Software Requirements Specification

for

Electric Car Charger Locater

**Version 1.0**

**Prepared by Richard DeYoung**

**Central Washington University and Envorso**

**1/9/2022**

# Table of Contents

[**Table of Contents ii**](#_heading=h.30j0zll)

[**Revision History**](#_heading=h.41mghml) Error! Bookmark not defined.

[**1.**](#_heading=h.1fob9te) **Introduction 1**

[1.1](#_heading=h.3znysh7) Purpose 1

[1.2](#_heading=h.2et92p0) Document Conventions 1

[1.3](#_heading=h.tyjcwt) Intended Audience and Reading Suggestions 1

[1.4](#_heading=h.3dy6vkm) Product Scope 2

[1.5](#_heading=h.1t3h5sf) References 2

[**2.**](#_heading=h.4d34og8) **Overall Description 3**

[2.1](#_heading=h.2s8eyo1) Product Perspective 3

[2.2](#_heading=h.17dp8vu) Product Functions 3

[2.3](#_heading=h.3rdcrjn) User Classes and Characteristics 3

[2.4](#_heading=h.26in1rg) Operating Environment 4

[2.5](#_heading=h.lnxbz9) Design and Implementation Constraints 4

[2.6](#_heading=h.35nkun2) User Documentation 4

[2.7](#_heading=h.1ksv4uv) Assumptions and Dependencies 4

[**3.**](#_heading=h.44sinio) **External Interface Requirements 5**

[3.1](#_heading=h.2jxsxqh) User Interfaces 5

[3.2](#_heading=h.z337ya) Hardware Interfaces 6

[3.3](#_heading=h.3j2qqm3) Software Interfaces 6

[3.4](#_heading=h.1y810tw) Communications Interfaces 6

[**4.**](#_heading=h.4i7ojhp) **System Features 7**

[4.1](#_heading=h.2xcytpi) Locating the Best Car Charger (Price or Speed) 7

[4.2](#_heading=h.1ci93xb) Enter in Charging Stations Along a Route and Modify the Time Accordingly 8

[4.3](#_heading=h.3whwml4) Users Can Connect their Subscription Services 9

[**5.**](#_heading=h.qsh70q) **Other Nonfunctional Requirements 11**

[5.1](#_heading=h.3as4poj) Performance Requirements 11

[5.2](#_heading=h.2grqrue) Safety Requirements **Error! Bookmark not defined.**

[5.3](#_heading=h.49x2ik5) Security Requirements 11

[5.4](#_heading=h.147n2zr) Software Quality Attributes 12

[5.5](#_heading=h.vx1227) Business Rules **Error! Bookmark not defined.**

[**6.**](#_heading=h.3fwokq0) **Other Requirements** Error! Bookmark not defined.

[**Appendix A: Glossary 13**](#_heading=h.23ckvvd)

[**Appendix B: High- and Low-Level Design 14**](#_heading=h.ihv636)

[B.1 High-Level Design 14](#_heading=h.32hioqz)

[B.2 Low-Level Design 17](#_heading=h.1hmsyys)

# Introduction

## Purpose

The purpose of this document is to provide a detailed description of the electric car charger locater application. The development of this software will be done by the company Envorso. The document will explain the purpose and features of the software, the software’s interfaces, what the software will do, and the constraints of the software. The document is intended for the developers of the software and the project managers.

## Document Conventions

For this document, all of the functional requirements are linked together by a product feature. All non-functional requirements are listed after the functional requirements and are grouped by type.

## Intended Audience and Reading Suggestions

This intended audience for this document is the developers for the project, Dr. Vajda, and the project managers at Envorso.

The next chapter of this document contains the overall description of the software. It discusses the general idea of the project and the types of users for this product.

The third chapter of this document contains how the product will interact with the different types of interfaces. Specifically, user interfaces, hardware interfaces, software interfaces, and communication interfaces.

The fourth chapter will list out the system’s features and the requirements associated with each feature. Requirements will be analyzed from the perspective of feasibility, validity, and consistency. The analysis appears below each individual requirement.

