Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2022

Clean Team
Garbage Truck Route

Mohammed Ahmed
Amay Kadakia
Shubhayu Shrestha
Nicholas Soliz

Revision History

Revision	Date	Author(s)	Description
0.1	13.06.2022	MA , AK , SS , NS	Document creation
0.2	14.08.2022	MA , AK , SS , NS	Final Submission

CONTENTS

1 Introduction **5**

2 System Overview **5**

3 Application Graphical User Interfaces **6**
- 3.1 Layer Hardware . . . 7
- 3.2 Layer Operating System . . . 7
- 3.3 Layer Software Dependencies . . . 7
- 3.4 Login . . . 7
- 3.5 Location and Route . . . 7
- 3.6 Registration . . . 7
- 3.7 Displaying Vehicle and User Information . . . 8

4 Firebase Authentication **10**
- 4.1 Layer Operating System . . . 10
- 4.2 Layer Software Dependencies . . . 10
- 4.3 Current User and Current Vehicle . . . 10
- 4.4 New User and Vehicle Validation . . . 10

5 Route Processing **12**
- 5.1 Layer Software Dependencies . . . 12
- 5.2 Special Pickup Instructions . . . 12
- 5.3 Defined Routes . . . 13

6 Data Storage **14**
- 6.1 Layer Operating System . . . 14
- 6.2 Layer Software Dependencies . . . 14
- 6.3 API Key table . . . 15
- 6.4 User Data table . . . 15
- 6.5 Vehicle Data tables . . . 15

7 Appendix A **16**

List of Figures

1 A simple architectural layer diagram 6
2 Example subsystem description diagram 8
3 Fireabase Subsystem 11
4 Route Processing Subsystem 12
5 Data Storage Subsystem 14

List of Tables

1 Introduction

Bin Buddy is a multi-platform application for homeowners and waste management companies that will be utilized to make the garbage experience far more admirable, including features such as garbage truck tracking and proper waste disposal methods. The application will help track a garbage truck and inform the user if the truck has passed their home or they still have time to move their trash bins to the curb for the garbage truck to pick up. Additionally, the app will also be able to send reminder notification on a day prior to the scheduled garbage collection day. This application will be used by waste management companies and homeowners that utilize their services.

2 System Overview

For Bin Buddy we have four sections of importance when considering our data flow. The first UI can be split into two, User UI and Vehicle UI. These systems are used to provide a user interface to the two user types expected to utilize the application. Giving the users a method to login/register, view truck location or route and display user information is key to our UI section. To verify the user and truck driver information is legitimate there is the Firebase Authentication layer. This connects to the data storage layer to validate information being input into the application. The data storage layer is the simplest portion that provides robust information about the users as well as providing Maps and geolocation information to the rest of the layers. Lastly there is the route processing layer that provides calculations and feedback to drivers and at home users about the location of trucks that are on their routes as well as specialty considerations for special pick-up days.

