Institute of Technology Carlow, Kilkenny Road, Carlow Bsc (Hons) Software Development

Home SecuriPi

Design Document

Author: Karl Redmond
Student Number: C00196815
Submission Date: 18/04/2018
Supervisor: Paul Barry

Abstract

The purpose of this document is to provide a detailed design specification for the Home SecuriPi product. It will outline how the system functions, and describe the overall interaction of the entire system. This document will enable any reader to understand the intricacies of the product.

Table of Contents

1. Introduction

Designing any software product involves a considerable amount of deliberation around how the software should ultimately function, and how the system will interact amongst its architecture, and also how the system should act upon user interaction.

This document will provide an architecture overview, database design, use case diagrams, detailed use cases and system sequence diagrams. The purpose of this document is to give the reader a deep understanding of how the entire systems provides its functionality specified in the functional specification.

2. System Architecture

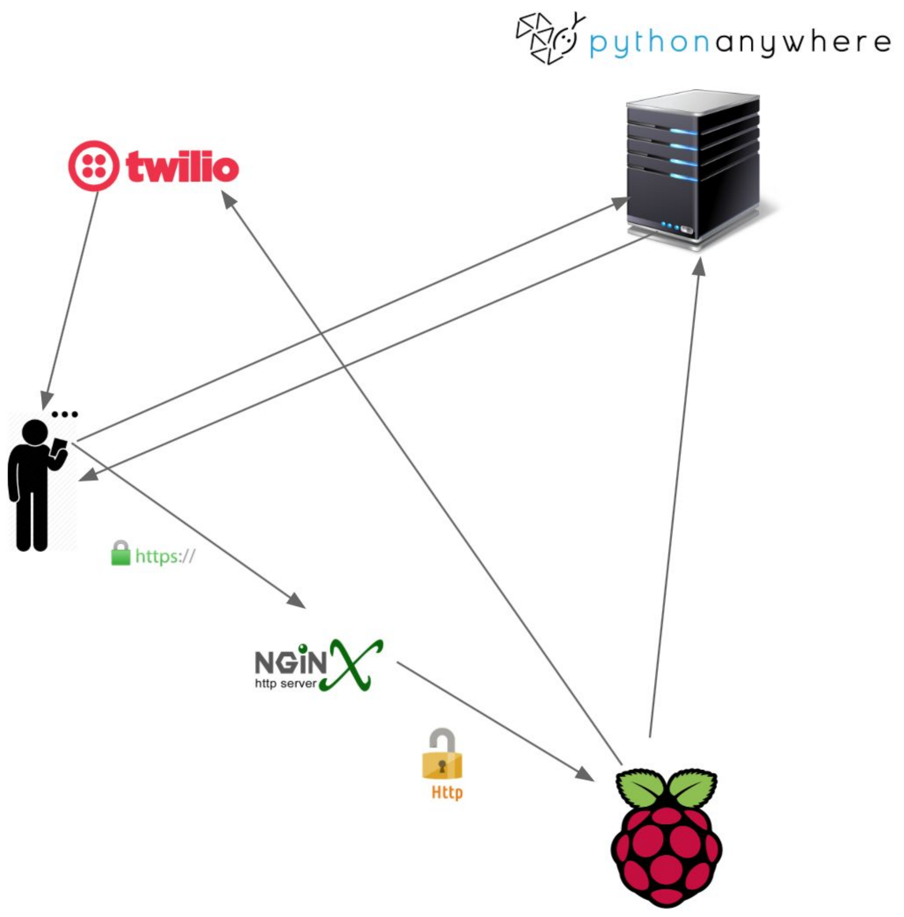


Fig. 2.1

The system centers around the Raspberry Pi mini computer which will be situated in a strategic position on ones property. With reference to Fig 2.1, the Pi will be running a Flask server which will be used to catch commands, via API endpoints, sent from an end users device. The device will be anything capable of running a web browser, like a mobile phone, laptop or tablet. A motion sensor wired to the Pi will be used to detect motion upon which a camera, also attached to the Pi, will be used to take a picture. This picture will be sent to a Flask server, via a HTTP request, running on Pythonanywhere which will store the picture for later viewing. The image will be repetitively sent to pythonanywhere until such a time that pythonanywhere can confirm that the image has been retrieved successfully. This is achieved by instructing the server to measure the size of the image received from the HTTP request, and respond to the request with the measurement. The Pi will then measure the size of the image which it has stored locally and compare it to the size from the HTTP response, if the sizes are the same the Pi will not submit another HTTP request, otherwise, the request will be sent again.
Once the image has been stored successfully, the system will utilize Twilios Platform as a Service (PaaS) to send an SMS message to a specified user(this could potentially be many users). The user can then use their device to retrieve the homepage of our application from Pythonanywhere, where they will be able to see the most recent image submitted to the server. From here our user can make a decision of what action they wish to take, such as issuing a verbal warning over a speaker attached to the Pi, or possibly turning on a light in an effort to insinuate that there is someone in the home, thus deterring a would-be burglar or intruder. The command sent to the Pi will be captured by the Flask server, which will utilize the hardware of the device to carry out the command.

2.1 Hardware Components

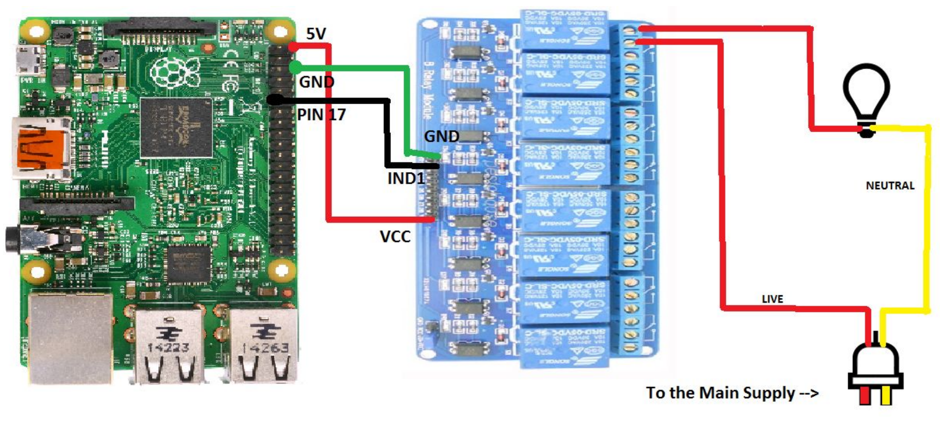


Fig. 2.1.1 [1]

The Home SecuriPi also has some circuitry involved which is displayed in Fig. 2.1.1. As the Raspberry Pi can only handle a maximum of 5V, a relay switch is needed for the appropriate control of higher voltage devices, in this case a light. The live feed of the light switch (110V) is broken and connected to the terminals of the relay switch, more specifically the bottom and middle terminals of Fig 2.1.2(provided for illustration, may not be exactly the same in practice). When it is desired to turn the light on, a HIGH digital signal is sent from the Pi's General Purpose Input Output(GPIO) pins. When this high signal gets sent, the coil gets magnetised thus pulling the switch closed(Fig. 2.1.2) and completing the circuit of the light, which will turn on. Similarly if the signal from the Pi is LOW, the coil will release the switch and break the circuit, turning the light off.

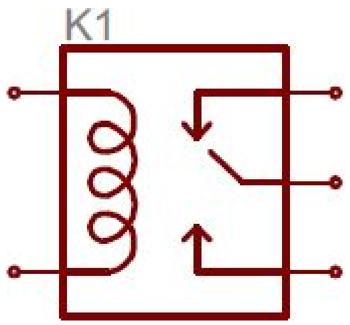


Fig. 2.1.2 [2]

The Home SecurPi project also has a 12V solenoid controlled door latch which works in much the same fashion as the light.

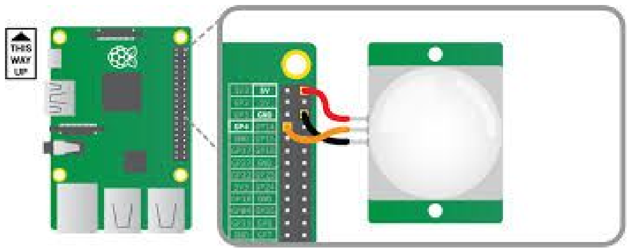


Fig. 2.1.3 [3]

A motion sensor attached to the Pi(Fig 2.1.3) signals whether or not there is any motion detected in an area of roughly 10 meters of radius. Powering the motion sensor through the Pi's 5V output pin, a GPIO pin on the Pi receives a signal from the motion sensors middle pin. If the output of the pin is a HIGH digital output, it means that there is motion in the area, vice versa if there is a LOW signal, no motion is currently detected. Harnessing this functionality we can decide when to take pictures.

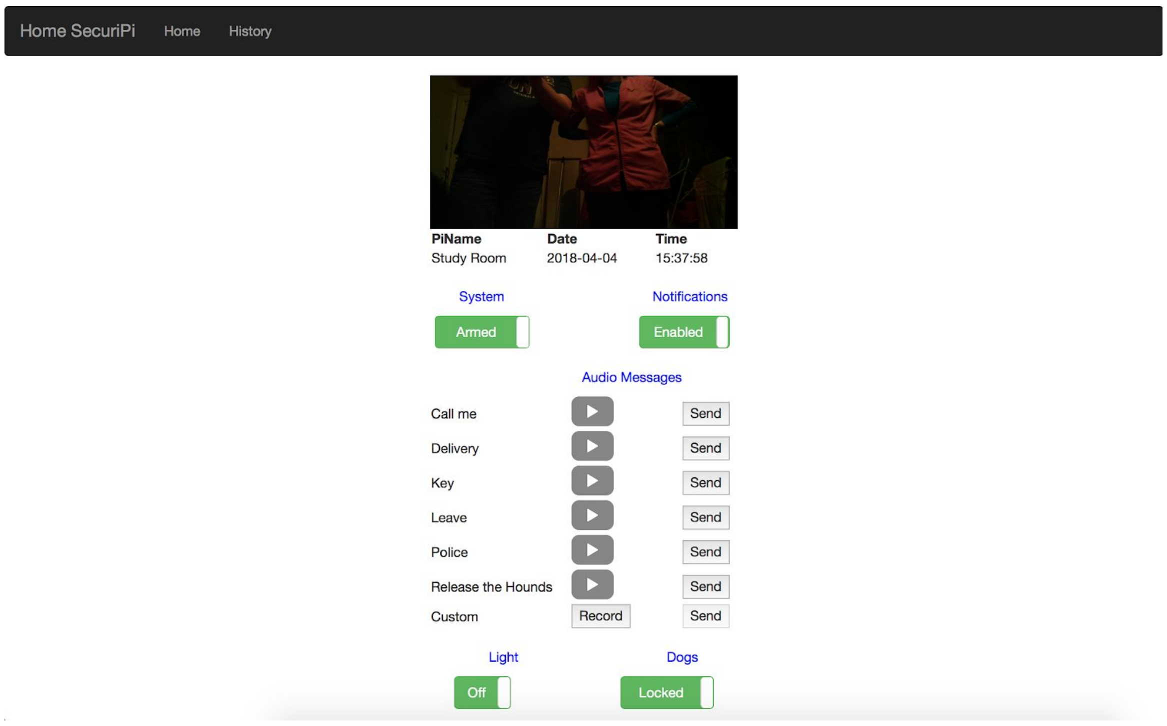
3. User Interface

“The ability to pick up a gadget and do things without overthinking how it works is not only a good thing, it’s what buyers have come to expect” [4]

With this quote in mind, the design of the interface should be simple, informative and easy to use. We believe everything of interaction should be possible from one screen, meaning a one-stop-shop for the core functionalities. The design will attempt to be responsive, meaning it can be used on any device capable of running a browser. Some design possibilities will be provided below.

