CS 4850 Section 02 Spring 2024

Indy-1 Fitness App

Professor Perry

Michael Bolnik, Adam Borowski, Luke Zeches

April 27, 2024

Website: https://indy01-fitnessapp.github.io/

GitHub: https://github.com/INDY01-FitnessApp

Lines of Code in our Project: 16712

Number of Components in our Project: 6

Table of Contents

1. Introduction 4
    1.1 Overview 4
    1.2 Project Goals 4
2. Design Constraints 4
    2.1 Environment 5
    2.2 Project Scope 5
3. Functional Requirements 6
    3.1 Account creation screen 6
    3.2 Login screen 6
    3.3 Allow user to create new trip/continue trip 6
    3.4 Track and plot user’s movement 7
    3.5 User view of current trip and progress 7
    3.6 User view of profile information 7
4. Non-Functional Requirements 7
    4.1 Usability 7
    4.2 Security 8
5. External Interface Requirements 9
    5.1 User Interface Requirements 9
    5.2 Software Interface Requirements 10
6. Narrative Discussion 10
    6.1 Purpose 10
    6.2 System Overview 10
    6.3 Goals and Guidelines 11
    6.4 Development Methods 11
    6.5 Technical Design 12
        6.5.1 Assumptions and Dependencies 12
        6.5.2 General Constraints 12
        6.5.3 Policies and Tactics 13
        6.5.4 Architectural Strategies 13
        6.5.5 System Architecture 14

6.6 Detailed System Design 15
6.6.1 Screen Mockups 15
Figure 1: Sign-In Page 15
Figure 2: Sign-Up Page 16
Figure 3: Reset Password Screen 17
Figure 4: New Trip Screen 18
Figure 5: Continue Trip Screen 19
Figure 6: Sidebar Display 20
Figure 7: Location Menu 21
Figure 8: Map Screen 22
Figure 9: Profile Screen 23
6.6.2 User experience system 24
6.6.3 System design 25
7. Development 27
7.1 Analysis of Technologies 27
7.2 Issues Encountered 28
8. Application Testing 28
8.1 UI/UX Testing 29
8.2 Functional Testing 29
8.3 Results/Reporting 30
9. Source Code Version Control 30
10. Future Work 31
11. Final Summary/Conclusion 32
12. Appendix 32
12.1 React Native Tutorial Completion 32

1. Introduction

1.1 Overview

With our project, we plan to make a fitness app that makes exercising fun and engaging. While users are on the app, it will use a GPS system to track their location. With this information, our app will count the distance traveled and save it into a database. To gamify this system, there will be multiple destinations that one can virtually travel to. For example, if a user wished to simulate a trip across the United States, our app would track the distance traveled and show how far along they were. While on a trip, the app will show your current location on the path using the Google Routes API. Every user will also be able to see their total distance traveled and the number of destinations reached. For security reasons, every user will have to create an account with a username and password and must login when opening the app. Our app will be built for Android devices using React Native, to allow for future cross-platform development. It will start with a simple UI and a few separate locations to visit but can be updated over time to add more destinations to track.

1.2 Project Goals

Many people do not get as much exercise as they should, due to time constraints or lack of motivation. We hope to solve these issues with our app. By giving users goals to work toward while exercising, people will feel more incentivized to walk or run in their daily lives. There are smaller goals that will contribute to our main goal as well. These include the aims of interface design and systems. The systems we plan to build will make it easy for users to begin a trip and track their progress which will push them to use our app more often. Our user interface will play into this too. The design will make our app easy to use at a glance with few learning curves. More details on the user interface will be discussed in a section below.

2. Design Constraints

The design of our project is formulated under a few set constraints that we will work around and use as a baseline. We will need to implement a few tools in order for our functionality to work properly and give us the full systems scope that we are looking to put together. The use of the React Native library will give us a lot of the front-end functionalities needed in order to implement all the important features that we will be displaying in our app. We will have a handful of functions and classes that will help us to build our user interface and be constrained to the contents of the library.

In terms of possible options for our database implementation, one option would involve using Firebase in order to implement our database functionalities and store the user data that is needed. The option to add additional security to our database through the form of encryption is also possible in order to safehouse any data that might need to be kept confidential through any agreements that are made with customers. Firebase Authentication also supports creating users and securely storing their login information. In order to sync the database to the app, we would most likely have to configure the database to the app through the use of functionalities provided by Firebase. The other option for our database constraints would be to use the built-in React Native functionality for creating a local database which would allow for an alternative form of data storage built into the framework. In terms of language constraints, we would be using JavaScript, with our runtime environment being Node.js.

The use of the Google Routes API is also crucial to our design constraints as we will be constrained to the functionalities that are provided by it. We will also be constrained by the data that we can receive from the mobile device as the information being sent will be sent in a certain format and specification that will need to be adapted to by our own app. In terms of any corporate or regulatory policy restraints, we would need to work around any issues regarding the usage of data in terms of both fitness and personal data and make sure to keep that data secure and inform the user what data is necessary for our app to provide the full functionality that is applicable.

2.1 Environment

Our app will be developed for Android version 13.0. The platform we will use to code our app is React Native v. 0.73.4, the latest long-term support version. Front-end and back-end development will be done in Android Studio using an emulator to perform testing. We will use Expo Go as a simulated environment to run our code and test certain functionalities in order to make changes. Version control will be done using Git, with the remote repository hosted on GitHub.

2.2 Project Scope

Our project will integrate multiple systems in order to accomplish our goals. For our first release we plan to have the basic requirements such as a login screen, account settings, and the ability to start a trip. This version would track the user’s distance traveled and use this information to determine how far they have traveled along the planned trip’s route.

Our final release includes a database that stores user information about their profile, current trip, and any completed trips. This includes the total distance traveled and the number of trips completed. We will update the trip screen to show a map of the entire trip along with the user’s location on it. Google Routes will be integrated and have the best path drawn out from one point to another on the map that is displayed to the user. Multiple libraries will be implemented in order to draw the correct polyline over the routed path created by the API, with an obvious display of the progress that the user has made at a certain point in the trip.

3. Functional Requirements

3.1 Account creation screen

3.1.1 Description: allows the user to create a personal account to store their data. The user can choose to log in to an existing account if they already have one. The provided data is used to create an account in the database.

3.1.2 The following information will be provided by the user:

3.1.2.1 First and last name

3.1.2.2 Email address

3.1.2.3 Username

3.1.2.4 Password

3.2 Login screen

3.2.1 Description: allows the user to enter a username and password to verify their identity and gain access to their account. The user can choose to create an account if they don’t already have one.

3.3 Allow user to create new trip/continue trip

3.3.1 Description: users can start a trip if they aren’t on one already. If the user is currently on a trip, they can press a button to continue their progress.

3.4 Track and plot user’s movement

3.4.1 Description: our app will track how far a user traveled during a current time frame. This information will be used to determine their progress on a current trip.

3.5 User view of current trip and progress

3.5.1 Description: users will be able to view information about their current trip. This includes how far they’ve traveled on it and elapsed time.

3.6 User view of profile information

3.7.1 Description: the user will be able to view their personal information associated with their profile, as well as statistics about their app usage. These statistics include information about previous trips completed and total distance traveled.

