# 

# Envorso Car Charging App

# Software Requirements Specification

Craig Turnbull, Kirsten Boyles, Joe Corona, Lucas Keizur, Richard DeYoung

1/27/2021

Central Washington University

Contents

[**1. Introduction 3**](#_heading=h.3znysh7)

[1.1. Problem Statement 3](#_heading=h.2et92p0)

[1.2. Purpose 3](#_heading=h.tyjcwt)

[1.3. Scope 3](#_heading=h.3dy6vkm)

[1.4. Definitions 3](#_heading=h.1t3h5sf)

[1.5. references 3](#_heading=h.2s8eyo1)

[1.6. Overview 4](#_heading=h.17dp8vu)

[**2. Overall description 4**](#_heading=h.3rdcrjn)

[2.1. Product perspective 4](#_heading=h.26in1rg)

[2.1.1. User Interfaces 4](#_heading=h.lnxbz9)

[2.1.2. Hardware interfaces 4](#_heading=h.35nkun2)

[2.2. Product functions 4](#_heading=h.1ksv4uv)

[2.3. User characteristics 4](#_heading=h.44sinio)

[2.4. Constraints 4](#_heading=h.2jxsxqh)

[**3. Specific requirements 5**](#_heading=h.z337ya)

[3.1. Functional requirements 5](#_heading=h.3j2qqm3)

[3.1.1. High Priority 5](#_heading=h.1y810tw)

[3.1.2. Medium Priority 5](#_heading=h.4i7ojhp)

[3.1.3. Low Priority 5](#_heading=h.2xcytpi)

[3.2. Nonfunctional Requirements 5](#_heading=h.1ci93xb)

[3.2.1. High Priority 5](#_heading=h.3whwml4)

[3.2.2. Medium Priority 6](#_heading=h.2bn6wsx)

[3.2.3. Low Priority 6](#_heading=h.qsh70q)

[**4. Requirements Analysis 6**](#_heading=h.3as4poj)

[4.1. Feasibility 6](#_heading=h.1pxezwc)

[4.2. Consistency 6](#_heading=h.49x2ik5)

[4.3. Validity 6](#_heading=h.2p2csry)

[4.4 Conflict Resolution 6](#_heading=h.147n2zr)

[**5. Design 7**](#_heading=h.3o7alnk)

[5.1 High-Level Design 7](#_heading=h.23ckvvd)

[5.2 Low-Level Design 11](#_heading=h.ihv636)

[**6. Conclusion 12**](#_heading=h.32hioqz)

[**7. Table of Figures 12**](#_heading=h.1hmsyys)

# 1. Introduction

## 1.1. Problem Statement

Tesla owners have a very robust user experience. Using a Tesla app, they can easily find compatible charging stations near them. They can also plan trips around when and where they will need to charge their electric vehicle (EV) based on distance, cost, and time to charge.

People that own other EVs (such as Ford) have an inferior experience. The planning software is inaccurate, and non-Tesla chargers vary in quality, speed, and distance. There are also multiple charger networks that they will likely need to pay for, especially when travelling larger distances.

The goal of this project is to improve the user experience of non-Tesla EV owners by creating an iOS app that uses federation single-sign-on to access all charging services in one place. The app should compete with the Tesla experience by allowing them to plan trips using the chargers along the way. The user should be able to prioritize speed, cost, or distance based on their preferences.

## 1.2. Purpose

The purpose of this document is to compile and analyze all software requirements set by the clients to develop an app to improve the user experience for non-Tesla electric car owners. This document is designed for use by any team members assigned to the development of this project, including but not limited to:

* Team leaders
* Programmers
* Designers

## 1.3. Scope

The app described in this document should allow electric car owners to plan trips, choose charging stations, and pay for service more easily. The app should find and navigate to nearby compatible chargers, and display the charge time for each station. Additionally, the app should implement federation single-sign-on for all or most charging services so that the user does not need to use different apps for different chargers and services.

## 1.4. Definitions

Client – the organization that has commissioned the software and introduced the requirements.

Team – the group of software engineers assigned to this project.

EV – Electric Vehicle.

