National College of Ireland

BSc in Computing
2016/2017

Conor Breen
X13316381
conor@breen.ie

OpenSesamessage

OpenSesamessage Smart Home Entry System

Technical Report

Table of Contents

1 Executive Summary

OpenSesamessage uses multiple technologies to provide a cross-platform solution to smart home entry. Working off the concept of usability and utilizing a peoples almost universal knowledge of messenger services. OpenSesamessage aims to provide a smart home security solution with ease of use and security as paramount.

Built using a ruby on rails web application to allow simple user integration with the service, and intuitive account management. The popular messaging service telegram which boasts high security and exemplary encryption as standard and the raspberry pi to offer an easy to use, safe and affordable solution to smart home entry.

OpenSesamessage is a fully functioning smart home entry system for the future, allowing for keyless entry into buildings, or containers with the use of a smartphone. Its key features include:

- Simple to use web application with usability in mind
- Security rich features to allow users to safely manage who has access to the service based on their unique phone number
- A simple to use user dashboard, allowing users to add and remove sub users when needed.
- Integration with the popular messaging service Telegram, which does not require the installation of a bulky app
- Security features such as the option to receive a photo of the person entering the building sent straight to the admin's phone.
- A truly keyless entry system, without the need for easily lost items such as RFID cards or key fobs

With the use of the OpenSesamessage users can grant access to their house, building or container from anywhere in the world via the internet. This negates the use if keys, RFID cards and other forms of entry. All you need is an internet connection and the permission of the owner. This service could benefit almost everyone. Home owners, who regularly forget their key. Landlords who rent out their property are no longer required to arrive with a key, the use case is scenarios are infinite

“open sesamessage brings consumer level home entry and home security to the modern era”

2 Introduction

The concept for this project came from my own necessities. In a family of seven people, the need for us all to have a key to the house was essential, however the occurrences of one of us forgetting a key and returning home late at night only to have to ring someone to get up and answer the door exposed a huge flaw in our home entry system. Keys are a great solution for home entry. But like all things keys can go missing. Leaving a key out for someone raises huge security concerns as it means anyone can pick it up and enter the house while everyone is asleep, or out.

With this system, users or home owners can create an account on the OpenSesamessage web application, enter their phone number and establish themselves them as the admin user. From there they can allow access to other simply by entering their phone numbers on the dashboard. They can monitor who is coming and going through the system and seamlessly add and remove users to their account granting and revoking the privilege of home entry through the web application.

3 Background & research

The initial research for this project came late in December as the project I was undergoing raised concerns for my project supervisor. I was building a front-end frame work on a company’s pre-existing website, and it was becoming apparent that this project was not meeting crucial marking scheme topics. So, after a meeting I decided to create my own ruby on rails project, implement front end framework technologies into it and implement an IOT aspect to better reflect my speciality.

My interest in the project became apparent when one night I forgot my key, and had to ring and wake somebody up to let me in. realising a flaw with the system we have for our home entry and the observation that regardless of whether I have a key or not I always seem to have my phone with me I began to research possible solutions to my problem

Researched methods

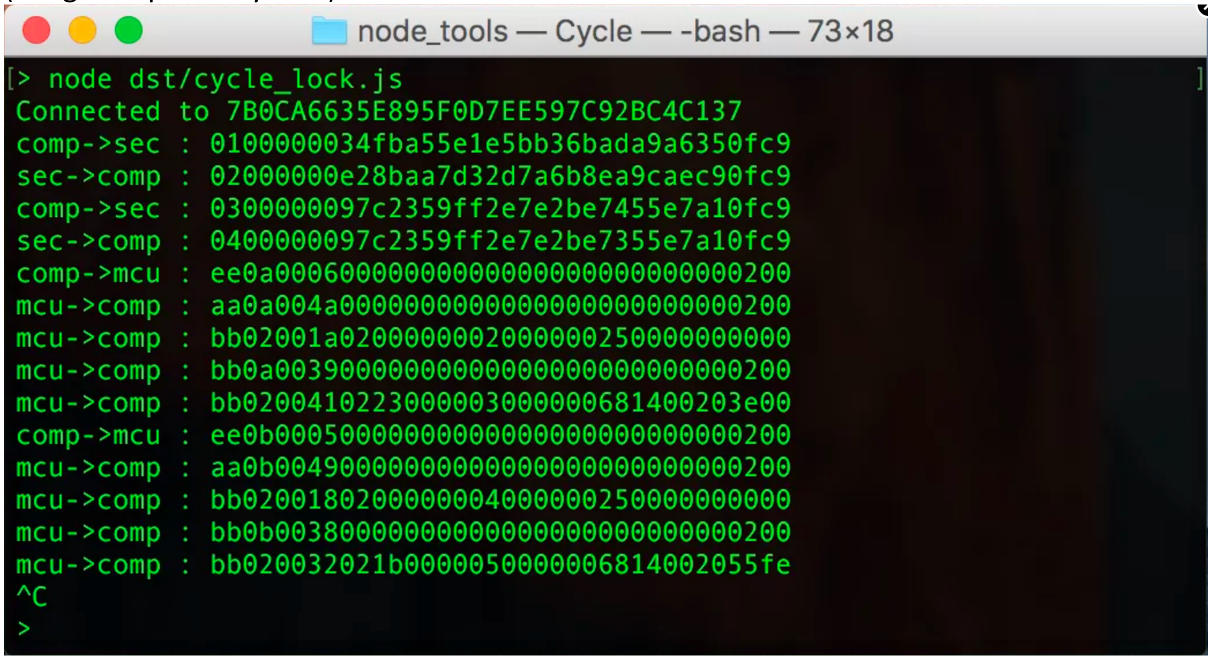
The methods I researched are as follows

- RFID entry
- Keypad Entry
- Fingerprint entry
- Voice recognition entry

These systems had amazing benefits over conventional use of keys, but in my eyes, they were all still flawed. Each time I researched a possible solution I found fault with it, and further questioned how it could be improved upon. Firstly, I considered having a pin entry system, one where a pin code would be established and distributed amongst tenants.

Pin code entry

(Image 1 – pin entry flaw)



```
[> node dst/cycle_lock.js
 Connected to 7B0CA6635E895F0D7EE597C92BC4C137
 comp->sec : 0100000034fba55e1e5bb36bada9a6350fc9
 sec->comp : 02000000e28baa7d32d7a6b8ea9caec90fc9
 comp->sec : 0300000097c2359ff2e7e2be7455e7a10fc9
 sec->comp : 0400000097c2359ff2e7e2be7355e7a10fc9
 comp->mcu : ee0a00060000000000000000000000000200
 mcu->comp : aa0a004a0000000000000000000000000200
 mcu->comp : bb02001a020000000200000025000000000
 mcu->comp : bb0a00390000000000000000000000000200
 mcu->comp : bb020041022300000300000068140020 3e00
 comp->mcu : ee0b00050000000000000000000000000200
 mcu->comp : aa0b00490000000000000000000000000200
 mcu->comp : bb020018020000000400000025000000000
 mcu->comp : bb0b00380000000000000000000000000200
 mcu->comp : bb020032021b0000050000006814002055fe
 ^C
 >
```

using a pin code as an entry system seems a logical choice for entering a building without a key or any physical means of entry. However, in recent times it has become a growing concern that hackers could gain access to these systems by brute forcing their way in by rapidly cycling through all code possibilities in minutes.

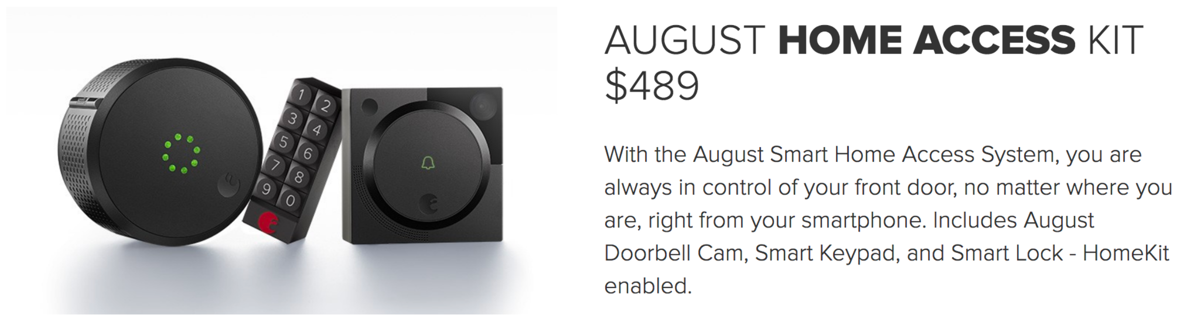
Provided the keypad on a system has the numbers 0 – 9, the possible combinations for a 4-pin code are 10 x 10 x 10 x 10 which is 10,000 possibilities. With modern technology, today that would only be a few minutes of a brute force attack.

RFID entry

The second form of home entry I researched was RFID entry. This system has obvious benefits. Including heightened security over pin entry. RFID entry allows for multiple unique keys to be used and each having encrypted information meaning the chance of a hacker gaining access to the system is greatly reduced over the previous case. However, RFID entry is very expensive. According to Ireland leading supplier of RFID entry systems the cost of a RFID entry system can upwards of one thousand euro. This proves too expensive for the average home user at the consumer level. Considering that every time a new user is needed to have access to the system a new RFID card or fob is required to be purchased.

Smartphone entry

(image 2 - smartphone entry system)



AUGUST **HOME ACCESS** KIT
$489

With the August Smart Home Access System, you are always in control of your front door, no matter where you are, right from your smartphone. Includes August Doorbell Cam, Smart Keypad, and Smart Lock - HomeKit enabled.

in recent years, a new system came onto the market form a company called August. This system allows users to enter their homes using their smartphones. This seemed promising, the ability to enter a home or office building using your smartphone was a great idea. But further research revealed some major flaws in the system.

- No android support
- Substantial price point
- Useless without internet connection

These flaws in the system proved too great a barrier for most consumers. The system is only compatible with IOS users, meaning a substantial portion of the potential user base is already alienated from the get-go. Another barrier is the fail safes in the system, or lack thereof. If the system experiences a loss in internet connectivity the system is rendered useless. Furthermore, the cost of almost 500 dollars seems too great for average users.

Research conclusion

It became evident that the solution that I wanted was not available. Before I set out on this project it is quite clear that there were no home entry solutions that accurately met my requirements. So, I set myself to the task to create a smart home entry system that would allow users simple but secure access to their homes, while providing cross platform support and keeping the solution modular so that it would work in conjunction with traditional methods of security. This was the starting point for my final year project.

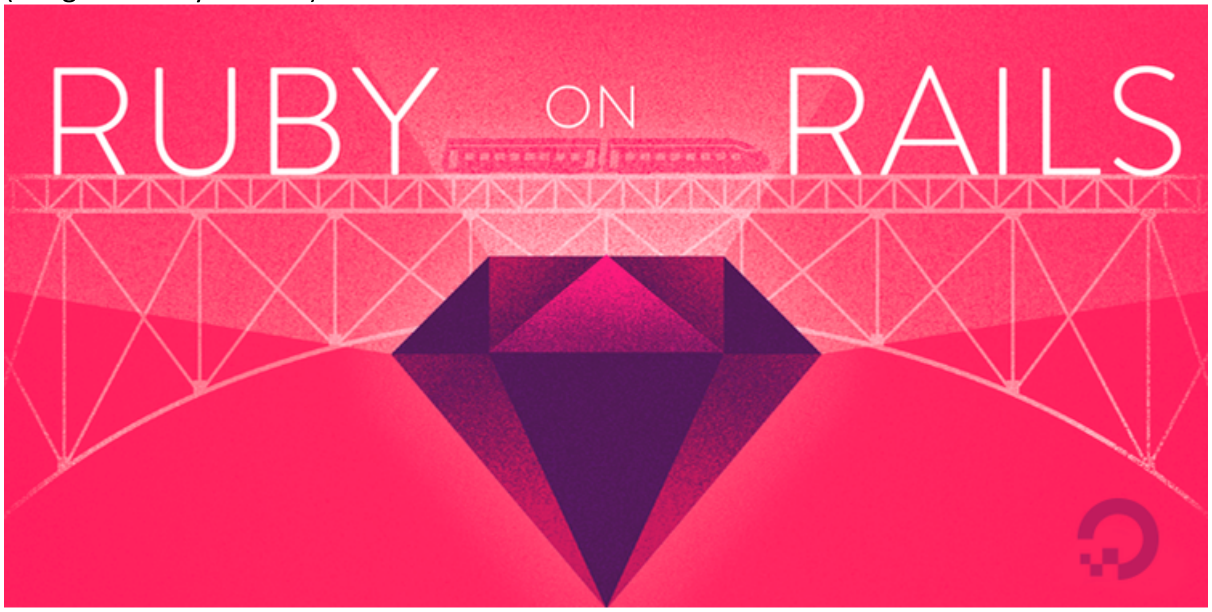
To begin planning this project I knew I needed a way for users to both grant access and revoke access privileges to users in a simple and intuitive way. For this, a web application was required. After considering options such as:

- Django
- .net
- Ruby on Rails

I have experience with these three frameworks, and each framework offers certain benefits to the project. I performed some research into which one would suit the project the most.

Firstly, ruby on Rails. Ruby on rails is a web development framework designed to work with the ruby programming language. Ruby is a pleasant language to write in and has a huge active community contributing to its success. There is a plethora of gems or plugins available for ruby developers to use that are open source and quick to implement. it offers simple generation of scaffolds using the rails command line. This means that I would be able to generate the MVC files needed and simple implementation of APIs using rails' powerful package system.

(image 3 - ruby on rails)



the second choose would be Django. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Django offers huge benefits for programming with devices such as the raspberry pi as it is built around the python programming language. Making it easy to integrate into the project.

the third option is Microsoft's .net framework. Microsoft's .net framework is not so much a framework but rather a collection of application programming interfaces and shared library code that can be used to develop applications. This means that you don't have to write the code from scratch. In the .net framework the shared libraries of code are called the Framework class library, or FCL. Buts of code in the FCL can be shared and accessed for use in programming a whole wide variety of applications.

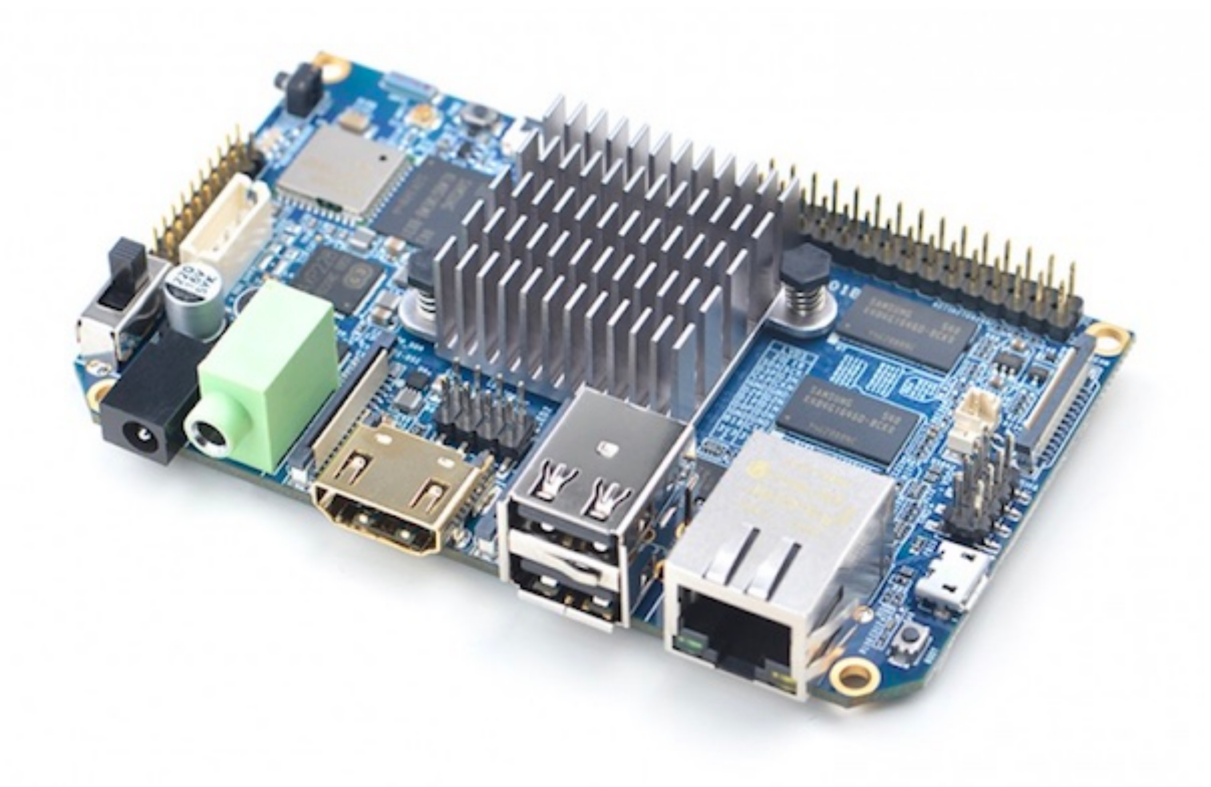
For the purposes of this project I have decided to use the web framework Ruby on Rails for its rapid content creation and extemporary package management system, ruby gems.

With the framework for the web application chosen, I needed to research what use of controller devices would best suit this project. from my research, I narrowed it down to three possible devices.

