RSET

RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Software Requirement Specification

On

RideShare

By

Dhiraj Bobby (U2103076)

Jerin Vincent (U2103112)

John Kurian (U2103116)

Kalidas Jayakumar (U2103122)

Under the guidance of

Dr. Saritha S

Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039
February 2024

Software Requirements Specification

for

RideShare

Version 1.0 approved

Prepared by

U2103076 Dhiraj Bobby

U2103112 Jerin Vicent

U2103116 John Kurian

U2103122 Kalidas Jayakumar

Guided By: *Dr. Saritha S, Professor, DCS RSET*

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) defines the functional and non-functional requirements for the RideShare carpooling mobile application, Version 1.0. It outlines the core functionalities, user interactions, system integrations, and quality attributes that will guide the development process. This SRS focuses solely on the initial functionalities of the RideShare mobile application for Android devices.

1.2 Product Scope

RideShare is a user-friendly mobile application designed for Android devices. Its primary objective is to facilitate carpooling by connecting riders seeking cost-effective transportation with drivers who have available seats. The application aims to address traffic congestion, reduce transportation costs for users, and promote environmental sustainability by encouraging carpooling as a viable alternative. This version will focus on core functionalities of the carpooling service. Future versions may incorporate additional features based on user feedback.

2. Overall Description

2.1 Product Perspective

RideShare is a carpooling app that provides a platform that connects drivers and passengers for ride-sharing, aiming to reduce the number of cars on the road, lessen commuting costs for passengers and drivers, and offer a convenient and potentially social alternative to single-occupancy vehicles. This app is a standalone product and is self-contained. However, it will integrate with existing mapping services like Google Maps

for route navigation and geo-location functionalities.The app will also integrate with other services like payment gateways, and communication platforms to provide users with a comprehensive solution for planning, booking, and sharing rides.

2.2 Product Functions

- User registration and login: Users can create accounts and login securely.
- Account verification: Users can add essential information like profile picture, Identification documents, phone number to verify their account.
- Offer rides: users can offer rides by specifying starting location, destination, time and available seats.
- Search for rides: Users can search for rides based on destination, time and other criteria.
- Request to join rides: Users can request to join rides offered by other users if it is suitable for them.
- Accept or Reject requests: Users can accept or reject join requests
- Communication: In-app chat allows co-passengers to communicate directly to arrange pickup/drop-off locations and confirm details.
- Navigation and Tracking: The app integrates with mapping services for route navigation and potentially real-time ride tracking for safety and peace of mind.
- Payment Processing: Secure payment options allow passengers to pay drivers through UPIs for their share of the ride.
- Rating and Review System: Users can rate and review other users, promoting accountability and trust within the carpooling community.
- Carbon Footprint Tracking: A feature that calculates the estimated carbon footprint saved by carpooling compared to single-occupancy vehicles, promoting environmental awareness.
- Safety Features: Additional features like emergency contact information, in-app emergency buttons can enhance user safety.

- Recurring Rides: Facilitate setting up recurring carpools for regular commutes (e.g., daily or weekly to work), streamlining the process for frequent riders on the same route.
- Multi-Stop Support: Allow drivers to offer rides with multiple pick-up and drop-off points along their route, catering to passengers with slightly detoured commutes.

2.3 Operating Environment

The app will operate on mobile devices with Android operating system with Android version 7.0(nougat) or above.

2.4 Design and Implementation Constraints

- Regulatory Compliance: RideShare app must adhere to local transportation regulations and ride-sharing laws in all regions where it operates. This includes compliance with government regulations, and data protection laws such as DPDP.
- Compatibility with Third-party APIs: The app will rely on external APIs for services such as mapping and payment processing. Developers must ensure compatibility with these APIs and adhere to their usage terms and conditions.
- Privacy Considerations: The app must respect user privacy and obtain explicit consent for obtaining their location and processing of personal data.

