

**CptS323**

# **Software Design Document**

# **Software Design Document for Home Visits Vaccination**

Team: Kovid Killers

Project: Home Visits Vaccination

Team Members:

*Ryan Blanchard*

*Jason Toth*

*Jeremiah Strzelczyk*

**Last Updated: 4:20 P.M. April 5, 2021**

# Table of Contents

<b>Table of Contents</b>	<b>3</b>
<b>Document Revision History</b>	<b>5</b>
<b>List of Figures</b>	<b>6</b>
<b>List of Tables</b>	<b>7</b>
1. INTRODUCTION	8
1.1 Purpose	8
1.2 Scope	8
1.3 Overview	8
1.4 Definitions and Acronyms	8
2. SYSTEM OVERVIEW	10
2.1 Overview	10
2.2 Functional Requirements	10
2.3 Nonfunctional Requirements	12
3. SYSTEM ARCHITECTURE	14
3.1 Architectural Design	14
3.2 Decomposition Description	16
3.3 Design Rationale	17
4. DATA DESIGN	19
4.1 Data Description	19
4.2 Data Dictionary	19
5. COMPONENT DESIGN	21
6. HUMAN INTERFACE DESIGN	23
6.1 Overview of User Interface	23
6.2 Screen Images	23
6.3 Screen Objects and Actions	27
7. REQUIREMENTS MATRIX	30
8. APPENDICES	32



## Document Revision History

Revision Number	Revision Date	Description	Rationale
1.0	4-7-2021	First Draft	First Draft

## List of Figures

Figure #	Title	Page #
3-1	Administration	14
3-2	Providers	14-15
3-3	Vans	15
3-4	Schedule Vaccination	15-16
3-5	Display Interface	16
3-6	Top Level Data Flow Diagram	17
3-7	Subsystem Decomposition	17
6-1	Administration Login	23
6-2	Main Interface (Map)	24
6-3	Main Interface (Stats)	25
6-4	New Provider Interface	25
6-5	New Van Interface	26
6-6	Remove Provider Interface	26
6-7	Remove Van Interface	26
6-8	Provider Search Interface	27
6-9	Van Search Interface	27

## List of Tables

Figure #	Title	Page #
1	Document Revision History	5
2	Requirements Matrix	30

# 1. INTRODUCTION

## 1.1 Purpose

This Software Design Document describes the architecture and system design of the Home Visits Max Rate Vaccination System. It is intended to be used by developers to aid in implementation.

## 1.2 Scope

The intended use of this software is to find individuals who are in need of an in-home COVID-19 vaccination. The software notifies the closest provider and displays everything on a graphical user interface. The goal and objective is to maximize the number of vaccinations delivered. This software will benefit those who need vaccination but are unable to receive it in-person.

## 1.3 Overview

This software design document provides the following sections: the system overview provides the functionality, design, and background of HVMRS (section 2.0), the system architecture will explain the relationships between the required modules at a high level (section 3.0), the data design will explain what data structures are used and how major data is stored within (section 4.0), the component design will explain the components in a systematic way (section 5.0), the human interface design will describe the system from the user's perspective (section 6.0), the requirements matrix will examine cross references between our data structure implementation and how they relate to the requirements from our SRS document (section 7.0), and finally an appendix (section 8.0).

## 1.4 Definitions and Acronyms

HVMRS: Home Visits Max Rate System. The software described.

SDD: Software Development Document. This document.

Firebase: Online database.

GMaps: Google Maps Application Programming Interface.



PID: Provider Identification

VID: Van Identification

GUI: Graphical User Interface

UI: User Interface

## 2. SYSTEM OVERVIEW

[Give a general description of the functionality, context and design of your project. Provide any background information if necessary]

### 2.1 Overview

This section will discuss the functional and nonfunctional requirements that will be fulfilled. The functional requirements include a login UI for admin, a UI intended to visualize the real-time delivery of vaccinations, and a UI for adding, removing, or editing providers and vans. The nonfunctional requirements include: the usability, reliability, performance, supportability, implementation, interface, operational, packaging, and legal requirements of the software.

### 2.2 Functional Requirements

#### Login Page

- **Description:** A user interface for admin to enter a name and password to enter restricted access.
- **Source:** This requirement was requested by Dr. DeLaTorre.
- **Constraints:** None
- **Standards:** None
- **Priority:** Medium

#### Add/Remove Provider/Van

- **Description:** A UI for administration to add or remove a van or provider. Administration will enter pertinent information for either and submit add van or add provider.
- **Source:** This requirement was requested by the client.
- **Constraints:** None
- **Standards:** Will be accessed from the Update Provider/Van page.
- **Priority:** Medium

#### Delivery Display

- **Description:** A UI to display the routes and locations for each Provider/Van. The display will show real-time location and all data required by the client, such as the number of successful vaccinations, the number of failed vaccinations, the profit of each Provider/Van, and the total profit.