3.7.2 The user will be able to view the following information about their previously completed trips:

3.7.2.1 Start and end location
3.7.2.2 Start and end date
3.7.2.3 Total distance traveled
3.7.2.3 Total time spent exercising

4. Non-Functional Requirements

4.1 Usability

In terms of usability, there will be multiple requirements that are implemented in order for the user to have an easy and efficient experience interacting with our app. We will look to have a very simple log-in screen that looks to verify the username and password information in order to link to the account data. The opening screen will be devoid of any unnecessary information and be free of any distracting graphics with a simple, sleek look. The simplicity of the opening screen will also promote error avoidance as there will not be many ways for the user to cause an error, outside of possible issues in input information that will be dealt with by error handling in our software. Maneuverability will be simple with the only option from the log-in screen being to hit a button that will allow for the movement to the next screen.

There will also be an option for creating an account which will require the user to fill in details that will allow for the successful verification of the account. This screen will be extremely simple to use with bars to fill in for the information that we require to be filled out. This will end in with a simple create account button which will store the information and bring the user back to the log-in screen so they can go ahead and log in. There will be some built-in functionality to detect errors in the user input data so that all the logged information is valid for the input fields.

For the trip selection screen, there will only be simple options that can be chosen so that the user will not have difficulty in understanding what each option is. The buttons will be displayed in a fashion where they will be very visible, and a simple click will take the user to the destination they prefer. This screen will not have to deal with any error detection as the only options are to click one of the options listed and choose what to do. This will also promote efficiency and accessibility with quick options that can be chosen without having to search for them on the screen. In terms of the actual trip screen, it will be a little more complex in design as we will need to have multiple displays of data occurring on the screen at one time in set locations on the interface. This will be a slight learning curve for a new user but will be relatively simple to where the user will be able to memorize and grow comfortable with the interface design after a few attempts of usage. There will also be a button that will need to be pressed in order to start the trip and end the trip that will allow for the user to simply begin and end their trip. The accessibility and efficiency of the interface for this system will be relatively simple in regard to the functionalities that the user shall be accessing as the interface will provide a rather easy to use screen that will allow for the quick action of what this system is designed to do.

The usability of the statistics page will also be rather simple as it will provide a quick report of certain stats that describe the user’s fitness journey recorded in the app. The main practices that we looked to invoke in our project were keeping the interface as simple as possible and placing the users in control of the interface by giving easy to use tools and simple accessibility to the functionalities of the system.

4.2 Security

The main security risks are the user’s login credentials, and the user’s personal data that is stored as a part of their profile. This information is most at risk when it is being sent between the client and server. The use of Firebase's authentication functionalities will make sure that the user's information stays private and cannot be accessed by any malicious parties. Also, in any case where fitness data is being used, the application will require permission from the user.

5. External Interface Requirements

5.1 User Interface Requirements

Our user interface design allows users to navigate our app quickly and easily with little confusion. Starting with the login screen, it prompts users to enter their username and password with two textboxes or create a new account if they don't already have one. The account creation screen includes text boxes for first name, last name, username, password, and email address. Users can go to the reset password screen by tapping the "forgot password" button. This allows them to reset their password if they forgot it or it was stolen.

The home screen shows information about the current trip such as progress and total distance traveled so far. If no trip has been started it will prompt you to start one. On the home screen, a side menu will contain buttons to go to various other screens. The menu button will start minimized and can be expanded by tapping on it. Inside this menu will be multiple buttons that allow that user to be redirected to other screens where different functionalities can be accessed including profile information or other relevant screen functions that are required in our app.

Current trip details will show the progress on a current trip on a map along with other details. These details will be accessed by scrolling down in the map screen and locating a field that shows the variables that we included as being recorded under the current trip. When a new trip is started, a list of possible trips will be shown as well as a search bar and a filter for popular trips. This will allow the user to choose the starting and destination points and create the trip that the user is wishing to embark upon during their current exercise session. Overall, the goal is to keep the layout consistent and simple while being able to contain all these details in an easy-to-see manner. All UI elements use neutral colors and similar shapes in order to maintain consistency and eliminate any distracting elements that could cause issues for the user in terms of ease of use of the app. Every screen will be designed to be easy to find. Certain elements such as the trip view screen are accessible via multiple methods.

5.2 Software Interface Requirements

5.2.1 React Native

5.2.2 React Native Maps

5.2.3 Google Routes API

These were the three main components that were accessed and used in order to implement the interface of our app in the way that we wished too. React Native provided us with the ability to craft internal components of every screen in order for all of our external displays to appear to the user and develop our user interface. React Native Maps was essential for providing us with a map component so that we could display our map to the user. The Google Routes API allowed us to generate a routed line on our map is capable of using current barriers of access in terms of getting from one point to anther and determining the distances and estimated times of arrival from matrices of starting to endpoint combinations. This gives us the best route from point A to point B and allows us to display this path to the user, while giving us the route that the user's distance metrics will be mapped upon.

6. Narrative Discussion

6.1 Purpose

The purpose of this section is to explain the design approach that we took with developing our app. We wanted our design to be relatively simple while also appealing to the eyes of any users. Our design methodologies and decisions are explored in this section and a general look of the systems interactions is explained and described. This section takes our earlier requirements and design introduction and focuses more on how the systems that enforce those requirements and design work together in order to display the visual product that we display to our user. This section is a general mapping and layout overview of our app's design.

6.2 System Overview

Our app will use several systems to ensure correct functionality. This application tracks the distance traveled, so our program will only need to fetch the distance information. This functionality will only work when the user is currently on a trip. These distance metrics will be used to update the current trip while users have the app open. To ensure no data leaks, we will use Firebase's security functionalities to securely store a user's password and email. This means that passwords are not stored in the database, but instead the Firebase Authentication system. The security system will be in effect while a user logs into their account or creates a new one.

The next system included will be a database that holds data about each user. The database is implemented with Firebase Realtime Database, which allows for real-time updates. Included in the database is a user information section, current trip section, and a list of completed trips. When a new account is created, an entry in user and current trip is made. Completed trips are only added once a trip is finished. Each user is identified with a unique user ID which allows us to search for data based on the current user. This also ensures that each user has a unique identifier.

The last system included is our mapping system using the Google Routes API. Once selected, the app will redirect the user to a new interface that will display a map that will have an already mapped out route from a certain location to the destination location that the user chose in a selection menu. This interface will allow the user to see their location updated on the path as the distance they are traveling is mapped out on the screen. Once these systems are in place, the app will have full functionality.

6.3 Goals and Guidelines

Our goal for the design of the app was to keep it simple yet functional. This means that there is no learning curve for using the app and that the functions of each design element are immediately recognizable. We also want to focus on making sure that our app does not suffer from any possible overuse of memory that might have brought down the performance of our app, so we looked to deploy memory management practices for any system that required storage of data. The last goal was to make sure that the design of our app was simple with a sleek look that had only the necessary information that the user needed to maneuver through the app. We want the app to have the feel of a regular fitness app with the addition of a couple of custom features which will display the app's originality. We hope to develop this app without necessarily innovating in the main features of fitness apps but rather innovate in the possibility of extending outside of the usual realm of data by providing the user with a different type of experience and interfacing method.

6.4 Development Methods

Since we are developing our app to be simple and functional, we aim to make the UI easy for anyone to understand. This guideline is true for the design of our systems too. The app is meant for tracking exercise, so this should be immediately clear for users. We don’t want to waste any of the user’s time navigating menus or setting up accounts, so starting and viewing a trip will be quick and simple. We also plan to make our app feel unique but still have a familiar foundation as other fitness apps. There are countless other apps which track your steps, so we want ours to stand out. The integration of the Google Routes API and mapping out the route that is tracked by our distance tracker will provide a way to make our app unique.