## 1.5. references

This document is supplemented with the following resources.

“Smartcar · API platform for connected car data” (2022). https://smartcar.com/. Retrieved 1/12/2022.

“ChargePoint API” (2022). https://github.com/lefthandwriter/ChargePointAPI. Retrieved 1/12/2022.

“NREL All Stations API” (2022). https://developer.nrel.gov/docs/transportation/alt-fuel-stations-v1/all/. Retrieved 1/26/2022.

## 1.6. Overview

The remainder of this document will go into detail about the requirements and design of the software. Section 2 will cover the overall description of the software, including the product perspective, product functions, user characteristics, constraints, and assumptions and dependencies. Section 3 will organize the specific requirements of the software, organized by priority, and partitioned by essentiality to the deployment of the software. Section 4 will cover the analysis of the requirements based on feasibility, consistency, and validity. Section 5 contains one possible solution for the high and low-level design for the application. Section 6 concludes the document, and section 7 contains a table of figures.

# 2. Overall description

## 2.1. Product perspective

The calculator software described in this document will not be a part of any larger software suite, and should be developed as a standalone product.

### 2.1.1. User Interfaces

Envorso branding may be used in the user interface, but there are currently no hard requirements on this. The team may use creative freedom when designing the UI. These designs should be pitched to the client early to prevent lost time. Because the software is a mobile app, it should be designed with touchscreen in mind. The app should also be voice compatible, as the user may be using it while driving.

### 2.1.2. Hardware interfaces

The product should run on iOS devices, with the potential to be ported to Android later.

## 2.2. Product functions

The app will primarily help users find nearby compatible car chargers and see how long a charger will take to charge. The app should display other information for a charger, including plug types, voltage, price and associated subscriptions. It must also only require the user to sign in once to use any services they may pay for, and as such the app should be a one-stop product for any non-Tesla EV owners. The app should allow the user to plan trips based on the chargers along the way.

## 2.3. User characteristics

The target users are all non-Tesla EV owners. The user will typically be searching for the closest charger in their area that is compatible with their vehicle. The user may not own a home charger, or they may be planning a trip. The typical user is likely to have experience with using mobile apps.

## 2.4. Constraints

The software must be completed before March 3, 2022.

Beyond that, it is critical that the software handles all errors and prevents all crashes.

# 3. Specific requirements

This section will list all the requirements based on the client’s feedback. The functional requirements will be listed first, followed by the nonfunctional requirements. These requirements will be ordered from highest to lowest priority. The requirements are also either *High priority* (required), *Medium priority* (requested but not necessary), and *Low priority* (suggested but not necessary).

## 3.1. Functional requirements

### 3.1.1. High Priority

1. The app must allow users to find and navigate to nearby chargers.
2. The user must be able to view relevant information on each charger, including
   * distance from user
   * charging speed
   * socket type
   * associated service
3. The app must implement a SSO system for most or all charging services in the supported area.
4. The user should be able to sign in and be remembered to keep their settings and subscriptions.
5. The app should have a brief first-launch tutorial to demonstrate how to use the app, as well as help dialogues that can be accessed when needed.
6. The app should securely store membership information to identify which services the user has paid for.
7. The user should be able to filter results based on aspects such as charging port type, speed, and price.
8. The user should be able to prioritize results by charging speed, distance, or price.
9. The app should be able to plan long trips by finding chargers on the path and calculating the charge necessary to complete the trip.

### 3.1.2. Medium Priority

1. The app should use branding such as logos and color palettes. This may be Envorso-themed or completely original.

### 3.1.3. Low Priority

1. The app should have a button for users to send feedback or request features.

## 3.2. Nonfunctional Requirements

### 3.2.1. High Priority

1. The app should connect one or more databases to support most or all major charging services.
2. The user should support most non-Tesla electric cars, notably Ford.
3. The app should function anywhere in the United States.
4. The app should run on iOS.
5. The service only needs to support a test environment number of users, but should be scalable to any level of growth later.
6. The app should remember the user after initial sign-on.
7. The app should be completed before March 8th, 2022.
8. The app should use voice commands, so it is safe to use while driving. This means that the user should be able to navigate to a charger without touching or looking at the screen.

