Virginia Tech

Blacksburg, VA 24061

CS 4624 – Multimedia, Hypertext, and Information Access

Spring 2020

Final Report

Mobile Application for Personal Bike Trip Data Collection

Devin Drake, Alex Frigault, Sam Larsen, Lucas Soohoo

Clients: Prof. Dr. Hesham Rakha, Dr. Mohamed Magdy

Professor: Edward A. Fox

May 13, 2020

1 Table of Contents

2 Table of Figures

3 Executive Summary/Abstract

The main goal of the project, VTTI Bike Mobile App (VBMA), is to create a mobile app for bike data collection for the Virginia Tech Transportation Institute. The mobile app will be cross-platform to operate on both iOS and Android devices. To meet these requirements, VBMA utilizes Microsoft's program, Xamarin. VBMA allows a user to login through Google authentication. Once a user is logged in, they will be prompted to enter personal information such as weight, height, gender, and type of rider. After this data is entered, a user is brought to the app screen. VBMA has its components modularized, and represented with tabs on the user interface.

The main use of the VTTI Bike Mobile App is to record a user's ride statistics and data via the phone's sensors. During a user's bike trip, data regarding the gyroscope, accelerometer, and geolocation will be tracked by Xamarin's "Essentials" package. The data retrieved over the course of the bike trip is then pushed up to the user's Google Firebase database entry. In addition, the current weather of a trip is also recorded and stored in the Firebase database. The openWeatherMapsAPI helps get the surrounding weather conditions based on the user's location.

The app has support for a user to look up a destination and to get a route to their destination from their current location. Lastly, our app has support to get the history of past trips that a user has made. These features allow a user to go for a bike trip and collect data. This data will be analyzed by VTTI for future interpretation and research.

4 Introduction

Biking is an important mode of carbon-free alternate transportation. This is an extremely important concept in highly populated areas such as cities and college campuses. With this in mind, there is a significant importance in collecting data to figure out ways we can improve bicycling experience. VTTI's report on "Street Noise Relationship to Bicycle Safety", states that "Transportation agencies at all levels generally lack the data to support safety analysis specific to bicycling.[1]" This is the specific problem that the VTTI Bike Mobile App is looking to solve. By using the application, VTTI is able to take the raw data generated by the user's biking experience and utilize it in their traffic models to help synthesize new ideas and solutions to how biking can be improved.

VTTI Mobile App is able to take in the user's age, weight, location, and rider category to get an idea of the personal statistics of the rider. These are based on the user's input and create the account connection through Google sign in.

After a user is signed in and has put in their personal information, a user will be brought to the dashboard where they can interact with various components of the app such as previous statistics, navigation, and different tools for their trip. The dashboard is broken up into different tabs allowing for easy navigation of the app for the user.

In order to start a trip, a user will navigate to the map tab and look up the destination of where they want to go. The VTTI app will then run in the background and start collecting data based on the input from the phone. On the interface side, the phone will take the user to Google Maps where their trip will be ready to start with their destination already pre-populated with the biking option on. A user can start their trip and the data will collect. At the end of the trip, the data is then sent to a Firebase server where it can be retrieved and looked at in the future.

5 Requirements

5.1 Cross-Platform

The primary requirement of the application is that it should be cross platform. This allows the researchers to distribute the application to a wider range of participants, while only requiring the software to be developed once.

5.2 Google Firebase

Another requirement for this application is the use of the Google Firebase Database to store the collected data. This allows the researchers to have one centralized location to view all data, rather than requiring users to physically go to VTTI to offload data, or requiring users to figure out how to upload data manually. Data should be automatically uploaded without additional user input.

5.3 Data Collection

Data about the user should be collected and uploaded for VTTI to analyze. When creating an account, the application will record the user's email (for account verification purposes), and prompt the user to describe what type of biker they are by entering information about themselves, such as their age, weight, gender, and biker status. During a trip, the app will record a real-time stream of the device's location, accelerometer, and gyroscopic data. For each trip, the weather conditions that day will also be recorded.

6 Design

6.1 Initial Wireframes

Figures 1a through 1f show the initial wireframes that were generated prior to the first client meeting and were used to guide the design process. These wireframes are low-fidelity and were only intended to cover all of the intended views. Each view should present to the user a minimal amount of information and options to reduce clutter and make the view's functionality as clear as possible.