3.1 Home Screen

Landing on the Home screen after receiving a text message, the user should be able to see the most recent image taken by Home SecuriPi. From that image they can make a decision on what to do next. There are a few possibilities of what might be seen, but what this project focuses on is the main possibilities of it being either friend or foe. Options for friend could be simply asking the person to “Call Me”. While options for a foe can be as dramatic as “Releasing the Hounds”.



PiName	Date	Time
Study Room	2018-04-04	15:37:58

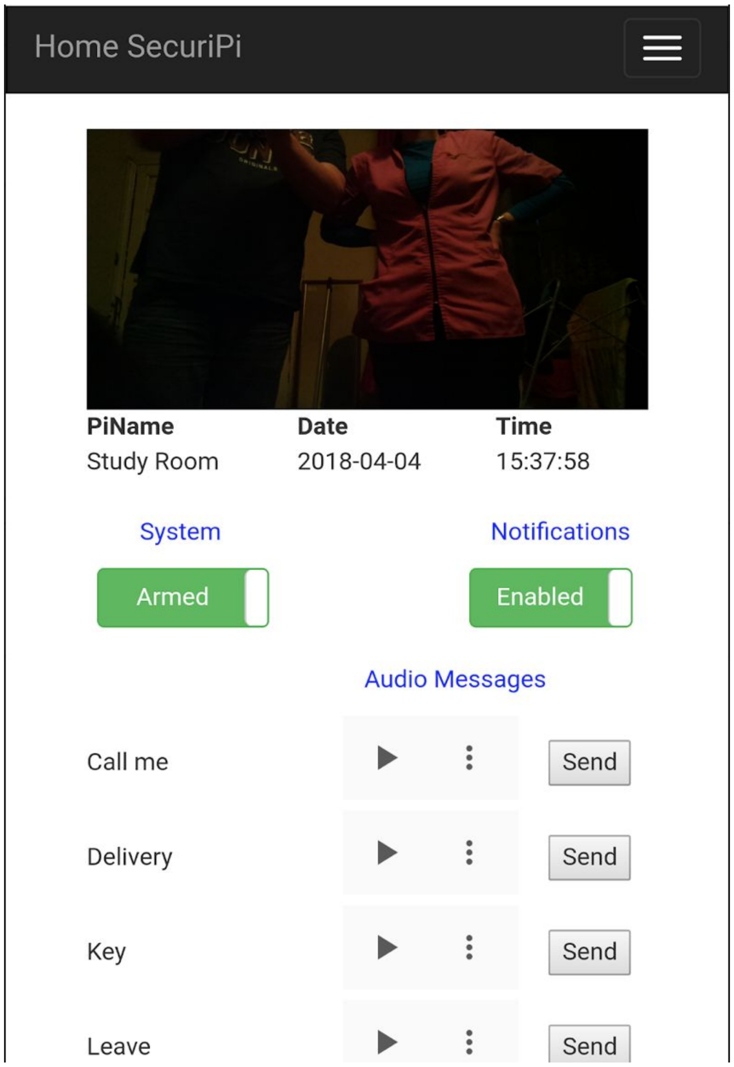
System Notifications
Armed Enabled

Audio Messages

Call me	▶	Send
Delivery	▶	Send
Key	▶	Send
Leave	▶	Send
Police	▶	Send
Release the Hounds	▶	Send
Custom	Record	Send

Light Dogs
Off Locked

Fig. 3.1.1 Laptop Device Home Screen

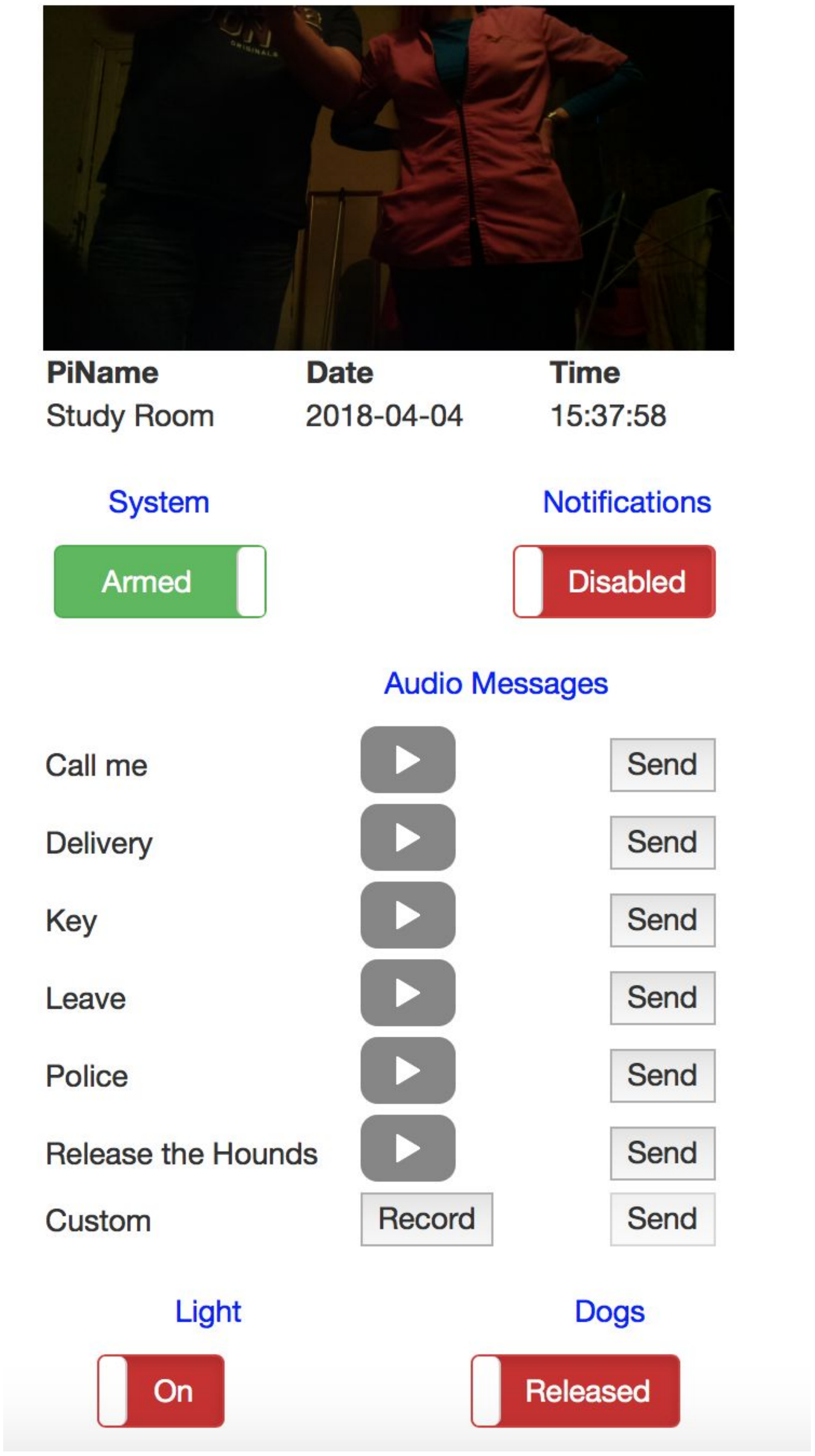


PiName	Date	Time
Study Room	2018-04-04	15:37:58

Call me
Delivery
Key
Leave

Fig. 3.1.2 Mobile Phone Home Screen

The navigation bar of a mobile device obviously has serious limitations concerning screen real estate. The design of both screens should feel similar, and a proposed design would be to have a "Hamburger" icon in the top corner as seen in Fig. 3.1.2. This should provide a drop down showing all the same functionalities as the laptop or some larger device as seen in Fig. 3.1.1.
Any device will have the active state of the system displayed through the use of toggle switches ie. if there is more than one user using the system concurrently, each user should be able to see what every other user is doing. If one user decides to turn on a light, every other user should be made aware of this action. This will be described further on in this document.
For clarity, a screenshot of the system in a different state is provided in Fig. 3.1.3.



PiName	Date	Time
Study Room	2018-04-04	15:37:58

Fig. 3.1.3 Live State of System

3.2 History

For the purposes of looking back through the images taken by the Home SecuriPi system, a history option should be provided. This should allow the user to scroll through the images while being able to see whereabouts on the property the picture was taken, as well as the date and time of capture. A proposed design is given below in Fig. 3.2.1 & 3.2.2, with Fig. 3.2.2 showing a proposed drop down list on a smaller mobile device.