2.5 Assumptions and Dependencies

- User Behavior: We assume users will provide accurate and up-to-date information in their profiles and ride postings.
- GPS Accuracy: We assume a base level of GPS accuracy for location services, acknowledging potential discrepancies that may require user adjustments during rides

- Third-Party Integrations: The app heavily depends on successful integration with Google Maps API for functionalities like navigation, location services, and estimated travel times.
- Payment Gateway: We depend on partnering with a secure payment gateway that supports the desired payment methods and adheres to data security regulations

3. External Interface Requirements

3.1 User Interfaces

The RideShare app will prioritize a user-friendly UI designed for ease of use on Android devices. Some of the key UI characteristics are:

- **Clarity and Simplicity:** Information will be presented hierarchically with clear labeling for optimal readability.
- **Consistent Design Patterns:** A consistent look and feel will be maintained throughout the app with standardized button styles, icons, and navigation elements. This will create a familiar and predictable user experience.
- **Adaptability:** The UI will be designed to adapt to different screen sizes effectively. This ensures optimal viewing experience across various Android device models.
- **Core User Interfaces:** This version will focus on essential functionalities with dedicated UI components for:
    - **Login/Signup:** Screens for user registration, login, and profile management.
    - **Ride Offer:** Interface for drivers to specify details like origin, destination, available seats, date, time, and price for their rides.
    - **Ride Search:** Functionality for riders to search for existing rides based on origin, destination, date, and time (with optional filtering by price or other criteria).
    - **Ride Details:** Detailed information screens for both drivers and riders showcasing ride specifics like origin, destination, time, driver/rider information

(including profile picture and ratings), and estimated travel time via map integration.

- **Communication:** A basic communication channel (e.g., push notifications) to enable basic communication between riders and drivers regarding ride coordination (confirmation, pick-up details, etc.).
- **In-App Navigation:** Integration with Google Maps API to provide turn-by-turn navigation for drivers and an estimated route overview for riders during a carpool journey.
- **Settings:** User settings for managing preferences (e.g., notification settings, payment methods if applicable), and potentially viewing past ride history.

- **Error Handling:** User-friendly error messages will be displayed in a clear and concise manner, guiding users towards resolving any issues encountered within the app.

3.2 Hardware Interfaces

The RideShare app will primarily interact with the following hardware components on user devices:

- **Touchscreen:** The primary mode of user interaction will be through touchscreen gestures (taps, swipes) for navigation, selection, and data input.
- **GPS (Optional):** The app uses device GPS functionalities to determine the user's current location, potentially enhancing search results and navigation accuracy.

3.3 Software Interfaces

RideShare will interact with several external software components to deliver its core functionalities. Here's an outline of these interactions:

- **Google Maps API (Version to be specified):** This API will provide location services (user location retrieval), map data, and turn-by-turn navigation functionalities within the app.
- **Payment API (To be chosen):** A secure payment gateway integration will be implemented to facilitate in-app transactions for ride payments.
- **Communication API (To be chosen):** This API will enable basic communication functionalities between riders and drivers, potentially through push notifications or a messaging system (depending on chosen API capabilities).
- **Firebase:** This backend-as-a-service platform will be used for user authentication, secure data storage (user information, ride details, potentially payment information if applicable), and real-time database functionalities for managing ride requests and confirmations.

3.4 Communications Interfaces

The RideShare app will rely on the following communication interfaces:

- **Internet Connectivity:** A stable internet connection will be required for all functionalities including user login, data exchange with backend services (Firebase), map loading, and potentially real-time communication features.
- **Secure Communication Protocols:** Secure protocols like HTTPS will be implemented to safeguard user data transmission between the app, user devices, and backend services. This ensures data privacy and security.
- **API Communication Protocols:** Each chosen external API (Google Maps, Payment Gateway, Communication) will have its defined communication protocols for data exchange. The app will adhere to these protocols to ensure proper integration and functionality.

4. System Features

4.1 User Registration and Login

4.1.1 Description and Priority

- Users can add transportation routes by specifying start and end locations, time, and number of empty seats.Only a registered user can create and join a carpool.
- This feature is of high priority.

