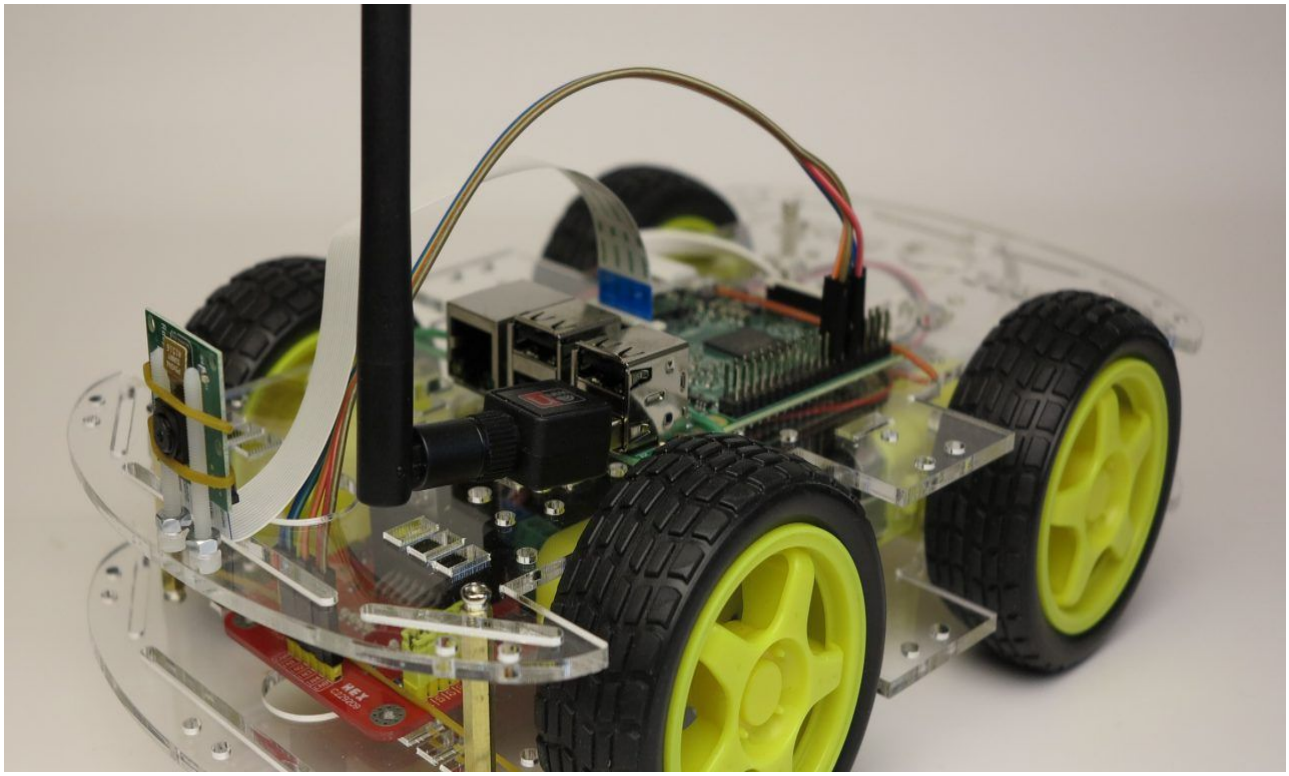


Terrain Exploration and Rescue Vehicle - TEARv

Design Document

Team 36

- *Rishabh Ramsisaria*
- *Xu He*
- *Dominic Miller*
- *Shaurya Sinha*
- *Rahul Balla*

Table Of Contents

Purpose

"Every year natural disasters kill around 90 000 people and affect close to 160 million people worldwide. Natural disasters include earthquakes, tsunamis, volcanic eruptions, landslides, hurricanes, floods, wildfires, heat waves and droughts. They have an immediate impact on human lives and often result in the destruction of the physical, biological and social environment of the affected people, thereby having a longer-term impact on their health, well-being and survival." (1)

First responders and rescue operatives put their life at risk when inspecting areas struck by natural disasters. The manual process of looking for survivors and checking for damage puts even more human lives at risk. There is a need for a safer solution and so, we propose a mechanized one to automate the scouting and post-disaster assessment process.

We present Terrain Exploration and Rescue Vehicle (TEARv), a Raspberry Pi powered remote controlled vehicle equipped with a camera to look around, multiple sensors to provide real time data, and a custom app with analytics capabilities also serving as the control center for the TEARv, which can be sent in to explore the disaster struck environment so that appropriate authorities may determine the best course of action for rescue and clean-up.

From our research, a specific commercial product targeted towards post-disaster management doesn't exist in the market today. The closest products available are basic RC cars with limited functionalities which have little to no room for customization or improvement. However, the TEARv with a Raspberry Pi as the main processing unit allows users to install different sensors such as proximity sensors, temperature sensors and RH sensors which enable specialization and customization of the car to achieve better functionality at a similar price point.

Functional Requirements

Mobile Application

Registration

As a user,

1. I would like to be able to register for a *TEARv* account using username, password, and email so that my information would remain personal to me
2. I would like to be able to recover both, my username and password, if I don't remember it so that I do not get locked out of my account using my registered email
3. I would like to have the option to change my username and password
4. I would like to be able to register multiple products under the same account to use for different purposes such as one for surveillance and another for testing

Video

As a user,

1. I would like to be able to receive a live feed from the vehicle so that I can see where I am going and navigate appropriately
2. I would like to be able to record videos in real-time so that I can capture important information
3. I would like to be able to store videos locally so that I can view them later also
4. I would like to view the history of previously recorded videos so that I can visit them later again

As a disaster rescue operative,

5. I would like the resolution to be good enough to analyze the video feed to determine structural integrity of the buildings and the rubble

Controllability

As a user,

1. I would like to be able to rotate the camera mounted on the vehicle so that I can get a 360° view of my surroundings
2. I would like to move the car up, down, right, and left through a mobile app so that I can maneuver the car through difficult terrain

3. I would like to have the ability the control the speed of the car through the mobile app so that I move the car in varying speeds (if time allows)
4. I would like the car to navigate autonomously from one place to another so that the user can focus on other tasks as well (if time allows)

Connectivity

As a user,

1. I would like to be able to connect to the car via wifi to be able to operate the car from a large distance
2. I would like to be able to connect to the car via bluetooth in case there isn't wifi (if time allows)
3. I would like the car to have its personal internet dongle to provide connectivity to the module (if time allows)
4. I would like the car to be able to return to the location of the phone it was connected to in case of a connection loss (if time allows)

UI & UX

As a user,

1. I would like the app to be neat and easy to navigate so that it does not take time to learn
2. I would like the temperature readings to be displayed live on the screen as part of the live feed
3. I would like the temperature reading to change colours according to the level of danger (red for high or unsafe temperatures, green otherwise)
4. I would like the app to report if any of the sensors are malfunctioning so that they can be replaced (if time allows)
5. I would like the vehicle to report back battery life as accurately as possible so that I can get the vehicle back to safety before the battery runs out (if time allows)
6. I would like to be able to have a GPS location reading so that I can know where the car is exactly at all times throughout its operation (if time allows)
7. I would like frequent software updates so that the vehicle can continue to run efficiently (if time allows)
8. I would like to be able to store data from sensors such as the temperature and relative humidity (RH) sensors and display them on the app in graphical form so that they can be analyzed easily (if time allows)

As a disaster rescue operative,

9. I would like the car to have the ability to report the average temperature readings of the surrounding area so that I know how hot the environment is

Hardware

Controllability of car

As a user,

1. I would like to be able to rotate the camera mounted on the vehicle so that I can get a 360° view of my surroundings
2. I would like the hardware to respond to movement commands quickly so that the actions and movements of the car happen in real time and can be seen through the camera

As a project owner,

3. I would like to reverse the polarity of the motors so that the car can move in all directions smoothly

As a disaster rescue operative,

4. I would like to have a grid laser displayed on the terrain in front of the car so that it can simulate depth perception and highlight cracks or bumps in the terrain (if time allows)
5. I would like the car to have the ability to sense surrounding rubble and not crash into them so that it does not get damaged (if time allows)

