Team Status 200

RideFind
Supplementary Specifications

Version <1.1>

Revision History

Date	Version	Description	Author
01/10/2022	<1.0>	Initial Creation	Isabel Loney
2/10/2022	<1.1>	Added interfaces and standards	Joel Clement

Table of Contents

Supplementary Specifications

1. Introduction

1.1 Purpose

The purpose of this document is to identify and describe the non-functional requirements and constraints of RideFind. It also describes the interfaces of RideFind, including the User Interfaces, Hardware interfaces, Software Interfaces, and Communication Interfaces.

The intended audience of this document includes the prospective developers of the web page and class leaders among EECS 448.

1.2 Definitions, Acronyms, and Abbreviations

Artifact: physical entity that results from an activity. Required artifacts are defined by the software engineering process. Examples of artifacts are SRS, architecture diagrams, UML diagrams, source code, test scripts, user manuals.

API: Application Program Interface. A way for two or more computer programs to communicate.

App: an application, especially as downloaded by a user to a mobile device.

Portal: A vendor's personal website or app where their ride-share services are offered to the public.

Ride-Share: participate in an arrangement in which a passenger travels in a private vehicle driven by its owner, for free or a fee.

React: A JavaScript framework for creating responsive web applications

SRS: Software Requirement Specification.

UML: Unified Modeling Language

Webpage: a hypertext document on the world wide web.

Web App: a client server program. Meaning there is a client-side that speaks to a server-side sending a receiving requests.

W3C: The World Wide Web Consortium. They are the organization that creates the standards for web technologies such as HTML and CSS.

1.3 References

UPEDU_template_example, Ecole Polytechnique de Montreal, 2002 JSP: https://canvas.ku.edu/files/4015731/download?download_frd=1

Merriam-Webster.com Dictionary, Merriam-Webster, https://www.merriam-webster.com/dictionary/rideshare. Accessed Sep. 2022 – Oct. 2022.

2. Assumptions and Dependencies

RideFind depends on the APIs of the Ride-Share services Uber and Lyft. It assumes that data regarding ride options is available through the respective APIs.

3. Usability

3.1 New users should be able to learn to use the app in 2 minutes

3.2 Users should be able to find and book their ride in 3 minutes or les

4. Performance

4.1 API Response time should be <2 seconds

4.2 Page should load in <5 seconds on all devices

4.3 Up to 50 ride options should be shown at once. If there are more than 50 results, the web app should be able to separate the results into multiple pages

5. Design Constraints

5.1 Ride-Share APIs provide a limited amount of data

5.2 The webpage must be developed using the REACT Framework

5.3 The webpage must be compatible with mobile devices

5.4 The webpage must be aesthetically pleasing on mobile devices

6. Online User Documentation and Help System Requirements

6.1 The help page accessed from RideFind should contain thorough and up to date information on how to use the service

6.2 The help page should be usable by people with disabilities (ex. Using screen reader)

6.3 The help page should contain information about searching rides, booking rides, and searching for bus journeys

7. Interfaces

7.1 User Interfaces

The user interface will consist of a map taking up most of the screen. At the top of the screen there will be a search bar. The user can select this search bar and it will bring up the screen to search for and filter rides. From the search screen there are options to specify the starting point, destination, cost & time filters, and an option to include bus services.

7.2 Hardware Interfaces

RideFind will use the geolocation feature of mobile devices. This feature is dependent on the device's hardware supporting it. The app can be used without the geolocation feature.

7.3 Software Interfaces

RideFind will use the React JavaScript framework. It will also use HTML and CSS to form the webpage.

7.4 Communications Interfaces

RideFind will use APIs to communicate with Uber & Lyft servers. It will also use the Google Maps API to interract with Google Maps.

8. Applicable Standards

Since RideFind is a web app, it will need to follow the W3C standards for web applications. These standards cover HTML, CSS, and includes standards for making web apps accessible to people with disabilities.

