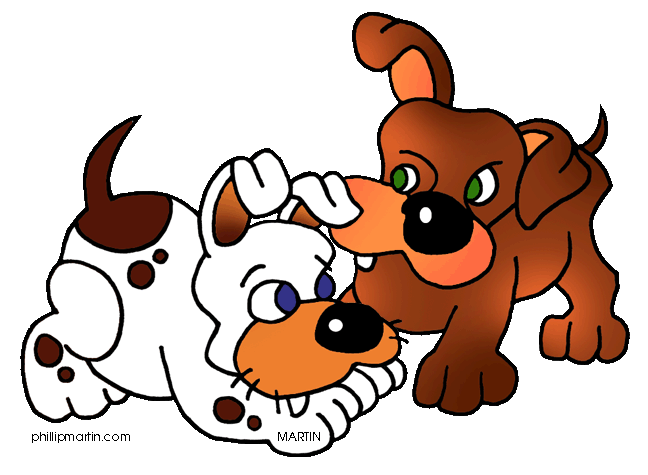
**Boston University**

**Electrical & Computer Engineering**

**EC464 Senior Design Project**

*2nd Prototype Testing*

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)

Testing Day: March 4th, 2022

**Content**

1. Required Material ………………………………………………….. 3
2. Demonstration Set Up
   1. Set Up ………………………………………………………… 4
   2. Pre-Testing Setup Procedure ………………………………… 4
3. Testing Procedure and Success Criteria
   1. Testing Procedure…………………………………………….... 6
   2. Measurable Criteria for Success ……………………………… 8
4. Data Flow Chart
   1. User Interface Screen Flow …………………………………….. 9
   2. End to End Data Flow ………………………………………….. 9
5. Suggestions

Required Materials

* Adafruit ESP32 Feather - microcontroller
* INMP441- microphone sensor
* jumper wires (x5)
* Python Flask - Backend ML API (Hosted on t2.xlarge ec2)
* Node JS - EC2 Server
* React Native - User Interface
* MySQL Amazon RDS - Database
* Wireless Router w/ ethernet cable - internet connection

Demonstration Setup

2.1 Setup

Our setup consists of wiring up our microcontroller to collect audio and connecting it to the internet. Our Machine learning API is stored in an EC2 server ready to be bootstrapped. Our node.js server is also in a public facing EC2 server, but in a smaller instance, and it is ready to be deployed. Its security group is configured to accept connections to the private MySQL AWS RDS instance based on inbound rules allowing traffic over EC2’s elastic IP address with port 3306. Our User Interface is running on react native on one of our member’s laptops, and it displays an iOS front end using an emulator. The UI connects to the backend through REST API calls.

2.2 Pre-testing setup procedure

We will set up our testing environment to be able to show end-to-end data flow. We will show data being collected with our microcontroller making an ML API call to receive a prediction. The ML API makes a post call with the username and prediction to our node server. The node server stores the prediction associated with the user’s identification number in the database. Our front end will perform a GET request to the server to display the new prediction.

* Set up ESP32
  + Wire up microphone
  + Connect to internet router
* Set up ML API
  + Connect to EC2 t2.xlarge instance
  + cd Senior/Design/model\_flask
  + ./bootstrap.sh
* Set up Node.js server
  + Connect to EC2 t2.micro instance
  + cd node
  + node ./server/index.js
* Set up Main Application User Interface
  + cd into integTestOne
  + npx react-native start
  + npx react-native run-ios
* Set up Chat Application User Interface
  + npx react-native start
  + npx react-native run-ios
  + cd server-directory
  + node index.js
  + node rand\_moves.js
* Sending data
  + Set up postman on local device hosting ESP32

Testing Procedure and Success Criteria

Our demonstration will proceed as follows:

*ESP → Audio → ML API → Node.js Server → Database → Node.js server → UI*

1. We will connect our ESP32 to the INMP441 via jumper wires, and connect it to a laptop containing code to record audio and download the corresponding .wav file.
2. We will then playback the audio to confirm we successfully captured the data.
3. We will make a POST call with the collected .wav file to the t2.xlarge EC2 instance hosting our FLASK ML API running the flask ML API, using postman
4. The ML API receives a call with a ‘file’ and ‘username ’indicating a file and string, and it runs our prediction algorithm on the file. It then creates a new json object with the prediction and username for another POST call before returning the prediction back to Postman.
5. From here the ML API also makes another POST call to the node.js server running on EC2. This call hits an endpoint which directly uploads the information passed through to the MySQL server.
6. As proof, we will both show the response in the postman console, however more importantly we will show the node.js server’s console as well as the mySQL database as it is populated with the incoming information.
7. Now, the UI will go to the ‘Friends’ Page and include the dog’s name in the requested field. From here hitting the receive classification button will show the classification in the terminal. Hit the show classification button to display the classification correctly on the screen.