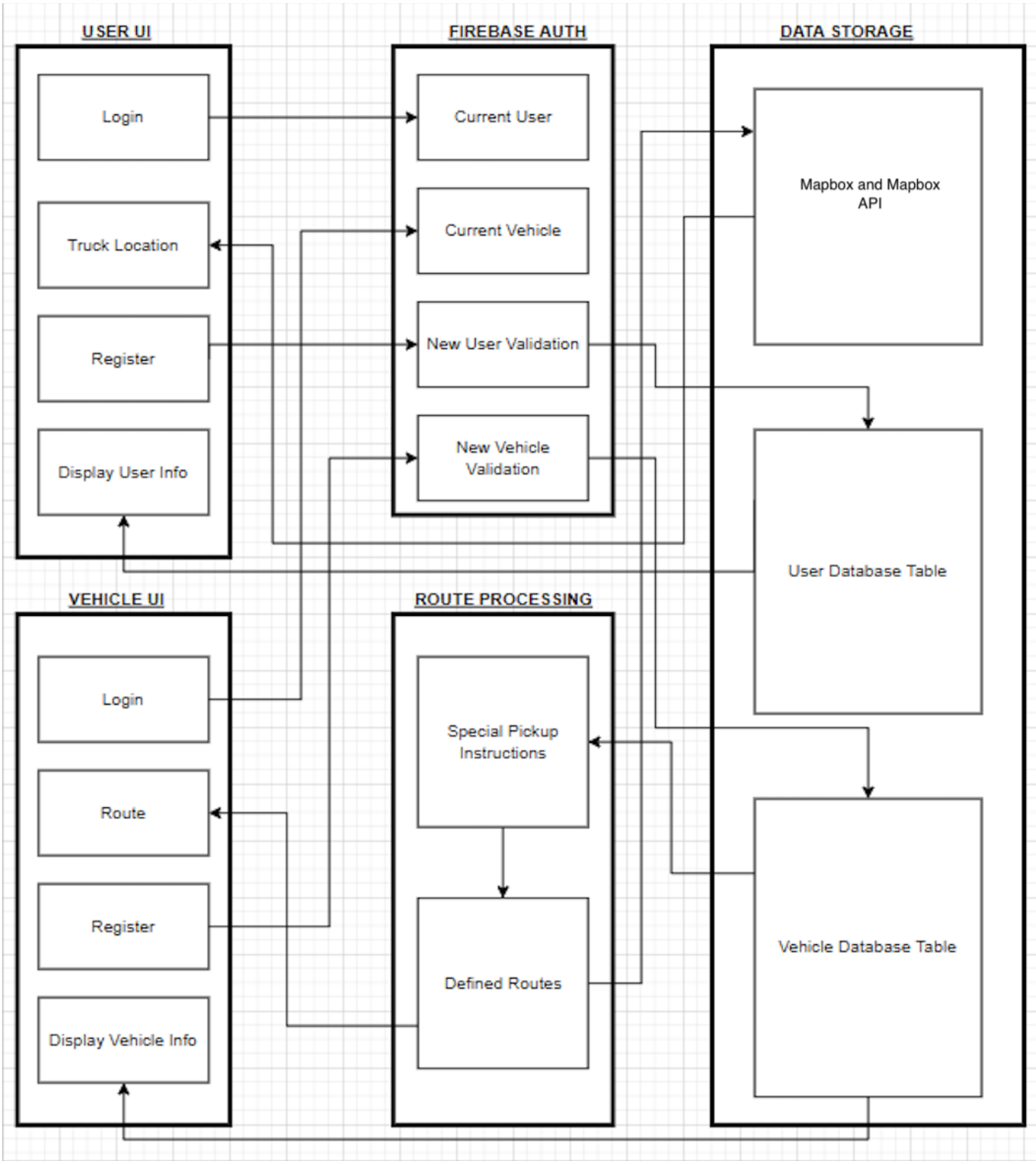

Figure 1: A simple architectural layer diagram

3 Application Graphical User Interfaces

3.1 Layer Hardware

The hardware that the user can utilize to access the software application is any computer, smartphone, or laptop.

3.2 Layer Operating System

This layer can run on Windows, MacOS, or Linux on a computer, or iOS and Android for mobile devices.

3.3 Layer Software Dependencies

The software dependencies required for the UI layer is Angular, specifically the Ionic Framework for the development of the application, as well as internet browsers such as Google Chrome, Safari, or Microsoft Edge in order to view the application.

3.4 Login

The purpose of this subsystem is to allow the user to log into the application. It is assumed that the user already has an account, and it will allow them to access the features of their account, as well as view their information. The login works for both customer and vehicle accounts.

3.4.1 Subsystem Software Dependencies

The login page is dependent on Firebase Authentication, specifically Google Sign-On, in order for the user to log into the account.

3.4.2 Subsystem Programming Languages

The subsystem is built using JavaScript and Angular.

3.4.3 Subsystem Data Structures

The login page will call the Firebase API in order to authenticate the user

3.5 Location and Route

The responsibility of this function is to display a map with the location of the garbage truck, as well as the Estimated Time of Arrival (ETA) of the truck.