Modules

As a user,

1. I would like to have the car be easy to modify using additional sensors or add-ons

As a disaster rescue operative,

2. I would like to have the car equipped with flashlights so that I will be able to evaluate surroundings in low-light situations (if time allows)

Non-Functional Requirements

Architecture and Performance

As a developer,

1. I would like the application to run without crashing
2. I would like the backend to respond to requests as soon as possible (< 500ms)
3. I would like the application to be responsive to all user requests

Security

As a developer,

1. I would like the application to protect user's data against common exploits such as SQL exploits
2. I would like all the user requests to be authenticated
3. I would like to restrict access to the database only to the user and not to any public organizations

Appearance

As a developer,

1. (If time allows) I want the application to be aesthetically pleasing

Connectivity and Scalability

As a developer,

1. I would like there to be a stable wireless connection between the Raspberry Pi and the mobile app
2. (If time allows) I would like there to be a backup system, such as offline reverse route, if the wireless connection fails

Usability

As a developer,

1. I would like the user interface should be straightforward and user-friendly
2. I would like the color scheme to be visually pleasing so that the user can understand the graph easily
3. I would like to be responsive so that it can resize to devices of different sizes

Hosting and Deployment

As a developer,

1. I would like the frontend and backend to be able to maintain and updated separately
2. I would like the backend to be deployed to a remote server running at all times on the Raspberry Pi as long as the vehicle is running

Controllability

As a developer,

1. I would like to be able to reverse the polarity of the motors using an H-bridge to move the car in opposite directions
2. I would like to control the Pi through the android application after authenticating it using IP addresses and a unique serial number

Design Outline

TEARv is a remote controlled vehicle which allows its users to observe a live stream of disaster-struck areas and access pertinent data, like the temperature of the area. The data from sensors on the car will be stored locally on the Raspberry Pi attached to the car. Our project will use the client-server model, with one server serving requests from one client. The server will access databases which contain sensor data and user login information.

Software

1. Android Client
    a. The Android client, running on the user's phone, will be the interface to our system.
    b. The Android client will send movement commands to the vehicle to control its locomotion.
    c. The Android client will send and receive data to and from our server via HTTP requests. The data will be in the JSON format.
    d. The data retrieved will be filtered, formatted and visualized to the user to provide information, such as temperature fluctuations over time.

2. API Server
    a. The server will be the point of contact for the user-facing client to request and send information.
    b. The server will translate user input from the client to mechanical information to make the vehicle move appropriately.
    c. The server will accept HTTP requests using REST URI conventions.
    d. The server will query the database to retrieve requested information.
    e. The server will return any requested data in JSON format in an HTTP response.

3. Authentication Database
    a. The authentication database will contain the login information of all users.
    b. The authentication database will use Firebase to reside in the cloud, and enable user authentication and user persistence.

4. Local Database
    a. The local database will store all the data generated or collected by the vehicle during its use, like recorded videos and sensor data.
    b. The local database will reside on the Raspberry Pi itself, to enable fast retrieval of data.
    c. Typical data will include pictures and videos, temperature readings, and recorded input from other sensors.

5. Statistics Module
    a. The Statistics Module will simple transforms and statistics operations on data stored in the local database to provide useful data to the user.

Hardware

1. Raspberry Pi
    a. The Raspberry Pi will act as the all-in-one server and database storage system for the vehicle.
    b. The Pi will be connected to the Android client via WiFi and will receive HTTP requests.
    c. The Pi will directly control the motors to move the vehicle in a controlled manner.
    d. The Pi will have direct access to sensors and read and store the data generated by them.

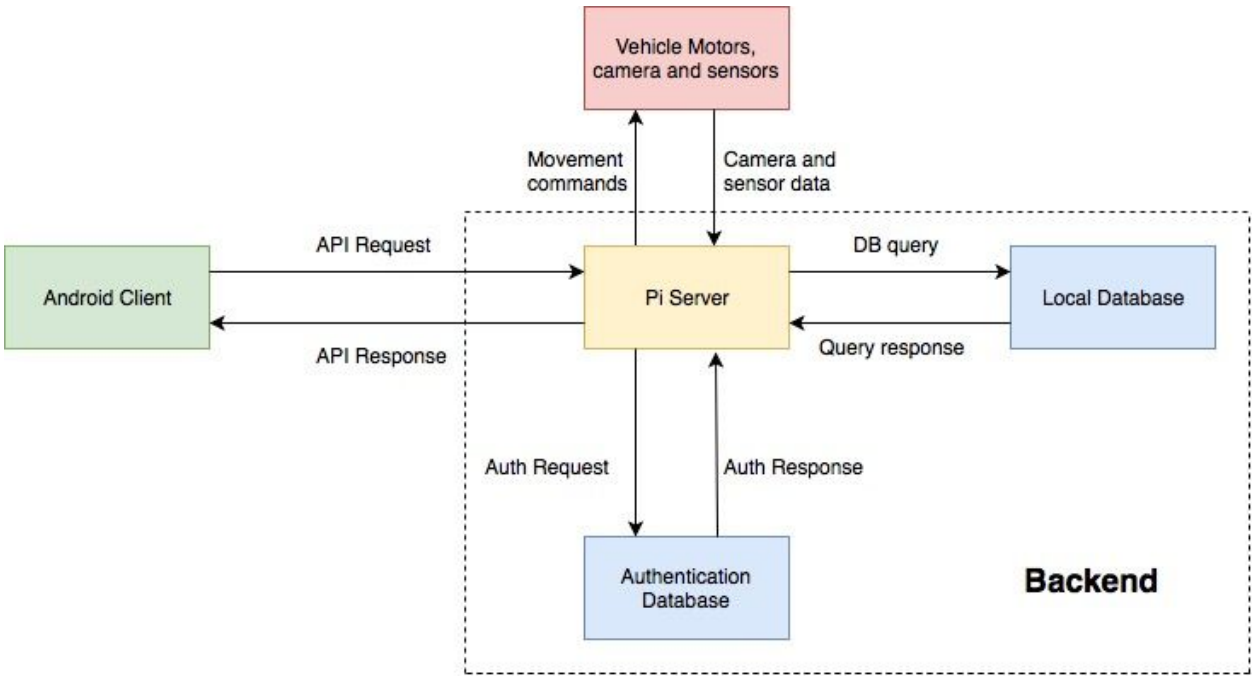
2. Sensors
    a. The sensors will be attached directly to the vehicle and send its input to be stored on the Pi.
    b. The user will be able to turn any sensors on and off from the Android Client.

3. TEAR vehicle
    a. The vehicle will be constructed out of cost-effective material and will be designed to be durable and heat-resistant.
    b. The vehicle will be controlled through input to its motors from the Raspberry Pi.

4. Camera
    a. The camera will be mounted on the vehicle and allow the user a 360 degree view of the vehicle's surroundings.
    b. The camera will be controlled by the user from the Android client and its movement will be facilitated by a motor attached to the Pi.