Team Status 200

RideFind
Use-Case Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
01/10/2022	<1.0>	Initial Creation	Isabel Loney

Table of Contents

Use-Case Specifications

1. Use-Case Model

1.1 Introduction

This webapp consolidates information from rideshare apps (Uber, Lyft) and public transport for users to find their preferred ride. Each type of general user: new user, rideshare user, and public commuter, have access to the same interface to search and rank transport options, but may send different requests from the service.

1.2 General Actors Descriptions

1.2.1. New User: A new user can connect Uber and/or Lyft accounts and will be given an optional tour

1.2.2 Rideshare User: searches and orders a rideshare

1.2.3 Public Commuter: searches for bus routes

2. Use Case 1: New User

2.1 Brief Description

Users who are visiting the page for the first time will be given an optional tour and ride-booking demo.

2.2 Flow of Events

2.2.1 Basic Flow

2.2.1.1 New users are presented with the option to take a tour of the app upon first use

2.2.1.2 Users will be shown how to search for and filter available rideshare options

2.2.1.3 Users will have the option to take a similar tour through the public transit ride search

2.2.1.4 Users will be given the option to link their Uber and Lyft accounts

2.2.1.5 Users will be shown where to access the "Help" page for any further inquiries

2.2.2 Alternative Flows

2.2.2.1 User may skip tour and access the page as a general rideshare user

2.2.2.2 User may use the rideshare tour but skip the public transit tour, or vice versa

2.2.2.3 User may skip connecting their Rideshare accounts, in which case they can connect them later when they book their first real ride

2.3 Special Requirements

2.3.1 The tour shall be thorough and easy to navigate.

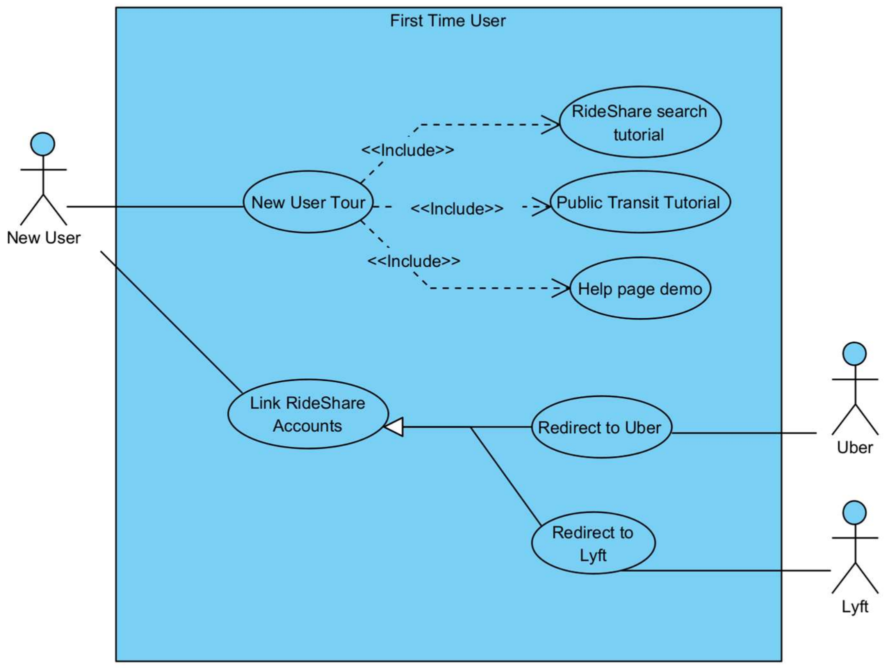
2.4 Preconditions

2.4.1 The user must have never accessed the page from their device before

2.5 Postconditions

2.5.1 After completing the tour and linking their rideshare accounts, the actor now accesses the page as a Rideshare User

2.6 Use-Case Diagrams



3. Rideshare User

3.1 Brief Description

Users who have completed the tour are now able to search and book a rideshare service.