- NannoPC-T3
- Raspberry pi
- Asus tinker board

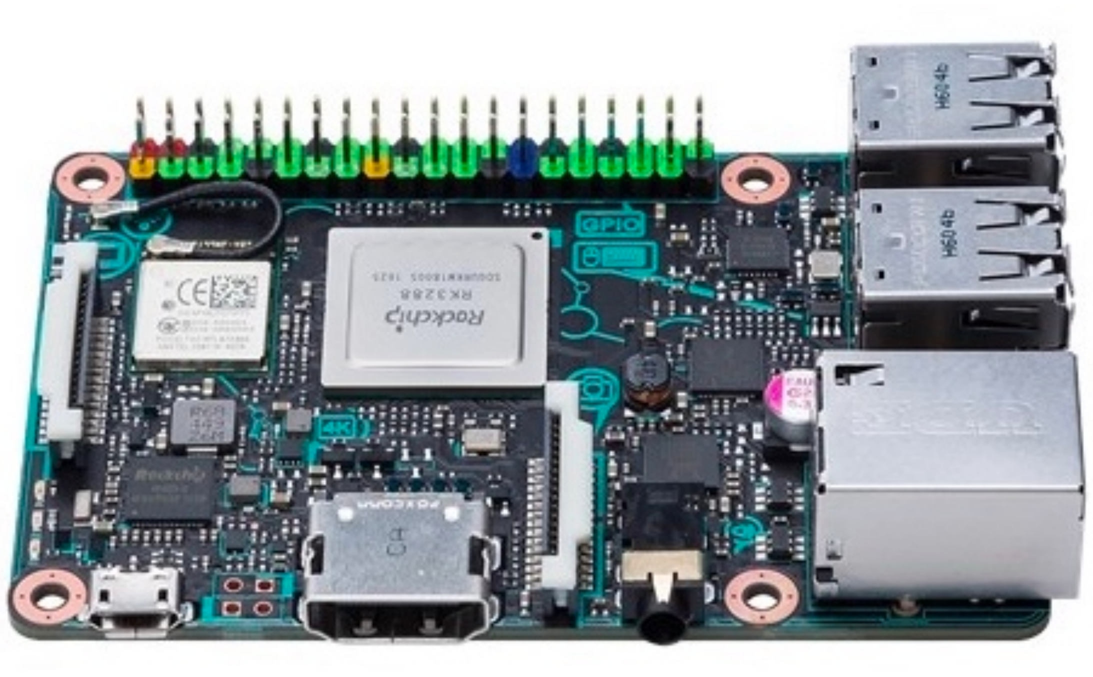
Each of these boards runs a variation of the Unix operating system, which was a requirement for this project as I would be developing on mac os, this would mean less teething problems down the line when new features and new packages needed to be implemented. For this project to remain affordable for users I needed to pick a cost-effective solution, but it also required adequate processing power and features to allow devices to be attached to them easily and effectively.

(image 4 - NannoPC-T3s)



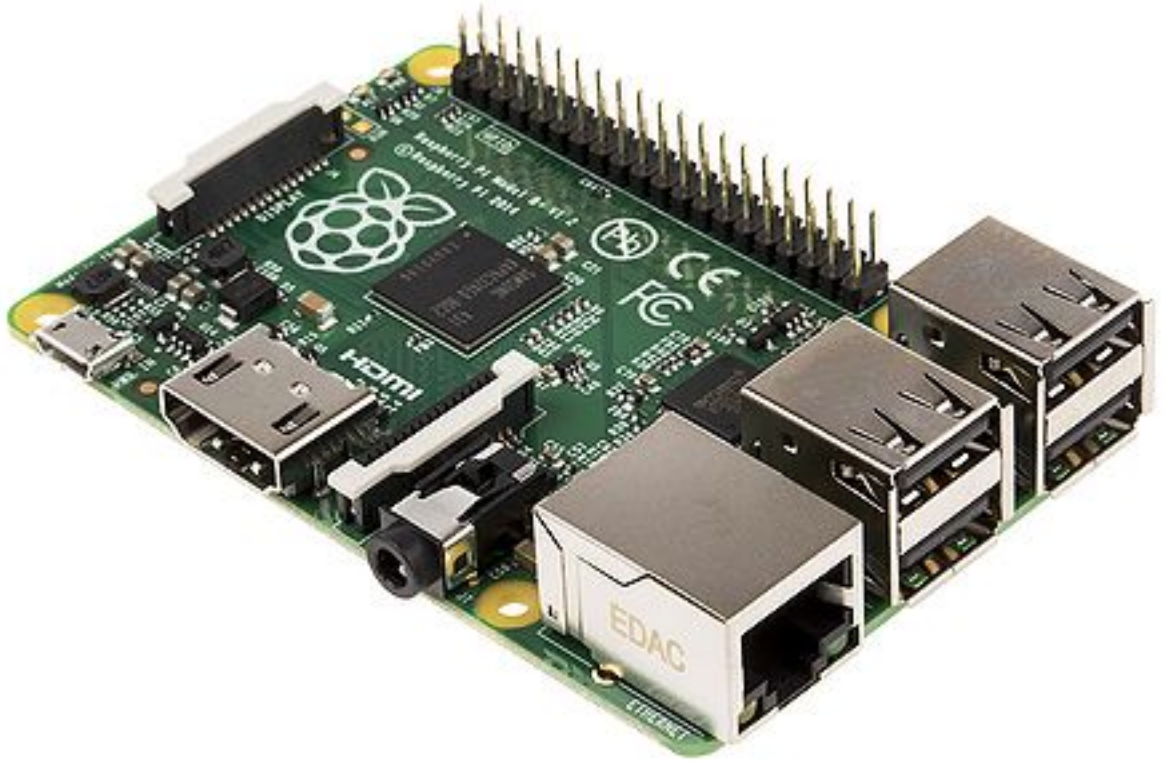
The NannoPC-T3
NanoPC-T3 is a single-board computer developed by FriendlyArm. It houses Samsung's octa-core SoC clocked @1.4GHz, Mali 400 GPU and 1GB DDR3 RAM. It supports microSD expansion, Wi-Fi, Bluetooth and an on-board microphone. It can run Android 5.1, Debian, Ubuntu Core operating systems. It is equipped with 8 GB of storage and is extensible via microSD card. It includes 4 USB 2.0 ports, HDMI (1080p support) and Ethernet support. Other multimedia ports include LVDS, LCD, MIPI-DSI and the 3.5mm audio Jack.
NanoPC-T3 CPU is clocked slightly higher than Raspberry Pi 3's CPU but features the same amount of RAM. One advantage it offers over the Pi 3 is that it comes with an inbuilt storage of 8GB. Other than these things, both are pretty neck-to-neck with each other.

(image 5 – Asus tinker board)



Perhaps the newest entry in this list, Tinker Board is a single-board computer from Asus. Along with a nice name, it also comes with some great specifications. On the hardware side of things, it is equipped with a quad-core CPU (Cortex A17) clocked at 1.8 GHz, Mail-T764 GPU, 2GB of RAM, 4 USB 2.0 ports and an HDMI port. The Tinker Board comes with a custom OS named TinkerOS, which is based on Debian. The headline feature is the support for 4K video playback for videos encoded with H.264 or H.265. It also supports Wi-Fi and Bluetooth 4.
Tinker Board beats Raspberry Pi 3 by miles in benchmarks, thanks to its more powerful processor and an extra gig of RAM. The Tinker Board brings support for 4K videos, whereas Raspberry Pi 3 model B supports (only!) 1080p videos. It also brings support for 192k/24bit audio playback support compared to Pi's 48k/16bit. It consumes a little more power than Pi, but that shouldn't be much of a problem because it's very marginal. The Tinker Board is all good but is available only in the UK, as of now. If you're willing to shell out almost twice what you'd otherwise pay for Raspberry Pi, the Tinker Board can be a great board to tinker around!

(image 6 - Raspberry Pi B+)



The Model B+ is the final revision of the original Raspberry Pi. It replaced the Model B in July 2014 and was superseded by the Raspberry Pi 2 Model B in February 2015. Compared to the Model B it has More GPIO. The GPIO header has grown to 40 pins, while retaining the same pinout for the first 26 pins as the Model A and B. More USB. We now have 4 USB 2.0 ports, compared to 2 on the Model B, and better hot plug and overcurrent behavior. Micro SD. The old friction-fit SD card socket has been replaced with a much nicer push-push micro SD version. Lower power consumption. By replacing linear regulators with switching ones we’ve reduced power consumption by between 0.5W and 1W. Better audio. The audio circuit incorporates a dedicated low-noise power supply. Neater form factor. We’ve aligned the USB connectors with the board edge, moved composite video onto the 3.5mm jack, and added four squarely-placed mounting holes.

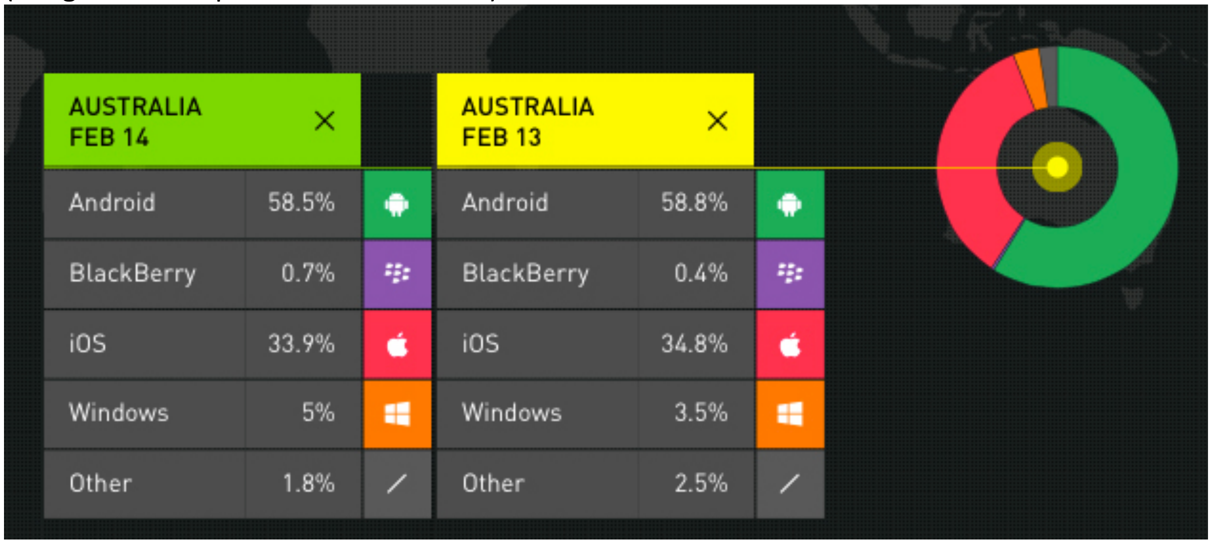
In the end, I decided to go with the raspberry pi b+ model. this decision is mainly because that is the device I have the most experience with, and its easy integration with IOT sensor devices, wireless abilities with the use of a simple Wi-Fi dongle and the easy SSH capabilities for wirelessly editing files on the machine. I have also grown accustomed to the Raspian operating system, and am comfortable with developing on it.

The last major part of the project that required serious investigation before work could commence was the method in which the communication between the web application, the user's smartphone and the IOT controller device would be handled. This would be the most pivotal decision for the project as it would have to have many features:

- Cross platform
- Non-intrusive
- Available to everyone
- Free to use

This is a tall order. Luckily the options for such a system are plentiful, however there was no clear 'best' option. I had to decide whether to develop an application, use a pre-existing multi-platform solution or find an application that ticked these boxed and implement it in such a way as to have it perform the actions I required it to. My options were as follows.

(image 7 –smartphone os breakdown)



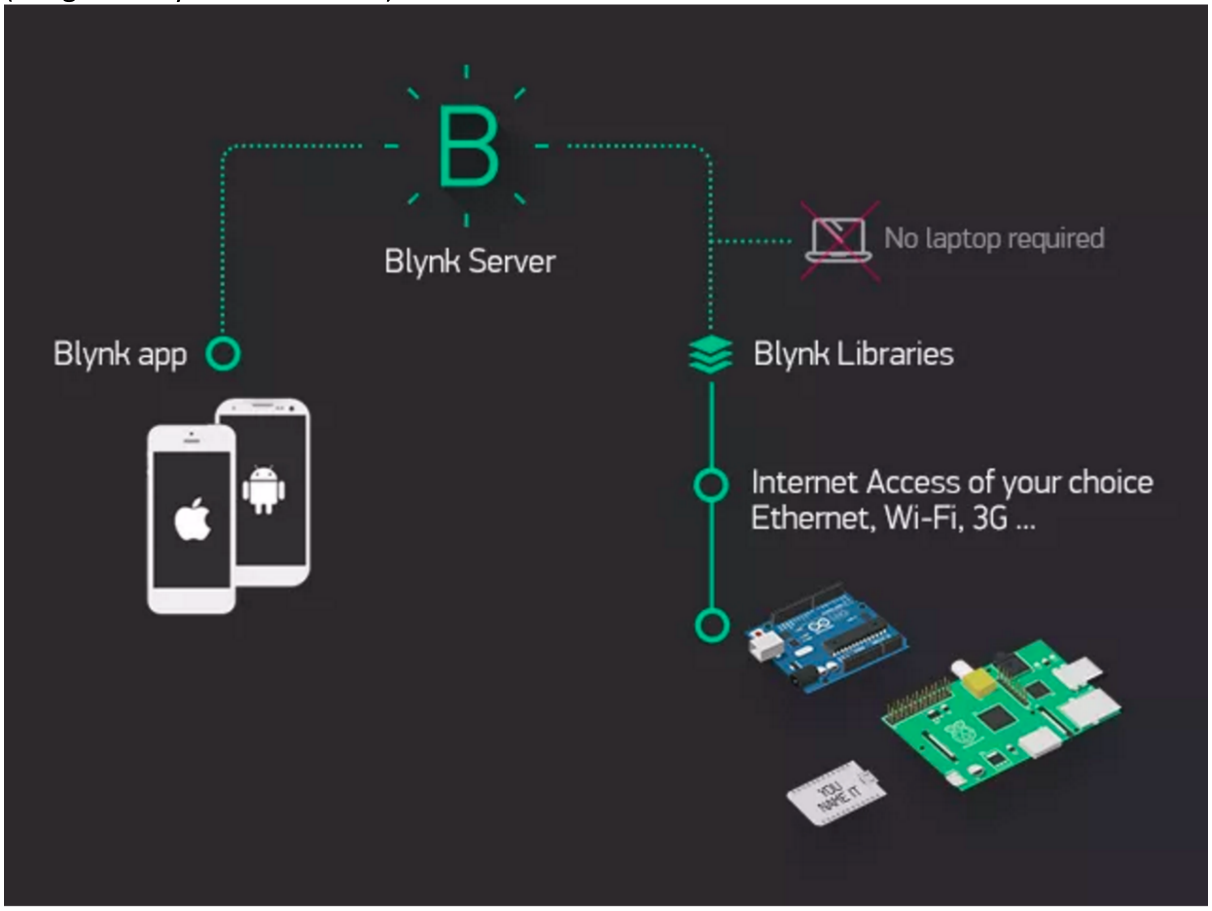
AUSTRALIA FEB 14		AUSTRALIA FEB 13	
Android	58.5%	Android	58.8%
BlackBerry	0.7%	BlackBerry	0.4%
iOS	33.9%	iOS	34.8%
Windows	5%	Windows	3.5%
Other	1.8%	Other	2.5%

development of an application

for a long time this was what I considered to be the best option. However, this would require the development of an application that worked on most if not all the operating systems currently being used by smartphones. Further research revealed that creating a smartphone app that would work on all systems would prove difficult. It became apparent that in order to have a truly cross platform application to handle the requirements of the project I would have to develop multiple applications using multiple languages. Something that would have gone way and beyond the scope, and time of this project.

the next option was to use an IOT development tool for android application. Blynk is a new platform that allows you to quickly build interfaces for controlling and monitoring your hardware projects from your iOS and Android device. After downloading the Blynk app, you can create a project dashboard and arrange buttons, sliders, graphs, and other widgets onto the screen. Using the widgets, you can turn pins on and off or display data from sensors.

(image 7 - Blynk architecture)



Blynk was perfect for my purposes, except for one major flaw, the cost. Blynk is a paid service. They do however offer a free service, but it would not have met my requirements. To get the service I required would have cost me at minimum $199 dollars a month or $1990 a year, and for a college project of this scale this proved not feasible for my purposes.

Finally, after extensive research I found the solution. A messenger service called Telegram.

(image 8 - Telegram)

Telegram is a messaging app with a focus on speed and security, it's super-fast, simple and free. You can use Telegram on all your devices at the same time — your messages sync seamlessly across any number of your phones, tablets or computers.
With Telegram, you can send messages, photos, videos and files of any type (doc, zip, mp3, etc.), as well as create groups for up to 5000 people or channels for broadcasting to unlimited audiences. You can write to your phone contacts and find people by their usernames. As a result, Telegram is like SMS and email combined — and can take care of all your personal or business messaging needs

telegram is the perfect service for this project. it is widely used across all popular platforms. There would be no learning curve for this application as everyone has used a messenger service before. Telegram has an open api allowing for development on the raspberry pi, python support and ruby support. This would be the final piece of the puzzle.

