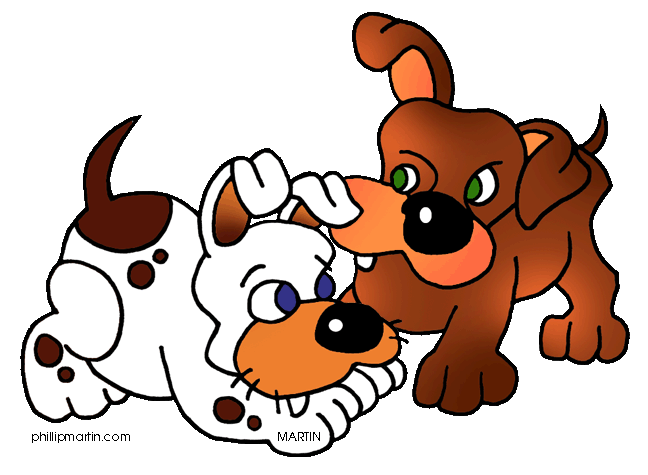
Logo%20Main%20200

**Boston University**

**Electrical & Computer Engineering**

**EC464 Capstone Senior Design Project**

User's Manual

WOOF

*By: Team 02*

**Members**

Nafis Abeer [nafis@bu.edu](mailto:nafis@bu.edu)

Justin Lam [jlam8080@bu.edu](mailto:jlam8080@bu.edu)

Chase Maivald [chase1@bu.edu](mailto:chase1@bu.edu)

Rajiv Ramroop [raj19@bu.edu](mailto:raj19@bu.edu)

Daniel Shimon [dshimon@bu.edu](mailto:dshimon@bu.edu)

:

#### Woof User Manual

#### Table of Contents

Executive Summary ii

1 Introduction 3

2 System Overview and Installation 4

2.1 Overview block diagram 4

2.2 User interface 4

2.3 Physical description. 6

2.4 Installation, setup, and support 6

3 Operation of the Project 7

3.1 Operating Mode 1: Normal Operation 7

3.2 Operating Mode 2: Abnormal Operations 7

3.3 Safety Issues 7

4 Technical Background 9

5 Relevant Engineering Standards 12

6 Cost Breakdown 13

7 Appendices 14

7.1 Appendix A - Specifications 14

7.3 Appendix C – Team Information 16

# Executive Summary

*Woof!* is an ideal iOS social media application for dog owners. Rather than relying on individual owners to actively maintain dog friends, we propose a React Native maps interface to display locations of “friends” added by the user. To cater the application for dogs’ preferences, we require owners to purchase a dog-tag (dog-fleece due to size constraints) fitted with a GPS tracker, accelerometer, and a microphone. We are implementing a recommender system based on the owner’s dog’s interactions with other dogs wearing the tag. This recommender system will leverage the dog-tag’s proximity to other tags to start collecting audio. The collected data would be processed in the cloud before being sent back to the user as a friend suggestion, if the interaction is classified as friendly. *Woof!* is expected to deliver accurate suggestions for “friends” whenever the dog has a positive interaction with another dog. Users will be provided a profile page, instant messaging, “friends” page, interaction-history page along with a Dogmaps screen.

# Introduction

The *Woof!* application aims to bridge the disconnect in communication between humans and their pet dogs. Dogs are unable to tell their humans when they would like to go for walks, and it is up to the human to incentivize themselves to take their pets out. Additionally, dogs cannot tell their humans that they like hanging out with another dog. Unless the owner is attentive to every interaction their dog has, they miss most of the positive interactions their dog experiences. These positive interactions are beneficial for dogs, and repeated interactions with the same dog are great for reinforcing behavior learned during previous exchanges.

Studies have shown that dogs can greatly benefit from long term friendships and playtime outdoors. In fact, “if deprived of the opportunity to play with other dogs, they have no other choice but to direct play behavior towards humans,” meaning repeated dog-to-dog interaction could reduce responsibilities for owners. Dogs tend to show more brain activity when shown other dogs, indicating that they enjoy the presence of their companions. Furthermore, playtime for dogs could serve the following functions: 1) developing motor skills; 2) training for the unexpected; 3) social cohesion; and 4) play as a by-product of biological processes. These benefits are enough to show why it's important for owners to let their dogs make friends, or at least take them out for walks.

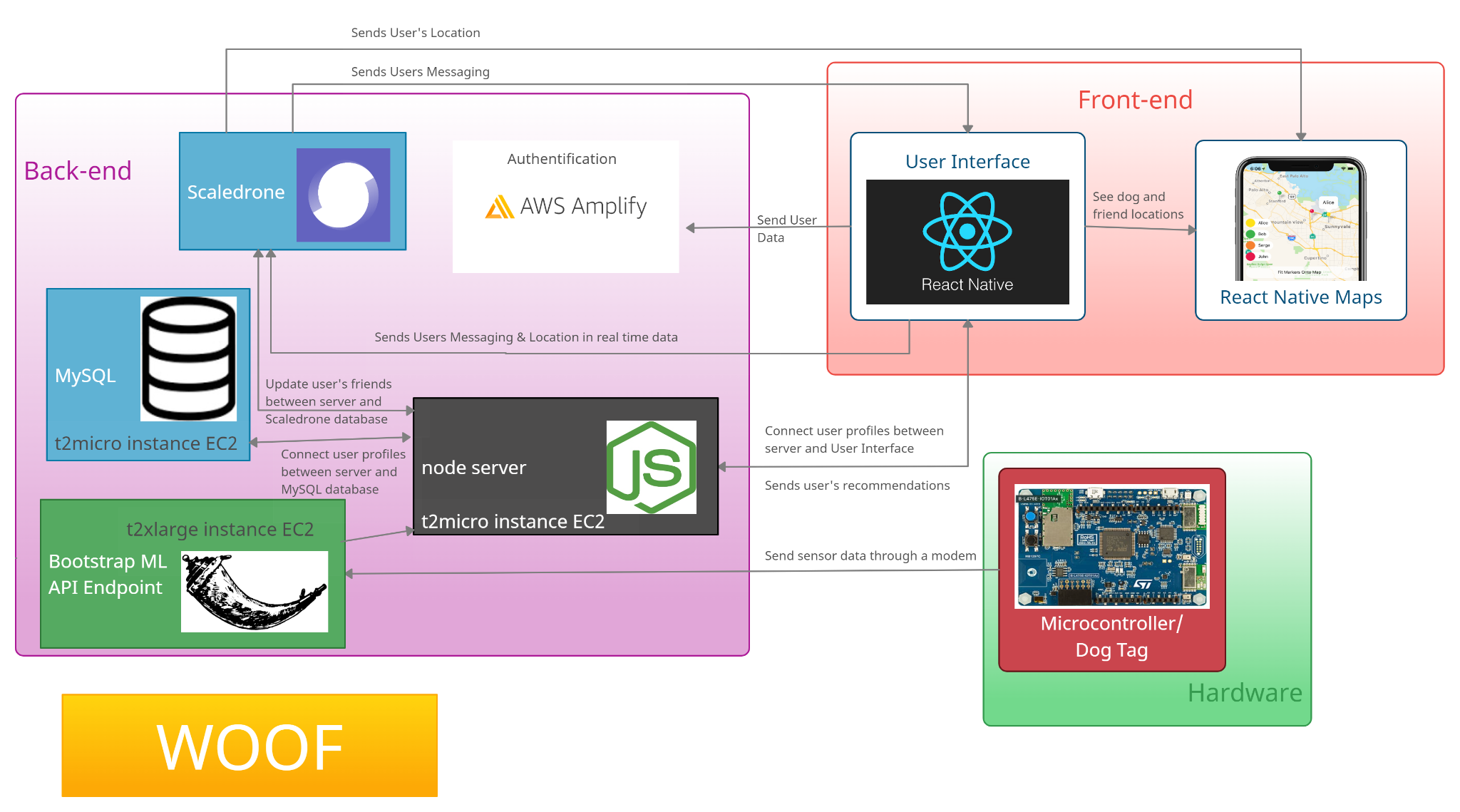
Our solution, Woof, is going to help foster dogs’ relationships with other dogs, allowing them to connect more frequently. We are proposing a social media platform to allow owners to connect with other friendly dogs, as well as recognize other dogs who aren’t too friendly with your own dog. The application will leverage a recommender system based off of audio samples taken during interactions. The interactions are recorded with a microphone when BLE modules of other dogs’ devices are detected. Our hardware interface is wrapped in a dog-fleece that will be equipped with a power source, ESP32 microcontroller, GPS module and microphone. We deliver a social media app to the user to be able to view friends’ locations, track their dogs, message friends and have a user profile.