6.5 Technical Design

6.5.1 Assumptions and Dependencies

The operating system that our app will run on is Android. For this reason, any APIs we use will need to be compatible with this OS. This includes the distance tracking system, the database we use, and the security system. Our group considered developing for Android and IOS together, but IOS was deemed too difficult as it's not open source.

At first, we decided to use Expo SQLite as it was a lightweight database used for many JavaScript applications. However, we realized that it didn’t work in real-time, so we switched to Firebase which does support this functionality.

6.5.2 General Constraints

Since our app is being developed for mobile devices, there are certain constraints we must follow. The first of these involves the GUI. All interface design should be compatible with the 9:16 aspect ratio of a phone and be readable on small screens. Mobile devices impart other restrictions not seen in desktop applications. The app is built to be used outside, so we need to ensure that it doesn’t consume too much battery life while running. Storage space is also a factor since phones don’t have as much space as a desktop computer. If the app uses more space than necessary, it could be difficult for users to download. Performance is another concern. While running the app, it shouldn’t be slow or stutter. Distance will be updated every few seconds, but our goal is to make movement on our map component as smooth as possible. Connection to a network must also be fast and secure. Experiencing connection issues while using the app would ruin the user experience.

6.5.3 Policies and Tactics

In the beginning stages we discussed how our trip system would work. There were two possible ways to implement this feature; either a predetermined trip would be set, or users would begin a trip from their current location. We decided that creating predetermined trips would work better since it would be much less complicated to implement. We made the decision to code our app using Android Studio and use React Native as the framework. With these tools, we plan to write our code so it's easy to read and debug. As mentioned above, we used a database to store user information. In this database, usernames, passwords, and emails will be used as primary keys which will identify users from each other. Data such as distance and elapsed time will be attributes that users can retrieve from their account settings. To secure this database, all passwords will be encrypted using Firebase's authentication services. Passwords won't be encrypted in the database, but the token which sends the password from the user to the database will be. This is to stop anyone who tries to steal passwords while they're transmitted from the app.

6.5.4 Architectural Strategies

One of the first architectural strategies that we employed started with the framework that we chose to work with. We decided to go with React Native due to its purpose of developing apps, more specifically Android apps, so that we had all the tools and methods we needed to develop the app in the scope that we wanted to. We are also constrained to using JavaScript as this is the language that is used when working with the React framework. These options were chosen in order to maintain a relatively simple approach to developing the app and making sure that all the components were compatible.

In terms of reuse of features, our design was relatively simple and straightforward in the first place but there were different areas where information such as personal information or statistics could be reused for different output results on the interfaces that included relative information. This software will have much room for improvements and extension as the baseline model will simply look like an ordinary fitness app with some custom features and the map view while further extensions could involve systems such as new map possibilities where the user could find themselves in virtual spaces rather than real life locations or we could look to incorporate further functional capabilities such as a system where the app will allow you the option to block certain sites or apps such as YouTube or Instagram until a your preset exercise goal is complete.

In terms of any user interface paradigms, the main input our systems take entails customer information and distance information that will be gained from the health app that we draw our data from. There will also be a section of our system that is responsible for recording fitness input data that will be important for calculating any statistics that we wish to include as well as to link the input data to the mapped-out interface that we will show the path on. Error detection will mainly be used to ensure that the user only follows the possible actions that will be displayed on our interfaces with certain functions to make sure that issues such as validity of enter information and possible issues with the input data to map translations occur without any problems that could crash or severely harm the functioning of our systems. We will make sure to efficiently use memory by reusing any objects that can be used in multiple areas as well as optimizing our user interfaces through the use of simple layouts that don't lead to any complicated view hierarchies.

6.5.5 System Architecture

Our app will be broken down into several systems and subsystems. The most important system that will run through our app will be the distance tracker. By taking the distance information collected, we store this information and use it as a metric for mapping out the distance that we want to be recorded through the polyline that is being generated. When the user travels a certain distance, their view will update on the map as their distance traveled will be routed onto the map through the generation of the polyline. This system will only be in effect once the user allows system access to their location. Another important system will be the application of the Google Routes API. It is responsible for routing a coordinate path from one location to another location which will be chosen by the user, and making sure that the path is the shortest possible walking path from one point to the other. This resource will be triggered by the creation of a trip and used once the starting and destination location variables have been populated so that they can be pulled, and a path can be generated. The last major system will be a database which stores user data. Each user will be able to access their account information through the app and all of the data will be stored in a secure database. This database will also be updated as users accumulate more miles while using the app. Current trip information will also be stored in the database. This included the origin name, destination name, coordinates, total distance of the trip, and how far the user has traveled on the trip. When the user is exercising, the distance and time is updated. These updates are sent to the database when the current session is finished. If the current distance is equal or greater than the total distance, the trip is completed. In this occurrence, the counter for number of trips completed goes up and the current trip information is cleared. All the information for the completed trip, such as the total time it took, will be stored in the list of completed trips. This list is unique for all users.

6.6 Detailed System Design

6.6.1 Screen Mockups

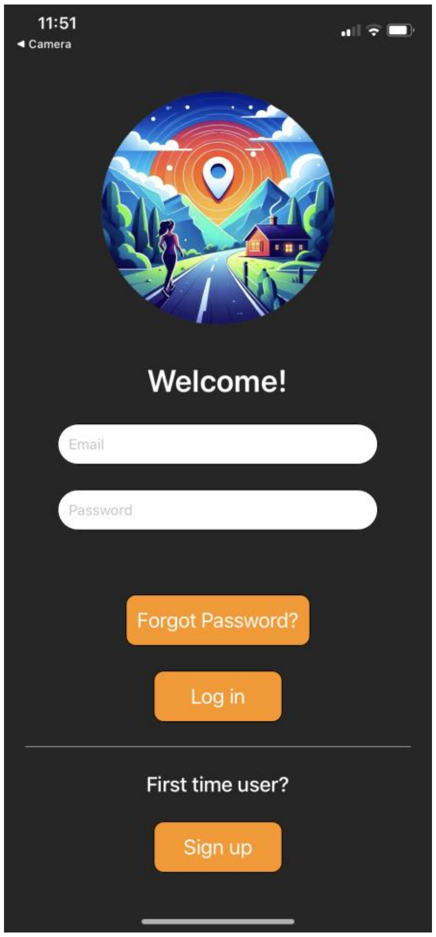


Figure 1: Sign-In Page

This is the sign-in page for our app. It consists of multiple fields that the user can access in order to navigate to different pages or authenticate a certain action. The design incorporates the use of rounded text boxes for the username and password fields as well as rounded buttons for the log-in and sign-up buttons. We also have welcome text to politely introduce our clients to our app as well as text that signifies the location for first time users to correctly navigate to the sign-up screen so they can properly log-in and not be blocked from access to our app. The log-in button allows for the user to head to our home screen once credentials have been verified, while the sign-up button redirects the user to a sign-up screen where the user can enter their information so it can be properly added to our database. The forgot password also allows the user to change their password in the case that they have forgotten theirs, which was a functionality we realized we should probably add during our testing sessions.