High Level Overview

The Raspberry Pi will act as the server with which clients can interact. This will be done using an Android client. The relationship between the Pi server and the Android client will be one-to-one because the vehicle will be designed to be controlled by only one user. The Android client will communicate with the Pi using a Representational State Transfer (REST) API. The API will allow the user to maneuver the vehicle, access data from the sensors and camera, and instruct the vehicle to save images and videos. The figure below demonstrates a high-level overview of the system:

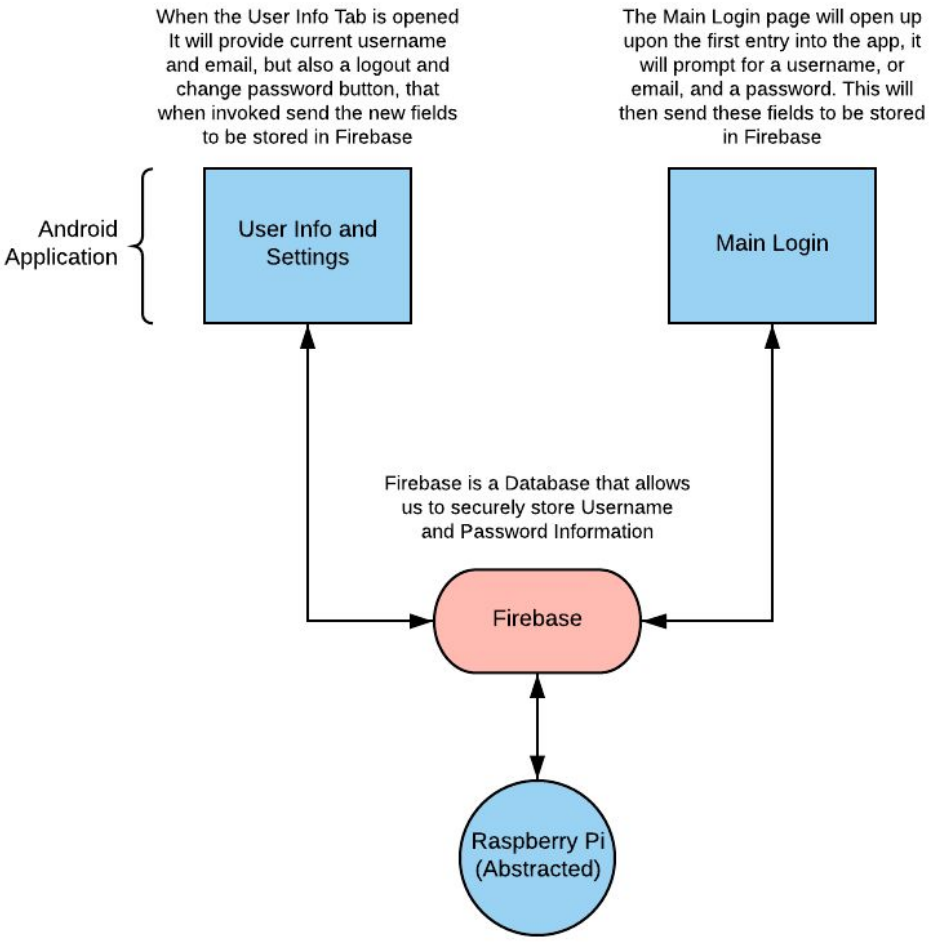


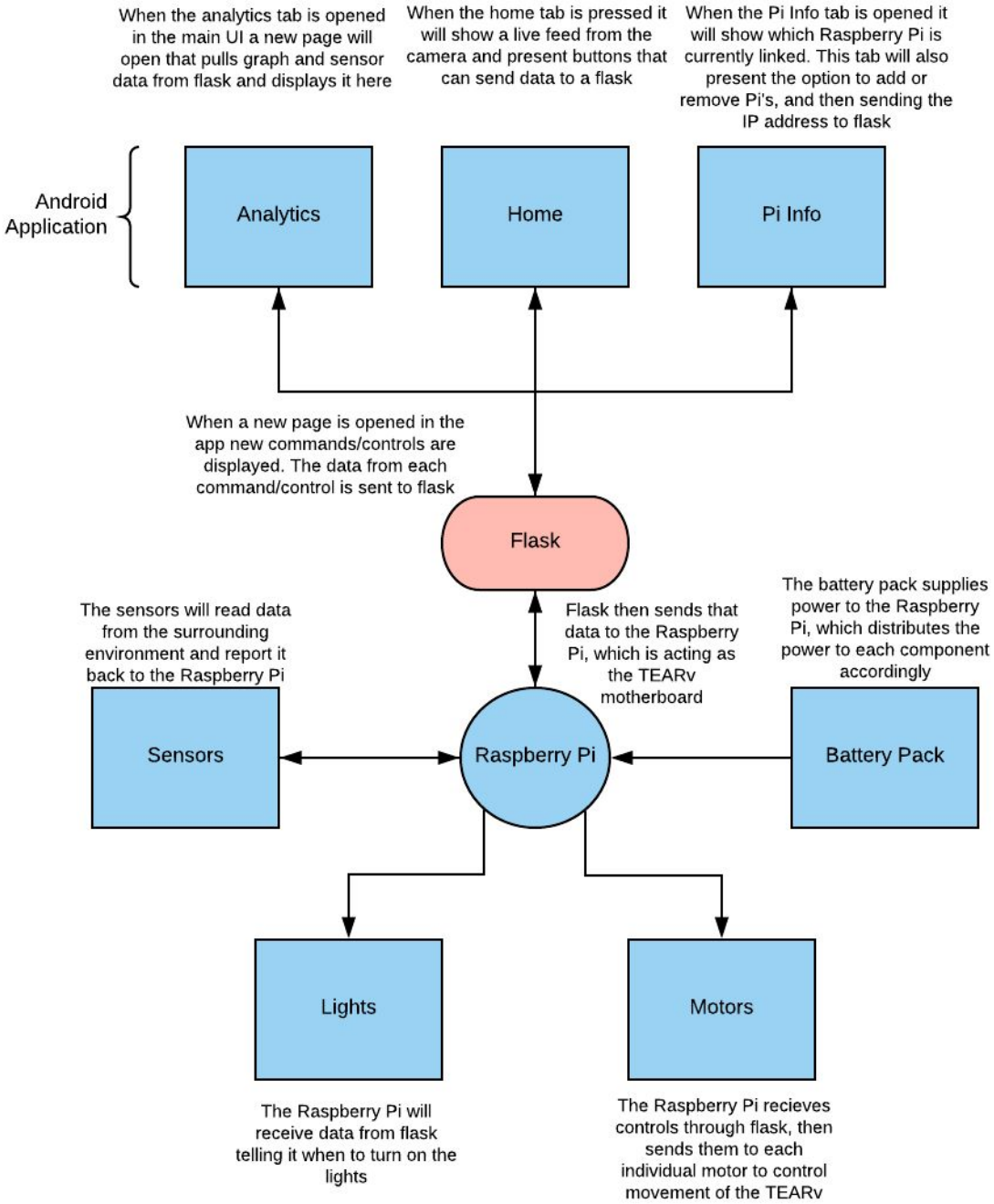
Note:

1. *The local database is just the internal storage of the Raspberry Pi*
2. *The authentication database refers to Firebase where the user info is stored*

Flow of Events

User interaction with the vehicle begins with the user opening the Android app on their mobile device. Users will begin by registering for a TEARv account if they haven't already. The next step is signing in, if they aren't signed in already. The app uses this one time authentication to decide whether to let the user proceed with using the app. The Android app will give the user full access to the vehicle's camera and sensors. The user can then decide to either access data from the sensors or move the vehicle. The server receives and sends all data and requests in JSON format through HTTP requests. The diagram below shows a typical flow of events:




When the analytics tab is opened in the main UI a new page will open that pulls graph and sensor data from flask and displays it here
When the home tab is pressed it will show a live feed from the camera and present buttons that can send data to a flask
When the Pi Info tab is opened it will show which Raspberry Pi is currently linked. This tab will also present the option to add or remove Pi's, and then sending the IP address to flask
Android Application
Analytics
Home
Pi Info
When a new page is opened in the app new commands/controls are displayed. The data from each command/control is sent to flask
Flask
The sensors will read data from the surrounding environment and report it back to the Raspberry Pi
Flask then sends that data to the Raspberry Pi, which is acting as the TEARv motherboard
The battery pack supplies power to the Raspberry Pi, which distributes the power to each component accordingly
Sensors
Raspberry Pi
Battery Pack
Lights
Motors
The Raspberry Pi will receive data from flask telling it when to turn on the lights
The Raspberry Pi recieves controls through flask, then sends them to each individual motor to control movement of the TEARv

Design Issues

Hardware Design Issues:

1. How should the body of the robot be created?
    a. Completely design the whole body of the robot from scratch
    b. Buy pre-made components of the robot and build it

Choice: b

Explanation: We decided to buy the robot kit because it contains all the necessary components needed to create the robot. This also helps decrease the time that would take to design the body of the robot and more time can be spent in developing the mobile app and establishing a connection with the Raspberry Pi. The robot kit also contains enough space to mount additional sensors on it.

