SP-6 Blue – Time Mileage Location Tracker

Final Report

CS 4850 – Section 02 – Spring 2024 –April 14, 2024
Professor Sharon Perry

https://github.com/SP-6-Blue-TimeMileageApp/TimeMileageApp
https://sp-6-blue-timemileageapp.github.io/TimeMileageApp/
Lines of code in Repo – 4500
Number of components - 10

Written By: Ethan Strickland, Armando Ortiz, David Nguyen, Andrew Millsap, Jacob Carrington

Table of Contents

1.0 Introduction 3

1.1 Overview 3

1.2 Project Goals 3

1.3 Definitions and Acronyms 3

1.4 Assumptions 3

2.0 Design Constraints 4

2.1 Environment 4

2.2 User Characteristics 4

2.3 System 4

3.0 Requirements Phases 4

3.1 Phase 1 Requirements 4

3.1.1 Functional Requirements 4

3.1.2 Non-Functional Requirements 7

3.1.3 External Interface Requirements 7

3.2 Phase 2 Requirements 7

3.2.1 Functional Requirements 7

3.2.2 Non-Functional Requirements 8

3.2.3 External Interface Requirements 8

4.0 Analysis & Design 9

4.1 Analysis 9

4.2 UI Design 12

4.3 Backend Design 17

5.0 Development 19

5.1 System architecture and backend 20

5.2 Information flow of mobile app 21

5.3 Information flow of mobile app 24

6.0 Testing 26

6.1 Test Plan 26

6.2 Test Report 27

7.0 Version Control 28

8.0 Conclusion 28

APPENDICES 29

Appendix A – Figma App Skeleton 29

Appendix B – Project Plan 29

Appendix C React Native Tutorial 32

1.0 Introduction

1.1 Overview

Throughout this document, we will outline the development process as we created our app. We will provide details regarding our initial plan, the requirements and design that we followed, and the explanation of how each aspect was implemented.

This document should demonstrate how we are designing and building out a new map app that sets itself apart from the preexisting map apps. This will be accomplished by the various new features being offered to them such as GPS tracking, trip history, user friendly UI, and the ability to export trip data for tax purposes, just to name a few.

1.2 Project Goals

- Create a functional app that can help users keep track of their travel history between various locations.
- Provide a map display and direction generation within the app to allow users to only require one app for all for their travelling needs.
- Track information about trips such as location and distance so that the user can use this information during tax season to get write-offs for driving between work locations.

1.3 Definitions and Acronyms

Here are the definitions or acronyms are used throughout this documentation:

- API: Application programming interface. A way for two or more computer programs or components to communicate with each other.
- DB: Database. An organized collection of information or data normally stored electronically in a computer system.
- IDE: Integrated development environment. A software application that provides tools for software development.
- Modal: component from React Native that was used to create pop up sections within the app. Such as when the user wants to update their account information. A pop screen would be displayed and ask the user to change their username, email, or passwords.

1.4 Assumptions

Our app will be quite simple to both use and understand if we assume the following:

- All devices that use the app will have location services enabled.
- All devices that use the app will have the necessary GPS hardware for location tracking.
- All devices that use the app will either have iOS or Android operating systems.

2.0 Design Constraints

2.1 Environment

- Languages: JavaScript
- Frameworks: React Native
- Database: Firebase
- IDE: Visual Studio Code

2.2 User Characteristics

- This app is designed for the general consumer.
- The interface of the app must be intuitive and easy to learn.
- Functionality is clearly labelled and easily understood.

2.3 System

- The app must be functional on both Android and iOS devices.

3.0 Requirements Phases

The following sections will outline the requirements of our app in two main phases.

Phase 1 outlines the current implementation of our app with the features that we considered the most important for the functionality of our app.

Phase 2 outlines our future plans that could improve the quality and functionality of our app that we have not had time to implement yet.

3.1 Phase 1 Requirements

3.1.1 Functional Requirements

a. **Create Account**
   - Allow a user to create an account using an email address and password.
   - This account data will be stored within the database of the system.

b. **User Login**
   - Allow a user to sign into an account they have created using a combination of email and password.
   - A successful sign-in will allow the user to access all the information and features their account has permissions to access.

c. **Reset Password**

- Allow a user to reset the password to their account.
- Users will be prompted to enter the email associated with their account.
- An email will be sent to the user.
- The email will link to a password reset page located on the website associated with the application (unless there is a better way to do this)
- The password entry in the database associated with the account will be changed after the reset password request is submitted.

d. **Start and Stop Trips**

- A user must be able to begin and end a trip when they desire.
- When a trip is started, the app will take note of the time and begin tracking the location of the user.
- While a trip is being tracked, information such as location and time will be stored in the database.
- When a trip is ended, tracking will stop, and a final time will be added to the trip in the database.

e. **Save Trip Information**

- Allow a user to store data taken from trips tracked by the app.
- The app will track the start and end locations of a trip, the distance travelled during the trip, and the date/time of the trip.
- Each trip will be stored independently in the database, linked directly to the account that started the trip.

f. **Generate Trip History Report**

- Allow a user to generate a report consisting of data regarding their trips over a certain period.

g. **Generate Directions**

- When a user enters the home page, allow them to generate directions from their location to a searched location.

h. **Display Visual Map**

- When a user is on the app home screen, a map will be displayed with the user's current location.
- A search bar will be displayed at the bottom of the page, allowing the user to search for a location.
- When a place is searched, the map will shift to the searched location.
- Once directions are generated, buttons allowing the user to start and stop a trip will appear.

i. **Display Ads**
   - For premium accounts no test banner ad would be displayed and for non-premium accounts a test banner ad would be displayed on the history and account detail page.

j. **View Account Details**
   - Users can access an Account Details page that will display a list of Preferences, Resources, and Account Information.
   - Under The Preferences Section:
     - Users will be able to change the existing page language.
     - Switch between current mobile application screen display of either Light Mode or Dark Mode.
     - Have access to their current Subscription Plan and update it as they like.
   - Under The Resources Section:
     - Contains a “**Report a Bug”** button functionality, where users can report issues found within the mobile application that they are facing or within their own user account.
     - Contains a **“Contact Us”** button functionality, where user can reach out to the development team with any issues or concerns that they are facing within the mobile application.
   - Under The Account Info Section:
     - Contains the current email and password of current user logged in.
     - Users can **“Change”** their email and password.

k. **Premium Account**
   - Users will be able to purchase a premium version of the app.
   - This version will remove ads from being displayed to the user.
   - Certain features will be unlocked when a user is using a premium version of the app.

l. **User Logout**
   - Users can logout across the entire app by clicking the logout icon in the top right corner.

3.1.2 Non-Functional Requirements

a. **Security**

a.1 Users Passwords

- Store user's passwords in database with a hashed version of the password.

a.2 Permissions

- Limit data access with permissions (Users can only see their trips)

b. **Usability**

- Users will be able to use the app with minimal to no training.
- Users will be able to generate a report of their past trips easily and quickly.

3.1.3 External Interface Requirements

a. **Software Interface Requirements**

- Google Maps Platform
- Maps API for Dynamic Visual Map
- Routes API for generating directions between locations within the app.
- Connection to Firebase database

b. **Communication Interface Requirements**

- Users must have some form of internet connection to interact with the maps and database.

3.2 Phase 2 Requirements

3.2.1 Functional Requirements

a. **Two-Factor Authentication**

- Allow a user to enable the Two-Factor Authentication feature.
- This feature will send an email to the user's email with a randomly generated code.
- If the user inputs the correct code, they will be successfully signed in

b. **Temporary Data Backup**

- Temporarily store trip data locally on the device
- In the case of the database being down, this local cache will be stored temporarily and uploaded once the database is back up.

c. **Display Ads**

- While the user is in the app, display banner ads provided by Google AdMob
- Track ad clicks and views in database.

d. **Generate Report**
   - User can customize the trip period on the report such as 1 week, 2 weeks, 3 months, or 6 months.
   - Users will be able to include an overall summary in their report which gives them the total distance traveled, number of trips taken in the selected period, and total time spent traveling.

e. **Directions**
   - When users search for a location to travel to and click "Go," onscreen directions will appear for upcoming turns or hazards.

3.2.2 Non-Functional Requirements

a. **Security**

a.1 Users Passwords
   - When a user is entering their password, blur the password unless they click the button to unblur the password. This behavior can be seen both during logging in and or changing passwords in settings.

a.2 User emails
   - Obscure email when displaying to the user (i.e. instead of JohnDoe123@gmail.com, display J***123@gmail.com)

a.3 Subscription Payment handling
   - Setup payment handling through PayPal or each respective app store.
   - Encrypt credit card information in case user wants to make future resubscriptions.

b. **Accessibility**
   - Users will be given an option of different visual themes according to their preferences (dark, light)
   - While travelling, directions will be read aloud to the users (The speed of the speech can also be adjusted)
   - Translate app into other languages so more users can use the app.

3.2.3 External Interface Requirements

a. Hardware Interface Requirements
- Compatibility with Apple CarPlay and Android Auto

4.0 Analysis & Design

The Time Mileage app that is being created went through several design iterations between its backend and frontend. Sections 4.1 and 4.2 will walk you through the design process of our app, specifically with how the UI was developed through this project and how the backend is setup and connects to screens throughout the app.

4.1 Analysis

a. React Native

React Native for Mobile Application Programming

React Native helped create the user interface and functionalities of the application. React Native gave us the advantage of code reusability across Android and IOS devices.

React Native also allowed us to easily use several valuable libraries and packages, and this allowed us to integrate more modern design to the application. For example, in the Account details page of the app (*Figure 4.j*) the icons on the left side of the buttons came from a package called react-native-vector-icons/feather. This package allowed us to get icons for the buttons and it gives the user a visual understanding of where the button navigates to if its pressed.

Limitations of React Native

Some of the group members were not familiar with JavaScript. JavaScript is the language that React Native uses to create an application. Not having a good understanding of JavaScript hindered our ability to use React native to its maximum since the use of other libraries or packages could have helped implement functional requirements such as dark mode (3.1.1.j) and changing the language (3.1.1.j).

b. Firebase

Firebase for database management

Firebase handled the data from the user's trip information. Firebase provided us with various backend services like real time database, authentication, and cloud storage. Firebase aided us in having good synchronization of mileage and trip information data.

