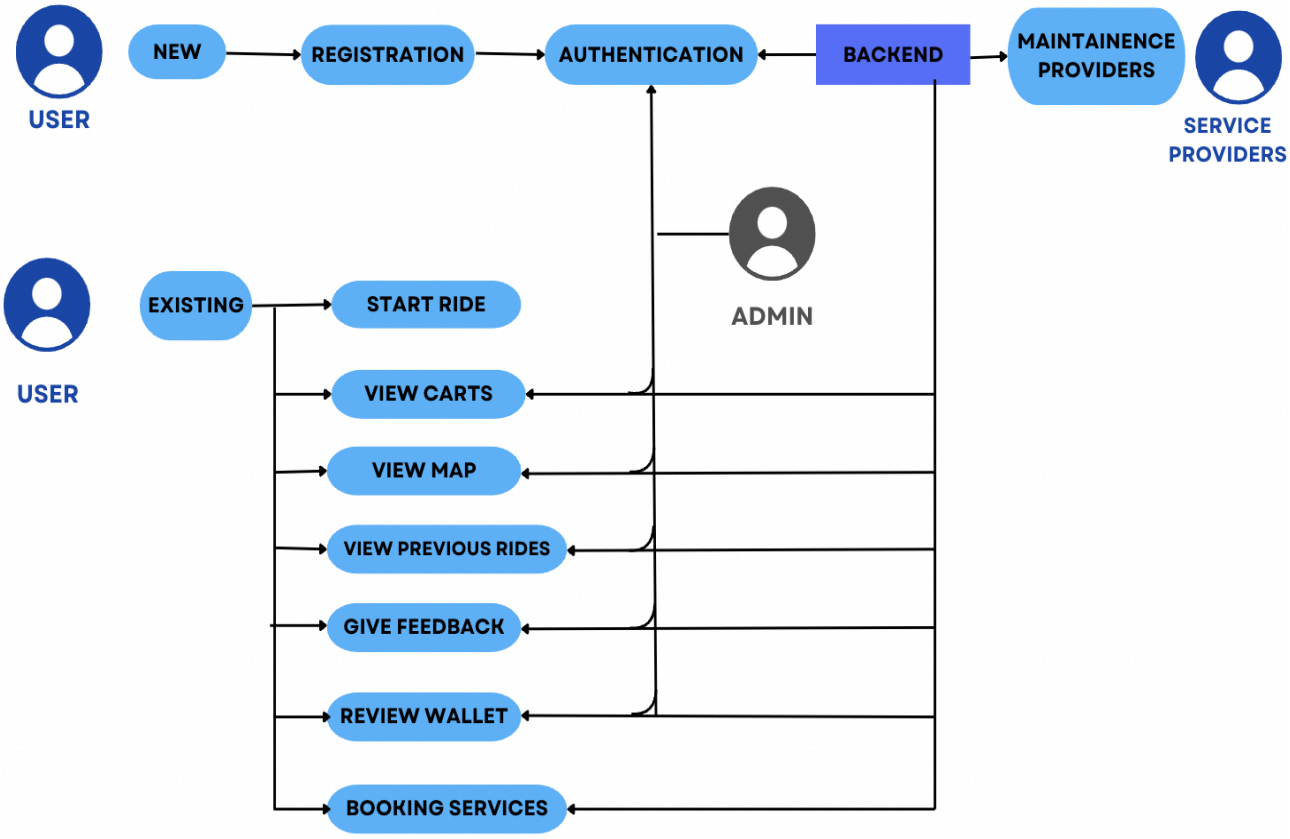
3.2.6 During the Ride

- The cart route is displayed on the map so the user can track the drive.
- An emergency button is provided to report any mishappenings during the ride.
- The driver ends the user's ride, and the user makes the payment via the RazorPay API.
- The user can give feedback for the overall ride experience by providing a star rating to the driver.

