Deliverable #1: Software Requirement Specification (SRS) for TaxiMate

SE 3A04: Software Design II – Large System Design

Tutorial Number: T01
Group Number: 10
Group Members:

- Vaya, Rishi
- Bahsoun, Mohamad-Hassan
- Giles, Isaac Ryan Godfrey
- Rajkarnikar, Umang

1 Introduction

1.1 Purpose

The purpose of the following document is to formally specify an app that fulfills the goal of providing carpool matching for a local taxi company. This document thoroughly covers relevant topics including but not limited to; viewpoints, business events, product perspective functionality, constraints, use cases, functional and non-functional requirements, usability, performance, maintainability, security, and legal specifications. This document contains all relevant and necessary information to begin development on a taxi carpooling application.

1.2 Scope

This project is a mobile taxi sharing software application aimed to manage taxi carpool services. The purpose of this application is to create a convenient and easy-to-use application that can handle offers and requests for taxi carpools. The goal is to provide a system which lets users create taxi carpools with other passengers in order to reduce the total cost of a trip. Customers will be provided details of their trip, including estimated fares and travel route. In addition, customers will have the option to match with other passengers based on common music taste. Customers will also have an option to add and remove other passengers to their friend lists on the app, where they can communicate and coordinate future carpools. Overall, this application intends to provide an easy-to-use carpool system that lets customers reduce the total cost of their trip.

1.3 Definitions, Acronyms, and Abbreviations

User:
Individual using the application. For the purposes of this requirements specification document, the terms user and passenger will be used interchangeably.

Offeror:
A user who is offering a carpool ride to other passengers.

Acceptor:
A user who is responding to a friend request from another user (friend requester)

Requester:
A user who is requesting a carpool ride from the application.

Friend Requester:
A user who is sending a friend request to another user (acceptor)

Dispatcher:
Functionality that processes incoming requests

1.4 References

[1]Uber Technologies, “Uber - request a ride,” App Store, 21-May-2010. [Online]. Available: https://apps.apple.com/us/app/uber-request-a-ride/id368677368. [Accessed: 21-Feb-2023].

[2] “Add Maps nbsp;: nbsp; Android Developers.” Android Developers, https://developer.android.com/training/maps.

[3] Rosa, Aurelio De. “An Introduction to the Geolocation API.” Code Envato Tuts+, Envato Tuts, 28 Apr. 2014, https://code.tutsplus.com/tutorials/an-introduction-to-the-geolocation-api–cms-20071.

[4] "Data Encryption 101: A Guide to Data Security Best Practices." Prey Blog, 29 Mar. 2021, https://preyproject.com/blog/data-encryption-101.

[5] Jena, Baivab Kumar. "What Is SHA-256 Algorithm: How It Works and Applications [2022 Edition]: Simplilearn." Simplilearn.com, Simplilearn, 23 Feb. 2023, https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm.

1.5 Overview

This application intends to provide an easy-to-use carpool system that lets customers reduce the total cost of their trip. Below in Section 2, it will further discuss what the product is overall (functions, characteristics, constraints). Section 3 will provide a use case diagram for the system that is being developed. Section 4 will highlight the functional requirements for the system. Lastly, Section 5 will conclude with the Non-Functional requirements of the system.

2 Overall Product Description

2.1 Product Perspective

This SRS outlines a standalone product that is not a part of any larger system but does utilize the Google Maps API. The product is going to provide users the ability to reduce the total cost of their taxi fares by matching them with other users to carpool. This product will be using the Google Maps API for accurate directions and the use of real-time traffic information in order find the best route to various destinations. Users will be able to create an account and customize their profiles, request taxis, and offer carpools all within the product. A feature that separates this product from others is its musical feature. Users will be able to provide their musical tastes in order to listen to their favourite music during their ride. They can even connect with other users who share the same taste in music.

2.2 Product Functions

Customer Management:

- Upon installing the application for the first time, users will be presented with an option to sign up for a new account.
- Upon opening the application for the first time, users will be presented with an option to log in to a pre-existing account.
- Upon opening the application, in the settings tab users will be presented with an option to log out of their account.
- Upon account creation, Users will be prompted to include a description about themselves and things that they like.
- Users will be able to edit their profile anytime after its creation.
- Users can remove their account and profile from the application permanently.