- **Source:** This requirement was requested by the client.
- **Constraints:** One constraint is the use of the internet. Because the information will be displayed in real-time, internet access will be required. In addition, data will be saved to- and read from an online data storage center.
- **Standards:** The page will display all pertinent information, it will display all the current Providers and their respective Vans, and there will be an option to update Provider/Van.
- **Priority:** High

#### Find Optimal Prospect

- **Description:** The system must be able to locate the optimal prospect for each van to maximize the number of vaccinations that can be achieved for each van.
- **Source:** This requirement was requested by Dr. DeLaTorre
- **Constraints:** Time remaining for each vial and competition from other companies which means the algorithm must decide quickly.
- **Standards:** The algorithm should consider time to each prospect and time before the vials expire. The shortest time should be used for each van.
- **Priority:** High

#### Get and Send Prospect Info

- **Description:** The system must be able to Communicate with Dr. DeLaTorre's firebase to receive the list of prospects and to notify when each prospect is selected or vaccinated.
- **Source:** This requirement was requested by Dr. DeLaTorre
- **Constraints:** Company must be registered correctly and have the correct information to send to the firebase to receive a list back. Company must also ensure that each vaccination is reported.
- **Standards:** The system will access the firebase and request the prospect list frequently as it continues to vaccinate prospects and move onto the next prospect.
- **Priority:** High

#### Map Display

- **Description:** Real-time map displaying all Providers with their respective Vans and their real-time current positions and their current route.
- **Constraints:** The map will require an active internet connection to retrieve real-time data.
- **Standards:** The map should display a portion of the map to maximize the number of current routes for the user. The map should allow for locking on specific vans/providers.
- **Priority:** High

#### Cloud Data Storage

- **Description:** The system should store and retrieve all pertinent data from the Firebase cloud.
- **Constraints:** Cloud data storage will require an active internet connection.
- **Standards:** None
- **Priority:** High

## 2.3 Nonfunctional Requirements

#### Usability

- A member of administration should be able to log in to the system within their first day of using the system.
- A member of administration should be able to add, remove, or edit a Provider or Van on their first day of use.
- A client should be able to differentiate the vans from their destinations.

#### Performance

- The login algorithm should take less than five seconds.
- The position of the cars should update once per second.
- The algorithm to calculate vaccinations should take one second or less.

#### Supportability

- The software should run on Windows 10

#### Implementation

- The software should be implemented in C#
- The software should be able to work with Firebase

#### Interface

- The software should include a login interface
- The software should include an update provider/van interface
- The software should include a map interface

#### Operational

- The software should not depend on any third-party software
- The software should be operational by one individual

#### Packaging

- The client should be able to install the software independently
- The client should only need to install one piece of software

### Legal Requirements

- The software should not use any technologies without explicit consent from its owners
- The development of the software should not violate any public or federal laws

### 3. SYSTEM ARCHITECTURE

#### 3.1 Architectural Design

The software is built upon the following modular components: administration, van fleet, vaccination scheduling, Gmaps, and Firebase cloud services.

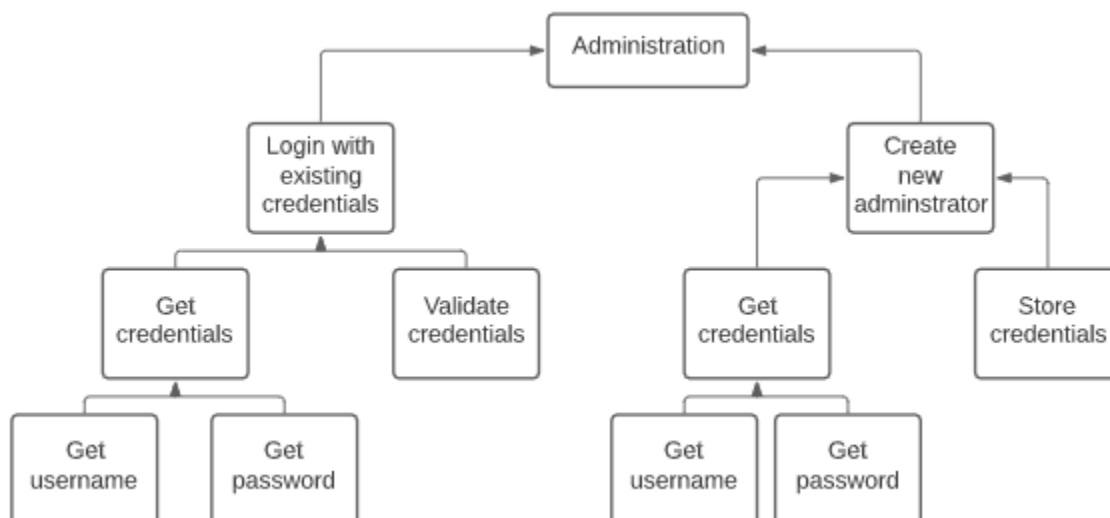


Figure 3-1: Administration

The software will allow for administration to assume control of the system. A member will be required to either create a new administration profile or log into an existing administration profile. The required fields for each are a username and a password. These credentials will be stored via the Firebase cloud.