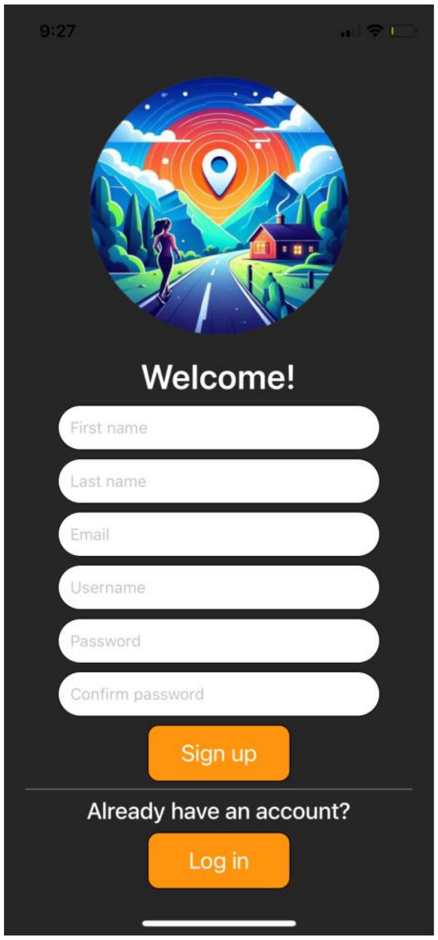


Figure 2: Sign-Up Page

The sign-up screen was designed to collect all the necessary information that we need in order to provide the user with a unique account that will be recognizable through the log-in credentials. We looked to get and set up important information such as name, email, username, and password as well as a confirm password to make sure that there was an option to verify the password the user wants to set. This screen was relatively simple and had one main function in storing all the data that was being entered by the user and then having a signup button that would push the creation of that account into our database.

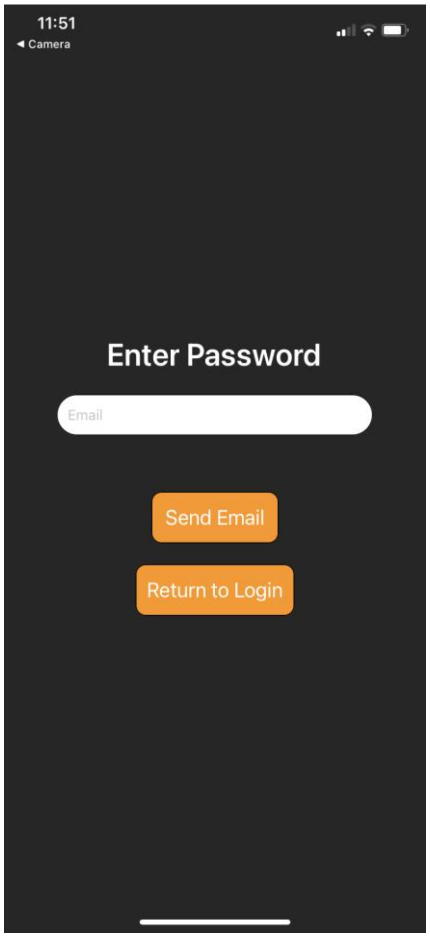


Figure 3: Reset Password Screen

From the login screen, users can navigate to the reset password screen. Here, the user can enter in their email in order to reset their password. There are only two buttons; one to submit your email and one to return to the login screen. Once an email is entered, Firebase Authentication is able to send an email which will prompt the user to enter in a new password. After the new password is submitted, the user can go back to the app and return to the login screen. Their new password will now be set.

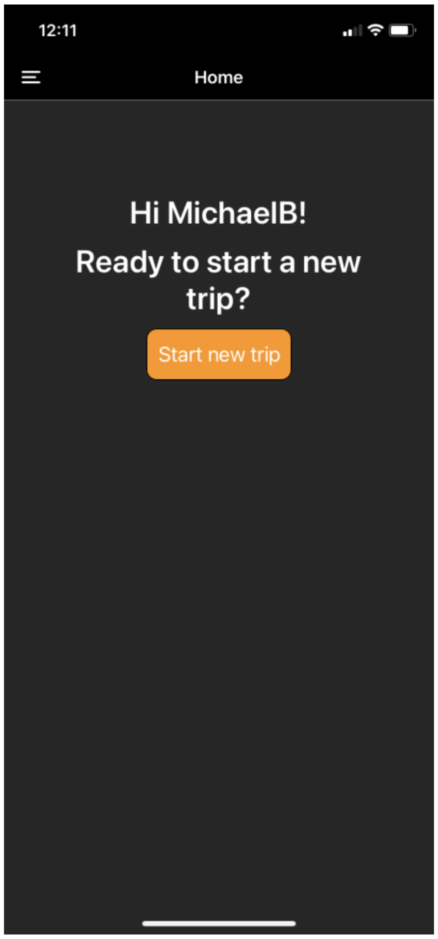
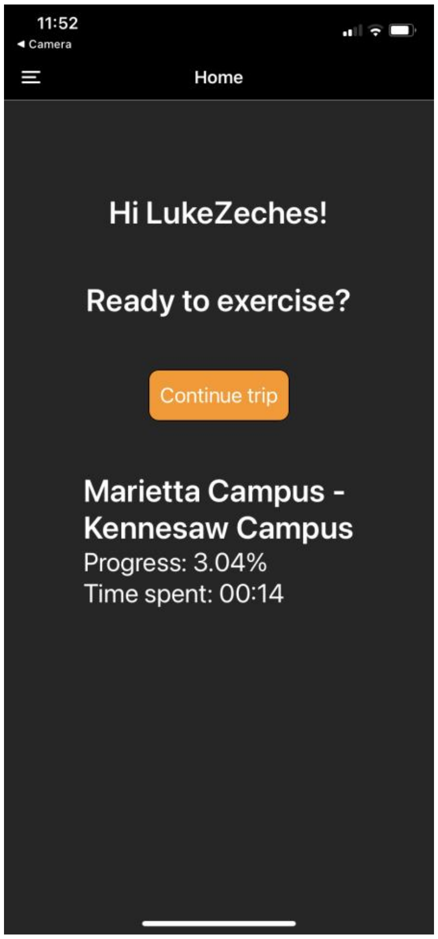


Figure 4: New Trip Screen

This screen is responsible for representing the functionality that allows a user that has logged in to start a new instance of a trip. Starting a new trip involves clicking the “Start new trip” button that is visualized above, which creates a new instance of a trip that is saved to the storage with initialized variables that represent all the fields that will be getting tracked during the actual tracking portion of our app’s deployment. There are stored variables that include the origin name and latitude and longitude of the user’s position as well as variables that include the destination name and latitude and longitude that will be stored as the signification of a created trip from point A to point B. The importance of this screen is the creation of a new trip and storage of that trip in the database as this will be a functionality used for the first time a user wants to create a trip as well as every time afterwards that the user wants to create a trip that is not already located in our database.



Hi LukeZeches!

Ready to exercise?

Marietta Campus - Kennesaw Campus
Progress: 3.04%
Time spent: 00:14

Figure 5: Continue Trip Screen

This screen is similar to the create trip screen but instead of adding a new trip and initializing the trip in the database, this screen is responsible for searching the database for a certain trip that the user wants to continue and pulling all of the tracked progress that was previously saved in the database in order for the user to continue on their fitness journey that they had already started. This screen has a button that allows for the user to click which on press will handle multiple actions and allow the user to pick from a saved trip in our database. The importance of this screen is that we can save progress and allow the user to access previous fitness sessions if they so desire, adding another level of application for our app.