Limitations of Firebase

Firebase, like React Native, was a database that the group was not familiar with, and we had to learn how to use the database during the development of the app.

c. Google maps API

Google Maps API for Display and Navigation

Google Maps APIs provides us with a powerful set of tools and services that greatly enhanced both the functionality and user experience of our app. These APIs enable location search, provide directions, offer distance and duration information, and geocoding.

Limitations of Google Maps API

When using Google Maps APIs, we had to be careful with how many API calls we made within a certain period of time. Exceeding certain limits can result in additional charges. While developing this app, we have made use of Google's free trial, which lasts for 3 months and allows up to $300 worth of free API calls. Once either of those run out, we would have to begin paying for what our app uses, which is not cheap.

Google Maps APIs require an active internet connection to function properly, so if users are in areas with poor internet connectivity, the app's mapping and location-based features may not work well.

d. Expo Go

Expo Go App for Development Testing and Analyzing.

The Expo go app provides a live sandbox environment for our testing development and allows us to analyze how the current existing code stack runs on a mobile device. In addition to this, Expo Go reports any live issues or bugs appearing when the mobile application's development environment is running.

Limitations of Expo Go

Having React native working alongside Expo Go was troubling in some instances because some libraries that would normally work on React native did not work on Expo go.

A major issue we had was Implementing ads with Google AdMob. Using Expo Go made it complicated since the react-native-google-mobile-ads library would have been better implemented within the React native environment since we were using the Expo Go environment to test and develop the app. The Ads could not be displayed within the Expo Go app.

Using React Native navigation could not be used with expo. Expo has its own navigation called expo router. This form of navigation needed to be implemented into our code to get the app working in Expo Go and to navigate to tabs and pages from buttons.

An example of an issue we had was when we created a file called SubsriptionPay.js in the home folder the file would show up as a tab. The file was not supposed to show up as a tab, it was supposed to be pressed from the Account details page and was supposed to navigate to the file.

e. Google AdMob

Google AdMob for Monetization

In the application we tried to implement Ads with Admob, but we couldn't get it to work alongside Expo Go. Admob does work well if you are working exclusively within the React Native environment.

Google AdMob is great to use for adding Ads to an application because it lets you use test ads, and they have a variety of ads that can be displayed such as Banner, interstitial, Rewarded, and App open ads.

To implement Ads, you would need to download the package to your program. Then, you would need to create an account in Google AdMob. Once you create the account, you would need to get the application ID and the app unit ID.

Then, with the trip IDs, you will need to write out the code in your program for the ads to be displayed and you would also need to import the package into the file you want the ad to be displayed. If we had been using services other than expo, implementing ads directly into the app would have been more feasible.

Limitations of Google Admob

The limitations of Admob are that it is complex when implementing the ads within expo. The reason it is so complex for Google AdMob to work with Expo is because the Ads do not work with Expo Go since it cannot install the package needed for it to be viewed.

In the resources we used they explained that the ads could be used if we used a simulator, but for it to work on the simulator we had to create an EAS file. This file would be used to command the terminal to open the application on the simulator. When we got the app to work on the simulator, the file where we would call the ads to be displayed would show up as a tab.

Our last option was to try to see if it worked if we created a developer account, but the developer account costs around 100 dollars. We decided not to get it since we were not entirely sure if it was going to work in the end.

4.2 UI Design

Before the development of our app could begin, we needed to create a prototype of what the app would look like and the basic flow of the user, which was accomplished using Figma.

All three figures above 4.a (login screen mockup), 4.b (map view screen mockup), and 4.c (history screen mockup) are the initial design of the app using Figma. This prototype gave us an idea on how the user flow would work and gave us the foundation to continue the development process. After developing the prototype, we began developing the frontend using React Native and Expo Go.

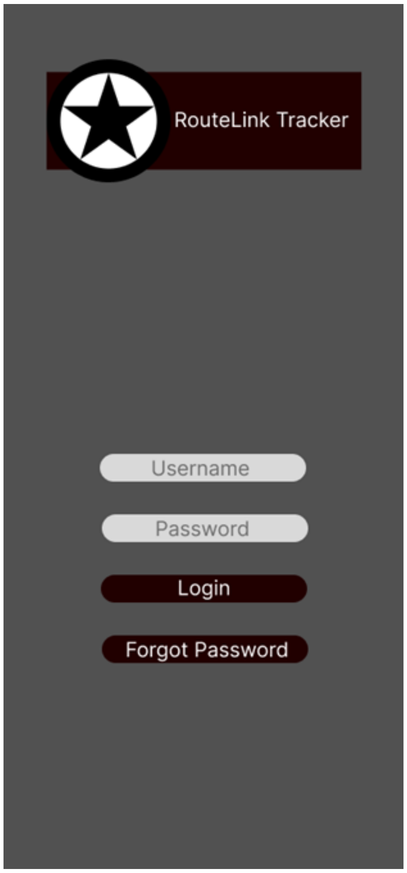


Figure 4.a: Login Screen Mockup

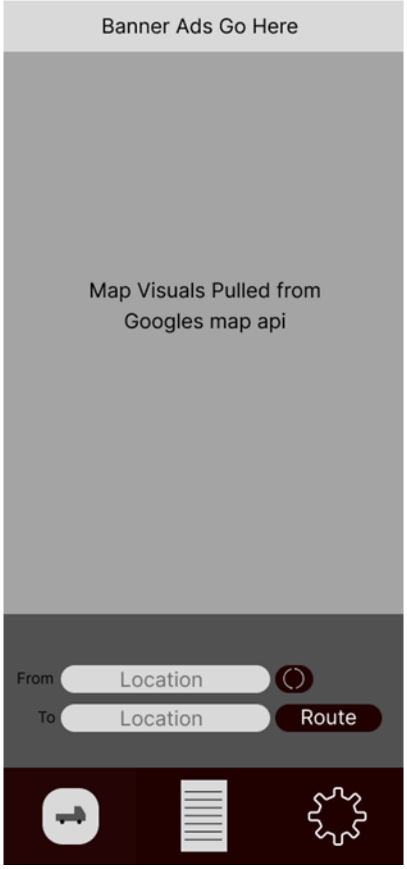
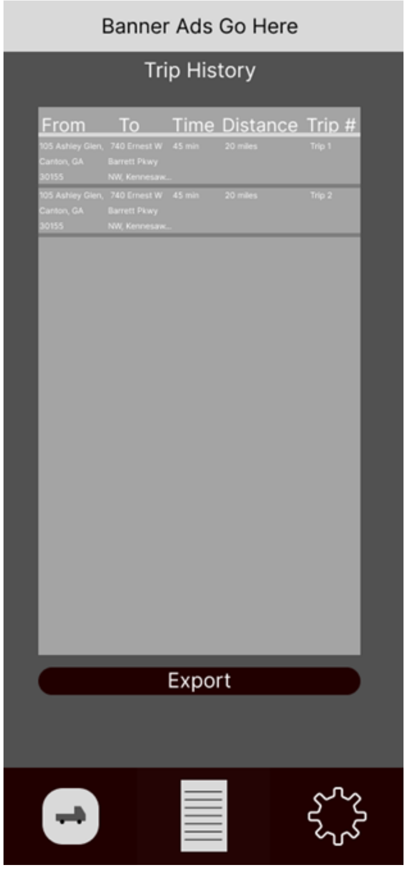


Figure 4.b: Home Screen Mockup



From	To	Time	Distance	Trip #
105 Ashley Glen, Canton, GA 30155	740 Ernest W Barrett Pkwy NW, Kennesaw…	45 min	20 miles	Trip 1
105 Ashley Glen, Canton, GA 30155	740 Ernest W Barrett Pkwy NW, Kennesaw…	45 min	20 miles	Trip 2

Figure 4.c: History Screen Mockup

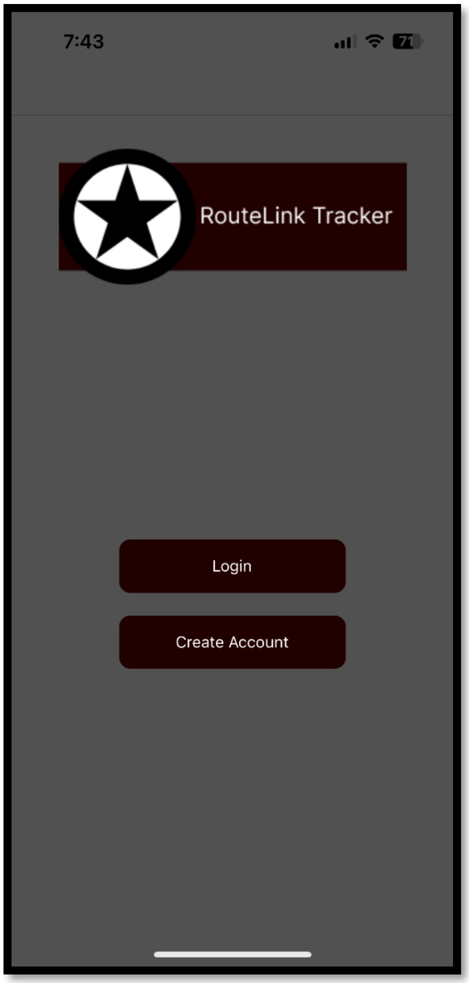
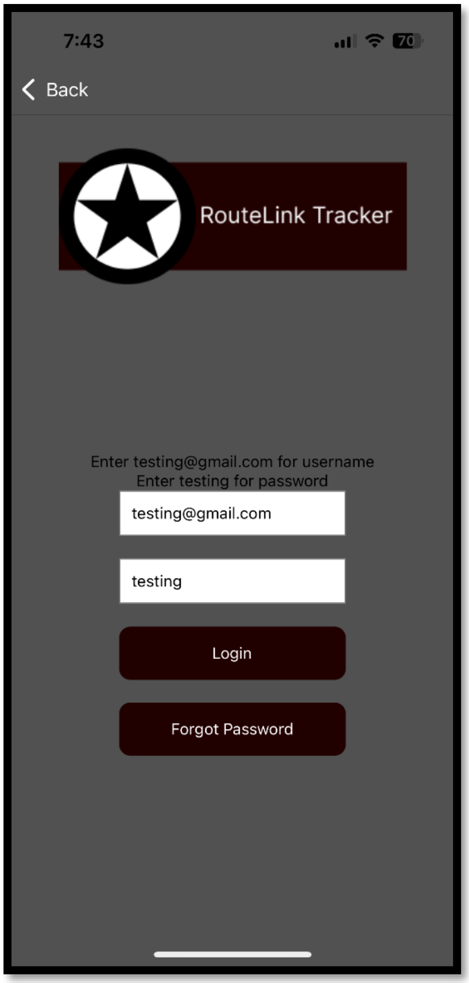


Figure 4.d: Start Screen



Enter testing@gmail.com for username
Enter testing for password

testing@gmail.com

testing

Figure 4.e: Login Screen

Figure 4.d: Start Screen

This screen will be what the users first see when they open the app. The Start Screen gives the users two options which are to navigate to the create account screen and login screen. This screen does not communicate to any APIs or our DB.