Recap

Recap of what options I picked for my project from the research conducted:

- Chosen web framework: Ruby on Rails
- Chosen IOT device: Raspberry Pi B+
- Chosen Application: Telegram

4 Aims

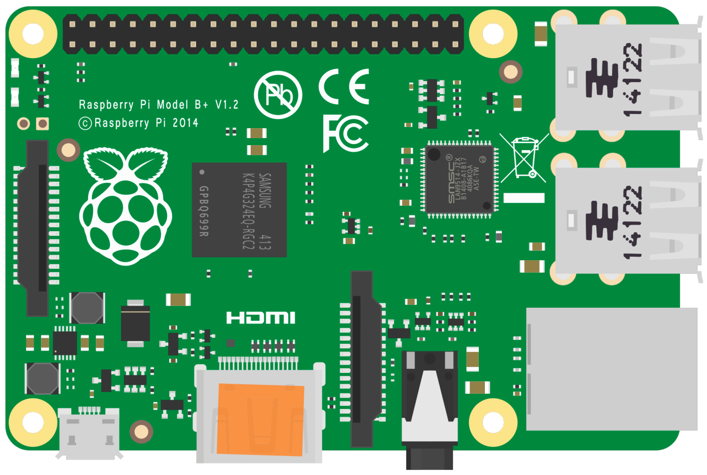
The aim of this project is to create a smart home entry system using IOT technologies. There are a number of solutions on the market, however there are no cross platform solutions that utilise already established applications that require little to no user experience to operate. The solutions on the market already such as RFID and PIN entry systems are not capable of adding and removing users easilty and effectively. there is either a cost to allowing new users or having to make the entry process the same for everyone making the device succeptible to hacking.

This device will tackle and hopefilly improve upon the devices present today. With Open Sesamessage I hope to allow for a cost effective home entry system available to everyone

5 Technologies

Hardware

- **Raspberry Pi Model B+**
  - The raspberry pi b+ will be used as a controller station for this project's main functions. It will be responsible for opening and closing the lock
  - Responsible for taking pictures of who is entering the building
  - Filtering out unknown users who try to gain disallowed access



802.11n Wi-Fi dongle:
allows the device to connect wirelessly to the internet.

- **servo motor**
    - Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire.
    - The control wire is connected to the raspberry pi and controlled using python
    - The motor will be used to open the lock on the door
    - The model used: TowerPro mg995

-

- **Webcam**
    - The webcam will be used to take photos on request of the admin user. This will allow them to see who is coming and going through their property. It will also alert them the malicious intent.
    - Model used: Microsoft LifeCam Cinema 720p

Software

List of software being used:

- Atom
- OSX Terminal
- Github
- Waffle.io
- Travis
- nano

Frameworks/languages

List of frameworks/programming languages used:

- Ruby on Rails
- Ruby
- Python
- Lua
- HTML
- CSS
- Tg: telegram API
- PostgreSQL

Brief how the software will be used:

The technical difficulty of this project is spread out into three main parts. The creation of the web application using ruby on rails, the establishment of the Raspberry Pi and its devices and the implementation of Telegram's api to allow communication between the web application and the IOT device

Ruby on rails

Ruby on Rails will be used for the following:

- a base for the admin user
- facilitate permission of new users
- revoke permissions of users
- contact service
- communicate with raspberry pi

python
What I will use it for is the following:

- **IOT functionality:** this is the programming language that I will use to operate the sensor devices and motors connected to the raspberry pi

Lua
Lua is a lightweight scripting language, I will use it for is the following:

- **Interpretation of messages:** I will use Lua to read the messages sent by users and perform actions based on their commands. Such as triggering the python code to open the door upon request
- **Get commands:** to get commands from the web application adding new users to the approved list, or to remove users from the list
- **Take photos:** to trigger python program to take photo, then retrieve the image from the file directory and send it back to the user

HTML/CSS/Bootstrap
What I will use it for is the following:

- Styling of the web application

Telegram
What I will use it for is the following:

- **Communication:** this cross-platform messenger service will act as the basis for all communication between the web application and the IOT device, as well as users and the raspberry pi

6 Full Project Concept Breakdown

The entire concept of the project is to create a home entry system that it is:

- Security focused
- Easy to setup
- Easy to use
- Intuitive
- Works in conjunction with traditional methods
- Unobtrusive
- Cost effective

7 Structure

The structure of this document is as follows. I will describe the system and its architecture pointing out its main functionalities and requirements, including the projects non-functional requirements. I will then discuss the projects' design and architecture describing the relationships between the web application, the OpenSesamessage device and the users smartphone. I will discuss the creation of the prototype and the process in which I created the door and the integration of the IOT device into the prototype. I will then discuss the Graphical User Interface in which the user will operate the system, testing of the application and the market research performed. And Finally the conclusion

8 System

Requirements

The requirements specification will outline several variables that the user will achieve while using the system, they are as followed:

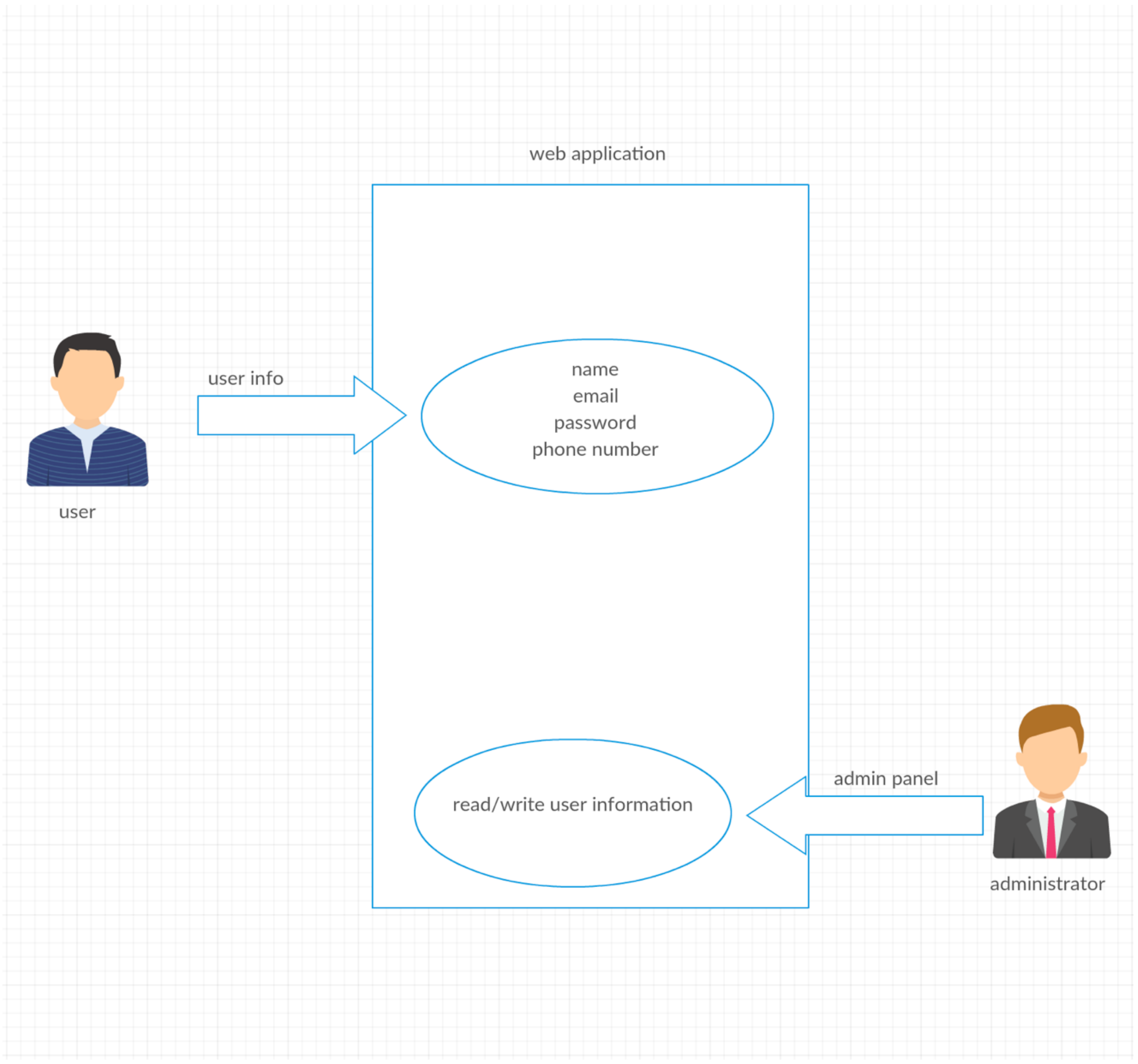
1. The user will be aware of the workings of the system, and how to control the device within 10 minutes of introduction
2. The user will understand the simple commands
3. The user will be able to add, update and remove allowed people on their device
4. New users will share the first two requirements

Functional requirements

Requirement 1 Registration

This is the registration process for the application
Description & Priority
This process however trivial is essential to access the features and functionality

Use Case



Scope

The scope of this use case is to handle the registration process

Description
This use case allows the user to create an account entering an email, password, phone number and their name

Flow Description
Precondition
This use case is called into action when the user enters the registration screen. From their they enter their details and submit them
Activation
This use case starts when the user registers with the web application
Main flow

1. The system accepts the user information and creates a user profile in the database
2. The User creates their account
3. The user receives an email notifying the success of their registration
4. The system accepts the information and creates a database entry
5. The admin can access this information through the admin panel

Alternate flow

1. Administration requests user information
2. The system responds to the request
3. The application populates the screen with a table of user information

Exceptional flow

1. The system encounters an error
2. The system returns an error message to the user
3. The system notifies the administrator

Termination

The system terminates when the registration process is complete

Post condition

The system goes into a wait state

Requirement 2 Password Reset

Description & Priority

This requirement allows the user to reset their password, instructions will be sent to the users email address, they will enter a password reset page

Use Case

The user will request a password change, and the system will update the password entry in their information

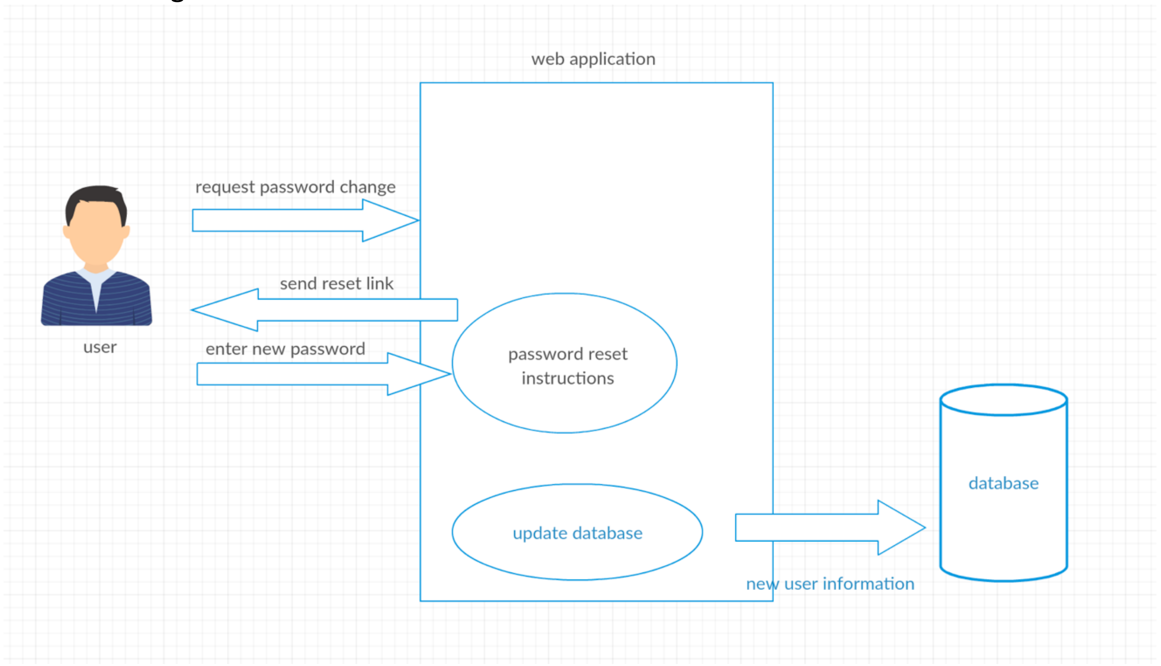
Scope

The scope of this use case is to allow the user to change their password

Description

This use case describes the interaction between the user and the system in this application requirement

Use Case Diagram



Flow Description

Precondition

The system is initialized when the user opens the password reset view

Activation

This use case starts when a requests a password change

Main flow

1. The system accepts the users email information
2. The user requests an email reset link
3. The system sends a password reset link
4. The user proceeds to the password reset view
5. The system accepts the information and updates the database

Alternate flow

1. The system cannot find user email

2. The system displays an error message
3. The use case returns to its initial state

Termination

The system updates the database

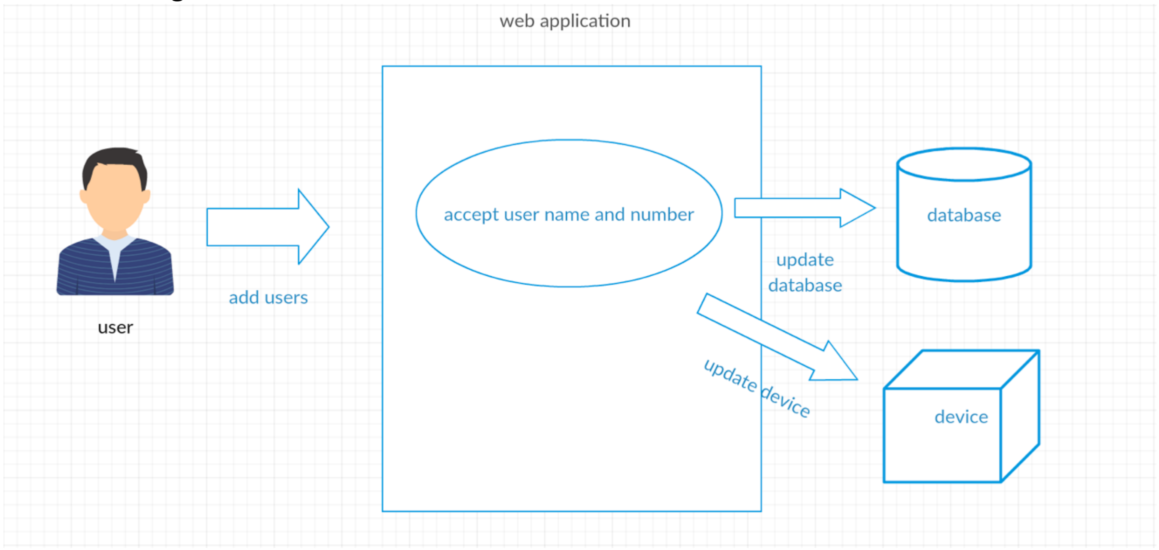
Post condition
The system goes into a wait state

Requirement 3 Add Users

Description & Priority
This requirement allows users to add users to their account. These users will be stored in the database, as well as be sent to the device to be stored locally
Use Case
The user will add the users from their user dashboard

Scope
The scope of this use case is to allow the user to add users

Description
This use case allows for a user to add secondary users to their account, these names and numbers will be able to access the device
Use Case Diagram



Flow Description

Precondition
The system is initialized when the user enters their user dashboard view

Activation
This use case is activated when new information is entered into the fields

Main flow

6. The system accepts the new user's information
7. The system updated the user's database entry
8. The system sends the new users to the device
9. The user gets confirmation
10. The device and database go into a wait state

Alternate flow

4. The system encounters an error

5. The system displays an error message
6. The use case returns to its initial state

Termination
The system updates the database and device

Post condition
The system goes into a wait state

Requirement 4 Open device (message)

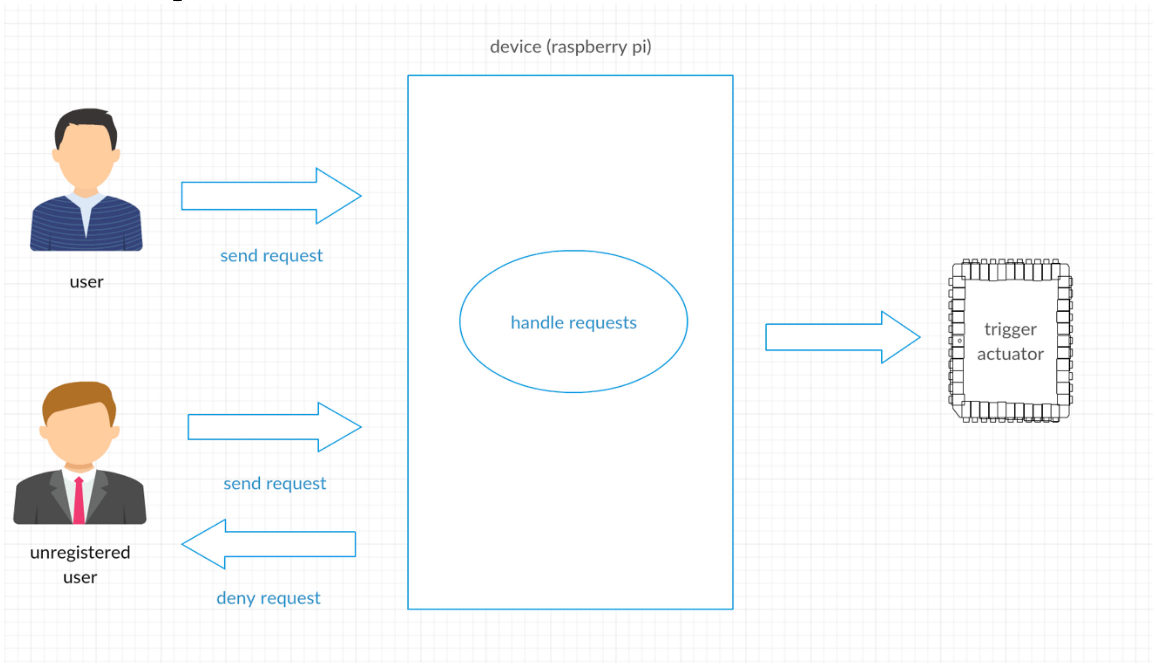
Description & Priority
The requirement allows the user to send a message to the device and request the device to open

Use Case
The user message the device using a popular messaging application

Scope
The scope of this use case is to allow the trigger the device actuators via smartphone