### 3.2.2. Medium Priority

1. The app should be ported to Android or show potential to be ported later.
2. The development team should avoid using paid tools, and total costs should be near zero.

### 3.2.3. Low Priority

1. The app may optionally expand coverage to EU regions.

# 4. Requirements Analysis

This section will analyze the requirements given in section 3 to determine if they conflict with the capabilities of the team or with other requirements. First to be covered is the feasibility analysis, then the consistency check, and finally the validity check.

## 4.1. Feasibility

Completing the project by the deadline without spending money on tools will be achievable, especially when using free EV APIs such as Smartcar or ChargePoint. The client notes that there will likely not be enough time to port the app to Android or to expand outside the US, so these should be considered stretch goals.

## 4.2. Consistency

There are no consistency errors in these requirements as of this document version.

## 4.3. Validity

Discussion of the UI requirements has been kept vague, and the team has been given creative freedom for UI design. Therefore, and UI designs should be proposed to the client early on.

## 4.4 Conflict Resolution

There are currently no conflicts that need to be resolved. Should such issues arise, the team may compile questions into an email for the client to be sent by the point-of-contact, or it may be brought up in a client meeting.

# 5. Design

The following section will cover one possible design that may be implemented and expanded upon to meet the requirements. Section 5.1 will cover the high-level design concepts including the user interface and functionality, and section 5.2 will cover the low-level design including the class relationships and APIs.

## 5.1 High-Level Design

This section will first discuss a visual mockup, then provide two flowcharts: one for the touch input progression, and one for voice input progression.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated















*Figure 1* above shows one potential design. The app opens to the left display, with the map centered on the user and with recommended charging stations highlighted in orange. An indicator shows the percentage of battery remaining in the user’s car. A tray at the bottom of the screen may be swiped up to display a list of chargers instead, and a button in the top right lets the user sort and filter charging stations. Charging time is displayed above each station. Users can rate chargers upon expanding its info card, and a button on each card lets the user quickly begin in-app navigation to the station.

*Figure 2* below shows the initial signup page and the setup page. The user sees the left screen when first opening the app, or when not signed in. If the user does not want to sign in, they may swipe down to use the app without customization.

