Software Requirements Specification

for

Profit Path

Version 1.0 approved

Prepared by Smart-XPlorers

Table of Contents

Table of Contents ..... **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience 2
1.4 Product Scope 2
2. Overall Description **3**
2.1 Product Perspective 3
2.2 Product Functions 3
2.3 User Classes and Characteristics 3
2.4 Operating Environment 4
2.5 Design and Implementation Constraints 5
2.6 User Documentation 6
2.7 Assumptions and Dependencies 6
3. External Interface Requirements **8**
3.1 User Interfaces 8
3.2 Communications Interfaces 9
4. System Features **10**
4.1 Users 10
4.2 Ride Services: 13
4.3 Delivery Services 15
4.4 Driver Services: 17
5. Other Nonfunctional Requirements **19**
5.1 Security Requirements 19
5.2 Reliability Requirements 19
5.3 Portability Requirements 19
5.4 Performance Requirements 20
6. Data Dictionary **21**

1. Introduction

1.1 Purpose

The requirements given here and onwards refer to the product ProfitPath conceptualized and ideated by the team Smart-XPlorers. This document lays out the description, deliverables, features, and all other required documentation required throughout the Software Development Life Cycle of the product. This document is meant to provide detailed descriptions of the product to all stakeholders involved directly or indirectly with the product. Note that this document **DOES NOT** contain any ideation or any subsequent details such as the Use Case model, the class diagrams, and laid out plans for testing. This document is simply meant to elicit the requirements and the expectations from the project. A final list of deliverables and a comprehensive plan on the development process is included in this document.

1.2 Document Conventions

This document follows the Atomic Function Requirements Convention

In order to streamline the traceability and verification process for developers, the atomic function requirements in this document follow a specific convention. Each requirement is associated with a priority number that indicates its immediate implementation status within the outer point it belongs to. Each function requirement is broken down into atomic units to facilitate tracking and verification.

1.2.1 Guidelines for Reading Functional Requirements for developers:

1. Clarity: Ensure each requirement is clear, concise, and directly related to the software functionality.
2. Atomicity: Break down complex requirements into smaller atomic units for better manageability.
3. Traceability: Link each requirement to its corresponding outer point for seamless tracking.
4. Verification: Regularly verify that all requirements have been fulfilled as per their priority numbers.

By adhering to these conventions and guidelines, developers can effectively trace, verify, and implement the function requirements with precision and efficiency.

1.3 Intended Audience

The intended audience for this document includes the developers actively engaged in the software development process, the team leader overseeing the project, judges for national competitions, potential investors, and other stakeholders interested in the application. Additionally, this document serves as an introduction to the public, offering insight into the innovative product being developed. Public engagement with the application may lead to increased interest and support, further enhancing its potential impact and success.

1.4 Product Scope

ProfitPath is a groundbreaking community-focused application designed to transform transportation and delivery services in Singapore. Inspired by the successful "Carousel" model that revolutionized retail and second-hand markets in the country, ProfitPath offers a unique platform where users can seamlessly switch between being passengers and drivers, fostering deeper engagement within the community.

At the core of ProfitPath is the innovative concept of enabling users to earn income by providing transport and delivery services during their daily commutes. This concept not only aligns with the Smart Nation initiative of the Singapore Government, promoting a people-centric digital ecosystem to enhance quality of life, but also supports the Green Nation objective by reducing carbon emissions and contributing to Singapore's environmental sustainability.

By facilitating a dual-role system for users and promoting a shared economy approach to transportation, ProfitPath serves as a driving force for socio-economic progress. It encourages creative solutions to everyday commuting challenges while empowering individuals to make a positive impact on the community and the environment.

Moreover, ProfitPath offers a convenient solution for individual transport and delivery requirements. With flexible options for immediate or scheduled services, users can access affordable and efficient transportation or have their goods delivered in a cost-effective manner.

In summary, ProfitPath not only addresses the evolving needs of modern transportation and delivery services but also embodies a vision of community collaboration, environmental responsibility, and economic empowerment. Its innovative approach aligns with corporate objectives and business strategies, positioning it as a transformative force in Singapore's digital landscape.

2. Overall Description

2.1 Product Perspective

ProfitPath is a new product which is not a member of or associated with any other product on the market today. All the software requirements and functionality described in this document is meant to build a final Version 1.0 of the software which is completely self-contained and does not depend on any other dependencies apart from the ones listed in this document.

2.2 Product Functions

- Enable users to switch between passenger and driver roles seamlessly.
- Allow users to earn money by providing transport and delivery services during daily commutes.
- Offer immediate or scheduled transport and delivery options for users.
- Support a shared economy model for transportation services.
- Promote community engagement and environmental sustainability through innovative transportation solutions.