Description
This use case allows the user to message the device and request it open

Use Case Diagram



Flow Description

Precondition
The precondition is created when the user has the number for the device

Activation
The use case is activated when the device is messaged

Main flow

1. The user sends a message to the device
2. The device interprets the message
3. The system activates the actuator
4. The device goes into a wait state

Alternate flow

1. The system interprets the message

2. The system does not approve of the message
3. The system returns a message of disapproval
4. The device goes into a wait state

Termination

The system will return a message

Post condition
The system goes into a wait state

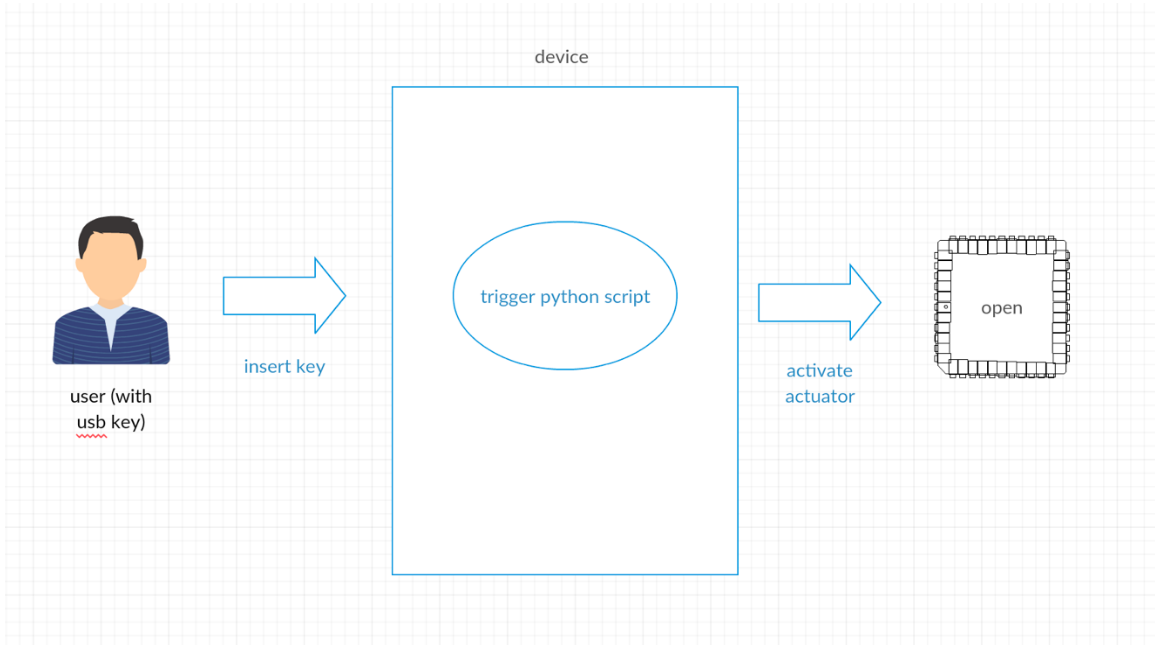
Requirement 5 Open device (key)

Description & Priority
The requirement allows the user to use a USB key to trigger the actuator and open

Use Case
The user inserts the USB key into the device trigger the actuator

Scope
The scope of this use case is to allow the trigger the device actuators via USB

Description
This use case allows the user insert USB the device and request it open
Use Case Diagram



Flow Description

Precondition
The precondition is when the device is on

Activation
The use case is activated when the USB is inserted

Main flow

5. The user enters the USB
6. The device triggers the open program
7. The device activates the actuator
8. The device goes into a wait state

Alternate flow

5. The system cannot interpret the command
6. The system does not approve of the command
7. The system returns false
8. The device goes into a wait state

Termination
The system opens the device

Post condition

The system goes into a wait state

Requirement 5 Get device status

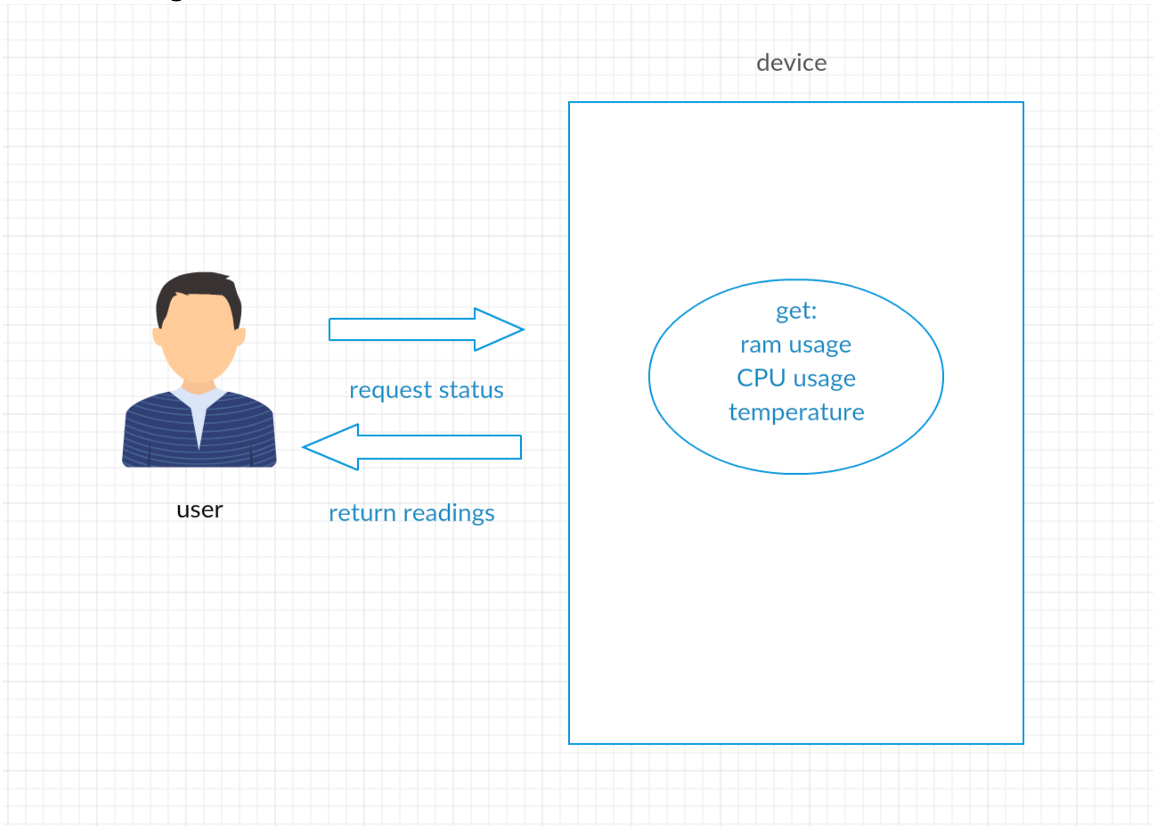
Description & Priority
This requirement allows the user to message the device requesting status information

Use Case
The user requests information via message

Scope
The scope of this use case is to get status information

Description
The user messages the device requesting information such as CPU usage, ram, and core temperatures

Use Case Diagram



Flow Description

Precondition
The precondition is when the device is on

Activation
The use case is activated when the user requests device status

Main flow

1. The user requests status readings
2. The device returns sensor readings
3. The device goes into a wait state

Alternate flow

1. The request is not sent by an approved number
2. The system responds with a disapproval message
3. The system returns false
4. The device goes into a wait state

Termination
The system returns information

Post condition
The system goes into a wait state

Requirement 6 Password Reset

Description & Priority

This reqirement allows the user to message to device and request an image to be taken, and responded to that message with the .jpeg

Use Case

The user will request an image, the system will execute the image.py program to take a picture, save the image and the Lua will respond to the user with the image.

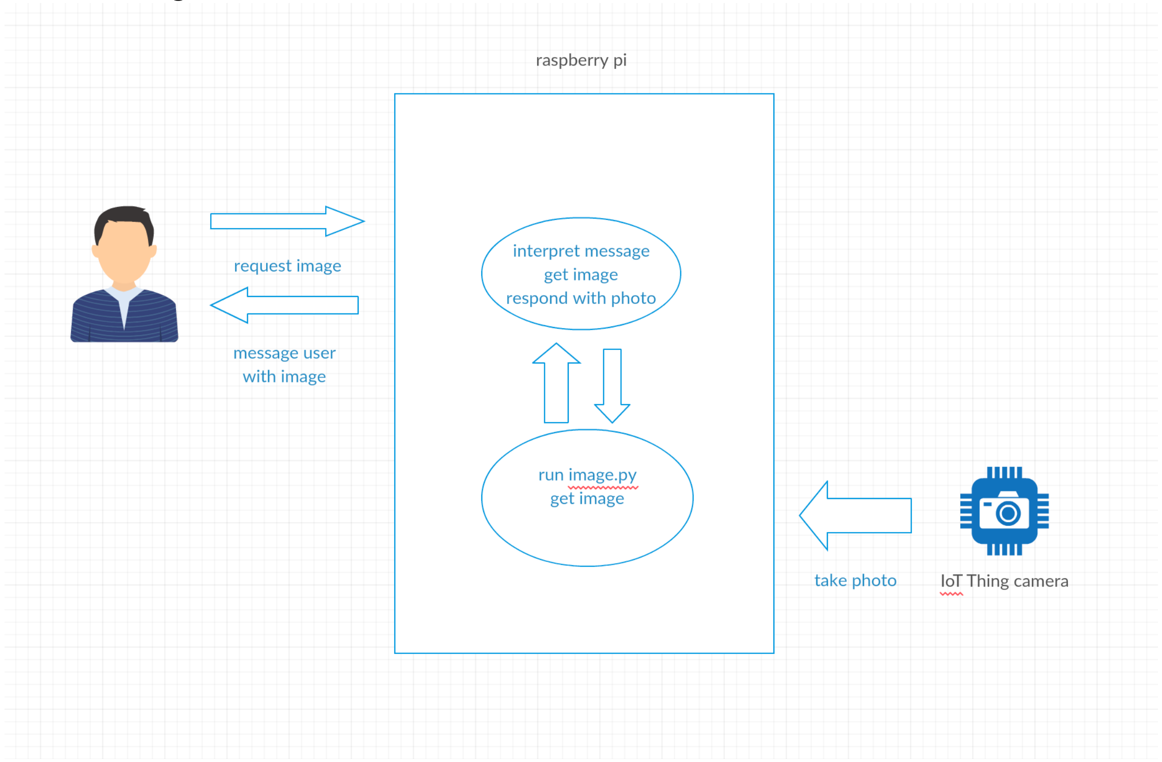
Scope

The scope of this use case is to allow the user to request an image, have the device take it and respond with a multi media message containing the captured image

Description

This use case describes the interaction between the user and the raspberry pi in this application requirement

Use Case Diagram



Flow Description

Precondition

The system is initialized when the user messages the device requesting an image

Main flow

11. The user requests an image
12. The system interprets the message
13. The system responds with an image to the user

Alternate flow

7. The image fails

8. The system responds with an error message
9. The use case returns to its initial state

Termination

The system responds to the message

Post condition

The system goes into a wait state

Non-Functional Requirements

Multi-platform support

This performance is measured by the accessibility to as many users as possible. The web application allows for access for anyone with a modern web browser, while the telegram application allows for access to anyone with a smart phone running one of the main operating systems

Supported web browsers

- Microsoft edge
- Google chrome
- Firefox
- safari
- internet explorer

supported messenger platforms
iOS

- android
- Microsoft windows phone
- OSX
- all main Linux distributions

the cross-platform aspect of this project is paramount. The main non-functional requirement for this project is to allow users access to the project regardless of the systems that they're running.

User requirements

The user requirements are as follows:

- The user's requirements are their smartphones. It must have one or more of these:
    - Android smartphone
    - IOS smartphone
    - Microsoft Windows smartphone
    - Chrome, Firefox, safari
    - Desktop or laptop with mac os
    - Desktop or laptop with windows
    - Desktop or laptop with Linux

- The user must have the telegram app installed on one of the above devices
- The user must have a telegram account
- The raspberry pi connected with stepper motor and webcam

Availability requirement

requirement 1: web application

- This requirement would be a web application accessible to anyone using any of the main web browsers
- This requirement would be of the maximum uptime possible with modern web hosting. Digital ocean offers multiple servers, with load balancing which will contribute to maximum uptime
- It would also have to offer control of the users and they're sub user's information. I.e. the user's dashboard

requirement 2: IOT availability

- The availability of the raspberry pi is a requirement
- The uptime of the raspberry pi will be based mainly on the network capabilities of the user's location.
- The network provided by the user will have to be consistent for the server to remain open for incoming messages from telegram
- The python programs will need to be written with error handling so that interrupts, such as accidental keyboard interrupts will not halt the program

- The Lua program will have to be written in such a way that it will be able to recover from any interruptions
- The system is required to run for an indefinite amount of time, there will have to be systems in place to clear the cache and remove the images taken once sent to the user to avoid filling the system's memory

requirement 3: telegram

- This requirement is for the user to have a unique phone number to create a telegram account if they do not already have one
- The network capabilities of the user's phone is another requirement, the need to be able to access the telegram servers is an important requirement
- The ability to use telegram outside of the scope of this project as a messenger service

Security requirement

Security for this project is an important aspect for the scope of it. The users need to be able to send messages securely between the IOT device and their phones. One of the main reasons I chose to use telegram is because of its exemplary encryption. Telegram offers end to end encryption meaning that the only devices capable of decrypting the messages are the device that sent the message and the device that received the message

Reliability requirement

This requirement is straight forward, the Tg server needs to be called and once running and the tgl repository will maintain it until the safe quit function is called.
The servers also need to be running and accessible to anyone on the web

Maintainability requirement

The maintaibality of the project will be managed through Github. Updates to the repository will be able to be called using the git command git pull in the terminal. This will merge the updated files in the Github repository with the local files on the machine Telegram offers updates as standard, as does its well-maintained API.

Extendibility requirement

the main code for this project is easily updated and extended. The project would benefit from a plethora of new functionalities.

Reusability requirement

The code for this project will work on multiple raspberry Pi devices, it just requires to be setup with a new number.

9 Design and Architecture

In this section I will describe the design and architecture of the project, including the web application, the raspberry pi IOT device and the smartphone application required to operate the system.

Implementation

Each implementation of this project will span throughout the web application, the telegram functionality and the working project. along with images and code snippets.

Web application implements

This section of the implementation will cover the overall web application systems, these systems are crucial to the running of the website and are linked to many of the functional requirement's listed and documented above.

User account creation

Creates a user account, action mailer, stores in database

- **Name:** used to track username, allows User mailer to address the user by their name when getting in contact
- **email:** This used in conjunction with the user_id is the user's unique identifier. Email is used by the mailer, it plays an important part in password reset functionality
- **password:** the password is encrypted using becrypt through the devise gem
- **sign up:** triggers a user save function which triggers user mailer. Saves the user to the database

Implementation & code snippets:

At the account creation, the User controller handles saving the user and runs through the sign-up code process.

```
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  has_many :authorised_users
  has_many :numbers

  after_create :send_admin_mail
  def send_admin_mail
    UserMailer.send_welcome_email(self).deliver
  end
end
```

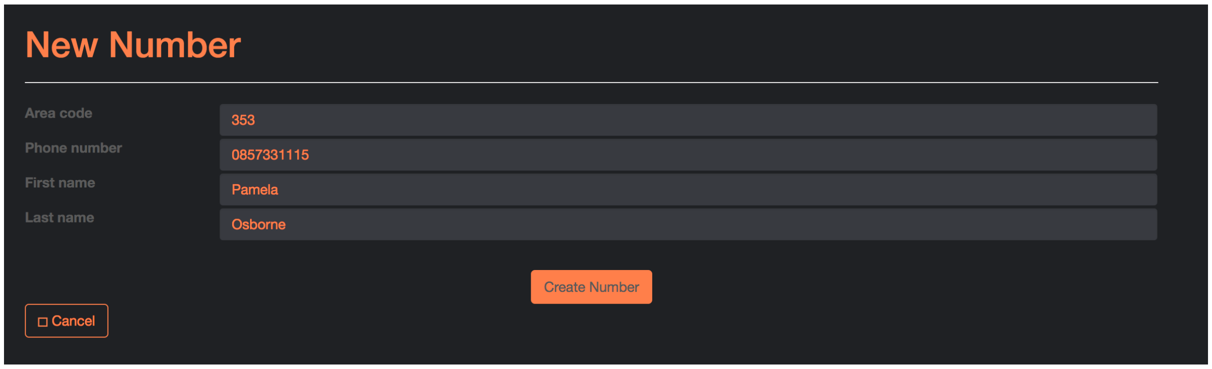
after the account is created the user is designed an association with the numbers model. each user can have many numbers associated with it. This allows for the user to add. Modify and delete the list of numbers they want to allow access to the OpenSesamessage service

Granting access to users

Controls, Maintains and Implements: Large blueprint system some systems will be broken down and detailed in sections, this section will detail first options the user has access to.

- **Add number function:** allow the user to add numbers to the device, these numbers will have the privileges required for accessing the raspberry pi through the telegram messenger
- **Edit function:** allows the user to update phone numbers if a person changes phone, update any field necessary to maintain an up to date record of each permitted number
- **Delete function:** allows the administrator user to delete numbers, this revokes their privileges for accessing the raspberry pi through the telegram service. This allows for granting temporary access to a user rather than lending them a key.