*UI → Node.js Server → Database*

1. Navigate to the ‘Profile’ Page and fill with the necessary information.
2. Register the user information to the Node.js server
3. Show the Node.js server’s console and how it received the information
4. Show the mySQL database and the newly populated data

*Map Component*

1. Enter the user’s name
2. Show the User’s location on the map (San Francisco because on an emulator).
3. Run the rand\_moves.js file to show the friends population the map and moving

*Chat Component*

1. Enter a new message and hit send.
2. Each message sent should be entered on the chat screen.

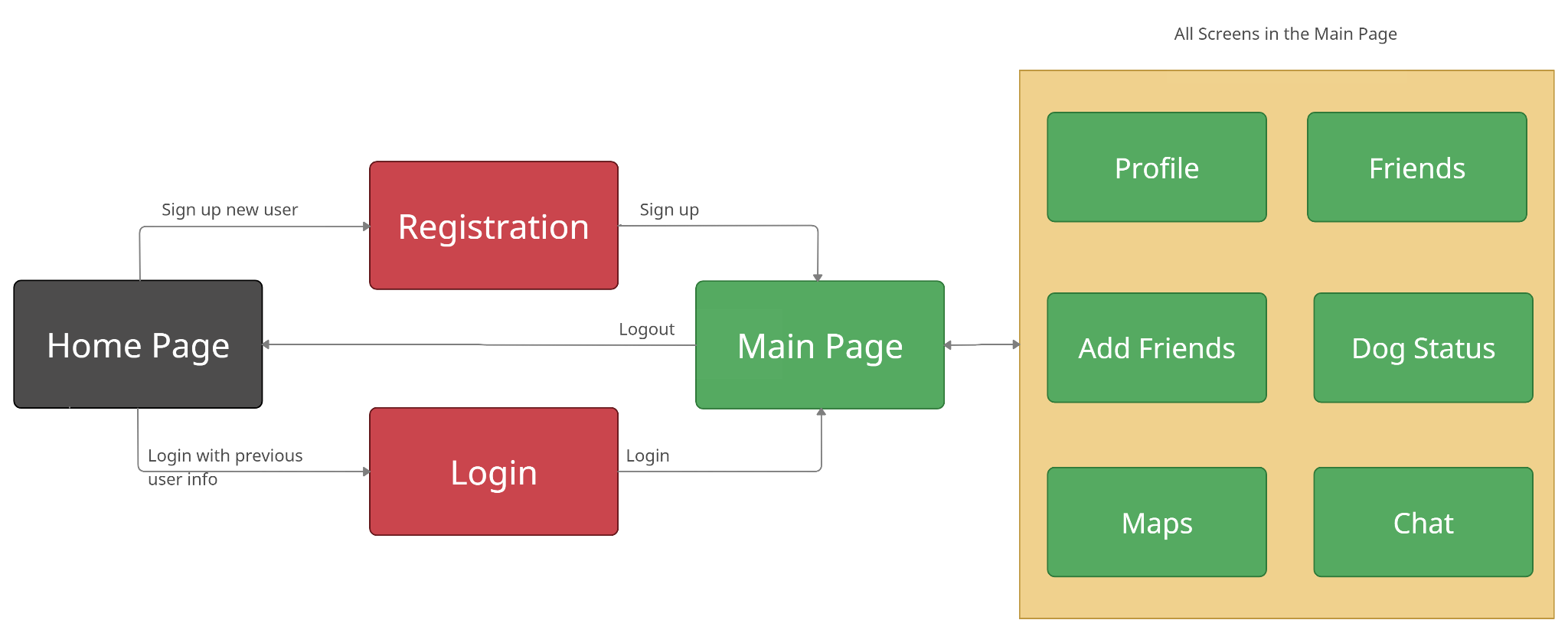
**Roles during end to end testing:**

1. Chase launches the node server on EC2 with npm start or node ./server/index.js
2. Chase launches the ml API bootstrap on ec2 by changing directory into model\_flask and running bootstrap.sh which exports Flask API, and it starts a virtual environment for the API
3. Rajiv collects sensor data
4. Rajiv downloads this sensor data
5. Rajiv sends post request to the ml api (this will return the prediction but more importantly run another post request to the node server
6. Node server receives prediction and stores it in database, with the dog’s name
7. (Run node server again) Dan enters friends page, enters the dog’s name, and hits receive classification. Node server receives call and retrieves information from database
8. Now we see the classification in the console
9. Press show classification and we will see the classification in the UI
10. Dan shows current state of Map + Chat components