Request Taxi Carpool: Application allows users to request for a carpool ride.

Offer Taxi Carpool: Applications allows users to scan a code, which will be sent to the Dispatcher for processing.

Music Feature:

- Users will be able to specify their music taste for a more enjoyable ride
- Users will be able to view other Users musical tastes.

Friend Feature:

- Users will be able to send friend requests to other users through phone numbers or people they have ridden with.
- Users will have the option to accept or reject incoming friend requests.
- Users will have the ability to chat with their friends on the app to coordinate and schedule carpool times and destination.
- If the option is available, users will be able to request a carpool with their friends.

Co-Passenger Rating: Application will allow users to assess the quality of the carpool passengers by providing a rating form.

2.3 User Characteristics

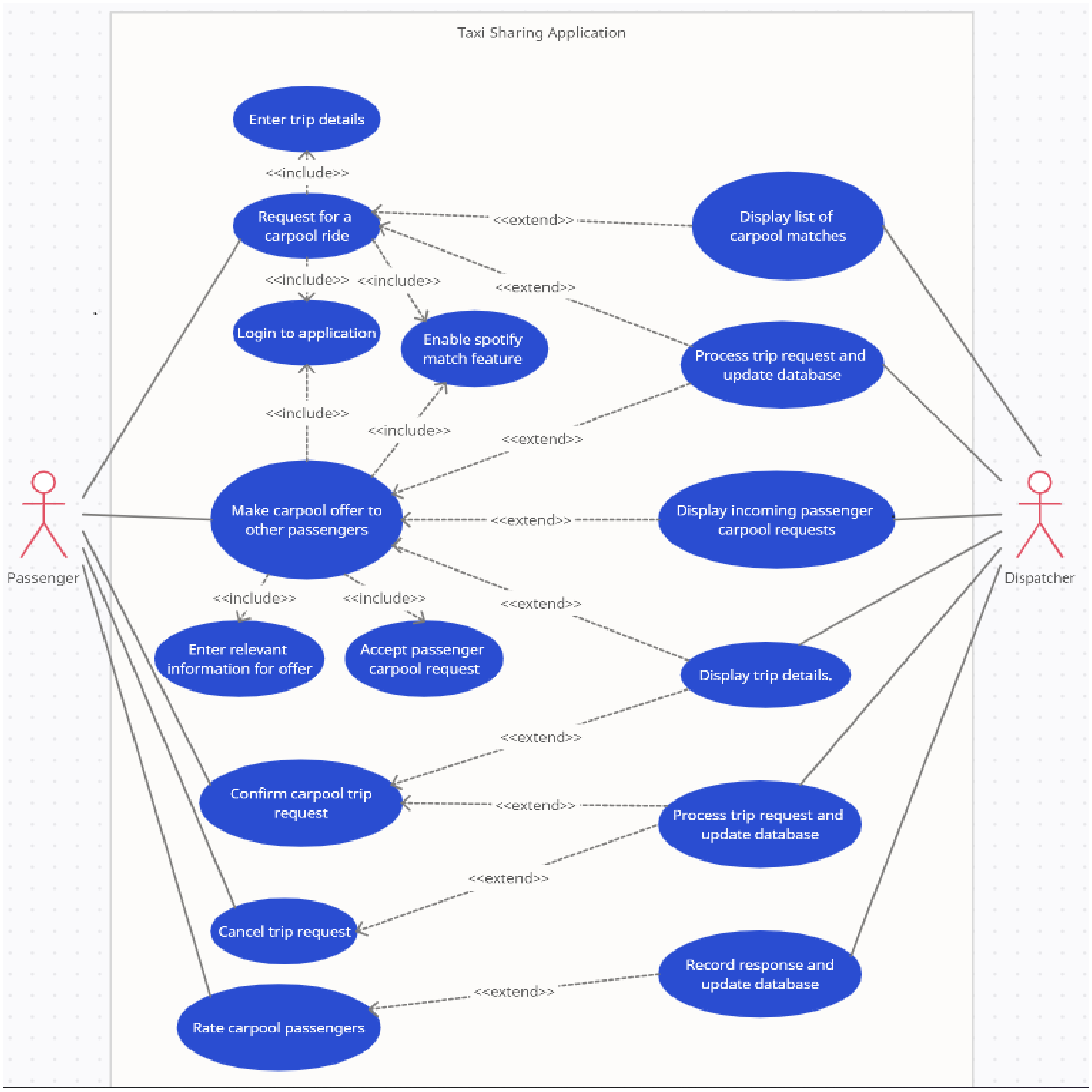
Passenger:

- Should be familiar with using a smartphone

2.4 Constraints and Assumptions

- Assume everyone tells the truth and does not lie when making their accounts
- Assume that users will have some form of internet connection in order for the request to go through
- Assume that this app with be available to Android users
- Assume users are not using a VPN
- Assume there is always an available carpool
- Assume that customers will have cash because they are required to pay for their ride with cash
- Assume that users have a cellular device

3 Use Case Diagram


Taxi Sharing Application
Enter trip details
<<include>>
Request for a carpool ride
<<extend>>
Display list of carpool matches
<<include>>
<<include>>
<<extend>>
Login to application
Enable spotify match feature
Process trip request and update database
<<include>>
<<include>>
<<extend>>
Passenger
Make carpool offer to other passengers
<<extend>>
Display incoming passenger carpool requests
Dispatcher
<<include>>
<<include>>
<<extend>>
Enter relevant information for offer
Accept passenger carpool request
Display trip details.
<<extend>>
Confirm carpool trip request
<<extend>>
Process trip request and update database
<<extend>>
Cancel trip request
Record response and update database
<<extend>>
Rate carpool passengers

4 Highlights of Functional Requirements

BE1. User requests to create an account

VP1. Passengers

- S_1**:** System prompts the user to enter profile information.
- E_1**:** User fills in information.
- S_2**:** System attempts to create and sign up a new user.

  $S_{2.1}$ System successfully creates a new user and saves the user's information. End scenario.

  $S_{2.2}$ System fails to create a new user. Loop back to VP1.S1.

VP2. Dispatcher

- N/A

Global Scenario: A passenger signs up.

- S_1**:** System prompts the user to enter profile information.
- E_1**:** User fills in information.
- S_2**:** System attempts to create and sign up a new user.

  $S_{2.1}$ System successfully creates a new user and saves the user's information. End scenario.

  $S_{2.2}$ System fails to create a new user. Loop back to VP1.S1.

BE2. Users Update their Profile.

VP1. Passenger

- S_1**:** System displays a menu with the following options: Edit Profile, Remove Profile.
- E_1**:** User selects an option from the menu.

  $E_{1.1}$ User selects the Edit Profile option. Proceed to S2.

  $E_{1.2}$ User selects the Remove Profile option. Proceed to S4.
- S_2**:** System displays the user's pre-existing profile information, which includes their name, email, contact information, and payment information. Each of these fields are displayed as editable fields and the contents of each field can be saved to the application.
- E_2**:** The user updates their profile information and initiates the request to save and update their form to the system.
- S_3**:** System processes the user's response and updates the user's profile information. The system displays the user's profile information with the updated information.
- S_4**:** System displays a popup instructing the user that this action is irreversible, and confirms their request to remove their account.
- E_3**:** User chooses whether or not to remove their account.

  $E_{3.1}$ User proceeds with the action to remove their account. Proceed to S5.

  $E_{3.2}$ User chooses to not remove their account. Loop back to S1.
- S_5**:** System removes the user's account and redirects the user to the signup page. Proceed to BE1.

VP2. Dispatcher

- N/A

Global Scenario: Customer makes an update to their profile.

- E_1**:** The user wishes to update their profile information.
- S_1**:** System displays a menu with the following options: Edit Profile, Remove Profile.
- E_2**:** User selects an option from the menu.
  - $E_{2.1}$**:** User selects the Edit Profile option. Proceed to S2.
  - $E_{2.2}$**:** E1.2 User selects the Remove Profile option. Proceed to S4.
- S_2**:** System fetches and displays the user's current profile information.
- E_3**:** The user updates their profile information and initiates the request to save their information to the system.
- S_3**:** System saves the user's updated profile information. End scenario.
- S_4**:** System removes the user's account and saves this information. Proceed to BE1.VP1.S1.

