Team Status 200

RideFind
Supplementary Specifications

Version 1.3

Revision History

Date	**Version**	**Description**	**Author**
01/10/2022	1.0	Initial Creation	Isabel Loney
02/10/2022	1.1	Added interfaces and standards	Joel Clement
10/10/2022	1.2	Final editing / formatting	William Powers

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 < 3 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-Realization Specification

Version 1.3

Revision History

Date	Version	Description	Author
21/10/2022	1.0	Initial Creation; Introduction section	Isabel Loney
23/10/2022	1.1	Add Sequence Diagrams	Isabel Loney
23/10/2022	1.2	Final Editing / Formatting	William Powers

Table of Contents

Use-Case-Realization Specification

1. Introduction

1.1 Purpose

This document contains information on use-case-realizations for the systems, that being, implementation details for each specified use case. It is intended to capture and convey what objects will be used in the system and how they will collaborate in order to achieve system functionality.

1.2 Definitions, Acronyms, and Abbreviations

Vendor: A company offering ride-share services to the public via their own application interface.

Rideshare: An arrangement in which a passenger travels in a private vehicle driven by its owner, for free or a fee.

Driver: A employee of a rideshare company vendor that uses their car to give others rides and is in turn paid through the vendor.

1.3 References

1. RideFind – Use Case Specifications (Deliverable 4)
2. RideFind – Supplementary Specifications (Deliverable 4)
3. RideFind – Software Requirements Specifications (Deliverable 4)

1.4 Overview

The sections of the Use-Case Realization document describe use cases in terms of their flow of events, participant objects and corresponding diagrams.

2. Use Case 1: First Time User

2.1 Brief Description

Users who are visiting the page for the first time will be given an optional tutorial.

2.2 Flow of Events - Design

Basic Flow: New users are presented with the option to take a tour of the app upon first use, users will be shown how to sort and filter available rideshare options, user will be shown where to access additional resources and help.

Alternative Flow: User may skip tour and access the page as a returning user.

2.3 Interaction Diagrams

Use Case Diagram -> Deliverable 4, Section 2.7
Sequence Diagram -> Deliverable 5, Section 2.3.1

2.3.1 Sequence Diagrams

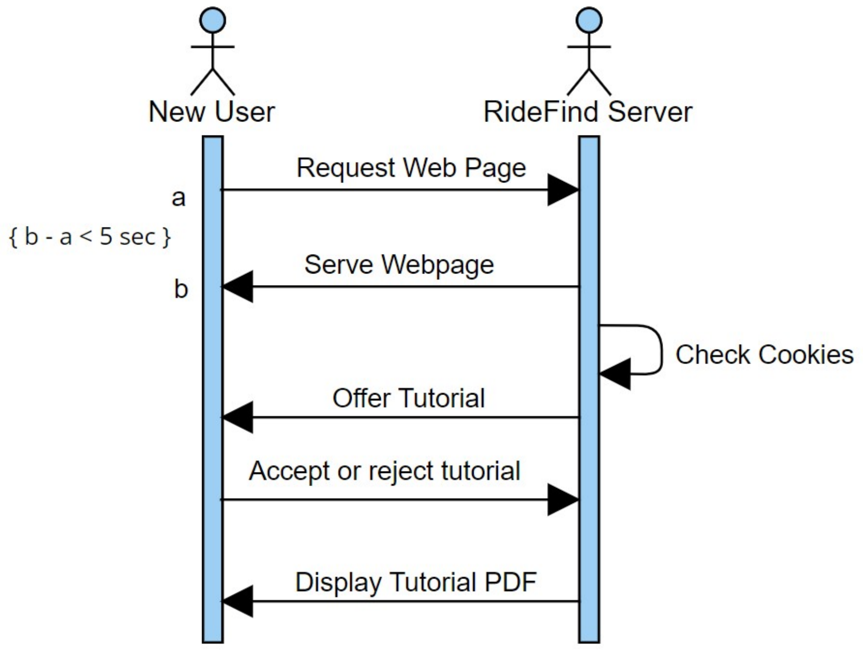


Figure 1: Use Case 1

2.3.2 Participating objects

Object	Description
Tutorial	This represents a pdf with information for new users

2.4 Class Diagrams

Class diagram can be viewed in the Software Architecture Document -> Deliverable 5, Section 5.2.3

3. Use Case 2: Rideshare User

3.1 Brief Description

Returning users that have already completed the tutorial will be able to view local rideshare drivers upon webpage load.