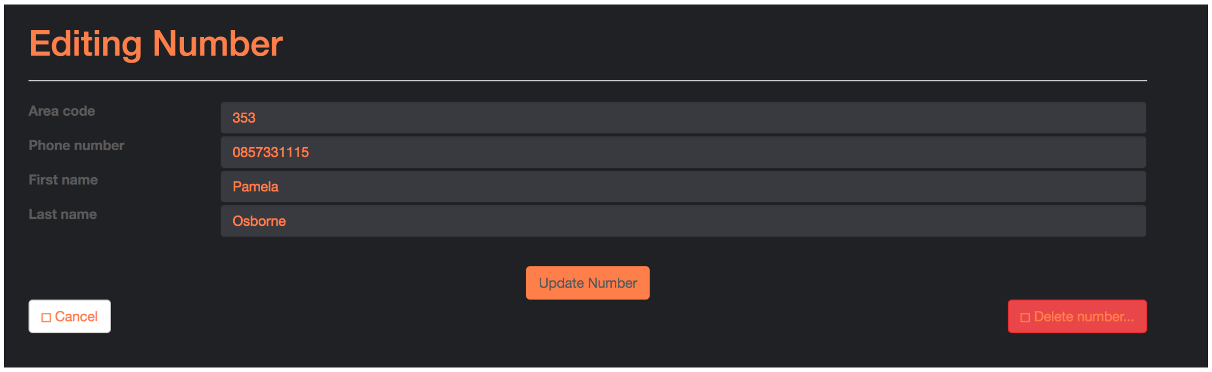
Implementation & Screenshots:
Add User:
Allows the addition of new numbers:


New Number

Area code	353
Phone number	0857331115
First name	Pamela
Last name	Osborne

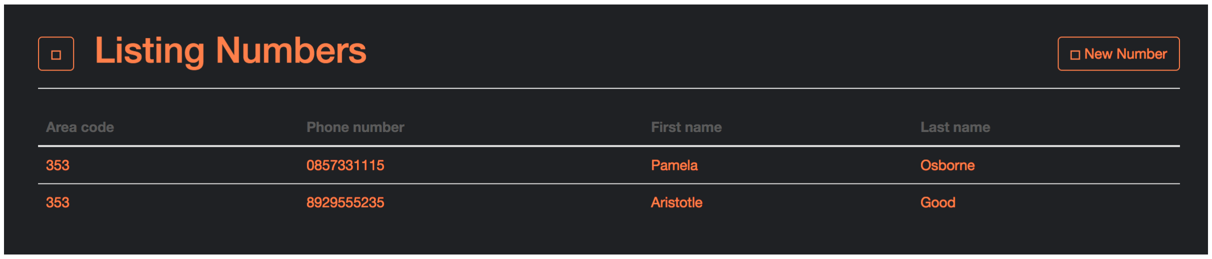
Edit User: calls the edit view, and provides a new form for altering records. Also provides a delete button for quick and easy removing of numbers from the system

Once the form is completed, it calls upon the Number.update function to alter the active record entry associated with the number


Editing Number

Area code	353
Phone number	0857331115
First name	Pamela
Last name	Osborne

index view: the index view allows for a visual representation of the numbers with privileges required to access the raspberry pi through telegram opening the door


Listing Numbers

Area code	Phone number	First name	Last name
353	0857331115	Pamela	Osborne
353	8929555235	Aristotle	Good

authenticate user

Controls, Maintains and Implements:

- **Add number as authenticated user:** allows the user with the designated phone number to access the service
- **On create function:** on create of the Number record a rails call sends a message to the raspberry pi via telegram containing the user details
- **Add user to contacts:** the incoming telegram message is interpreted by the raspberry pi device

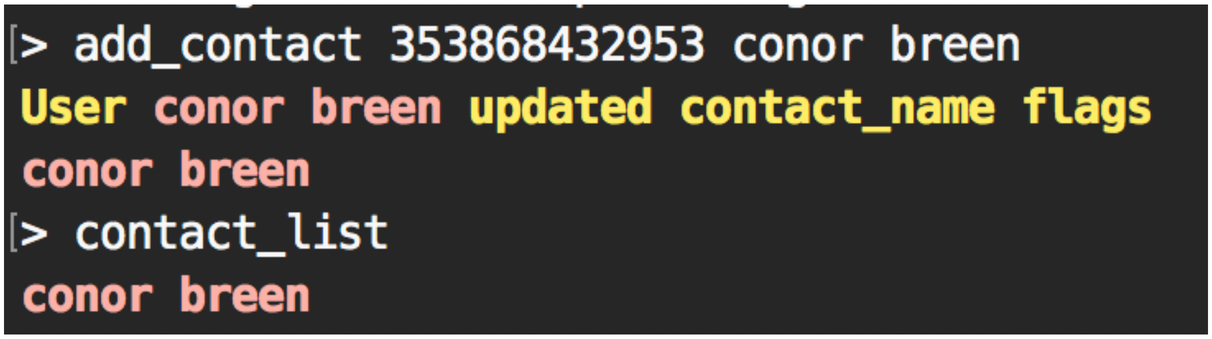
Implementation & Screenshots:
Add number function: is an on create event, its overall function is to grab the Number record information and pass it through telegram to the raspberry pi
Its first function is to create the number, as illustrated above
The second function lies in the on create function of the Number record, that sends a message via telegram

```
def create
   @number = current_user.numbers.build(number_params)
   TeleNotify::TelegramUser.find(1).send_message("#{@number.phone_number} ,
#{@number.first_name} , #{@number.last_name}")

   # @number = Number.new(number_params)

   respond_to do |format|
    if @number.save
      format.html { redirect_to @number, notice: 'Number was     successfully created.' }
      format.json { render :show, status: :created, location: @number }
    else
      format.html { render :new }
      format.json { render json: @number.errors, status: :unprocessable_entity }
    end
   end
  end
```

as you can see above, once the user is created the @number.phone number, @number.first_name and the @number.last_name variables are passed into the telegram TeleNotify method to pass these arguments onto the raspberry pi
The message is interpreted by the action.lua code, and returns a cli command to add_contact. Once the contact is added to the raspberry pi it will have the permissions required to access the services.
Note: a user with the privileges of triggering the open-door function does not have access to the camera function, this feature was added as it would seem a security violation to have anyone able to access the camera.

```
function on_msg_receive (msg)
  if msg.out then
    return
  end
  if  (msg.from.print_name == "opensesamessage") then
      string = msg.text
      number, first_name, last_name = string:match("([^,]+),([^,]+),([^,]+)")
      add_contact (number, first_name, last_name, ok_cb, false)
  end
end
```

once the message is received, the function on_msg_recieve takes in the new message, checks that it came from the rails application
In the telegram cli console you can see the message being received. The message contains the phone number, first name and second name.
As it stands this string contains all the required information to create a contact, however it is one string. This string to prove useful needs to be converted into variables. As you can see in the code above this is done with the `number,` first_name, last_name string:match ("([^,]+),([^,]+),([^,]+)") line. This separated the three needed variables by the "," delimitator.



```
[> add_contact 353868432953 conor breen
User conor breen updated contact_name flags
conor breen
[> contact_list
conor breen
```

mailer implementation

outlines the mail response required for a users' creation, password recovery and the queries put forward using the contact.
all options are handled using the user_mailer model within the rails application, the zoho mail SMTP (simple mail transfer protocol) and the cloud application service cloudflare
Controls, Maintains and Implements:

- **Account creation mailer – user:** triggers an email once the account is created emailing the user welcoming them to the service.
- **Password reset mailer – user:** creates a password reset token for the user, and sends them an email using this token to allow them to set a new password.
- **Mailer contact – user:** the message flow for the web application's contact us form responds to the user's query with an email informing them of the received and with the intent to reply to the email
- **Mailer contact – administrator:** the message flow for this function is for the administrator account, it receives the email containing the user information of the person querying them, and allows them to promptly respond to the message.

Implementation & code snippets:
Account creation mailer: this mailer is triggered once the user creates an account.
Below you can see the code that relates to the welcome mail sent to the user once their account is created

```
class UserMailer < ActionMailer::Base
 default from: "hello@opensesamessage.com"

 def send_welcome_email(user)
  @user = user
  mail(:to => user.email, :subject => "Welcome to opensesamessage.com")
 end
end
```

furthermore, the mailer is called into action once someone sends a message through the contact us form. This creates a new record using the contact model in the contact form in the on create method and triggers the action mailer.

```
class ContactsController < ApplicationController
 def new
  @contact = Contact.new
 end
 def create
   @contact = Contact.new(contact_params)
   if @contact.save
     redirect_to root_path, notice: 'Thank you for contacting us.'
   else
     render :new
   end
 end
 private
   # Use callbacks to share common setup or constraints between actions.
   # Never trust parameters from the scary internet, only allow the white list through.
   def contact_params
     params.require(:contact).permit(:name, :email, :query)
   end
end
```

the mailer is called into action to trigger emails for both the user (the person submitting the query) and the system administrator. The administrator email hello@opensesamessage.com is handled by the service provider zoho mail. The code for this action can be seen below.

```
class ContactMailer < ActionMailer::Base
 default from: "hello@opensesamessage.com"

 def send_contact_notification(contact)
   @contact = contact
   mail(:to => 'hello@opensesamessage.com', :subject => "New contact submitted")
 end

 def send_ack_mail(contact)
   @contact = contact
   mail(:to => contact.email, :subject => "Thank you for contacting us")
 end
end
```

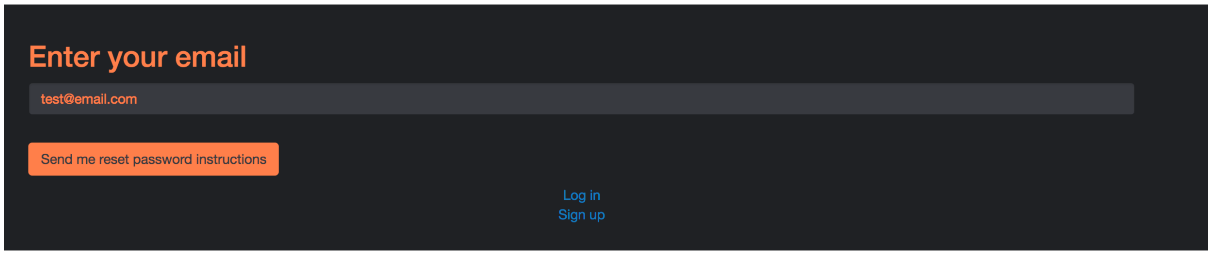
password reset system implementation

outlines how the password reset functionality is handled, with the user Controls, Maintains and Implements:

- **Reset system:** allows the user to update their password record for the system, allowing them to recover an account if they forget their login.
- **Password reset view:** this view allows the user to enter their email address requesting a password change.

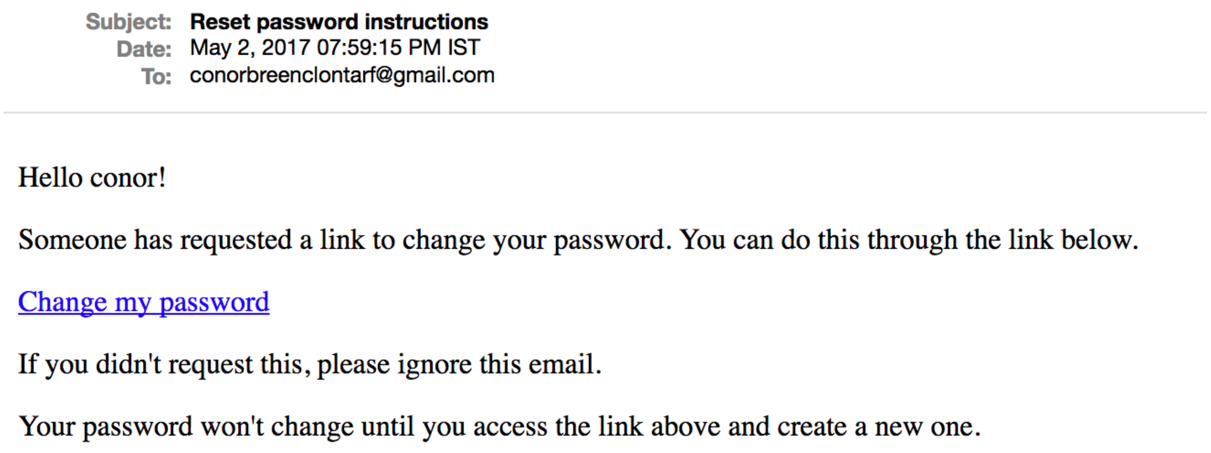
Implementation & Screenshots:

Password reset view: this view allows the user request a change to their account login details. The view for password reset takes in a user's email and if there is a corresponding email listed in the User database it sends an email containing password reset instruction


Enter your email

test@email.com

error handling: if there is no email in the database that matches the email requesting the password change the system will return an error stating that the password was not found on the system.


Subject:	Reset password instructions
Date:	May 2, 2017 07:59:15 PM IST
To:	conorbreenclontarf@gmail.com

Hello conor!

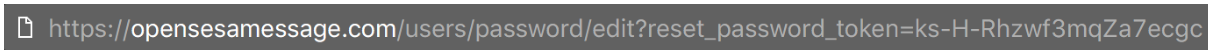
Someone has requested a link to change your password. You can do this through the link below.

Change my password

If you didn't request this, please ignore this email.

Your password won't change until you access the link above and create a new one.

upon clicking the provided link, the user is redirected to a password change view, this view is only accessible with the password reset token generated for the user when they request a password change. Only users with this unique token can access this screen.


https://opensesamessage.com/users/password/edit?reset_password_token=ks-H-Rhzwf3mqZa7ecgc

below is the code that handles the authentication, and calls for a token to be generated:

```
<%= form_for(resource, :html => {class: 'form-signin'}, as: resource_name, url:
password_path(resource_name), html: { method: :post }) do |f| %>
<%= devise_error_messages! %>
<div class="container margin-top-10em">

<div class="wrapper ">
  <h2 class="form-signin-heading">Enter your email</h2>
  <div class="field">
    <%= f.email_field :email, class: 'form-control-custom', autofocus: true %>
  </div>
  <div class="Send me reset password instructions">
    <%= f.submit "Send me reset password instructions", class: 'btn btn-primary-custom
margin-top-2em' %>
  </div>
 <% end %>
  <div class="sign-up-links">
    <%= render "devise/shared/links" %>
  </div>
</div>
</div>
```

raspberry pi status update

Outlines in detail the full device status check system, allowing the user to see the cpu load, ram usage and memory on the device. This is an important part as the device is designed to run indefinitely and remotely accessing the device status is integral to the device's seamless operation

Controls, Maintains and Implements:

- **status:** this message sent to the device through telegram
- **parsed message:** the device integrates the message and preforms a bash command on the system and saves it as a string
- **respond function:** grabs the string handled and replies to the message, with a message containing the system's information. Cpu load, memory usage etc.

Implementation & Screenshots:
Status: is a custom python program I have written contained within the raspberry pi system. Once triggered it returns a string containing system information. This string can then be gathered by a system call triggered within the Lua file that parses the incoming messages.
cpu: checks the load the cpu is currently under
ram: checks the percentage of ram being used
memory: returns the amount of memory currently in use and the percentage of memory free
the image below shows the message being sent to the device through the telegram messenger service

```
import os

# Return CPU temperature as a character string
def getCPUtemperature():
   res = os.popen('vcgencmd measure_temp').readline()
   return(res.replace("temp=","").replace("'C\n",""))

# Return RAM information (unit=kb) in a
list
def getRAMinfo():
   p = os.popen('free')
   i = 0
   while 1:
      i = i + 1
      line = p.readline()
      if i==2:
         return(line.split()[1:4])

# Return % of CPU used by user as a character string
def getCPUuse():
   return(str(os.popen("top -n1 | awk '/Cpu\(s\):/ {print 1.84€}'").readline().strip(\
)))

# Return information about disk space as a list (unit
included)
def getDiskSpace():
   p = os.popen("df -h /")
   i = 0
   while 1:
      i = i +1
      line = p.readline()
```

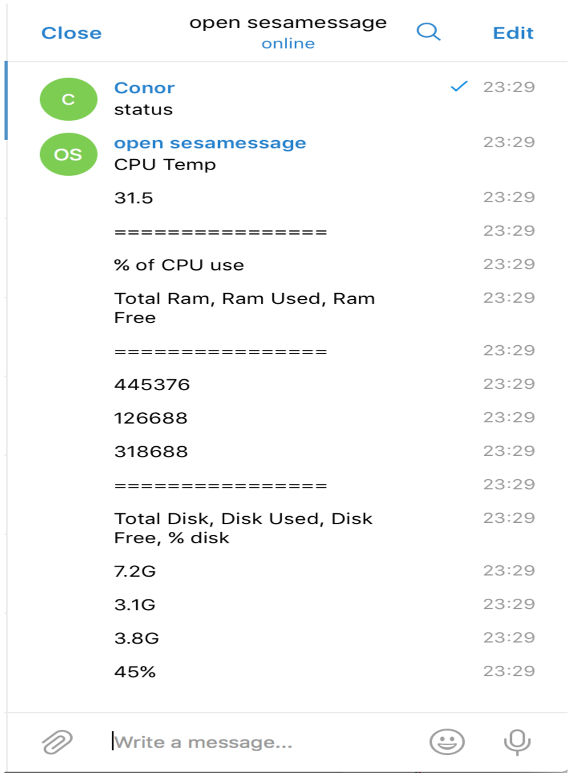
```
        if i==2:
            return(line.split()[1:5])
```

execution: once this code executes the user is responded to with a message with the string containing the system information. This allows the user to remotely monitor the system.