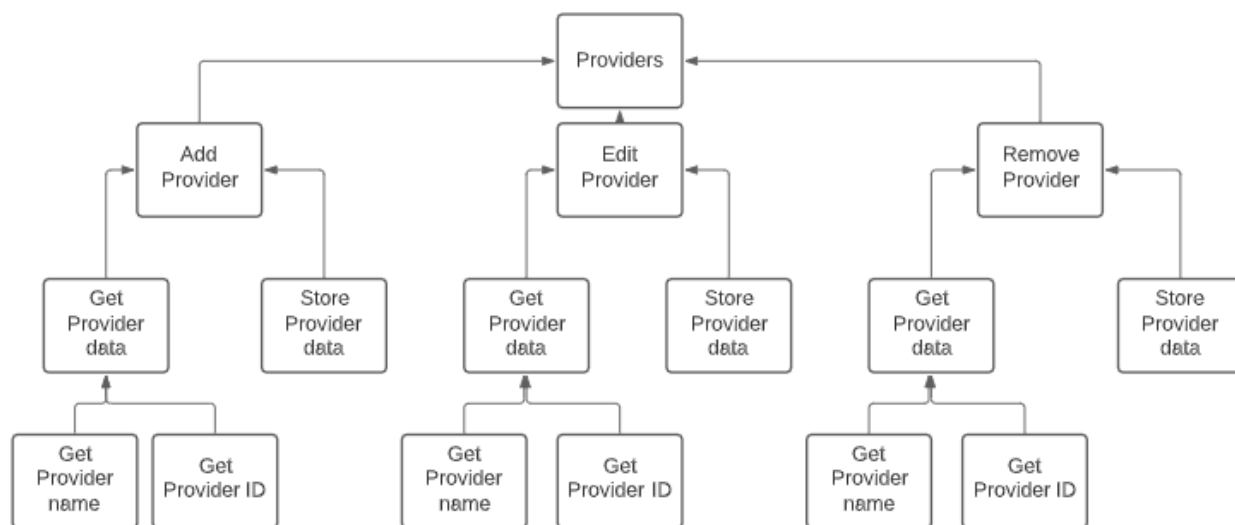


Figure 3-2: Providers

The software allows for adding, removing, or editing existing Providers. A Provider name and ID is required to add a Provider, while either is required to edit or remove a Provider.

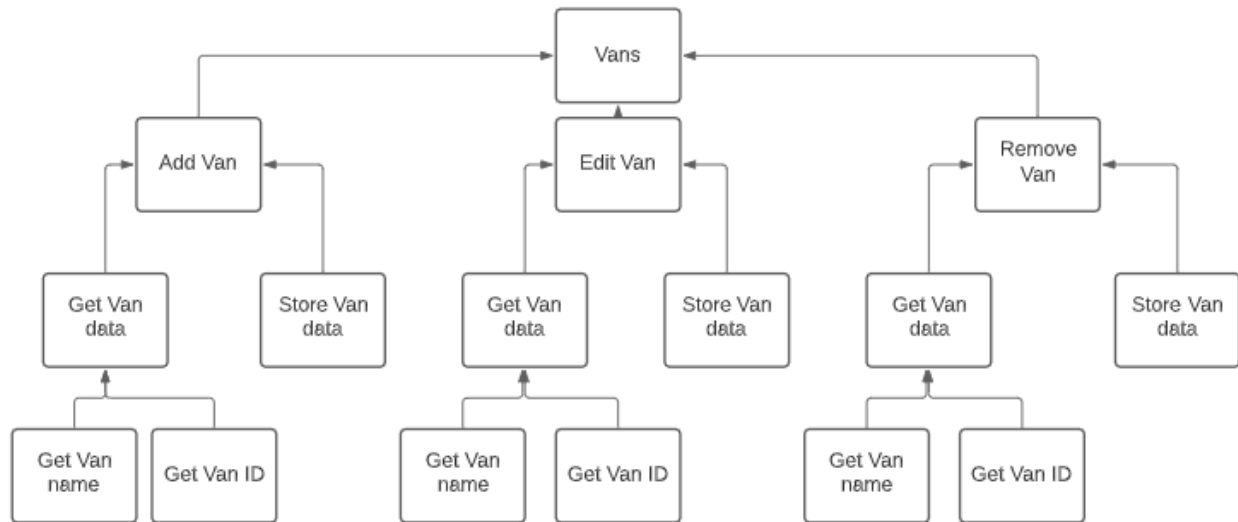


Figure 3-3: Vans

The software allows for adding, editing, or removing existing Vans. A Van name and ID is required to add, while either is required to edit or remove an existing Van.