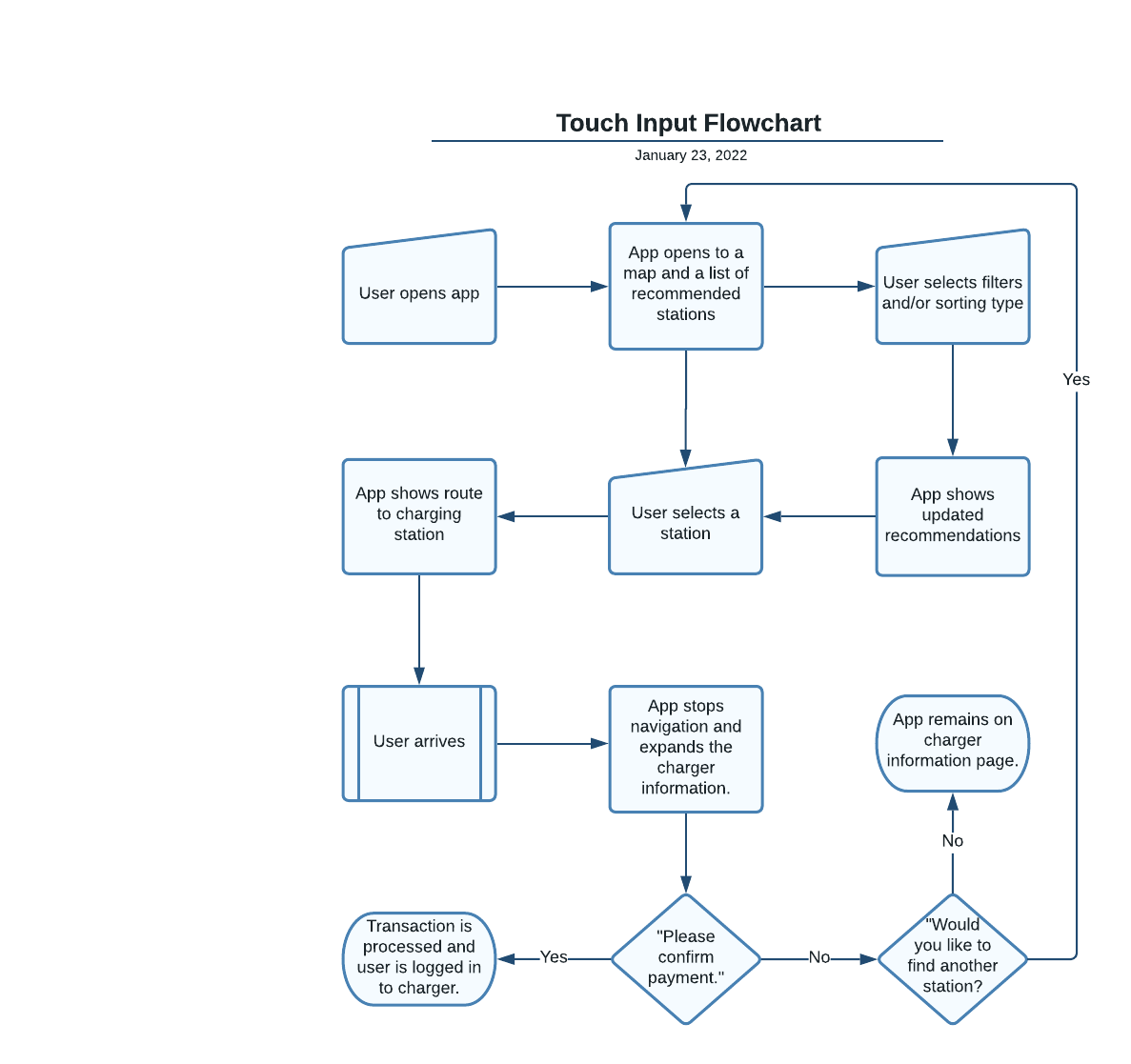
The right screen displays after signing up. By selecting settings such as charger type, range, voltage, and price, the app will be able to recommend chargers without needing the user to apply settings later. A similar page would be displayed when the user selects the filter button seen in *Figure 1.*