BE3. Passenger Offers a Taxi Carpool

VP1. Passengers

- E_1**:** User scans a barcode located inside the taxicab.
- S_1**:** System processes the barcode information.
  - $S_{1.1}$**:** System successfully processes the barcode information. Proceed to S2.
  - $S_{1.2}$**:** Exception: System is unable to process the barcode information. Loop back to VP1.E1.
- S_2**:** The system prompts the user to enter their information relevant to their offer.
- E_2**:** The user enters information relevant to the carpool offer. Proceed to VP2.S1.

VP2. Dispatcher

- S_1**:** The system saves the user's carpool offer and makes it available for other passengers to view. End business scenario.

Global Scenario: A passenger offers a taxi carpool.

- E_1**:** User scans a barcode located inside the taxicab.
- S_1**:** System processes the barcode information.
  - $S_{1.1}$**:** System successfully processes the barcode information. Proceed to S2.
  - $S_{1.2}$**:** Exception: System is unable to process the barcode information. Loop back to VP1.E1.
- S_2**:** The system prompts the user to enter their information relevant to their offer.
- E_2**:** The user enters information relevant to the carpool offer. Proceed to VP2.S1.
- S_3**:** The system saves the user's carpool offer and makes it available for other passengers to view. End business scenario.

BE4. User Requests for Carpool Ride

VP1. Passenger

- S_1**:** System prompts user to enter trip information and relevant details for their carpool ride.
- E_1**:** User fills out trip information and relevant details for their carpool ride.
- S_2**:** System saves the user's response. Proceed to VP3.S1.
- E_2**:** User selects a carpool ride.
- S_3**:** S3. System saves the user's response. Proceed to VP3.S2.

VP2. Offeror

- S_1**:** System notifies the user of an incoming carpool request.
- E_1**:** User views their request(s) and chooses whether or not to accept a carpool request.

  $E_{1.1}$ User accepts the carpool request.

  S_1 System saves the user's response and checks if the carpool request is still on the way or it has been passed.

  $S_{1.1}$ System sees that the carpool request is no longer on route. Proceed to VP3.S3.

  $S_{1.2}$ System sees that the carpool request is still valid. Proceed to VP3.S4.

  $E_{1.2}$ User chooses to decline or not respond to a carpool request. S1. System saves the user's response. Proceed to VP3.S4.

  S_1 System saves the user's response. Proceed to VP3.S4.

VP3. Dispatcher

- S_1**:** System attempts to find a list of carpool matches.

  $S_{1.1}$ System finds and displays a list of carpool matches to the user. Proceed to VP1.E3.

  $S_{1.2}$ System is unable to find a carpool match with the provided criteria. Loop back to VP1.S1.
- S_2**:** System sends the requester's carpool request to the selected offeror. Proceed to VP2.S1.
- S_3**:** The system notifies the requester that the offer is no longer valid. Loop back to VP1.E3.
- S_4**:** The system updates both passengers with the revised arrival times and trip fares. End scenario.

Global Scenario: User requests for a carpool ride.

- S_1**:** System prompts requester to enter trip information and relevant details for their carpool request.
- E_1**:** Requester fills out trip information and relevant details for their carpool request.
- S_2**:** System saves the user's response and displays a list of carpool matches to the user.
  - $S_{2.1}$**:** System is unable to find a carpool match with the provided criteria. Loop back to S1.
  - $S_{2.2}$**:** System finds and displays a list of carpool matches to the requester. Proceed to E2.
- E_2**:** Requester selects a carpool ride from the list of available matches.
- S_3**:** System forwards the carpool request to the selected offeror.
- E_3**:** Offeror views their request(s) and chooses whether or not to accept the carpool request.
  - $E_{3.1}$**:** Offeror accepts the carpool request.
    - $E_{3.1.S1.}$**:** System saves the user's response and checks if the carpool request is still on route.
      - $S_{1.1}$**:** System sees that the carpool request is no longer on route. Proceed to S4.
      - $S_{1.2}$**:** System sees that the carpool request is still valid. Proceed to S5.
  - $E_{3.2}$**:** Offeror declines carpool request.
    - $E_{3.2.S1.}$**:** System saves response and notifies requester. Proceed to S4.
- S_4**:** S4. The system notifies the requester that the offer is no longer valid. Loop back to S2.
- S_5**:** The system updates both passengers with the revised arrival times and trip fares. End scenario.

BE5. Passenger arrives at destination.