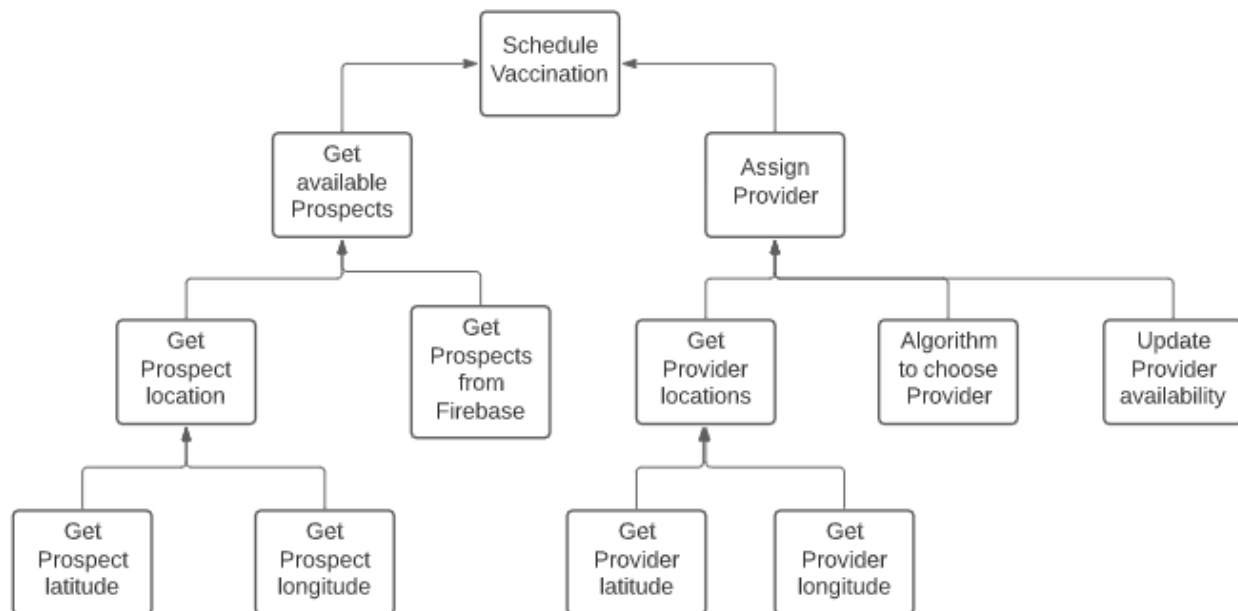


Figure 3-4: Schedule Vaccination

Prospect availability will be determined by Dr. DeLaTorre's algorithm. The Provider chosen to administer vaccination will be determined by an algorithm. The locations and availability of each must remain up-to-date.

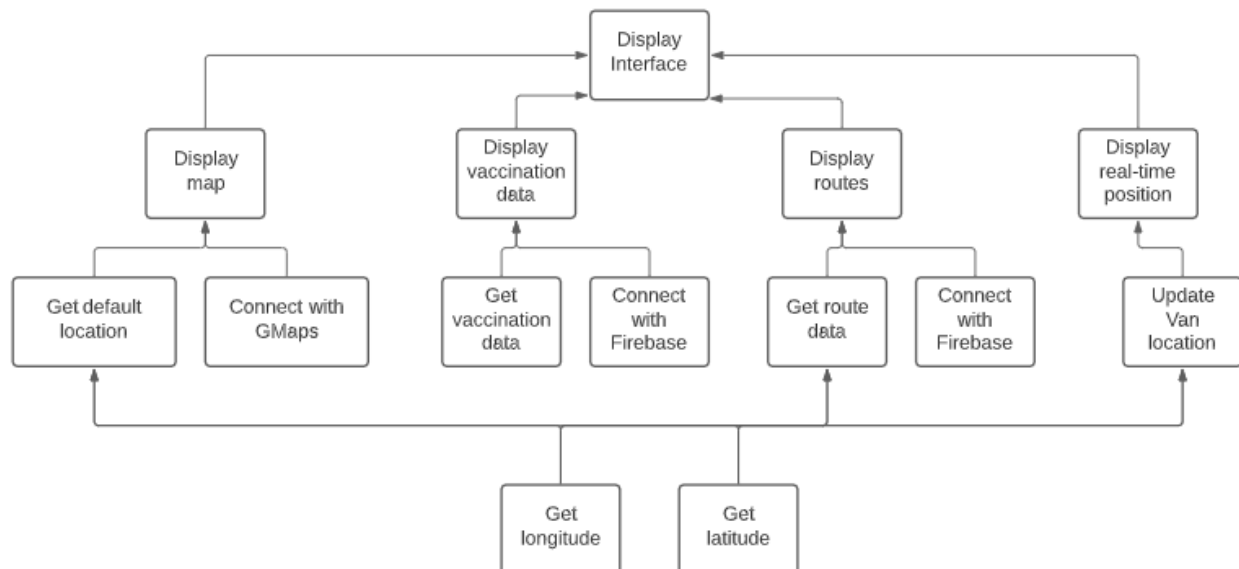


Figure 3-5: Display Interface

The Display Interface will visualize all pertinent data for the vaccination. The display interface will need a default location to reduce load time. The display interface will also be responsible for displaying the routes from Provider to Prospect.