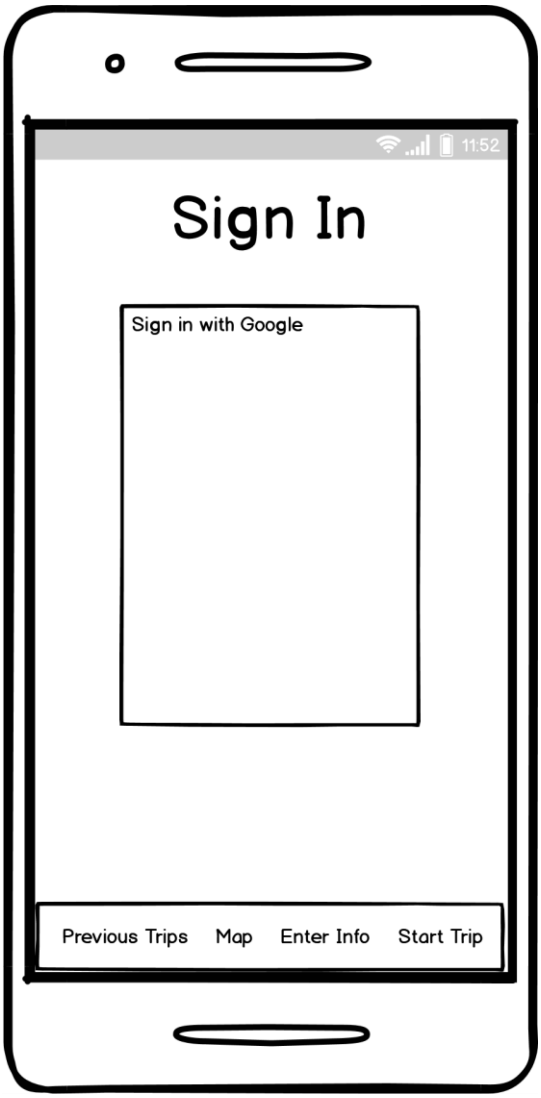

Fig. 1a: Wireframe "Sign In" Page

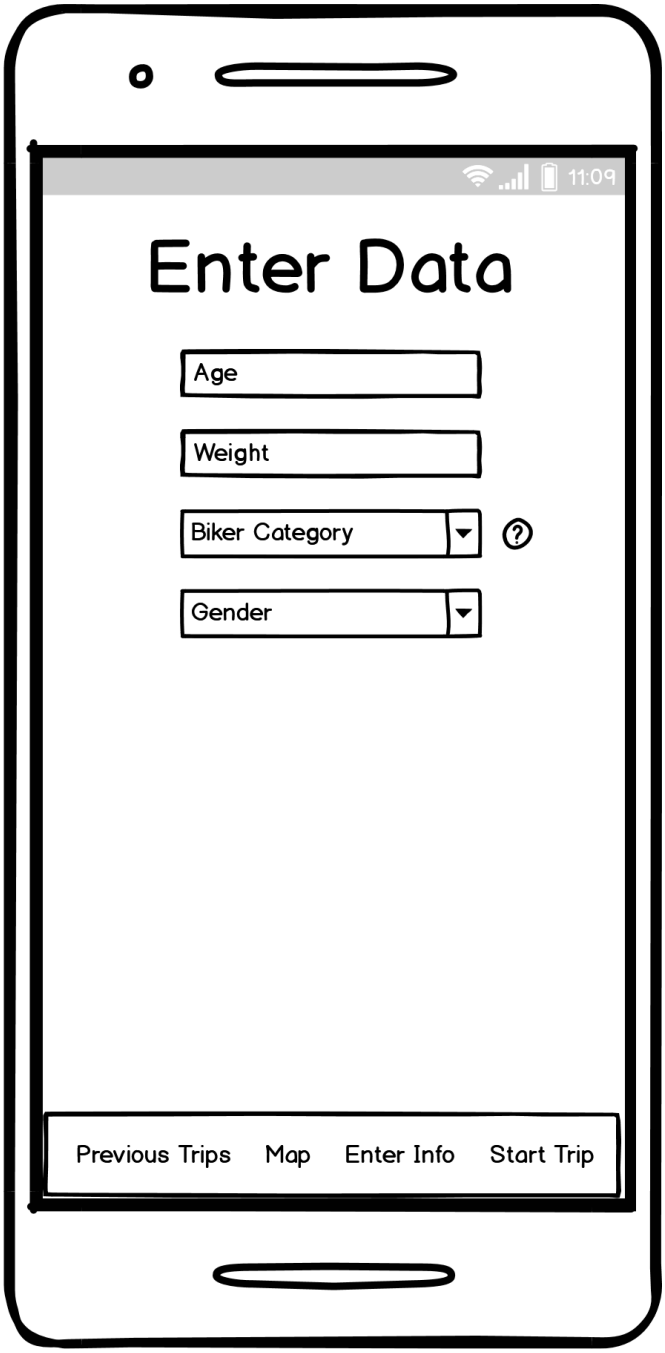

Fig. 1b: Wireframe “Enter Personal Information” Page

Fig. 1c: Wireframe “Map View” Page

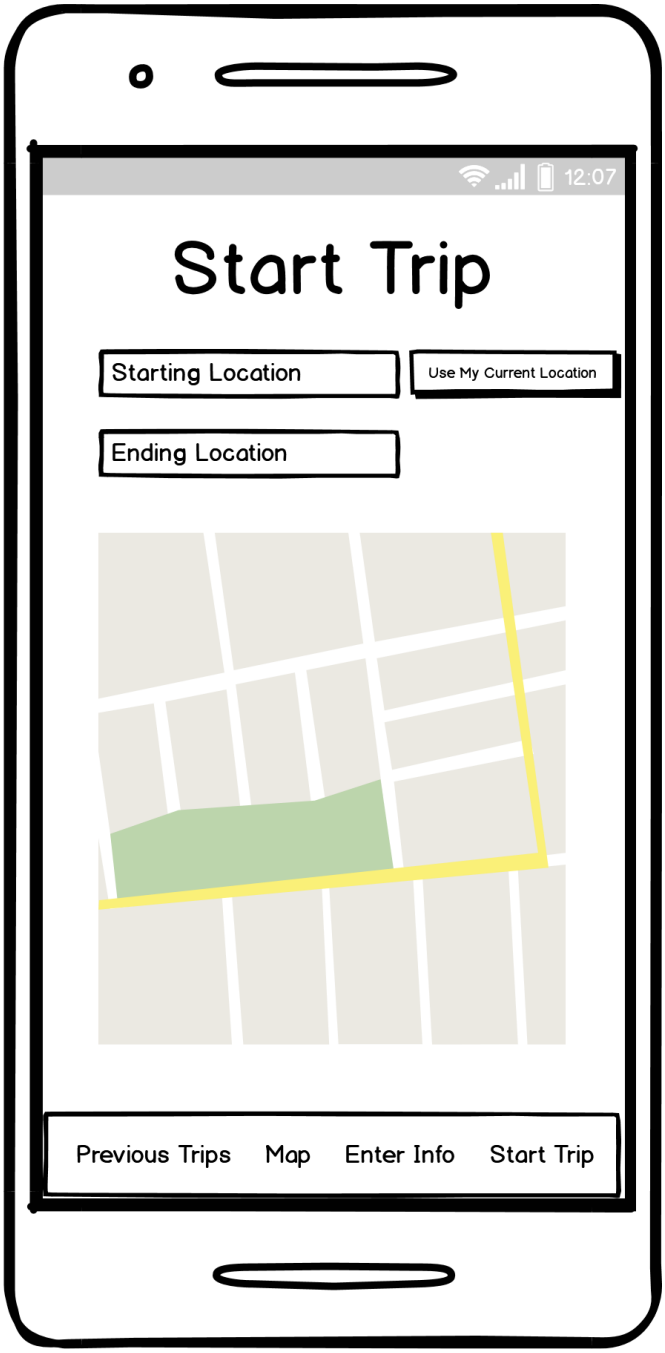

Fig. 1d: Wireframe “Start Trip” Page

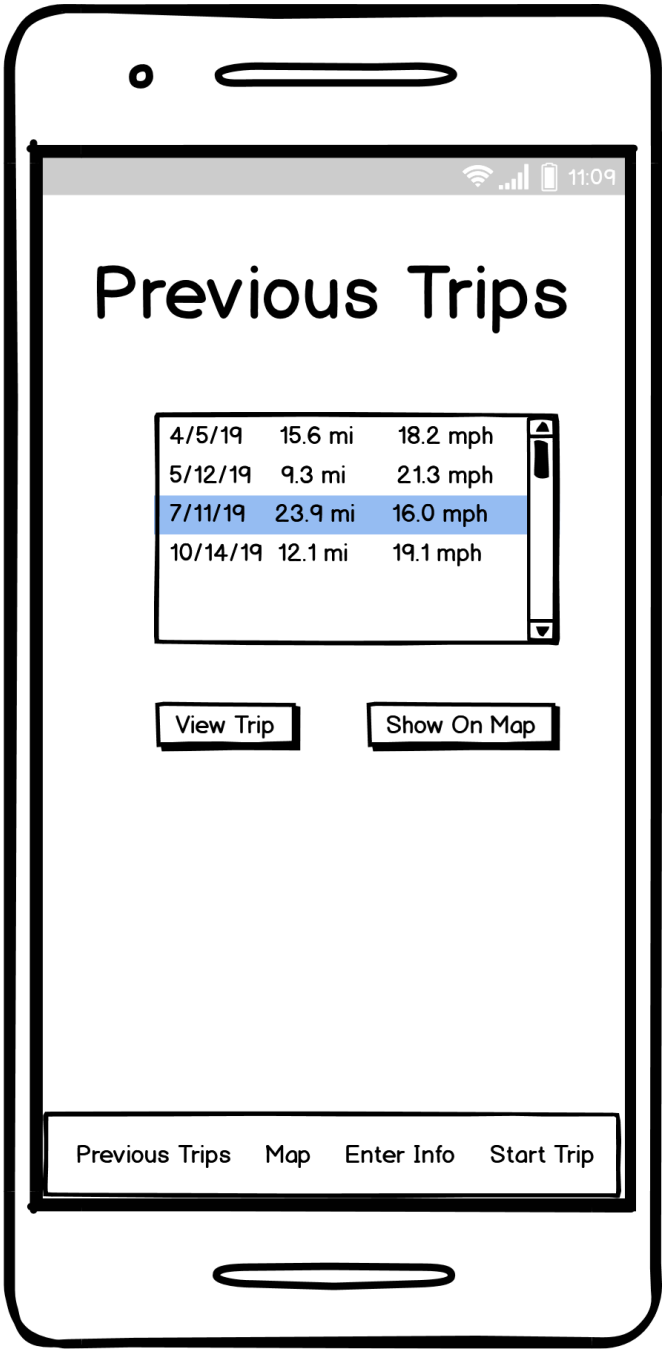

Fig. 1e: Wireframe “Previous Trips (General)” Page

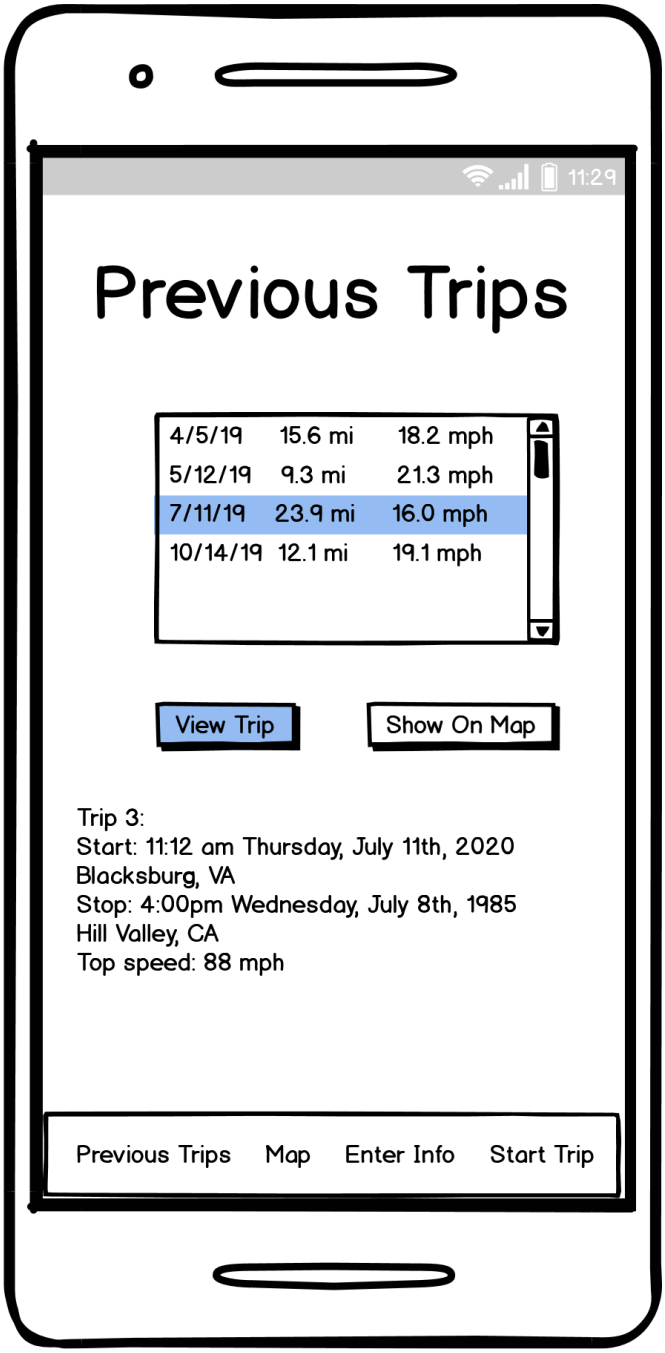

Fig. 1f: Wi-reframe “Previous Trips (Specific Trip)” Page

6.2 Proof-of-Concept Screenshots

Figures 2a through 2f show an intermediate, prototype version of the application in which all data collection functionalities of the application are present, along with Google sign in functionality. This stage was not intended to show the final design of the UI's and overall app design, but rather to create a prototype showing that the app is functioning as initially intended. This prototype was not connected to the database and instead displayed all information to the developers through the debug menu.