VP1. Passenger

- S_1**:** The system prompts the passenger with the option to rate their carpool buddies.
- E_1**:** User chooses whether or not to fill out the rating form.
  - $E_{1.1}$**:** User chooses to not fill out the rating form. End scenario.
  - $E_{1.2}$**:** User fills out the rating form. Proceed to S2.
- S_2**:** System saves the user's response.

VP2. Dispatcher

- N/A

Global Scenario: Passenger arrives at destination.

- S_1**:** The system prompts the passenger with the option to rate their carpool buddies.
- E_1**:** User chooses whether or not to fill out the rating form.
  - $E_{1.1}$**:** User chooses to not fill out the rating form. End scenario.
  - $E_{1.2}$**:** User fills out the rating form. Proceed to S2.
- S_2**:** System saves the user's response.

BE6. Users send and accept friend requests

VP1. Friend Requester

- S_1**:** The system displays the current list of friends and gives the option to search and add new friends.
- E_1**:** The user searches for another passenger.
  - $S_{1.1}$**:** System is not able to find the user. Loop back to VP1.S1.
  - $S_{1.2}$**:** System finds user. Proceed to E2.
- E_2**:** The user sends a friend request to the selected passenger.
- S_3**:** The system saves the user's response and forwards their friend request to the selected passenger. Proceed to VP2.S1.

VP2. Acceptor

- S_1**:** System displays the user of their list of incoming friend requests.
- E_1**:** User decides whether or not to accept the friend request.
  - $E_{1.1}$**:** User accepts the friend request.
  - $E_{1.2}$**:** User declines friend request.
- S_2**:** System saves the user's response and notifies the passenger sending the friend request of the user's decision. End scenario.

VP3. Dispatcher

- N/A

Global Scenario: A user sends a friend request.

- S_1**:** The system displays the current list of friends and gives the option to search and add new friends.
- E_1**:** The user searches for another passenger.
  - $S_{1.1}$**:** System is not able to find the user. Loop back to S1.
  - $S_{1.2}$**:** System finds user. Proceed to E2.
- E_2**:** The user sends a friend request to the selected passenger.
- S_2**:** The system saves the user's response and forwards their friend request to the selected passenger.
- S_3**:** System notifies and prompts the acceptor to accept or decline the friend request.
- E_3**:** User decides whether or not to accept the friend request.
  - $E_{3.1}$**:** User accepts the friend request.
  - $S_{3.2}$**:** User declines friend request.
- S_4**:** System saves the user's response and notifies the passenger sending the friend request of the user's decision. End scenario.

BE7. Passenger opens friend chat

VP1. Passenger

- E_1**:** Passenger opens an existing chat.
- S_1**:** Existing chat messages are displayed.
  - $S_{1.E1}$**:** Passenger sends a message.
  - $S_{1.E2}$**:** Passenger receives a message.
- E_2**:** Passenger searches for a friend using the search option.
  - S_1**:** System displays list of friends.
  - E_1**:** Passenger selects a friend from the search list.
    - $E_{1.1}$**:** If a chat with this friend exists then proceed to BE7.VP1.S1.
    - $E_{1.2}$**:** If a chat with this friend does not yet exist, then create one. Proceed to VP1.S1.

VP2. Dispatcher

- N/A

Global Scenario: Passenger opens friend chat.

- E_1**:** Passenger opens an existing chat.
- S_1**:** Existing chat messages are displayed.
  - $S_{1.E1}$**:** Passenger sends a message.
  - $S_{1.E2}$**:** Passenger receives a message.
- E_2**:** Passenger searches for a friend using the search option.
  - S_1**:** System displays list of friends.
  - E_1**:** Passenger selects a friend from the search list.
    - $E_{1.1}$**:** If a chat with this friend exists then proceed to BE7.VP1.S1.
    - $E_{1.2}$**:** If a chat with this friend does not yet exist, then create one. Proceed to VP1.S1.

BE8. User unfriends a user

VP1. User Removing Friend

- S_1**:** The system displays the current list of friends matching search criteria and gives the option to search for a specific friend.
  - $S_{1.1}$**:** User selects a friend from their friend list. Proceed to S2.
  - $S_{1.2}$**:** User types in the search bar, and the system updates the search criteria. Loop back to S1.
- S_2**:** System displays the selected friends profile with options to remove this user from their friend list or close the user's profile.
  - $S_{2.1}$**:** User removes selected friend from their friend list. Proceed to VP2.S1.
  - $S_{2.2}$**:** User closes the selected friend's profile. Proceed to S1.