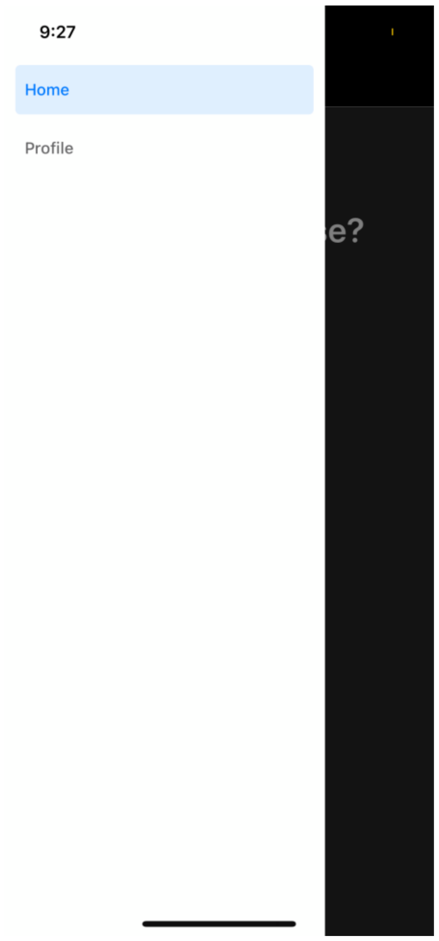
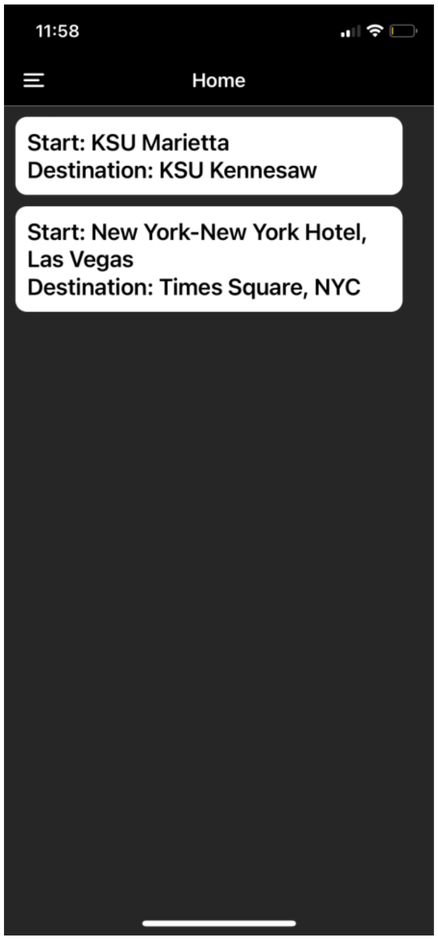


Figure 6: Sidebar Display

This screen just shows our sidebar and what its responsibility is for our application. This sidebar can be pulled to show two text boxes that include “home” and “profile” which when clicked, will navigate the user back to the respective screens. This sidebar was implemented in order to allow the user to not get roadblocked on a particular screen once to the flow of information stopped at a particular point, allowing for circumnavigation and a path to return back to the original branch of paths that starts with the home screen. Without the sidebar, it is impossible to go back once a trip has been finished and even accessing the profile screen is impossible as there is no way for the user to escape that information path once it has been entered. This was a crucial tool that we had to use in order to allow for simple navigation through the application and created a much more dynamic experience.

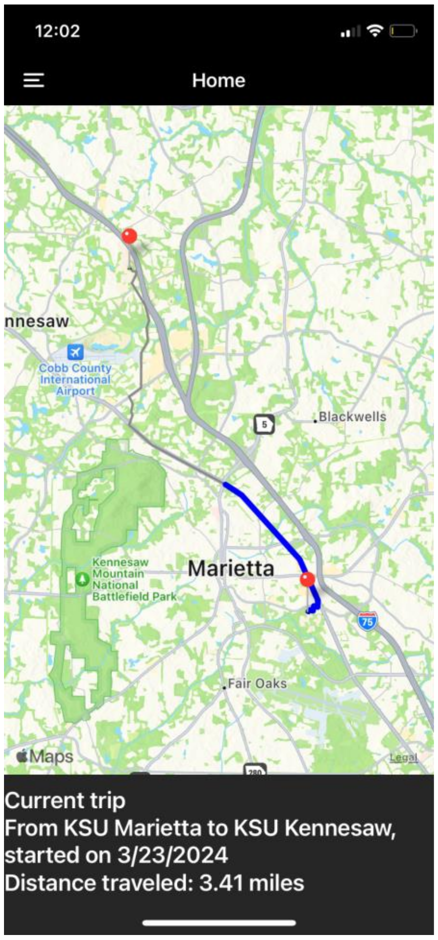


Start: KSU Marietta
Destination: KSU Kennesaw

Start: New York-New York Hotel, Las Vegas
Destination: Times Square, NYC

Figure 7: Location Menu

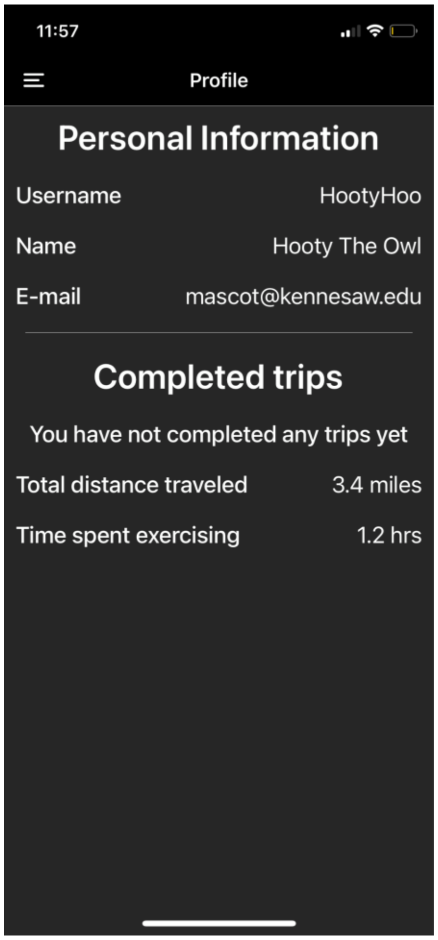
The location menu was our implementation of giving the user a chosen start and end point that they can begin their journey from. These fields are extremely important as the input that is chosen here is responsible for the heaviest set of functionalities that comes from the mapping portion of the app. With these two boxes, we have the two locations that will be set on the map and will be accessed by the Google Routes API in order to generate the best possible route between these two locations. This screen was designed to be relatively simple and provide the user with their options of choosing where they want to start and end their journey at, and then storing that information in the proper variable declarations.


12:02
Home
nnesaw
Cobb County International Airport
Blackwells
Kennesaw Mountain National Battlefield Park
Marietta
Fair Oaks
Maps
Legal

Current trip
From KSU Marietta to KSU Kennesaw, started on 3/23/2024
Distance traveled: 3.41 miles

Figure 8: Map Screen

The map screen includes a simple view of the current progress that the user has had during the time that they had this screen open. Here, it can be seen that there has been a generated map and the start point and end point have been defined on the map. The grey line that signifies the best possible walking route from the two points has been generated by the Google Routes API taking the two locations given in the location menu and finding the best path. There is also the virtual mapping of the user's current progress in the form of a polyline, which is signified by the blue line overlapping the grey line. In addition to the map representation, the maps screen also displays a live tracker that shows the distance in miles that the user has currently gone in their fitness journey as well as the date the trip was started and the locations that were used in generating the trip. This screen is the crucial tool that our app features and helps display the user's fitness journey as a virtual representation of progress.