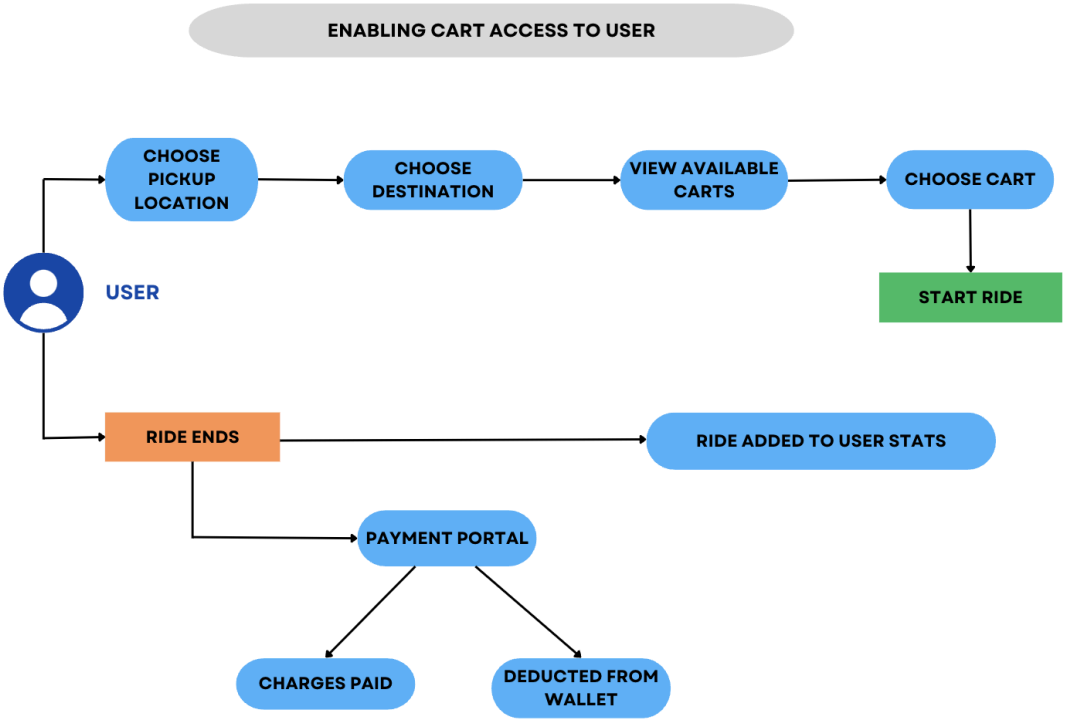
3.2.7 Notification system

- The system regularly notifies users about the latest updates and cart maintenance schedules.
- The system will show the ETA of carts and updates on the cart that you have booked for the route.