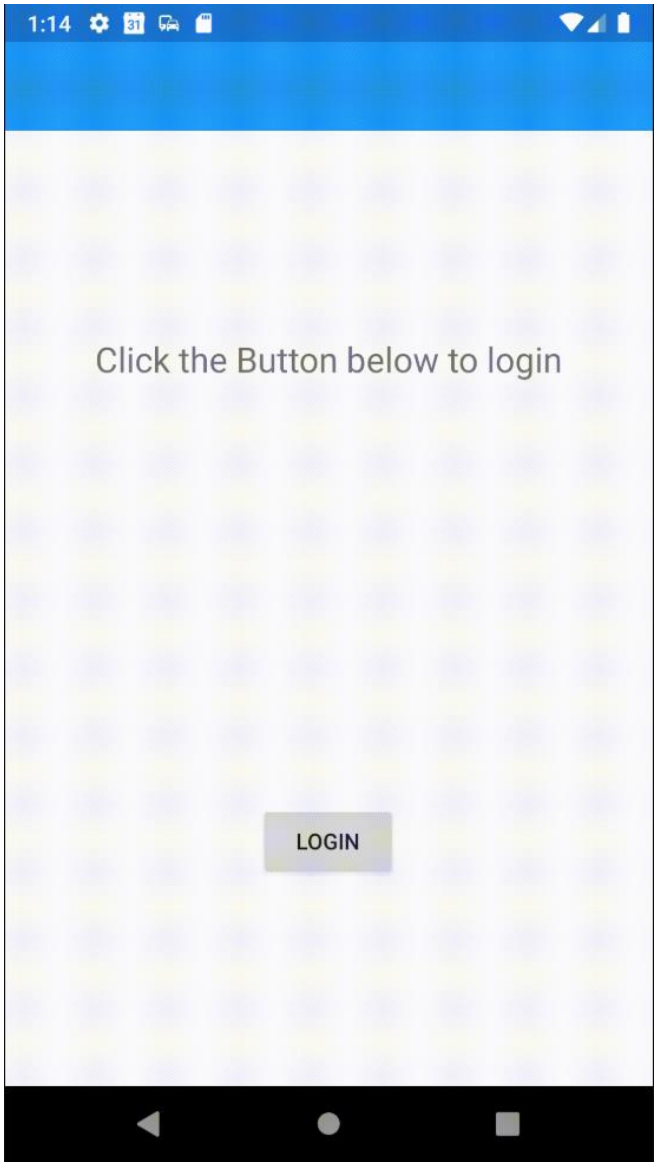

Fig. 2a: Proof-of-Concept "Login" Page

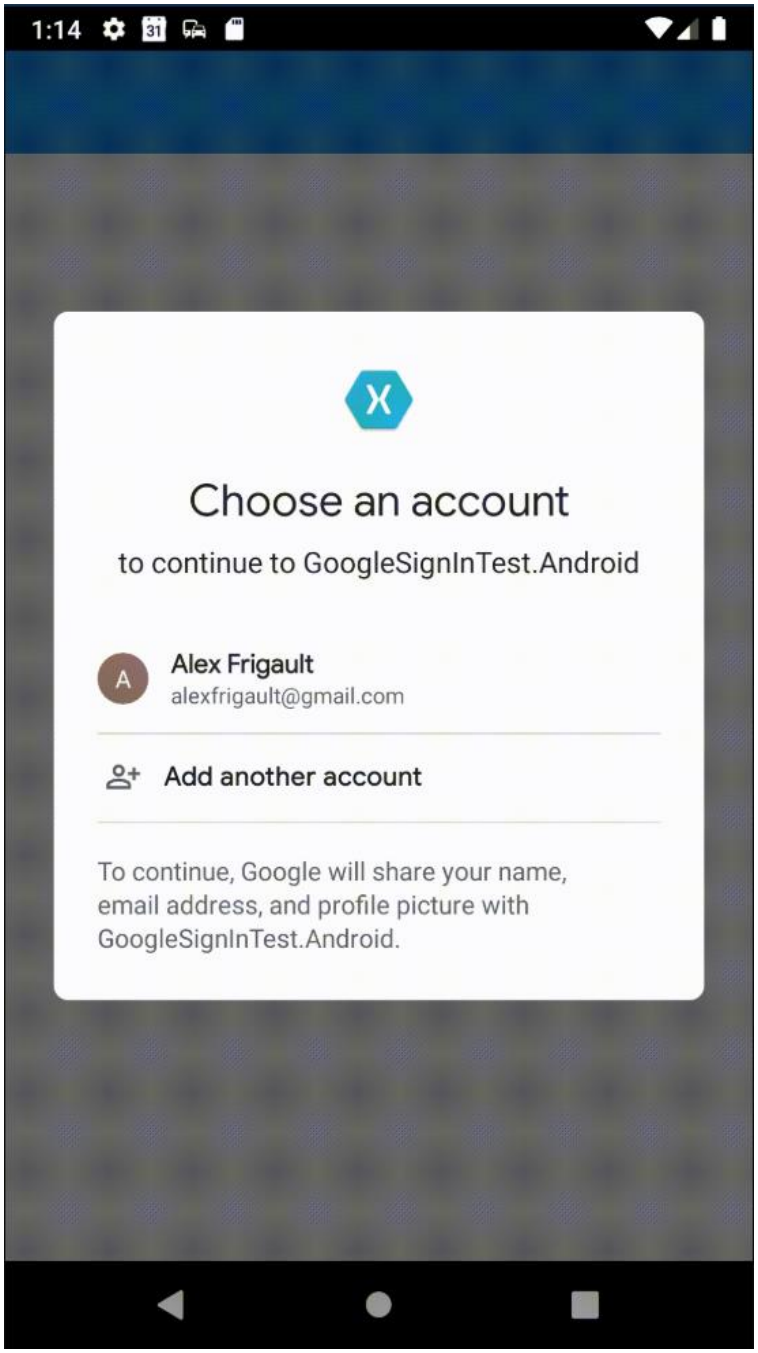

Fig. 2b: Proof-of-Concept “Login (Select Account)” Page

Fig. 2c: Proof-of-Concept "Welcome" Page

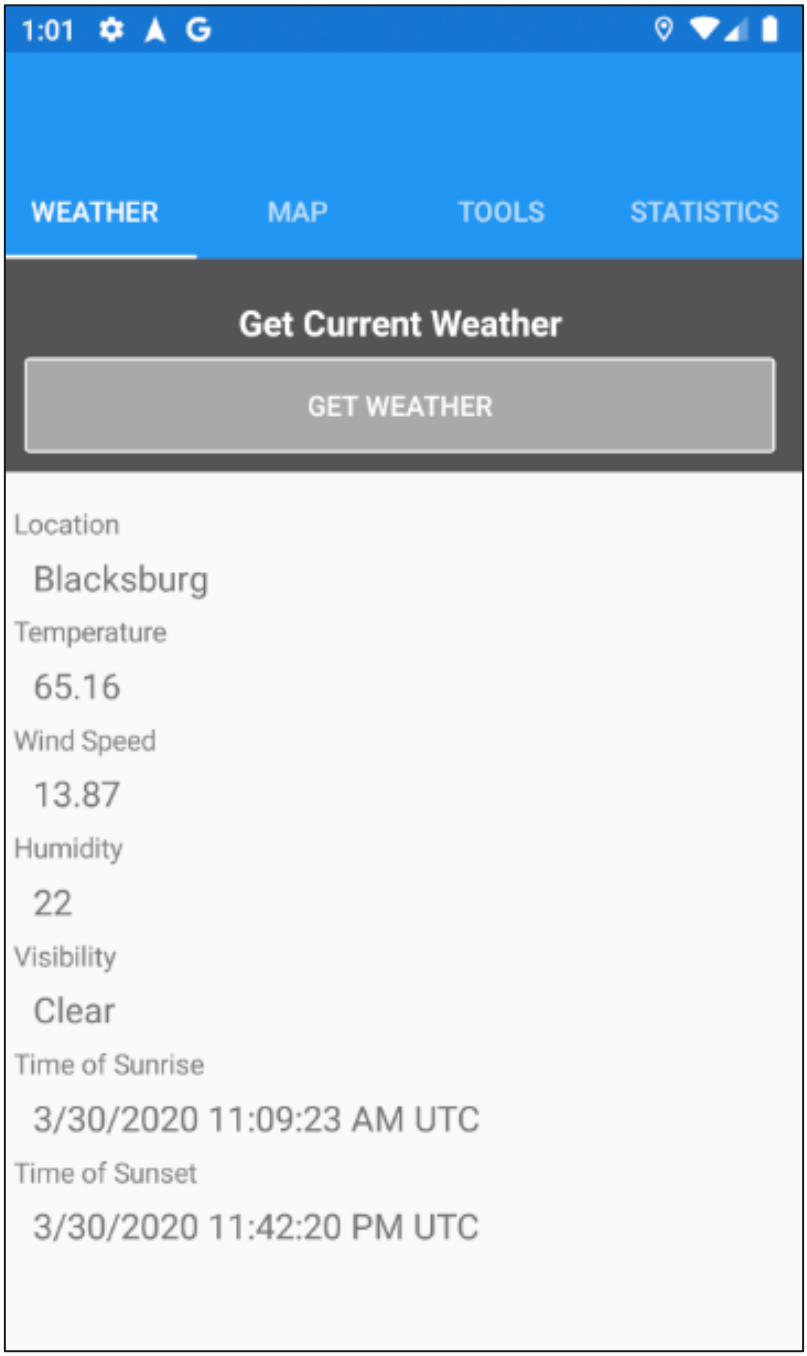

Fig. 2d: Proof-of-Concept “Weather” Page

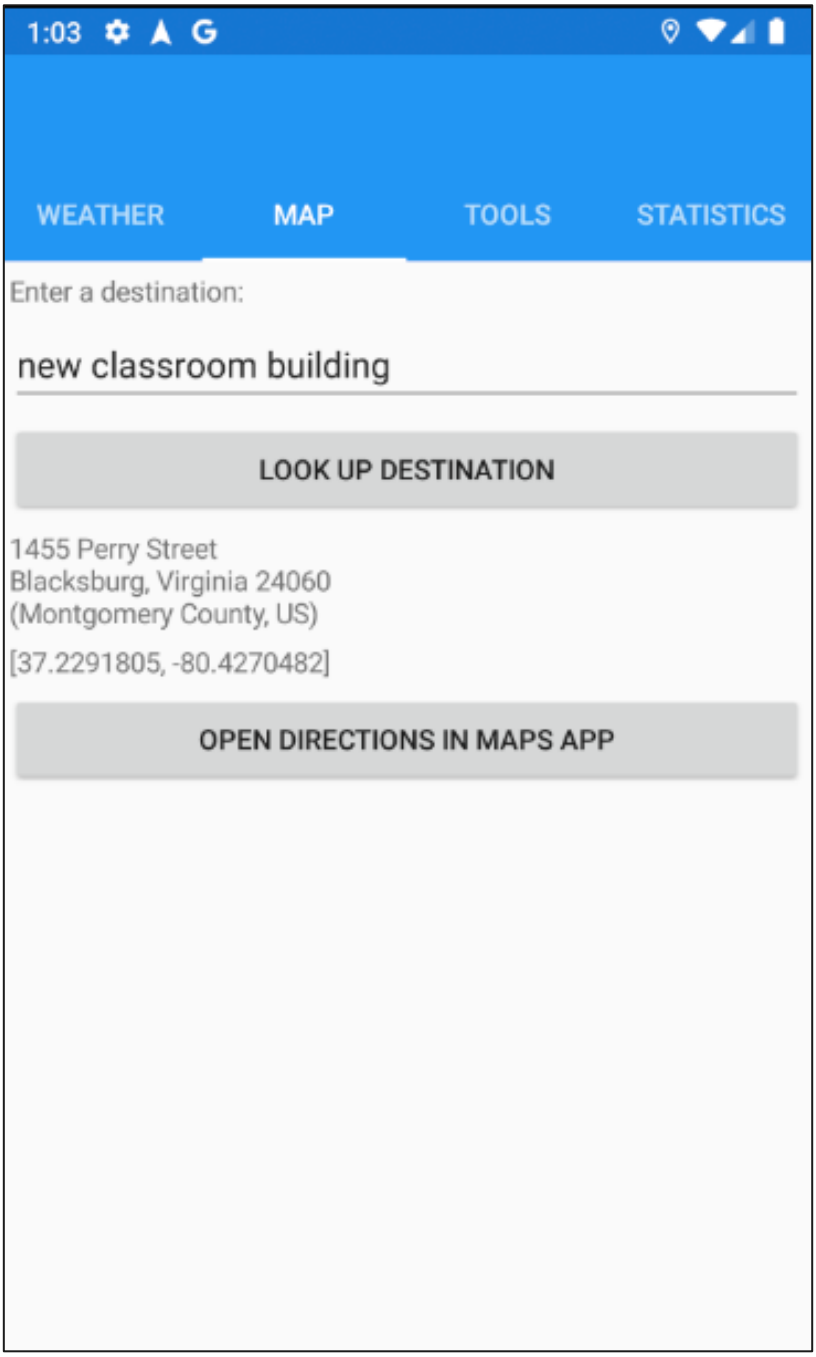

Fig. 2e: Proof-of-Concept “Map” Page

Fig. 2f: Proof-of-Concept “Tools” Page

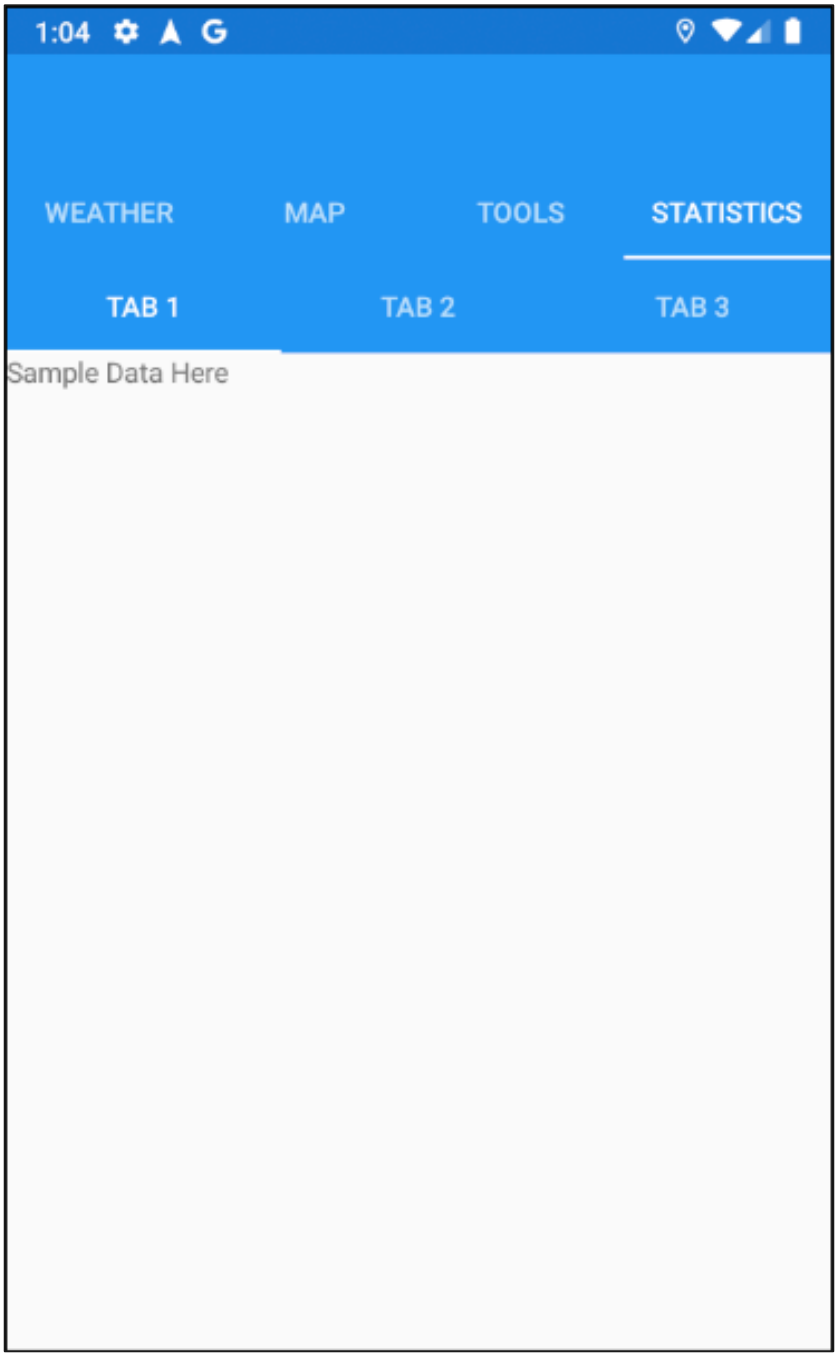

Fig. 2g: Proof-of-Concept “Statistics” Page

6.3 Implementation

In order to develop the app in a collaborative manner, a Xamarin project was created and added to a GitHub repository. From there, an agile development approach was taken to allow team members to work individually on smaller parts of the application, bringing them together for large deliverables and the final delivery to the client. Testing was done by the individual developers, as well as by other members of the team after code was pushed to the repository.

The project began by developing the data-gathering parts of the program. This includes gathering GPS, accelerometer, gyroscopic, and weather data. Concurrently, the map search function and the sign-in functionality was developed and added. These features were wrapped up into a release, which was shared to the client and received positive feedback. The future development efforts to finish the project will include integrating these features to work with a Google Firebase database.

6.4 Final Design

Our implementation design changed from our initial idea in a few different ways. We chose to go with a green theme overall to represent the eco-friendly alternative source of transportation that bicycling brings. In addition, we based our design off of the different tools and packages that we used. By switching from Flutter to Xamarin Forms, this allowed us to go with an easier tabbed version in terms of navigation. We maintained the minimalist design in an attempt to keep functionality as clear as possible.