2. How to implement the movement of the car?
    a. Unilateral
    b. Allow Polarity Reversal

Choice: b

Explanation: Having the car move in the reverse direction would help with the maneuverability of the car in areas that are difficult to reach. This would also help the car rotate around its axis and move in any direction. If the car moved in one-direction, the car would have to take a large turn to move in the opposition direction which is not the case if the polarity of the motors is reversed. An H-bridge is used to reverse the polarity of the motors.

Software Design Issues:

Functional:

1. Where should the recorded videos from the Raspberry Pi camera be stored?
    a. Raspberry Pi
    b. Cloud

Choice: a

Explanation: We decided that the recorded videos be stored in the Raspberry Pi because uploading the videos to cloud would take a long time. Since the videos would be in HD quality, the files would be large, and it would also take a long time for the app to request it from the cloud and show it to the user.

2. When should the app start recording the videos?
    a. Video feed is always recorded
    b. The user should decide when they want to record the video

Choice: b

Explanation: The Raspberry Pi camera is set up in such a way that all the video feed is recorded automatically. However, this feature needs to be turned off because the Pi would run out of memory quickly if the video feed is always recorded. Therefore, the user needs to decide when the recording should start to only record important videos. This will prevent the Pi from running out of memory quickly.

3. Where should we handle user authentication?
    a. Mobile app
    b. Raspberry Pi
    c. Both, Mobile app and the Raspberry Pi

Choice: a

Explanation: Handling user authentication on the mobile app would be sufficient because the user needs to be logged in to the app which would trigger a connection between the app and the Raspberry Pi. The app also gives the user the option to select which Pi he/she wants to connect to if he/she has multiple Pis registered on their account. Therefore, authenticating them again on the Pi would be redundant.

Non-Functional:

1. Which programming language would be the most appropriate for our backend services?
    a. Python
    b. JavaScript

Choice: a

Explanation: We decided to use Python because of its clean and easy to read syntax as compared to other programming languages. There are many built-in python libraries for the Pi which can be used for controlling input-output control. Python was also decided to be used as the backend for the mobile app to maintain consistency.

2. How many backend servers are needed for this project?
    a. 1
    b. 2

Choice: b

Explanation: Two backend servers are needed for this project. One, for the Pi to store data from the different modules that are mounted on the car such as temperature sensor, proximity sensor, and some other sensors. Second, for the mobile app to use endpoints to access data from the cloud.

3. Which database would best serve our purposes?

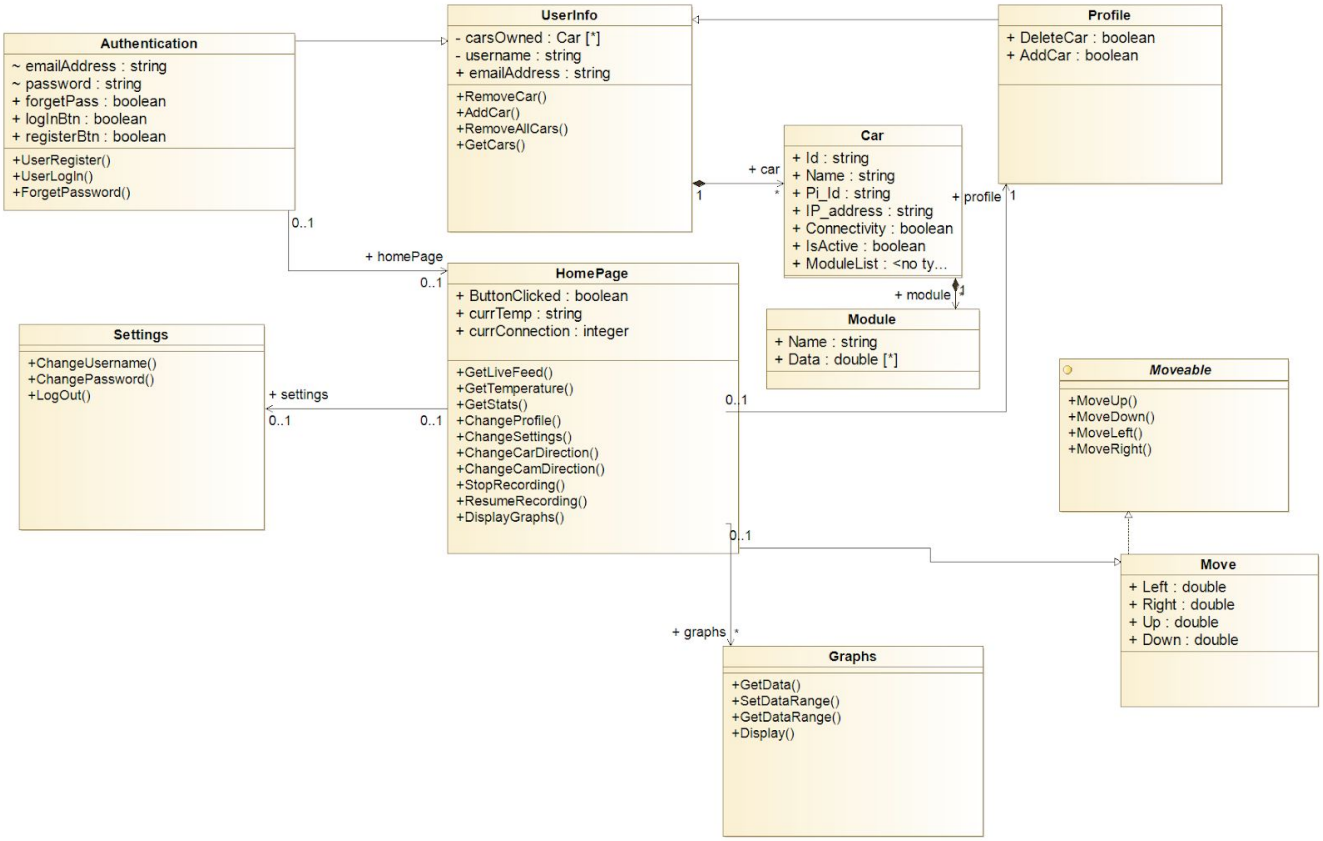
a. AWS
b. Firebase
c. Heroku

Choice: b

Explanation: Firebase is used in our project to handle user authentication. It is easy to use Firebase and we do not have to worry about coming up with our authentication system to protect the user's information. This allows to focus on building other features of the app.

Design Details

Class Design Diagram and Descriptions



Authentication

- Represents authentication system in the front-end
- Users enter their email address and password to log in to the application if the user has an account.
- If the login information is incorrect, an alert message will be displayed on the mobile app and then return the user to the login page again
- If the login information that has been entered is correct, user is taken to the home page

- Allows a potential user to register an account using a valid email address and with the provision of a suitable display name and password
- Connects to the back-end database to obtain/modify information that is associated with the user

User Info

- Represents the information that is specific to a registered user
- After the authentication is completed, the account's information is retrieved from Firebase and stored in JSON format in this class to be accessed by the front end
- The user has the option to modify their information such as changing their password and username
- The modified information is then updated in the database

Homepage

- It is the main page where the user can control the car through direction buttons and also navigate to other tabs such as Settings, Analytics , and User Info.
- The live feed from TEARv's camera , temperature reading, and the connection status from the Pi is displayed on this page
- The user also has the option to start and stop the recording of the video which will be stored locally on the Pi

Graphs

- The data collected by the sensors on the active car is displayed in graphs
- The data stored on the Pi in CSV files will be accessed via the Flask framework and displayed on the app under the analytics tab

Move

- This class represents all the information required for moving the car in a certain direction

- The 4 directions possible are forward, backward, left and right

Moveable

- An interface with abstract methods of directional change
- MoveUp will tell the Pi via Flask to move the wheels in such a direction so that the car moves forward. The other methods would work in equivalently.
- This interface is implemented to change the car direction as well as change camera direction