3.3 Use Case Model

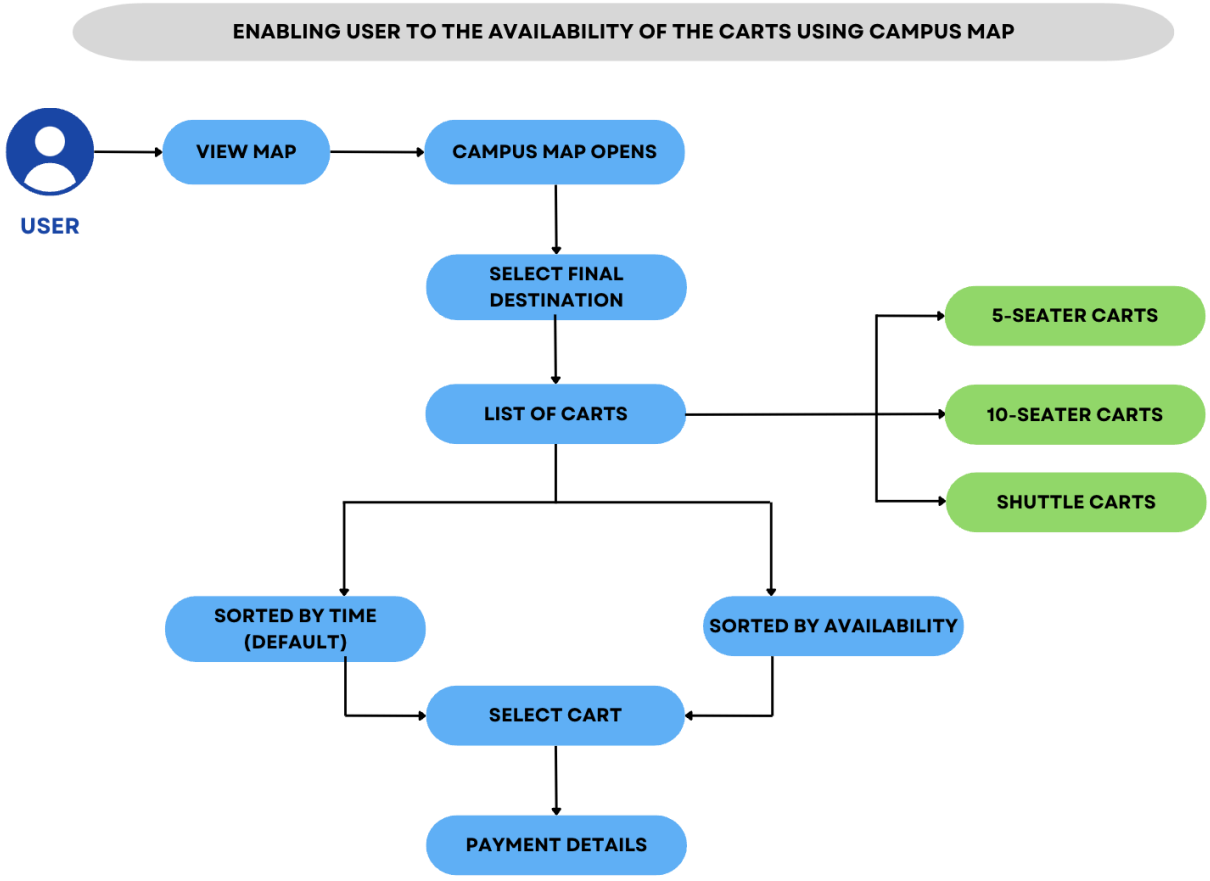


3.3.1 Use Case 1



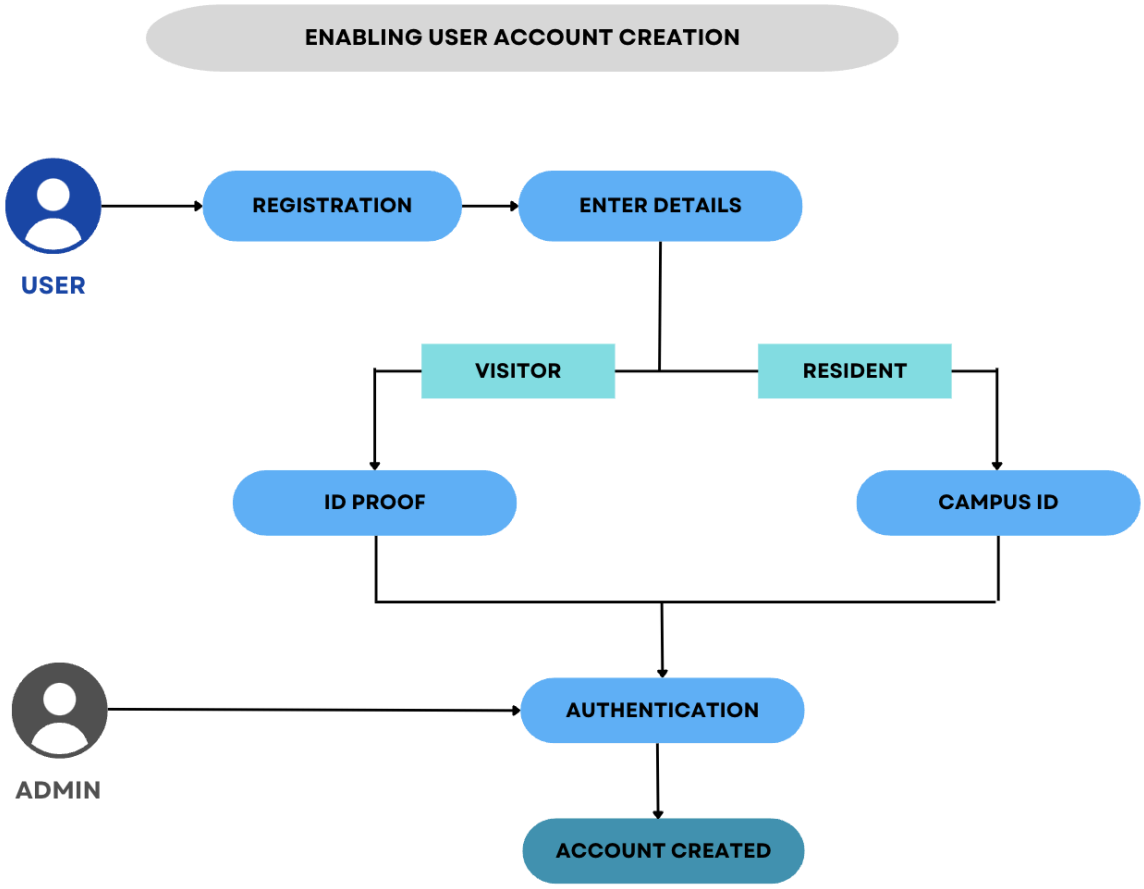
Author	Dharvi Singhal, Saksham Parihar
Purpose	To let the user book golf carts from anywhere inside the campus and charge them according to the distance travelled
Requirements Traceability	The user must be logged into the system and must have access to the internet must be physically present at the pickup location
Priority	The priority of this use case is High (as it directly supports the core functionality of the system)
Pre Conditions	The user must be physically present at the campus pickup location (station). Golf carts are available for reservation.
Post Conditions	The user reaches his/her destination location and will be charged based on the distance travelled. The system records ride details (pickup, drop-off, duration, fare, etc.) for future reference. The user receives a confirmation or receipt of the completed ride.
Actors	The actors involved in this use case are the residents/visitors who want access to golf carts to travel across the campus