3.2 Flow of Events

3.2.1 The user accesses the main page

3.2.2 The user inputs their starting and ending location

3.2.3 The user has the option to filter and sort based on time, cost, and driver rating

3.2.4 When the user finds their preferred ride, they will select and book it through the appropriate organization

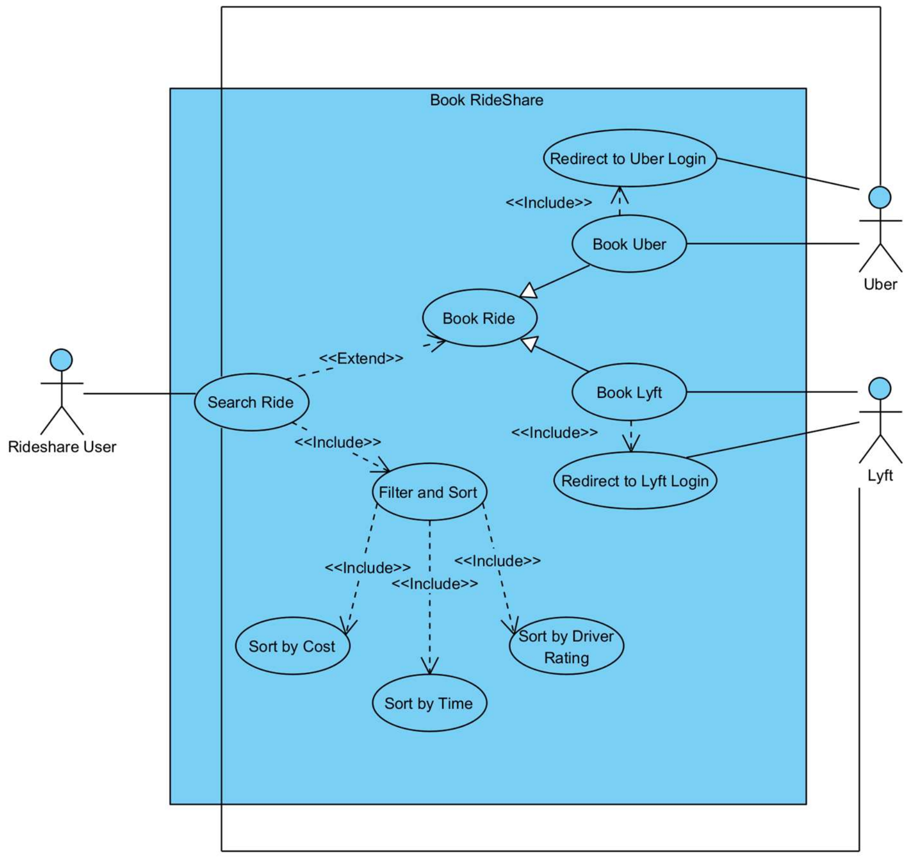
3.3 Alternative Flows

3.3.1 If the user has not yet connected their Uber/Lyft accounts, they will be prompted to do so when they attempt to book their ride

3.3.2 The user will be notified if there are no rideshares available in the area

3.4 Special Requirements

3.5 Use Case Diagrams



4. Public Commuter

4.1 Brief Description

The webpage will include a section dedicated to public transport for non-rideshare users, or rideshare users who have opted to not use rideshare for a specific trip.