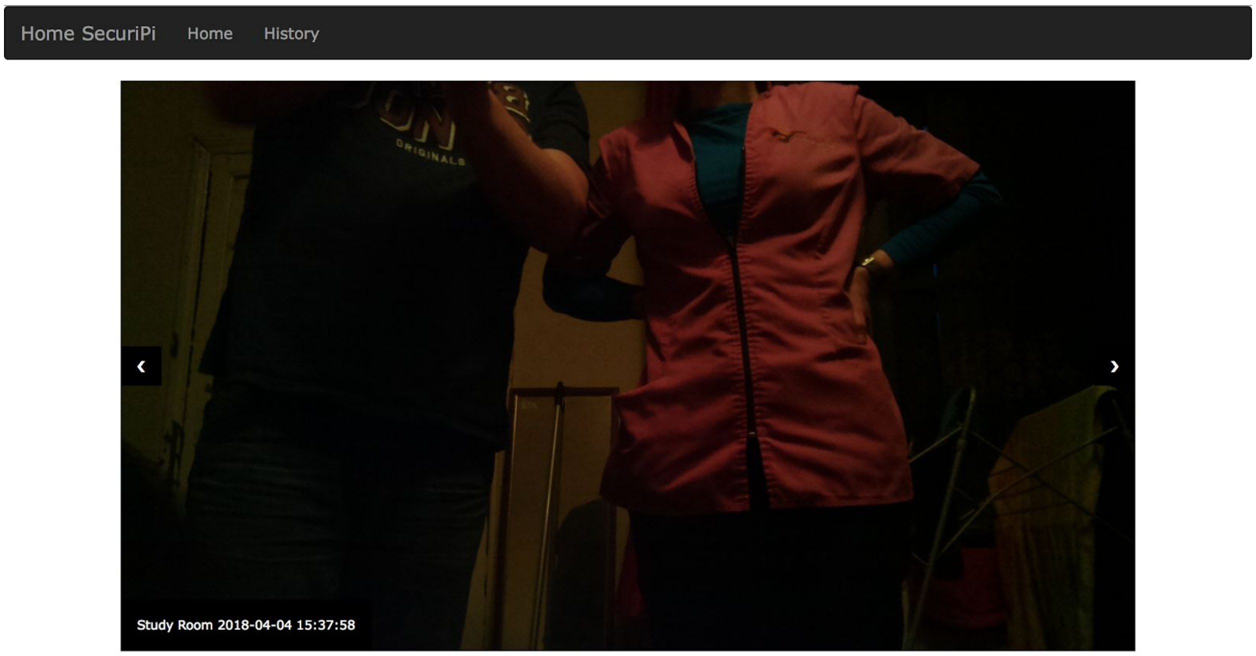


Fig. 3.2.1 Laptop History Screen

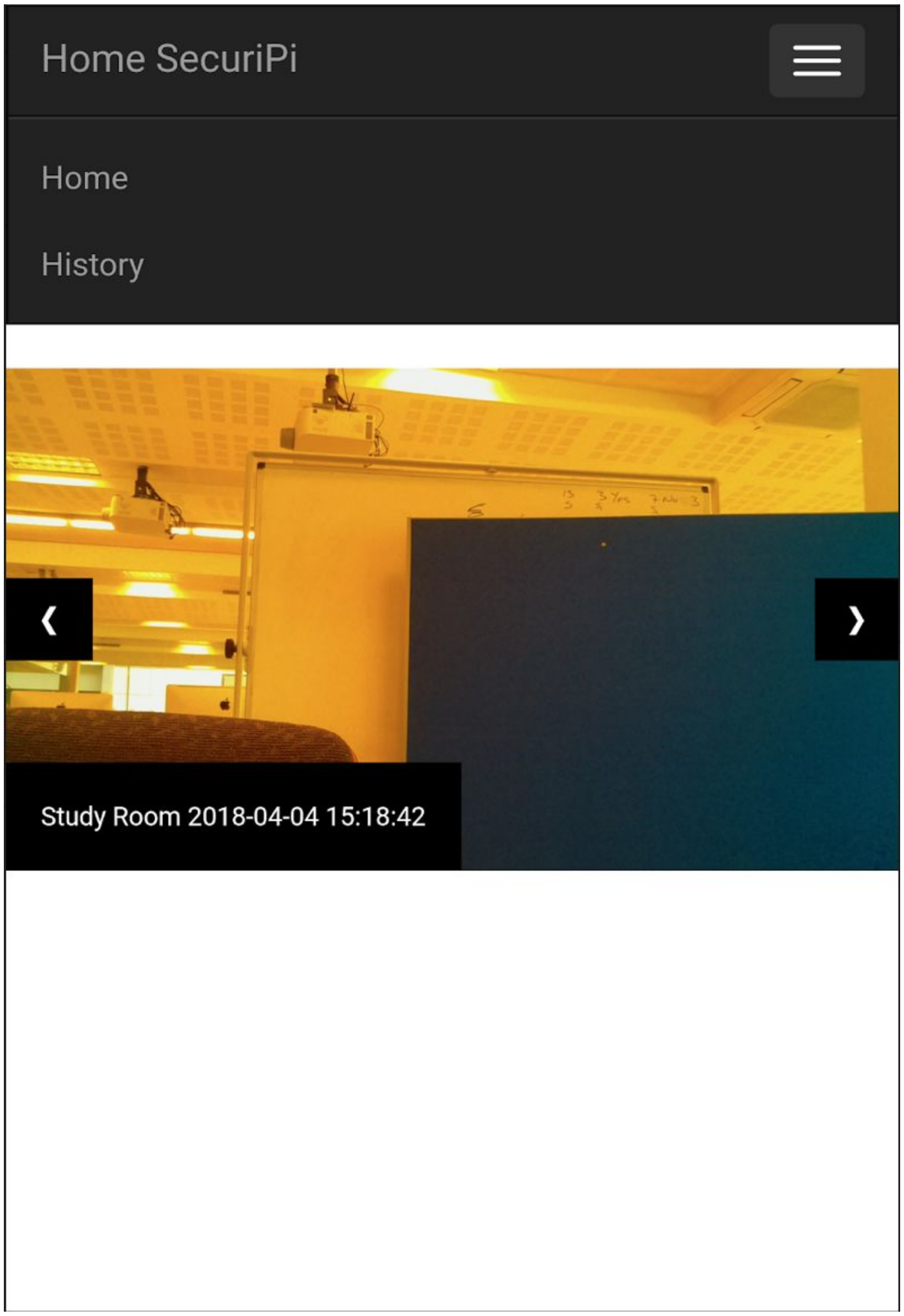


Fig. 3.2.2 Mobile History with “Hamburger” icon clicked

4. Database Design

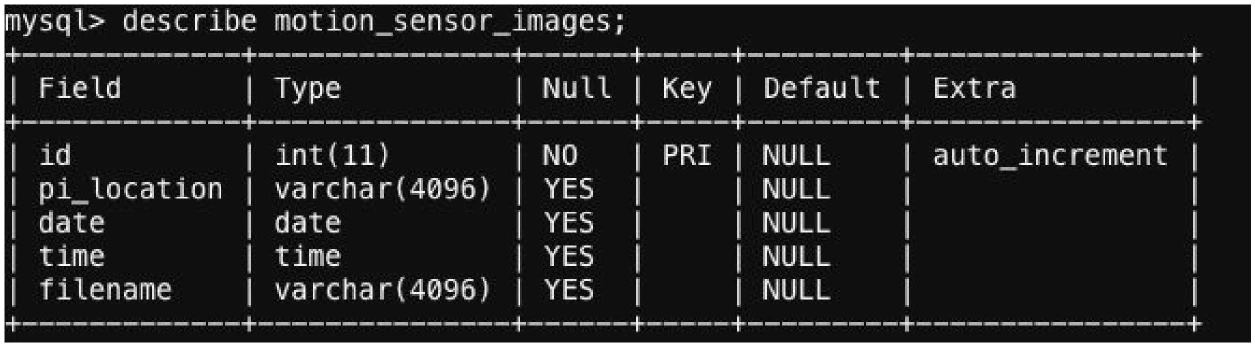
The design of the database is relatively simple for this project. Initially we simply need one table in our database for storing picture information.

Database Name: *PiProject*

Table Name: *motion_sensor_images*

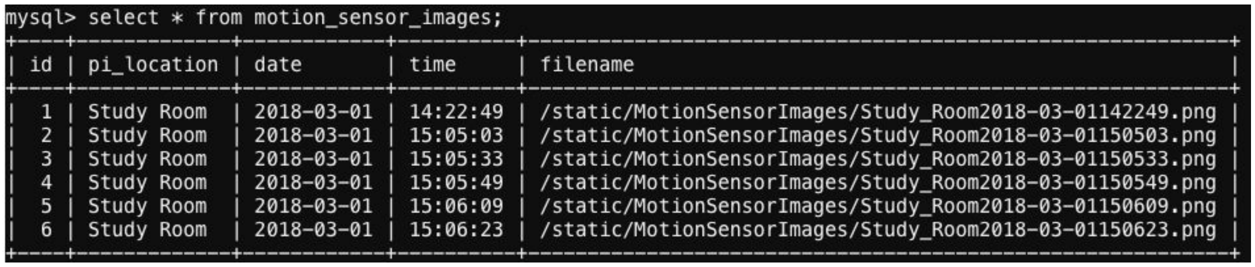
Description: *Used to store details of image taken, such as time, location, date and location/name of image to be displayed.*

Table Structure:



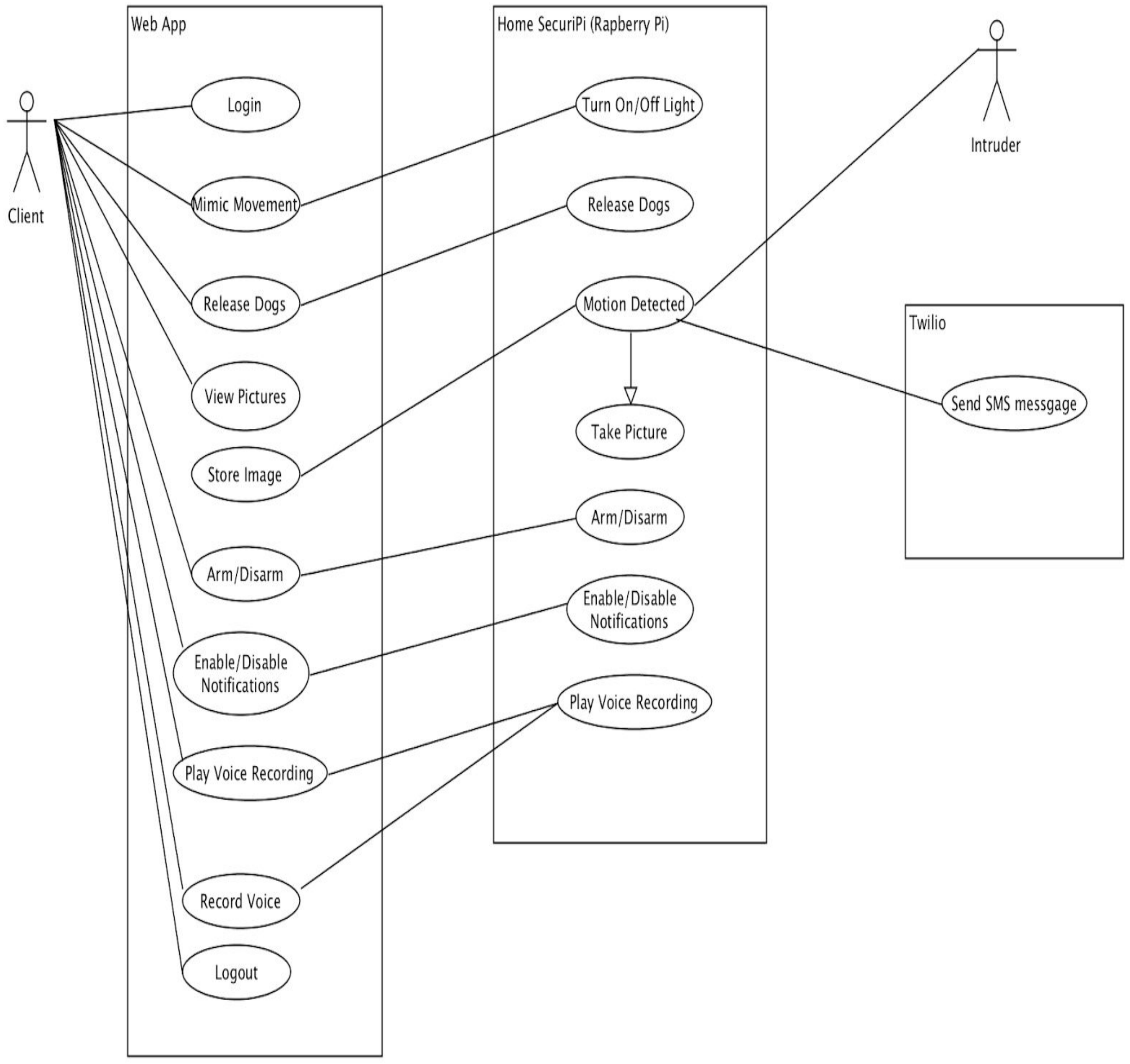
```
mysql> describe motion_sensor_images;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| id          | int(11)       | NO   | PRI | NULL    | auto_increment |
| pi_location | varchar(4096) | YES  |     | NULL    |                |
| date        | date          | YES  |     | NULL    |                |
| time        | time          | YES  |     | NULL    |                |
| filename    | varchar(4096) | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
```