3.3.2 Use Case 2



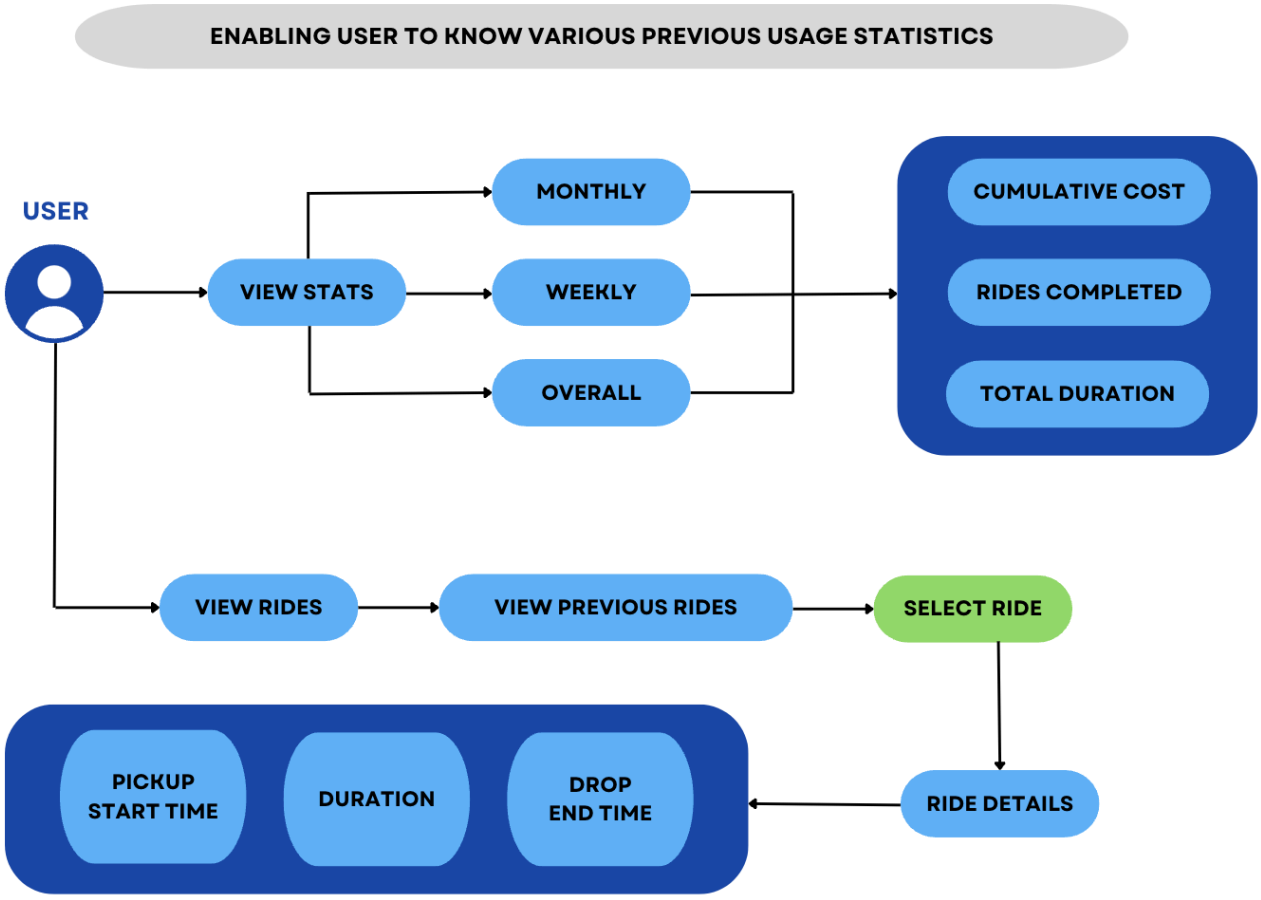
Author	Dharvi Singhal, Saksham Parihar
Purpose	To let the user know the availability and ETA of different golf carts running across the campus with the help of a campus map
Requirements Traceability	The user must be logged in and should give access to the location or enter the pickup location manually
Priority	The priority of this use case is medium, as it enhances user experience but is not essential for the core functionality of the system
Pre Conditions	The user must be physically present at the pickup location and should give access to the location or enter the pickup location manually
Post Conditions	The user is provided with information about available golf carts in his/her vicinity.
Actors	The actors involved in this use case are campus residents/visitors who want access to golf carts to travel across the campus