3.2 Flow of Events - Design

Basic Flow: User accesses main page, user inputs their starting and ending location into map, user has option to filter, and sort based on time, cost, vehicle type, and driver rating, when preferred ride is found, user can select and be sent to organizations site to book.

Alternative Flow: If no rideshare driver available in area user will be notified via driver list.

3.3 Interaction Diagrams

Use Case Diagram -> Deliverable 4, Section 3.5

Sequence Diagram -> Deliverable 5, Section 3.3.1

3.3.1 Sequence Diagrams

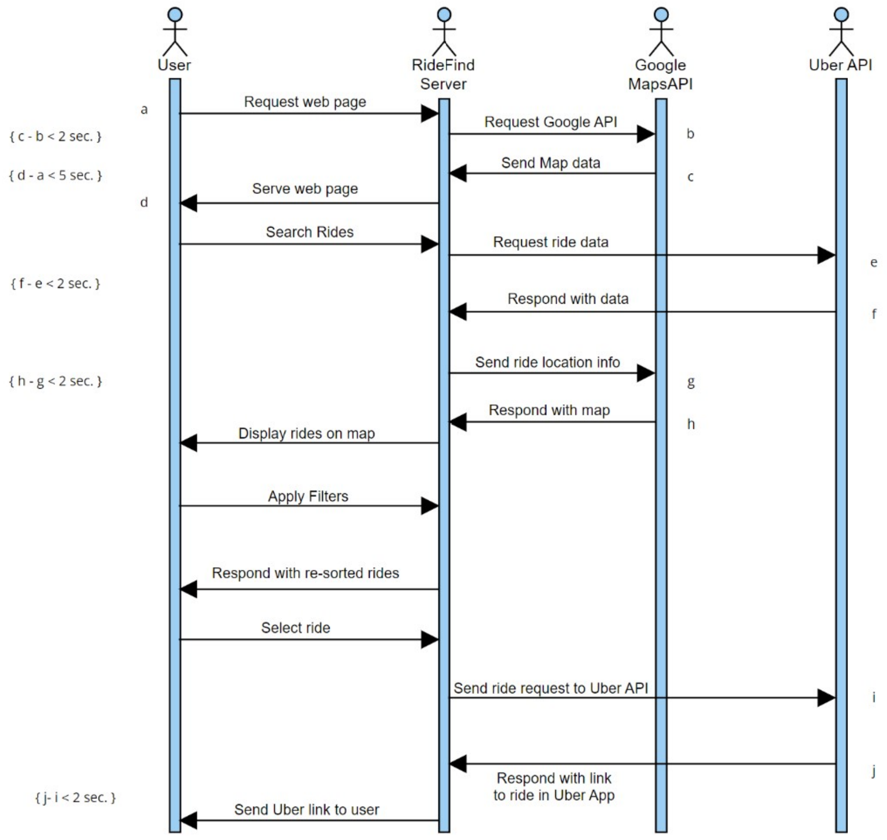


Figure 2: Use Case 2

3.3.2 Participating objects

Object	Description
Main Page	This represents the components of the visible part of the application that allows users to search and view rides
Local APIs	This represents the public APIs of rideshare services that receive requests and respond with data pertaining to a user's desired ride
Google Maps API	This object represents the Google Maps API which provides visual data

	pertaining to a user's location and their desired ride

3.4 Class Diagrams

Class diagram can be viewed in the Software Architecture Document -> Deliverable 5, Section 5.2.3

4. Use Case 3: Commuter <Optional>

4.1 Brief Description

The webpage will include a section dedicated to public transport for non-rideshare users, or user who wish to not use a traditional rideshare driver for their trip.