4.1.2 Stimulus/Response Sequences

- User launches Ride-share app.
- User taps on Sign-Up button
- User has to enter valid personal information by filling up registration form
- Upon successful validation:
    - System creates a new user account.
    - System sends a confirmation email to the user.
    - System logs the user in and directs them to the app's main dashboard
- Upon unsuccessful validation
    - System displays error message for each invalid field
    - User can correct the invalid fields and resubmit the form

4.1.3 Functional Requirements

REQ-1	System shall display a signup form upon user selection
REQ-2	The signup form shall include fields for name, email address, and password.
REQ-3	System shall validate the email address format.
REQ-4	System shall enforce minimum password length and complexity requirements.
REQ-5	System shall check for existing user accounts with the entered email address.

REQ-6	Upon successful registration, the system shall create a new user account.
REQ-7	System shall send a confirmation email to the user's registered email address
REQ-8	System shall log the user in and redirect them to the main dashboard upon successful registration.
REQ-9	System shall display clear and specific error messages for invalid user input during registration.

4.2 Search Carpools

4.1.1 Description and Priority

- Search option is displayed on the homepage of the application that allows users to search for available carpools based on their current location or user's specified location.This feature is of high priority.

4.2.2 Stimulus/Response Sequences

- User selects the "Find Pool" option on the app.
- Users can search carpools by giving required origin and destination places.Users can only select destination location given in the dropdown menu in the search option.
- System searches for carpools matching the user's criteria.
- If matching carpools are found, the system displays a list of results to the user, showing details such as driver information, route, departure time, and available seats.
- If no matching pools are available,the system displays a message indicating that no carpools are available at the moment.

4.2.3 Functional Requirements

REQ-1	System shall provide a dedicated interface for searching carpools.
REQ-2	Users shall be able to enter origin and destination location.
REQ-3	System should provide search filters based on user preferences like time,cost,car type and driver rating.
REQ-4	System shall search the database for available carpools that match the user's specified criteria.
REQ-5	System shall display a list of matching carpools with relevant details including driver information,departure time,available seats and car details.
REQ-6	If no matching carpools are found, the app shall display a message indicating that no carpools meeting the specified criteria were found.

4.3 Create Carpool

4.3.1 Description and Priority

- Only a registered user having a car can create a carpool by specifying the ride details and available seats.This feature has high priority.

4.3.2 Stimulus/Response Sequences

- Users can create a carpool by clicking on the “share ride” button on the app.
- App displays a form for carpool creation.
- User has to enter necessary information including origin,destination,time,car type,available seats and vehicle number.
- User then submits the form for generating a carpool.
- System validates the provided information.
- Upon successful validation, the system displays the carpool information in the app.
- Upon unsuccessful validation, the system displays an error message.
- System shall provide a feature to save the carpool information for future use to avoid users to enter the same information again.

4.3.3 Functional Requirements

REQ-1	The app shall provide a form for users to input details such as start and end locations, departure time, and number of available seats.
REQ-2	The app shall validate the user's input to ensure that all required fields are filled out and that the route and departure time are in a valid format.
REQ-3	The app shall enforce constraints on the number of available seats, ensuring it is a positive integer.
REQ-4	Upon successful creation of a carpool, the app shall store the carpool details in the database and associate them with the user's account.
REQ-5	App should also have a feature to save carpool information for future use.
REQ-6	If there are any errors during carpool creation, such as an invalid route or departure time in the past, the app shall display appropriate error messages and ask the user to correct them.

4.4 Join Carpool

4.4.1 Description and Priority

- System allows registered users to request a ride from one of the many available options of carpools. If a user has already joined a carpool, the app shall allow them to leave that carpool. This is a high priority feature.

4.4.2 Stimulus/Response Sequences

- Once the user selects a carpool after searching for available carpools,they can select the “Ride Request”option.
- Upon requesting the ride, the app will send a ride request to the carpool driver and display request status.