Figure 4.e: Login Screen

The Login Screen will be where the users can log into the app and navigate to the Forgot Password Screen. This screen does not communicate with any of the API's but does communicate with the DB when the user authenticates to the DB with their username and password in the app. Once the user is authenticated with the DB, they are logged into their account and brought to the Map Screen.

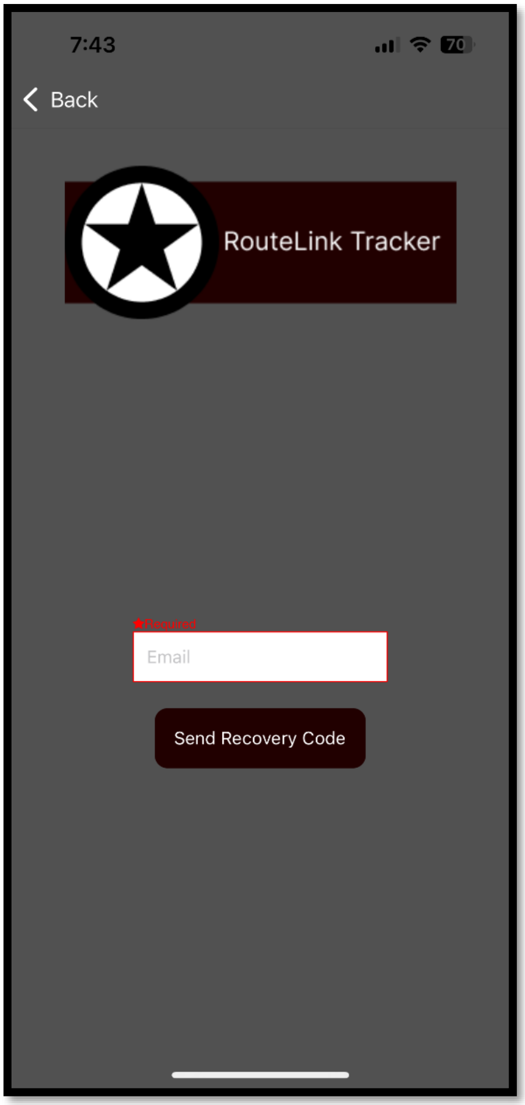


Figure 4.f: Create Account Screen

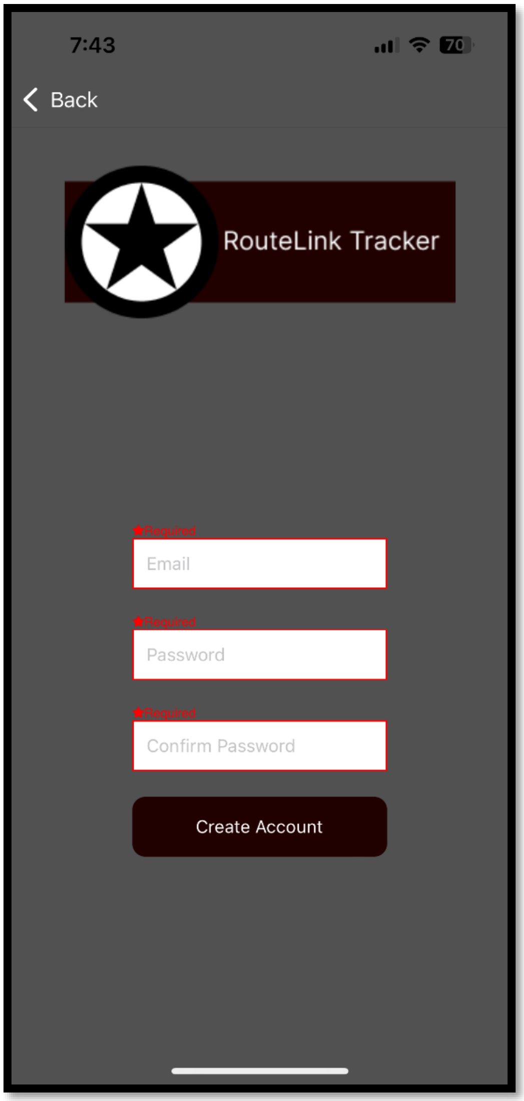


Figure 4.g: Forgot Password Screen

Figure 4.f: Create Account Screen

The Create Account Screen will be where the users are able to create a new account for our app. This screen does not communicate with any of the API's but communicates with the DB when the user fills out the form and clicks the create account button.

Figure 4.g: Forgot Password Screen

This screen will be where users can reset their password for their account. The Forgot Password Screen does not communicate with any of the API's but communicates with the DB when the user fills out their email and clicks "Send Recovery Code." Once they click the button the DB will verify the email is attached to a user, if so, the DB will then send a reset password link to that email. Once the user fills out the form with a new password and clicks submit, the DB will update with their new password.

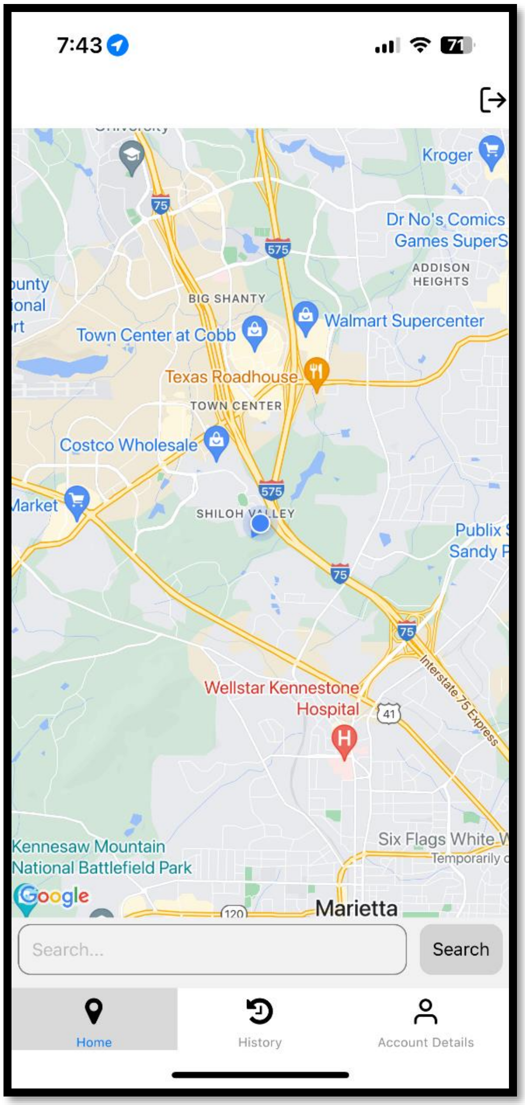
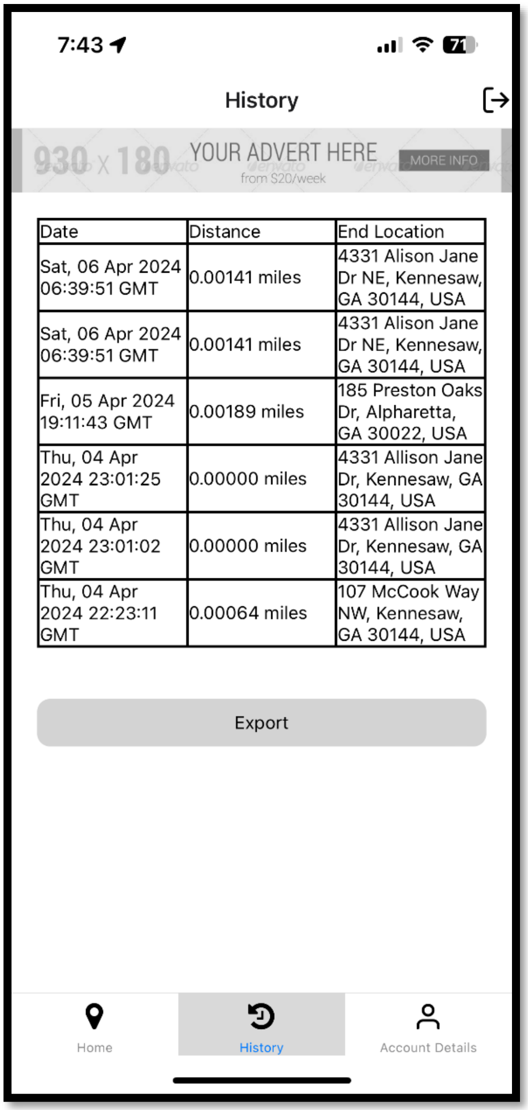


Figure 4.h: Map Screen



Date	Distance	End Location
Sat, 06 Apr 2024 06:39:51 GMT	0.00141 miles	4331 Alison Jane Dr NE, Kennesaw, GA 30144, USA
Sat, 06 Apr 2024 06:39:51 GMT	0.00141 miles	4331 Alison Jane Dr NE, Kennesaw, GA 30144, USA
Fri, 05 Apr 2024 19:11:43 GMT	0.00189 miles	185 Preston Oaks Dr, Alpharetta, GA 30022, USA
Thu, 04 Apr 2024 23:01:25 GMT	0.00000 miles	4331 Allison Jane Dr, Kennesaw, GA 30144, USA
Thu, 04 Apr 2024 23:01:02 GMT	0.00000 miles	4331 Allison Jane Dr, Kennesaw, GA 30144, USA
Thu, 04 Apr 2024 22:23:11 GMT	0.00064 miles	107 McCook Way NW, Kennesaw, GA 30144, USA