4.2 Flow of Events - Design

User enters main page, user inputs their starting and ending locations, user can select to view local public transit options that satisfy their trip.

4.3 Interaction Diagrams

Use Case Diagram -> Deliverable 4, Section 4.5

Sequence Diagram -> Deliverable 5, Section 4.3.1

4.3.1 Sequence Diagrams

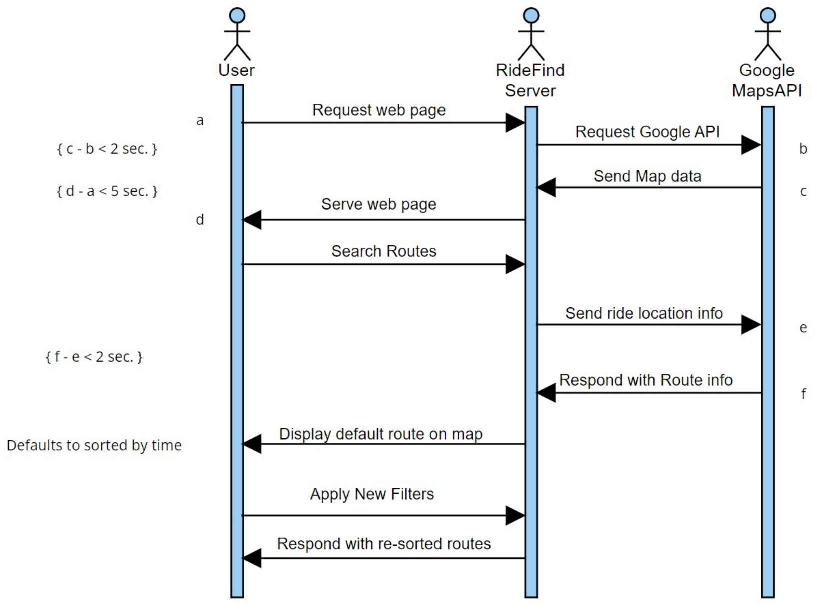


Figure 3: Use Case 3

4.3.2 Participating objects

Object	Description
Main Page	This represents the components of the visible part of the application that allows users to search and view rides
Google Maps API	This object represents the Google Maps API which receives requests and responds with data pertaining to a user's desired public commute

4.4 Class Diagrams

Class diagram can be viewed in the Software Architecture Document -> Deliverable 5, Section 5.2.3

Team Status 200

RideFind
Software Requirements Specifications

Version <1.4>

Revision History

Date	Version	Description	Author
9/27/2022	1.0	Initial Creation, Describing Functional Requirements	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
10/4/2022	1.4	Refactored requirements and Models	William Powers

Table of Contents

Software Requirements Specifications

1. Introduction

1.1. Purpose

The purpose of this document is to describe the requirement specifications for a web page for ride share commuters that displays the best possible ride comparing between multiple different rideshare vendors. 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 comparing multiple different vendors all on one centralized site. This product is a dynamic webpage and 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 their inputted specifications. Whether the user wants lowest ride time, best driver rating, lowest cost, or a specific size/type of vehicle, they will be able to choose that desired ride and launch to ordering that ride in the corresponding vendors app. With future integration we hope to be able to directly order the car from our page to exponentially increase ease-of-use.

This webpage could be used by a variety of users but for now they are grouped into the following three types: already existing ride share app users looking to centralize all their app feeds, new users to ride share apps searching for a one-time ride, users looking to strictly compare routes and rides offered by across different vendors.

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.

Endpoints: A specific point of entry in an API and the main component of making requests to them.

Lyft: The second most used rideshare vendor in the United States.

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

RideFind: Product webpage being built by team.

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

Uber: The most used rideshare vendor in the United States.

Vendor: A company offering ride-share services to the public via their own application interface.

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 via requests and responses.

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.