Sample Data:



```
mysql> select * from motion_sensor_images;
+----+-------------+------------+----------+-----------------------------------------------------------+
| id | pi_location | date       | time     | filename                                                  |
+----+-------------+------------+----------+-----------------------------------------------------------+
|  1 | Study Room  | 2018-03-01 | 14:22:49 | /static/MotionSensorImages/Study_Room2018-03-01142249.png |
|  2 | Study Room  | 2018-03-01 | 15:05:03 | /static/MotionSensorImages/Study_Room2018-03-01150503.png |
|  3 | Study Room  | 2018-03-01 | 15:05:33 | /static/MotionSensorImages/Study_Room2018-03-01150533.png |
|  4 | Study Room  | 2018-03-01 | 15:05:49 | /static/MotionSensorImages/Study_Room2018-03-01150549.png |
|  5 | Study Room  | 2018-03-01 | 15:06:09 | /static/MotionSensorImages/Study_Room2018-03-01150609.png |
|  6 | Study Room  | 2018-03-01 | 15:06:23 | /static/MotionSensorImages/Study_Room2018-03-01150623.png |
+----+-------------+------------+----------+-----------------------------------------------------------+
```

5. Use Case Diagram



6. Detailed Use Cases

6.1 Login

Actors involved: Client, Web App

Brief Description: This use case begins when a **Client** wishes to log into the web app. The **Client** will enter their details, which will include their Username and Password. Upon completion the details are sent to the **Web App,** which verifies the details. This use case ends when the user has successfully logged in.

Main Success Scenario:

1. The **Client** chooses the option of Login in the web app.
2. The **Client** is presented with a Login screen with the username and password fields to be filled in.
3. The **Client** enters their details and submits the form.
4. The web app validates the form.
5. The **Web App** checks the username and password against records.
6. The **Web App** sends positive confirmation to the web app and stores login details.
7. The web app displays the appropriate home screen.

Alternatives:

7a. The **Cloud Server** responds with negative confirmation.

1. The web app increases the count of attempts by 1, up to a maximum of 3.
2. The web app informs the **Client** that their username or password is incorrect and displays the Login Screen again.
3. The **Client** re-enters their details and submits.
4. Steps repeated until positive confirmation or if maximum attempts have been reached(3), freeze access for a period of time and log attempts.
5. The web app returns to step 4.

6.2 Motion Detected

Actors involved: Intruder, Home SecuriPi , Web App
Brief Description: This use case begins when an **Intruder** sets off the motion sensor. **Home SecuriPi** will take a picture and send a HTTP request to the **Web App.** The **Web App** will store the image. This use case ends when the image has been successfully stored.
Main Success Scenario:

1. **Home SecuriPi** detects motion.
2. **Home SecuriPi** takes a picture.
3. The picture gets stored on the **Home SecuriPi** device(Raspberry Pi).
4. A HTTP request gets sent to the **Web App** with the picture attached**.**
5. The **Web App** stores the image on the server.
6. The **Web App** updates the database.
7. The **Web App** responds to the request with the size of the image received.
8. **Home SecuriPi** confirms the size of the image received equally compares with the size of the image stored locally.
9. **Home SecuriPi** sends a message to a specific mobile phone.

Alternatives:
7a. The size of the image does not equate to the size of the image stored locally.

1. **Home SecuriPi** repeats from step 4.

6.3 Mimic Movement

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to mimic movement in their home. The **Client** will click the On/Off button for the light. This use case ends when the light has been successfully turned On/Off.
Main Success Scenario:

1. The **Client** clicks the On/Off button for the light.
2. The **Web App** sends a HTTP request to the **Home SecuriPi.**
3. **Home SecuriPi** sets the output pin to HIGH/LOW.
4. The relay switch circuit closes/opens.
5. The light turns On/Off.
6. **Home SecuriPi** responds with a status 200.
7. The **Web App** toggles the On/Off button.

Alternatives:
3a. **Home SecuriPi** returns a status 500.

1. The **Web App** displays an error message.
2. The toggle doesn’t change.

6.4 Release Dogs

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to release their dogs(or open a door). The **Client** will click the Lock/Release button for the door latch. This use case ends when the door latch has been successfully Locked/Released.
Main Success Scenario:

1. The **Client** clicks the Lock/Release button.
2. The **Web App** sends a HTTP request to the **Home SecuriPi.**
3. **Home SecuriPi** sets the output pin to HIGH.
4. The relay switch circuit closes/opens.
5. The door Locks/Releases.
6. **Home SecuriPi** responds with a status 200.
7. The **Web App** toggles the Lock/Release button.

Alternatives:
3a. **Home SecuriPi** responds with a status 500.

1. The **Web App** displays an error message.
2. The toggle doesn't change.

6.5 Arm/Disarm

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to Arm/Disarm the system. The **Client** clicks the Arm/Disarm button. This use case ends when the system has been successfully Armed/Disarmed.
Main Success Scenario:

1. The **Client** clicks the Arm/Disarm button.
2. The **Web App** sends a HTTP request to the **Home SecuriPi.**
3. **Home SecuriPi** Arms/Disarms the system.
4. **Home SecuriPi** responds with a 200 status.
5. The **Web App** toggles the Arm/Disarm button.

Alternatives:
3a. **Home SecuriPi** responds with a status 500.

1. The **Web App** displays an error message.
2. The toggle doesn't change.

6.6 Enable/Disable Notifications

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to Enable/Disable the notifications. The use case ends when the notifications have been successfully Enabled/Disabled.
Main Success Scenario:

1. The **Client** clicks the Enable/Disable button.

2. The **Web App** sends a HTTP request to the **Home SecuriPi.**
3. **Home SecuriPi** Enables/Disables the notifications.
4. **Home SecuriPi** responds with a 200 status.
5. The **Web App** toggles the Enable/Disable button.

Alternatives:
3a. **Home SecuriPi** responds with a status 500.
1. The **Web App** displays an error message.
2. The toggle doesn't change.

6.7 Play Voice Recording

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to play a voice recording via the audio output of the Raspberry Pi. The **Client** will click send on the message they wish to play. The use case ends when the recording has been successfully played.
Main Success Scenario:
1. The **Client** clicks the send button related to the message they wish to send.
2. The **Web App** retrieves the selected message from local storage.
3. The **Web App** sends a HTTP request to the **Home SecuriPi** with the voice message attached**.**
4. **Home SecuriPi** saves the file locally.
5. **Home SecuriPi** opens and plays the voice message through the Raspberry Pis audio jack.
6. **Home SecuriPi** responds with a 200 status.

Alternatives:
4a. **Home SecuriPi** responds with a status 500.
1. The **Web App** displays an error message.

6.8 Record Voice

Actors involved: Client, Web App, Home SecuriPi
Brief Description: This use case begins when the **Client** wishes to send a live voice message to the **Home SecuriPi** device. The **Client** selects record, and once they are finished speaking they click send. This use case ends when the voice message has been played.
Main Success Scenario:
1. The **Client** clicks the Record button.
2. The **Web App** asks the user for permission to access the microphone.
3. The **Client** allows access.
4. The **Web App** opens the microphone stream.
5. The **Web App** records the message through the microphone of the device.
6. The **Client** clicks the send button.
7. The **Web App** formats the file into wav format.
8. The **Web App** saves the file locally.
9. The **Web App** sends a HTTP request to the **Home SecuriPi** with the voice message attached**.**

10. **Home SecuriPi** saves the file locally.
11. **Home SecuriPi** opens and plays the voice message through the Raspberry Pis audio jack.
12. **Home SecuriPi** responds with a 200 status.

Alternatives:
6a. **Home SecuriPi** responds with a status 500.
1. The **Web App** displays an error message.

6.9 View Pictures

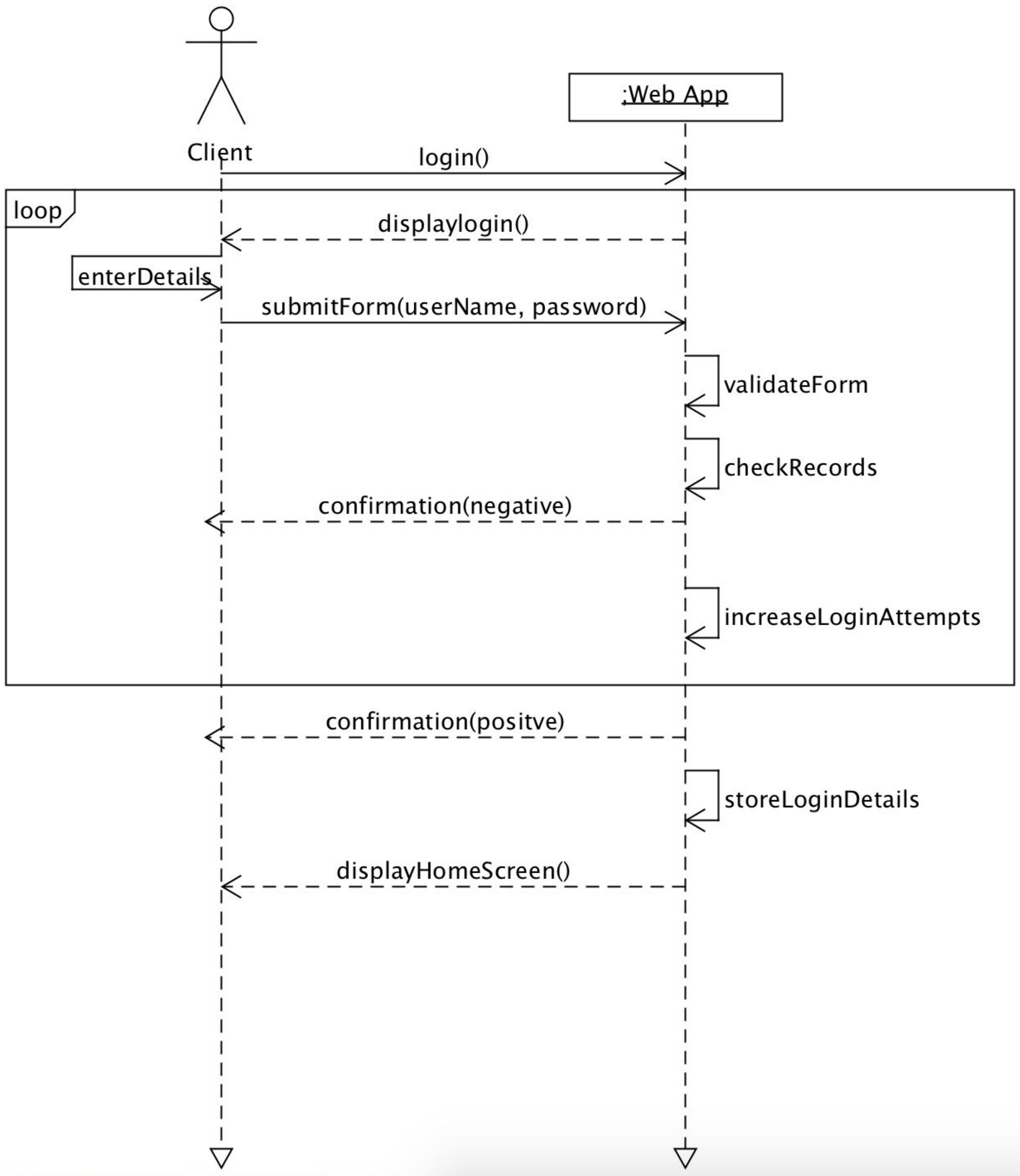
Actors involved: Client, Web App
Brief Description: This use case begins when the **Client** wishes to view the stored image history. The **Client** will select the History button on the Navbar. The **Web App** will display the images. This use case ends when the images are successfully displayed.
Main Success Scenario:
1. The **Client** clicks the History button.
2. The **Web App** queries the database to find the name of all the images stored.
3. The **Web App** retrieves the images from local storage.
4. The **Web App** displays the images.