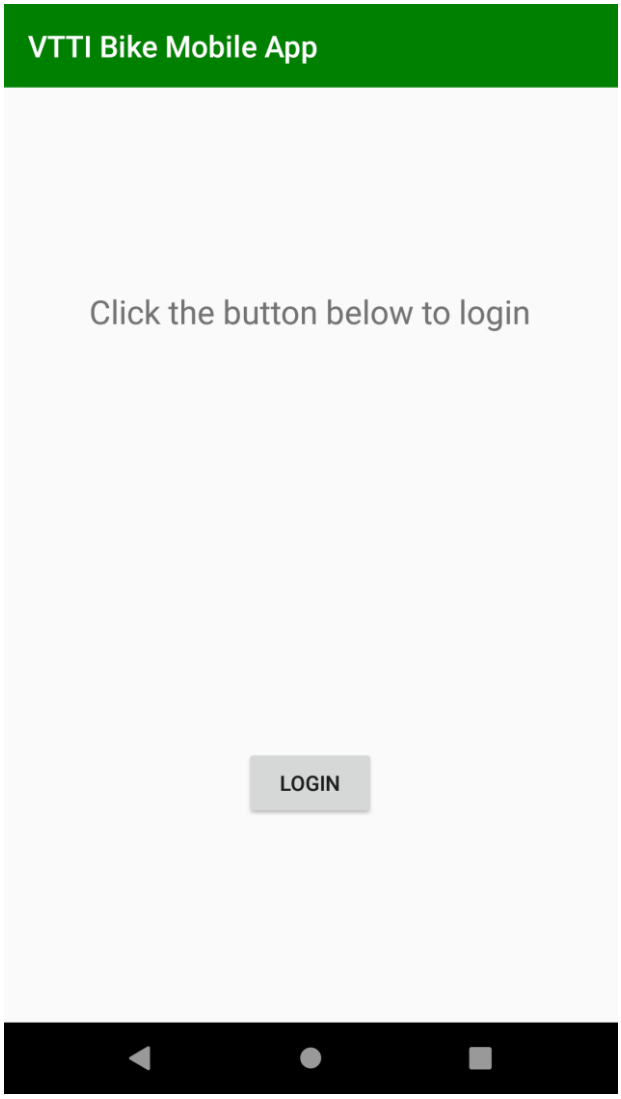

Fig. 3a: Final Design: “Login” Page

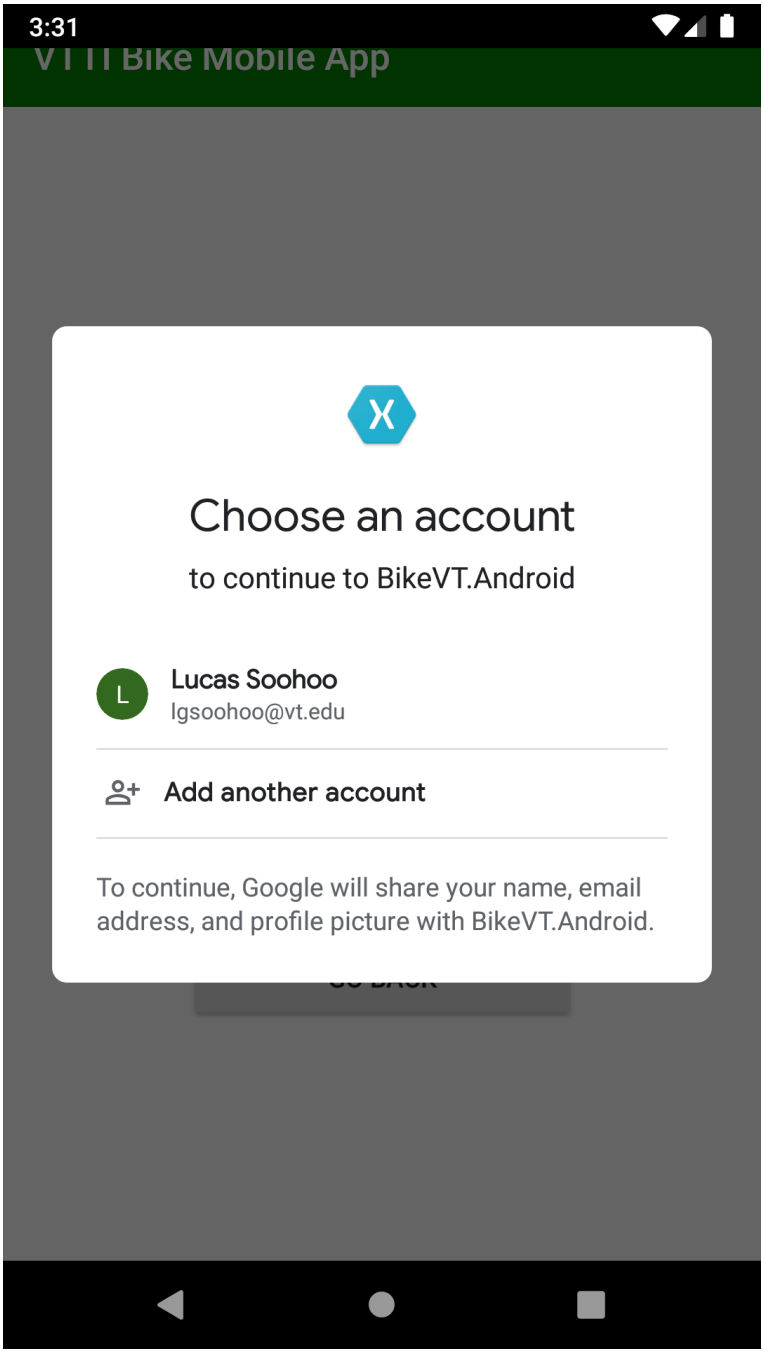

Fig. 3b: Final Design: “Login (Select Account)” Page

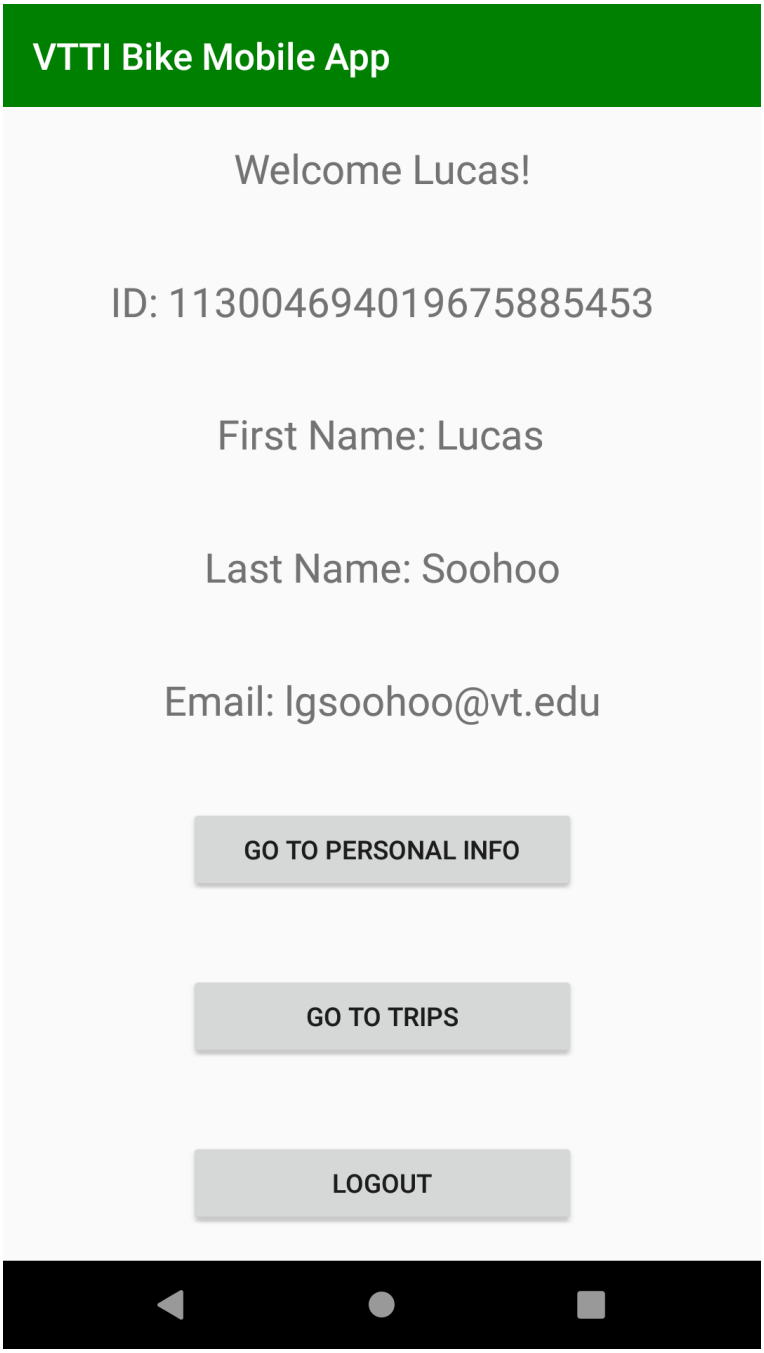

Fig. 3c: Final Design: "Welcome" Page

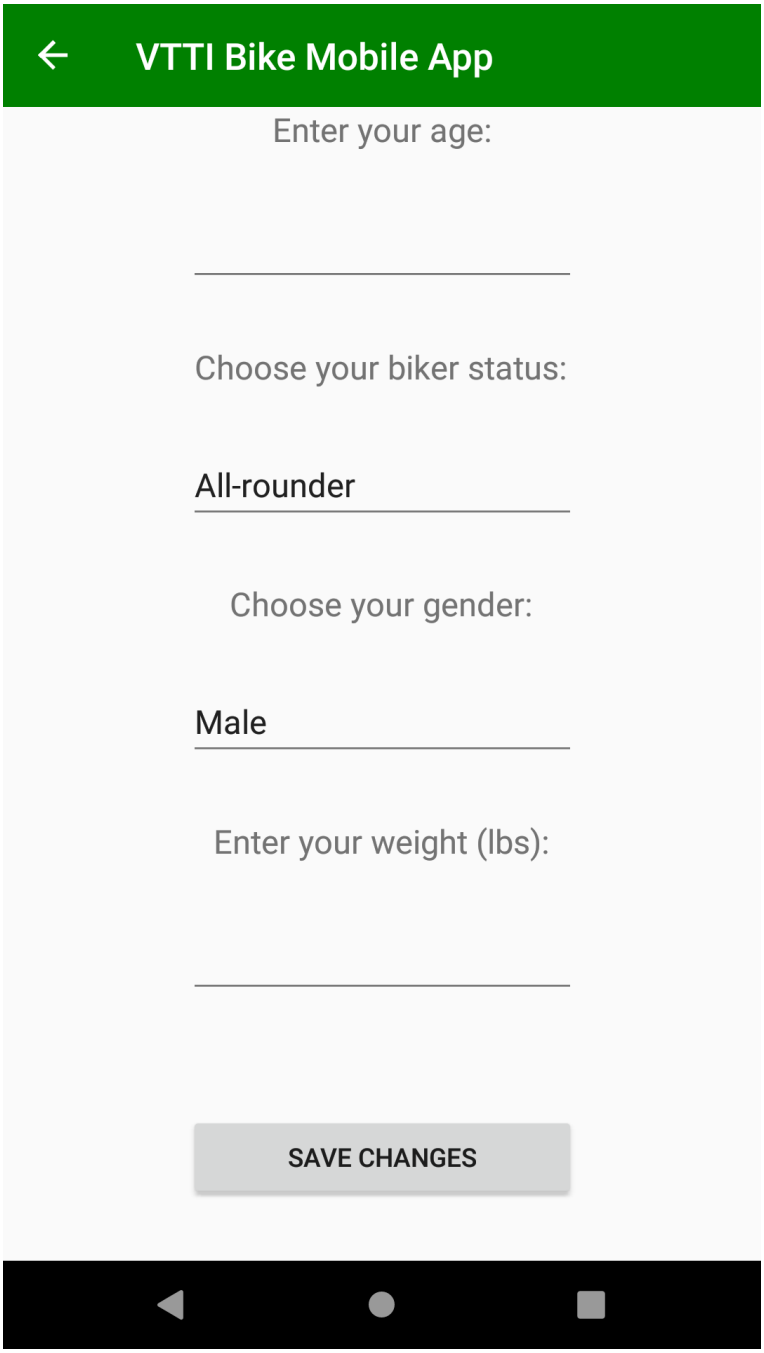

Fig. 3d: Final Design: “Personal Information” Page

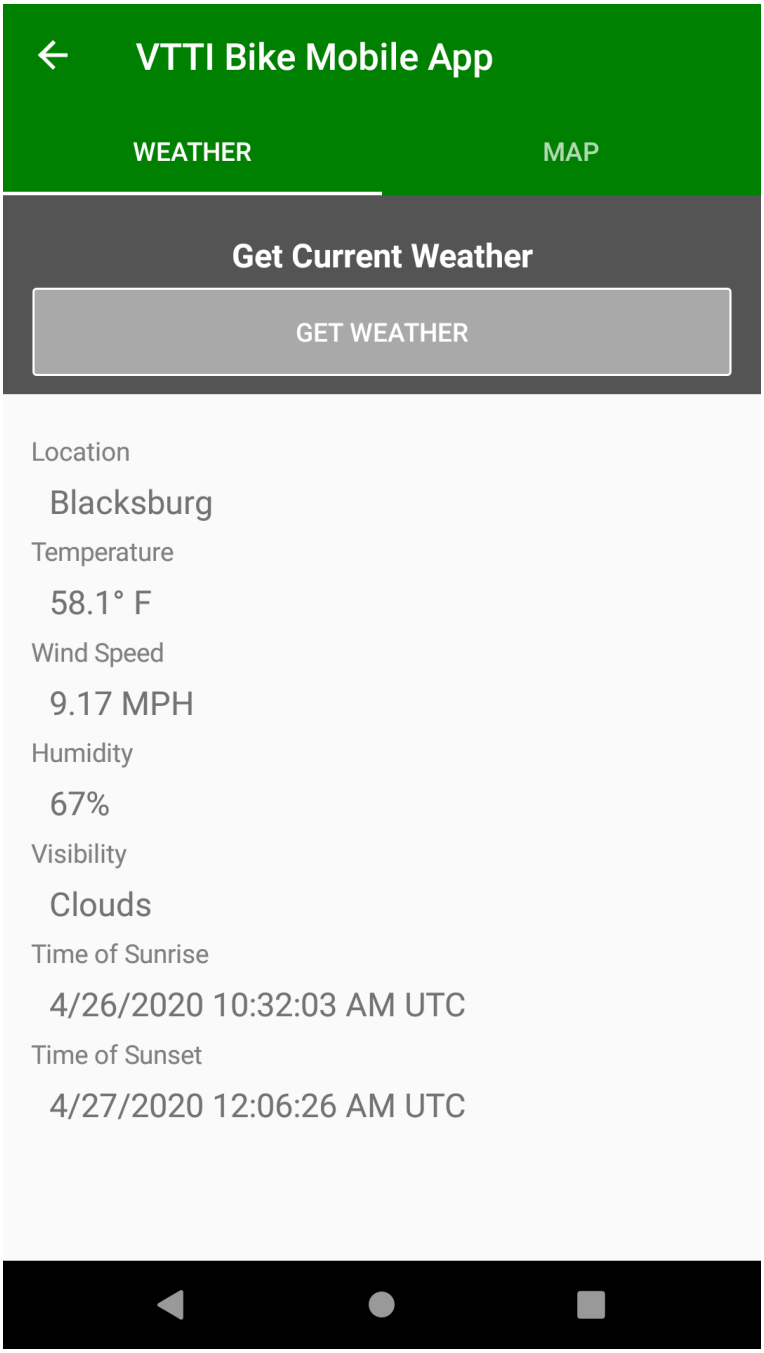

Fig. 3e: Final Design: “Weather” Page

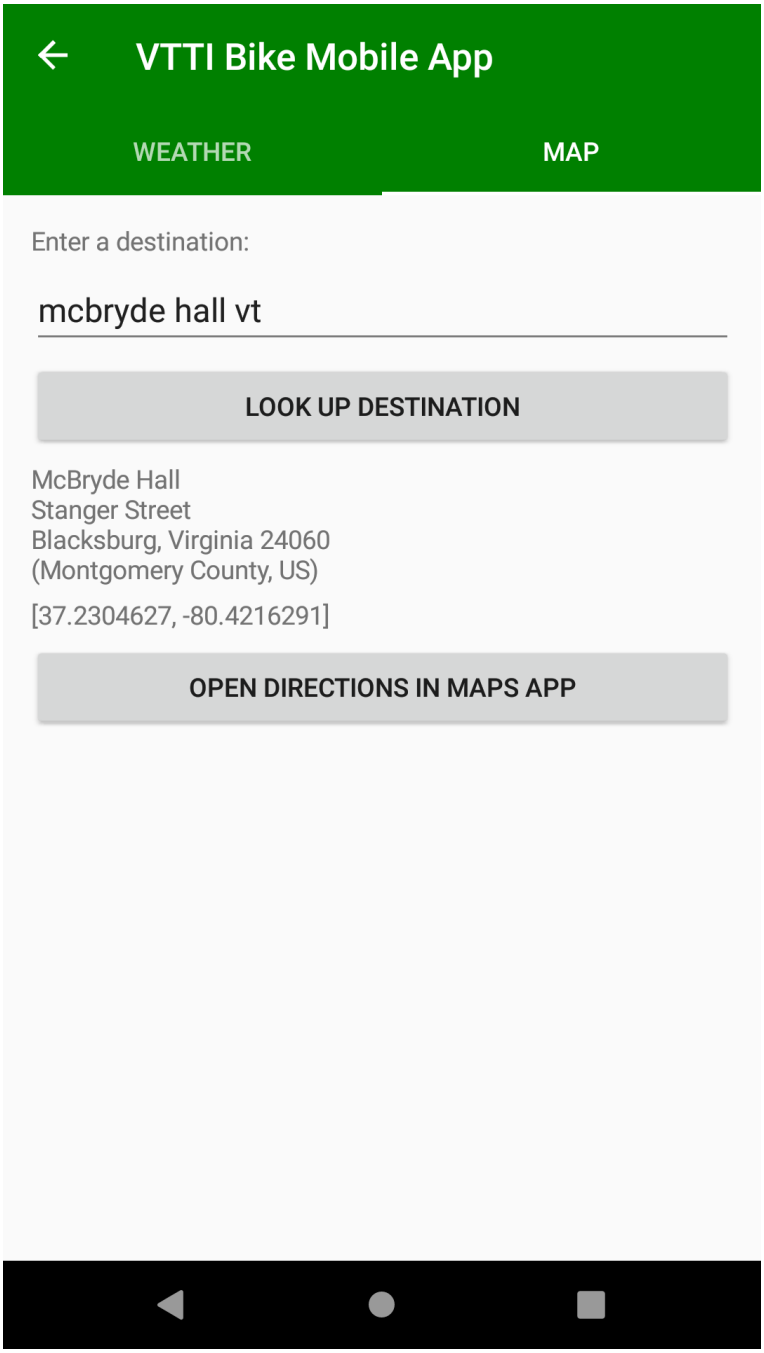

Fig. 3f: Final Design: “Map” Page

The final design features three tabs: one for the weather, one for maps, and one to view previous trips. The weather tab requires the user to have their location enabled, and will prompt the user to enable it if they attempt to get the weather with location disabled. The local weather report is obtained from an API call to Open Weather Map.

The maps page uses a user-entered destination to resolve their desired destination to GPS coordinates. Once the user has specified a desired location and visually confirmed that the resolution is correct, they can click another button to open their default map's app in bike mode with their desired location as the end point. While in the default map app, BikeVT continues to run in the background, collecting data. This data includes gyroscope data, GPS coordinates, and accelerometer data. These data streams are stored individually as strings and are uploaded to the database once each string reaches 100,000 bytes (three separate streams of data uploaded independently of each other). This reaches a good balance between using up space on the user's phone as well as reducing the number of time's the user's phone needs to connect to the database.

The previous trips tab allows the user to view their previous trips and some basic statistics about those trips. This would allow the user to use the app to track their biking progress, something commonly done by serious bikers. The data that is shown to the user will include starting location, ending location, average time, and average speed. The user is not able to see the raw data collected, such as gyroscope, accelerometer, and GPS data. The user has no ability to alter this data nor delete any of the data points.

6.5 Database

BikeVT utilizes a real-time Firebase database by Google. Firebase is a schema-less database, allowing users to push data to nodes that have different children. For example, one user could push the following to a table named `users`:

```