## 3.2 Decomposition Description



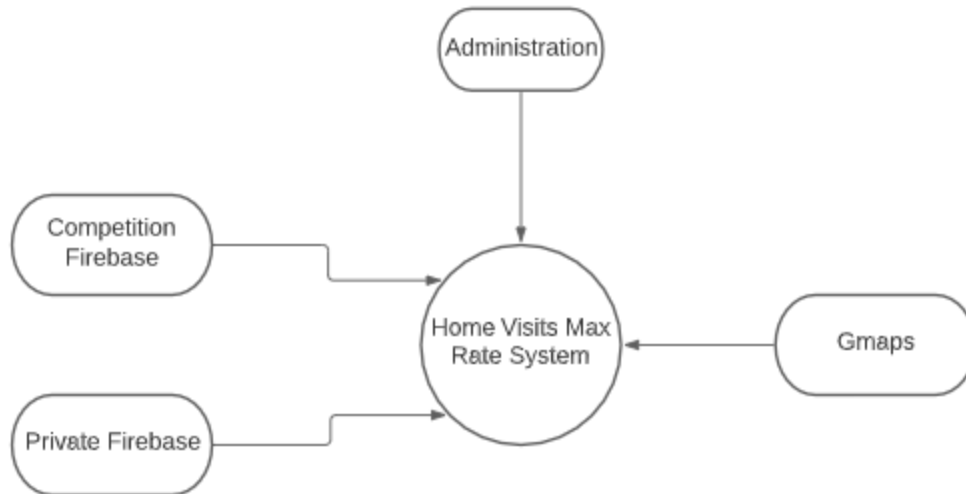


Figure 3-6: Top level Data Flow Diagram

The software retrieves information about Prospects from the competition firebase. From the private Firebase, we can send and receive data on providers, vans, prospects, and gmaps; this allows us to maintain up-to-date information on each. The software retrieves prospect location data from GMaps. Administration can manage provider and van data.

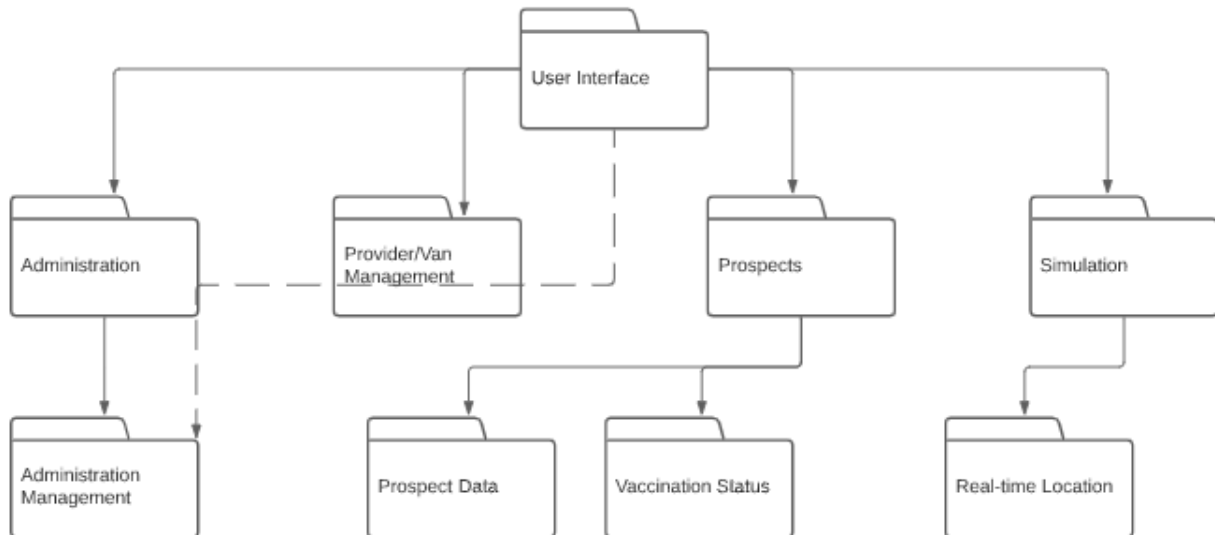


Figure 3-7: Subsystem Decomposition

### 3.3 Design Rationale

The architecture was chosen to decouple the major components of the system by means of separating the major functions into individual modules. Firebase was chosen

for cost purposes. We could have chosen to omit Firebase, but since the competition would come from Firebase, it made sense to incorporate it into our architecture.

## 4. DATA DESIGN

### 4.1 Data Description

All information will reside on Firebase servers in JSON format. The data stored will be information on provider, van, prospect, and administration. The software will pull this information and stored locally to be used during program execution. Arrays will be used for fast access and because there will be a limited number of vans, prospects, and providers. As changes are made to provider, van, prospect, or administration objects, Firebase will be updated immediately to reflect the changes.

### 4.2 Data Dictionary

[Text]

Competition Firebase

#### 4.2.1 Competition Firebase

Methods:

- updatePosition()
- getProspects()
- prospectVaccinated()

#### 4.2.2 Firebase

Attributes:

- "admins": {...}
- "prospects": {...}
- "providers": {...}

Methods:

- validateAdmin()

#### 4.2.3 Prospect

Attributes:

- string pid
- double longitude
- double latitude
- bool inRoute
- int status

Methods:

- bool acceptVaccination()

- void updateStatus()

#### **4.2.4 Provider**

Attributes:

- string firstName
- string lastName
- string pid
- int vaccinationsDone
- bool inRoute

Methods:

- bool designateProspect()
- void waitPeriod()

#### **4.2.5 Van**

Attributes:

- string licensePlate
- int vid
- bool inUse
- int associatedProvider

## 5. COMPONENT DESIGN

### **Admin Management**

Create administrator takes a login and password as input

Create administrator

Save to Firebase

Return success

Administrator logon takes a login and password as input

Validate login

Validate password

Return success/failure

### **Provider/Van Management**

Create Provider takes a first and last name as input

Creates Provider

Assigns a unique provider ID

Save data to Firebase

Return provider ID

Create Van takes a plate number as inputs

Creates a van

Assigns a unique van ID

Saves data to Firebase

Return van ID

Update Provider takes a provider ID, or a provider first and last name as input

A struct is filled with Provider data.