Car

- Represents a TEARv car
- Each car is associated with an account in the database
- This class contains unique information of each car that obtained from database such as its IP address, name, and connectivity status
- Indicates if current object or car is active on the app or not

Module

- Represents a module which includes the sensors connected to the TEARv such as temperature sensors
- The data collected from the sensors on the car is stored locally on the Pi

Profile

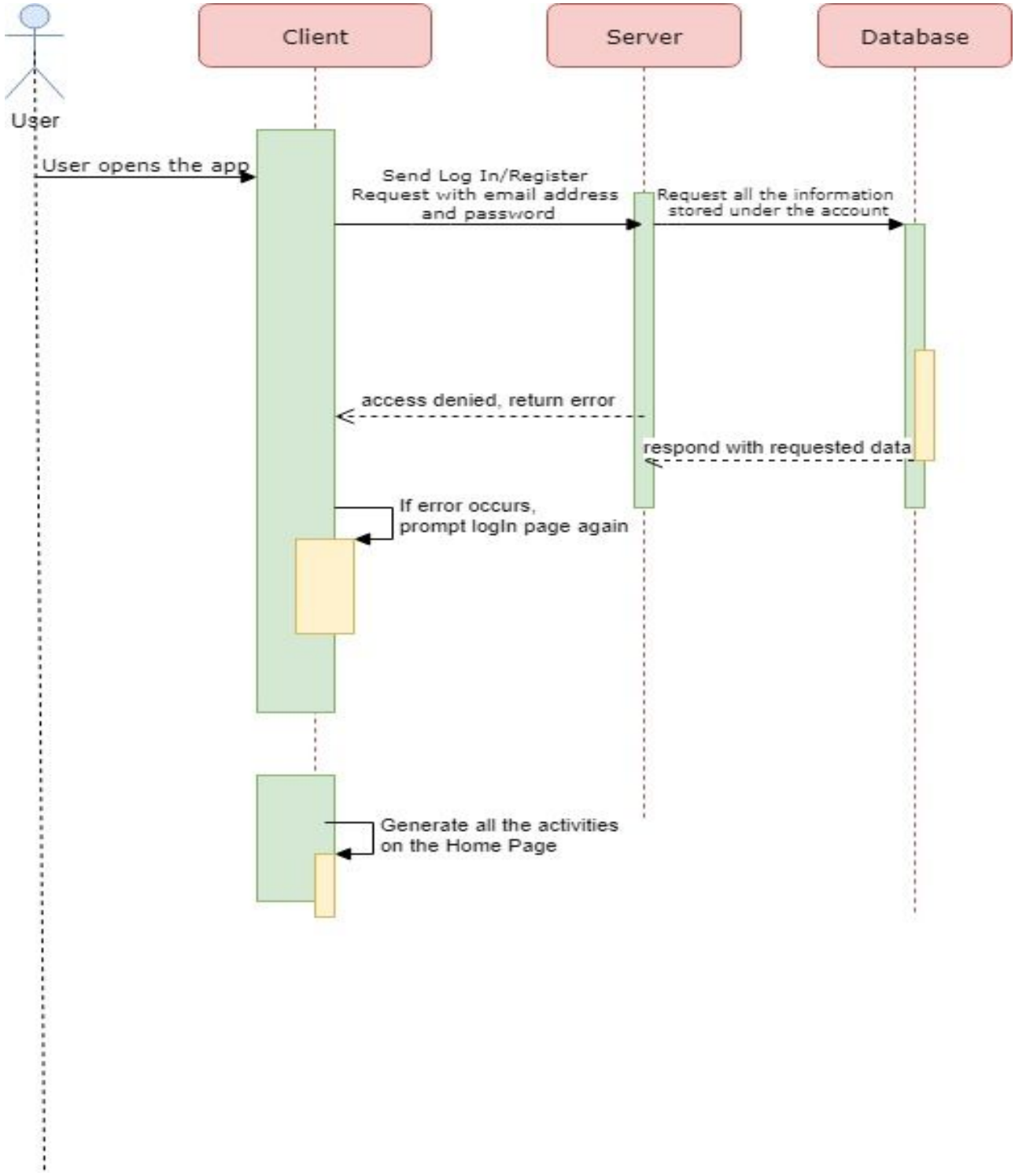
- Represents a list of the cars associated with the current user
- Information such as the IP address and name of all the cars is retrieved from the database and is displayed in a list
- Allows user to add and delete a TEARv car associated with the account
- The car list is updated in the database when the user adds or deletes a car

Settings

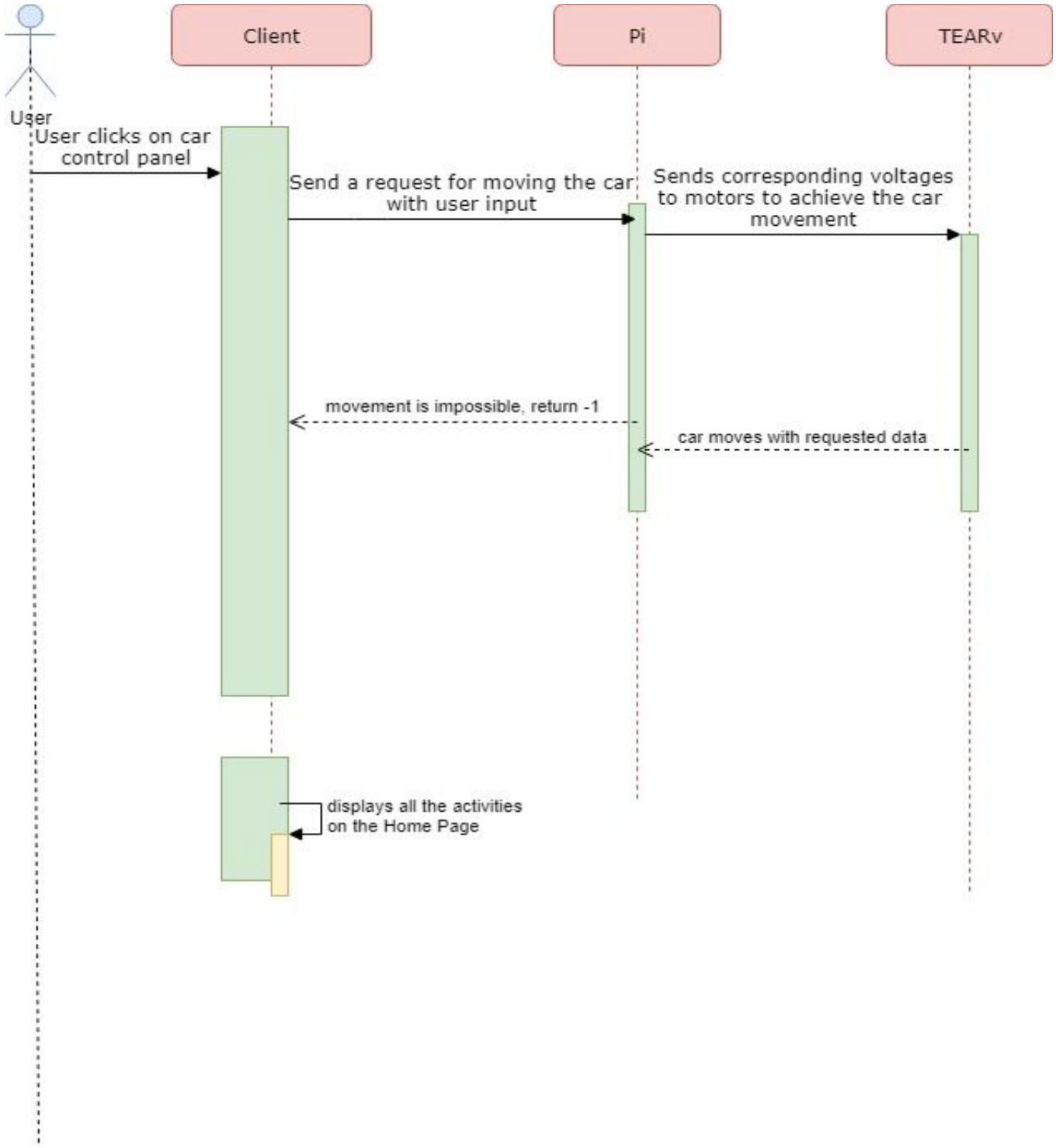
- Represents the Settings page in the application
- Connects to the database and executes requests such as password change
- Listeners for Logout, Change Password and Change Username options are included in this class
- Change Password and Change Username prompt the user to enter the old and a new password or username respectively, which will also be modified in the database
- Clears all the local data stored and disconnects the current TEARv and database when Logout is requested