4.2 Flow of Events

4.2.1 The user accesses the bus information section

4.2.2 The user inputs their starting and ending location

4.2.3 The user has the option to filter and sort based on cost, transfers, and time

4.3 Alternative Flows

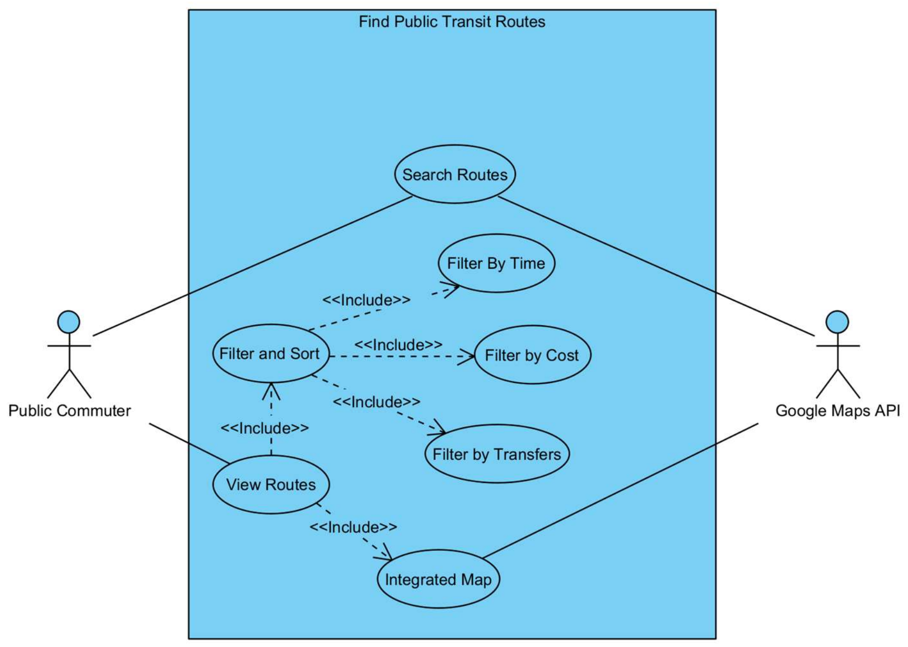
4.3.1 The user will be notified if no viable routes are found

4.4 Special Requirements

4.4.1 This option is dependent on Google Maps API having access to the user's city's public transport schedule

4.4.2 The available routes shall be cleanly displayed on the integrated map

4.5 Use Case Diagrams



Team Status 200<Company Name>

RideFind
Software Requirements Specifications

Version <1.1>

Revision History

Date	Version	Description	Author
9/27/2022	1.0	Initial Creation	William Powers
10/1/2022	1.1	Elaborate on meeting notes and add bus use case; fix some formatting	Isabel Loney
10/1/2022	1.2	Add Hardware interfaces, New User Use Case	Joel Clement
10/1/2022	1.3	Added Product functions, Assumptions/dependencies, fixed formatting, added more info to some sections.	Ehtisham Mushtaq

Table of Contents

Software Requirements Specifications

1. Introduction

1.1. Purpose

The purpose of this document is to describe the requirement specifications for a ride finder display web page for ride share commuters using multiple separate applications. The intended audience of this document includes the prospective developers of the web page and class leaders among EECS 448.

1.2. Scope

The webpage being created is a display map showing the nearest ride share option across multiple separate vendors all centralized on one site. This webpage will be referred to as "RideFind" for the remainder of the document. Ridefind will allow users that have and use multiple separate ride-share apps to find the ride they want based off of their inputted specifications. Whether those be speed, driver rating, cost, size of vehicle, etc., the user will be able to then choose their desired ride which will launch to ordering that ride in that vendors app. With future integration we hope to be able to directly order the car from our page to push further ease-of-use. This webpage could be used by a variety of users including the following main types: already existing ride share app users looking to centralize all of their app feeds, new users to ride share apps searching for a one-time ride, users looking to strictly compare routes for upcoming events or travel.

1.3. Definitions, Acronyms, and Abbreviations

Artifact: physical entity that results from an activity. Required artifacts are defined by the software engineering process. Examples of artifacts are SRS, architecture diagrams, UML diagrams, source code, test scripts, user manuals.

API: Application Program Interface. A way for two or more computer programs to communicate.

App: an application, especially as downloaded by a user to a mobile device.

Portal: A vendor's personal website or app where their ride-share services are offered to the public.

Ride-Share: participate in an arrangement in which a passenger travels in a private vehicle driven by its owner, for free or a fee.

SRS: Software Requirement Specification.

UML: Unified Modeling Language

Webpage: a hypertext document on the world wide web.

Web App: a client server program. Meaning there is a client-side that speaks to a server-side sending a receiving requests.