User can change name

Data is updated to Firebase

Returns success/failure

Update Van takes a van ID as input

A struct is filled with Van data

User can change the plate

Data is updated to Firebase

Returns success/failure

Remove Provider takes a provider ID as input

User is prompted for admin password  
Password is validated  
User is prompted to finalize removal  
If yes then data is removed from Firebase

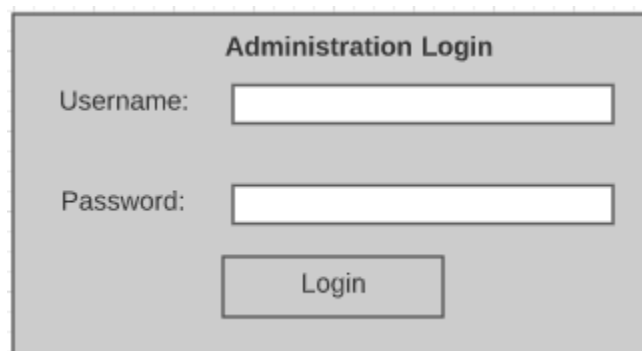
Remove Van takes a van ID as input  
User is prompted for admin password  
Password is validated  
User is prompted to finalize removal  
If yes then data is removed from Firebase

## 6. HUMAN INTERFACE DESIGN

### 6.1 Overview of User Interface

The software will display a login screen for administration. The user will be prompted to enter an administration login and an associated password. If the credentials are not validated then an error message will display and the user may try again. If the credentials are validated then the software will display the Map interface. The map interface will display the saved default location

### 6.2 Screen Images



The image shows a login form titled "Administration Login". It contains two input fields: "Username:" and "Password:". Below the password field is a "Login" button. The form is enclosed in a gray border.

Figure 6-1: Administration Login

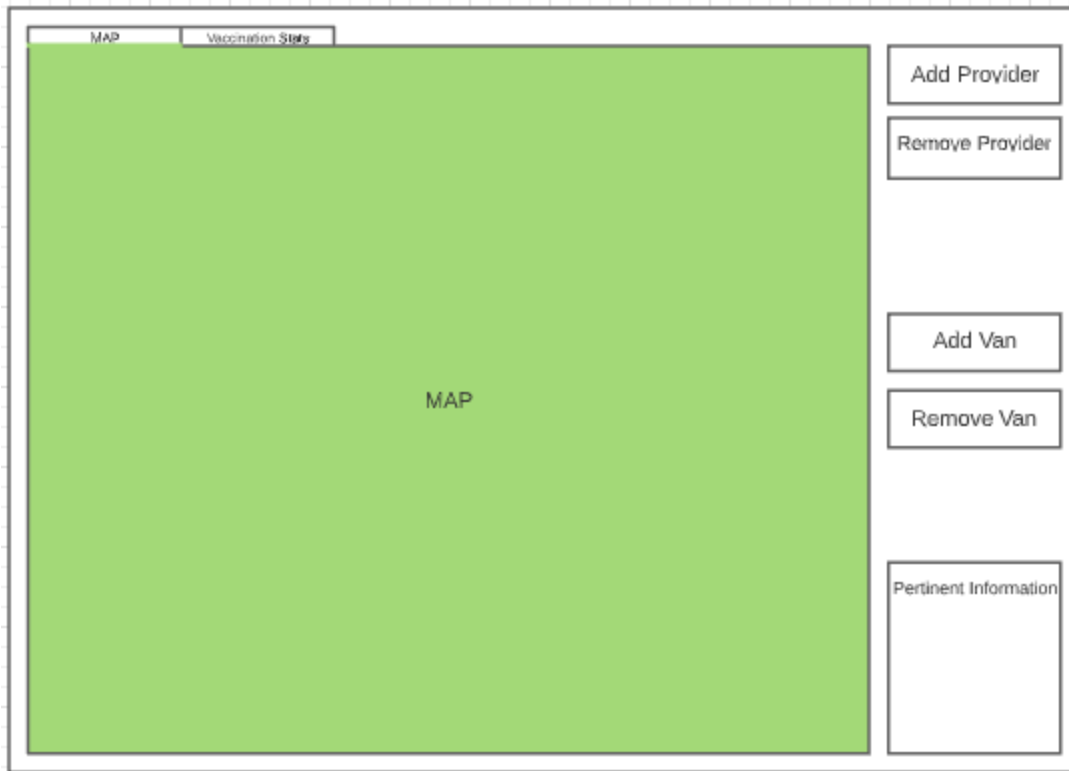
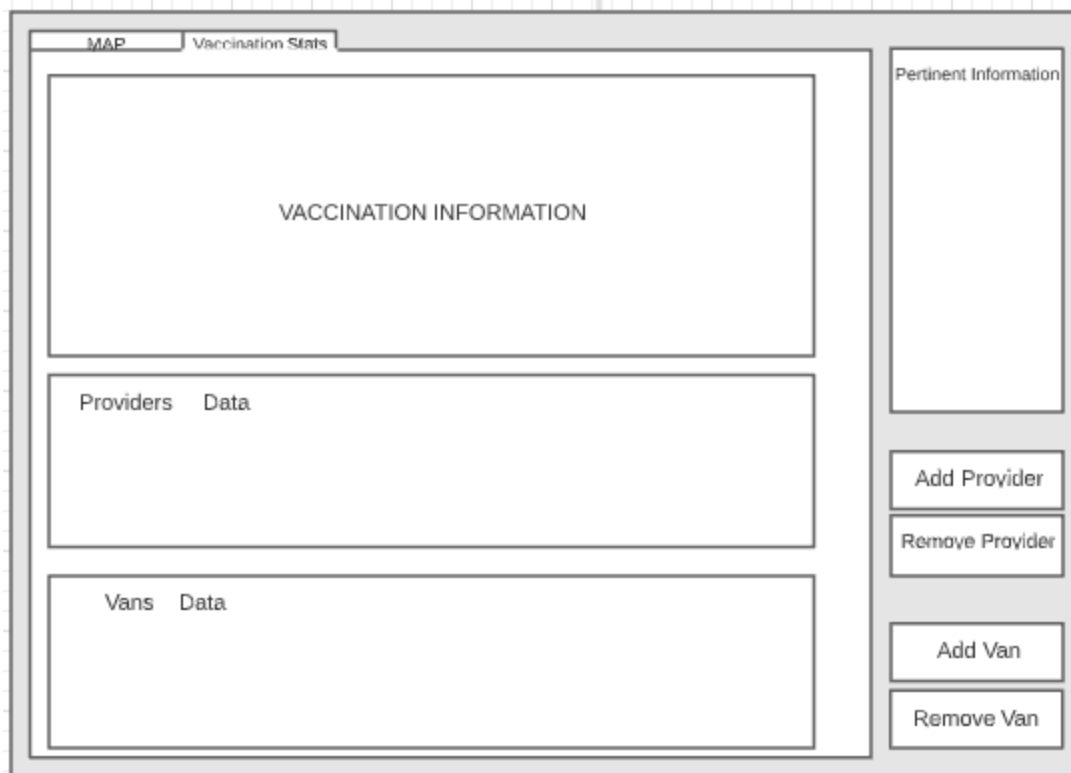