The fifth chapter will discuss any other non-functional requirements, such as performance and security. Each requirement is analyzed similar to functional requirements but is also ranked in terms of priority.

The document ends with three appendices. The first appendix is the glossary; listing off any terms that are unclear. The second appendix contains flow charts that depict high- and low-level design of this soft. The third appendix contains sketches of what the user interface may end up looking like.

## Product Scope

This software is an electronic car charger locater. Users will be able to enter in all the services they are a part of, and then the software will search for nearby chargers. This will solve the issue of non-Tesla owners have a poor experience attempting to locate charging stations. The software will not handle or store any payment information; it is purely a software for finding the nearest charger, fastest charger, and cheapest charger.

The main benefit to this software is allowing users to be able to use a single application to handle finding the best charger.

Due to this being a Capstone project and not having as much time or resources available, only Ellensburg and Moses Lake will be considered in development. This will allow the team to properly test the functionality of the software. However, by the end of development it will be easy to add any additional services that are located outside of Ellensburg and Moses Lake.

## References

Google Maps API: <https://developers.google.com/maps>

PlugShare API: <https://developer.plugshare.com/docs/>

# Overall Description

## Product Perspective

This software will rely on the pre-existing businesses that have chargers. The software will need to pull the charger information from those businesses, which include charger location, charger price, and charger speed.

Non-Tesla owners do not have a convenient way to find chargers once they are on the road. There are so many charging services in place now. These chargers range in quality and price, and there is no way to know any of this information before pulling up the charger. This application aims to solve that problem. Users would be able to easily locate chargers and plan a trip with charging stops. Users would also be able to be directed to a charger on the fly. Based on what the user wants, the application will direct them to the fastest or cheapest charger.

## Product Functions

* Locate a car charger.
* Track how fast the charger searched for is.
* Allow user to enter in current charge.
* Allow user to store the businesses that they are a part of.
* Allow user to create an account with our software.
* Allow user to plan a trip with stops for charging along the way.
* Allow user to add stops for charging in the middle of a trip.

## User Classes and Characteristics

The intended users for this system are owners of electric cars, especially non-Tesla owners. Non-Tesla owners do not have an all-in-one application like Tesla does. The application should be simple enough for drivers of all technical experience to use. This will not only involve making the interface simple but providing quality documentation to the users in the form of first-time login instructions and help menus.

## Operating Environment

The software will function on mobile devices, both iOS and Android. All functionality must work on both kinds of devices in order for this software to be successful.

## Design and Implementation Constraints

The biggest constraint is time. We will have roughly six to seven weeks to develop a fully functional application, while being full time students. On top of that, creating an application that is cross-platform will be a constraint.

Another constraint is how the application will connect with the subscription’s payment services. Since the application will not be saving payment information locally, the application must ensure that a user does not have to log into the subscription every time they want to pay for a charger.

## User Documentation

Users will be provided with a first-time login tutorial. This will guide users through the software and all of its features. A help menu will also be in the app for users to enter if they ever need help navigating the software.

## Assumptions and Dependencies

This application will be using existing APIs to make development smoother. These APIs will be for creating the map interface and locating the chargers.

It is assumed that the end result for this product is a prototype. Most if not all of the database used will be filled with dummy accounts and information to allow easy testing.

# External Interface Requirements

## User Interfaces

The first time a user enters the application, they will be prompted with a first-time login in screen. Here they can make an account with the system. They will also be able to enter in their vehicle(s), and any charging services they are already subscribed to. Users will be able to add or remove services at any point. The first-time login is a convenient way for users to enter in their pre-existing subscriptions all at once. Once the user logs in, they will stay logged in on that device; much like how other apps operate.

This screen may be skipped entirely if the user does not want to create an account with the application and just wishes to search for a charger. The downside to this is the user will not be able to fully experience the app.

Once users log-in, they are able to search for chargers or plan a trip. Before a user can search for a charger, they must enter in the following information: their location and the current charge of their vehicle. From that, they will be given the chargers that are close by, charge quickly, and are cheap.