Graphical user interface, application

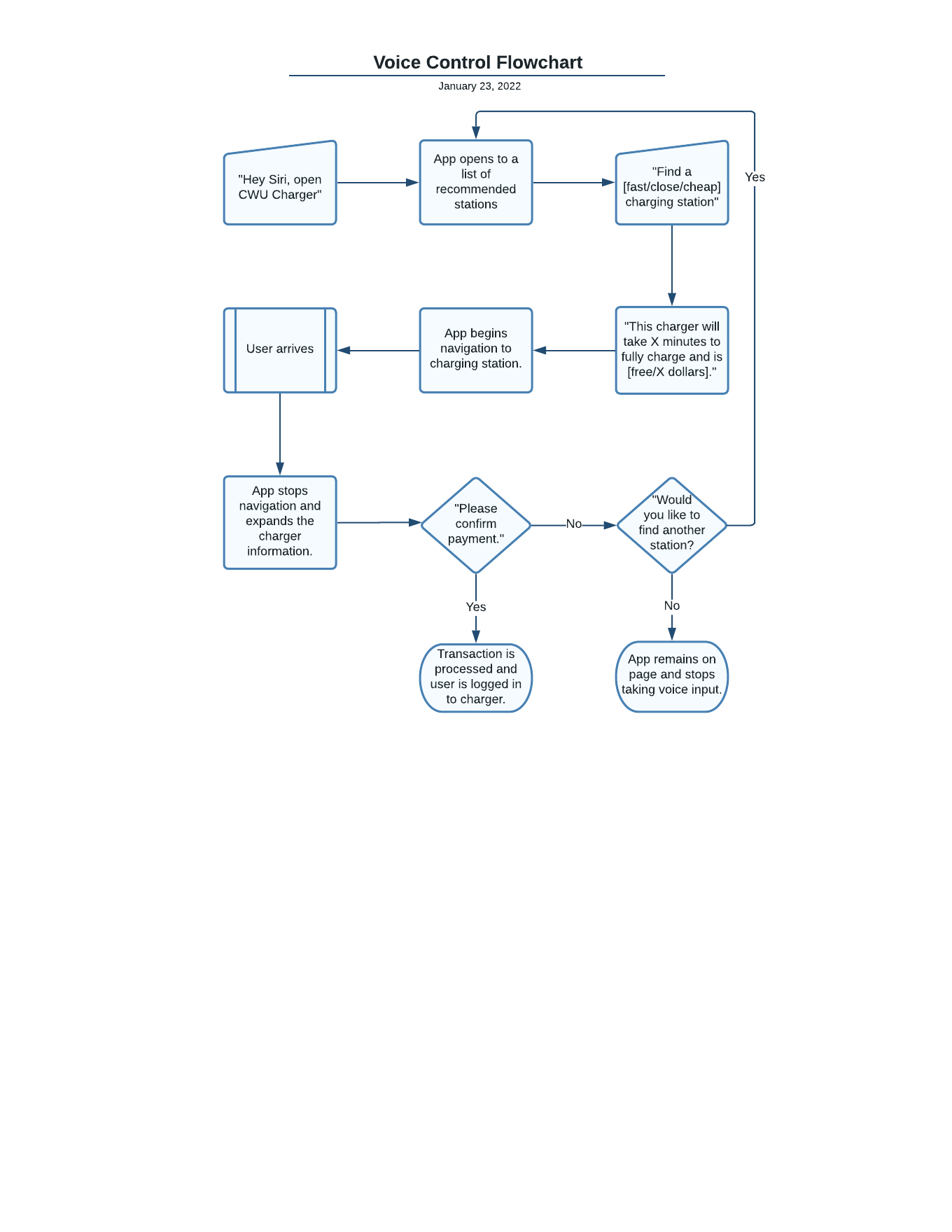
Description automatically generatedGraphical user interface, application

Description automatically generated

*Figure 3* below demonstrates a typical touchscreen navigation through the app. Upon opening the app, the user sees the map and may swipe up to view the list of stations. The user may sort and filter prior to in-app navigation to a charger. Upon arrival, the app should check if the user is signed into the station’s service. If not, it should prompt the user to confirm a payment and allow them to begin charging.





*Figure 4* below details the user voice control flow. If the app is opened via voice, it should be automatically waiting for voice input once opened. The user can ask the app to find the closest, fastest, or cheapest station nearby. Without one of these modifiers, the app should find the “best” station based on the user’s preferences. The app should then read out the station’s speed and price, and once confirmed with the user, should show navigation to the station. The app then authenticates the user against the service, or confirms payment for the service if the user is not subscribed.