6.10 Logout

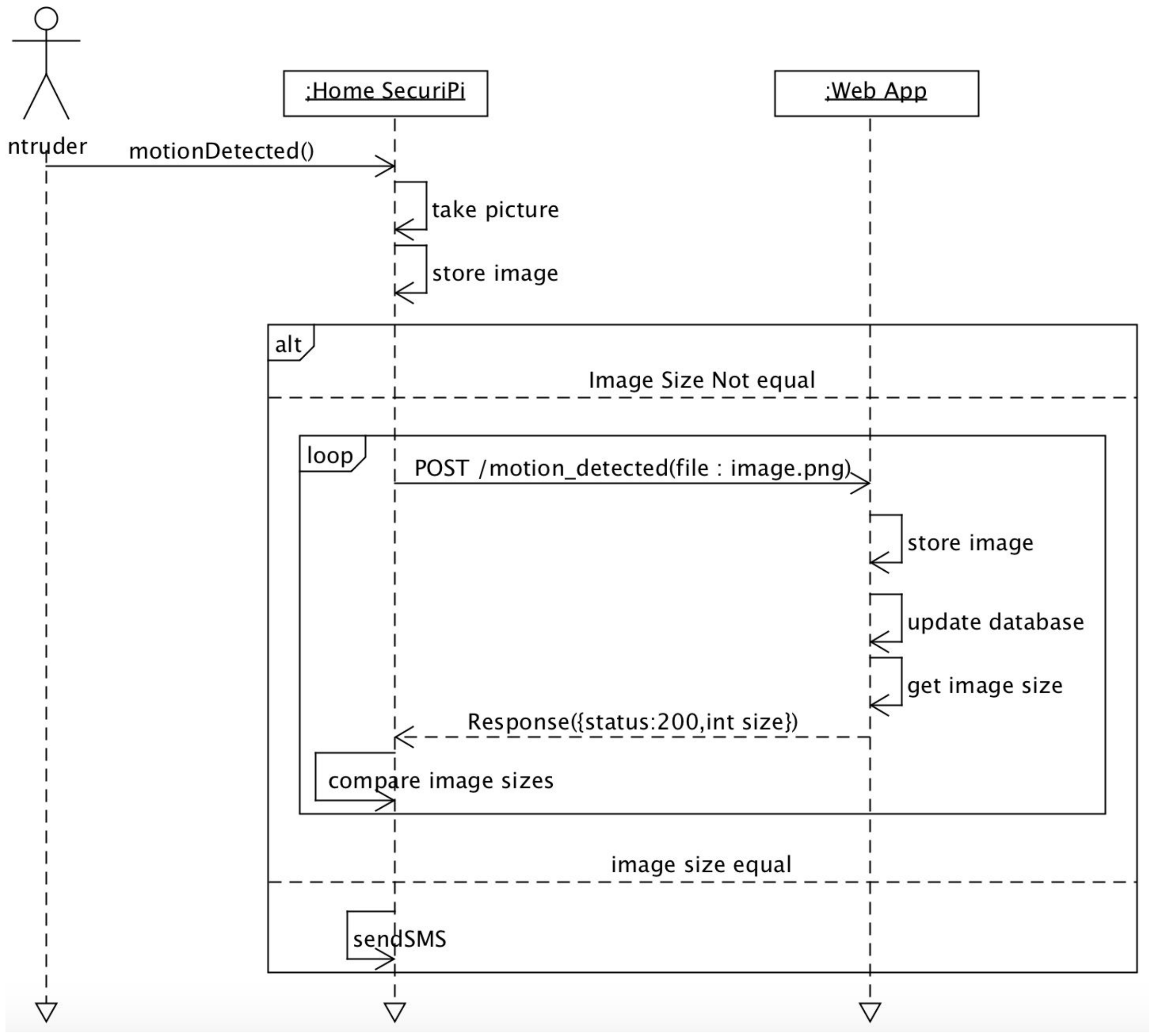
Actors involved: Client, Web App
Brief Description: This use case begins when a **Client** wishes to logout of the web app. The **Client** will click on the "Logout" button. This use case ends when the **Client** has been successfully logged out.
Main Success Scenario:
1. The **Client** clicks the "Logout" button.
2. The **Web App** responds by providing a pop-up which asks for confirmation of logout.
3. The **Client** clicks the "Confirm" button.
4. The **Web App** destroys the session.
5. The **Web App** displays the Login screen.