11:57

Profile

Personal Information

Username	HootyHoo
Name	Hooty The Owl
E-mail	mascot@kennesaw.edu

Completed trips

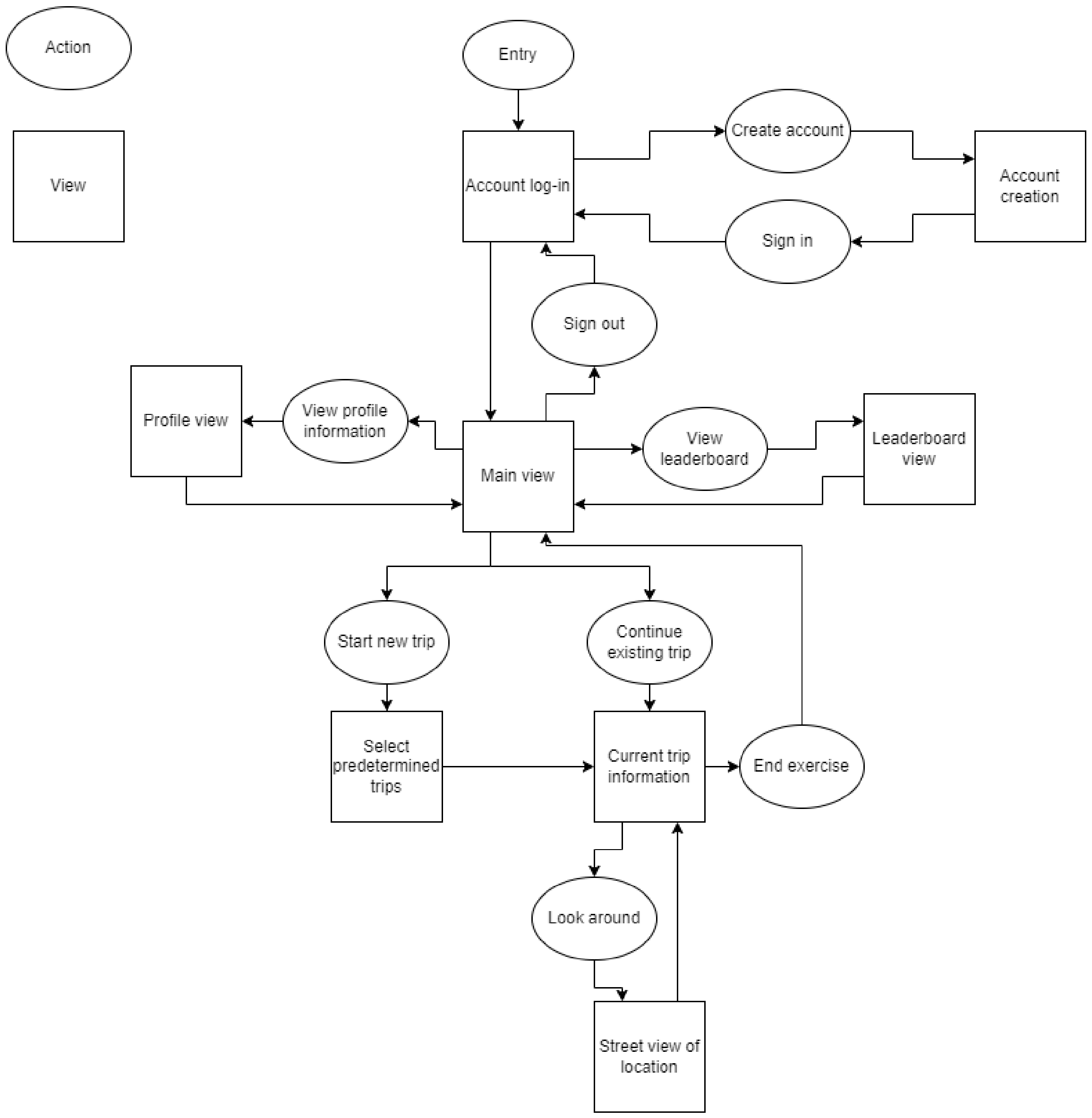
You have not completed any trips yet

Total distance traveled	3.4 miles
Time spent exercising	1.2 hrs

Figure 9: Profile Screen

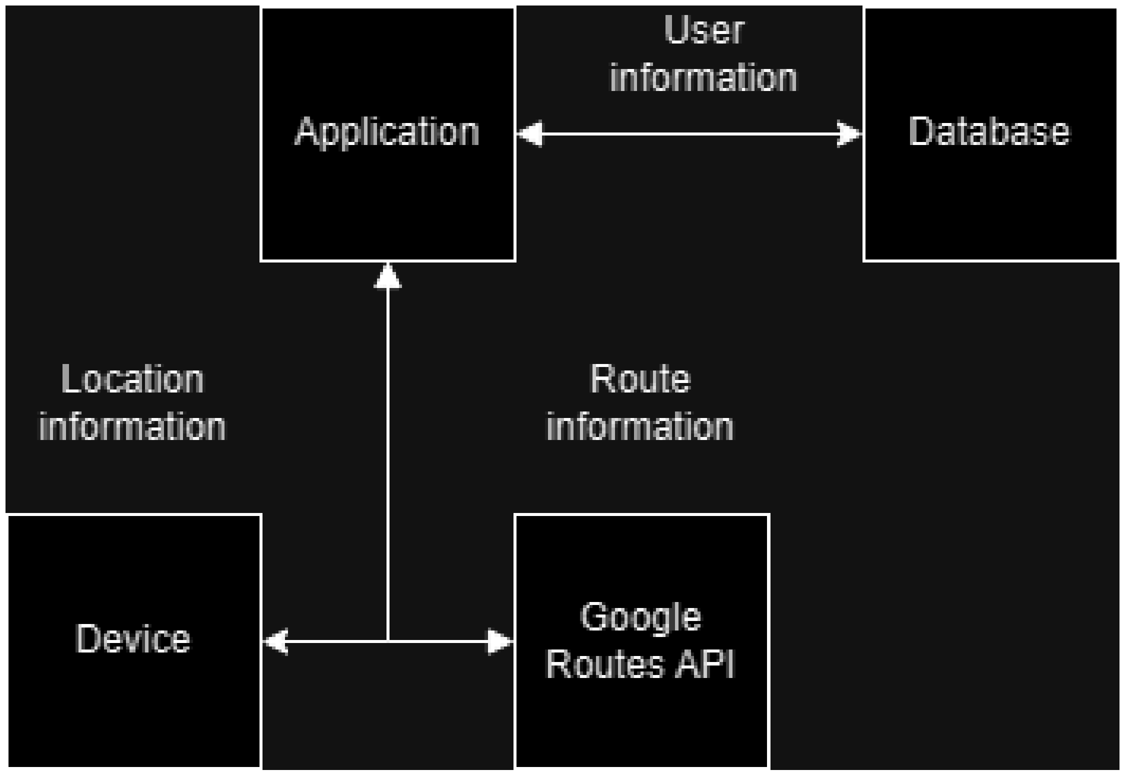
The profile screen is the place where users can see some of the statistics and personal information that is stored throughout the usage of the app. We display the username, name and email of the account that was created in the top section of this page as a reference to see who is logged in as well as a portable feature that could be used to implement a leaderboard system in the future. Underneath the personal information section, we have the trip statistics of the user in terms of total usability of the application. This section stores the total amount of fitness activity of the user while using the maps functionality that we have developed and represents that as an account value that shows the total progress taken while using our app. This profile screen is essentially an area where the user has access to all the interesting information about themselves that they have been involved with in the process of using the app and is a benchmark for the implementation of a possible leaderboard system.