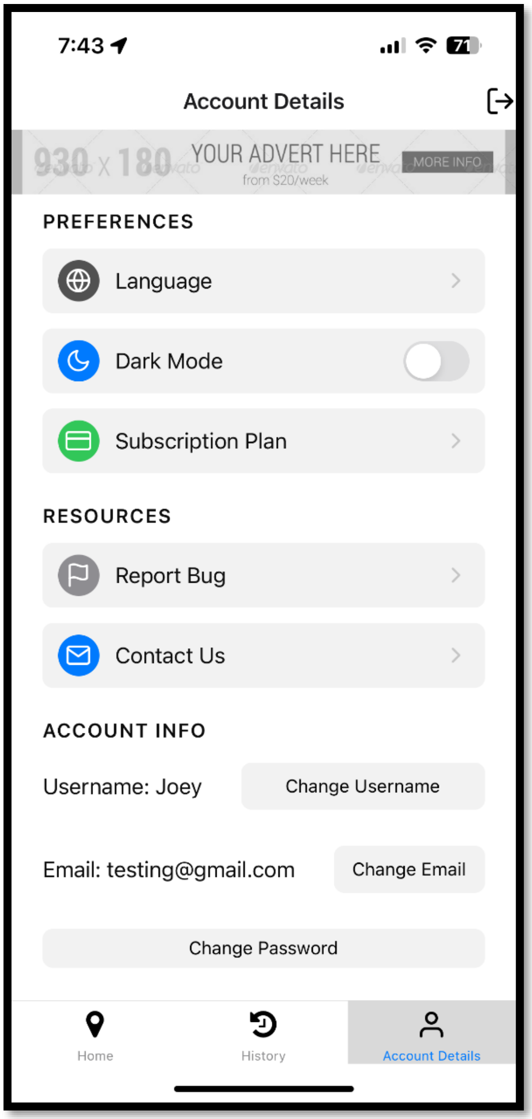
Figure 4.i History Screen

Figure 4.h: Map Screen

This is the Map Screen where users will be able to search for a location to travel to, start tracking their distance traveled, and stop tracking their distance traveled. The Map Screen communicates with both the DB and the API's which will be explained in detail in a later section. In general, anything to do with the map is hitting the API's and location tracking is hitting the DB.

Figure 4.i: History Screen

This is the History Screen where users will be able to see their trip history and be able to export their data as an excel file for use on taxes each year. The History Screen does not communicate with the API's but does communicate with the DB when it pulls down user trip history.



PREFERENCES

Language

Dark Mode

Subscription Plan

RESOURCES

Report Bug

Contact Us

ACCOUNT INFO

Username: Joey

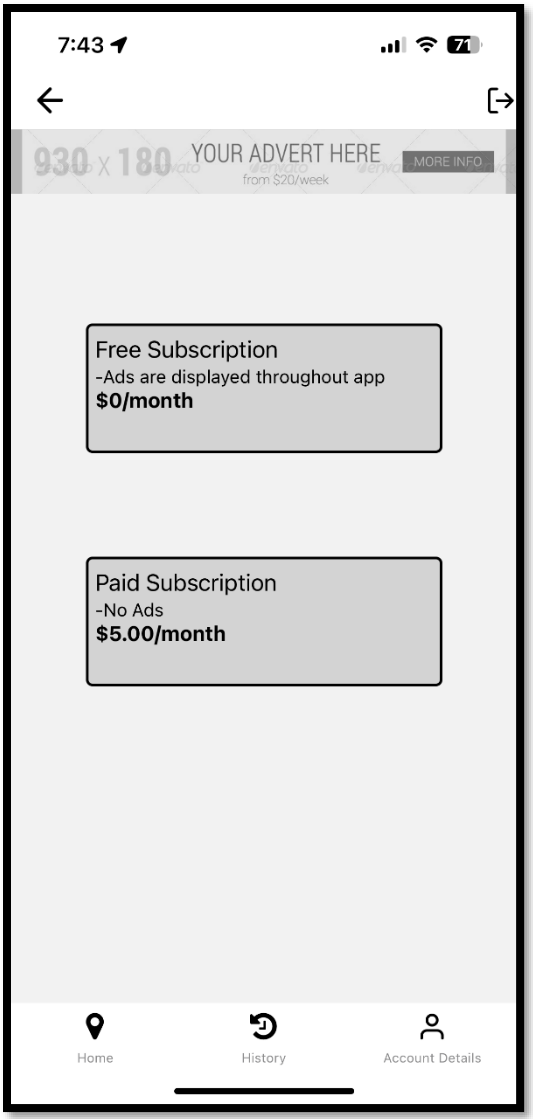
Change Username

Email: testing@gmail.com

Change Email

Change Password

Figure 4.j: Account Details Screen



Free Subscription
-Ads are displayed throughout app
$0/month

Paid Subscription
-No Ads
$5.00/month

Figure 4.k: Subscription Screen

Figure 4.j: Account Details Screen

This is the Account Detail Screen where users will be able to subscribe to the paid or free tier of the app. The users will also be able to change account info such as password, username, or contact email. There are also two buttons that will allow for the user to report bugs or contact us if they have any questions regarding the app. Currently this screen does not communicate with any of the API's but does communicate with the DB when updating account information.

Figure 4.k: Subscription Screen

This is the Subscription Screen where users will be able to switch between subscription tiers. As of now this screen does not communicate with any of the API's but does communicate with the DB to set their premium status depending on what subscription tier they choose. ￼

4.3 Backend Design

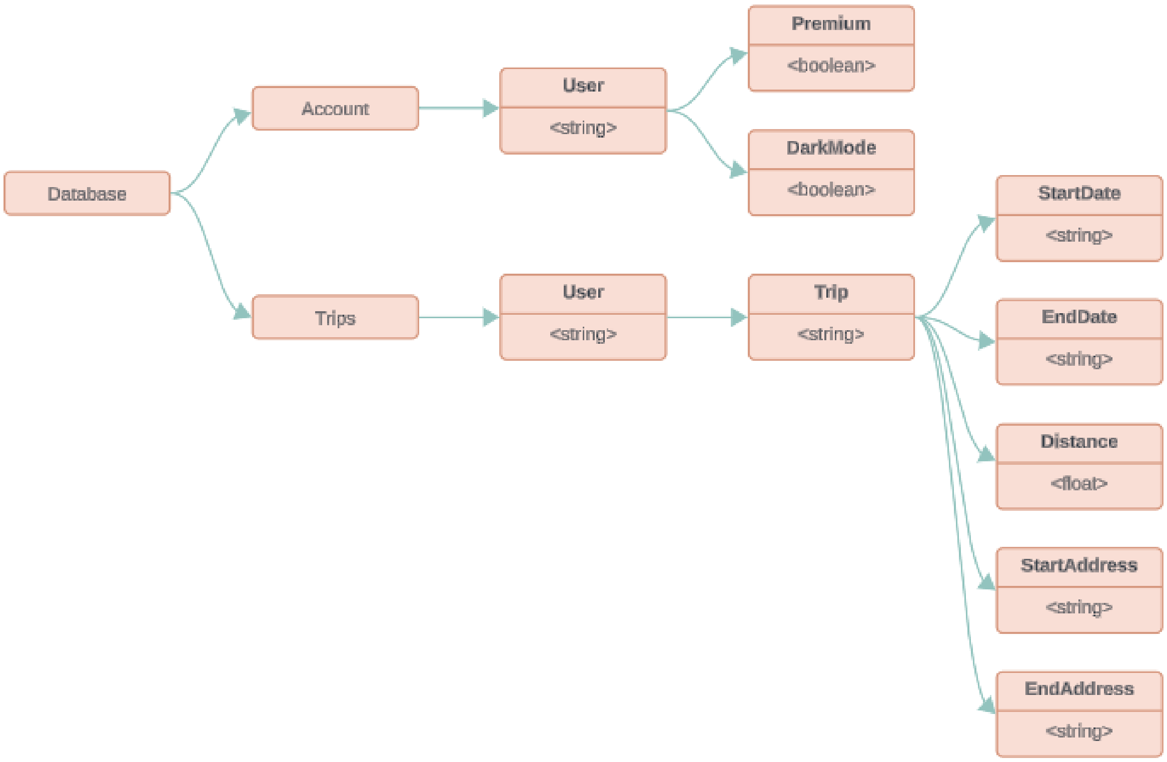


Figure 4.3.a: Diagram of Database Design

The database design we used for this project involves two primary sections in one JSON document. These sections are the Account section and the Trips section. Both sections are indexed by the User ID which is a string format.

Account – The Account section of the database stores information relating to the user account, though not the account details themselves as those are stored using Firebase Authentication.

Premium – A <*Boolean*> value used to represent whether a given user is a premium user.

DarkMode – A <*Boolean*> value used to represent whether a user prefers to use dark mode. This was done in the database instead of local device storage to serve user preferences in case a user switches to a new device.

Trips – The Trips section of the database stores the trips that a user has tracked.
For each user, every trip is tracked using a unique chronological string so that they will always be ordered.

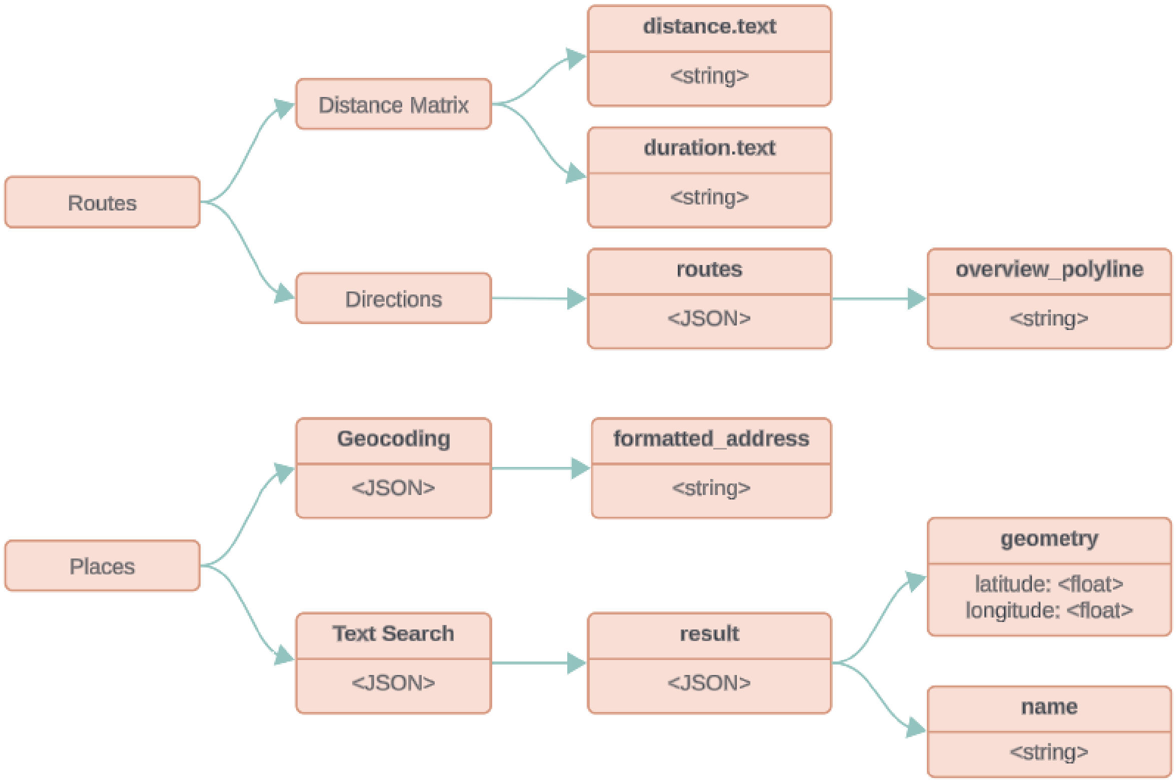
StartDate – A *<string>* to represent the date and time at which the trip started being tracked. This data is stored in UTC format to provide a clear, readable outline of when the trip was exactly started.