2.3 User Classes and Characteristics

1. Delivery Users:

- Characteristics: Regular users who rely on delivery services for transporting goods.
- Usage: Require delivery services for personal or business needs.

2. Ride Users:

- Characteristics: Individuals seeking transportation services for commuting or travel.
- Usage: Use the platform to book rides for various purposes.

3. Driver Users:

- Characteristics: Users interested in earning money by providing transport and delivery services.
- Usage: Perform delivery tasks during their daily commutes to earn income.

4. Administrator:

- Characteristics: Responsible for managing and overseeing the platform.

- Usage: Access to database stores and administrative functions to ensure smooth operation of ProfitPath.

The primary user classes for ProfitPath are Delivery Users, Ride Users, and Driver Users, as they are the key participants who engage with the core functions of the application. The Administrator user class, while important for system management, is considered less critical in terms of day-to-day interaction with the platform.

2.4 Operating Environment

2.4.1 Frontend Requirements:

1. Web Application Requirements:
    a. ProfitPath is designed as a web application that is compatible with modern web browsers. It requires the following browsers and their latest versions:
        - Google Chrome
        - Mozilla Firefox
        - Safari
        - Microsoft Edge
    b. JavaScript must be enabled in the browser for the application to function properly.
    c. Continuous internet connectivity is essential for real-time usage and updates.
2. Device Requirements:
    a. The software relies on live user location, necessitating access to GPS functionality on the device.
    b. Users must be able to share their GPS location with the software for accurate service provision.

2.4.2 Server Requirements:

1. Operating System: Linux OS required, specifically Amazon Linux or Ubuntu, to host the server environment for ProfitPath.

2. Software Dependencies: Python and other necessary dependencies must be installed on the server to support the execution of ProfitPath.

3. Load Balancer: Implementation of a load balancer is recommended if multiple servers will be used to ensure efficient distribution of incoming traffic.

4. Public IP Address: Each server should be assigned a public IP address to enable external communication and access to the ProfitPath application.

2.4.3 Database Requirements:

1. Database Management System: Postgres is utilized as the primary database for storing application data.

2. Data Backup: Regular automated backups of the Postgres database should be scheduled to prevent data loss in case of system failures.

3. Data Encryption: Implement data encryption protocols to secure sensitive user information stored in the Postgres database.

4. Scalability: The database should be designed for scalability to accommodate increasing data volumes as the application usage grows.

5. Performance Monitoring: Utilize tools for monitoring database performance to ensure optimal operation and timely troubleshooting of any issues.

2.5 Design and Implementation Constraints

1. Privacy Regulations: The collection of live user locations necessitates stringent measures to ensure data privacy and security. User location data must be encrypted and stored securely to prevent unauthorized access.

2. Hardware Limitations: Given that web browsers may not receive the same priority as mobile applications, optimizing the application for lightweight performance and fast loading times is crucial. Efforts should be made to enhance user experience on various devices with varying hardware capabilities.

3. Interface Compatibility: Integration with a Maps API is essential to provide users with map views and directions within the application. Seamless interface compatibility with the Maps API is critical for delivering location-based services effectively.

4. Scalability Requirements: Anticipating a user growth rate of 50% in the first month and a subsequent 10% increase per month for at least a year, scalability is paramount. The system architecture must be designed to scale seamlessly to accommodate the expanding user base and increasing data demands.

5. Payment Platform Security: Integration of a secure payment platform demands a focus on ensuring the integrity of transactions, safeguarding user payment information, and maintaining the security of wallet balances. Developers must prioritize security measures to protect financial data and uphold payment integrity within the application.

2.6 User Documentation

Along with the software there will be a user manual and an introductory video. The User manual will contain screenshots of the product along with detailed explanations of how each feature works and how to use it.

The video will contain a demo of the entire web application demonstrating all features along with audio explanations for easier access and more information.

2.7 Assumptions and Dependencies

1. Frontend:
    - Assumptions:
        - Frontend development will utilize React for building the user interface and TailwindCSS for styling.
        - Integration of a third-party Maps API, preferably Google Maps, for location-based services.
        - Developers may rely on additional NPM-based dependencies for frontend functionality.
    - Dependencies:
        - Successful integration with the selected Maps API is crucial for providing map views and directions within the application.
2. **Backend:**
    - Assumptions:
        - Backend development will be implemented using Django framework for server-side operations.
        - Active internet connection is required for seamless communication between the backend server and external services.
        - Utilization of Google Maps API for location-related backend functionalities.
        - Developers may need to install other pip-based dependencies as necessary for backend functionalities.
    - Dependencies:

- Reliable internet connectivity is essential for the backend to interact with external services and APIs effectively.