7. System Sequence Diagram

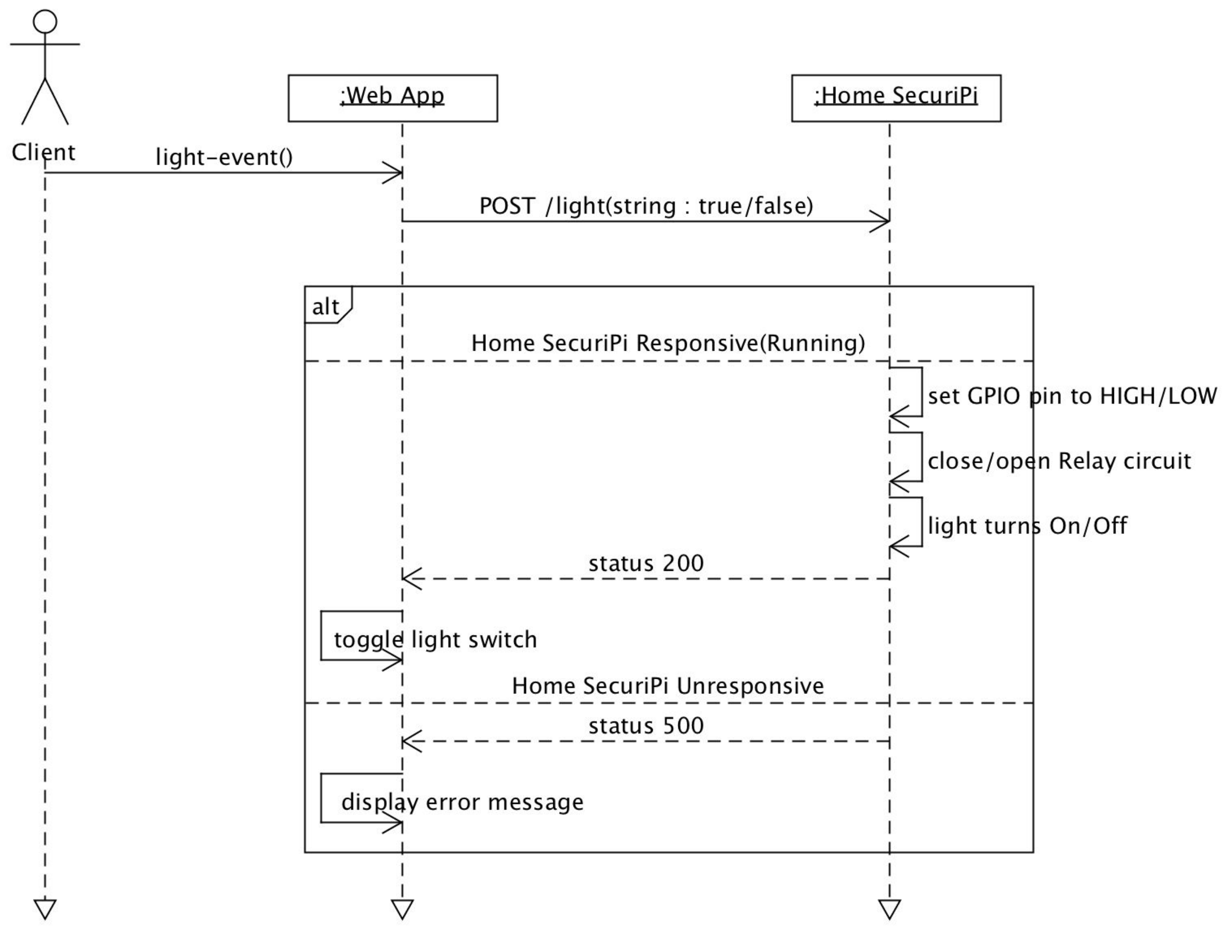
7.1 Login Sequence Diagram



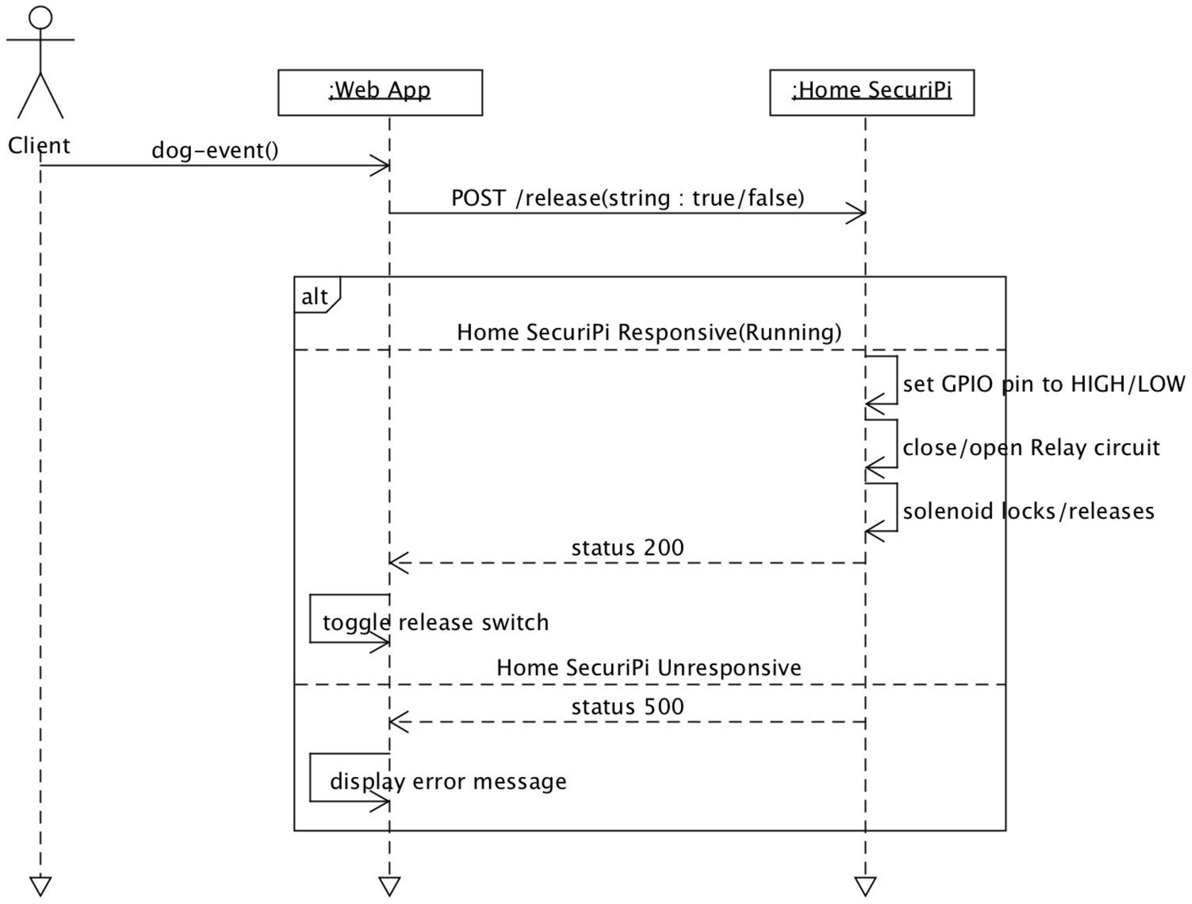
7.2 Motion Detected Sequence Diagram



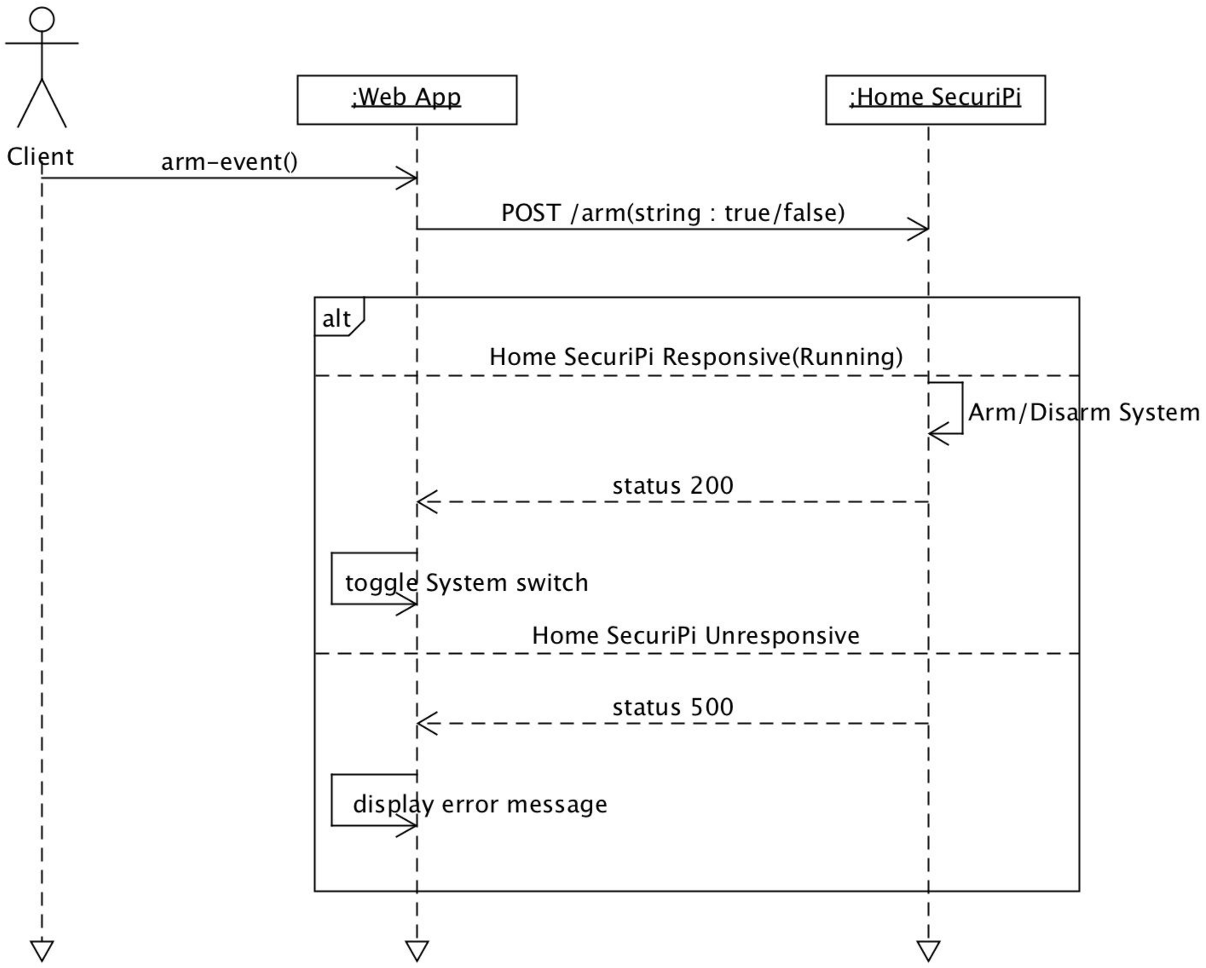
7.3 Mimic Movement Sequence Diagram



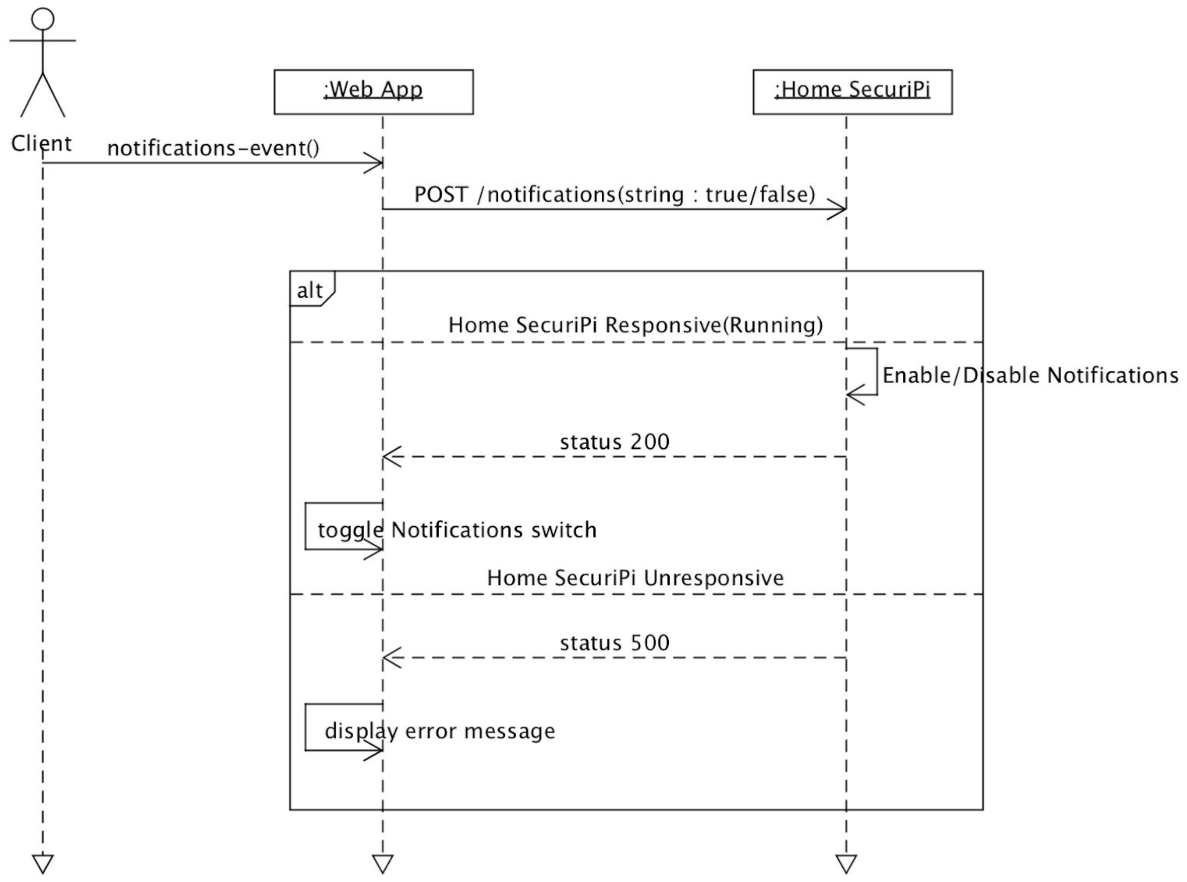
7.4 Release Dogs Sequence Diagram



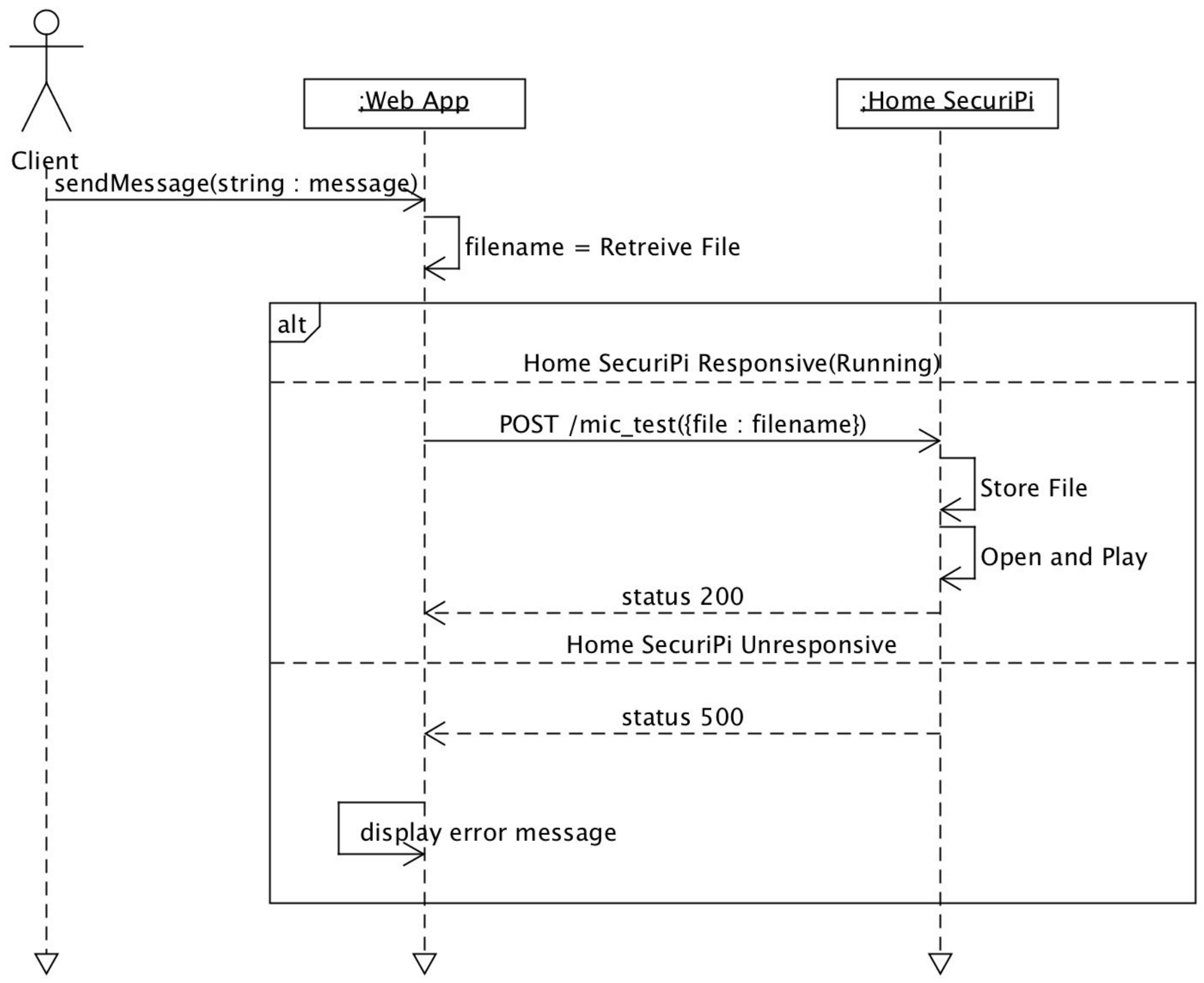
7.5 Arm/Disarm Sequence Diagram



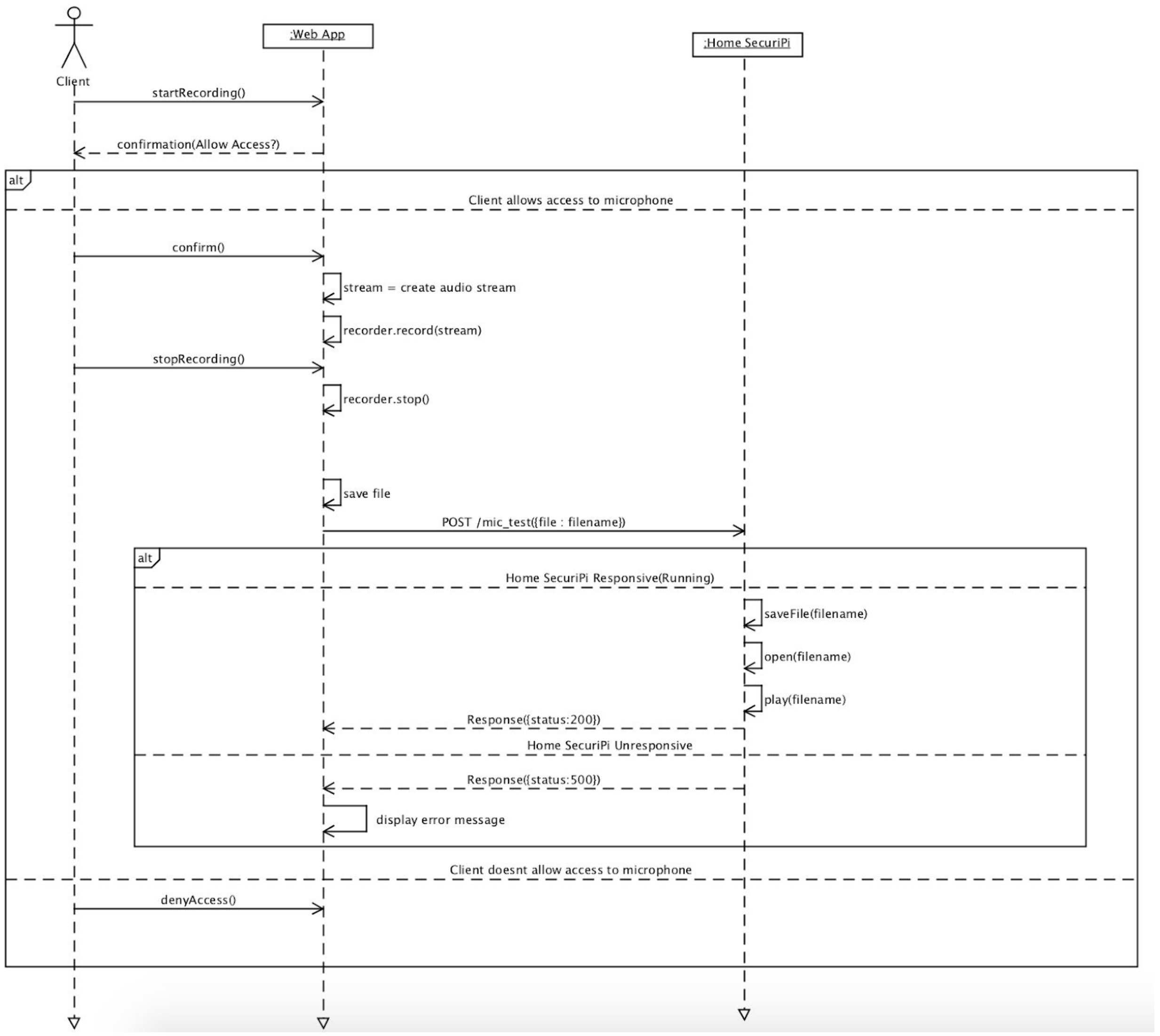
7.6 Enable/Disable Notifications Sequence Diagram


Client
:Web App
:Home SecuriPi
notifications-event()