For planning a trip, users can enter a route (think Google Maps) and can search for chargers along the way. They will enter in their current charge, location, destination, and if they want to prioritize speed or price. The app will calculate how long before a charge is needed and add that stop to the route. The app will also check to see if multiple stops will be needed based on how long the car can last with a full charge. At each stop, the user will be able to choose which charger they wish to stop at. Once all stops have been found, a route will be given to the user and the total trip time (charging time included) will be presented.

Another interface to be added is voice functionality. This will allow users to use this software in a safe manner while driving.

See Appendix C for sketches of the different screens.

## Hardware Interfaces

This software will function on iOS and Android, so both operating systems need to be considered during development. This includes design of the user interface. The software also needs to properly connect to the internet to be able to interact with the chargers.

The software will also need to interact with the phone’s microphone for the implementation of voice commands. The software must be able to receive voice commands correctly and respond appropriately.

## Software Interfaces

The software will connect with each subscription service’s website to handle payment. Since all of the user data will be stored on an internal database, the system can easily autofill all necessary information when it comes time to pay for a charger.

Data shared between interfaces includes a vehicle’s charger port, the charge left on the vehicle, the cost of a certain charger, the speed of a charger, the user’s current location, and the user’s destination.

The software will also connect with at least one API. One of the APIs will be Google Maps. This way the development team will not have to create a working GPS system from the ground up. Another potential API is PlugShare, which will help to locate chargers and their relevant information. However, this API may not be used because it may not have everything the client desires, and it would be harder to customize an API.

## Communications Interfaces

The application will have to launch an external website to pay for the charge. This will all be done automatically, to make it easy and fun for users. The application will be able to know which website to go to based on what charger the users are at. As the application grows, more services will be added. To begin, only the chargers in Ellensburg and Moses Lake will be considered, since that is where the team is located.

For Ellensburg, the app needs to support Electrify America, Clipper Creek, Chevy, Green Lots, and Power Charge.

# System Features

## Locating the Best Car Charger (Price or Speed)

**4.1.1 Description and Priority**

Finding the best car charger involves searching for a charger near the location entered, andweighing the current charger, the speed of the charger, and how expensive the charger is. This is a top priority feature.

**4.1.2 Functional Requirements**

1. Users can enter in the current charge of the vehicle.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
2. The software can match the vehicle’s charge port with chargers nearby. The software will not present users with chargers their car is incompatible with.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
3. The software will be able to determine the speed of the charger located.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
4. The software will be able to find the price of the charger.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
5. The software will be able to determine how long it takes for a vehicle to charge fully.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
6. The software will be able to determine which charger is the best based on price or speed of charger (up to user to decide).
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
7. The software will be able to determine if a charger is currently in use.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.

## Enter in Charging Stations Along a Route and Modify the Time Accordingly

**4.2.1 Description and Priority**

Users will also be able to plan out a route, similar to what Tesla has. This involves users entering in a starting point and destination, along with when they want to stop for a charge. The software will then modify the travel time based on how many stops the user will make, and how long a charge will take based on the method developed in section 4.1. This feature is medium priority as it may be more time consuming to implement. It also won’t be able to function until section 4.1 is implemented properly.

**4.1.2 Functional Requirements**

1. The software will be able to function like a mapping software (think Google Maps).
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* Assuming that the development team has access to the Google Maps API or something similar, this requirement is feasible. Otherwise, the development team does not have the time or resources to create a map-like API in the given timeframe.
2. Users will be able to add stops based on when they want to charge.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible if we have access to the Google Maps API.
3. The software will present users with the best charger near the requested stop (See section 4.1).
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
4. The software will modify the travel time based on how long each stop will take.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
5. The software will allow users to add stops while on the trip.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
6. The software will respond to voice commands to allow users to easily search for a charger while driving.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement may be tough to implement. The software would have to be able to recognize different kinds of voices. There will also have to be set commands and responses, which would have to be given to the user.