- If the driver accepts the request,the system will confirm booking and notify both driver and user.
- If the request was not approved by the driver,the app shall notify the user about the unsuccessful request.

4.4.3 Functional Requirements

REQ-1	System shall allow users to view detailed information about carpools displayed in search results.
REQ-2	System shall provide a way for users to request to join a carpool.This shall be implemented by using a “Request Ride” button.
REQ-3	Upon user confirmation, the app shall notify the driver of the carpool about the ride request.User should have the option to see the status of their request.
REQ-4	The driver should be able to accept or reject the ride request.
REQ-5	If the driver accepts the ride request, the app shall confirm the booking and notify both the user and the driver.
REQ-6	If the driver rejects the ride request or does not respond within a specified time frame, the app shall notify the user that the request was unsuccessful.
REQ-7	The app shall handle ride requests in real-time, updating the status of requests and bookings accordingly.

4.5 Location Based Services

4.5.1 Description and Priority

- The app can use the user’s location (with user’s permission) to make the searching process easier. It can automatically fill in your starting point and show you the nearest carpools first. This is of high priority.

4.5.2 Stimulus/Response Sequences

- User opens the app and grants location access.
- System retrieves the user's current location.
- System shall leverage user location data (when available) to auto-fill the origin field in the search interface.
- System shall prioritize search results based on the user's location, displaying carpools with origins closest to the user first.
- System shall integrate with mapping services to display the planned carpool route on the app.
- System retrieves the driver's current location and updates it in real-time during the carpool.
- Passengers requesting a ride can view the real-time location of the driver on the map within the app.

4.5.3 Functional Requirements

REQ-1	The app shall request and obtain permission from the user to access their location.
REQ-2	The app shall retrieve the user's current location using GPS or other location services.
REQ-3	The app shall display nearby carpools, drivers, or passengers based on the user's location.
REQ-4	The app shall provide accurate and up-to-date location-based information to users.
REQ-5	The app shall integrate Google maps API to display the planned route and should update data dynamically.
REQ-6	The app shall retrieve the drivers current location and updates it in real-time during the carpool
REQ-7	Passengers requesting a ride can view the real-time location of the driver on the map within the app.

4.6 Rating System

4.6.1 Description and Priority

- Enable users to rate and provide feedback on their carpooling experience, facilitating accountability and improving service quality.This feature is of medium priority.

4.6.2 Stimulus/Response Sequences

- After completion of each ride the user will be asked to rate the driver.
- Users can provide an optional star rating and a text review about the ride.
- System should store the review for the driver.
- System should calculate and update the driver's average rating.

4.6.3 Functional Requirements

REQ-1	System asks the user and driver to rate each other after each ride.
REQ-2	System allows users to rate each other by providing a star rating interface.Users can skip this step.
REQ-3	System also allows users to give a text review about the ride.
REQ-4	System stores the ratings and then calculates and updates the user's average rating.
REQ-5	The app should ensure that the ratings are anonymous to maintain privacy and confidentiality.
REQ-6	User's rating and reviews should be displayed in their profiles to help others to make informed decisions when selecting rides.
REQ-7	System should analyze repeated bad reviews and take appropriate actions like temporarily suspending their account.

4.7 Carpooled Distance and Carbon Emission Reduced

4.7.1 Description and Priority

- The system shall display the total distance traveled through carpooling and the corresponding reduction in carbon emissions in the user's profile.This feature's priority is Low.

4.7.2 Stimulus/Response Sequences

- User navigates to their profile page.
- The system will calculate the total distance carpooled and carbon emission reduced by calculating from the user's previous rides.
- The system will display and update the metrics after each ride.

4.7.3 Functional Requirements

REQ-1	The system shall track and record the distance traveled by the user through carpooling.
REQ-2	The system will calculate the reduction in carbon emissions achieved through carpooling, considering factors such as vehicle type, occupancy, and distance traveled.
REQ-3	The system will display the total distance carpooled and the corresponding emissions reduction in the user's profile with clear and intuitive visualization.