3.3.3 Use Case 3



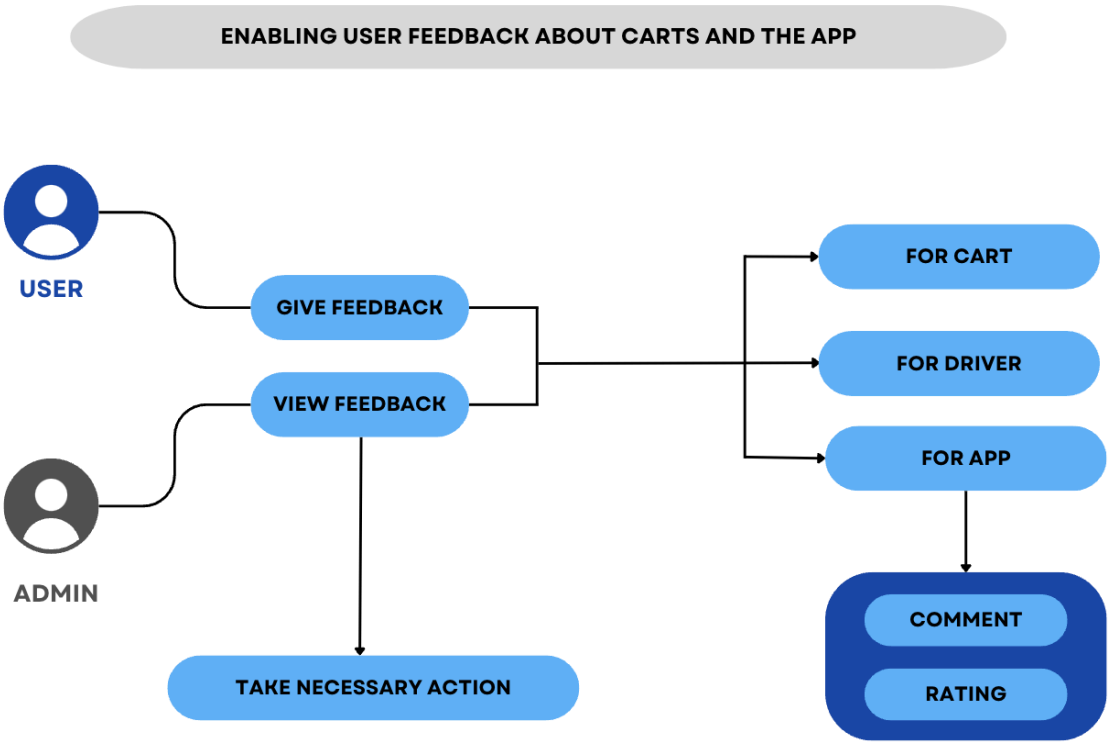
Author	Trijal Srivastava, Snehasis Satapathy
Purpose	To let the user create an account on the portal
Requirements Traceability	The user must have a valid ID Proof for the admin to authenticate him/her
Priority	The priority of this use case is High as if the users don't have an account, they cannot use the app to book golf carts
Pre Conditions	The user should enter the required details along with an ID proof for the admin to authenticate
Post Conditions	An account for the user will be created on the app
Actors	The actors involved in this use case are campus residents/visitors who want to create accounts and the admin who authenticates them