3.5.1 Subsystem Hardware

A mobile device with GPS functionality will be required in order for the application to send out the real-time location of the garbage truck.

3.5.2 Subsystem Software Dependencies

Mapbox will be utilized in order to display the map and retrieve location data.

3.6 Registration

This subsystem allows both vehicles and users to register for an account. For the vehicle registration, the responsibility of this function is to capture the information needed to register a new vehicle into a fleet of vehicles in the system database. The system will ensure that the data is entered fully by the user before submission of the vehicle registration.

3.6.1 Subsystem Software Dependencies

The registration page will processes and store the data using Firebase.

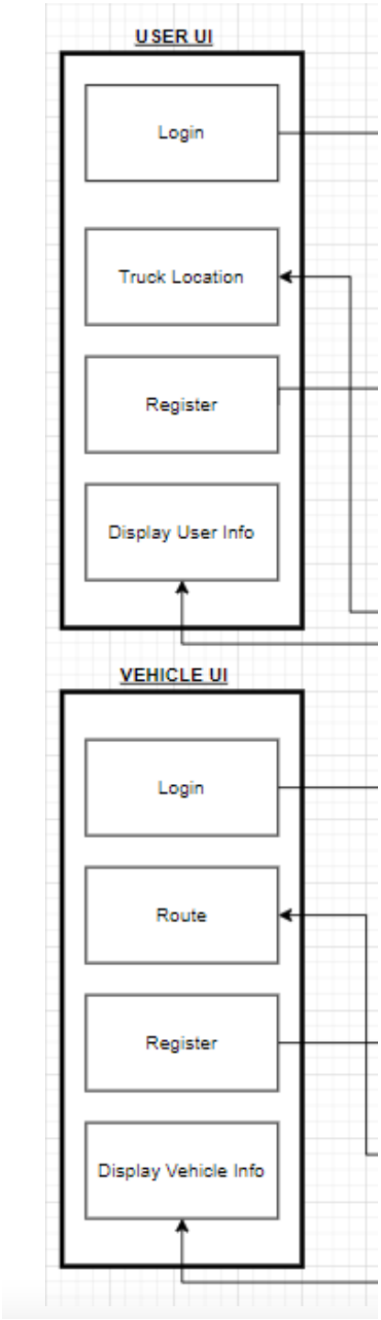

Figure 2: Example subsystem description diagram

3.7 Displaying Vehicle and User Information

The responsibility of this function is to display user information that is stored in data storage for the user or vehicle that is currently logged into the application. This functionality will not show any sensitive information of any user that is not the currently logged in user.

3.7.1 Subsystem Operating System

Windows, MacOS, or Linux on computers, as well as Android and iOS on mobile

3.7.2 Subsystem Software Dependencies

Firebase in order to retrieve vehicle and user information, as well as Angular and Ionic to display that information to the user.

3.7.3 Subsystem Programming Languages

Angular, TypeScript, JavaScript

4 Firebase Authentication

In this layer, this implements Google's Firebase Authentication to allow users to log in and create accounts for our application, implementing sign on methods such as Google Sign-On. This will be connected with the Bin Buddy application by implementing the Firebase SDK to our project.

4.1 Layer Operating System

This project can be accessed by any device that has an operating system and can access the web, which includes: Android, iOS, MacOS, Windows, Linux, and ChromeOS.

4.2 Layer Software Dependencies

This layer will utilize the Firebase SDK which will include the @angular/fire library. To run this application, a web browser is also needed such as Google Chrome, Safari, Firefox, and Opera Web Browser.

4.3 Current User and Current Vehicle

In these subsystems, we are checking if the user or vehicle already exists within the database using the Firebase Authentication back-end service.

4.3.1 Subsystem Operating System

No OS is needed, but as a developer, if it is needed to be managed, Firebase's website can be accessed and the project can be accessed.