4.8 User Safety

4.8.1 Description and Priority

- The system shall include an emergency contact SOS button that allows users to request immediate assistance in case of emergencies.Priority of this feature is high.

4.8.2 Stimulus/Response Sequences

- User activates the emergency contact SOS button within the app.
- System immediately sends an alert to designated emergency contacts, providing the user's location and relevant details.
- System notifies local emergency services or authorities if necessary.

4.8.3 Functional Requirements

REQ-1	The emergency contact SOS button shall be easily accessible from within the app interface, visible on all screens for quick access
REQ-2	Upon activation, the system shall send a real-time alert to designated emergency contacts, including the user's location and timestamp.
REQ-3	The system shall provide options for users to designate specific emergency contacts within their profile settings.
REQ-4	In case of activation, the system shall prompt the user to confirm the emergency before initiating alerts to prevent false alarms.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

A proper internet connection is needed for the users using this application and the user interface should be user-friendly and easy for Android mobile phone users. The database must function well in order for the user to have no delays in the program.

Reading is more significant than writing since users are more likely to spend time browsing than posting trips. As a result, few actions are written to the database. Hence a read operation should take less than 500ms and a write operation should approximately be 1s.

5.2 Safety Requirements

The app prioritizes user safety by mitigating risks of physical harm, data breaches, privacy violations, and fostering a false sense of security. Implemented safeguards include:

- User Verification: Mandatory ID verification for both drivers and passengers, background checks on drivers, and a reporting system for suspicious activity.

- Driver Screening: Requirements for proof of a valid driver's license and car insurance, enforcement of a minimum driver age, and maintenance of a driver rating and review system.

- Trip Monitoring and Tracking: Integration of real-time GPS tracking during rides, the option to share trip progress with designated contacts, and an in-app emergency button for direct connection to emergency services.

- Data Security: Encryption of user data stored on servers, adherence to relevant data privacy regulations, and regular security audits and penetration testing.

- User Education: Promotion of responsible carpooling behavior through in-app tutorials and safety tips, along with encouraging users to verify driver information and exercise caution during rides.

5.3 Security Requirements

The RideShare application will implement essential security measures to safeguard user data and ensure secure access to the platform. These measures include basic user authentication with password hashing to protect user credentials, encryption techniques for securing sensitive data during transmission and storage, and the implementation of role-based access control to differentiate between regular users and drivers, thereby controlling access to specific functionalities within the application. Additionally, the project will focus on minimal data collection to uphold user privacy. Adherence to secure development practices such as regular code reviews and following secure coding guidelines will be emphasized to mitigate common security vulnerabilities and ensure the integrity and reliability of the application.

5.4 Software Quality Attributes

- Maintainability

  The system should be maintainable in the future. The system should be well documented such that new developers can quickly comprehend the system.We should be certain that the technology we choose will be supported for a long time.

- Scalability

  Currently the carpool application is aimed only for students of Rajagiri but it could be expanded internationally. As the number of users and operations grows, the server and database must be able to scale both vertically and horizontally. The

system must be able to accommodate a user base from 100 users to 1 million users.

- Usability

  The app should provide users with a user-friendly user interface for easy use of the app. Text , buttons and other graphic elements should be large enough to improve clarity and enhance user experience.Navigation through the app should be simple so that users can easily navigate through the app with ease.

- Portability

  The initial version of the app shall be developed for the Android platform, ensuring compatibility with Android smartphones and tablets. The app architecture and codebase shall be designed with modularity and flexibility to facilitate future development for the iPhone platform. Targeting both Android and iOS allows the app to reach a broader user base, maximizing its potential reach and impact

References

1. Official Flutter Documentation - https://docs.flutter.dev/
2. Official Dart Documentation - https://dart.dev/guides
3. Flutter API Reference - https://api.flutter.dev/
4. Package Documentation - https://pub.dev/
5. Flutter Community - https://flutter.dev/community