3. **Database:**
    - Assumptions:
        - Postgres database management system (DBMS) will be used for storing and managing application data efficiently.
    - Dependencies:
        - The successful setup and maintenance of the Postgres database are critical for data storage and retrieval within the application.

3. External Interface Requirements

3.1 User Interfaces

1. **Homepage**
    - **Login and Sign-Up Pages:** Users can authenticate themselves through a login page requiring username and password. The sign-up page collects basic user details and card information.
    - **Navigation Options:** The homepage provides access to ride and delivery services, allowing users to choose their desired service.
    - **Current Requests Link:** A link on the homepage directs users to view ongoing ride or delivery requests.
2. **Profile Page:**
    - **User Details Display:** Users can view and edit their personal information such as username, email, address, and car details.
    - **Card Details Editing:** Ability to update card information for payment purposes.
    - **Details of Car owned:** Ability to update the details of the car that the user owns and plans to use for rides and deliveries.
    - **Service and Wallet History:** Users can access their service history, wallet transactions, and view past activities.
    - **Ratings:** The profile page should also display an aggregate rating received by the user as both a requester and a driver.
3. **Ride and Delivery Page:**
    - **Map Integration:** Features a map interface for users to input resource and destination locations, with real-time direction display.
    - **Pricing and Instructions:** Users can enter pricing details, additional instructions, and initiate ride or delivery requests.
4. **Current Request Page:**
    - **Live Location Display:** Shows the live location of the driver who accepted the request, along with request details.

- **Cancellation Option:** Users have the ability to cancel ongoing ride or delivery requests.
- **Chat Interface:** Enables communication between users and drivers through a chat window for updates and inquiries.

5. **Driver Pages:**

- **Request Management:** Drivers can view a list of current ride or delivery requests and select one to accept.
- **Pick-Up Confirmation:** Upon accepting a request, drivers receive details of the user and pick-up location, with an option to confirm pick-up.
- **Navigation Assistance:** Provides directions, live location tracking, and instructions for reaching the pick-up and destination locations.
- **Feedback Submission:** After completing the ride, drivers can provide feedback on the passenger's experience for continuous improvement.

3.1.1 Additional Details:

1. The Webapp UI should be optimised for mobile phones as the primary medium of delivery.
2. The average viewport size will be 400x850 pixels.
3. The UI should be responsive. Relative sizing and margins should be utilised wherever possible.

3.2 Communications Interfaces

1. Communication between Frontend and Backend: The communication between frontend and backend should utilize the HTTPS protocol. Secure communication should be enforced, and HTTP requests should be rejected if not encrypted.
2. Communication between Backend and Database: The communication between the backend and the database should be using the TCP stack protocol.

4. System Features

4.1 Users

4.1.1 Actors

Passenger, delivery requester, responder (synonymous with driver)

4.1.2 Functional Requirements

1 A user must have an account to use the services that the application offers.
1.1 The user shall be able to log-in to their account if they have created one beforehand.
1.1.1 The user shall be redirected to another page for the log-in process.
1.1.1.1 The user must fill in all fields required by the application in order to log-in.
1.1.1.1.1 The application must prompt the user for the username to his/her account.
1.1.1.1.1.1 The application shall deny the user entry into their account if the user provides an invalid username that cannot be found.
1.1.1.1.2 The application must prompt the user for the password to his/her account.
1.1.1.1.2.1 The application shall deny the user entry into their account if the user provides a non-matching password to his/her account.
1.2 The user shall be able to register to create a new account.
1.2.1 The application shall redirect the user to another page for the registration process.
1.2.1.1 The registration page must contain all fields that the user is required to fill in before his/her account may be created.
1.2.1.1.1 The user shall be required to provide his/her email address.
1.2.1.1.2 The user shall be required to provide his/her phone number.
1.2.1.1.3 The user shall be required to create a password for his/her account.
1.2.1.1.3.1 The user must be required to confirm his/her created password by re-entering the same password.
1.2.1.1.4 The user shall be required to create a username for his/her account.
1.2.1.1.5 The user shall be required to provide his/her name.
1.2.1.1.6 The user shall be required to provide his/her address.
1.2.1.1.7 The user shall be given the option to fill in his/her car details.
1.2.1.1.7.1 The user must provide his/her car's license plate number.
1.2.1.1.7.2 The user must provide his/her car's model.
1.2.1.1.7.3 The user must provide his/her car's colour.
1.3 The user shall be directed to the main user interface tied to his/her account after a successful registration or log-in.
1.3.1 The application shall display the account's username.
1.3.2 The application shall display a user icon.
1.3.2.1 The user shall be redirected to a page holding all user account information when the user taps on the user icon.
1.3.2.1.1 The application must display the user's current personal details.