Potential safety issues could be leaving the GPS on once the users enter their homes, as malicious users added from the friends’ list could gain access to user’s addresses. Additionally, exposing the hardware could prove to be an issue for hardware performance. Lastly, users should keep in mind that recommendations are not 100% accurate when adding new friends.

Our solution requires users to make a one- time purchase of the dog fleece, which will come equipped with the hardware. The user’s hardware “unique integer” automatically is stored as their user id after downloading the application. This will allow us to identify the user on our back-end servers. From there we can supply locations of the fleece, user recommendations and all other features of our application.

# System Overview and Installation

## Overview block diagram

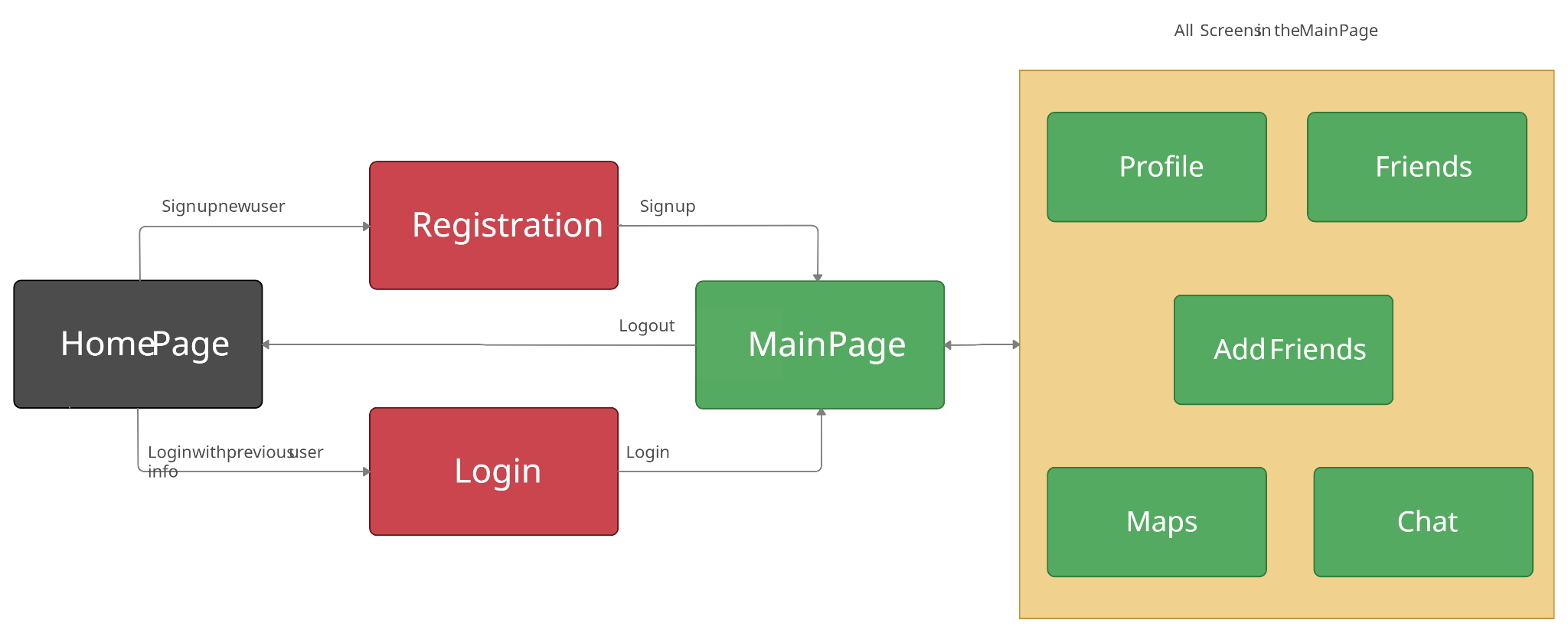


*Figure 2.1 System diagram of our company WOOF!*

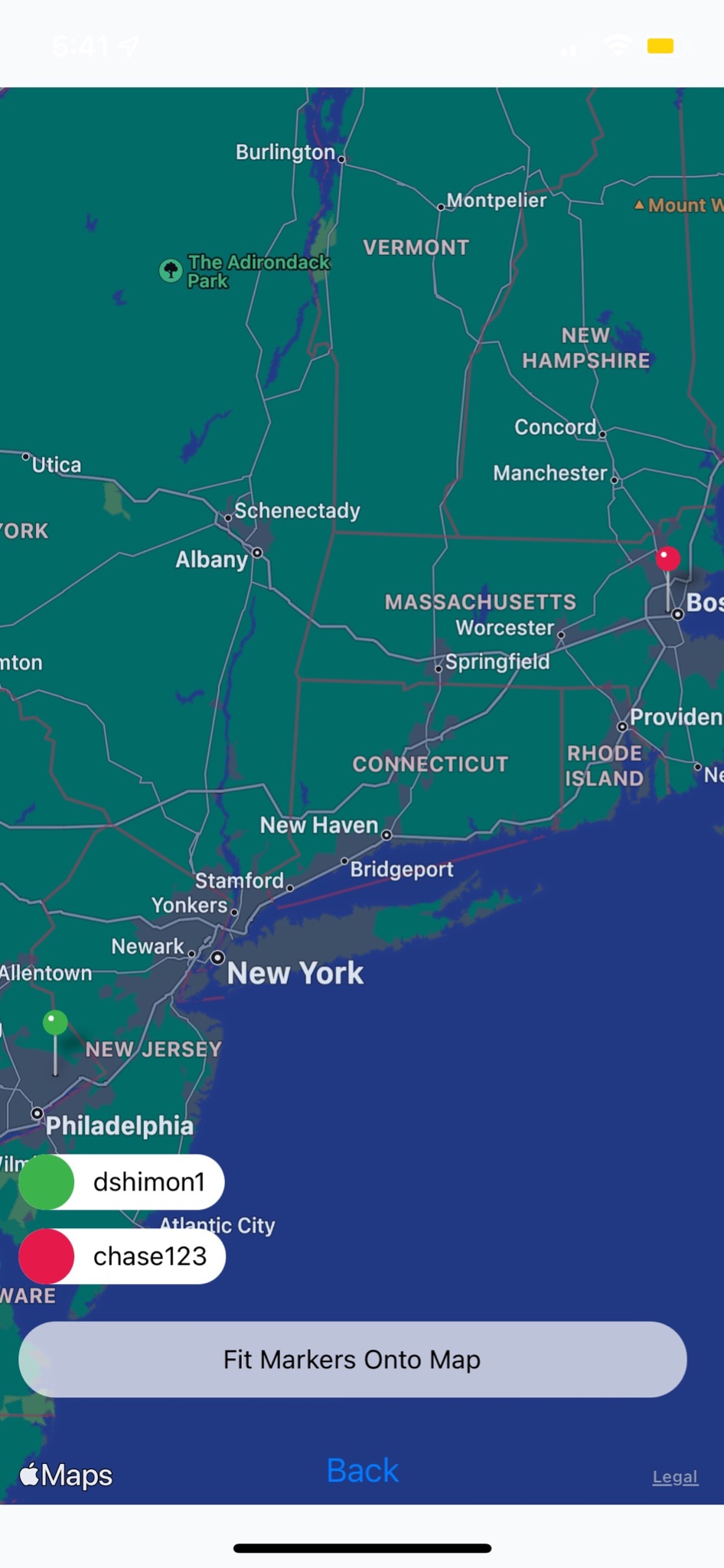
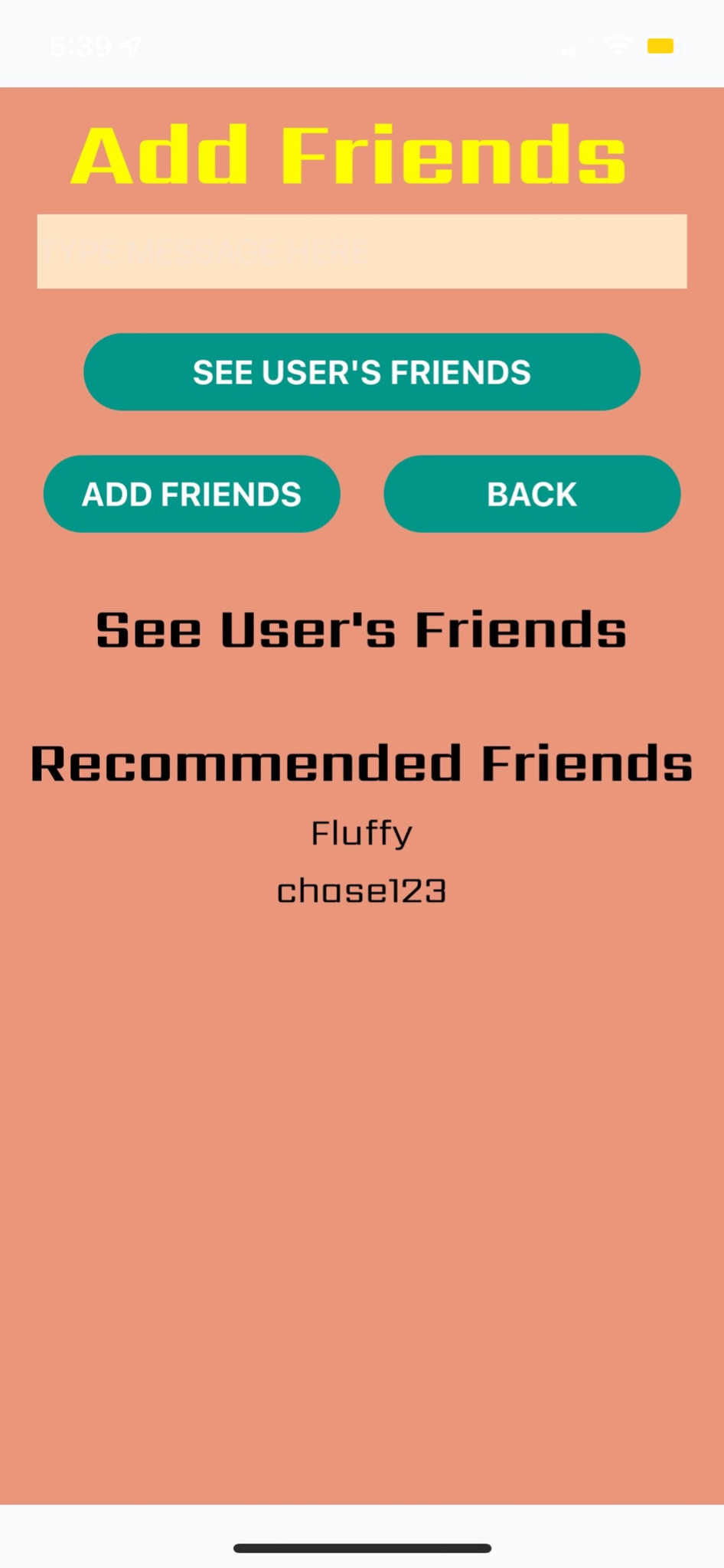
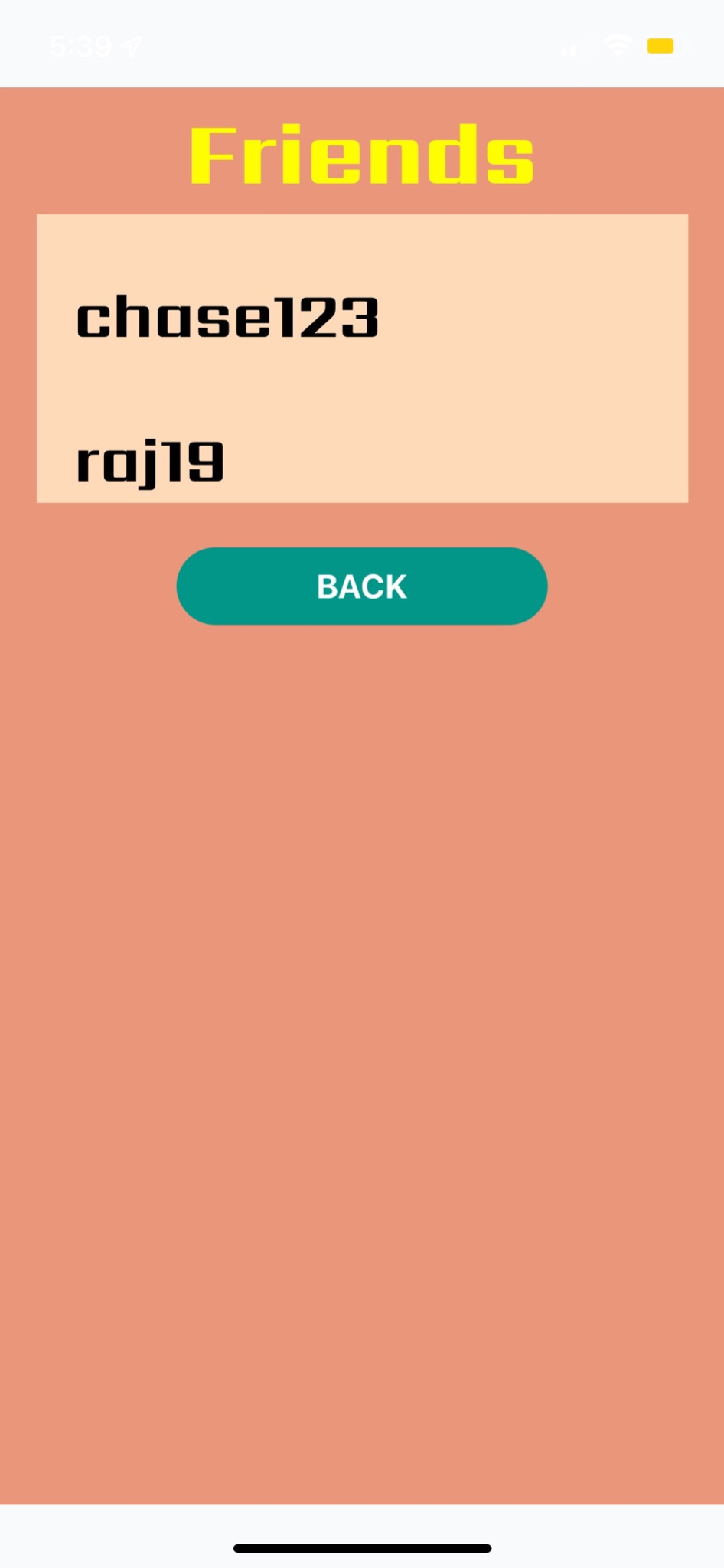
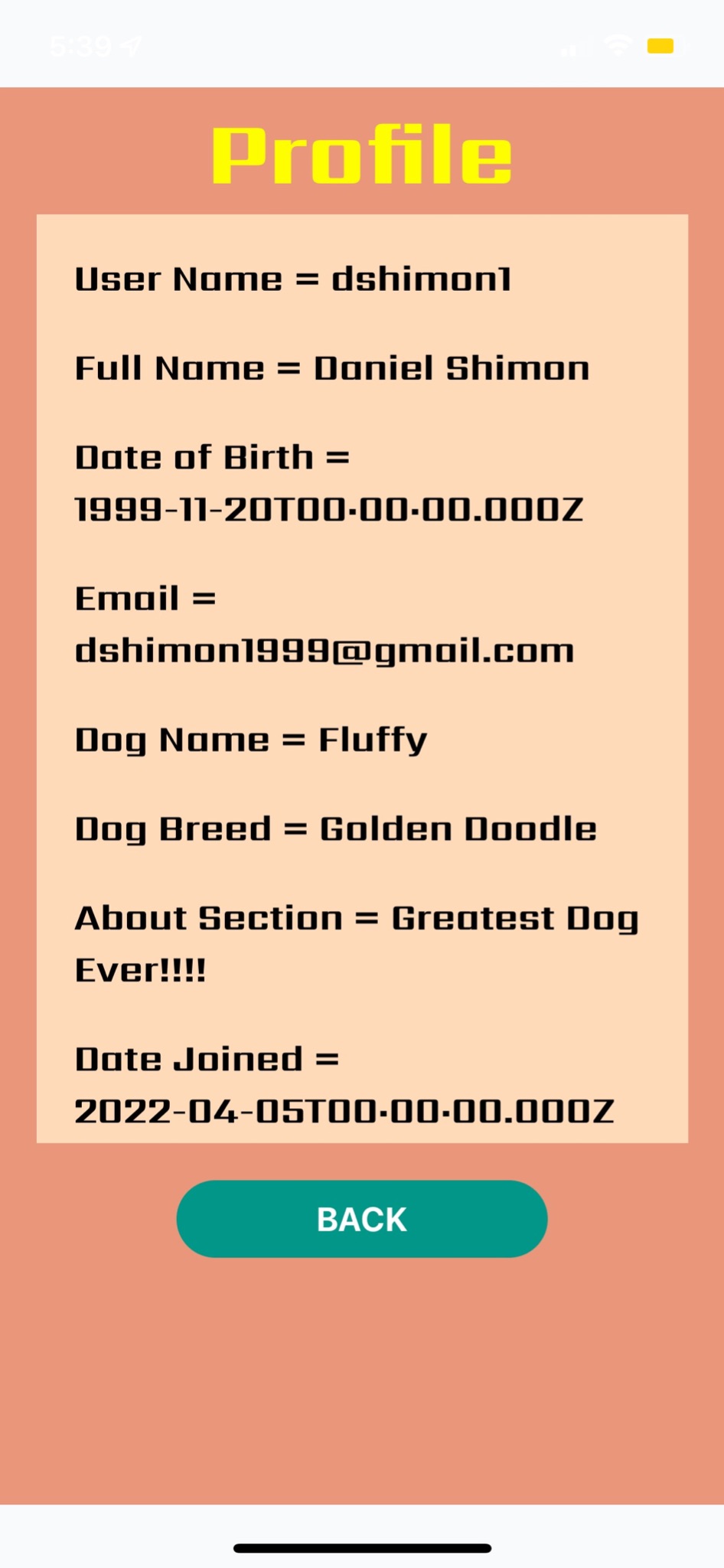