EndDate – A *<string>* to represent the date and time at which the trip stopped being tracked. Like StartDate, this data is stored in the UTC format to make the data more readable for users.

Distance – A *<float>* to represent the distance from the start point of the trip to its end.

StartAddress – A *<string>* to represent the address at which the trip started.

EndAddress – A *<string>* to represent the address at which the trip ended.



4.3.b: API Data Usage Diagram

a. Routes
Directions API
The Directions API is used to calculate directions between two locations. In our app, it is called in the "getDirections" function to retrieve the route from the user's current location to the searched location. The API returns a *<JSON>* response containing the route information, including the encoded polyline points representing the path. These polyline points are then decoded by a separate function before being displayed on the map. This is helpful to the user as it allows them to plan their trips and find their way to their destination.

Distance-Matrix API
The Distance-Matrix API is used to calculate the distance and duration between two locations. In our app, we call it in the "getDistanceInfo" function to retrieve the distance and duration information between the user's current location and the searched location. The API returns a *<JSON>* response containing both the distance and duration in text format. This is helpful for users as it allows them to estimate the travel time and costs of their trips.

b. Places
Text Search API
The Text Search API allows users to search for places based on a text query. It is called in the TextSearch function in our app, and it is used to find a location matching the user's search query. The API returns a *<JSON>* response containing a list of places that match the query, along with details such as the name, geometry, rating, and some other relevant information.

Geocoding API
The Geocoding API is used for converting geographic coordinates (longitude and latitude) into human-readable addresses. We call it in the "getAddressFromCoordinates" function to geocode a set of coordinates into a corresponding address. The API takes in the longitude and latitude as parameters and returns a *<JSON>* response containing the formatted address associated with those coordinates.

5.0 Development

The following sections will explain our thoughts on the resources we used such as React Native, Expo, etc. It also discusses the issues we had with these resources. We will also discuss the system architecture of the app and the information flow of the app. In those sections we discuss how the user interacts with the app and how certain buttons take you to different pages or tabs within the app.

5.1 System architecture and backend

When designing the Mileage Tracker app, the system architecture should be designed to manage the tracking and recording of the data efficiently. The system architecture will be made up of various key components responsible for specific functionalities.

User Interface

- The user interface will allow users to interact with the app.
- The user will be able to login into the app.
- The user should be able to change the settings of their app like light or dark mode.
- The user should be able to choose where they want to go to and come from.

GPS Mileage Tracking

- This component should allow the user to use the app as a navigation system.
- The tracking must provide the time it will take to reach the destination.
- The app must generate the path and provide directions for users to reach their destination.

Data Management

- Our data management component should be able to handle storing the distance that the user has driven.
- Must be able to save and store the users overall trip information.

Reporting

- This component will give the user mileage reports such as their trip history which can be used for tax write-offs each year.

Monetization

- This component will let us make money from the app.
- Banner ads will be added to the app, but they are designed to not be intrusive like the pop-up video ads on other applications.
- There will also be a subscription feature users would need to pay to get other functions from the app.

5.2 Information flow of mobile app

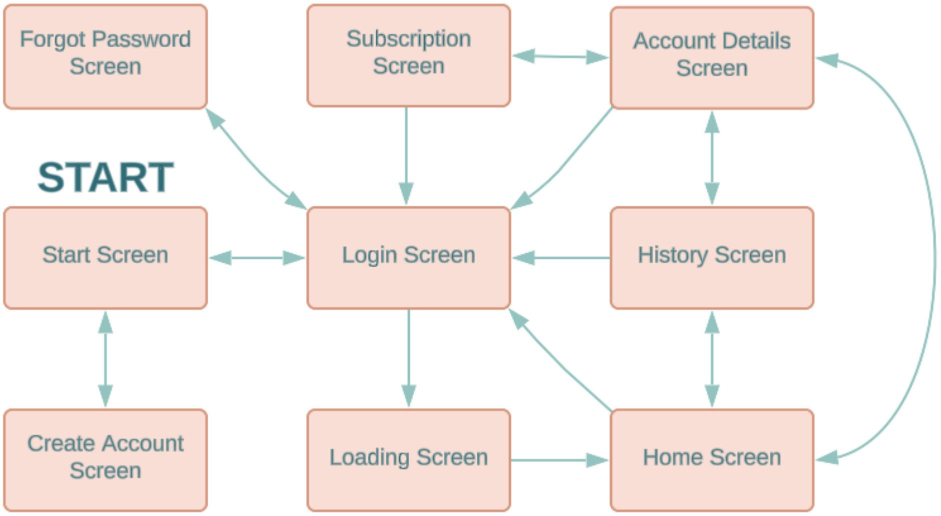


Figure 5.2.a: User Flow

Start Screen
This screen is where all users will land on whenever they open the app. There are two buttons displayed on this screen which are "Login" and "Create Account". If the user presses either of these two buttons, they will be navigated to the appropriate screen. There is no data being collected on this screen since this is just a placeholder whenever the user opens the app.

Create Account Screen
This screen is navigated to by the "Create Account" button on the Start Screen. Once the user presses this button, they are sent to the Create Account Screen where they can create a new account. Once the user fills out the form with their username and password and they click the "Create Account" button they will receive an email to confirm their account was created successfully and will then be able to log into the app with the new user. When the user navigated back to the app after verifying their email address, they will be redirected to the Start Screen.

Login Screen
This screen is navigated to by the "Login" button on the Start Screen, when the user presses this button, they are sent to the Login Screen where they can enter their password and username to login. This screen is also where the user can click a button to reset their password with the Forgot Password Screen.

Loading Screen

This screen can only be navigated to when the user successfully logs into the app with their password and username. Once the user fills out the login form and clicks “Login” on the Login Screen they will be taken to this Loading Screen for a few seconds to allow the backend to fully load all necessary data into their app. After a few seconds have passed, the user is then taken to the Map Screen.

Home Screen

This screen can be navigated to by either clicking the left tab on the bottom navigation bar or when the user first logs into the app. The Home Screen is where the user can search for a location to travel to, start their trip, and stop their trip. Users are also able to look around on the map for other locations they might want to travel to, like existing maps applications. The user can logout by pressing the logout button in the top right corner of the screen and on the bottom the user is able to navigate to two other screens. The center tab on the bottom of the screen will take the user to the history screen and the far right will take the user to their account details screen.

History Screen

This screen can be navigated to by clicking the middle tab on the bottom of the screen. When the user is taken to this screen, they will see the six most recent trips taken. If the user clicks the export button, then they will be prompted with options on how to save the excel file that was generated from their trip history. The user can logout of the app by clicking the logout icon located in the top right corner.

Account Details Screen
When the user presses on the Account details tab on the bottom right of their screen. The user will be shown a screen that displays Preferences, Resources, and Account Info.

In the Preferences section there are 3 buttons Language, Dark Mode, and subscription plan. We are still working on implementing Language and Dark Mode, and for now these buttons are unusable. The subscriptions plan button does work and once the user presses it, it navigates the user to a different page where they can decide if they want to continue with the free or paid version.

In the Resources section the user will see two buttons called Report Bug and Contact Us. These buttons will navigate them to their web browser, and it will display a google form that asks them what the bug is or what issues they are having with their account for the contact us button.

The final section is Account Info. The user will see buttons for username, email, and password. These buttons will let them change their username, email, and password. The buttons will not take them to another page, but it will have a small screen called modal in react native. The modal pops up and asks them what they would want to change their username, email, or password to.

Subscription Screen
When the user presses the subscription plan button from the Account details tab. Their screen will display two boxes. They can click these boxes, but one box is to continue with the free subscription and the box below is for the premium subscription.

Once you click these buttons there will be a "modal" that pops up on the screen for the free subscription there will be a small page that says thanks for your purchase. Then for the premium subscription once the user clicks it the page will display another modal like the free subscription modal, and it will ask for the user's credit card information.

Forgot Password Screen
When the user presses "Forgot Password" button on the Login Screen they will be sent to the Forgot Password Screen where they will enter the email address attached to their account and click "Send Recovery Code". Once this button is pressed the user will receive an email with a link, once the link is pressed, they enter their new password and click submit. When they navigate back to the app, they should have been sent back to the Login Screen.