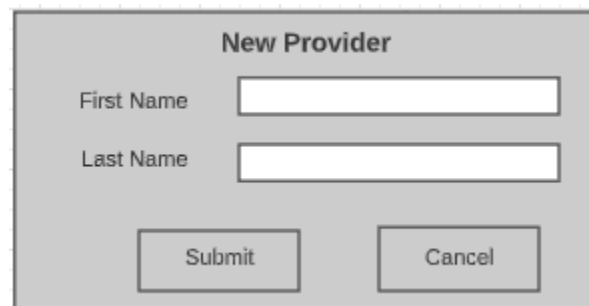
Figure 6-2: Main Interface (Map)





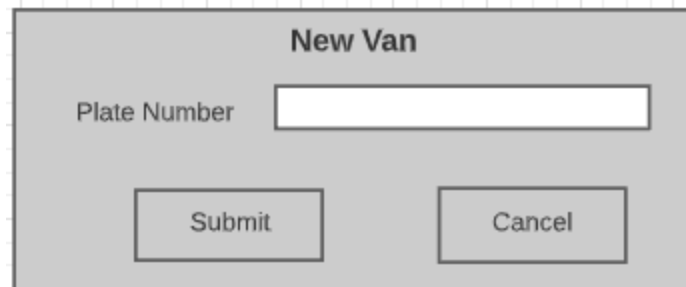
The 'Main Interface (Stats)' window features a title bar with two tabs: 'MAP' and 'Vaccination Stats', with 'Vaccination Stats' being the active tab. The main content area is divided into three horizontal sections. The top section is a large rectangle labeled 'VACCINATION INFORMATION'. The middle section is labeled 'Providers Data' and the bottom section is labeled 'Vans Data'. To the right of these sections is a vertical sidebar. At the top of the sidebar is a box labeled 'Pertinent Information'. Below this are four buttons stacked vertically: 'Add Provider', 'Remove Provider', 'Add Van', and 'Remove Van'.

Figure 6-3: Main Interface (Stats)



The 'New Provider' dialog box has a title bar with the text 'New Provider'. Inside, there are two text input fields. The first is labeled 'First Name' and the second is labeled 'Last Name'. At the bottom of the dialog are two buttons: 'Submit' and 'Cancel'.

Figure 6-4: New Provider Interface



The 'New Van' interface is a gray rectangular box with a title bar at the top. The title 'New Van' is centered in bold black text. Below the title, on the left, is the label 'Plate Number' in black text. To the right of this label is a white rectangular input field. At the bottom of the box, there are two buttons: 'Submit' on the left and 'Cancel' on the right, both with black text and gray borders.

Figure 6-5: New Van Interface



The 'Remove Provider' interface is a gray rectangular box with a title bar at the top. The title 'Remove Provider' is centered in bold black text. Below the title, there are three rows of text. The first row shows 'PID' followed by '#9123'. The second row shows 'First Name' followed by 'NAME OF PROVIDER'. The third row shows 'Last Name' followed by 'LAST NAME OF PROVIDER'. At the bottom of the box, there are two buttons: 'Confirm' on the left and 'Cancel' on the right, both with black text and gray borders.