1.4. References

UPEDU_template_example, Ecole Polytechnique de Montreal, 2002 JSP: https://canvas.ku.edu/files/4015731/download?download_frd=1

Merriam-Webster.com Dictionary, Merriam-Webster, https://www.merriam-webster.com/dictionary/rideshare. Accessed Sep. 2022 – Oct. 2022.

1.5. Overview

The rest of this document contains an overall description of the RideFind webpage functionality and the specific requirements for the system, including system interfaces, user interfaces, hardware interfaces, software interfaces, communication interfaces, memory constraints, and operations.

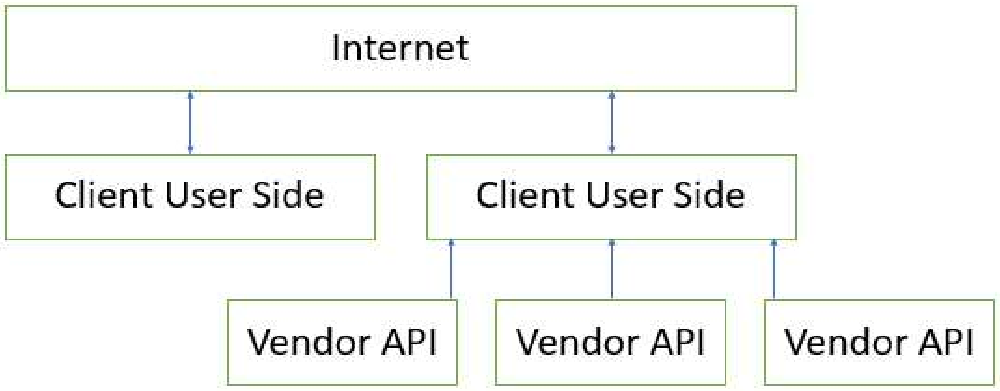
2. Overall Description

2.1. Product perspective

In many urban concentrated areas, there is a large part of the commuter population that use ride-share apps like Uber and Lyft. The delima with this is the extensive number of different vendors offering the same service but through their separate vendor portal. Connecting all of these separate vendors in one centralized location would save users time in sifting through separate apps and allow them to find most coinvent route to be found based on their desired specifications. Some specifications include different filters that will enable them to find a ride based in their input. Some filters include "filter by time," "filter by price," "filter by route."

2.1.1. System Interfaces

RideFind will be developed as a stand-alone client server program. There will be no backend database as no information will need to be recorded from users only a frontend displaying real-time rider/driver feed. The system architecture is modeled as follows:



The client-server flow with start with the user launching the site through their browser in which requests will be sent for initial API data to be displayed in their various fields. From here the user will be able to navigate through the provided information or make requests for different specifications via the API and server end. Client-server communication will be managed over the internet using http protocol.

2.1.2. User Interfaces

The first component of the webpage with display a live map showing your current location or inputted location, and your destination as well as the specifications of the current fastest ride from is corresponding vendor (when site is initially launched default best ride with be fastest arrive time, user can change these settings form here to find their best ride). The second component will consist of an ordered list of drivers ranked by fastest to slowest estimated arrival time (default). In this table the user will be able to filter their results and/or sort them to their liking. The following model shows the initial hypothesized webpage layout:

<Input drawing here>

2.1.3. Hardware Interfaces

RideFind has no backend server since it is a front-end app only, however there is a client-server relationship between RideFind and the Uber/Lyft APIs.

2.1.4. Software Interfaces

The front-end interface must be built on REACT, which is built upon JavaScript, HTML, and CSS.

2.1.5. Communication Interfaces

RideFind will communicate with Lyft and Uber's APIs to retrieve data about nearby rides from each service. RideFind will also communicate with the Google Maps API to display and retrieve routes. The result will be a webpage that displays real time location of Uber and Lyft drives on a live map.

2.1.6. Memory Constraints

RideFind has no memory constraints as it is a front-end app only.

2.1.7. Operations

New users will be given an option to link their Uber and Lyft accounts, but RideFind itself has no login portal. clean google maps integration, easy to use filter and sort. Clear rankings on which driver is best, etc.