5.3 Information flow of mobile app

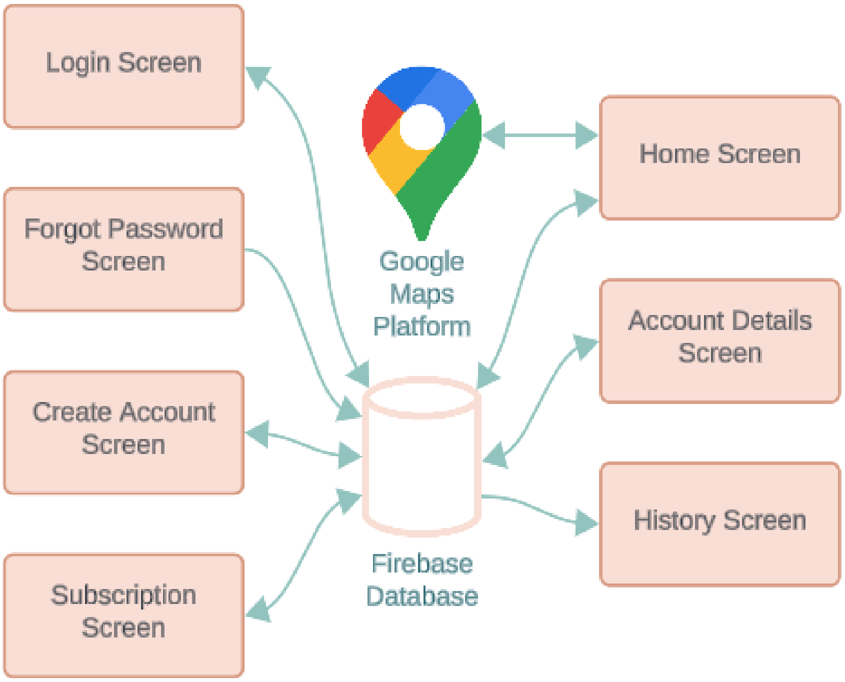


Figure 5.3.b: API & DB Flow

Login Screen

When the user attempts to login to the app, the login information will be pushed to firebase. There, firebase authentication verifies that the email and password that the user input are correct and belong together. Firebase authentication then sends a response saying whether the login was a success, and a user auth object is received to use the signed-in account information.

Forgot Password Screen

When the "forgot password" button is pressed, an email with a verification code is sent to the user's email. Once the user inputs the verification code, the code will be verified with firebase authentication, and when the correct response is received, the user can input a new password. Once the new password is input, the password is sent to the firebase authentication database to update.

Create Account Screen

When a user enters account information on the create account screen, the email that is used is first sent to firebase authentication. Firebase authentication will verify that this email does not exist in the database and send confirmation back to the app. After confirmation is received, all the account information is pushed to the firebase authentication database.

Subscription Screen
When a user first enters the subscription screen, the *premiumStatus* variable associated with the account will be pulled from the database. If the status is premium, the user will not be able to pay for additional premium. However, if the status is not premium, the user will be able to input credit card information and purchase it. Once purchase confirmation is received, the app will update the *premiumStatus* variable associated with the user.

Home Screen
The home screen retrieves most of the trip data used to describe trips. When a user clicks the start trip button, the *"startTime"* (UNIX Epoch Time) and *"startLocation"* (GPS Coordinates) are retrieved. After a trip has occurred, the user will click the stop trip button, when the *"endTime"*(UNIX Epoch Time), *"endLocation"* (GPS Coordinates), and *distance* (miles) will be retrieved.

Finally, once all data is collected, UNIX Epoch Time data is converted to date information with UTC format and GPS Coordinates are converted to addresses using a Geocoding API. This information is then pushed to a new trip in the trips section of the firebase real-time database.

When a user decides to search for a new location using the app, an API call is made to the Places API to retrieve the GPS Coordinates and name of the target location. Once a trip is started, a call is then made to the routes API to retrieve the distance matrix and directions to the target location.

Account Details Screen
The account details screen displays all data about a user's account, such as that in firebase authentication or in accounts section of the database. When a user clicks the button that enables dark mode in the app, the *darkMode* variable in the database will be changed. When a user selects to change their password, that updated password is sent to firebase authentication to change to the new password.

History Screen
The history screen displays previous trip data from a user. Once a user enters this screen, the list of trips associated with a signed in user is retrieved, and the first ten elements of that list are used. In the table itself, the *startDate*, *distance*, and *endLocation* are displayed providing a user an outline of when the trip was, how far it was, and where the trip ended up.

6.0 Testing

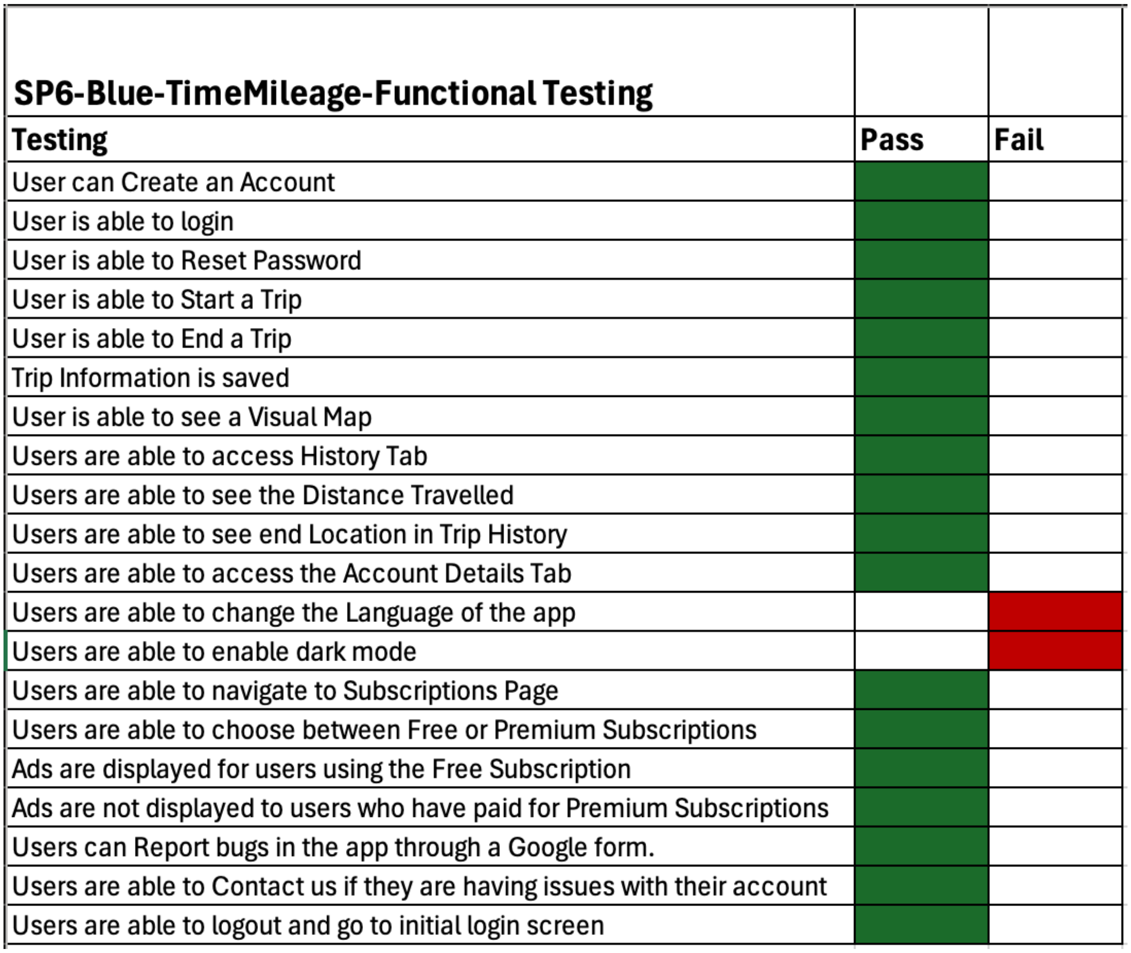
This section of the document will explain what functional requirements were tested. It will also outline which one of our requirements either passed or failed during testing. There will be a table that shows the performance of the functional requirement and below that table there will be an explanation on why the test failed and what we are doing to try and fix the issue. Testing is needed to ensure that the user has an enjoyable experience using the app and that they do not run into issues while using the application.

6.1 Test Plan

To test the application, we would need to check if the functional requirements are working and if users would find any issues while using the app. The functional requirements that we would need to test for are.

- (3.1.1.a) User can Create an Account?
- (3.1.1.b) User is able to login?
- (3.1.1.c) User can Reset Password?
- (3.1.1.d) User can Start a Trip?
- (3.1.1.d) User is able to end a Trip?
- (3.1.1.e) Trip information is saved?
- (3.1.1.h) User can see a Visual Map?
- (3.1.1.f) Users are able to access history Tab?
- (3.1.1.f) Users are able to see the distance?
- (3.1.1.f) Users are able to see the end Location in trip history?
- (3.1.1.j) Users can access the Account Details Tab?
- (3.1.1.j) Users are able to change the Language of the app?
- (3.1.1.j) Users are able to enable dark mode?
- (3.1.1.j) Users are able to navigate to subscriptions page?
- (3.1.1.j) Users are able to choose between free or premium subscriptions?
- (3.1.1.k) Ads are displayed for users using the free subscription?
- (3.1.1.k) Ads are not displayed to users who have paid for premium subscriptions?
- (3.1.1.i) Users can Report bugs in the app through a Google form?
- (3.1.1.i) Users can Contact us if they are having issues with their account?
- (3.1.1.l) Users can logout and go to the initial login screen?

For now, we have tested the functional requirements for phase 1. In phase 2 we plan on implementing and fixing the tests that have failed. For example, we are still working on enabling dark mode to the application. In the test report the dark mode test failed so it’s a functional requirement that we are still working on implementing.



SP6-Blue-TimeMileage-Functional Testing		
Testing	**Pass**	**Fail**
User can Create an Account	🟩	
User is able to login	🟩	
User is able to Reset Password	🟩	
User is able to Start a Trip	🟩	
User is able to End a Trip	🟩	
Trip Information is saved	🟩	
User is able to see a Visual Map	🟩	
Users are able to access History Tab	🟩	
Users are able to see the Distance Travelled	🟩	
Users are able to see end Location in Trip History	🟩	
Users are able to access the Account Details Tab	🟩	
Users are able to change the Language of the app		🟥
Users are able to enable dark mode		🟥
Users are able to navigate to Subscriptions Page	🟩	
Users are able to choose between Free or Premium Subscriptions	🟩	
Ads are displayed for users using the Free Subscription	🟩	
Ads are not displayed to users who have paid for Premium Subscriptions	🟩	
Users can Report bugs in the app through a Google form.	🟩	
Users are able to Contact us if they are having issues with their account	🟩	
Users are able to logout and go to initial login screen	🟩	

Figure 6.2: Test Report

6.2 Test Report

In the table above, *figure 6.2* you can see that some of our tests failed. These tests are still being worked on and we plan on implementing them during phase 2. For now, most of the functional requirements can be used by the user. The user should be able to comfortably use the app.