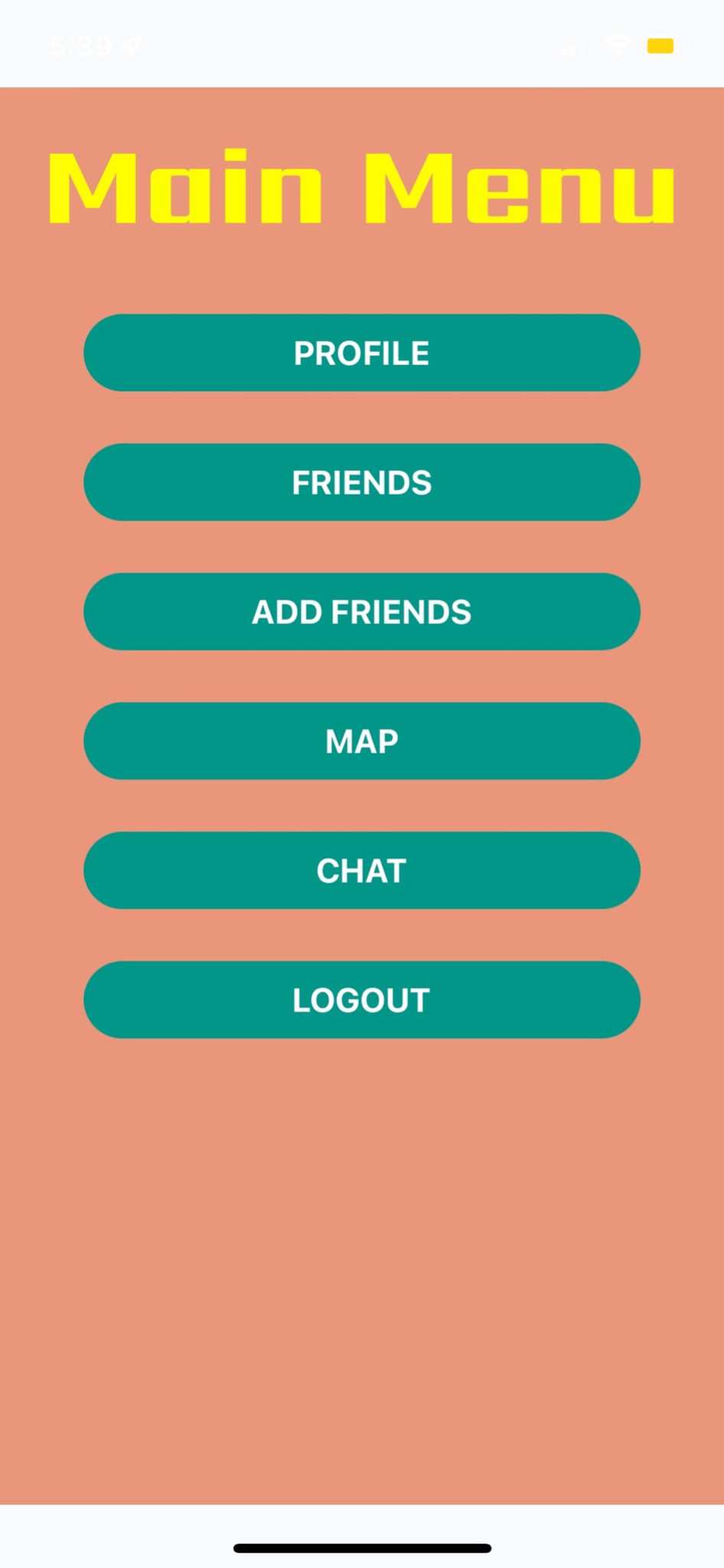
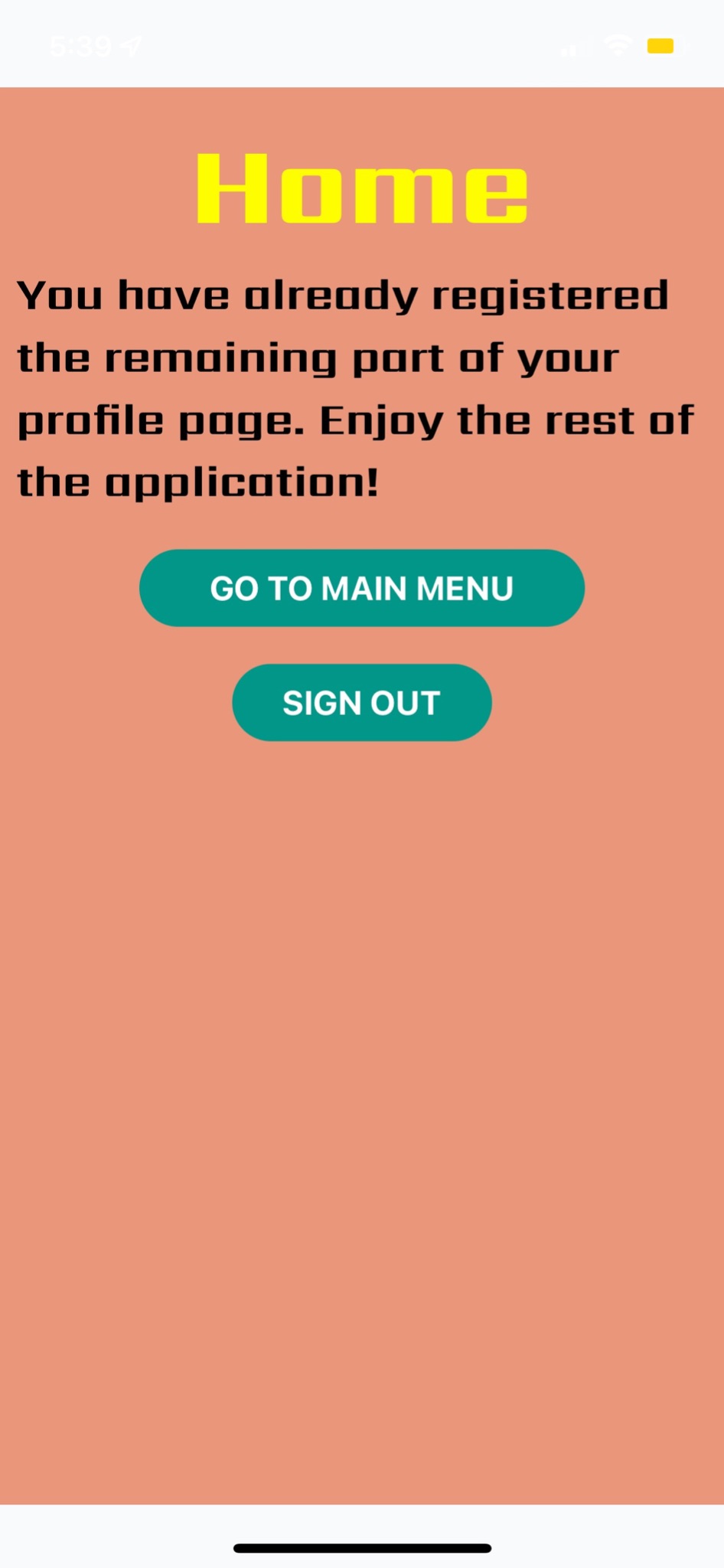
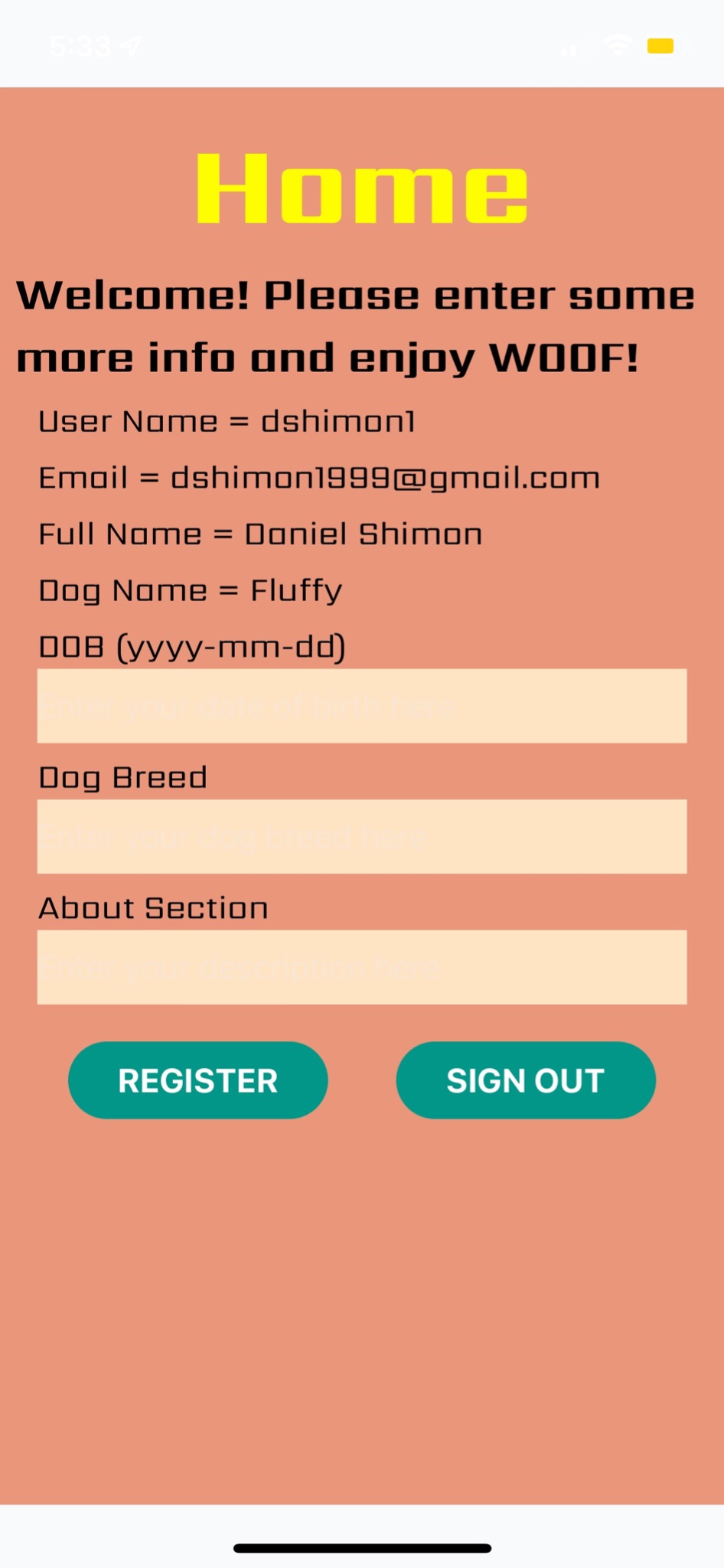
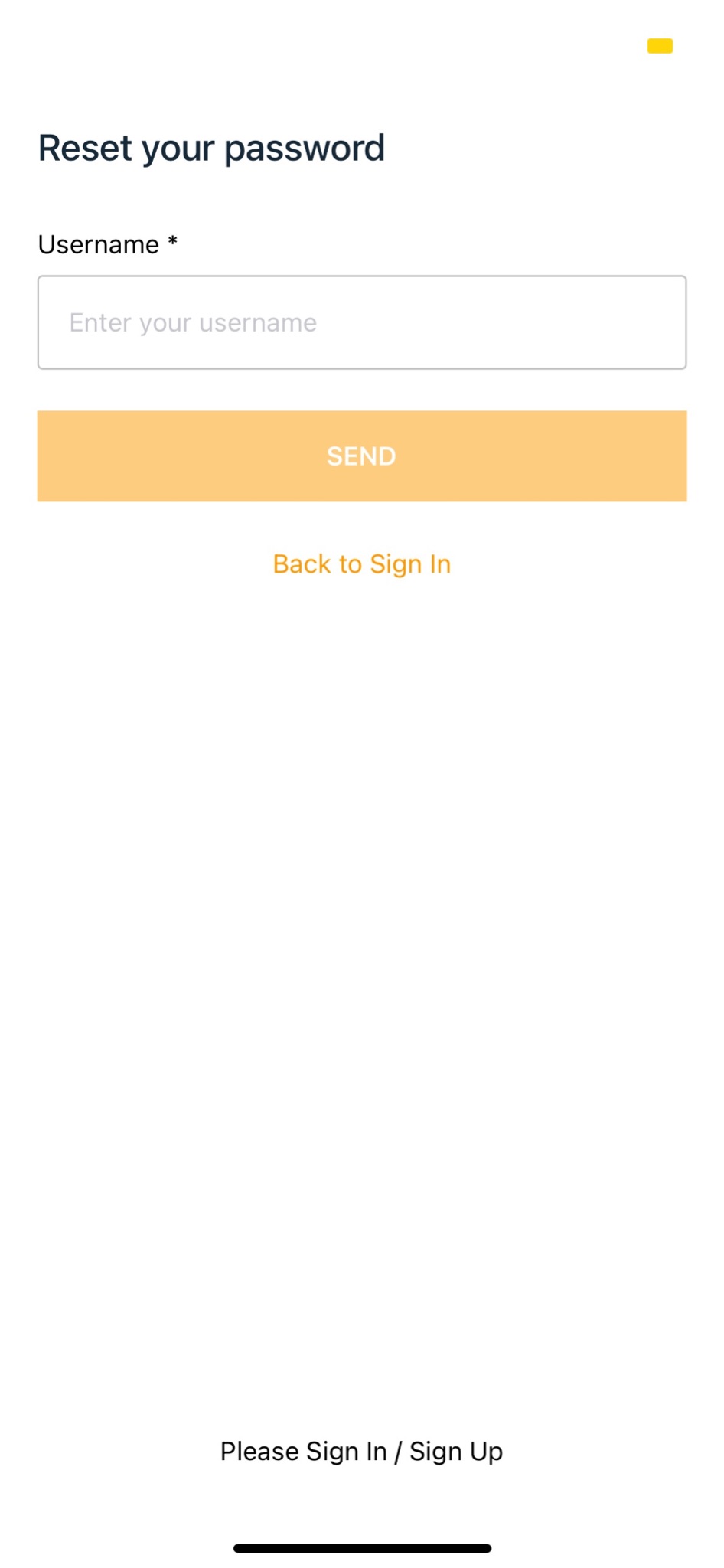
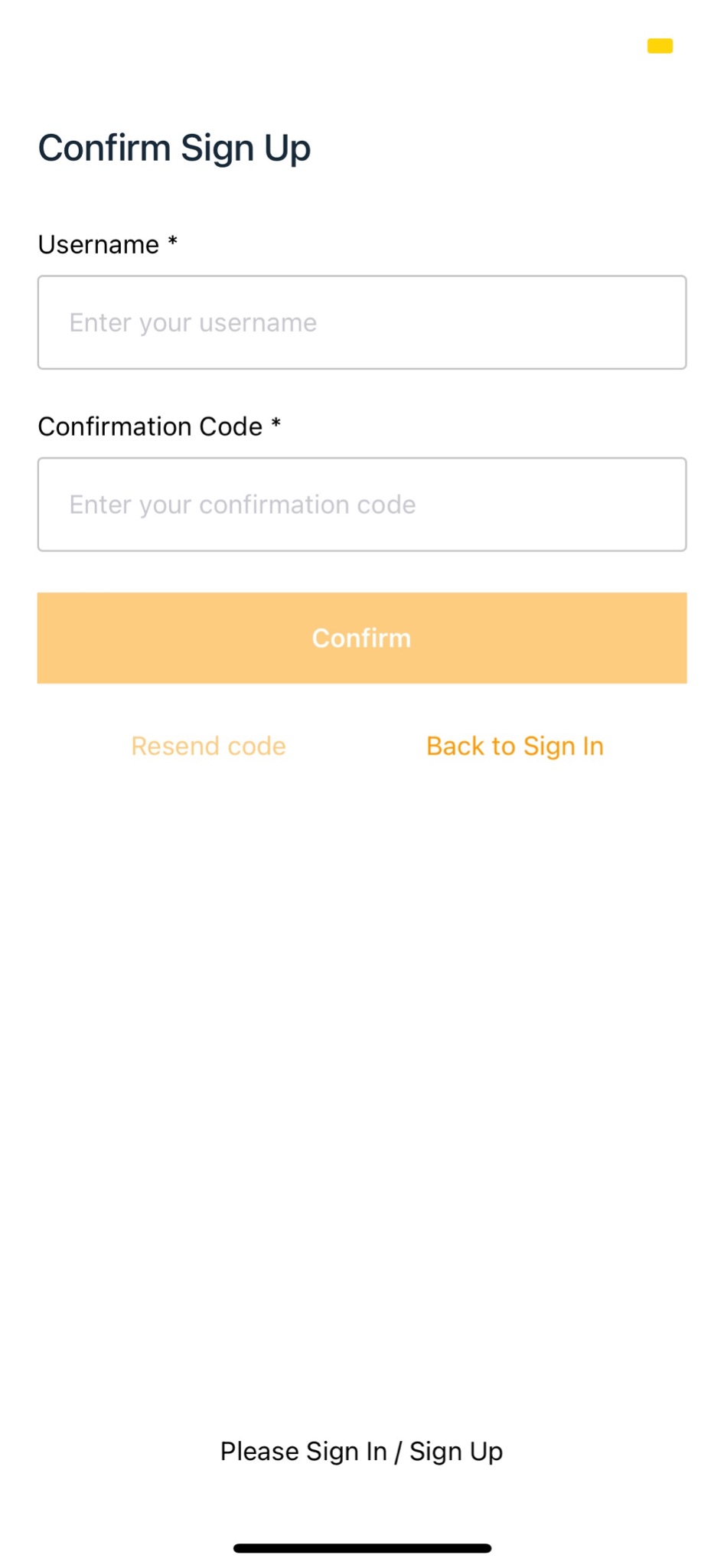
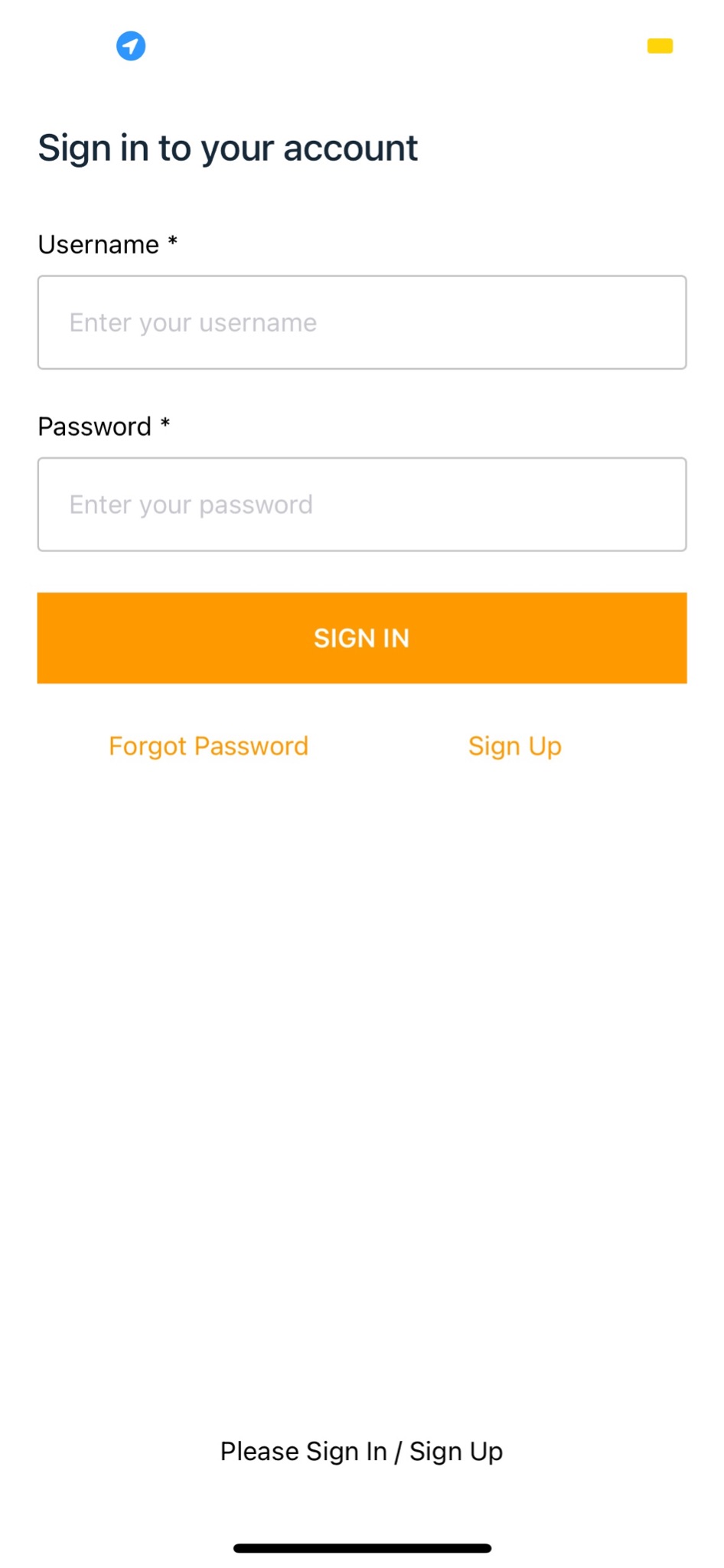
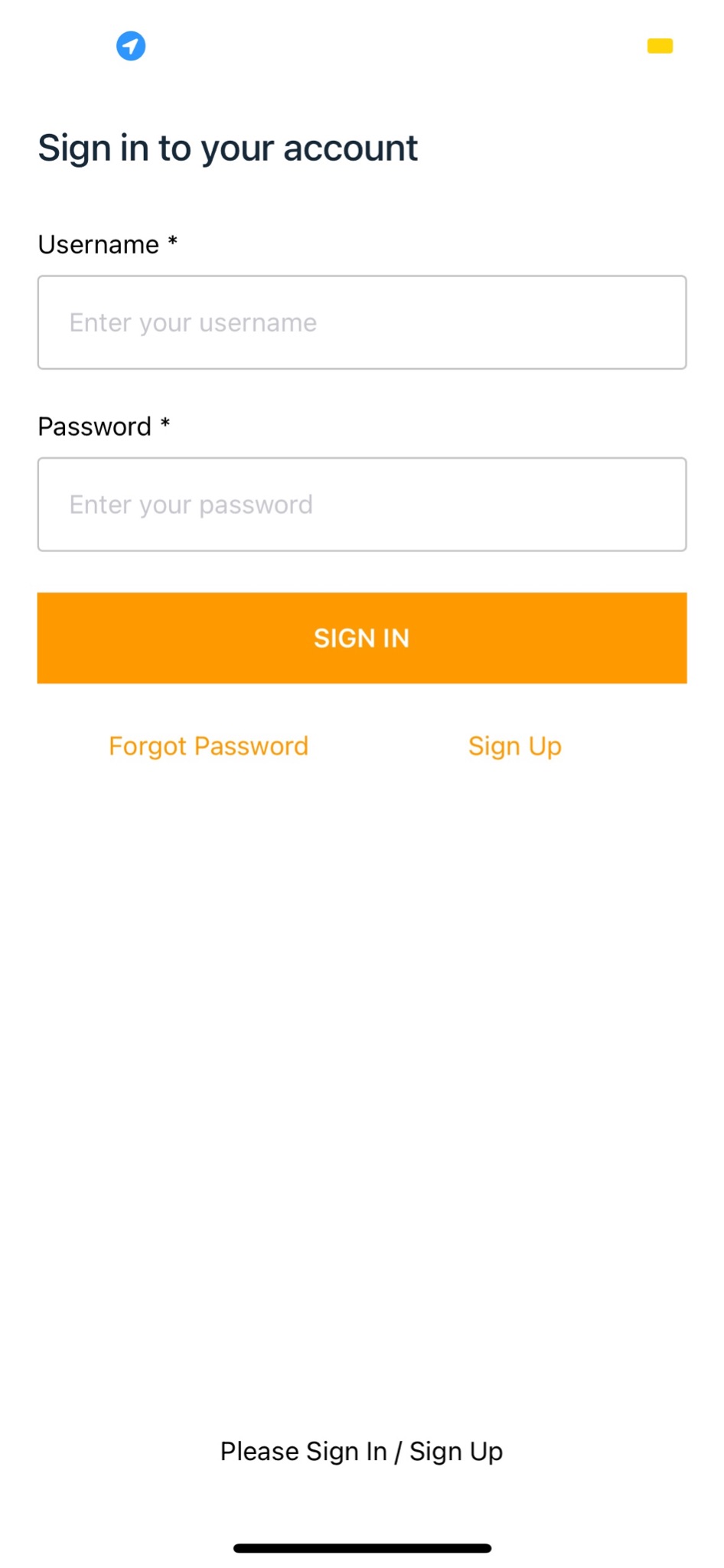
## 