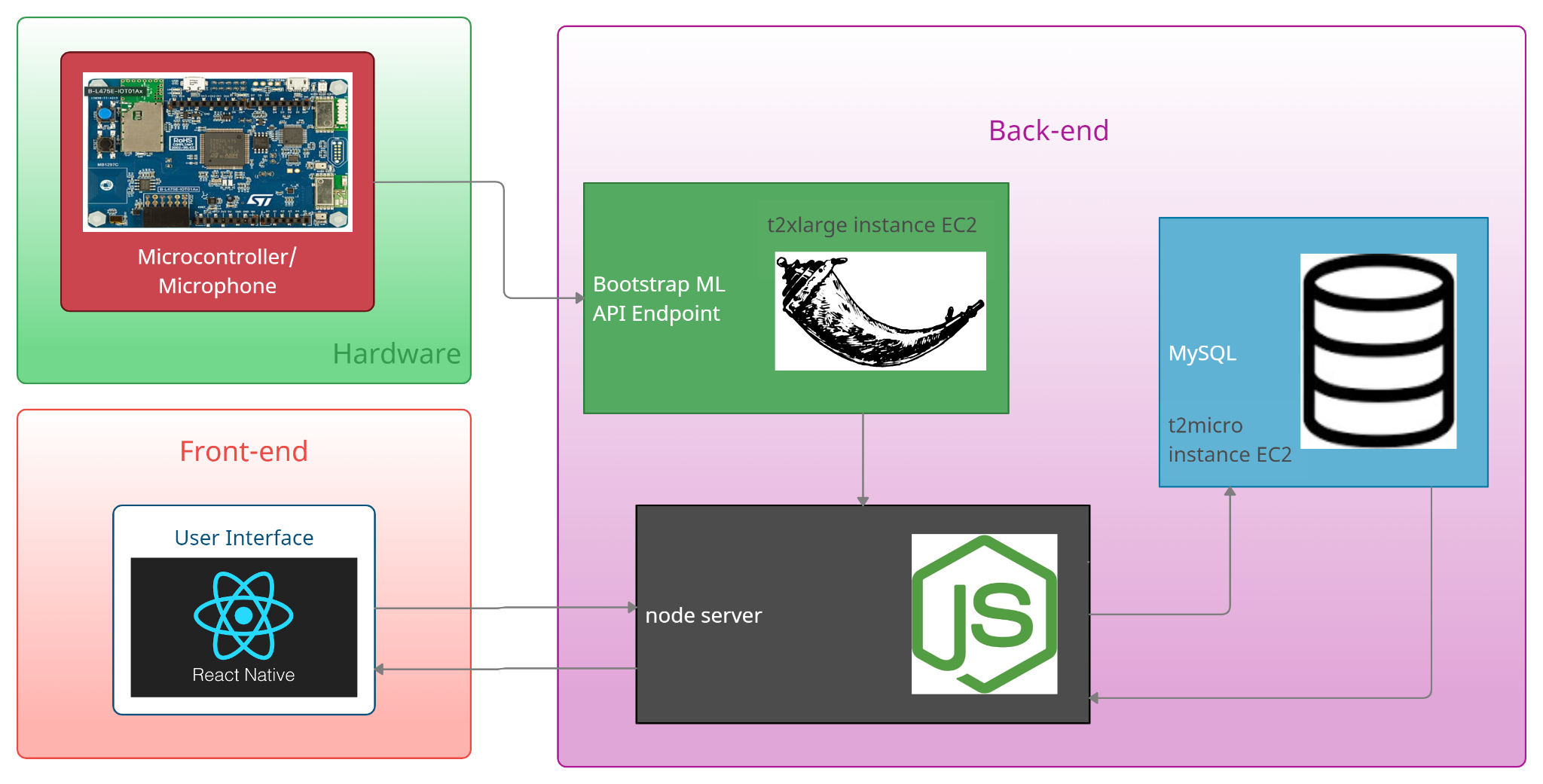
Measurable Criteria for Success

1. The user interface supports signing up for an account. It supports navigation to the separate pages of maps, profile, friends, chat. It has end to end connectivity to the database, hardware, and back-end to support saving and displaying profile and sensor data.
2. The database should support storing the password/login information associated with each profile, showing how it has passed from the UI to the server, and furthermore to the database.
3. The hardware should support recording audio files and uploading these files’ predictions to the back-end of the application.
4. The machine learning model should support returning real-time predictions from the hardware to the back-end, and through to the front-end. Its accuracy is such that dog barks, and related sounds can be classified appropriately.
5. The Maps component is able to render new friend’s as they are added, as well as supports the user’s current phone location.
6. The Chat Component is able to render new messages from the user as they are imputed.

4.1 User Interface Screen Flow



4.2 End to End Data Flow



5 Suggestions (Pisano)

1. The WiFi interfacing over our router should be interfaced with the BU Guest network
   1. We have been approved to use the router as is however, because it does not violate the use of BU’s resources and ESP32 being approved to use BU Guest may not be possible, ESP32 needs our custom router
2. SQL should be able to support persistent, simultaneous requests such that multiple data tables can have data inserted into them or data read from them at the same time
   1. As of 3/13, SQL is able to perform persistent, multiple conc current requests, which supports the future of executing many queries from many users
3. POST and GET requests should be performed within the code, instead of through the Postman GUI such that all data transfer is modularized
4. The hardware should be able to be mounted on the dog in some form of a collar or vest, such that dog safety and wearability is accounted for before ECE day
   1. As of the User manual report, we have designed the layers of a custom vest which will house all the sensors and ESP32s and be waterproof in light precipitation