Figure 6-6: Remove Provider Interface



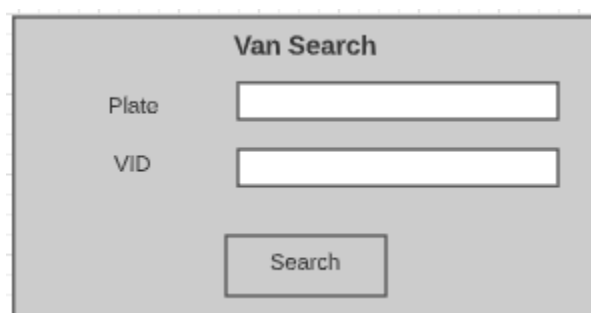
The 'Remove Van' interface is a gray rectangular box with a title bar at the top. The title 'Remove Van' is centered in bold black text. Below the title, there are two rows of text. The first row shows 'VID:' followed by '#0123'. The second row shows 'Plate #' followed by 'PLA-TE'. At the bottom of the box, there are two buttons: 'Confirm' on the left and 'Cancel' on the right, both with black text and gray borders.

Figure 6-7: Remove Van Interface



The Provider Search interface is a rectangular form with a light gray background. At the top center, the title "Provider Search" is displayed in bold. Below the title, there are two input fields. The first field is labeled "Name" and the second is labeled "PID". Both labels are positioned to the left of their respective input boxes. Below these two fields, there is a single button labeled "Search" centered horizontally.

Figure 6-8: Provider Search Interface



The Van Search interface is a rectangular form with a light gray background. At the top center, the title "Van Search" is displayed in bold. Below the title, there are two input fields. The first field is labeled "Plate" and the second is labeled "VID". Both labels are positioned to the left of their respective input boxes. Below these two fields, there is a single button labeled "Search" centered horizontally.

Figure 6-9: Van Search Interface

## 6.3 Screen Objects and Actions

### 6.3.1 Login Interface

The login interface will accept a login and password from the user. If the credentials do not match what is in the system then the form will reset itself and allow the user to try again. Upon validation, the user will be brought to the main interface.

### 6.3.2 Main Interface (Map)

The map interface will display pertinent information about the providers such as: which are available to deliver vaccination, which are in route to vaccinate and the proposed route they will take, potential prospects, and real-time location of vans. The map interface will be powered using GMaps.

### **6.3.3 Main Interface (Stats)**

The Stats page of the main interface will show the individual providers along with the individual vans. It will contain all relevant information for each, including: full name, PID, VID, and license plate. It will also show how many vaccinations each have administered, and how many failed post tests happened.

From both states of the main interface, the user has the option to do any of the following: add a provider, add a van, remove a provider, remove a van.

### **6.3.4 Add Provider Interface**

The add provider interface will allow admin to enter a first and last name for an aspiring provider. The user has the option to cancel the transaction or submit the information. If one of the fields were left blank, then they must try again.

### **6.3.5 Add Van Interface**

The add van interface will allow admin to add a van to the available list. The user must enter a valid plate number, then he or she will have the option to cancel or submit the new van.

### **6.3.6 Remove Provider Interface**

The remove provider interface will allow admin to remove a provider from the available list. The user will be given an option to confirm or cancel the removal. Prior to this interface, the user must first search for a provider to remove.

### **6.3.7 Remove Van Interface**

The remove van interface will allow admin to remove a van from the available list. The user will be given an option to confirm or cancel the removal. Prior to this interface, the user must first search for a van to remove.

### **6.3.8 Search for Provider Interface**

The search for provider interface allows the user to search for a provider given a provider ID or a name. If the search results in success then the user is directed to the appropriate interface.

#### **6.3.9 Search for Van Interface**

The search for van interface allows the user to search for a van given a van ID or a plate. If the search results in success then the user is directed to the appropriate interface.

## 7. REQUIREMENTS MATRIX

Component Or <b>Data Structure</b>	<b>Requirement Satisfied</b>
<b>FireBase Database</b>	Cloud Data Storage
Admins	Login and Add/Remove Vans
Prospects*	Get and Send Prospect Info
Provider*	Add/Remove Provider
<b>Prospect</b>	Finding/Vaccinating Prospect
Pid (Patient)	Finding/Vaccinating Prospect
Longitude and Latitude	Getting Route
inRoute	Finding/Vaccinating Prospect
Status	Finding/Vaccinating Prospect
<b>Provider</b>	Finding/Vaccinating Prospect
First Name/ Last Name	Finding/Vaccinating Prospect
Pid (Provider)	Finding/Vaccinating Prospect
vaccinationsDone	Vaccinating Prospect And Delivery Display
inRoute	Finding Prospect and Delivery Display
<b>Van</b>	Add/Remove Provider
LicensePlate	Add/Remove Provider
Vid	Add/Remove Provider
inUse	Add/Remove Provider and Vaccinating Prospect
associatedProvider	Add/Remove Provider

## 8. APPENDICES

Requirements Analysis Document