1.3.2.1.1.1 The application shall display the user's name.
1.3.2.1.1.2 The application shall display the user's phone number.
1.3.2.1.1.3 The application shall display the user's email address.
1.3.2.1.1.4 The application shall display the user's address.
1.3.2.1.2 The user must be allowed to update his/her personal details on their account.
1.3.2.1.2.1 The user must be allowed to edit his/her name.
1.3.2.1.2.2 The user must be allowed to edit his/her phone number.
1.3.2.1.2.3 The user must be allowed to edit his/her address.
1.3.2.1.3 The user shall be redirected to the car detail page when accessing their car details.
1.3.2.1.3.1 The application must display the user's car details on the car detail page.
1.3.2.1.3.1.1 The application shall display the car license plate number.
1.3.2.1.3.1.2 The application shall display the car model.
1.3.2.1.3.1.3 The application shall display the car colour.
1.3.2.1.3.2 The user must be allowed to update his/her car details.
1.3.2.1.3.2.1 The user must be able to edit his/her car license plate number
1.3.2.1.3.2.2 The user must be able to edit his/her car model.
1.3.2.1.3.2.3 The user must be able to edit his/her car colour.
1.3.2.1.4 The application must display the current balance that the user has in his/her account.
1.3.2.1.5 The application shall be redirected to the wallet page when accessing their balance.
1.3.2.1.5.1 The user shall be given the option to top-up money.
1.3.2.1.5.1.1 The user must specify the amount he/she wants to top-up.
1.3.2.1.5.2 The user shall be given the option to withdraw money from his/her wallet.
1.3.2.1.5.1.2 The user must specify the amount he/she wants to withdraw.
1.3.2.1.5.3 The application shall display the user's transaction history.
1.3.2.1.5.3.1 For each transaction, the application must display the type of transaction that the user had performed.
1.3.2.1.5.3.1.1 The type of transaction must be one of the four - addition, deduction, withdrawal, or top-up.
1.3.2.1.5.3.2 For each transaction, the application must display the date when the transaction was enacted.
1.3.2.1.5.3.3 The transaction display must contain the amount involved in the transaction.
1.3.2.1.6 The application must display an aggregation of the user's obtained ratings.
1.3.2.1.7 The application must display an aggregation of the user's obtained reviews.
1.3.2.1.8 The application shall display the user's history of services requested and responded.
1.3.2.1.8.1 For each service history entry, the application must display the type of service.
1.3.2.1.8.1.1 The type of service must be either delivery or ride.
1.3.2.1.8.2 For each service history entry, the application must display the fare paid from the requester to the responder, or received from the requester depending on the role the user played.
1.3.2.1.8.3 For each service history entry, the application must display the date where the service was requested or responded to.
1.3.2.1.8.4 For each service history entry, the application must display in a string: '<pick-up location> to <drop-off location>'.
1.3.2.1.8.5 A user must be able to select any service history entry, in which the application shall then provide the user with additional details on that service history entry.

1.3.2.1.8.5.1 The application must further display the responder's username.
1.3.2.1.8.5.2 The application must further display the requester's username.
1.3.2.1.8.5.3 The application must further display the rating the responder has received for responding to this particular service.
1.3.2.1.8.5.4 The application must further display the rating the requester has received for requesting this particular service.
1.3.2.1.8.5.5 The application must further display the time where the responder had accepted the requester's request.
1.3.2.1.8.5.6 The application must further display the pick-up time where the responder has picked-up the requester for a ride, or the requester's package for a delivery.
1.3.2.1.8.5.7 The application must further display the completed time where the responder has dropped-off the requester at the end of the ride, or the requester's package at the end of the delivery.
1.3.2.1.8.5.8 The application must further display the estimated distance between the pick-up and drop-off locations.
1.3.2.1.8.5.9 The application must further display the estimated duration between the pick-up and drop-off locations.