Sequence of Events Diagrams

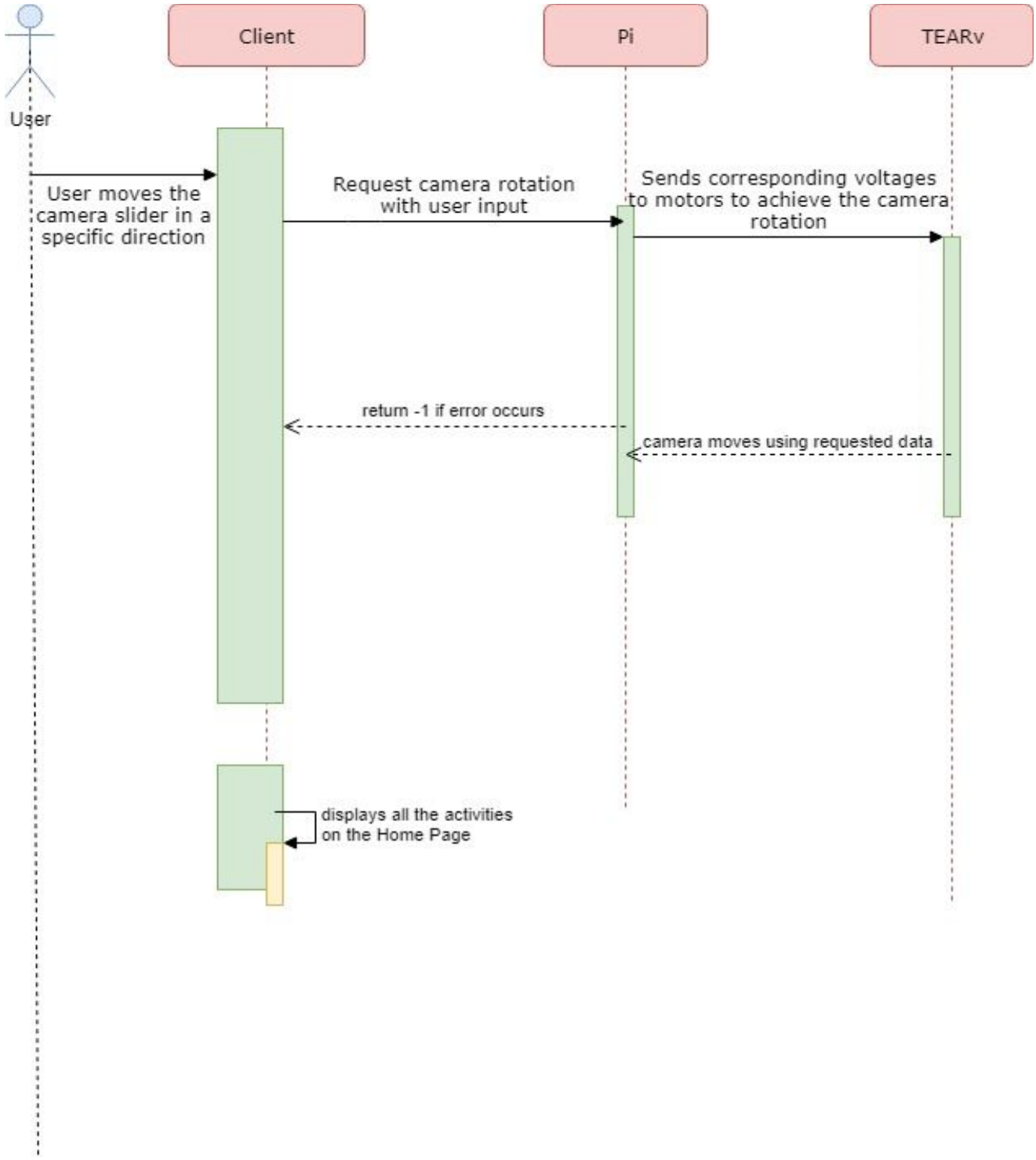
Login



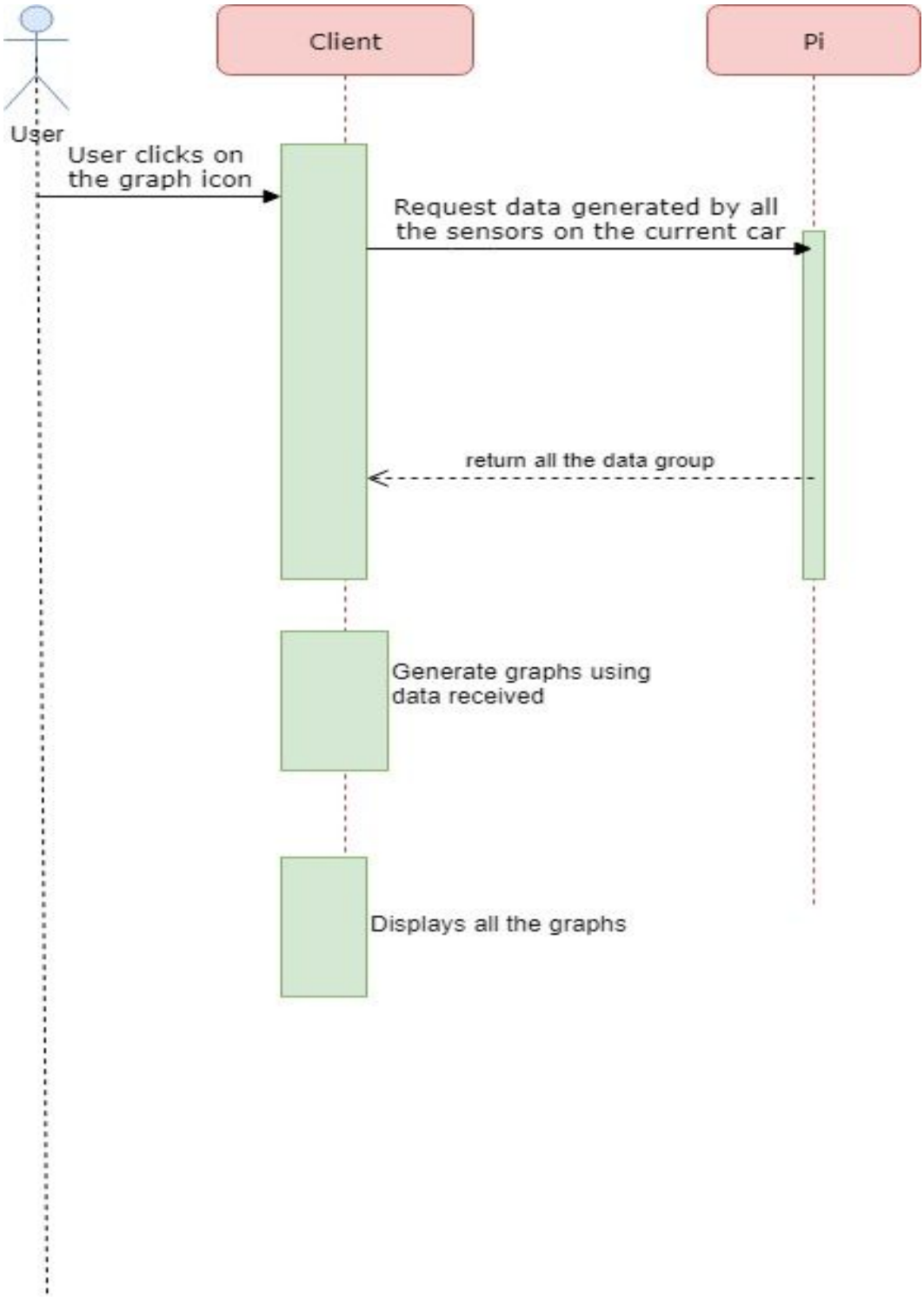
Car Control


Client
Pi
TEARv
User
User clicks on car control panel
Send a request for moving the car with user input
Sends corresponding voltages to motors to achieve the car movement
movement is impossible, return -1
car moves with requested data
displays all the activities on the Home Page