POST /notifications(string : true/false)
alt
Home SecuriPi Responsive(Running)
Enable/Disable Notifications
status 200
toggle Notifications switch
Home SecuriPi Unresponsive
status 500
display error message

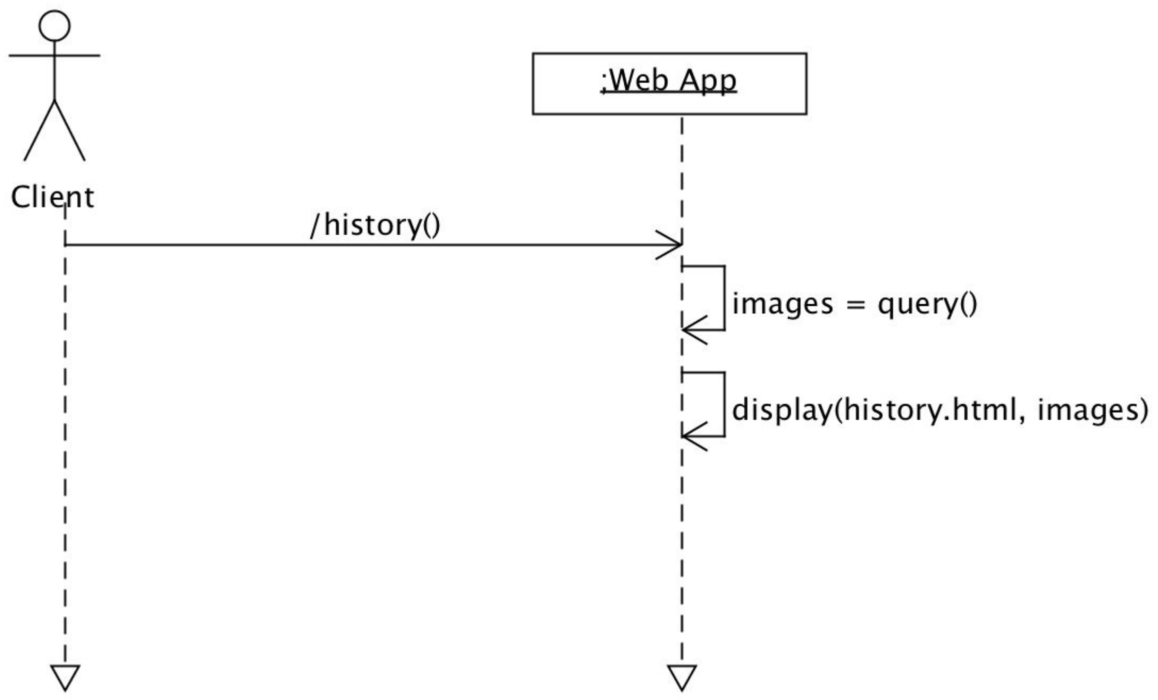
7.7 Play Voice Recording Sequence Diagram


Client
:Web App
:Home SecuriPi
sendMessage(string : message)
filename = Retreive File
alt
Home SecuriPi Responsive(Running)
POST /mic_test({file : filename})
Store File
Open and Play
status 200
Home SecuriPi Unresponsive
status 500
display error message