1.3.3 The application shall display the services offered to the user.

4.2 Ride Services:

4.2.1 Actors

Passenger

4.2.2 Functional requirements

2 The application shall offer a ride service where he/she shall be able to request a ride.
2.1 The user shall be redirected to the ride page for making a request.
2.1.1 The user must enter a pick-up location where he/she will be picked-up from.
2.1.1.1 A passenger must only be allowed to choose one location as the pick-up point per request.
2.1.1.2 A passenger shall be allowed to change the pick-up point before confirming his/her ride request.
2.1.1.3 The application shall use the google API to cross reference the user's input of pick-up location with its list of valid locations and produce the closest matching locations.
2.1.1.3.1 The user shall be able to choose from the list of closest matching locations if there is no exact match with his/her inputted pick-up location.
2.1.2 The user must enter a drop-off location where he/she will be dropped off.
2.1.2.1 The user must only be allowed to choose one location as the drop-off point.
2.1.2.2 The user shall be allowed to change the drop-off point before confirmation of his/her ride request.
2.1.2.3 The application shall use the google API to cross reference the user's input of drop-off location with its list of valid locations and produce the closest matching locations.
2.1.2.3.1 The user shall be able to choose from the list of closest matching locations if there is no exact match with his/her inputted drop-off location.
2.1.3 The user must be able input the fare he/she is willing to pay for the service.
2.1.4 The user must be able to input a set of instructions that will be conveyed to the driver/responder.
2.1.5 The application shall need to provide a confirmation button for the user which on being pressed will prompt checking of valid inputs and then the creation of the ride request.
2.1.5.1 If the amount in the passenger's wallet is less than the inputted fare, then the application must not allow the passenger to make a confirmation.
2.1.5.1.1 The user must be notified that there are not enough funds in his/her wallet.
2.1.6 The application shall display a summary of the details provided for the ride request after the confirmation has been made by the user.
2.1.7 The application shall display the live status of the user's request.
2.1.7.1 The user shall be able to see updates to this live status as his/her request progresses onto different stages.
2.1.7.1.1 If a driver/responder has accepted the user's request, the live status must be updated to "Awaiting Pick-up".
2.1.7.1.1.1 The application shall further display a map showing the driver's current location which will be updated in real-time as the driver makes his/her way to the pick-up location.
2.1.7.1.1.2 The application shall further update the ride details by including the driver or responder's information and car details.

2.1.7.1.1.3 The user shall be able to use a chat feature to communicate with the driver/responder.
2.1.7.1.1.2.1 The responder's name must be displayed.
2.1.7.1.1.2.2 The responder's car model must be displayed.
2.1.7.1.1.2.3 The responder's car license plate number must be displayed.
2.1.7.1.1.2.4 The responder's car colour must be displayed.
2.1.7.1.2 If the driver/responder has picked-up the user at the pick-up location, the live status must be updated to "On Route",
2.1.7.1.2.1 The application shall further display a map showing the route of the ride which will be updated in real-time as the driver and the user make their way to the drop-off location.
2.1.7.1.2.2 The user must still be able to use the chat feature to communicate with the driver/responder.
2.1.7.1.3 If the driver/responder has dropped-off the user at the drop-off location, the user shall be redirected to the ratings page where he/she must be able to give a rating to the driver.
2.1.8 The user shall be able to assign a rating to the driver after the user has arrived at the drop-off point.
2.1.8.1 The rating must be on a scale from 1 to 5; 1 being the lowest and 5 being the highest.
2.1.8.2 A passenger shall be able to choose to write a review about the driver after the user has arrived at the drop-off point.
2.1.8.2.1 The review must be at least 0 characters.
2.1.9 The fare shall be deducted from the user's wallet once the user has been picked up from the pick-up point.

4.3 Delivery Services

4.2.1 Actors

Delivery requester

4.2.2 Functional requirements

3 The application shall offer a delivery service where he/she shall be able to request a delivery.
3.1 The user shall be redirected to the delivery page for making a request.
3.1.1 The user must enter a pick-up location where he/she wants their package to be picked-up from.
3.1.1.1 A passenger must only be allowed to choose one location as the pick-up point per request.
3.1.1.2 A passenger shall be allowed to change the pick-up point before confirming his/her delivery request.
3.1.1.3 The application shall use the google API to cross reference the user's input of pick-up location with its list of valid locations and produce the closest matching locations.
3.1.1.3.1 The user shall be able to choose from the list of closest matching locations if there is no exact match with his/her inputted pick-up location.
3.1.2 The user must enter a drop-off location where he/she wants their package to be dropped off.
3.1.2.1 The user must only be allowed to choose one location as the drop-off point.
3.1.2.2 The user shall be allowed to change the drop-off point before confirmation of his/her delivery request.
3.1.2.3 The application shall use the google API to cross reference the user's input of drop-off location with its list of valid locations and produce the closest matching locations.
3.1.2.3.1 The user shall be able to choose from the list of closest matching locations if there is no exact match with his/her inputted drop-off location.
3.1.3 The user must be able input the fare he/she is willing to pay for the service.
3.1.4 The user must be able to input a set of instructions that will be conveyed to the driver/responder.
3.1.5 The application shall need to provide a confirmation button for the user which on being pressed will prompt checking of valid inputs and then the creation of the delivery request.
3.1.5.1 If the amount in the passenger's wallet is less than the inputted fare, then the application must not allow the passenger to make a confirmation.
3.1.5.1.1 The user must be notified that there are not enough funds in his/her wallet.
3.1.6 The application shall display a summary of the details provided for the delivery request after the confirmation has been made by the user.
3.1.7 The application shall display the live status of the user's request.
3.1.7.1 The user shall be able to see updates to this live status as his/her request progresses onto different stages.
3.1.7.1.1 If a driver/responder has accepted the user's request, the live status must be updated to "Awaiting Pick-up".
3.1.7.1.1.1 The application shall further display a map showing the driver's current location which will be updated in real-time as the driver makes his/her way to the pick-up location.