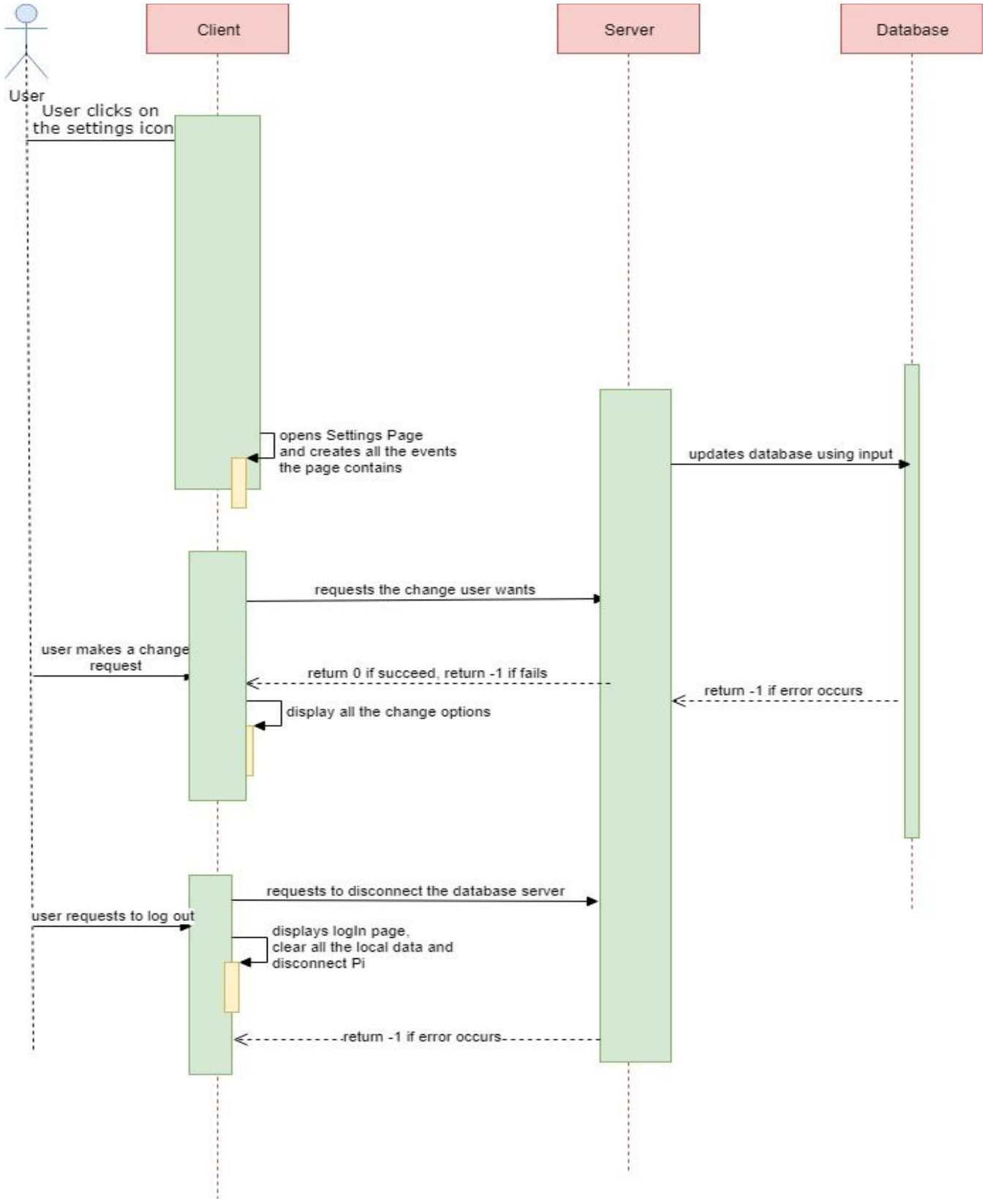
Camera Control



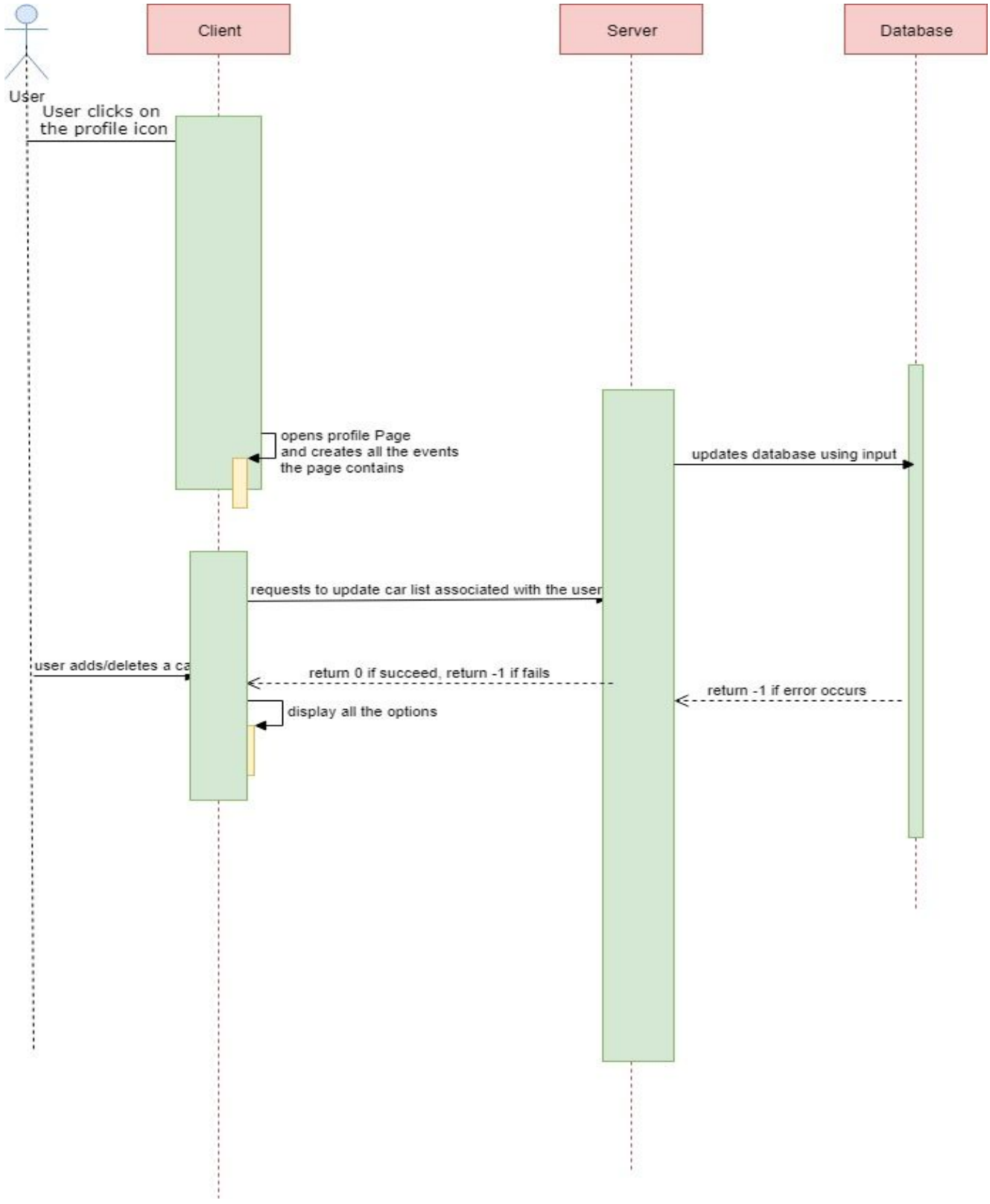
Analytics


Client
Pi
User
User clicks on the graph icon
Request data generated by all the sensors on the current car
return all the data group
Generate graphs using data received
Displays all the graphs