4.3.2 Subsystem Software Dependencies

This requires the UI code in order to handle the input from the users in order to log in. This will also utilize the Firebase Authentication SDK which will include the @angular/fire library. To run this application, a web browser is also needed such as Google Chrome, Safari, Firefox, and Opera Web Browser.

4.3.3 Subsystem Programming Languages

This subsystem will be implemented utilizing ionic in a typescript file.

4.3.4 Subsystem Data Structures

This will be implemented in our application utilizing async calls, page components, and form controls.

4.3.5 Subsystem Data Processing

Firebase Authentication backend services will verify the credentials and return a response to the client.

4.4 New User and Vehicle Validation

In these subsystems, we are creating a new vehicle or user if the user or vehicle does not already exists within the database using the Firebase Authentication back-end service.

4.4.1 Subsystem Operating System

No OS is needed, but as a developer, if it is needed to be managed, Firebase's website can be accessed and the project can be accessed.

4.4.2 Subsystem Software Dependencies

This requires the UI code in order to handle the input from the users in order to log in. This will also utilize the Firebase Authentication SDK which will include the @angular/fire library. To run this application, a web browser is also needed such as Google Chrome, Safari, Firefox, and Opera Web Browser.

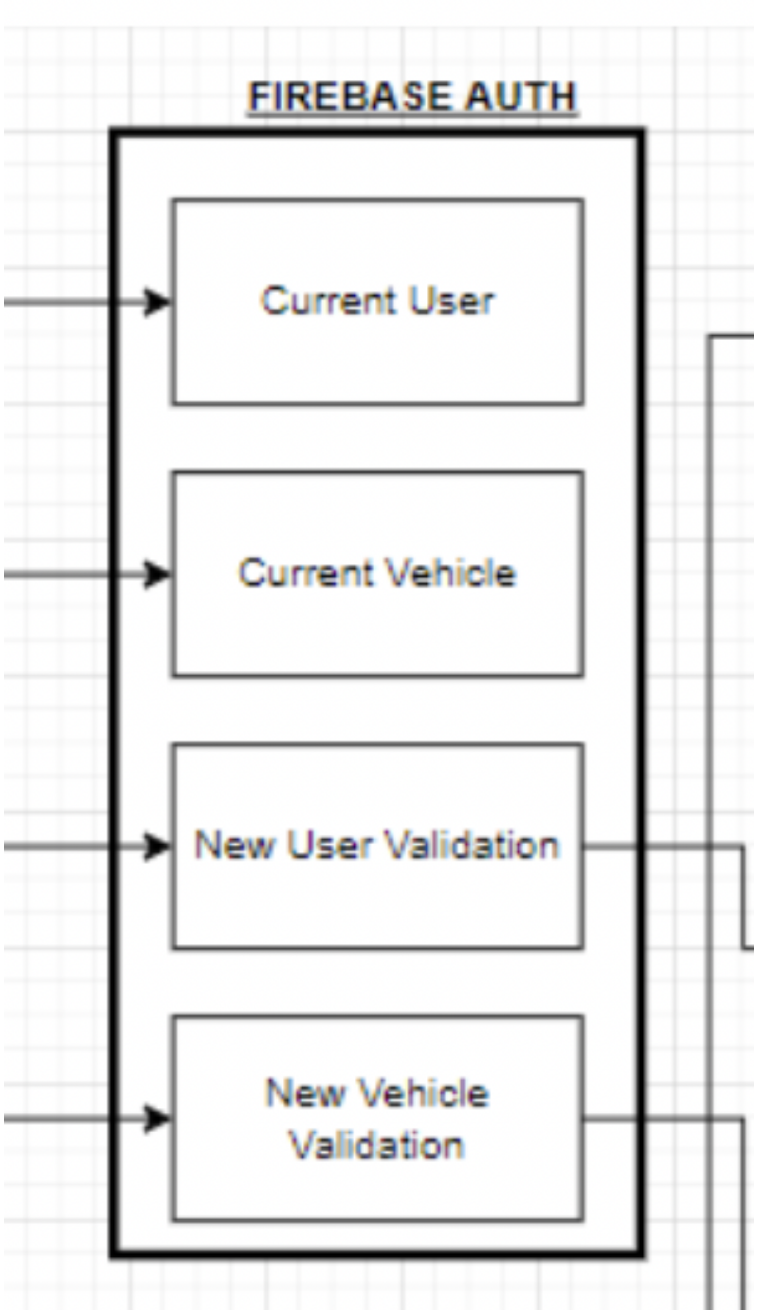

Figure 3: Fireabase Subsystem

4.4.3 Subsystem Programming Languages