users
     uid_0 name: John
     city: LA
```

Fig. 4a: Database: Push Example 1

And another user could push:

```
users
     uid_1 name: Doe
     citie: D.C.
```

Fig. 4b: Database: Push Example 2

This would result in the table `users` having two different users with different children: `city` and `citie`. In order to avoid this issue, we developed our own `schema` to control how we upload data into the database. Our database has a `Users` table under which users are stored as a hashed value assigned by Google. Under this hashed value is a set of key-value pairs to store the user's age, gender, biker status, email, name, weight, and unique user ID. They also have a `_Trips` table which holds the raw data from their biking trips. Under this table, trips are stored by a unique hash value. Under that hash value includes the raw data from the trip (under the Data table), as well as basic trip data, such as the starting and ending coordinates (true coordinates, not the user's desired ending location), starting and ending time, and the weather conditions at the start of the trip.

The raw data is broken into three categories for the three types of raw data collected by BikeVT: `gyro` for gyroscope data, `acel` for accelerometer data and `GPS` for GPS data. Each value stored in each of these tables represents 100,000 bytes of raw data collected, which includes a date-time stamp for each recorded event so that they can be tracked later on during analysis. Figure 5c shows a sample of the data stored in the Firebase database. Each entry in the 100,000 bytes of raw data is separated by a slash ("`/`") and each element in each entry is separated by a comma ("`,`"). For the gyroscope and accelerometer data, an entry follows this format: `TIME,X,Y,Z/`. For the GPS data, an entry follows this format: `TIME,LAT,LONG/`.

Fig. 4c: Database: Structure

7 Testing/Evaluation/Assessment

In order to test and evaluate the application, a user was given the task of attempting to start a trip to some destination, and was asked to comment on each of the screens' GUI and talk about their user experience.

Login Page: The user noted that this page was a bit plain and could have used a logo or image to fill the space and improve the aesthetics. When they attempted to log in, the user saw the "Sign in Failed" message because the Google Sign-in box had not yet appeared. The user quickly pressed "Go Back" since they got confused and thought the application had failed.