3.1.7.1.1.2 The application shall further update the delivery details by including the driver or responder's information and car details.
3.1.7.1.1.3 The user shall be able to use a chat feature to communicate with the driver/responder.
3.1.7.1.1.2.1 The responder's name must be displayed.
3.1.7.1.1.2.2 The responder's car model must be displayed.
3.1.7.1.1.2.3 The responder's car license plate number must be displayed.
3.1.7.1.1.2.4 The responder's car colour must be displayed.
3.1.7.1.2 If the driver/responder has picked-up the user's packages at the pick-up location, the live status must be updated to "On Route",
3.1.7.1.2.1 The application shall further display a map showing the route of the ride which will be updated in real-time as the driver and the user's packages make their way to the drop-off location.
3.1.7.1.2.2 The user must still be able to use the chat feature to communicate with the driver/responder.
3.1.7.1.3 If the driver/responder has dropped-off the user's packages at the drop-off location, the user shall be redirected to the ratings page where he/she must be able to give a rating to the driver.
3.1.8 The user shall be able to assign a rating to the driver after the user has arrived at the drop-off point.
3.1.8.1 The rating must be on a scale from 1 to 5; 1 being the lowest and 5 being the highest.
3.1.8.2 A passenger shall be able to choose to write a review about the driver after the user's packages have arrived at the drop-off point.
3.1.8.2.1 The review must be at least 1 character.
3.1.9 The fare shall be deducted from the user's wallet once the user's packages have been picked up from the pick-up point.

4.4 Driver Services:

4.2.1 Actors

Responder (Synonymous with driver)

4.2.2 Functional requirements

4 The application shall offer a method/service that allows users to earn money responding to service requests.
4.1 The user shall be redirected to the earn page to respond to a request as the driver.
4.1.1 The user must not be able to access this service if he/she does not own a car.
4.1.2 The driver must enter his/her current location.
4.1.3 The driver will see a list of ride/delivery requests based on his/her current location.
4.1.3.1 The list of requests are sorted based on the closeness of the pick-up points of each request from the driver's current location.
4.1.3.2 For each request, the application shall display relevant information about the request.
4.1.3.2.1 The application must display the type of service.
4.1.3.2.1.1 The type of service must either be of a ride or a delivery.
4.1.3.2.2 The application must display the pick-up location.
4.1.3.2.3 The application must display the drop-off location.
4.1.3.2.4 The application must display the fare offered by the requester.
4.1.3.2.5 The application must display the estimated distance from the driver's current location to the request's pick-up point.
4.1.3.2.6 The application must display the estimated duration for the driver to reach the request's pick-up location from his/her current location.
4.1.4 The driver shall be able to choose only one request out from the list to accept.
4.1.5 The driver shall be redirected to a new page after accepting the request.
4.1.5.1 The application shall display details about the request.
4.1.5.1.1 The application must display the pick-up location.
4.1.5.1.2 The application must display the drop-off location.
4.1.5.1.3 The application must display the type of service.
4.1.5.1.4 The application must display the fare offered by the requester.
4.1.5.1.5 The application must display additional instructions used by the requester.
4.1.5.2 The driver shall be able to use a chat function to communicate with the requester.
4.1.5.3 The application must provide the driver control features which will update as the request progresses onto different stages.
4.1.5.3.1 The driver must be able to press 'Confirm pick-up' only after picking up the requester or the requester's packages for delivery.
4.1.5.3.2 The driver must be able to press 'Complete ride' only after dropping off the requester or the requester's packages for delivery.
4.1.5.4 The driver shall be redirected to the ratings page after the driver has dropped off the requester or the requester's packages at the drop-off location.

4.1.5.4.1 The driver shall be able to assign a rating to the passenger after he/she has alighted at the drop-off point.
4.1.5.4.1.1 The rating must be on a scale from 1 to 5; 1 being the lowest and 5 being the highest.
4.1.5.4.2 A driver shall be able to choose to write a review about the passenger after he/she has alighted at the drop-off point.
4.1.5.4.2.1 The review must be at least 1 character.
4.1.5.5 The fare will be awarded to the driver's wallet after the passenger has alighted at the drop-off point.