Lastly the python program creates a text file called status_file which contains the information, this file is then interpreted by the Lua code when a request for status is made. Every time the status request is called the file gets overwritten with new information. This avoids a build-up of text files over periods of long use.
Python code handling file creation and update:

```
file = open("status_file","w")

file.write("\n"+"CPU Temp")
file.write("\n"+getCPUtemperature())
file.write("\n"+"================")
file.write("\n"+"% of CPU use")
file.write("\n"+"\n".join(getCPUuse()))
file.write("\n"+"================")
file.write("\n"+"Total Ram, Ram Used, Ram Free")
file.write("\n"+"\n".join(getRAMinfo()))
file.write("\n"+"================")
file.write("\n"+"Total Disk, Disk Used, Disk Free, % disk")
file.write("\n"+"\n".join(getDiskSpace()))

file.close()
```

image of chat screen requesting status update:



Conor 23:29
status

open sesamessage 23:29
CPU Temp

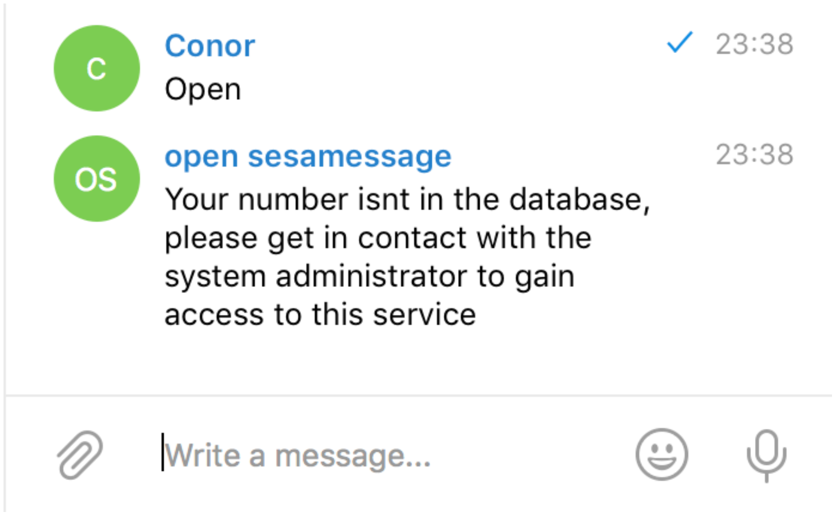
```
31.5                              23:29
================                  23:29
% of CPU use                      23:29
Total Ram, Ram Used, Ram          23:29
Free
================                  23:29
445376                            23:29
126688                            23:29
318688                            23:29
================                  23:29
Total Disk, Disk Used, Disk       23:29
Free, % disk
7.2G                              23:29
3.1G                              23:29
3.8G                              23:29
45%                               23:29
```

Open system

This system is the integral part of this project, this is the system responsible for the opening and closing of the door. there are certain authentication protocols that are required before this system is called into action.

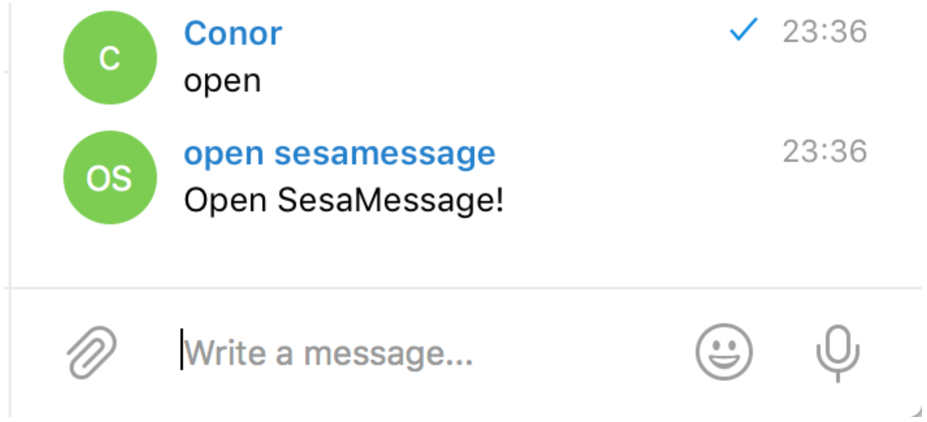
message to system:
The user initiates this action when the user messages the system requesting it to open, this needs to be as simple and as intuitive as possible to cater for all users and their abilities. Because of this I have chosen to designate the message for opening the door as simply ‘open’

response failed: if the user is not already approved by the administrator before they message the device requesting it to open.
The device performs a check once the open request comes through, it will search its contacts list of pre-approved users and numbers searching for the current requested user. If that user is not found in the database the system will respond with a message informing them that they do not have the required permissions to access this device and to contact the device administrator to gain approval.


Conor ✓ 23:38
Open

open sesamessage 23:38
Your number isnt in the database, please get in contact with the system administrator to gain access to this service

This stops unwanted people gaining access to the device and the building.


Conor ✓ 23:36
open

open sesamessage 23:36
Open SesaMessage!

response success: if the user has the required permissions to access the device their request is granted. This informs the device to trigger the python program to open the device.

This program is written in python and handles the movement of the servo motor. As you can see in the code snippet below the servo is working on pin 11 in the raspberry pi’s GPIO. This pin handles the control of this device. The servo is also operating at a frequency of 50hz.

Because the raspberry pi b+ GPIO pins only offer a voltage of 5v an external battery was used to provide enough power to open the bolt on a door’s lock. Because of this the raspberry pi’s pins offer the ground for both the servo and the external power source.

Raspberry pi image

Outlines in detail the process of taking a photo using the Microsoft LifeCam Cinema 720p webcam, and returning the image to the user via the telegram messanging system. This is an important security feature for the device as it allows the user to see who is entering the building remotely

Controls, Maintains and Implements:

- **request:** this message sent to the device through telegram
- **parsed message:** the device interprets the message and preforms a bash command on the system and saves it as a string
- **respond function:** the device stores the image locally, allowing the lua function to search and grab the image from the image location and send it to the user using the media function in the telegram API

Implementation & snippets:

image: is a custom python program I have written contained within the raspberry pi system. Once triggered it returns a image to the user. This image is stored in the device and sent to the user, to avoid taking up too much memory the image rewrites itself every time a picture is taken

```
#!/usr/bin/python
import os
import pygame, sys

from pygame.locals import *
import pygame.camera

width = 640
height = 480

#initialise pygame
pygame.init()
pygame.camera.init()
cam = pygame.camera.Camera("/dev/video0",(width,height))
cam.start()

#setup window
windowSurfaceObj = pygame.display.set_mode((width,height),1,16)
pygame.display.set_caption('Camera')

#take a picture
image = cam.get_image()
cam.stop()

#display the picture
catSurfaceObj = image
windowSurfaceObj.blit(catSurfaceObj,(0,0))
pygame.display.update()

#save picture
pygame.image.save(windowSurfaceObj,'image.jpg')
```

execution: once this code executes the user is responded to the user with a message with containing the image. This allows the user to remotely monitor who is entering the building.

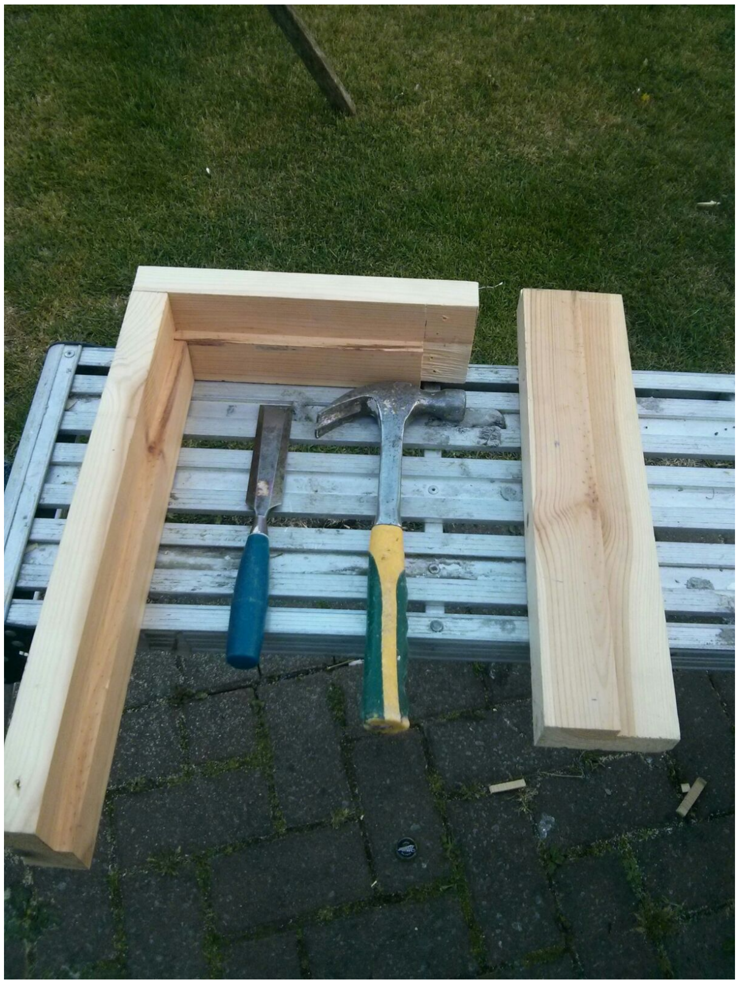
10 Prototype creation

Because of the nature of the project a prototype is required to accurately see how the device will function. This prototype will be implemented with the device operating, and executing all of the functionality.

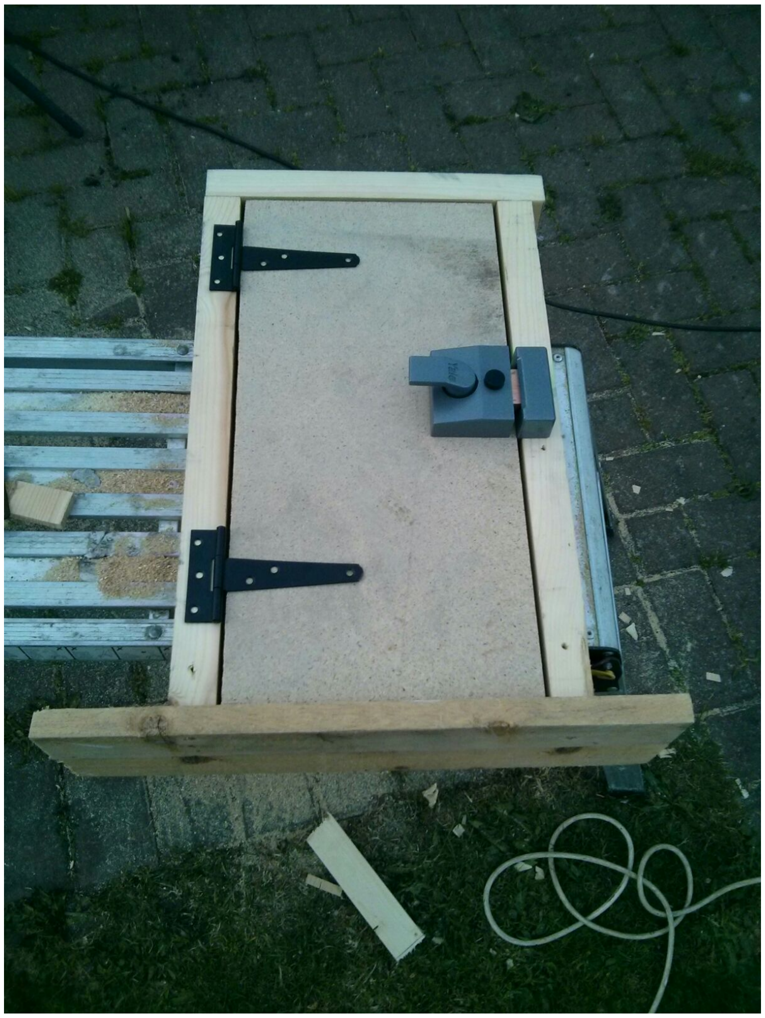
The prototype creation took place in two parts. The first part consisted of creating the door in which the device would operate, the second part of the prototype consisted of establishing the device in its working state onto the door.

Door creation

The creation of the door took place over the course of an afternoon. For this small project I created the door frame using 2x4 wood, and a piece of fire retardant MDF board for the door.



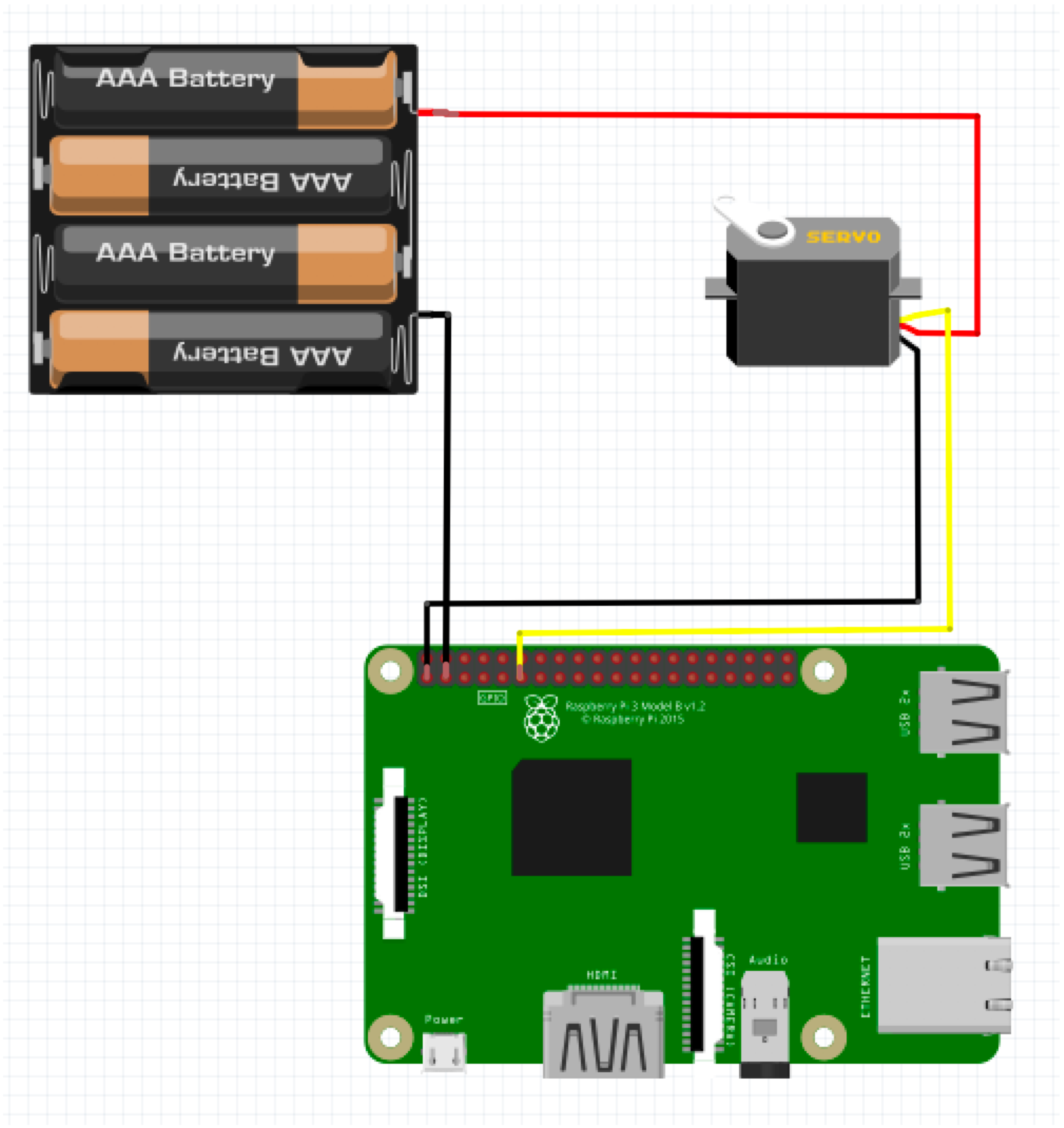
Making sure the door frame was square was ensured by using a square, and cutting the frame pieces to the required length. Once the frame was made, the door needed to be cut to length.



I used small hinges to mount the foor to the frame and the lock, which for this project was the yale p85 night latch. I chose this lock because it is a common door lock used in Ireland, and is consitant with the style of lock used commonly throughout the world. This decision was made to further broaden the application and the use cases for the project, and not to have it only hit a small niche.