6.6.2 User experience system

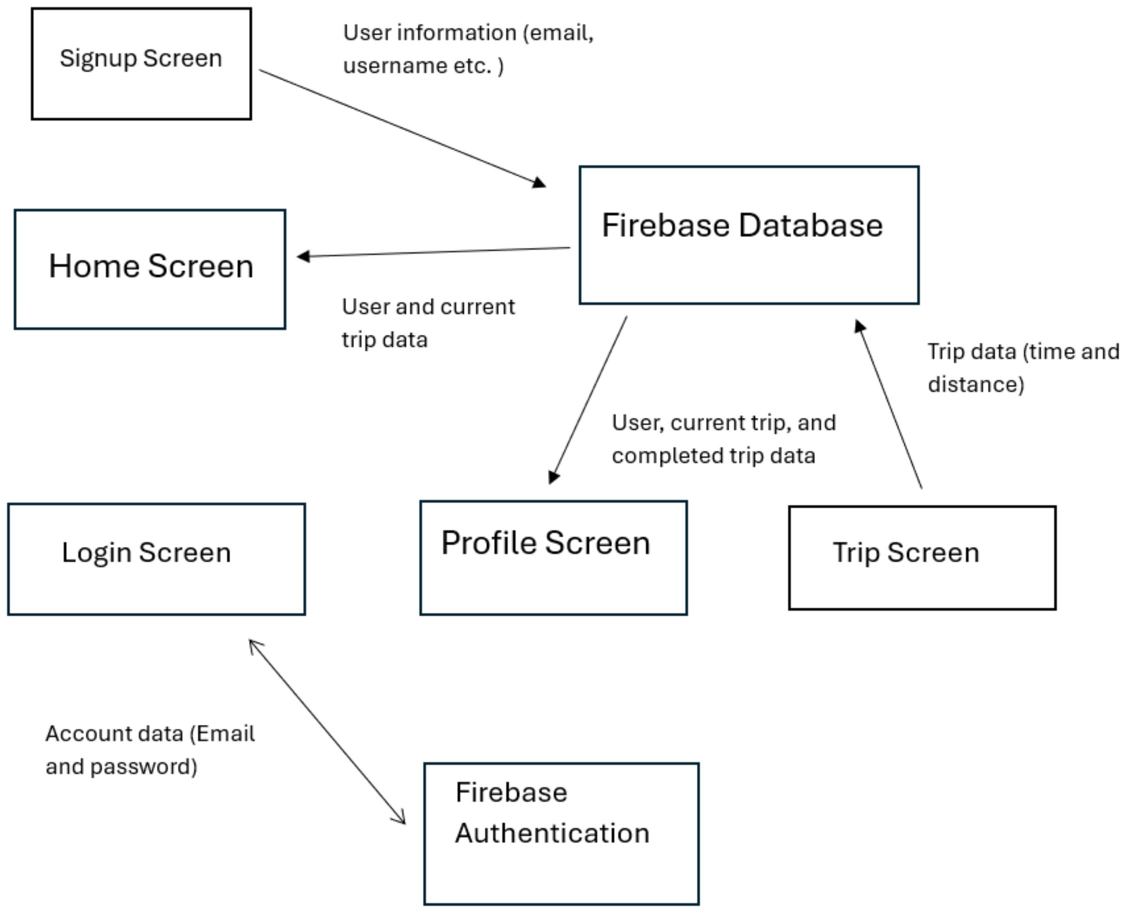


This diagram shows the various views available to the user and the connections between them. The user is first greeted with a log-in screen, with the option to move to an account creation screen or reset their password. The signup screen allows the user to create an account and return to the login screen. The reset password screen allows users to enter their password and return to the login screen as well. After logging in, they are taken to the main view, where they can start a new trip from a set of pre-generated trips or, if one already exists, they can choose to continue exercising on the existing trip. They can also exit exercising to return to the main view. This diagram shows the relative information flow of our systems and how certain sections of our app are accessed.

6.6.3 System design



These diagrams show the various components of the system and their connections. User information is passed between the application and database, such as name, username, and exercise metrics. The application communicates with the Google Routes API to display routes while the react native maps library allows for the creation of the map and all map interface functionalities that go into the front-end design of the map itself.


Signup Screen
User information (email, username etc.)
Firebase Database
Home Screen
User and current trip data
Trip data (time and distance)
User, current trip, and completed trip data
Login Screen
Profile Screen
Trip Screen
Account data (Email and password)
Firebase Authentication

Entries in the Firebase database is created when a user creates an account. Account creation is handled by Firebase's Authentication system, which takes in an email and password to create a new user. This information is also required to login to the app. When signing up, all the other information such as username, last name, and first name will be entered into the database under the user section. A current trip section is also made for each user. This section remains mostly empty until a user starts a trip. When a trip is begun, all the trip information is sent to the current trip section in the database. Metrics that update over time such as distance and time are updated once a session is ended. This data is sent to both the user and current trip sections. When the user finishes a trip, the current trip is sent to a list of completed trips and the current trip is cleared of all information. The profile screen queries all the user's information such as total distance, exercise time, and all completed trips. This information is all taken from the database and is unique to each user.

7. Development

7.1 Analysis of Technologies

We decided to use React Native to allow us to develop for both iOS and Android devices. Compiling native code for iOS requires the use of an SDK which is only available on Mac devices. Since we only had Windows machines available to us and emulating the Mac operating system would be slow and difficult, we used Expo Go for quick cross-platform development without needing to install any specific SDKs. React Native was quick to pick up and start working with because it uses concepts present in Vue.js and vanilla HTML and CSS, which members of our team have experience with.

Firebase provided an easy-to-implement solution for handling both storage and authentication. During early development, local storage was used for storing user information. This was migrated to a remote Firebase Realtime Database which required careful but minimal code modifications. Firebase Authentication allows for the creation of a new user and the authentication of an existing user. The authentication system holds the user's email and password. The Firebase Real-time Database supports real-time updates and queries to a database. Using the user's unique user ID, our app can query information specific to each user. This also allows for updating information for every user without changing the information of another user.

The Routes API, provided by Google, was necessary for getting information about a route between two locations. It provides the distance of the route and information about the polyline that displays the route on a map. This polyline information was encoded for space, so before drawing the polyline had to be converted to GeoJSON LineString format. The app then uses this LineString to construct a second LineString a certain distance along the original. Finally, the LineString objects were used to construct a set of coordinate objects to be used by the Polyline component from react-native-maps. The conversion and use of the polyline information was done using the mapbox/polyline and turf.js libraries. The react-native-maps library provided the necessary React Native components for displaying the map and polyline.

7.2 Issues Encountered

Although Expo Go was very convenient for quickly developing and testing the app, it provides limitations on writing platform-specific native code. This restricted our use of a library that would have provided street-view functionality that we were unable to otherwise implement.

At first, we tried using Expo-SQLite, which is an SQL-based database for JavaScript applications. However, we realized that SQLite doesn't allow for real-time updates, meaning that the app would need to be reloaded after each change. This didn't work for the functionality we needed, so we switched to Firebase, which worked much better.