3.3.4 Use Case 4



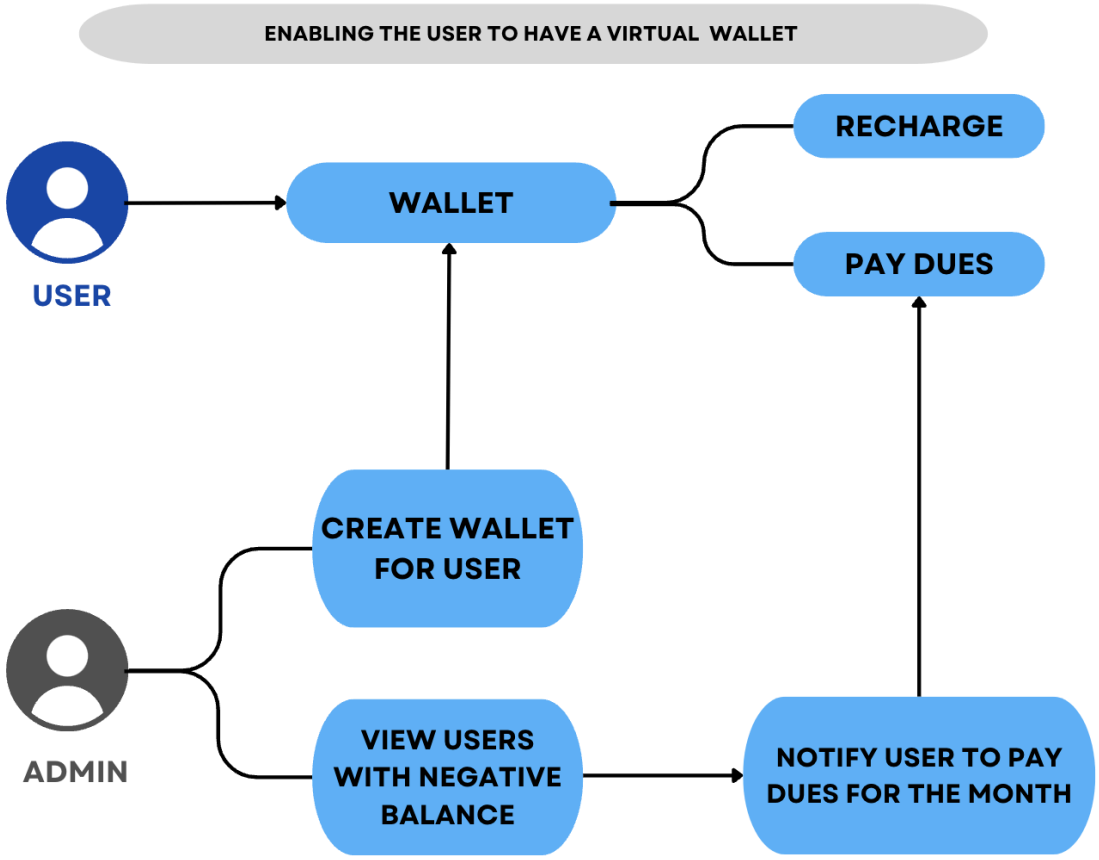
Author	Gautam Arora, Naman Gupta
Purpose	To show the user their previous rides, usage statistics like pickup location, drop location, and duration of the ride, these stats are classified into monthly, weekly and overall categories
Requirements Traceability	The user must have an account on the app
Priority	The priority of this use case is medium, as knowing the previous statistics enhances user experience and is not a major feature
Pre Conditions	The user must have completed at least one ride using their account on the application
Post Conditions	The user will get the statistics of his/her previously completed rides with the additional option of knowing the cumulative fare spent
Actors	The actors in this use case model are the users who want to know about their previous ride history and statistics

3.3.5 Use Case 5



Author	Divyam Agarwal, Udbhav Agarwal
Purpose	To let the user give feedback about the carts, the drivers and the overall experience of the app, which will help us improve the functionality of the app
Requirements Traceability	The user must have an account on the app
Priority	The priority of this use case is medium, as feedback helps us identify and eliminate shortcomings and improve user experience, and comprehensive testing of the app will be done to ensure the user finds no bugs
Pre Conditions	The user must have completed at least one ride using their account and should have given true feedback at the time of completion of the ride about their experience
Post Conditions	The app will record the feedback of the user, and the admin will take suitable actions by viewing the feedback if necessary
Actors	Actors involved in this use case are the users who provide feedback and the administrator who views the feedback

3.3.6 Use Case 6



Author	Deham Rajvanshi, Ayan Gupta
Purpose	To give the user an option of a virtual wallet from which charges will be deducted automatically and allow them to recharge it when needed
Requirements Traceability	The user must have an account in the app and must have a bank account/UPI to recharge the wallet
Priority	The priority of this use case is High, as maintaining a digital wallet is necessary for the payment of fares
Pre Conditions	The user must have created a digital wallet on the app for the ride fare to be deducted from it upon ride completion
Post Conditions	The wallet will show the remaining balance or the amount required to pay when a ride is completed, when the wallet is recharged, or whenever the user demands
Actors	The actors involved in this use case are users for whom the wallet is maintained and the admin

4 Other Non-functional Requirements

4.1 Performance Requirements

Dynamic Load Handling: The system must automatically adjust to fluctuations in user demand, allowing seamless booking and tracking even during peak hours (e.g. campus events or emergencies) without service degradation.

Cart Availability Check: All the cart availability queries must be processed within 2 seconds, ensuring users receive real-time updates without delay.

Booking Transaction Speed: The time a user takes to complete a booking (from initiating the confirmation request) should not exceed 2 minutes under standard server loads.

Route Update Synchronization: All the updates relating to the real-time availability of the golf carts and the real-time update to the number of seats available in a golf cart must be made within 5 seconds after any booking/cancellation is made through WebApp.

System Capacity: The backend system must support the simultaneous tracking of all 12 golf carts with location updates every 5 seconds, maintaining accurate real-time positioning for users.