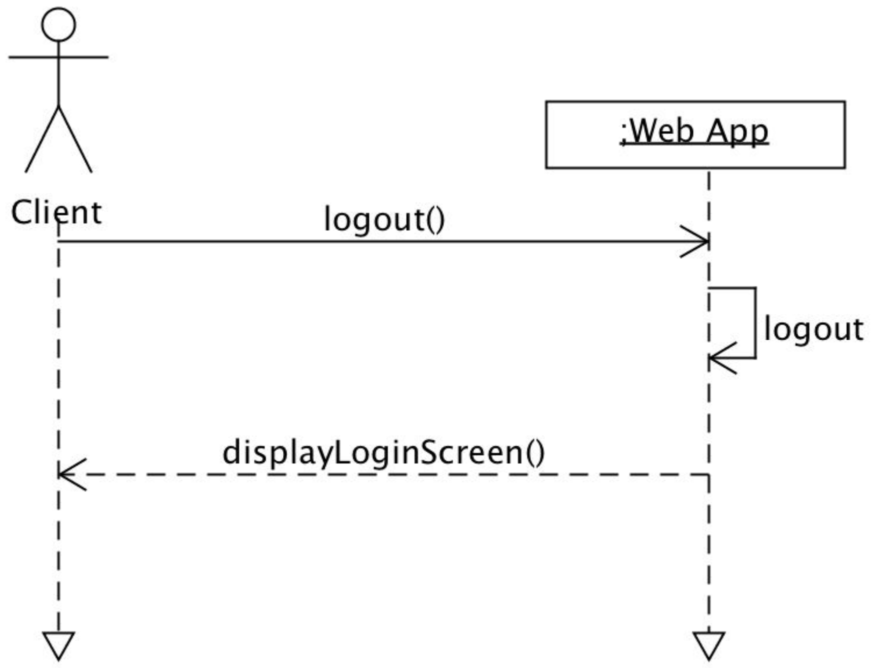
7.8 Record Voice Sequence Diagram



7.9 View Pictures



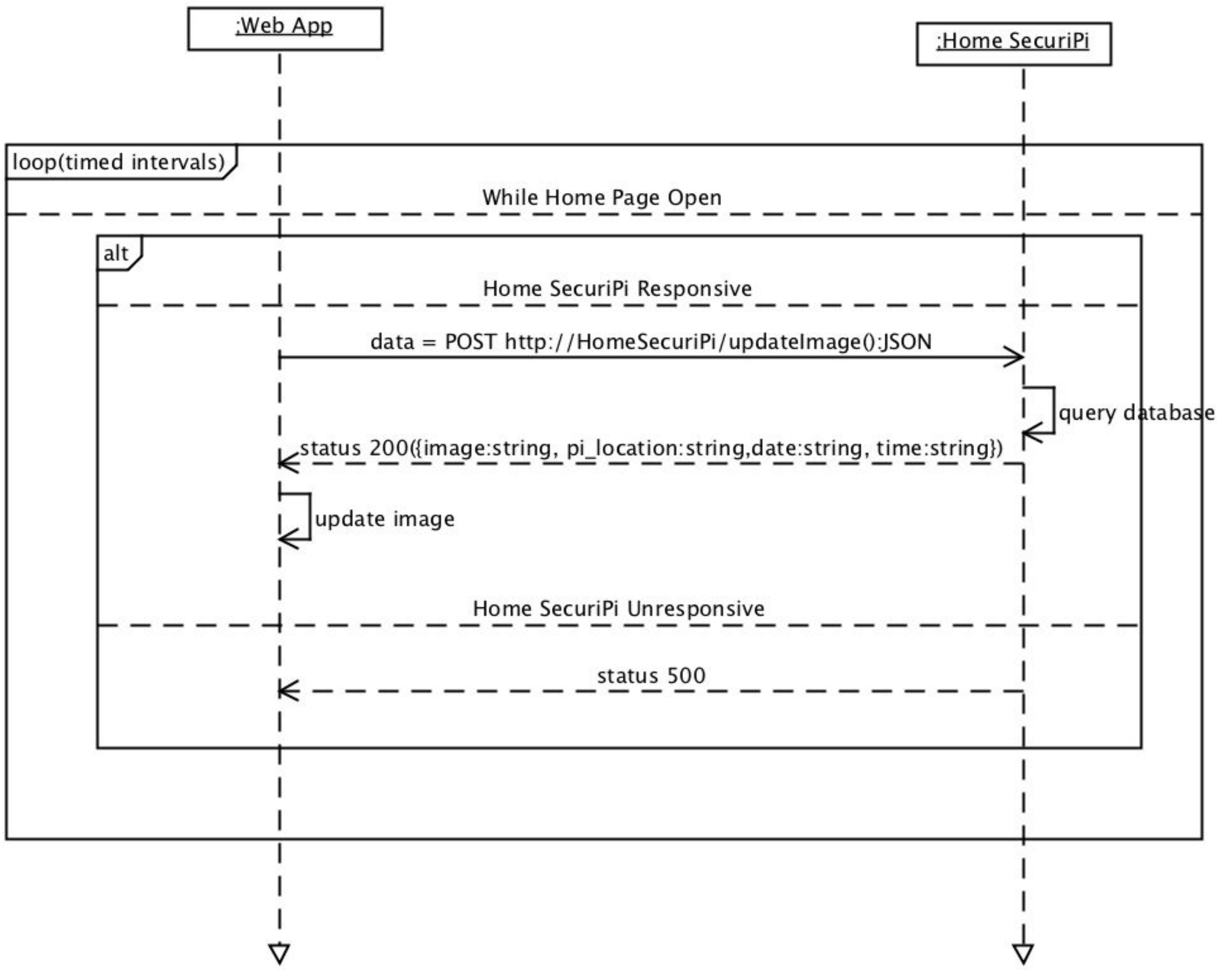
7.10 Logout Sequence Diagram



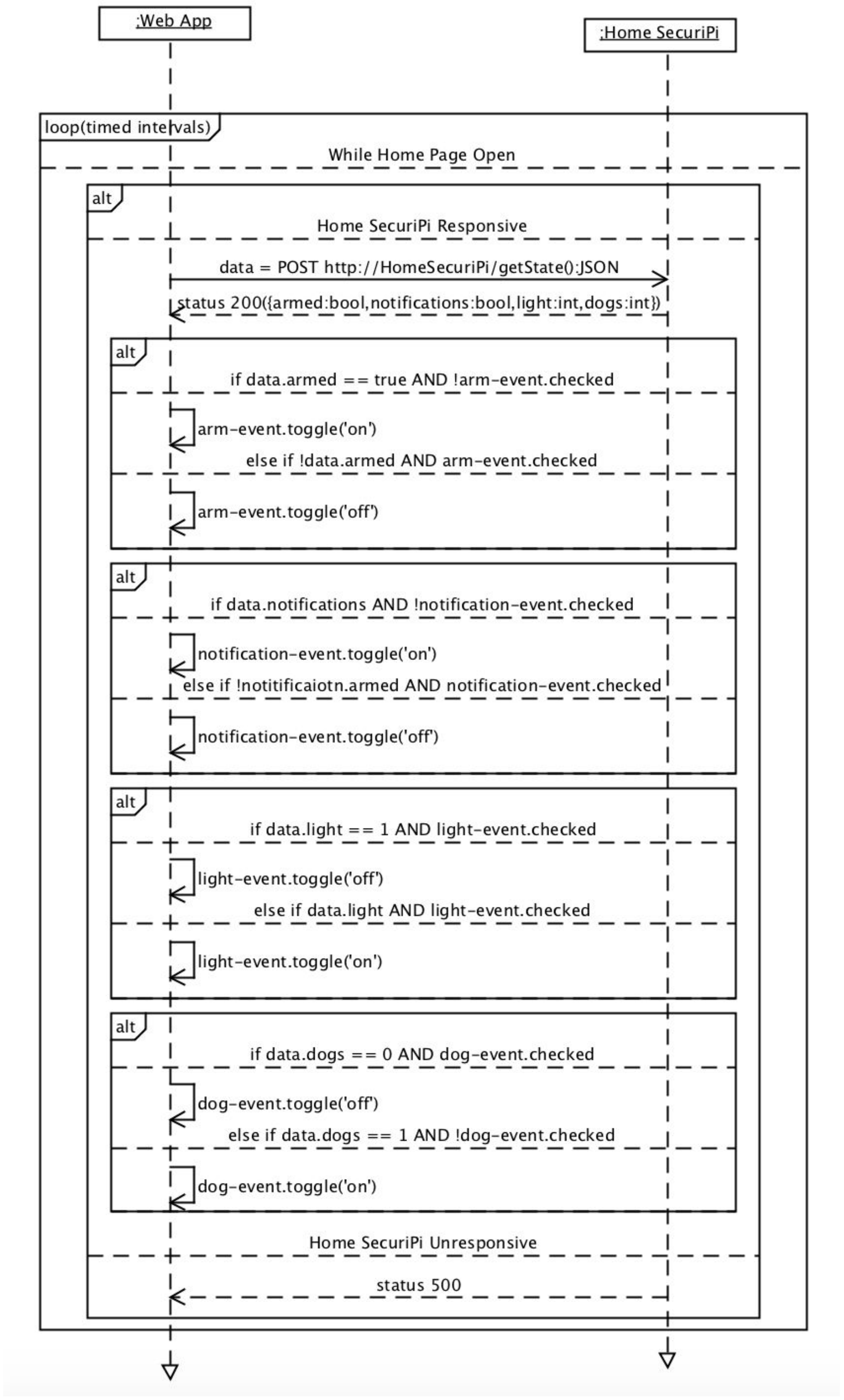
8. Background Services

In order for users of Home SecuriPi to be constantly aware of the current state of the system, which is important as knowledge of lights being turned on, or the system being in an armed/disarmed state is a crucial aspect to a security device, some background services need to be implemented. A service to update the image displayed on the Home screen as it is taken in a live environment should also be implemented. These will be outlined briefly with additional sequence diagrams.

8.1 Update Image Sequence Diagram


:Web App
:Home SecuriPi
loop(timed intervals)
While Home Page Open
alt
Home SecuriPi Responsive
data = POST http://HomeSecuriPi/updateImage():JSON
query database
status 200({image:string, pi_location:string,date:string, time:string})
update image
Home SecuriPi Unresponsive
status 500

8.2 Get State Sequence Diagram


:Web App
:Home SecuriPi
loop(timed intervals)
While Home Page Open
alt
Home SecuriPi Responsive
data = POST http://HomeSecuriPi/getState():JSON
status 200({armed:bool,notifications:bool,light:int,dogs:int})
alt
if data.armed == true AND !arm-event.checked
arm-event.toggle('on')
else if !data.armed AND arm-event.checked
arm-event.toggle('off')
alt
if data.notifications AND !notification-event.checked
notification-event.toggle('on')
else if !notitificaiotn.armed AND notification-event.checked
notification-event.toggle('off')
alt
if data.light == 1 AND light-event.checked
light-event.toggle('off')
else if data.light AND light-event.checked
light-event.toggle('on')
alt
if data.dogs == 0 AND dog-event.checked
dog-event.toggle('off')
else if data.dogs == 1 AND !dog-event.checked
dog-event.toggle('on')
Home SecuriPi Unresponsive
status 500

9. References

- [1] Stack Exchange(Nov 2017). Raspberry Pi[online], available:https://raspberrypi.stackexchange.com/questions/74570/controlling-switches-from-both-raspberry-pi-relay-manual-home-automation [accessed 10 Apr, 2018].

- [2] Stack Exchange(July 2017). Electrical Engineering[online], available:https://electronics.stackexchange.com/questions/318497/relay-component-to-schematic-are-these-the-same [accessed 10 Apr, 2018].

- [3] Raspberry Pi (n.d.). Santa Detector[online], available: https://projects.raspberrypi.org/en/projects/santa-detector/3 [accessed 13 Apr, 2018].

- [4] ZDNet (July 9, 2013). 3 Rules for a good user experience (UX)[online], available: https://www.zdnet.com/article/3-rules-for-a-good-user-experience-ux/ [accessed 13 Apr, 2018].