The user should also be able to click on the tabs displayed at the bottom of the home page. Such as the Home, History, and Account Detail tabs. The Home tab will let you start the trip and end the trip while displaying a visual map to the user. In the History tab users can see their trip history and they will be able to see the date, distance, and end location of their trip history. In the Account detail tab users cannot enable dark mode or change the app's language, but they can click on the subscription plan button and navigate to the subscription page to choose the type of subscription the user would want.

The Account details tab also lets the user report bugs and contact us. They will also be able to change their username, password, and email from the Account details tab. Also, all these tabs in the top corner have a logout icon that navigates the user back to the initial start screen.

7.0 Version Control

While developing the application, GitHub was used for the version control of the app. GitHub allowed us to create, store, and change the code or files for the application. GitHub also displays the history of the development of the app. It shows who worked on what parts of the application, and when they pushed the changes or developments.

Alongside GitHub, we used GitHub desktop, which is an open-source application that works with code hosted on GitHub. GitHub desktop allows the user to perform git commands in a graphical user interface, rather than using the command line. GitHub desktop also shows the changes that have been added to the code depending on which branch you are in. If there is a need to go back to the previous code and undo code, GitHub allows you to look back at the previous code and input it again.

GitHub allowed us to use branches. Branches are copies of the main file code and it allows collaborators to work on the application without messing up the main files code as it is getting developed. Branches were used to implement features to our app, such as ads, routing, and settings. The changes being made in the branches could be merged into the main branch using pull requests. Once the pull request is sent, reviews can be requested. Once the reviewers look over the code, they can approve it or make requests for changes. If the code is deemed acceptable, then it can be merged into the main branch. Collaborators would be able to pull the new changes in the main branch when they fetch the origin from GitHub desktop.

The downside to using GitHub was that when we pushed a file. We needed to remove the API key for the Google Maps API from the file before pushing the changes. If the API key wasn't removed, then it would be leaked, thus potentially costing us a lot of money if someone got a hold on the API key.

8.0 Conclusion

SP-6 Blue's goals in developing this mobile application are to help users keep track of their travel history between various locations. While the user is travelling to their destination, the application will track when the user started driving, where they went, and what the distance was. This information would be displayed in the trip history tab and users can use that information during the tax season to get write-offs for their overall commutes.

Overall, our goal within creating our mobile app, is to give users the opportunity to better analyze their commutes and have the option to have a higher return within their taxes for the previous year. All the while giving them a more ideal and smoother experience, in comparison to the other existing mapping mobile applications currently.

APPENDICES

Appendix A – Figma App Skeleton

Figma Mockup for App Skeleton

Appendix B – Project Plan

Project SP-6 Blue Time Mileage Location Tracker

Enhanced Overview

We are designing an app that will track a person's location from two points. While the app is running there will be data post-processed based on the information retrieved from google maps API that will provide data such as how far they went, their average speed, and route taken on the trip. The app will be available on all platforms and will have a two-tier system.

The base tier will provide basic information about trips such as the distance, average speed, time, start location, and stop location. This base version will also contain the ability to view the history of trips but will contain ads located both on the home page and history page.

The second tier will be a paid version of the app that will contain no ads across any page of the app. This tier will also contain more features such as more detail for trips taken such as gps tagging and the ability to view up to the past 5 years of trips taken.

Both tiers will contain features that will allow the user to track the location between two points, download a excel file for tax purposes around tax season, see information about previous trips, change basic information about account (contact email, password, etc...).

Website

https://sp-6-blue-timemileageapp.github.io/TimeMileageApp/

Deliverables

Deliverables include:

a. Application Design Mock-up (Figma)
b. Working Application Prototype (Figma)
c. User Flow Documentation (Admin/User)
d. Tutorial Documentation (In-App or Separate)
e. Testing Reports
f. Database (User Information)
g. Database Design Document
h. Requirements Documentation
i. Project Plan Documents
j. Source code files (zip file)

Milestone Events

Milestone #1:

Requirements Documentation | Date: 2/13/2024

- List of features that will be added to the application (With priority in case not all features can be added in time)
- Software/Hardware requirements to build/use the system (Emulators, languages, database systems, APIs, Device hardware, etc.)
- Permissions Requirements

Design Documentation | Date: 2/13/2024

- Outline platforms used for application development
- Provide technical documents specifying database design
- User interface design visual mockup and flow chart
- Outline of plan for monetization

Milestone #2:

Working Application Prototype | Date: 3/10/2024

- User will be able to interact with the application and experience the base level features offered within the application
- Advertisements will be displayed, or Advertising spaces will be indicated
- User will be able to create an account and sign in and out of the application
- User will be able to see basic travel statistics for a set period of time (distance, time, speed, etc.)

Database System | Date: 2/13/2024

- Database will hold user information as well as mileage tracking history
- Tables that must be implemented: **Meeting Schedule Date/Time**
-
    - USERS table which holds UserName, Password, Email, Status (Standard, Premium), etc.
    - TRIPS table which stores trip information such as Distance, Time, Speed, Location Set, etc.
    - Username field will be the key linking TRIPS to USERS

Milestone #3:

Finished Project w/ Monetization| Date: 4/29/2024

- Documentation used throughout project
- Final Working Application

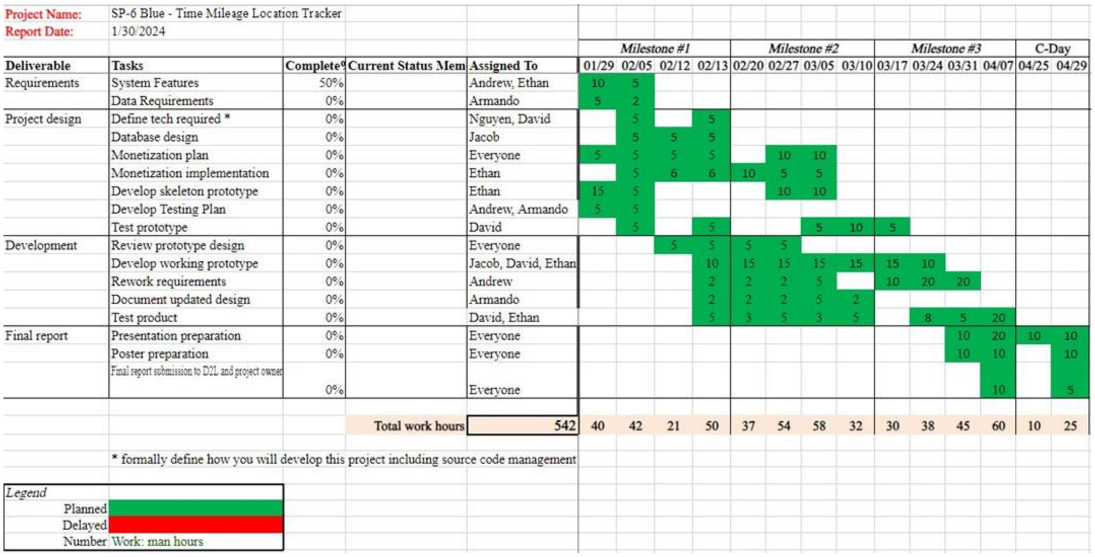
- Source Code
- Descriptions outlining the system design
- Video Demonstration of Application

Meetings are from 5:00-6:00 M/W in person if class is taking place, otherwise we will meet in Microsoft Teams. We also plan to meet on Fridays between 3:30-4:45 on Teams if necessary.

Collaboration and Communication Plan

For our project we are planning on using GroupMe for our main source of communication, but can also communicate by phone, email, or Microsoft teams. If we are unable to meet, we will join an online video call through Microsoft teams or discuss another day that works for the group besides Monday, Wednesday, or Friday. When it comes to sharing files, we will use Teams. For version control we will be using GitHub. Armando will take meeting notes and distribute them to us through Teams. We will give weekly status updates every Monday saying what we worked on last week, what we are planning on working on this week, and if we had any issues with anything. ￼

Project Schedule and Task Planning (GANTT CHART)



Project Name: SP-6 Blue - Time Mileage Location Tracker
Report Date: 1/30/2024

					Milestone #1				*Milestone #2*				*Milestone #3*				C-Day	
Deliverable	**Tasks**	**Complete%**	**Current Status Mem**	**Assigned To**	**01/29**	**02/05**	**02/12**	**02/13**	**02/20**	**02/27**	**03/05**	**03/10**	**03/17**	**03/24**	**03/31**	**04/07**	**04/25**	**04/29**
Requirements	System Features	50%		Andrew, Ethan	10	5												
	Data Requirements	0%		Armando	5	2												
Project design	Define tech required *	0%		Nguyen, David		5		5										
	Database design	0%		Jacob		5	5	5										
	Monetization plan	0%		Everyone	5	5	5	5		10	10							
	Monetization implementation	0%		Ethan		5	6	6	10	5	5							
	Develop skeleton prototype	0%		Ethan	15	5				10	10							
	Develop Testing Plan	0%		Andrew, Armando	5	5												
	Test prototype	0%		David		5		5			5	10	5					
Development	Review prototype design	0%		Everyone			5	5	5	5								
	Develop working prototype	0%		Jacob, David, Ethan				10	15	15	15	15	15	10				
	Rework requirements	0%		Andrew				2	2	2	5		10	20	20			
	Document updated design	0%		Armando				2	2	2	5	2						
	Test product	0%		David, Ethan				5	3	5	3	5		8	5	20		
Final report	Presentation preparation	0%		Everyone											10	20	10	10
	Poster preparation	0%		Everyone											10	10		10
	Final report submission to D2L and project owner	0%		Everyone												10		5
			Total work hours	542	40	42	21	50	37	54	58	32	30	38	45	60	10	25

* formally define how you will develop this project including source code management

Legend	
Planned	🟩
Delayed	🟥
Number	Work: man hours

Security:

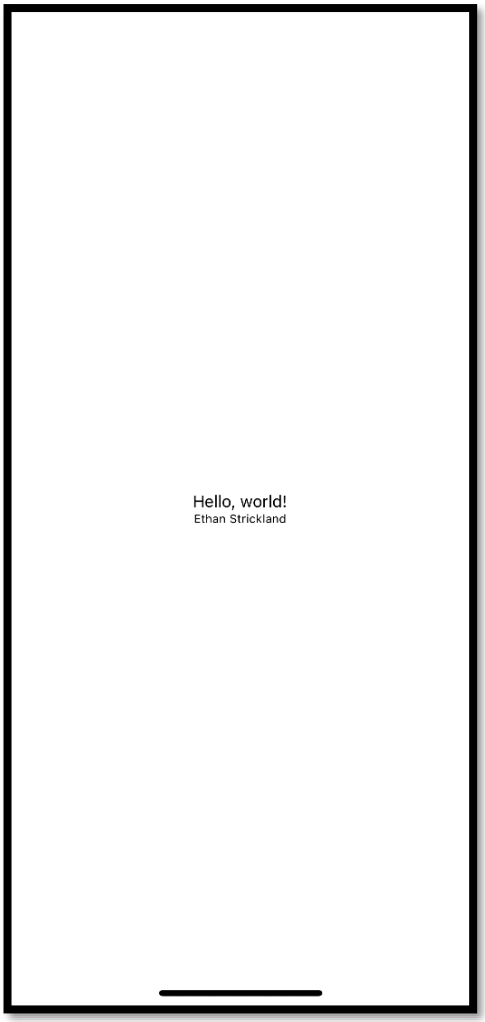
Here are a few things that will be considered for security when developing this app

- Have MFA setup when logging into the app
- Encrypt user's passwords when stored in the database

Version Control Plan:

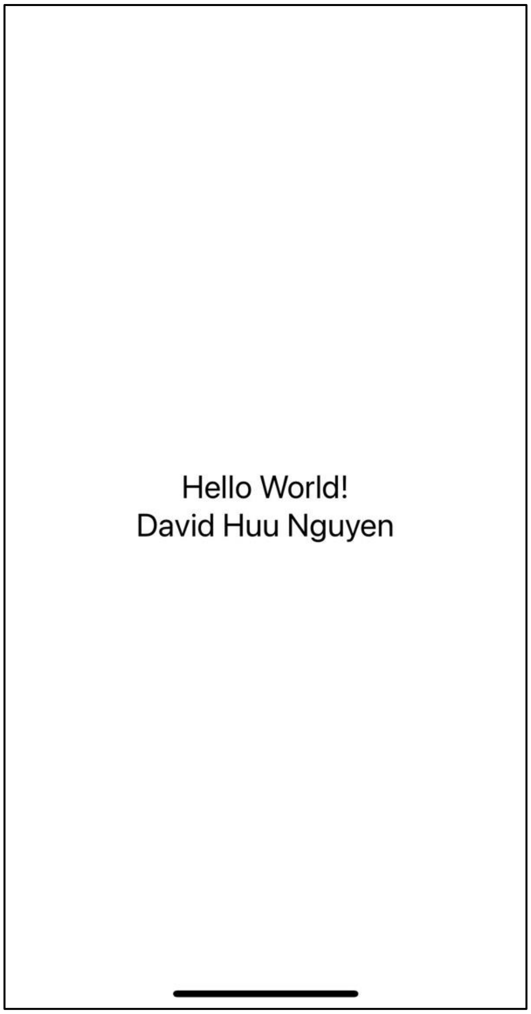
We are planning on using a GitHub Actions organization to help with version control and make sure everyone is on the same page when working on this project.

Appendix C React Native Tutorial



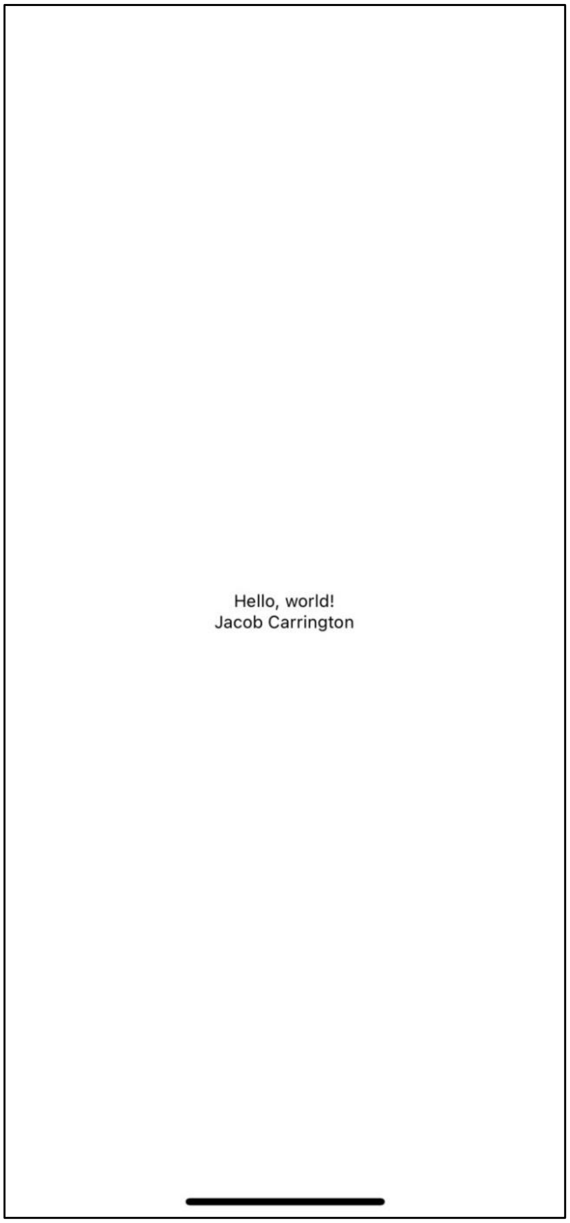
Hello, world!
Ethan Strickland

React Native Tutorial - Ethan Strickland



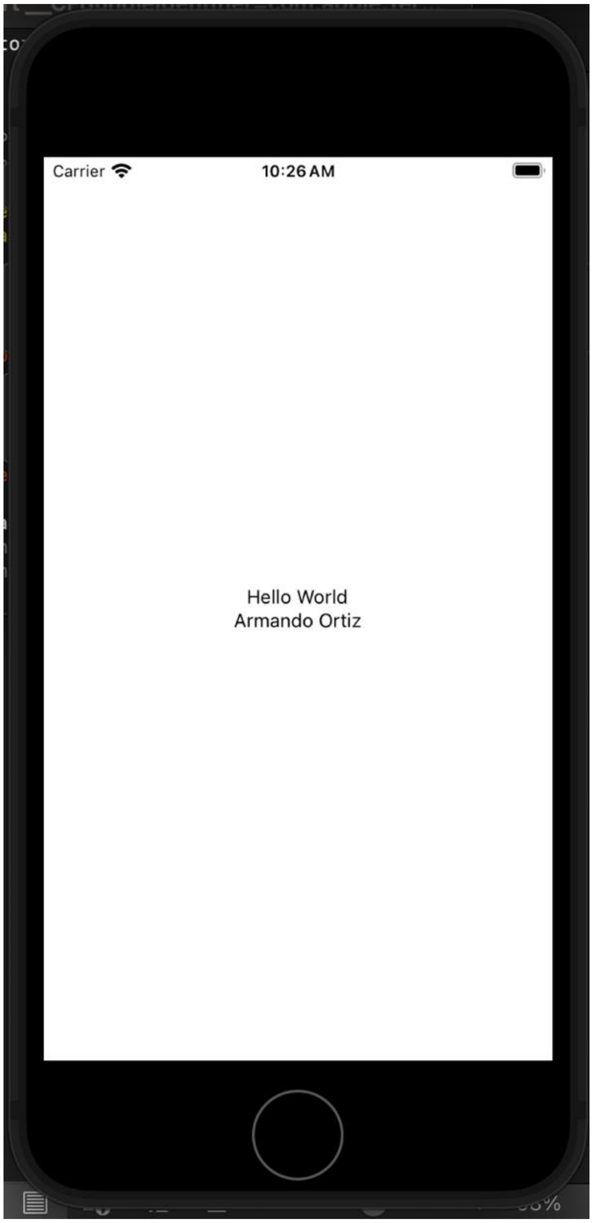
Hello World!
David Huu Nguyen

React Native Tutorial - David Nguyen



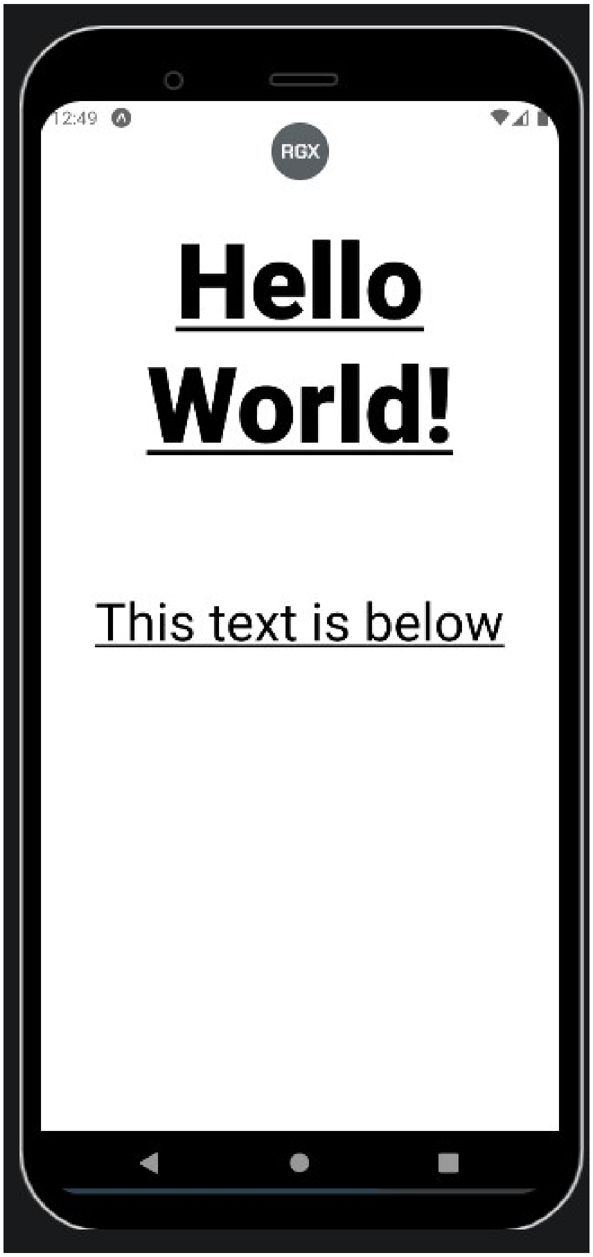
Hello, world!
Jacob Carrington

React Native Tutorial - Jacob Carrington



Hello World
Armando Ortiz

React Native Tutorial - Armando Ortiz



Hello World!

This text is below