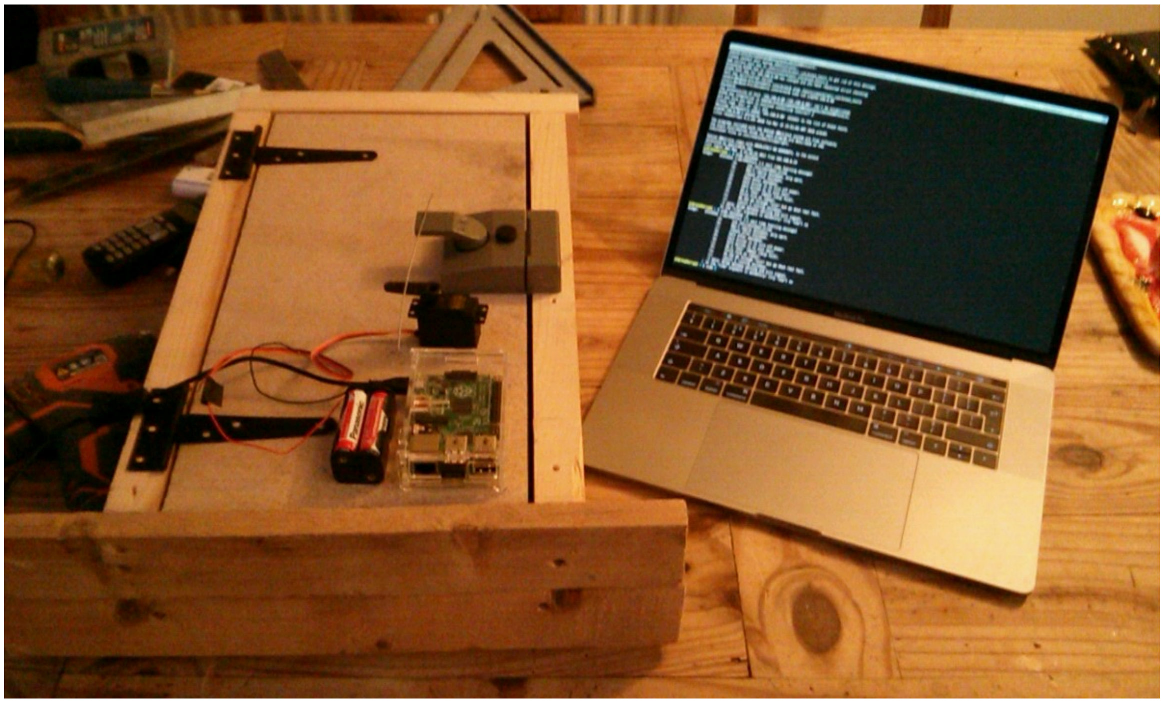
Device setup

The second part of creating the prototype consisted of adding the raspberry pi, the servo and the external power supply to the door. this shows that the device can be added to a standard door and used without a lot of configuration.



above you can see the system diagram for the raspberry pi, the ground wire for both the external power source and the servo are connected to pins 1 and 3 of the raspberry pi's GPIO board, while the controller wire is connected to pin 11. Finally, the power cord shown in red runs from the servo to the external power source.

once the door is opened the program is told to maintain the arc on the servo for a period of 5 seconds. This is the time in which the user will be able to push the door. the program initiated a time.sleep() function, once this is over the device returns to its default state.



with the creation of the prototype door and the device established, I was able to get a better understanding of how it would act in normal use throughout its life cycle.

11 Graphical User Interface (GUI) Layout

This section of the report will cover all the different GUI elements the user will have access to and need to navigate to use OpenSesamessage to its maximum potential. This section will be broken up into two parts, the graphical user interface for the web application and the graphical user interface for the telegram messenger service. This service will also cover the mobile view as this service will have to be accessible to users on mobile also.

Opensesamessage.com

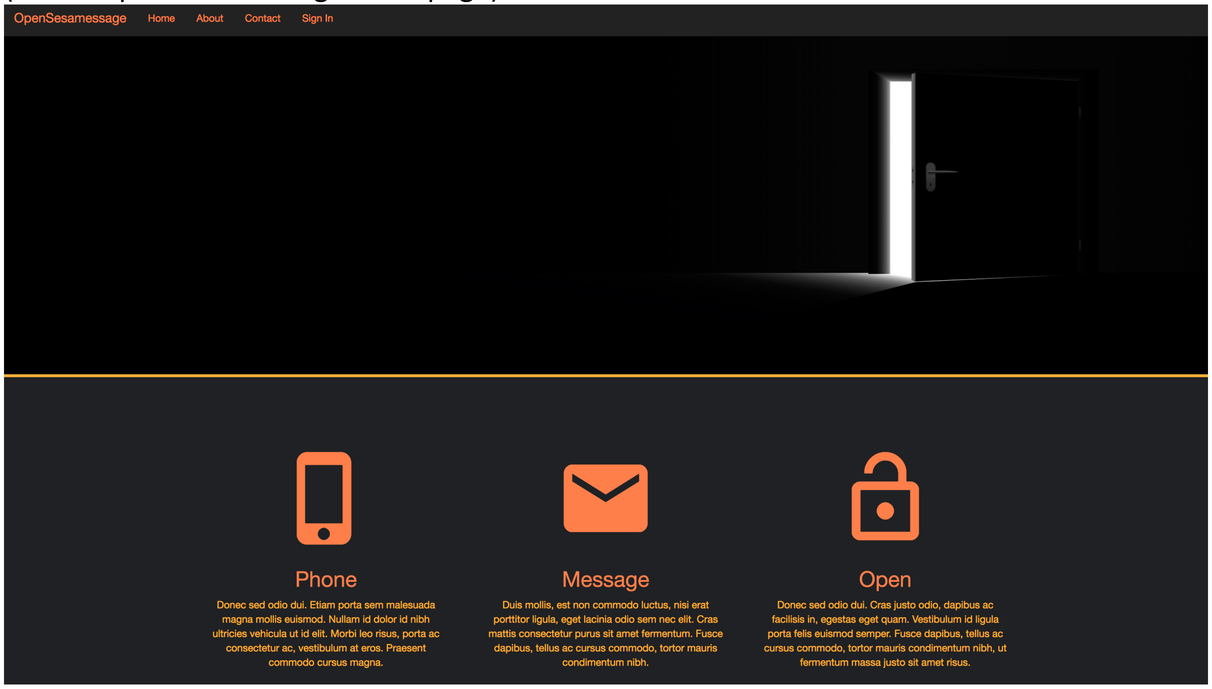
The GUI for the web application is made up of html and css, and the front-end framework bootstrap.
"Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes."

homepage

Description: this is the first interface the user will be met with, it is the homepage for the website with links to the sign in page, and the contact page:

- home, allows the user to return to the homepage from anywhere in the application.
- contact, allows the user to access the contact us page.
- Sign in, allows the user to enter the sign in page.

(GUI 1: open sesamessage homepage)



Phone

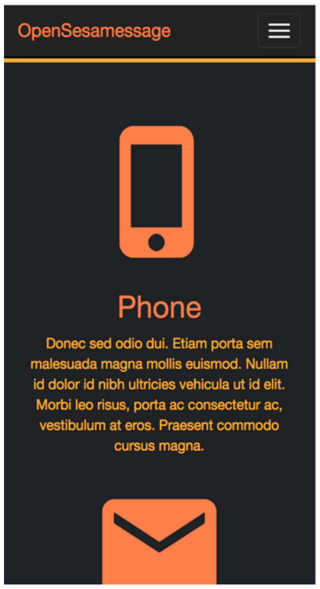
Donec sed odio dui. Etiam porta sem malesuada magna mollis euismod. Nullam id dolor id nibh ultricies vehicula ut id elit. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Praesent commodo cursus magna.

Message

Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh.

Open

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.



Phone

Donec sed odio dui. Etiam porta sem malesuada magna mollis euismod. Nullam id dolor id nibh ultricies vehicula ut id elit. Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Praesent commodo cursus magna.

mobile device

- Responsive navbar for screens under a certain size
- Responsive icons and text for smaller screen sizes

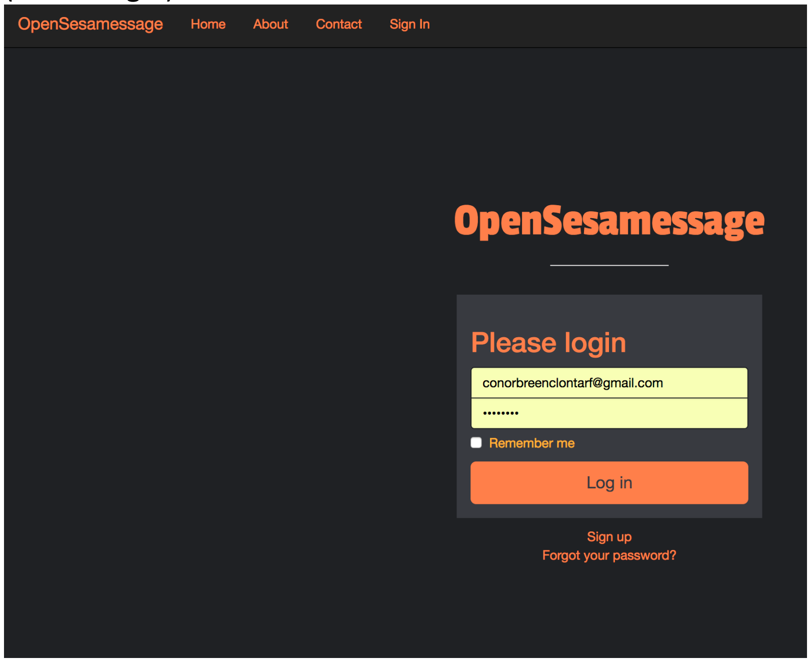
Reason: this is where the user is welcomed to the service, it needs to be clean responsive and clearly show how the product works on a consumer level.

Sign in

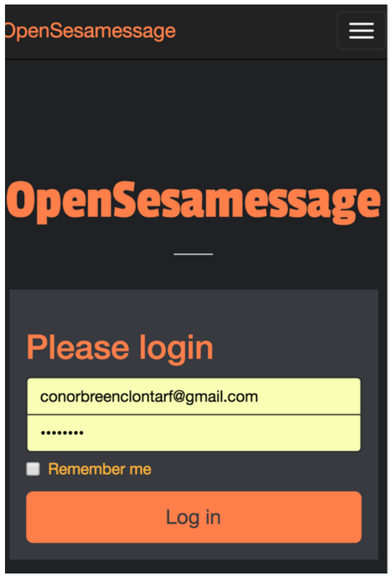
Description: This interface allows the user to modify all in-game settings to improve performance on their machine by increasing or decreasing certain options. It also allows users with higher end machines to increase the visual quality of the game.
This interface allows the user to sign in to their account, this is a simple process that allows users access to their account settings
There are two authentication options:

- ➢ email
- ➢ password

(GUI 2: login)



mobile device:



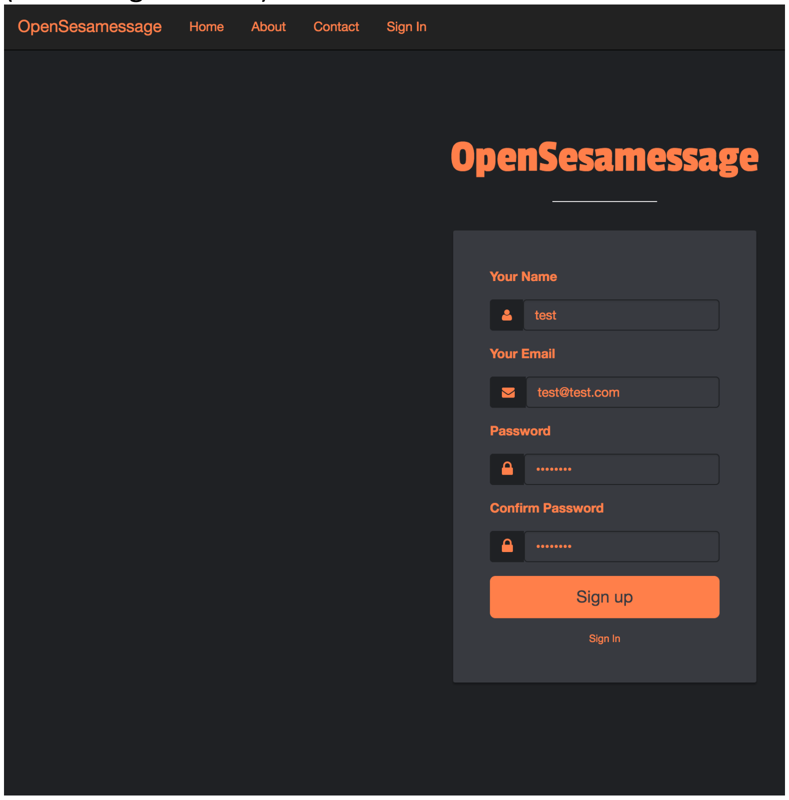
- responsive navbar for smaller screen devices
- resizing containers that allows for easy navigation of the sign up form for mobile users.
- Larger buttons for mobile users to allow for easier navigation to the next menu

Registrations

User registration GUI

Description: this interface allows the user to register for the service. It requires their name, email password. From here they will have access to their account's administrator functionality.

(GUI 3: registration)



Your Name

test

Your Email

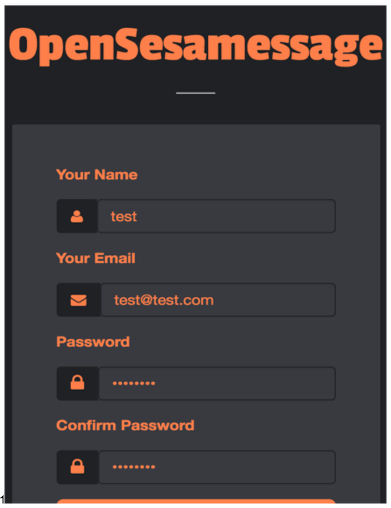
test@test.com

Password

••••••••

Confirm Password

••••••••



Your Name

test

Your Email

test@test.com

Password

••••••••

Confirm Password

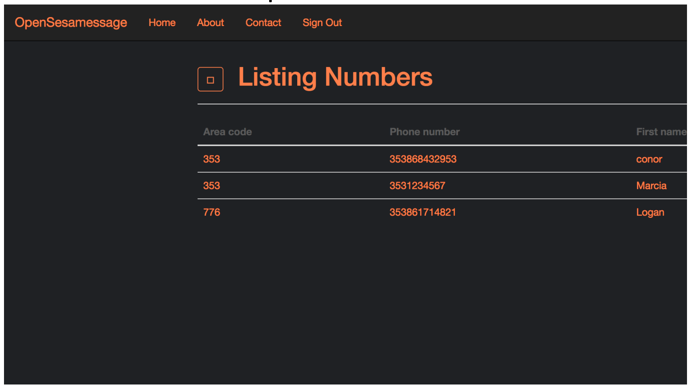
••••••••

mobile device

Description: This interface allows the user to start the chosen chapter from the beginning if they use the start chapter button, if they have play already and saved they can start from there using the continue chapter button.

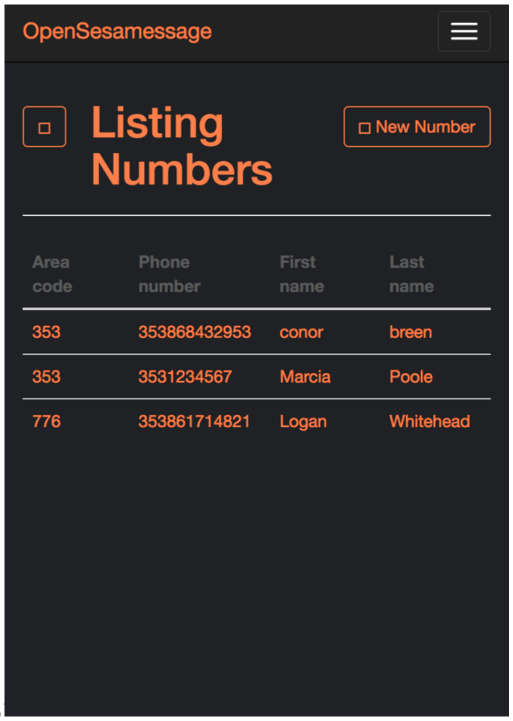
Add users

Description: this interface allows the administrator user to add phone numbers to the whitelist. This grants the required privileges to the telegram account to access the device and make requests


OpenSesamessage Home About Contact Sign Out

Listing Numbers

Area code	Phone number	First name
353	353868432953	conor
353	3531234567	Marcia
776	353861714821	Logan


OpenSesamessage

Listing Numbers

New Number

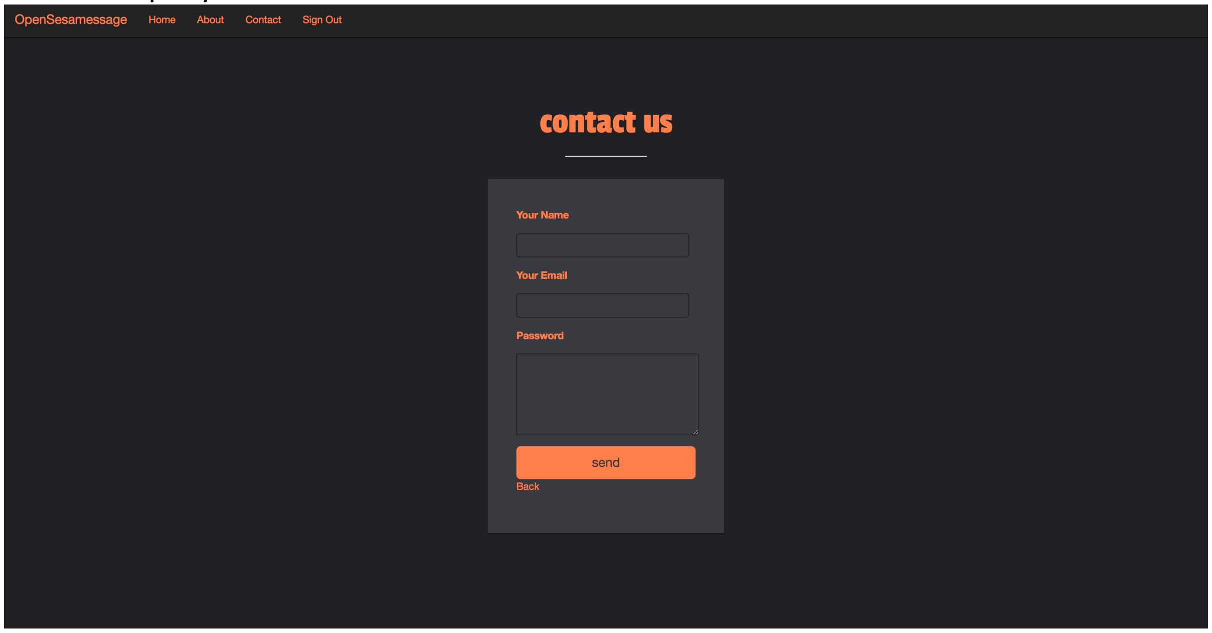
Area code	Phone number	First name	Last name
353	353868432953	conor	breen
353	3531234567	Marcia	Poole
776	353861714821	Logan	Whitehead

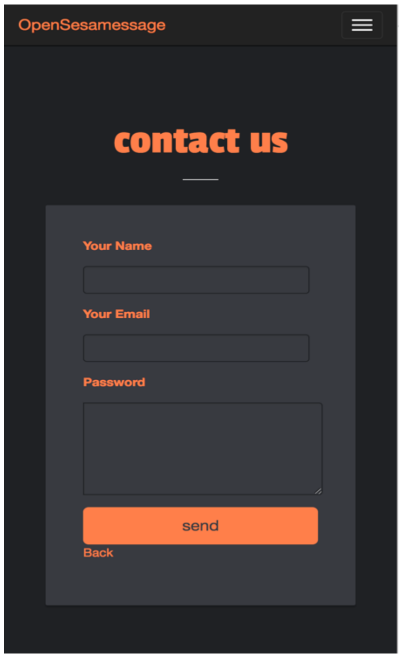
mobile device

- ➢ Responsive navbar
- ➢ Scaled down interface for smaller device.
- ➢ Larger buttons for mobile users to allow for easier navigation to the next menu
- ➢ As intuitive as possible

Contact GUI

Description: this view handles the user contact functionality. this allows users to get in contact with the service. The form consists of three text areas, one for name, email and the user's query.