## 

## 

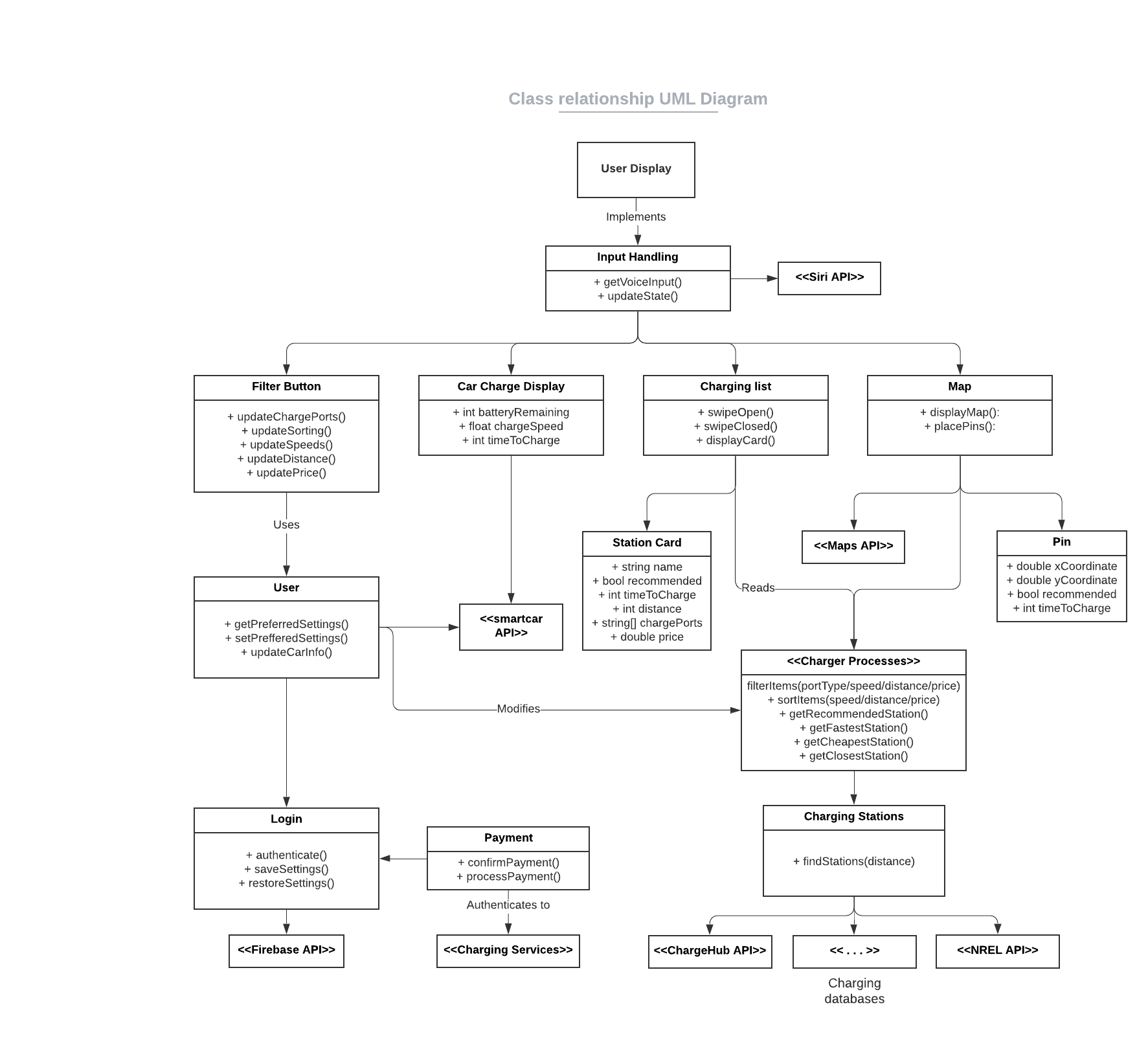
## 

## 

## 

## 5.2 Low-Level Design

This section contains a UML diagram of the back-end design, including classes and relationships, functions, variables, and API integrations. 

*Figure 5* above divides the classes into four components: the filter button and menu, the car charging display, the list of chargers, and the map. These are further sectioned into supporting APIs, helper components, and interfaces. The filter menu can modify user settings such as charge port types and sorting type, which is saved in the Firebase API. Both the charging list and map components use this information to update which stations are visible and which are recommended.

The SmartCarAPI is used by the car charge display and when calculating charge times to get charging information from the EV. The Charging Stations class combines information from multiple charging station databases, such as ChargeHub and NREL. Siri may be used to handle voice inputs, Google Maps may be used for the map display and navigation, and the Payment class will need to securely connect to any given charging service to authenticate or pay for the service.

# 6. Conclusion

This project should function as a proof of concept: show that a charging app that unifies the many different charging services can be done elegantly. Importantly, the app should determine an accurate charging time at any given charger. Because the team can use the many existing APIs and databases, most of the team’s time will likely go towards combining these backends with a robust, user-friendly UI. Care will also need to be taken to ensure that user information, including payment information, is secure.

# 7. Table of Figures

[Figure 1. Map and list pages.](about:blank) 7

[Figure 2. Signup and setup pages.](about:blank) 8

[Figure 3. Touch Input Flowchart.](about:blank) 9

[Figure 4. Voice Control Flowchart.](about:blank) 10

[Figure 5. Class Relationship UML Diagram.](about:blank) 11