## User Interface

WOOF’s mobile application uses AWS Amplify as its user profile authentication method. The app provides a registration page, login page, and a page for forgotten passwords. New users have the ability to connect an email to a new account through a verification code sent to the provided email. If the app detects a new user, it will prompt users to enter more details, such as date of birth and dog breed. Signing in will open the main menu, where several buttons clearly label all of the mobile application’s features. There are buttons for viewing your profile, list of friends, location of you and your friends, and chatting with your friends.



*Figure 2.2.1 User Interface Screen Flow*



*Figure 2.2.2 Mobile Application User Interface*

## Physical description



​​*Figure 2.3 Hardware installation Concept*

## Installation, setup, and support

The *Woof!* Jacket is first powered using 3 AAA batteries. Three standard AAA batteries produce approximately 4.5v when fully charged. The load is 0.5 watts. Watts=amps x volts, so 0.5 watts at 4.5 volts is about 110 milliamps, or .11 amps. An average AAA alkaline battery contains about 1000 milliamp-hours, so, therefore, the batteries would last a maximum of 9.09 straight hours of use, assuming perfect conditions. In reality, you can probably deduct about an hour from that rating, as the battery will not be 100% efficient as it becomes discharged. The wattage will remain the same, so as the voltage of the battery drops towards the end of its runtime, the amperage will increase, thus draining the battery faster. Therefore, one can estimate that7–8 hours of usage is how long it will take to drain the battery. Once the batteries are inserted into the battery pocket of the jacket, and the battery pocket is sealed, the *Woof!* jacket is to be safely placed onto the dog. Please ensure that the velcros are secure and the jacket is firmly fitted. Once the jacket is on the dog, open up the *Woof!* application and sign in to your account.

Installation of the software is relatively simple. The user needs to connect their phone and run the xcode application so the app downloads on their phone. The goal for the future would be to have this application be downloadable from the app store, which is definitely feasible. The software of the front-end and back-end is load- balanced to support up to 1,500 users utilizing the application a couple of times every week, supporting up to 10 TB in total of read/write traveling between the front-end, back-end, and database between all these users, which assumes approximately 10 hours of active usage per user per week.

# Operation of the Project

## Operating Mode 1: Normal Operation

Once the user has purchased the fleece jacket, and downloaded our application, the user must register their fleece onto our application with the following steps:

1. Follow on screen instructions to register the ESP32 Unique Integer ID
   1. Find the UID attached to the user manual.
   2. Enter the UID when prompted then select a user name.
2. Connect the ESP to the internet by plugging in the ESP to a laptop interface and manually inputting the SSID and PASSWORD of a wifi network or hotspot.
3. The user then gets created with their Unique Integer automatically associated as their userid and an automatically generated date joined, and they must fill in the rest of the information required to complete their profile such as: full name, date of birth, pronouns, email, password, dog name, dog breed, and about me.

Once registration is complete, the user is free to take their dog to the park and meet new users. Assuming they followed the steps to power up their fleece, the fleece will continuously be listening by its BLE sensor for other registered device’s BLEs. If it detects a nearby device(s), it will start recording the interaction audio. The audio feed is sent to our servers to be processed. If hostile noises are recorded, the user will receive anti-recommendations about the user with the BLE that initiated the negative interaction. The positive interactions will result in positive recommendations being sent to the user interface. From there the user will be able to:

1. Request the recommended user as a friend
2. With their friends the user can:
   1. Send messages
   2. View locations when the users are out of Ghost mode (location turned off)
3. View profile of added friends

## Operating Mode 2: Abnormal Operations

Abnormal operations are limited, as the application has limited functionality without an online fleece. With an offline mode, the user is able to simply view their profile, and view friends’ locations and message existing friends. The recommendation features must be enabled by an online fleece. If either the transmitter or receiver happens to go offline, contact us.

## Safety Issues

At *Woof!,* we are committed to providing our users with the best real-world, authentic dog walking experience. As we continue to enhance our application features and services, we want to provide you with clear and simple information about what and what not to do when using your *Woof!* Jacket. DO NOT expose your *Woof!* Jacket to water, as this WILL damage the onboard power supply. While the *Woof!* Jacket is waterproof, excessive exposure to water will damage the device. Please do NOT put this device on the dog when it is raining to ensure maximum protection. Also ensure that the size of the jacket is sufficiently large to fit your dog, as a jacket that is too tight may injure the dog.

The messaging feature of the application is incorporated to allow strangers to converse as fellow dog owners, and to build trust and rapport such that unsafe interactions are mitigated. Geocaching has become more popularly used through the 2000s, but simply marking pins to coordinates has proven to be unsafe because constantly displaying one’s location may tempt stalkers or others with malicious intent. Therefore, *Woof!* allows only the option to display one’s live location to approved friends. In addition, it has the option to hide one’s location for every approved friend when they return back home, and it has the option to automatically turn on live location in a public park/space.

# Technical Background

The *Woof!* Jacket has a water resistance rating of 50 meters under the ISO standard 22810:2010, meaning it may be used during light precipitation. The product is a textile, tailored, comfortable jacket capable of performing physical activity sensing in *vivo.* The jacket is composed of flexible multi-layer encapsulated sensor modules and microcontrollers. Specifically, it consists of a GPS module, a bluetooth low-energy (BLE) module, a cellular data module, and a microphone. The GPS module is a NEO-6M V2 that is connected to an ESP32 via jumper wires. The microphone is a IMNP-441 sensor that is connected to the ESP32 via jumper wires. The microphone data is collected once the *Woof!* Jacket comes in contact with another *Woof!* device’s BLE. Once the BLE received signal strength indicator (RSSI) is strong enough, microphone data is collected for the duration of the encounter. This audio data is collected via the IMNP-441 and is initially stored in the Serial Peripheral Interface Flash File System (SPIFFS) of the ESP32. After collection, the audio data is sent directly to a Flask API that returns a prediction of the encounter. Following this prediction, the application will then suggest that a user adds another user as a friend if the encounter was determined by the model to be “positive”. The data stored in SPIFFS is then deleted, so new microphone data may be collected for future interactions.

The entire front-end is developed using React Native. React Native is an open-source JavaScript framework, based on React, that is designed to build applications on a variety of platforms. Designers are thereby able to use the same code base for different platforms. Our application currently works only for iOS. React Native has a multitude of libraries, some owned and maintained by the React Native Framework, and others created by developers of various services. By using NPM as a package manager for the JavaScript Code, we can add different functionalities such as location services, usage of scaledrone, and other libraries simply owned by React Native such as Maps.

There are two types of components to use for React Native, functional and class based components. We decided to go with the class based style due to rendering information based on state changes as opposed to prop changes, meaning that every time the state of components changes they will be re-rendered.

The front-end of the application includes several different screens which are navigated to using React Native Navigation. Our user interface has a home page where users can click to register an account or log into a previously made account. Our user authentication is connected to AWS Amplify, where each username and account information will be securely stored. As shown in figure 2.2.1, the registration and login page will be accessible through the home screen. Once logged in, users will see the main screen of our application, which features six functional buttons. The buttons shown from top to bottom are labeled profile, friends, add friends, map, chat, logout. The profile page displays personal information that the user can edit, such as dog name, dog breed, address, etc. The friends page displays the user’s current friends. The add friends page allows users to search for new friends to add. The map button opens up a React Native Map that allows users to find their dog’s location through GPS. The chat button opens a page that allows users to communicate with each other through Scaledrone. The logout button signs the user out of their account and returns to the home screen. Each button opens a unique page and simultaneously sends a REST API to retrieve specific user data, which will be explained.Users can input text through Input capabilities, and every time information is entered when registering for an account via a final button click, the state of these inputs is sent to the server and ultimately the back-end, and the state of these inputs is reset and ready to accept other inputs if necessary.

As described, once users input their data, such as on the Login Screen for example, all relevant fields are saved to variables local to the front-end. From here on, the button click registers a REST API call to make POST Request, sending this data. REST stands for Representational State Transfer. API stands for Application Programming Interface, and is a set of rules to allow programs to converse. A POST Request is a type of REST call that can “post” data to a receiving party. We use these everywhere throughout the application to seamlessly transfer data from the application to our Machine Learning model, and to the back-end server. For example, once the user fills out their profile information, which is ultimately sent to the server, entering the back-end using these REST Calls, a click of a button can display all profile information by making a GET request to our back-end running on an EC2 server. EC2 is Amazon’s Elastic Compute Cloud which allows for scalable computing capacity and capabilities. By using an Elastic IP address we can run a server which can be reached from anywhere in the world. Therefore on the button click we can tell the server that we want to receive specific data.

From here the node server running on an EC2 instance will interact with our database. The database is also an Amazon Service, specifically it is an AWS RDS private endpoint. RDS stands for Relational Database running on MySQL architecture, an open source relational database, allowing us to relate users to all sorts of data concerning their dog, and ultimately their friends. It is configured with a private security group sharing a VPC (logically isolated virtual private cloud) with the public facing EC2 server such that connections are made indirectly between data entering the back-end and then entering the database. This arrangement is more secure than a direct connection alone, as it does not expose the login credentials. By pulling data from the database, it is that same GET request that can seamlessly show the relevant information from their profile, friends, device, and data collected. The same functionality works in terms of REST API calls interfacing with our node server in EC2 and ultimately the database, for login, retrieving friends predictions, adding friends, and keeping track of friends for chat and maps.