2.2. Product functions

RideFind acts as a middle agent between ride-share apps and users. Instead of sifting through various apps like Uber and Lyft to find the best ride, the user will be able to go on our webpage and find a ride based on their needs (if they want the fastest ride, or cheapest, or a driver with good ratings etc.). The landing pages of our website will showcase all the nearby available drivers form different ride-share apps on a live map and the user can choose a ride best suited to them. Our website will also feature a filter feature that will allow the users to quickly glance at drivers based on the filters chosen. For our first iteration, a user will need to go back to the desired ride-share app and book a ride, but our goal in the next iterations is to allow users to directly book from our website.

2.3. User characteristics

The users are current users of ride-share apps and/or public transport. Users that wish to find a ride already have accounts with Lyft or Uber and are familiar with the process of ordering from aforementioned services.

2.4. Constraints

RideFind must operate within a webpage rather than a phone application; the webpage must be functional and accessible on mobile devices. RideFind only has access to publicly available information through Lyft and Uber APIs – including nearby, available rides, but excluding exact location data of drivers.

2.5. Assumptions and dependencies

For the first iteration, there are no assumptions, but for the subsequent iterations, we are planning to incorporate a feature where a user can book directly from our website. In that case, it is assumed that a user has a working Uber or Lyft account and they give permissions to connect our website with their existing working Uber/Lyft accounts. For dependencies, our website has active needs and dependencies on Uber and Lyft. They should actively feed data to our website to allow real-time location of a driver. In conjunction with google maps, all this data will eventually show a live map with Uber and Lyft drivers.

3. Specific Requirements

3.1. RideFind shall communicate with the Google Maps API

3.1.1. The webpage shall implement a Google Map display that displays direct routes from the user's current location to a destination, or a different location to a destination

3.1.2. The user shall be able to search and view public transit routes retrieved from the Google Maps API

3.1.2.1. The user shall receive a specific message if no public transit routes are found

3.2. RideFind shall communicate with Lyft and Uber's public APIs to retrieve available ride information

3.2.1. Users shall be able to view a ranked list of available options when searching for a rideshare

3.2.1.1. The user shall receive a specific message if no rideshares are available

3.3. The user shall be able to sort available rideshares and public transit based on their own preferences

3.3.1. The user shall be able to sort their preferred rides based on timeframes

3.3.2. The user shall be able to sort their preferred rides based on cost

3.3.3. The user shall be able to sort their preferred rides based on driver rating

3.4. The user shall be redirected to the ride-share organizations authentication page to book their ride

3.5. A new user shall be given an optional tour of RideFind and how to use its features

3.6. RideFind shall have a dedicated "Help" page that remains available after the tutorial

4. Classification of Functional Requirements

Functionality	**Type**
3.1 RideFind shall communicate with the Google Maps API	REQUIRED
3.1.1 The webpage shall implement a Google Map display that displays direct routes from the user's current location to a destination, or a different location to a destination	REQUIRED
3.1.2 The user shall be able to search and view public transit routes retrieved from the Google Maps API	DESIRED
3.1.2.1 The user shall receive a specific message if no public transit routes are found	OPTIONAL
3.2 RideFind shall communicate with Lyft and Uber's public APIs to retrieve available ride information	REQUIRED
3.2.1 Users shall be able to view a ranked list of available options when searching for a rideshare	REQUIRED
3.2.1.1 The user shall receive a specific message if no rideshares are available	OPTIONAL

3.3 The user shall be able to sort available rideshares and public transit based on their own preferences	REQUIRED
3.3.1 The user shall be able to sort their preferred rides based on timeframes	REQUIRED
3.3.2 T*he user shall be able to sort their preferred rides based on cost*	REQUIRED
3.3.3 The user shall be able to sort their preferred rides based on driver rating	DESIRED
3.4 The user shall be redirected to the ride-share organizations authentication page to book their ride	DESIRED
3.5 A new user shall be given an optional tour of RideFind and how to use its features	DESIRED
3.6 RideFind shall have a dedicated "Help" page that remains available after the tutorial	DESIRED