## Users Can Connect their Subscription Services

**4.2.1 Description and Priority**

Since there are a lot of different subscription services that electric car owners can be a part of, the software should be able to store these. This makes it easier for the software to know which chargers to offer up to the user. This is a top priority as it makes the feature in section 4.1 operate better.

**4.1.2 Functional Requirements**

1. The software will require a place for users to enter in subscription services.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
2. The software will need to store each user’s subscription service on a database.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.

# Other Nonfunctional Requirements

## Performance Requirements

1. The charger locater must return results to the users in a quick and efficient manner.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.

## Security Requirements

1. Users will be prompted with a first-time sign in to create an account and enter their subscription services.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.
2. The software will store a user’s name, email, home address, phone number, credit card information, and anything else a subscription service needs for payment and signup.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.
3. The database that stores user data must be protected to prevent user data from leaked to external sources.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.

## Software Quality Attributes

1. The software will be a cross-platform mobile application. The software must function on iOS and Android.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is not feasible. In the given timeframe, it would be difficult to develop a cross-platform application that works well on both iOS and Android. It is feasible however to develop the application in a cross-platform environment set it up to be easily ported.
   * *Priority:* This requirement should be met. The development team should prioritize iOS, and then focus on getting the application to work on Android.
2. The software must be usable by people of all technical skill levels.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.
3. Help menus and tutorials should be present in the software.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met.
4. The software must automatically pay for the charger once the user confirms they wish to use the charger.
   * *Consistency:* This requirement is consistent with other requirements.
   * *Validity:* This requirement is valid.
   * *Feasibility:* This requirement is feasible.
   * *Priority:* This requirement must be met. This makes the process more streamlined and makes the software easier to use.

# Appendix A: Glossary

**API:** Application Programming Interface. To enables a piece of software to easily connect to another piece of software

# Appendix B: High- and Low-Level Design

## B.1 High-Level Design

Below are three diagrams representing the High-Level view of the application. The first diagram (Figure 1) shows the flow-chart of a user opens the application. The second diagram (Figure 2) shows the flow-chart of the main operations of the application (locating a charger or planning a trip). The third diagram (Figure 3) shows the flow-chart of a user searching for a charger while driving. In Figure 2, it is assumed that the user is not yet driving, or they are driving but have stopped. Figure 3 represents the user driving and using the application. They may use their voice or have a passenger use the application (hopefully they do not use their phone while driving).