Another main functionality using REST APIs as mentioned above is retrieving predictions. Once the ESP32 makes a POST call to the ML Model, with the relevant ‘.wav’ file including the dog bark, this ML Model is able to produce a prediction and associate the prediction with the source of the POST call. This prediction, which includes the name of the other dog, associated using the other device’s MAC address, is sent directly to the back-end server using a POST request. The node server receives this prediction and stores it in the database. From here the user can click a button on the recommendation page to receive and view the prediction. This button click is utilizing another GET Request, which sends this request to the server, pulling this prediction from the database, and displaying this data efficiently to the user.

From here the other major two components of our application which really provide to our social media are the map and chat components. Both of these components utilize Scaledrone, a realtime messaging service that allows us to send real time data to our users. This software is very important for components like chat and maps, and Scaledrone makes it super easy to connect users by placing authorized users in the same rooms, where they are able to interact with each other. The scaledrone component connects by utilizing the specific room per group of friends.

The Maps component utilizes React-Native-Maps, which is based on Apple Maps for iOS. Utilizing GPS data from the hardware component, we are able to display friends’ locations on the same page to allow users to go to locations where their friends are so that their dogs can interact and play with each other. The map locations are sent to the node server as well in order to ensure authentication.

The Chat Component utilizes React Native’s FlatList to create a list of messages sent from users to one another. Each incoming message is associated with a specific username and ID. This process allows the application to know whether or not the message is incoming from a friend or from yourself. Each message is typed into an ‘Input’ which registers, populates an array, and is sent to the Scaledrone instance. From here there is a function to analyze where the message and who the message came from, in order to properly display the message in the chatRoom on each respective user’s device.

# Relevant Engineering Standards

The *Woof!* Jacket has a water resistance rating of 50 meters under International Organization of Standardization (ISO) 22810:2010. This means that it may be used during light precipitation. However, as stated in Section 3.3, it is not recommended. The Jacket similarly follows the National Electrical Safety Code (NESC). Primarily, far less than 100 volts is used, which means the *Woof!* Jacket is classified as Low Voltage, meaning there is no requirement for licensing, training, or certification of installers, and no inspection of completed work is required. Additionally, this device is compliant with Article 210 of NESC, which addresses branch circuits - which are not used. Sections such as that of Conduit and Cable Protection and Temperature Rating are additionally satisfied, as all microcontrollers and sensors used in the design of the *Woof!* Jackets are compliant.

There are several coding standards that were needed to follow when developing the React Native application, such as naming conventions, layout conventions, commenting conventions, and especially language guidelines, including class components for the front-end user interface, and primary and foreign keys for consistent fields associated to be shared between the SQL tables. In terms of collaboration planning, we follow Agile project management, in which we constantly receive feedback from supervisors and sample end-users, while documenting and assigning tasks in an organized group message dedicated to this product.

HTTP is a client-server protocol, as requests are sent by one entity and received by another. An HTTP request is made by a client, and it is sent to a host located on a server. The aim of the request is to access a resource on the server. In our case these resources are either the database, or the ML Model’s predictions. In order to make the request, the user uses a URL (Uniform Resource Locator) to include the relative information needed to make the request, including a request line, a HTTP header, and sometimes a body if needed. More specifically GET API requests are used to retrieve resource representation/information only. POST API requests are used to create new subordinate resources.

These Requests are types of REST APIs, as they conform to the necessary RESTful architecture, including having a uniform interface, following a client-server architecture, being stateless, cacheable, and layered.

Furthermore, authentication for the Map Component is done using JSON Web Token. JWTs are an open standard (RFC 7519) that utilize compact ways for safely transferring information between parties. This is done using JSON objects that are verified and can therefore be trusted for authentication.

# Cost Breakdown

| Project Costs for Production | | | | |
| --- | --- | --- | --- | --- |
| Item | Quantity | Description | Unit Cost ($) | Extended Cost (1500 users) |
| 1 BLE module | 2 | Adafruit Bleu-art | $17.50x2 | $52,500 |
| 2 GPS module | 1 | Tile | $11.99 | $26,985 |
| 3 Microphone | 1 | INMP441 | $10.88 | $16,320 |
| 4 Wifi supported microcontroller | 3 | ESP32 | $11x3 | $49,500 |
| 5 Power management | 6 | CR123 3V lithium coin batteries | $21.85 | $32,775 |
| 6 Vest | 1 | Hardware housed in a water resistant vest | $50 | $75,000 |
| 7 Database | 1 | AWS RDS Cloud MySQL | free/$108 | $98 up front, $8 per month |
| 8 Back-end server | 1 | Cloud AWS EC2 | free/$90 | $90 per 10 TB per month |
| 9 Machine learning server | 1 | Makes REST API calls on EC2.xlarge | $90 | $90 per 10 TB per month |
| 10 Messaging, location sharing | 1 | Scalendrone suite | $9 per 20,000 events/day | $29 per 1,000,000 events/day |
| 11 Front end user interface | 1 | Vanilla React Native | free | N/A |
| 12 Total: hardware | 16 | All hardware | $184.71 | $262,065 |
|  |  |  |  |  |
| 13 Total: software | 5 | All software | $297 | $317 |
| 14 Total: total | 21 | All items | $481.71 | $262,382 |

# Appendices

## Appendix A - Specifications

| **Product/System** | **Specifications** | **Performance** |
| --- | --- | --- |
| Adafruit Feather ESP32 | Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz | Used to power INMP-441 to collect/ send microphone data using its onboard file system and WiFi module. |
| IMNP-441 | MEMS sensor, signal conditioning, an analog-to-digital converter, anti-aliasing filters, power management, and an industry-standard 24-bit I²S interface. | Used as the primary device for collecting audio data via connection to ESP32. |
| Tile | BLE Beacon that triggers bluetooth connectivity when near another device. | Used as the main tracker component via connection to EC2 Server. |
| AWS RDS Cloud MySQL | O[pen source](https://aws.amazon.com/products/databases/open-source-databases/) relational databases- set up, operate, and scale MySQL deployments in [the cloud](https://aws.amazon.com/what-is-cloud-computing/). | used as the database to store user data |
| Cloud AWS EC2 | part of Amazon.com's cloud-computing platform, Amazon Web Services, that allows users to rent virtual computers on which to run their own computer applications. | used as the backend server to query data requests |
| Scalendrone Small/Medium | realtime messaging service and platform. Send live updates, create chat rooms and collaborative tools | used for messaging feature and sharing locations between users |
| Vanilla React Native | ​​open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities | used as our front end to display data from backend server |

## Appendix B - Sources

[1] Rooney, Nicola J, John W.S Bradshaw, and Ian H Robinson. “A Comparison of Dog–Dog and Dog–Human Play Behaviour.” Applied Animal Behaviour Science. Elsevier, January 17, 2000.

https://www.sciencedirect.com/science/article/pii/S0168159199000787.

[2] Carroll, Linda. “Your Dog May Love You, but Doesn't Love the Sight of Your Face, Study Finds.” NBCNews.com. NBCUniversal News Group, October 5, 2020.

https://www.nbcnews.com/health/health-news/your-dog-may-love-you-doesn-t-love-sight-your-n1242079.

[3] Sommerville, Rebecca, Emily A. O’Connor, and Lucy Asher. “Why Do Dogs Play? Function and Welfare Implications of Play in the Domestic Dog.” Applied Animal Behaviour Science. Elsevier,

September 20, 2017. https://www.sciencedirect.com/science/article/pii/S0168159117302575.

[4] Chambers, Robert D., Nathanael C. Yoder, Aletha B. Carson, Christian Junge, David E. Allen, Laura M. Prescott, Sophie Bradley, Garrett Wymore, Kevin Lloyd, andScott Lyle. “Deep Learning Classification

of Canine Behavior Using a Single Collar-Mounted Accelerometer: Real-World Validation.” bioRxiv. Cold Spring Harbor Laboratory, January 1, 2020.

https://www.biorxiv.org/ content/10.1101/2020.12.14.422660v1.full

## Appendix C – Team Information

Nafis Abeer - Will be studying part-time in the ECE master's program at Boston University. Full-time job at Raytheon Technologies.

Justin Lam - Will be working for General Dynamics Electric Boat as an entry level Systems Engineer.

Chase Maivald - Will be a full-time software engineer in Tampa, FL doing IT contracts on the side.

Daniel Shimon - Will be working for MuleSoft - Salesforce as a Technical Consultant focusing on API Integration.

Rajiv Ramroop - Will be working for General Dynamics Electric Boat as an entry level Systems Engineer.