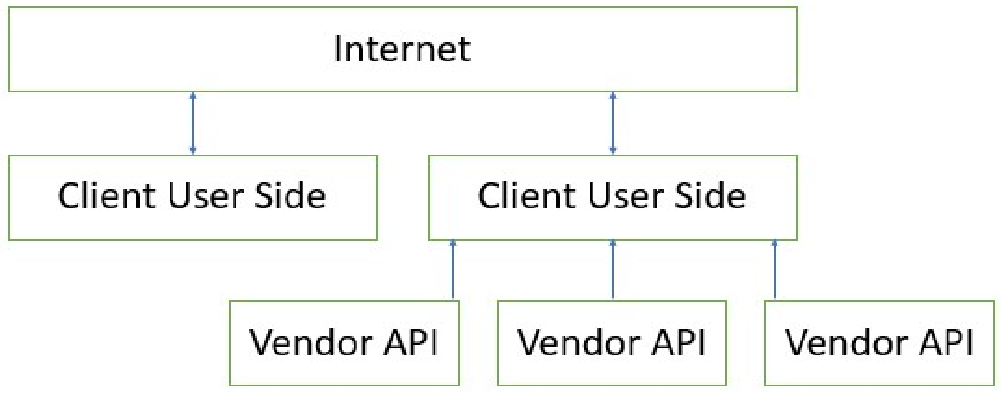
2. Overall Description

2.1. Product perspective

In many urban concentrated areas, a large part of the commuter population uses ride-share apps such as Uber or Lyft. An issue our team sees with this system is the increasing number of different vendors offering the rideshare to the public through their own application interface. With all of these vendors competing for riders in the same location users must open and close each application in order to find the best possible ride, often times settling for a slower or more expensive commute because they were not aware of the other options.
Connecting all of these separate vendors in one centralized location would save users time previously wasted in sifting through different vendors app and increase satisfaction due to being able to choose the best ride for them based on their inputted specifications and bring awareness to smaller companies that only occupy certain areas but are often overlooked in the rideshare market.

2.1.1. System Interfaces

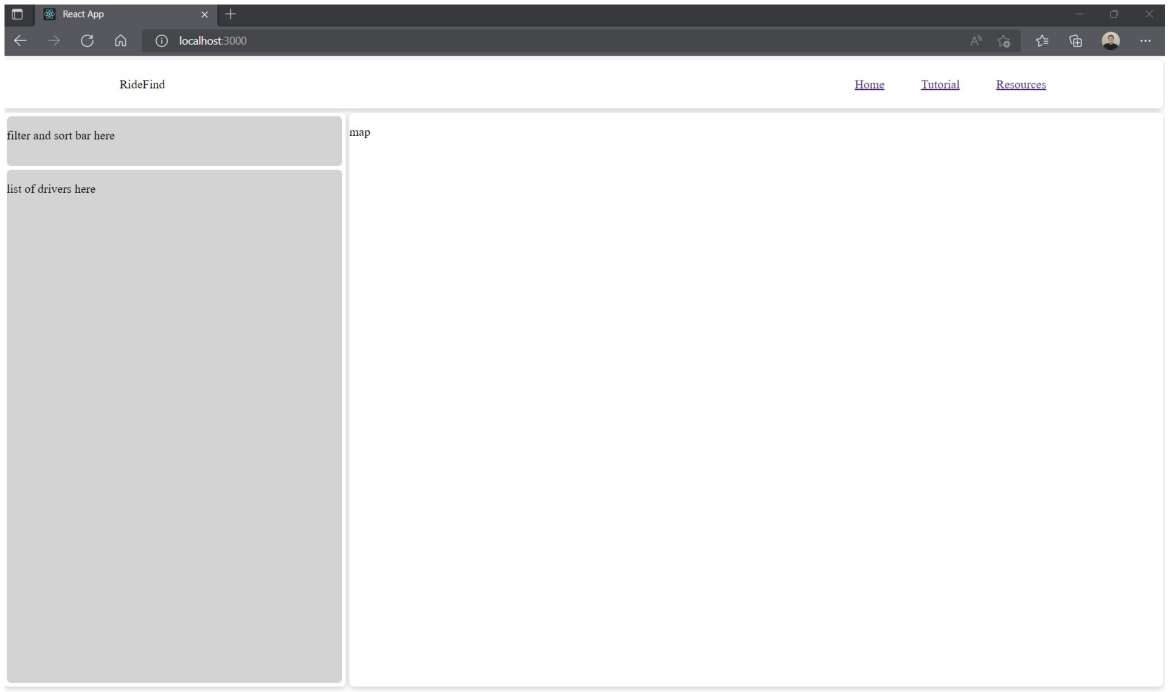
RideFind will be developed as a stand-alone client server webpage. There will be no backend database as no information will be recorded from users therefore only a frontend displaying real-time rider/driver feed is needed. The system interface architecture is modeled as follows:



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 API endpoints. 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:



2.1.3. *Software Interfaces*

The front-end interface will be built on REACT which is built upon the following languages: JavaScript, HTML, and CSS. All API requests will follow suit being made through JavaScript.

2.1.4. *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.

2.1.5. *Memory Constraints*

RideFind may utilize JavaScript's built in localStorage object and store small amounts of previous page information client-side which will not exceed 5 megabytes.

2.1.6. *Operations*

The operation and navigation of RideFind must be very simple and intuitive for all new users. The interface is modeled as a basic sort/filter catalog which will allow minimal clicks to sort or filter rides to their liking.

2.2. Product functions

RideFind acts as a middleman between ride-share apps and users. Instead of sifting through different vendor apps the user will be able to open RideFind and choose a ride based on their needs regardless of the vendor. The landing page will showcase all nearby available drivers on a live map where the user can choose a ride best suited to them. RideFind will also feature filter and sort which will allow users to only view drivers based on the chosen specifications. Once a ride is selected it will be opened in the vendors page to be ordered, confirmed, and paid for. During later iterations this functionality will be brought to RideFind's page where users can order a ride from our site using an API driven widget.

2.3. User characteristics

Users are current users of ride-share apps as well as first time rideshare users. Current users will be able to use the platform instantly via booking their chosen ride while first time users will be sent to the vendors signup page when their chosen ride is selected.

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 rideshare APIs including the following key endpoints: nearby drivers, available ride types, ride cost estimation, ride time estimation, driver rating/experience.

2.5. Assumptions and dependencies

Assumptions include user's area hosts at least one rideshares vendor. Without vendors in nearby location RideFind has no use to user.

3. Specific Requirements

3.1. Functionality

3.1.1. RideFind shall communicate with the Google Maps API.

3.1.2. The webpage shall implement a Google Map that displays direct routes from the user's inputted 'from' location (either current or address) to their inputted 'to' location (address).

3.1.3. The user shall be able to search and view public transit routes retrieved from the Google Maps API (subsequent iteration).

3.1.4. The user shall receive a specific message if no public transit routes are found (subsequent iteration).

3.2. Use-Case Specifications

3.2.1. RideFind shall communicate with vendors public APIs (such as Uber or Lyft) to retrieve available ride information.

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

3.2.3. The user shall view a specific message if no rideshares are available.

3.3. Supplementary Requirements

3.3.1. The user shall be able to sort/filter available rides and public transit based on their own preferences.

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

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

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

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

3.3.6. RideFind shall have a dedicated "Help" page that remains available in and after the tutorial.

4. Clarification 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 that displays direct routes from the user's inputted 'from' location (either current or address) to their inputted 'to' location (address).	REQUIRED
3.1.2 The user shall be able to search and view public transit routes retrieved from the Google Maps API (subsequent iteration).	DESIRED
3.1.2.1 The user shall receive a specific message if no public transit routes are found (subsequent iteration).	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/filter available rideshares and public transit based on their own preferences (subsequent iteration).	REQUIRED
3.3.1 The user shall be able to sort their preferred rides based on timeframes.	REQUIRED
3.3.2 The 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.	REQUIRED
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