Diagram

Description automatically generated

*Figure 1: Flow-Chart of a User Opening the Application*

Diagram

Description automatically generated

*Figure 2: Flow-Chart of Locating a Charger or Planning a Trip*

Diagram

Description automatically generated

*Figure 3: Flow-Chart of a User Locating a Charger while driving*

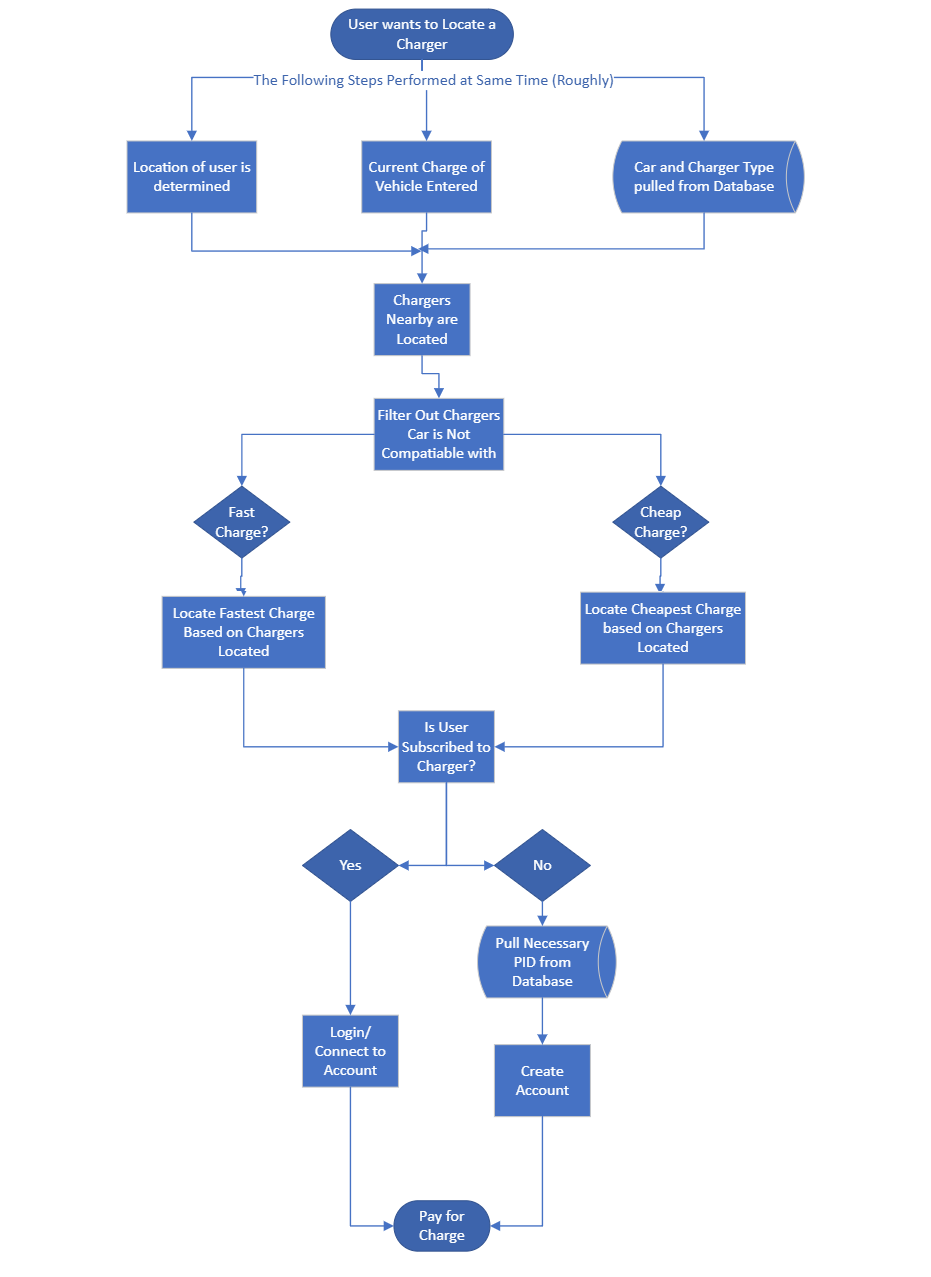
## B.2 Low-Level Design

Below are the Low-Level Designs for the various features. The first diagram (Figure 4) is a flow-chart of the sign-up process for the application. The second diagram (Figure 5) depicts how the software will locate a charger. The third diagram (Figure 6) depicts how the software and user will plan a trip, taking charging time into account.

Diagram

Description automatically generated

*Figure 4: Flow Chart of Sign-up Process*



*Figure 5: Charger Location Flow Chart*

Diagram

Description automatically generated

*Figure 6: Flow-Chart for Planning a Trip*

# Appendix C: User Interface Sketches

This section contains a series of sketches for the user interfaces. The interfaces match section 3.1 in terms of what they should do. (Please note: screen sizes and size of icons, buttons, and text entry are not final. For this section, just the layout of the screens was considered. In the final product, the widgets will fit to their screens).

Diagram

Description automatically generated

*Figure 7: First-time Launch Screen Layout*

Diagram

Description automatically generated

*Figure 8: Map Screen Layout*

Diagram

Description automatically generated

*Figure 9: Trip Planning Screen 1 Layout*

Diagram

Description automatically generated

*Figure 10: Trip Planning Screen 2 Layout*

Diagram

Description automatically generated with low confidence

*Figure 11: Locate Charger Screen 1 Layout*

Diagram

Description automatically generated

*Figure 12: Locate Charger Screen 2 Layout*

Letter

Description automatically generated

*Figure 13: Pay for Charge Screen Layout*