VP2. Friend Being Removed

- S_1**:** System removes the specified friend from the user's friends list.

VP3. Dispatcher

- N/A

Global Scenario: A user removes a friend.

- S_1**:** The system displays the current list of friends matching search criteria and gives the option to search for a specific friend.
- $S_{1.1}$**:** User selects a friend from their friend list. Proceed to S2.
- S_2**:** System displays the selected friends profile with options to remove this user from their friend list or close the user's profile.
- $S_{2.1}$**:** User removes selected friend from their friend list. Proceed to VP2.S1.
- S_1**:** S1. System removes the specified friend from the user's friends list.

5 Non-Functional Requirements

5.1 Look and Feel Requirements

5.1.1 Appearance Requirements

LF-A1. The application interface must display no more than 6 options at any given time, with additional options appearing only after making an initial selection.

Rationale: This will make the application more streamlined, and easier to use for new customers. Many existing applications have too many options being displayed at any given time and this can be overwhelming for new users.

5.1.2 Style Requirements

LF-S1. The application's logo must visually relate to transportation.

Rationale: This is important to convey to users what the purpose of the application is, without needing users to use the app.

LF-S2. The application must use a consistent color palette consisting of no more than 5 colors.

Rationale: This will keep the interface clean, simple, and consistent. This is important as the design of the application should not be overwhelming as the function of the app should be the focus.

5.2 Usability and Humanity Requirements

5.2.1 Ease of Use Requirements

UH-EOU1. The application must offer a report bug feature to directly notify the developers about the details of any bugs that occur.

Rationale: This will provide convenient feedback for developers to improve the application, and will also make customers feel more important because their voices are being heard.

UH-EOU2. System must support a colorblind friendly interface that overlays patterns on buttons for users with a variety of different types of colorblindness.

Rationale: This is important to ensure that all users can use the application without impairment.

UH-EOU3. System must provide text to speech options to support visually impaired users.
Rationale: This is important to ensure that all users can use the application without impairment.

5.2.2 Personalization and Internationalization Requirements

UH-PI1. N/A

5.2.3 Learning Requirements

UH-L1. N/A

5.2.4 Understandability and Politeness Requirements

UH-UP1. N/A

5.2.5 Accessibility Requirements

UH-A1. N/A

5.3 Performance Requirements

5.3.1 Speed and Latency Requirements

PR-SL1. Signing up and logging into the app should take no longer than 30 seconds.

Rationale: This ensures that users are able to quickly access the main carpooling services that the app provides with minimal latency.

PR-SL2. Publishing an update to a user's profile should take no more than 5 seconds.

Rationale: Updating a user's profile information is a one-way write operation to the system, it should be a relatively fast operation.

PR-SL3. The time it takes for the Dispatcher to find a carpool ride once a user has sent their request should take an average of 30 seconds.

Rationale: In order to find a carpool ride, the system will need to find a match based on the user's location and any additional criteria provided by the user. In comparison to other tasks, such as signing up and logging in, the latency is expected to be longer. Furthermore, this will still ensure that users receive a timely response.

PR-SL4. Messages sent between passengers and drivers, as well as passengers to other passengers, should be delivered in no more than 10 seconds.

Rationale: Passengers and drivers should be able to communicate about carpool rides in a timely and convenient manner.

5.3.2 Safety-Critical Requirements

PR-SC1. The application will allow passengers and drivers to file a report on any passenger or driver misconduct.

Rationale: This application may pose a safety risk because it will organize carpools among different individuals. This will allow both passengers and drivers to report any wrongdoing and promote safe travel.

5.3.3 Precision or Accuracy Requirements

PR-PA1. Calculation of estimated fares shall be within $\pm5\%$ of the actual fare.

Rationale: We imagined that users would prefer an accurate calculation of their trip cost because it can greatly assist them in deciding whether or not to use the app to travel.

PR-PA2. Calculation of estimated time shall be within $\pm5\%$ of the actual time taken to reach a destination.

Rationale: We imagined that users would want a precise trip time estimate, if they are in a position where they have to explore alternative routes.