This subsystem will be implemented utilizing ionic in a typescript file.

4.4.4 Subsystem Data Structures

This will be implemented in our application utilizing async calls, page components, and form controls.

4.4.5 Subsystem Data Processing

Firebase Authentication back-end services will verify the credentials and return a response to the client,

5 Route Processing

5.1 Layer Software Dependencies

There is a need to rely on Firebase for the collection and storage of data. Using the ability for Firebase to provide real-time data the processing of routes for special situations and days can be pulled in easily from a collection.

5.2 Special Pickup Instructions

This subsystem is a class that provides dynamic routing information to drivers and home users based on special conditions such as road closures, large item collection days and recycling to name a few. The class will update the route by providing flags and settings on the days specified by the regional directors of garbage disposal in a given city.

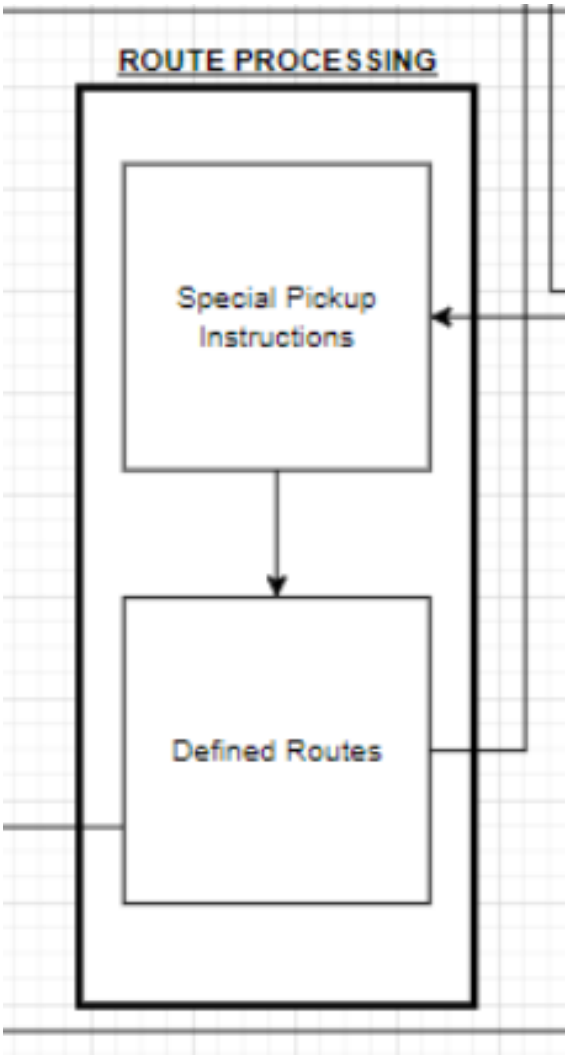

Figure 4: Route Processing Subsystem

5.2.1 Subsystem Software Dependencies

There is a need to rely on Firebase for the collection and storage of data. Using the ability for Firebase to provide real-time data the processing of routes for special situations and days can be pulled in easily

from a collection.

5.2.2 Subsystem Programming Languages

Angular, Typescript, Ionic Framework, JavaScript

5.2.3 Subsystem Data Structures

Flags will be provided to the class that will pull the needed data from Firebase collections.