The method of determining the distance the user has traveled involves checking their location every 10 seconds. However, the phone's location services are not always completely accurate and can give slightly different locations even when the phone has not moved. This means that progress can be made towards a trip even when the user is stationary.

8. Application Testing

Our React Native-based mobile application went through a couple of different rounds of testing in order to make sure that the functionalities that we wished to have incorporated were properly working and any possible lapses we might have had in the design process in terms of navigation were remedied. We wanted to mainly focus on testing to make sure that our database properly stored our data and that the UI/UX had all the proper components to make the app usable in a user-friendly manner. Testing of this app involved a mix of analyzing all the functionalities and possible interactions users could have with the app and seeing if everything that we implemented worked properly. It also involved seeing where the current iteration of our app fell short in certain functionalities that would cause the user's issues if they wanted to do a certain action but there was no ability for them to achieve that function. With our app, there were a lot of possible errors that occurred in the initial stages of development as the design and scope of what we initially planned and talked about did not cover all the possible situations that could occur when a user would try to navigate through the app. This section covers some of the testing approaches we took in order to fix some of the issues that were rooted in certain development stages of our app.

8.1 UI/UX Testing

In terms of overall UI/UX testing, we mainly focused on making sure that the buttons that we had incorporated properly performed their actions and that all the screen displays were showing all the objects that they were supposed to. When it comes down to the important issues that needed focus, one of these areas that we ran into in the beginning stages of development was dealing with an effect that involved the ability to swipe on the side of the screen in order to go back to the previous tab. We had to ensure that swiping on the side of the screen did not bring us back to the screen before as this was not a functionality that would be beneficial to our application where a user could be intending to use our sidebar and instead accidentally sends themselves back to the home screen. Every other issue that we ran into for the UI had to do with adjusting our components on the screen and making them appear in a better way for the user such as moving a button around or changing the type of container that holds certain information. Otherwise, our UI was relatively functional and organized even without testing, so most of the UI testing came from our own preferences on how things should be situated.

8.2 Functional Testing

Functional testing involved mainly looking to see if all the application features that we had established in our app were properly working and if there needed to be any additions to make sure that everything worked how we wanted it too. We made sure to test the functioning of our log-in button as this was a main gateway in terms of both authentication of a user and allowing navigation to our app's main features. This log-in button was tested to make sure that when it was pressed, the user's written in information was checked against the instances of signed up users in our database and made sure that the log-in credentials were valid. In addition to this, we also had to check the functionality of our sign-up button in order to make sure that all the users that signed up with our app had their information populated in the database. We also had to ensure that all systems, such as the distance tracker, worked properly. In order to test this, we had to walk a predetermined distance and watch the tracker to see if it records the correct distance. If the tracker worked properly, we could see the distance field updating during the duration of the walk in a manner that resembled the distance that we were covering. There was also some issues in terms of figuring out a technique that would allow us to correctly map the distance that was being tracked as in order to map the distance onto a map, there are certain requirements that have to be met which include providing longitudinal and latitudinal information of where the user is going to located along the path every few seconds that the current location is updated. This involved some testing as well as research into the best ways to approach this sort of problem, which we ended up having a part of our testing centered upon. These are some of the main testing areas of focus that were examined in order to optimize the functionality of our application’s objectives.

8.3 Results/Reporting

Functional Test Report		
	Pass	Fail
Create Account	yes	
Log-in Succesfully	yes	
Log-in Populates Database	yes	
Create New Trip Button Works on Press	yes	
Continue Trip Button Works on Press	yes	
Swipe on Edge of Screen Doesn't Change Page	yes	
Sidebar Displays Correctly	yes	
Line Generated from Start to Destination Location	yes	
Distance Tracking Works	yes	
Navigation Menu Works	yes	

In the earlier rounds of testing that we went through, we found that a lot of these columns were actually failing and the recognition of these problems were the main barriers to having our app be fully functional with no bugs or errors that could possibly pop-up. As we progressed through our rounds of testing, we were able to pick out these errors and determine how to fix them, with issues like swiping on the edge of the screen and sidebar displaying correctly in the fashion that we wanted it to. Currently, we have been able to test pretty much every feature that we have listed here and get all of these features to pass, resulting in a much cleaner and less buggy interface that has all the items we need as well as providing easier ways to get around the app. We have located most of the error conditions that were appearing in the earlier phases of development and were able to identify why certain things weren’t working, and testing helped us to make sure that those error conditions were taken care of. In the current state of the app, we are satisfied with the testing developments we have made and believe our app to be very polished in regard to the conditions shown above.

9. Source Code Version Control

In terms of our version control, we used the functionalities that GitHub provides in order to share access and develop our code together. We created our project on an organizational account which allowed for enhanced collaboration between our group on all things regarding our programming progress. Using

GitHub allowed us to access version control and we were able to develop different sections of code while not affecting any of the work that was being done by other group members. Using certain commands, we were able to pull previous versions of code that had been worked on by other group members so that we all had the same versions as well as commit and push our own code when we had finished a section that we wanted to add to the main branch. Version control simply allowed us to access and update a final draft of our code with multiple incoming requests possible that kept our main branch updated and functional to the most updated version that we had.

10. Future Work

This app was essentially a benchmark to see how far we could get with the Google Routes API and see some of the possible functionalities that we could implement with it to virtualize a fitness journey. We wanted to take a regular fitness app and the main components that identify it and add some sort of interesting tool that could transport a user from the regular fitness app experience into some virtual representation of that journey. The goal in the future would be to add more components to the app that make it stand out even more from other fitness apps such as a system that would allow for tying in the screen time settings on a mobile device and restricting access from certain apps that might distract someone from their fitness activity, with the user's approval. Other future work could involve adding an additional tool to the app that might personalize the user's fitness journey to an even greater extent or upgrade the current capabilities that we employed using the Google Routes API. This could involve finding some sort of way to import certain maps from video games or movies where the user could choose destinations and starting points from their favorite movies/video games and get to travel the journey through that world while they are exercising. These developments were out of the scope of our initial timeline as we wanted to focus on getting a working product with the one tool correctly implemented first before we thought of further additions, and intensive research would have to be done in order to figure out a way to gain permissions and construct maps that resemble the fictional landscapes that we would want to display. In addition to these ideas, we would probably look to just expand in terms of some of the information that we might be able to gather and display to the user in order to add more features and statistics for users to look at. Overall, this is a foundation for a much larger app that could combine multiple fitness features and be a very unique adaptation and representation of a fitness health journey.

11. Final Summary/Conclusion

In conclusion, the work that was done in order to plan, design and implement this app was extremely intensive and required a lot of collaboration and adaptation to new technologies. We all had very little experience in using the React Native framework as well as Firebase, so those are two technologies that we now have better understandings of. This was also most of our first times creating an app and developing with an emulator, so we were able to learn how to develop and test with the use of the Android Studio emulator as well as Expo Go. We were able to put together our app and achieve the scope that we set for us in the initial design and requirements stages, while also setting up our app for further developments that would be very interesting to implement as well as make it a very commercial product. This was a very interesting and informative journey for all three of us and gave us a great introduction and understanding of front-end development as well as application development in general.

12. Appendix

12.1 React Native Tutorial Completion

"I completed the React Native tutorial."
Signed: Adam Borowski
Signed: Luke Zeches
Signed: Michael Bolnik