User Settings



Pi Info Changes

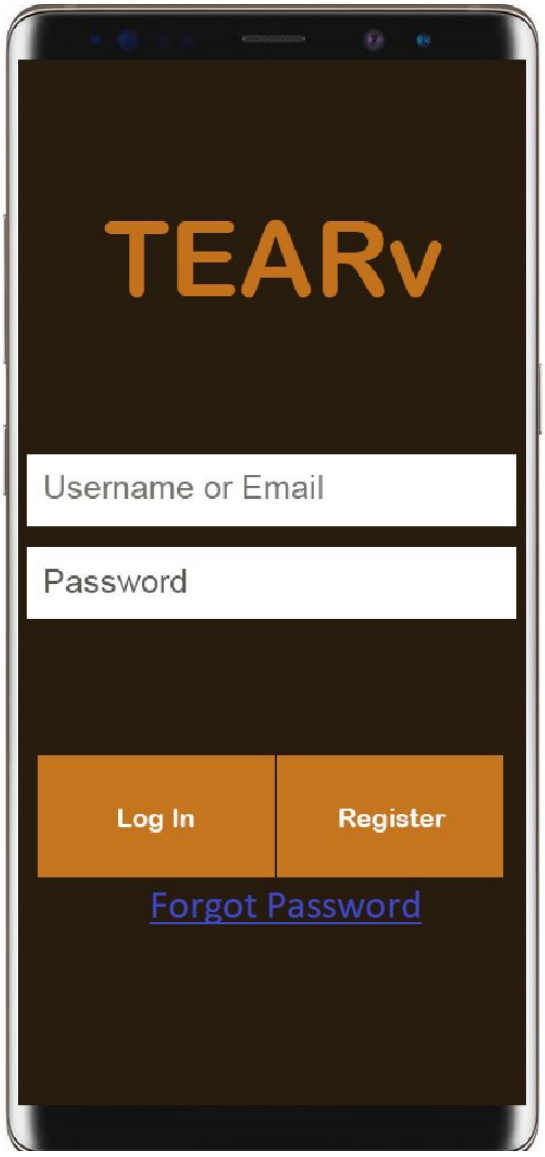


UI Mockups

Login Screen

The first screen the user sees after downloading the app containing:

- Text fields for entering email and password to log in
- Forgot Password button
- Register button



Home/Main Screen

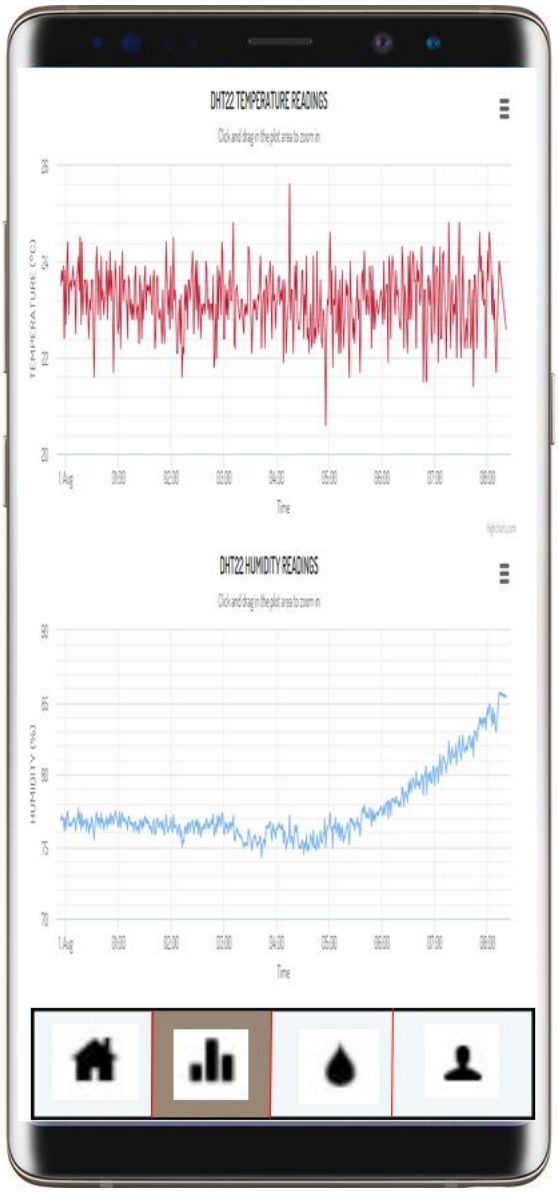
The main screen of the app which displays:

- The live feed from the camera as well as sensor readings
- Connection Status and active Pi name
- Controls for the car
- Controls for the Camera
- Different tabs of the app for navigation
- Record/Pause button

Analytics Tab

The analytics tab displays:

- All the graphs of the data gathered from the sensors on the Pi (temperature, humidity, etc.)
- Descriptive statistics on the data



Pi Info Tab

The Pi Info tab contains information about the TEARv's associated with the account that is currently logged in.

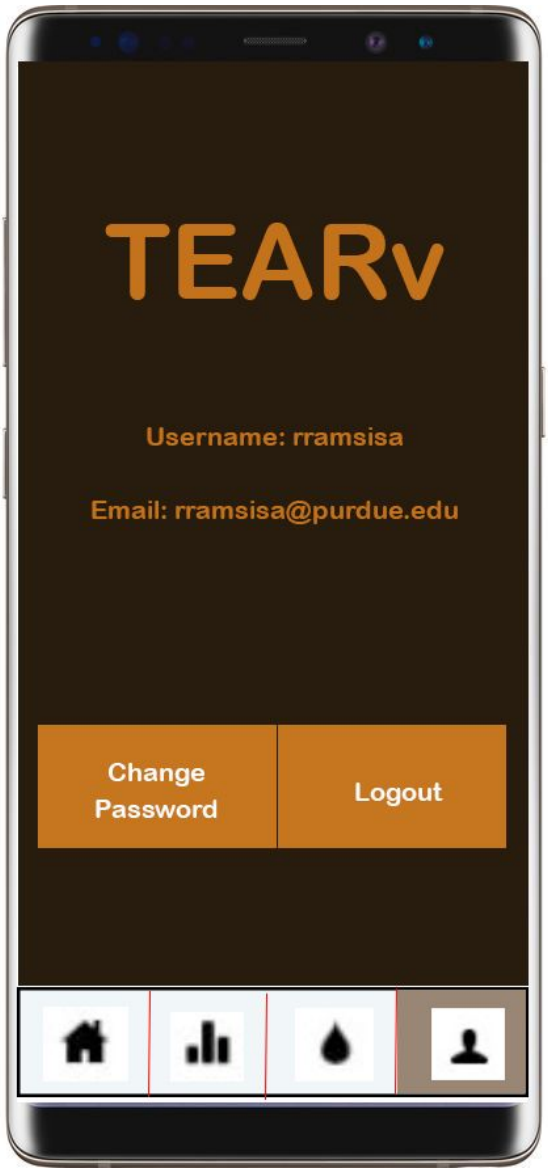
This information includes:

- Custom name of each TEARv
- IP address of each TEARv
- Status, i.e., which TEARv is currently being displayed on the home screen
- Add and Delete Buttons

User Info/Settings Tab

Contains the account info of the currently logged in account with the following buttons:

- Change Password
- Change Username
- Logout



References

1. Natural events. (2012, August 24). Retrieved from http://www.who.int/environmental_health_emergencies/natural_events/en/
2. Google Image Result for https://custom-build-robots.com/wp-content/uploads/2015/12/Smart-robot-car-chassis-Acrylic-finished-1280x640.jpg. (n.d.). Retrieved from https://goo.gl/images/5sP5Z3