5.3 Defined Routes

This subsystem is a class that provides dynamic routing information to drivers and home users through integration with the special pick-up instructions subsection. Based on the information received from the special instructions, the routes will be fetched with flags set from the Firebase Collection of routes available to use.

5.3.1 Subsystem Software Dependencies

There is a need to rely on Firebase for the collection and storage of data. Using the ability for Firebase to provide real-time data the processing of routes for special situations and days can be pulled in easily from a collection.

5.3.2 Subsystem Programming Languages

Angular, Typescript, Ionic Framework, JavaScript

5.3.3 Subsystem Data Structures

Flags will be received from special instructions and map data will be sent to the UI component to be drawn on them.

6 Data Storage

This is the database layer of the application, where all data will be stored. It consists of 3 main tables: API Key Table, User Information Table and Vehicle Information Table.

6.1 Layer Operating System

The data will be stored on Firebase and will use all Firebase system dependencies.

6.2 Layer Software Dependencies

The data base will use Firebase Software dependency.

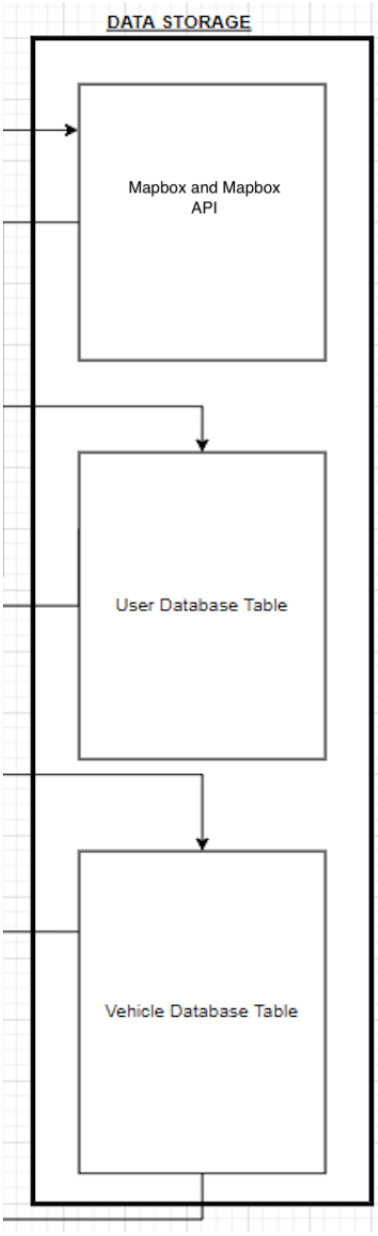

Figure 5: Data Storage Subsystem

6.3 API Key table

This subsystem will store all of the API keys for the application such as the Mapbox API and Geolocation API. These keys will provided to the code when called in order to limit accidental public exposure or leaks of API keys.

6.3.1 Subsystem Operating System

The database will store API key from Mapbox will use it's dependencies.

6.3.2 Subsystem Programming Languages

The API key will be integrated into the application that is built using Angular with an Ionic framework.

6.3.3 Subsystem Data Structures

The API will be used to track the location of the garbage truck and therefore it will use data from the drivers device.

6.3.4 Subsystem Data Processing

This data table will be used to process location data.

6.4 User Data table

This subsystem will store all of the user login credentials and account information.

6.4.1 Subsystem Hardware

This subsystem will obtain user data based on users input on their devices.

6.4.2 Subsystem Software Dependencies

This data table will require the input that the users make using the BinBuddy application.

6.4.3 Subsystem Programming Languages

BinBuddy will be built using Angular with ionic therefore this data will use the same.

6.4.4 Subsystem Data Processing

This data table will process the time for a garbage truck to arrive to the users location using the location gathered from truck driver.

6.5 Vehicle Data tables

This subsystem will store all of the vehicle account information and details about the vehicle.

6.5.1 Subsystem Software Dependencies

This data table will require the input that the users make using the BinBuddy application.

6.5.2 Subsystem Programming Languages

BinBuddy will be built using Angular with ionic therefore this data will use the same.

7 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.