Peak Traffic Response: The system must maintain a maximum response time of 3 seconds for non-essential functions like trip history or user profile changes during periods of high demand (e.g., more than 250 concurrent users).

System Downtime: Unscheduled downtime must not exceed 2 hours per year. Scheduled maintenance should be limited to biannual updates during early morning hours (1 a.m. – 4 a.m.).

4.2 Safety and Security Requirements

Geo-Fencing for Golf Carts: The software shall enforce geo-fencing to ensure carts operate within designated campus boundaries. If a cart breaches these boundaries, the software shall automatically notify administrators and issue a warning to the driver, accompanied by a three-minute timer, to facilitate the cart's return within the campus limits.

User Location Privacy: The software shall restrict access to real-time user location data exclusively during active rides. Upon the conclusion of each ride, the software must promptly delete the corresponding location data from the server to uphold user privacy.

Emergency Alerts with Location Sharing: The software shall incorporate an in-application emergency button that, upon activation, promptly transmits the user's current location, cart identification details, and the precise time of the alert to campus security. This feature is essential to ensure rapid response and enhance user safety during emergencies.

User Authentication: The software shall implement stringent user authentication protocols to verify the user's identity before granting access to the Golf Cart booking services. This includes enforcing robust password policies and incorporating multi-factor authentication mechanisms to prevent security hacks.

Emergency Assistance for Golf Cart Drivers: In emergencies or incidents such as a flat tyre or medical situations, the software shall provide a dedicated help button within the application. Activating this feature will notify the relevant authorities to dispatch assistance to the driver's location.

Automatic Time-Out: The software shall be configured to automatically terminate inactive sessions after 30 minutes to enhance overall security and prevent unauthorised access.

4.3 Software Quality Attributes

4.3.1 Reliability

The Golf Cart Management software must consistently perform its functions under specified conditions. Measures to ensure reliability include:

1. **Redundant Systems**: Deploy a failover mechanism with multiple backup servers to handle system failures seamlessly without disrupting user operations.
2. **Database Consistency**: Use ACID-compliant databases to maintain data consistency during transactions, ensuring no partial updates occur during booking or ride tracking.
3. **Monitoring and Alerts**: Implement real-time system monitoring with alerts for critical failures or anomalies, allowing administrators to respond immediately.
4. **Backup and Recovery**: Schedule daily automated database backups and ensure the recovery time objective (RTO) does not exceed 5 minutes in case of failure.
5. **Stress and Load Testing**: Perform regular testing under simulated peak loads to validate system performance under extreme conditions, such as during campus-wide events.

4.3.2 Adaptability

The system must be designed to accommodate future expansions and changing requirements. Key features to support adaptability include:

1. **Scalable Architecture**: Use a cloud-based infrastructure that allows for easy scaling of server resources as user demand increases.
2. **Modular Components**: Design core functionalities (e.g., booking, tracking, payment) as independent modules, enabling future enhancements or replacements without affecting the entire system.
3. **Customisable Features**: Administrators can modify routes, schedules, and notifications through a configurable backend interface.
4. **Integration-Ready APIs**: Develop APIs to support integration with third-party services, such as advanced navigation systems or payment gateways.
5. **Localisation Support**: Use internationalisation (i18n) frameworks to add multi-language support (e.g., English, Hindi), enabling easy adaptation for diverse user groups.

4.3.3 Portability

Portability ensures the system can function across different environments and platforms with minimal effort. Measures to achieve portability include:

1. **Device Independence:** Optimize the WebApp for various devices, including smartphones, tablets, and desktop systems, without compromising user experience or performance.
2. **Browser Compatibility***:* Ensure the system supports popular web browsers, including Chrome, Safari, Firefox, and Edge, across their latest and most widely-used versions.
3. **Easy Deployment**: A Docker-based containerised deployment solution shall be implemented to ensure seamless deployment of the MERN stack application across platforms and devices, eliminating compatibility issues by encapsulating all dependencies and runtime environments within the container for consistent performance on WebApp.

5 Other Requirements

5.1 Legal Requirements and Copyright

- User data must be stored in regional data centres to comply with national and international privacy regulations.
- The system must obtain explicit consent from users regarding data collection and usage.
- The development team must maintain copyright for all source code, documentation, and design, ensuring no unauthorised use of the intellectual property.

5.2 Authentication

- Implement a secure authentication mechanism using OTP (One-Time Password) sent via email for account verification and login processes.
- The system should integrate password hashing and multi-factor authentication to enhance security and prevent unauthorised access.