5.3.4 Reliability and Availability Requirements

PR-RA1. The application must always find a transportation option when prompted.

Rationale: The purpose of the application is to provide transportation services to users that are seeking these services, and as such if the application ever fails to do this it fails at its overall purpose.

5.3.5 Robustness or Fault-Tolerance Requirements

PR-RFT1. The application must store a user's current route such that in the case of internet loss, or a system crash; the application will resume its current route upon regaining internet access or rebooting.

Rationale: This is important to ensure that users do not lose access to their travel information in extraneous situations.

5.3.6 Capacity Requirements

PR-C1. The application must ensure low battery consumption as it will run for extended periods of time (sometimes in background during rides).

Rationale: Customers would want an application that is compact in size and does not require a lot of computational power, which would drain the battery life of a phone.

5.3.7 Scalability or Extensibility Requirements

PR-SE1. The database must be able to support a 5% increase in users month after month

Rationale: It is expected that the application will gain users over time, so its database should be able to handle a consistent increase each month.

5.3.8 Longevity Requirements

PR-L1. The application must support devices Android 9 or later.

Rationale: Many users do not use the most recent versions of Android and as such the application should still cater to these users as much as possible. This version information should also be publically available so that users know what version they need to download the app.

5.4 Operational and Environmental Requirements

5.4.1 Expected Physical Environment

OE-EPE1. The application must provide services to users requesting a ride within the physical area that the taxi company services.

Rationale: If a user requests a ride in an area that the taxi company can service, then the application should be able to offer the user a transportation option.

5.4.2 Requirements for Interfacing with Adjacent Systems

OE-IA1. There must be a barcode on the taxicab which the user can scan using the application.

Rationale: In the case that the application is in "Offer Carpool" mode, there needs to be a barcode inside of the taxicab for a user to request a carpool ride.

5.4.3 Productization Requirements

OE-P1. The app shall not exceed 200 mb in download size.

Rationale: Other competing platforms such as Uber have an application size of around 180mb to 190 mb. It is best to keep the application minimalistic and relatively small.

OE-P2. The application must be made available on Android play store for download.

Rationale: Android devices are widely used for mobile applications. Many users will be on Android devices and so, it is important for the application to be distributed on Android.

OE-P3. The application must utilize the Google Maps API to obtain and display route information for carpooling.

Rationale: The Google Maps API will be central to the functionality of the application, the app cannot create routes without it.

5.4.4 Release Requirements

OE-R1. Support up to 5 year old devices, as well as previous versions of Android as far back as Android 9.

Rationale: Many users will be using a device with old Android releases and as a result, it is imperative for our application to support software going back to at least 5 years.

5.5 Maintainability and Support Requirements

5.5.1 Maintenance Requirements

MS-M1. The application must be built upon standard software design patterns and principles.

Rationale: This allows for a faster development time which in turn makes it so new features can be quickly implemented. For example, having standardized boilerplate code for each page of the application ensures a speedy development for this new feature request.

MS-M2. In the presence of faults and errors in the application, minor bugs should be addressed within 1 day. This also includes bug reports sent in by users.

Rationale: Faults, errors, and bugs including typos, timeouts, and inconsistent system behavior should be resolved immediately, as they can really impede a passenger's experience using the app.

MS-M3. In the event that the application undergoes system changes and upgrades, the software should continue to run with little to no downtime. If there is downtime, the application should not be down for more than 1 hour.

Rationale: Since the app will be built using third-party software, such systems may require version upgrades in which case the application will need to migrate to the newer versions. This has the potential to cause downtime on the user's end, which should be minimized.

5.5.2 Supportability Requirements

MS-S1. Users shall be able to access step-by-step instructions on how to use different aspects of the application. These instructions will be provided both in the application and the Taxi company's main website.

Rationale: After understanding the flow at a high level, it may be easier to understand and use the app for first-time users or users who are interested in the app.

MS-S2. There shall be a Frequently-Asked-Section (FAQ) which will be dedicated to answering common questions that users may have.

Rationale: It is understandable that this app will be confusing for many first-time users because it is a new service. In addition, because this application will provide a new mode of travel for many taxicab customers, it is expected that many first-time users will have questions or concerns. The FAQ section will be a page devoted to addressing these inquiries.

5.5.3 Adaptability Requirements

MS-A1. The application shall be ported to various Android platforms.