5. Other Nonfunctional Requirements

5.1 Security Requirements

The application must adhere to specific security requirements to safeguard user data and ensure the prevention of unauthorized access. All user data must be securely stored, with measures in place to prevent any unauthorized access. This includes hashing the user's password in the frontend and encrypting it in the backend to enhance password security. Additionally, the application should utilize security measures like JWT tokens or session tokens to protect user sessions from compromise and unauthorized access by third parties. By implementing these security protocols, the application can maintain a high level of security and protect user information effectively.

5.2 Reliability Requirements

The application is required to meet various reliability and usability standards to ensure a seamless user experience. In terms of reliability, the application must maintain a high level of uptime, with a target of 99.99% availability to ensure users can access the platform consistently.

Regarding usability, several requirements must be met to enhance the user experience. These include ensuring that 95% of users can initiate a carpool or delivery request within a minute of accessing the application. Additionally, 95% of users should feel comfortable navigating the user interface within 5 minutes of use.

Furthermore, users should have the option to change the language settings within the application and be able to reset their passwords within a 5-minute timeframe. The application must also incorporate proper exception handling to verify each user action, preventing crashes caused by erroneous user inputs. By meeting these usability requirements, the application can offer a user-friendly interface and a reliable service to its users.

5.3 Portability Requirements

To ensure portability, the application must meet specific requirements that allow for flexibility and ease of integration with future enhancements. Firstly, the application should be designed in a flexible manner that enables the seamless integration of extended capabilities without the need to replace the entire codebase. This approach will facilitate the addition of new features and functionalities without significant redevelopment efforts.

Additionally, in terms of database portability, the application must support the interchangeability of databases. Specifically, the database should be replaceable with any commercial product that supports standard SQL queries. This requirement ensures that the application can adapt to different database systems based on evolving needs or preferences, enhancing its overall portability and scalability. By meeting these portability requirements, the application can effectively accommodate changes and upgrades while maintaining its core functionality and performance.

5.4 Performance Requirements

To ensure optimal performance, the application must be designed to scale effectively and maintain high performance levels under varying user loads. Key performance requirements include having a minimum load capacity of 1.4 times the expected or average load capacity at any given time to handle peak usage without compromising performance. In cases where multiple backend servers are employed, a load balancer must be utilized to evenly distribute the workload across all servers, preventing crashes and maintaining system stability.

Additionally, all computationally intensive tasks, such as passenger matching for carpooling requests, should be offloaded to backend servers to reduce the load on the local CPU. This offloading strategy helps optimize system resources and enhances overall performance and responsiveness. By implementing these performance measures, the application can effectively manage user demand, ensure scalability, and deliver a seamless user experience even during high traffic periods.

6. Data Dictionary

Term	*Definition*	*Attributes*	*Relationships*
Account	The personal area of a user from which individual settings and data can be accessed.	UserID: Unique Identifier for the user UserSettings: Specific configurations made by the user UserHistory: Data related to user's past actions	An account is owned by a user, it can be associated with multiple carpool and delivery requests
Backend	The server-side interface of the application	N/A	The backend processes requests from the frontend
Car details	Information about user's car including licence plate number, car model, and car color.	licencePlate: Licence plate number carColor: Color of the car carModel: Model of the car	Every car is associated with exactly 1 user account
Carpool Request	A request made by a potential passenger to be picked up and dropped off at a specific location by the driver.	userID: Unique Identifier for the passenger requesting requestID: Unique Identifier for each carpool request timestamp: Time of the request pickupLocation: Location of pickup dropoffLocation: Location of dropoff	Every carpool request is associated with exactly 1 user
Data backups	Copies of all user data for recovery in case of data loss	N/A	N/A

Delivery	Ongoing delivery service after the match was confirmed.	deliveryID: id associated with the delivery deliveryRequester: id of the delivery requester deliveryResponder: id of the delivery responder startPoint: start point of the delivery responder endPoint: end point of the delivery responder pickupPoint: start point of delivery requester dropoffPoint: end point of the delivery requester sharedRoute: shared route followed by the responder fare: Calculated fare for the delivery	Each ongoing delivery is related to a delivery. Requester, a delivery responder, start point, end point, pickup point, drop off point, shared route, and fare
Delivery Rating	The review given by a Deliery requester to a delivery man on a scale of 1 to 5, with 1 being the lowest and 5 being the highest possible rating.	deliveryID: id associated with the delivery rating: rating given on a scale of 1-5	Each rating is associated with a specific delivery
Delivery request	A request made by a potential delivery requester for a parcel to be delivered at a specific location by the driver.	userID: Unique Identifier for the user requesting delivery requestID: Unique Identifier for each delivery request timestamp: Time of the request pickupLocation: Location of pickup dropoffLocation: Location of dropoff details: Details of the parcel	Each delivery request is associated with exactly 1 user