Personal Information Page: When the user was entering numerical data (age, weight), they mentioned that having the placeholder "0" be automatically deleted might make filling out the form a little easier. They also ran into a bug where the app would accept a decimal value for their age, which would crash the app. The user also didn't know what each of the biker statuses meant, so they noted that the app could either add a small description as part of the option, or have a small blurb appear under the dropdown menu after selecting one of the statuses. In terms of UX, the user felt that the whole screen felt a little bit cramped at the top and bottom, and that using a scroll view could have solved the issue. Lastly, after saving their information, the user sat on the screen, waiting (not going back) so they may not have realized that they needed to press the back button to continue.

Weather Page: On the weather page, the user thought the "Get Weather" button's color scheme could be changed to help with accessibility, as currently the button uses a dark gray on light gray theme. However, they thought the white outline was a nice touch.

Maps Page: When searching for a destination, the user thought that sorting by locations based on proximity to the device would be better than selecting the first Google Maps result, as they had searched "Owens" referring to "Owens Hall" at Virginia Tech, but they got an address for a location in the Midwest. The address appearing below the search box helped them realize and correct this issue. The user also thought that it might be helpful to have an image of the destination appear on screen as well, for easier identification. They also thought that the text could be larger, as well as having all elements centered on the screen.

Getting Permissions: When the app asked for location permissions, the user thought that having the ability to turn on the permission within the pop-up would make it easier, especially for users who are less familiar with Android settings.

Starting/Ending a Trip: The user seemed to not realize that when Google Maps was opened, the VTTI Bike Mobile App was collecting data, so this fact could be made more clear to users. After arriving at their destination, the user seemed to not notice that the "Open Directions in

Maps App" button had switched to "End Trip", and that they needed to press this button. To fix this, they suggested having the "End Trip" button be a different color, and that the "Open Directions in Maps App" button should be disabled, rather than being hidden.

Overall UX: The user didn't understand why the app used a green theme and thought that Virginia Tech colors might make a better choice. They also mentioned having a different colored background or different style buttons might help the app feel more polished, however, they liked that the current colors (gray) help with accessibility.

8 User's Manual

8.1 Logging in

When the user opens the app, they will be greeted with the login page (Fig. 4a). In order to login, the user must have or make a Google account. Tapping the "Login" button will prompt the user to select a Google account their device is familiar with or to add a new account (Fig. 4b). If a user decides to back out of this prompt, they will be shown a "Go Back" button which will bring them back to the login page. If the user is successful in choosing a valid Google account, they will be logged in and guided to a welcome screen (Fig. 4c).

8.2 Adding / Updating Personal Information

Once the user has logged in, they will be able to tap on the "Personal Info" button and the user will be guided to a new page (Fig. 4d) where they can add or update their age, gender, weight, and biker type. Once the user is satisfied with their additions or changes, they can tap on the "Save Changes" button so that their information can be properly updated in the database. New users must enter this information before being able to continue.

8.3 Enabling Location and Storage Permissions

In order to use most of the features, the app must have Location and Storage permissions.

For Android devices, open the device's Settings application and select "Apps". Find the entry for the VTTI Bike Mobile App and select "Permissions". Use the toggle switch to enable both "Storage" and "Location".

8.4 Getting the Current Weather

When on the "Weather" page (Fig. 4e), users can view the current weather conditions by pressing the "Get Weather" button. The location, temperature, wind speed, humidity, visibility and time of sunrise/sunset will be displayed on the screen.

If the app does not have permission to use the device's location, pressing the button will notify users and show them a prompt to be taken to the device's Settings application. Follow the steps in Section 8.3.

8.5 Finding Directions

To find directions to your destination, open the map view by selecting “Maps” on the tab bar at the top of the screen (Fig. 4f). Ensure that your device’s location and mobile data are enabled. In the “Enter a destination” field, type in the name or address of your destination. Then press the “Look up destination” button. The address of your destination should be shown below. Verify that this is the correct location and then press “Open Directions In Maps App”. The app should open the device’s default maps application.

8.6 Starting and Ending a Trip

To start a trip, search your destination using the “Map” page as described in the “Finding Directions” part (Section 8.5) of this manual. Tapping “Open Directions In Maps App” will start the trip and begin data collection.

When you have finished your trip, return to the VTTI Bike Mobile App. Pressing the “End Trip” button will stop data collection.

8.7 Viewing Previous Trip Statistics

To view previous trips, simply press the “Previous Trips” tab on the top of the app. This will open a window that shows previous trips and the dates of these trips. To view the statistics of an individual trip, simply click on the trip and you will be taken to a new window with the specific details of that trip.

9 Developer's Manual

9.1 Setting Up Development Environment

9.1.1 Installing Xamarin and Visual Studio

To install Xamarin and Visual Studio, follow the guide from Microsoft's documentation[2]. Install the Community version of Visual Studio. Creating an account is not required.

9.1.2 Clone GitHub Repository

The source code for the Bike Mobile App can be cloned from here: https://github.com/Sllarsen/BikeVT.git

Once the repository is cloned, open the `.sln` file in Visual Studio. The file can be found under `/BikeVT/BikeVT.sln`.

9.1.3 Installing and Updating Packages

To install required packages, open the `Tools` menu, navigate to `NuGet Package Manager`, and select `Manage NuGet Packages for Solution...`. In the window that opens, you can install new packages, or update already installed ones.

9.2 Setting up the Firebase database

9.2.1 Navigating to Your Firebase Database

To view the database online, navigate your web browser to https://console.firebase.google.com. At the top, under `Your Firebase projects`, select your project.

9.2.2 Adding an Android Or iOS App To The Firebase

Navigate to your Firebase project as described in Section 9.2.1. Under the project title, click on `+ Add app`. Click on the iOS or Android icon, and fill out the form accordingly.

9.2.3 Linking Your Database To The App

Under the `Develop` tab in Firebase, go to `Database`. Select `Realtime Database` and copy your database's URL. Inside the App's solution, navigate to the FirebaseHelper.cs file under BikeVT/Models. Replace the URL of the FirebaseClient object with your database's URL.

9.2.4 Allowing Google Accounts As a Sign In Method

To allow signing in with a Google Account, navigate your web browser to the Firebase Console. In the left sidebar, select `Authentication` and click on the `Sign-in method` tab. Under `Sign-in providers`, enable `Google` and press `Save`.

9.2.5 Adding SHA Signing Certificate Fingerprint

To add the SHA Signing Certificate to Firebase, follow the guide from Google's documentation[3].

If Java is not added to your path, Windows users may find the keytool utility here (or in a similar directory) `C:\Program Files\Java\jdkX.X.X_XXX\bin\keytool.exe`.

Once you have your SHA-1 fingerprint, open your web browser to your Firebase project. In the left side bar, next to `Project Overview`, click on the gear icon. Then on the `General` tab, scroll to the `Your apps` section and click `Add fingerprint`. Paste your fingerprint here.

9.3 Creating an Android Emulator

Creating an Android Emulator may be useful if the developer does not own an Android Device, or would like to have the environment integrated, that is, they do not want to plug in their phone to test the app. Using an emulator also allows developers to have precise control of the sensors, such as setting the exact GPS location, something that they would not be able to do with a physical device.

If an emulator has not yet been created, click the `Android Emulator` button on the top bar (next to the green `play` button). A window titled `New Device` should appear. Adjust the emulator settings as necessary and select `Create`. Once the emulator has downloaded and been created, you can close the `Android Device Manager` window. There is no need to start the emulator. If the emulator has been set up properly, the `play` button should no longer say `Android Emulator`, but instead show the name of the emulator, along with the Android version and API. For instance, it may say: `pixel_2_pie_9_0_-_api_28 (Android`

`9.0 - API 28)`. Android may have to set up on the first time the emulator is booted. This may take a couple of minutes.

9.4 Deploying the App for Debugging

9.4.1 Using an Emulator

To run the app for testing, press the green `play` button if an emulator has already been set up. (If it has not yet been set up, see Section 9.2.) This should automatically open the emulator and deploy the app. If the screen is black, you may have to press the emulator's power button, by pressing the first icon on the right sidebar (outside of the phone frame) or by pressing `Ctrl+P`.

9.5 Firebase Database

9.5.1 Viewing or Editing the Data

Go to the Firebase Console (website) for the project. In the sidebar to the left, under the `Develop` category, select `Database`, and then `Realtime Database`. Under the `Data` section, all user's data (personal info and trip data) is available.

While viewing the data, mouse over the entries to manually add or remove elements from the database.

9.5.2 Downloading the Database

Use a web browser to view the data, as described in Section 9.4.1. Within the window containing the data, click on the kebab menu button (three vertical dots) and select `Export JSON`.

9.6 Methodology

Our aim is to have the interface that provides the desired support/functionality, which allows its users to carry out the actions, as designed. Our goal is to collect data about app users, both in account creation and real-time while using the app, and store that data for later analysis by VTTI. There are two different types of users: those who use the app and those who use data generated by the app to conduct analysis.