Rationale: The application will need to be compatible with many Android devices and versions as many users do not have the most updated versions.

5.6 Security Requirements

5.6.1 Access Requirements

SR-AC1. N/A

5.6.2 Integrity Requirements

SR-INT1. N/A

5.6.3 Privacy Requirements

SR-P1. All transmitted messages must be encrypted. **Rationale:** It is important to uphold the privacy of communications between two users.

5.6.4 Audit Requirements

SR-AU1. N/A

5.6.5 Immunity Requirements

SR-IM1. The system must encrypt all stored user data. **Rationale:** This will ensure the safety of user data.

5.7 Cultural and Political Requirements

5.7.1 Cultural Requirements

CP-C1. The app will support multiple languages (languages of the country it will be deployed in but main language is english)

Rationale: The application must be suitable for a diverse user base, especially since Canada is a bilingual country. This means that the application must at least support both English and French to be suitable for use in Canada.

5.7.2 Political Requirements

CP-P1. N/A

5.8 Legal Requirements

5.8.1 Compliance Requirements

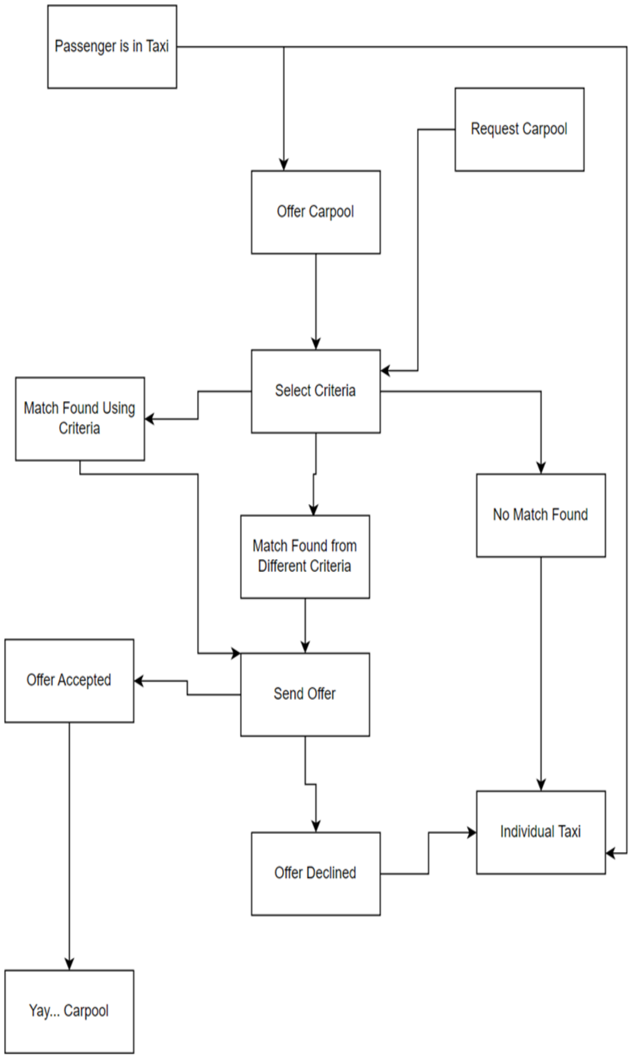
LR-COMP1. N/A

5.8.2 Standards Requirements

LR-STD1. The application mustn't violate any specific privacy laws of the area it is being deployed.

Rationale: If the application violates any laws of the area, the team could end up facing legal action.

A General Flow



B Division of Labour

Mohamad-Hassan Bahsoun
2.1, 2.2 (2 of the product functions), 2.4, 4 (BE 5-6), 5.1, 5.6

Rishi Vaya
1.3, 1.4, 2.2 (1 of the product functions), 2.3, 2.4, 4 (BE 3-4), 5.2

Isaac Giles
1.2, 2.2 (1 of the product functions), 4 (BE 1-2), 5.3, 5.5, 5.6

Umang Rajkarnikar
1.1, 1.5, 2.2 (2 of the product functions), 3, 4 (BE 7-8), 5.4, 5.8

Mohamad-Hassan Bahsoun

[illegible]

Rishi Vaya

[illegible]

Isaac Giles

Isaac Giles

Umang Rajkarnikar

[illegible]