Delivery requester	The app user who is requesting for a delivery to be performed	userID: Unique Identifier for the user requesting delivery	Every delivery requester is related to a user
Delivery Responder	The app user who performs the delivery for a delivery requester	userID: Unique Identifer for the user carDetails: Details of the car	All ongoing deliveries have a responder. All responders are associated with a user and a car belonging to that user
Driver	A person who drives a car, and is on his/her own route, but is willing to fetch a passenger to the passenger's specified destination.	userID: Unique Identifer for the user carDetails: Details of the car	All ongoing rides have a driver. All drivers are associated with a user and a car belonging to that user
Driver Rating	The review given by a passenger to a driver after the passenger has been dropped-off, on a scale of 1 to 5, with 1 being the lowest, and 5 being the highest possible rating.	rideID: id associated with the ride rating: rating given on a scale of 1-5	Each Driver Rating is associated with a specific ride
Drop off point	The agreed upon location where the delivery person drops the package, or where the passenger requests to be dropped off.	location: location of the drop off point	Every drop off point is a location
End point	The location where a driver or delivery man ends his/her journey	location: location of the end point	Every end point is a location
Fare	The monetary amount that a passenger/ delivery requester pays for a completed delivery service	value: amount that the person has to pay from: The userID of the person paying the fare to: The userID of the person receiving the fare	Every fare is associated with 2 users: the user paying the fare and the user receiving the fair
Frontend	The client-side interface of the application	N/A	The frontend interacts with the backend

JWT Token	JSON Web Token used for authentication purposes	token: The random token associated with the user signedBy: he system signing the token expiry:	Every JWT Token is associated with a specific user device
Load balancer	Software or hardware that distributes network or application traffic across a number of servers to improve performance	N/A	The Load Balancer may be applied to the backend to balance heavy traffic
Location	The coordinates of a particular place on the map	latitude: Global Latitude Longitude: Global Longitude	A location is associated with carpool and delivery requests.
Match	A driver and a passenger have successfully, and mutually agreed to carpool	requesterID: The ID of the user requesting the service responderID: The ID of the user fulfilling the service serviceType: Whether the service is a delivery or ride serviceID: The ID of the ongoing service	All Matches are related to the user account of the user requesting a service, the user accepting the service and is also related to the service that is initiated when the map happens
Passenger	A person who needs a ride to his/her destination and is willing to hitch a ride with a driver.	userID: userID of the passenger	Every passenger is a user
Passenger Rating	The review given by the driver for a particular passenger after a passenger has been dropped off, on a scale of 1 to 5, with 1 being the lowest, and 5 being the highest possible rating.	rideID: id associated with the ride rating: rating given on a scale of 1-5	Each Driver Rating is associated with a specific ride
Pick-up point	The agreed upon location where the delivery person picks up the package, or where the passenger requests to be picked up.	location: location of the pickup point	Every Pickup point is a location

Profile Picture	An image that represents the user in the application	userID: ID of the user picture: URI to the picture	Every user account has a profile picture
Review	A written feedback given to a user for their quality of service.	serviceID: id associated with the service review: review given	Each review is associated with a specific service
Ride	An ongoing ride that has started after a match between passenger and driver	rideID: id associated with the ride passenger: id of the passenger driver: id of the driver startPoint: start point of the driver endPoint: end point of the driver pickupPoint: start point of passenger dropoffPoint: end point of the passenger sharedRoute: shared route followed by the driver fare: Calculated fare for the ride	Each ride is associated with a passenger, a driver, a start point, an end point, a pickup point, a drop off point, a shared route and a fare
Route	The course taken in getting from a starting point to a destination	start: startLocation end: endLocation route: route object returned by Google Maps API	Each route is associated with 2 locations
Shared Route	The least time consuming route taken in which the driver can successfully pick up and drop off passengers at agreed upon locations and then reach their own preferred location	start: start Location for driver end: end Location for driver pickup: Location to pick up passenger dropoff: location to drop off passenger route: route object	Each route is associated with 4 locations

		returned by Google Maps API	
Start point	The location where a driver or delivery man starts his/her journey	location: location of the start point	Every start point is a location
Transaction	Payment of money from one user to another for a service done on the app.	fare: The fare from: The userID of the person paying the fare to: The userID of the person receiving the fare serviceID: The service ID related to the transaction	Each transaction is associated with a fare, the user paying the fair, the user receiving the fare, and the service id that caused this transaction
User	Any person who uses the app. Can be a passenger, driver, delivery requester.	name: Name of the User phoneNumber: phone Number of the user userID: Unique user id associated with the user emailaddress: email of the user password: The password of the user carDetails: The car details of the user	Related to most of the entities described above
Wallet	Digital wallet which is used to pay for transactions within the app.	amount: The amount of money stored in digial wallet userID: The user who owns the wallet	Each wallet is owned by a specific user.