- Client
    - Pull data from database for analysis

- User
    - Login with Google: prompt the user to login before the app becomes fully functional. Upload all data to that user's partition in the database and read data from that user's partition for trip statistics.
    - Input personal info (age, gender, weight, height, biker type, etc.): After a user has logged in, get the user's personal information from the database so that they can enter or update the info. If the user updates their personal information, update the database.
    - Get weather: get the current weather conditions for the rider's current location.
    - Record Trip Statistics: Get the statistics for the previous trips.
    - Get sensor data: record real-time sensory data from the phone that will be uploaded to the database for later analysis.
        - Geolocation
        - Accelerometer
        - Gyroscope

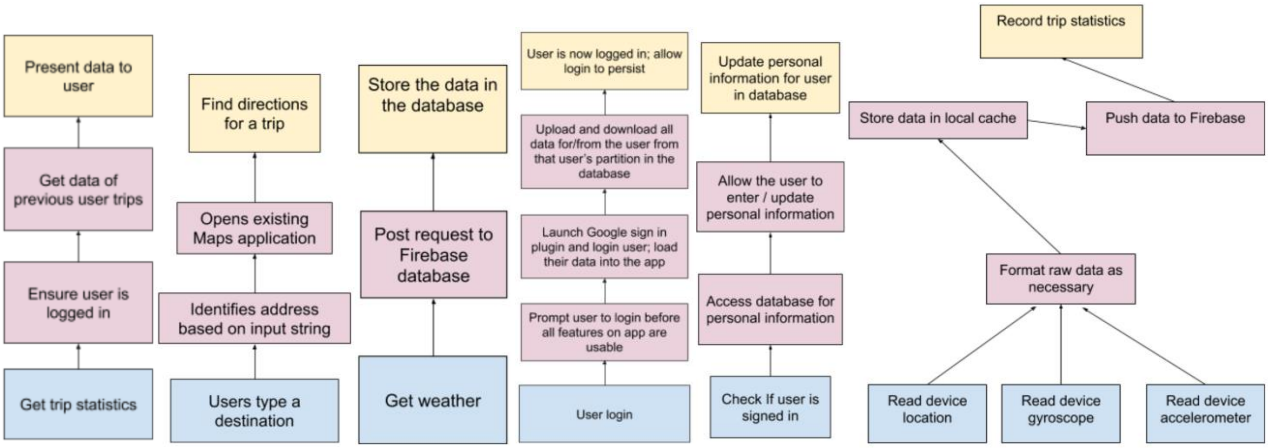

Fig. 5a: App Logical Flow

Listed are the inputs, libraries/packages, and outputs of each task used to implement each task:

- Trip Directions
    - Input: user-written destination (string)
    - Packages: Xamarin.Essentials
    - Functions from Xamarin.Essentials:
        - Geocoding.GetLocationsAsync(SearchDest);
        - Geocoding.GetPlacemarksAsync(latitude, longitude);
        - Map.OpenAsync(latitude, longitude, new MapLaunchOptions{...});
    - Output: Destination address and coordinates
    - Environment: Xamarin Forms, existing maps application (Google maps) on user's phone
- Logging in

    - Input: The users google account sign in information
    - Packages: Xamarin.Essentials, Plugin.GoogleClient, Xamarin.GooglePlayServices.Auth
    - The functions that are used with Plugin.GoogleClient are:
        - CrossGoogleClient.Current.LoginAsync()
        - CrossGoogleClient.Current.OnLogin
        - CrossGoogleClient.Current.Logout()
    - Environments: Xamarin Forms
- Get sensor data
    - Input: Reading sensor data off of the phone's sensors
    - Packages: Xamarin Essentials, FirebaseDatabase.net
    - Functions used for collecting sensor data
        - Accelerometer _readingChanged()
        - Gyroscope _readingChanged()
        - ToggleAccelerometer()
        - ToggleGyroscope()
    - Environments: Xamarin Forms
- Weather
    - Input: user location, OpenWeatherMapAPI
    - Packages: Plugin.Geolocator, FirebaseDatabase.net
    - Functions used for Geolocator Plugin:
        - CrossGeolocator.Current.GetPositionAsync()
    - Functions used from OpenWeatherMapAPI:
        - GetWeatherData()
    - Environments: Xamarin Forms
    - Output: information stored in the database
- Trip statistics
    - Input: Database data
    - Packages: Xamarin.Essentials, FirebaseDatabase.net
    - Functions: functionality not yet fully implemented
    - Environments: Xamarin Forms
    - Output: statistics from previous biking trips, including average speed, total distance, and time

- Goal: Get trip directions: Google Maps = Xamarin Essentials +GetLocationsAsync() + GetPlacemarksAsync() + Map.OpenAsync()
-
- Goal: Logging in = CrossGoogleClient.Current.LoginAsync() + CrossGoogleClient.Current.OnLogin + CrossGoogleClient.Current.Logout()
-
- Goal: Get sensor data: Workflow store data = Xamarin Essentials + ToggleGyroscope() + ToggleGyroscope() + store information in database
-
- Goal: Get weather = Current.GetGeolocationAsync() + OpenWeatherMap API call + Display information to user + store information in database
-

- Goal: Display previous trip statistics to the user: Ensure user is logged in + Get previous trip info from database + Display information to user

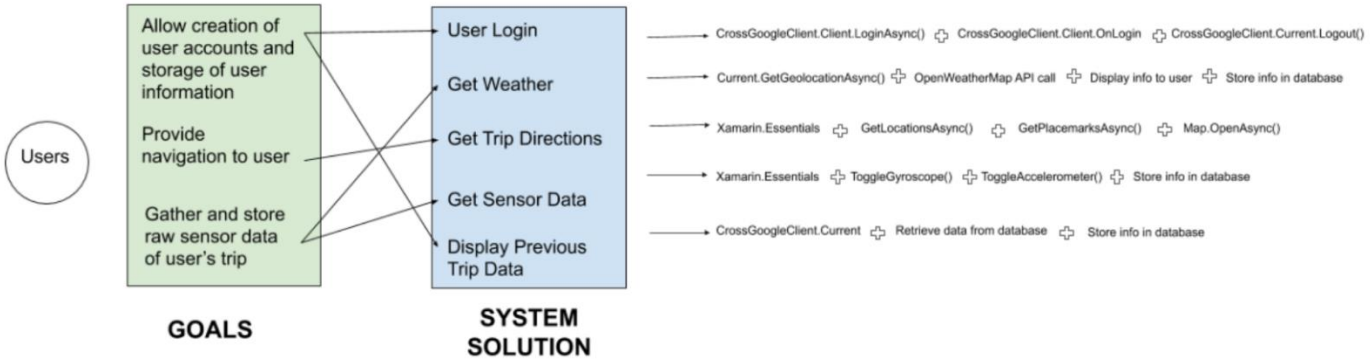

Fig. 5b: System Solution Mapping

10 Lessons Learned

10.1 Timeline

- Initial Client Meeting …….Jan. 31, 2020
- Project Approval ……Feb. 3, 2020
- Created Wireframes ……Feb. 6, 2020
- Initial Project Presentation ……Feb. 13, 2020
- Created Git Repository with Base Code ……Feb. 13, 2020
    - Loaded prototypes onto a phone
- Switched to Xamarin Forms …… Mar. 5, 2020
- Went over prototype with client …… Mar. 27, 2020
- Revised prototype and test …… Apr. 12, 2020
- Delivered final design to client …… May. 3, 2020

10.2 Problems, Constraints and Solutions

There are several key constraints that were identified during the development of the app that hindered development, as well as several constraints that were identified that changed the development of the app. The largest constraint was the development platform. One of the requirements listed by the client was that the app must be cross-platform and thus the client recommended a Google service called Flutter. Flutter, however, proved very cumbersome to use, featured packages that simply did not work properly, and had very limited documentation. As a

result, the client agreed to allow the project to transition to a different service from Microsoft called Xamarin. This switch allowed the project to remain cross-platform, while providing better documentation and support.

Another factor that proved to be integral to the design and development of the app was the immense amount of raw data generated from sensors on the phone. When collecting data at that client's recommendation of a continuous collection of accelerometer and gyroscope data and sampling of GPS coordinates at approximately 2 Hz, our preliminary tests resulted in the estimation of around 288,000 bytes/minute of raw data. Labels were removed to condense the data, but there was still a sharp increase in the collection of raw data when using the app in a deployment environment, collecting at an average rate of 800,000 bytes per minute when a "worst case scenario" test was conducted for 5 minutes (shaking the phone vigorously to maximize the number of gyroscope and accelerometer readings). This data either has to be stored locally on the user's phone and then uploaded to the database or uploaded continuously as each reading comes in. While the user could have a very limited amount of memory on their phone on any given trip, it is undesirable to upload data as it comes in, as that would result in more than 10 uploads per second. Thus, it was determined to store data in small, separate buffers. Once each buffer reaches 100,000 or more bytes, it is written to the database and the memory recycled. This creates a balance between occupying user memory and reducing the number of posts to the database.

10.3 Future Work

With what is developed thus far, we plan on integrating all the features shown in the design into one full app. After we have all the features in one app, the next step is to get our app uploading and downloading user information from a Firebase database. Once the user starts a trip, we plan on recording data from the phone's sensors and uploading that data to the database as well. Lastly, we plan on implementing the statistics page so that the user can view their trip history. The app will then be passed onto our clients, Dr. Hesham Rakha and Dr. Mohamed Magdy, for further development and testing.

11 Acknowledgements

11.1 Clients

The clients for this project are Prof. Dr. Hesham Rakha and Dr. Mohamed Magdy of the Virginia Tech Transportation Institute. They are located at *3500 Transportation Research Plaza Blacksburg, VA 24061*, and can be found online at https://www.vtti.vt.edu. Their reported purpose for the development of this application is to collect cyclist data that can later be analyzed and fed into traffic models, helping to make them more complete and accurate.

Dr. Hesham Rakha

hrakha@vtti.vt.edu

mmagdy@vt.edu

Dr. Mohamed Magdy

12 References

[1] Greg P. Griffin, Steve Hankey, Chris Simek, Huyen Le, Ralph Buehler, *Street Noise Relationship to Bicycling Road User Safety [Online]*. VTTI, 2018. Available: https://www.vtti.vt.edu/utc/safe-d/wp-content/uploads/2019/04/02-027_Final-Research-Report_Final.pdf. Accessed: April 6, 2020

[2] Craig "Conceptdev" Dunn, David Britch, *Installing Xamarin - Xamarin.* Microsoft, 2019. Available: https://docs.microsoft.com/en-us/xamarin/get-started/installation/. Accessed: April 26, 2020

[3] Google Developers, *Authenticating Your Client*. Google, 2020. Available: https://developers.google.com/android/guides/client-auth. Accessed: May 5, 2020