5.3 Database Requirements

- The database should be built with strong security to safeguard sensitive data, including user profiles, cart reservations, and payment information.
- It should have a recovery mechanism and automatic backups to resist faults or failures.
- For the database to allow numerous users to access the system simultaneously, concurrent read and write operations must be handled effectively.

Appendix A – Data Dictionary

A.1 User Database

Variables	Description	Operation	Requirements
user_id	Unique identifier for each user	Used for linking rides, payments, and bookings.	It must be unique and auto-generated by the system.
name	Full name of the user.	Captured during user registration.	It cannot be null.
email	The user's email address is used for login and notifications.	Used for authentication and notifications.	Must be unique and validated during registration.
phone_number	User's phone number for authentication	Used for ride updates.	Must be unique and validated during registration.
password	Encrypted password for account access.	Used for user authentication.	It cannot be null; it must meet security standards.

A.2 Driver Database

Variables	Description	Operation	Requirements
driver_id	Unique identifier for each driver.	Used to track rides and performance.	It must be unique and auto-generated by the system.
name	Full name of the driver.	Captured during driver registration.	It cannot be null.
license_no	Driver's license number for verification.	Used for validation during registration.	It cannot be null; it must match the driver's documents.
phone_number	Driver’s contact number.	Used for communication.	Cannot be null; validated.

active_hours	Number of hours the driver is actively logged in.	Tracks operational hours.	Auto-updated by the system.
total_rides	Total rides completed by the driver.	Tracks driver performance.	Auto-updated after each ride completion.
rating	Average rating received from passengers.	Dynamically updated post-feedback.	The default is 0; updated after each feedback.

A.3 Ride Database

Variables	Description	Operation	Requirements
ride_id	Unique identifier for each ride.	Links rides to drivers, users, and carts.	It must be unique and auto-generated by the system.
start_point	Ride start location (coordinates/address).	Captured during ride booking.	It cannot be null.
end_point	Ride end location (coordinates/address).	Captured during ride booking.	It cannot be null.
user_id	Reference to the user who booked the ride.	Links ride to the user profile.	Must exist in the User Database.
driver_id	Reference to the driver assigned to the ride.	Links ride to driver profile.	It must exist in the Driver Database.
golf_cart_id	Reference to the golf cart used.	Links ride to the golf cart.	It must exist in the Golf Cart Database.
start_time	Timestamp when the ride started.	Captured at ride initiation.	It cannot be null.
end_time	Timestamp when the ride ended.	Captured at ride completion.	It cannot be null.
seats_booked	Number of seats booked on the cart for the ride.	Validates against cart_capacity.	Cannot exceed cart_capacity.

amount	Cost of the ride.	Computed based on base_fare and fare_per_km.	It cannot be null.

A.4 Golf Cart Database

Variables	Description	Operation	Requirements
golf_cart_id	Unique identifier for each golf cart.	Links carts to rides and maintenance.	It must be unique and auto-generated by the system.
location	Current location of the golf cart.	Updated dynamically via GPS.	It cannot be null.
in_use	Indicates if the golf cart is currently in use.	Updated dynamically based on driver status and rides.	The default is false.
maintenance_due	Date when the next maintenance is scheduled.	Checked before assigning a ride.	It cannot be null.
age	Age of the golf cart in years.	Updated annually.	It cannot be null.

A.5 Payments Database

Variables	Description	Operation	Requirements
payment_id	Unique identifier for each payment.	Links payments to rides.	It must be unique and auto-generated by the system.
ride_id	Reference to the associated ride.	Links payment to ride.	Must exist in the Ride Database.
amount	Payment amount.	Captured during the transaction.	It cannot be null.
time	Timestamp of payment.	Captured during the transaction.	It cannot be null.

payment_ method	Mode of payment (e.g., credit card, UPI).	Captured during the transaction.	It cannot be null.
status	Payment status.	Updated post-transaction.	It cannot be null.

Appendix B - Group Log

S.No	Date	Timings	Venue	Description
1	07/01/2025	20:00 - 21:00	RM Building	Brainstormed various possible projects and valued them based on their merits. All the members agreed on implementing Golf Cart Management
2	14/01/2025	21:00 - 22:00	Google Meet	All the members discussed the functionalities to implement for the Golf Cart Management System and explored various routing algorithms
3	17/01/2025	16:00 - 17:00	Hall 12	Studied the SRS template, and then work was distributed among the team members based on mutual decision
4	20/01/2025	21:00 - 22:00	DOAA Canteen	First meet with our Teaching Assistant, Mr. Vikrant and discussed the project and the SRS documentation.
5	24/01/2025	20:00 - 21:00	Google Meet	Went through the SRS Document for proofreading and implemented changes based on the feedback given by the members of the team