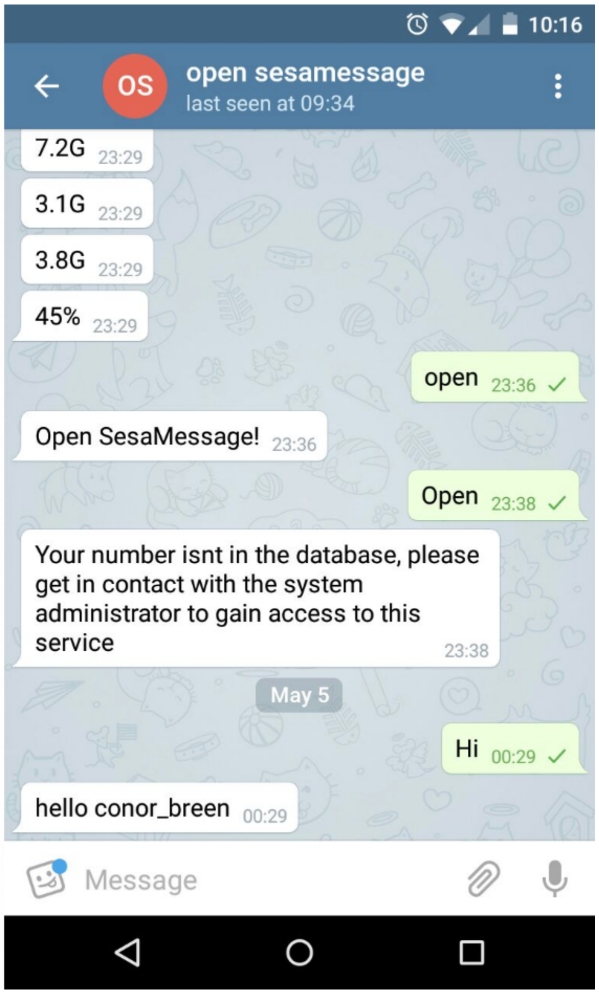
mobile device:

- this interface resizes itself to cater for the difference of screen sized
- This is handled in the form-control-contact css class that sets the display to block and resizes the text areas to 140%.

Telegram GUI

Description: the telegram interface is not one I have created, however I feel it is important to the look and feel of the projects operation. Each of the below graphical user interfaces are ways in which the users can interact with the raspberry pi device

Mobile GUI



10:16

open sesamessage
last seen at 09:34

7.2G 23:29

3.1G 23:29

3.8G 23:29

45% 23:29

open 23:36

Open SesaMessage! 23:36

Open 23:38

Your number isnt in the database, please get in contact with the system administrator to gain access to this service 23:38

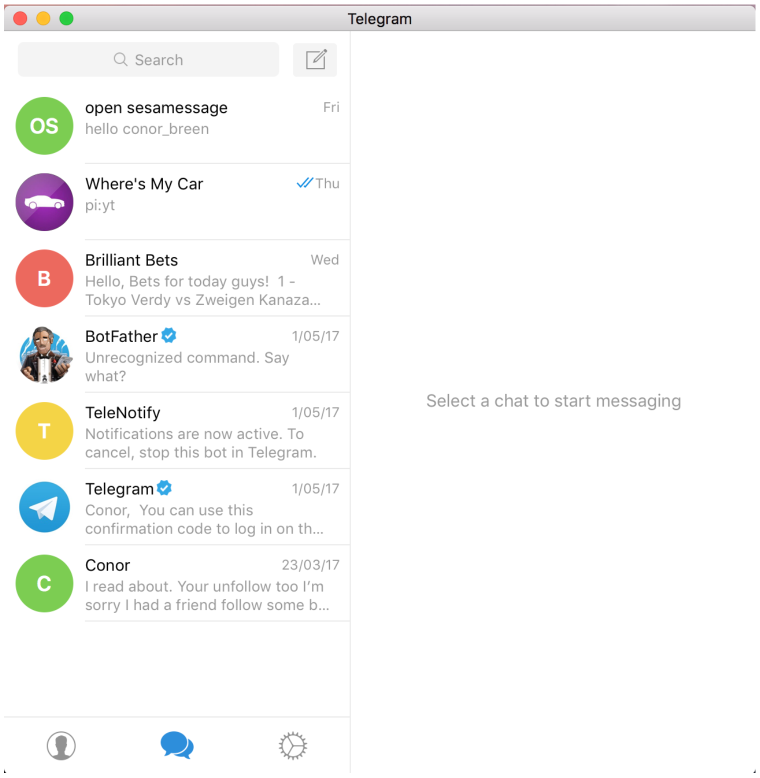
May 5

Hi 00:29

hello conor_breen 00:29

Message

Application GUI



Telegram

Search

open sesamessage — Fri
hello conor_breen

Where's My Car — Thu
pi:yt

Brilliant Bets — Wed
Hello, Bets for today guys! 1 - Tokyo Verdy vs Zweigen Kanaza...

BotFather — 1/05/17
Unrecognized command. Say what?

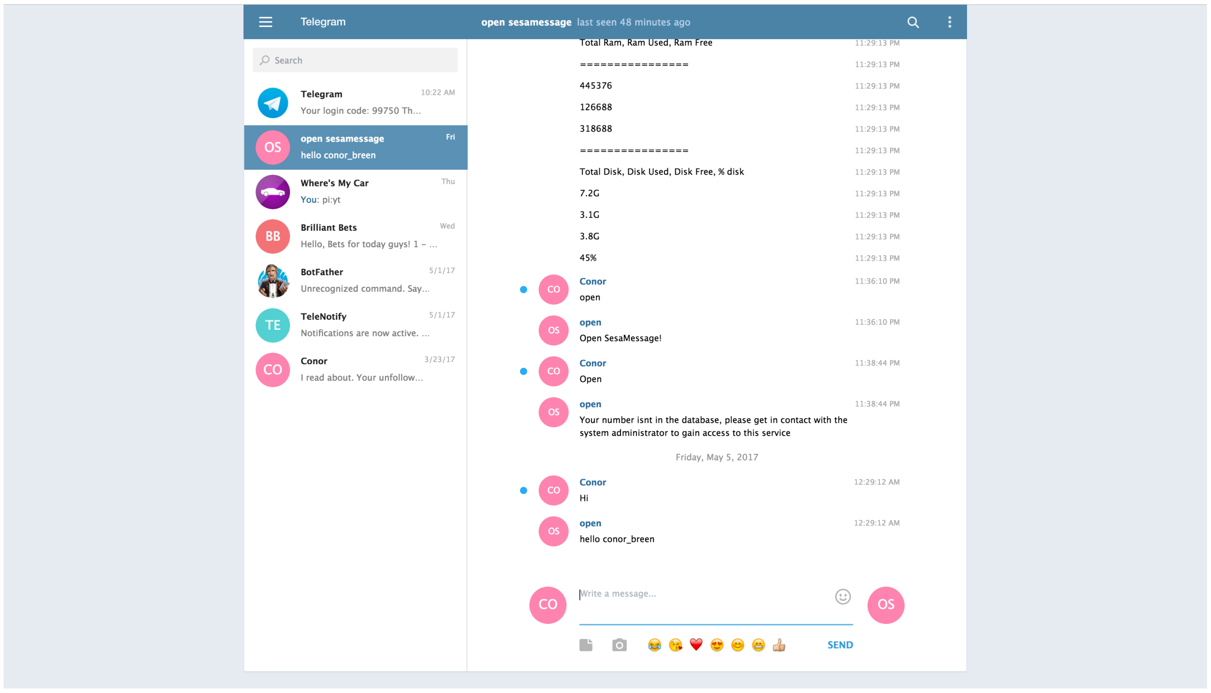
TeleNotify — 1/05/17
Notifications are now active. To cancel, stop this bot in Telegram.

Telegram — 1/05/17
Conor, You can use this confirmation code to log in on th...

Conor — 23/03/17
I read about. Your unfollow too I'm sorry I had a friend follow some b...

Select a chat to start messaging

Web GUI



Telegram

Search

Telegram — 10:22 AM
Your login code: 99750 Th...

open sesamessage — Fri
hello conor_breen

Where's My Car — Thu
You: pi:yt

Brilliant Bets — Wed
Hello, Bets for today guys! 1 – ...

BotFather — 5/1/17
Unrecognized command. Say...

TeleNotify — 5/1/17
Notifications are now active. ...

Conor — 3/23/17
I read about. Your unfollow...

open sesamessage last seen 48 minutes ago

```
Total Ram, Ram Used, Ram Free                 11:29:13 PM
================                              11:29:13 PM
445376                                        11:29:13 PM
126688                                        11:29:13 PM
318688                                        11:29:13 PM
================                              11:29:13 PM
Total Disk, Disk Used, Disk Free, % disk      11:29:13 PM
7.2G                                          11:29:13 PM
3.1G                                          11:29:13 PM
3.8G                                          11:29:13 PM
45%                                           11:29:13 PM
```

Conor — 11:36:10 PM
open

open — 11:36:10 PM
Open SesaMessage!

Conor — 11:38:44 PM
Open

open — 11:38:44 PM
Your number isnt in the database, please get in contact with the system administrator to gain access to this service

Friday, May 5, 2017

Conor — 12:29:12 AM
Hi

open — 12:29:12 AM
hello conor_breen

12 Testing

Overview

Testing is essential when it comes to consumer devices, and when it comes to security devices testing is paramount. It is no different for OpenSesamessage. Three main forms of testing were used for this project. this testing ensured the device was simple to use for all audiences regardless of age or technical ability.
Tools used in the testing process of OpenSesamessage:

- **Travis.io:** this testing was carried out by Travis. Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub.
- **Usability testing:** this testing was carried out by adding users to the system and having them run through the processes of operating the system
- **typeform:** I conducted online surveys to see what average consumers wanted and if those needs could be met within the project.

Methods of testing used in OpenSesamessage:

- online survey
- Unit & Integration Testing
- Usability testing

Each method of testing is explained under its own heading, tests where participants where used are detailed with the number of active participants and any collected responses from surveys untaken.

Unit Testing

Initial testing for this project began with creating a separate database for tests to be performed. In ruby on rails a test database gets populated while running through controller tests, and is dropped after the tests are done. To create the test database, I performed the following commands.

```
conorbreen$ bundle exec rake db:drop RAILS_ENV=test
```

```
conors-MacBook-Pro:opensesamessage-2.0 conorbreen$ bundle exec rake
db:create RAILS_ENV=test
conors-MacBook-Pro:opensesamessage-2.0 conorbreen$ bundle exec rake
db:schema:load RAILS_ENV=test
```

this generated the necessary test database. With the creation of the test database I could begin writing tests. The test suite I used was rSpec. rSpec is a testing framework for ruby on rails, it can be installed by adding the rSpec-rails gem to the gem file within the rails application. The installation process for rSpec is similar to most gems. First you have to add it to the gemfile in the appropriate group, in this case the development test group:

```
group :development, :test do
  # Call 'byebug' anywhere in the code to stop execution and get a
debugger console
  gem 'byebug'
  gem 'pry-rails'
  gem 'letter_opener'
  gem 'rspec-rails', '~> 3.5'
end
```

and running both **bundle** to download the gem, and then running the command **rails generate rspec:install.** This process can be seen in the terminal snippet below:

```
conors-MacBook-Pro:opensesamessage-2.0 conorbreen$ rails generate
rspec:install
Running via Spring preloader in process 67578
Expected string default value for '--jbuilder'; got true (boolean)
      create  .rspec
      create  spec
      create  spec/spec_helper.rb
      create  spec/rails_helper.rb
conors-MacBook-Pro:opensesamessage-2.0 conorbreen$ rake db:migrate
&& rake db:test:prepare
conors-MacBook-Pro:opensesamessage-2.0 conorbreen$ rake spec
/Users/conorbreen/.rbenv/versions/2.3.1/bin/ruby -
I/Users/conorbreen/.rbenv/versions/2.3.1/lib/ruby/gems/2.3.0/gems/rs
pec-core-
3.6.0/lib:/Users/conorbreen/.rbenv/versions/2.3.1/lib/ruby/gems/2.3.
0/gems/rspec-support-3.6.0/lib
/Users/conorbreen/.rbenv/versions/2.3.1/lib/ruby/gems/2.3.0/gems/rsp
ec-core-3.6.0/exe/rspec --pattern spec/\*\*\{,/\*/\*\*\}/\*_spec.rb
No examples found.


Finished in 0.00043 seconds (files took 0.09987 seconds to load)
0 examples, 0 failures
```

Once rSpec created the necessary test files within the application, I began to make the tests for each controller. These tests run through the CRUD functionality. for the testing of the application I needed to test each controller and make sure that each was functioning as it should. The controllers I tested were the Numbers controller, the user controller and the contacts controller.

Numbers controller:
The numbers controller is responsible for handling the users phone numbers. Each person they want provide the necessary privileges required for opening the door is added in the numbers controller. Therefore, this controller is critical to the applications operation. And therefore, the testing of this model is paramount.

Numbers controller file

```
require 'test_helper'

class NumbersControllerTest < ActionController::TestCase
  setup do
    @number = numbers(:one)
  end

  test "should get index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:numbers)
  end

  test "should get new" do
    get :new
    assert_response :success
  end

  test "should create number" do
    assert_difference('Number.count') do
      post :create, number: { area_code: @number.area_code,
first_name: @number.first_name, last_name: @number.last_name,
phone_number: @number.phone_number, user_id: @number.user_id }
    end

    assert_redirected_to number_path(assigns(:number))
  end

  test "should show number" do
    get :show, id: @number
    assert_response :success
  end
  test "should get edit" do
    get :edit, id: @number
    assert_response :success
  end

  test "should update number" do
    patch :update, id: @number, number: { area_code:
@number.area_code, first_name: @number.first_name, last_name:
@number.last_name, phone_number: @number.phone_number, user_id:
@number.user_id }
    assert_redirected_to number_path(assigns(:number))
  end

  test "should destroy number" do
    assert_difference('Number.count', -1) do
      delete :destroy, id: @number
    end

    assert_redirected_to numbers_path
  end
end
```

as you can see above the tests for this controller handle the CRUD functionality, some test smaller roles such as the "should get index" method which tests the index_path for this model. others such as the 'should create number' method tried to create a new number entry into the test database with the parameters I have given it. These parameters are the ones that are required for this controller, parameters such as area_code, first_name, last_name, phone_number, and user_id. Once it runs through these successfully is checks to make sure the user is correctly redirected to the number_path of the appropriate number.

Contacts controller:
The contacts controller handles the queries sent by users to the systems administrator. This is handled by creating a contact entry into the PostgreSQL database containing the parameters email, name and query. Once created the on_create method calls the user Mailer within the application that sends an email to the application's administrator account hello@opensesamessage.com with the contact.query, the contact.name and the contact.email parameters. It also takes the contact.email and sends and email to the person submitting the query notifying them of the receipt of their query and that the support team will be in contact with them shortly.

Contact controller test file:
```
require 'test_helper'
require 'test_helper'

class ContactsControllerTest < ActionController::TestCase
  setup do
    @contact = contacts(:one)
  end

  test "should get index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:contacts)
  end

  test "should get new" do
    get :new
    assert_response :success
  end

  test "should create contact" do
    assert_difference('Contact.count') do
      post :create, contact: { email: @contact.email, name:
@contact.name, query: @contact.query }
    end

    assert_redirected_to contact_path(assigns(:contact))
  end

end
```

the contacts controller test file preforms the standard tests such as should get index test that requires the correct redirect. It also tests for successful creation of contacts in the should get new method. This method tries to create a new entry into the contacts table and awaits the 200-success response.

Travis – CI

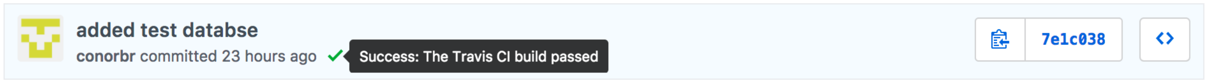
Travis-CI is one of several build automation tools that allows developers understand if their application is working. This is done by building, testing and finally reporting on the build sequences the developer has configured. The 'continuous' part reflects that these tools will scan and detect, in my case by each time a commit is pushed to Github, errors in the application. The integration part reflects the fast software development requires the interaction between multiple separate components working together smoothly. In most cases, there will be a tool to retrieve the latest release or development version, it will then compile the code, perform the necessary tests and deploy the code to the server.

setup
setting up Travis Is quite simple, I logged in via the website travis-ci.com and linked with my Github account. Then I created a configuration file called .travis.yml file to the project containing the necessary configuration parameters for my project. this file Is what Travis will reference when building this version and performing tests

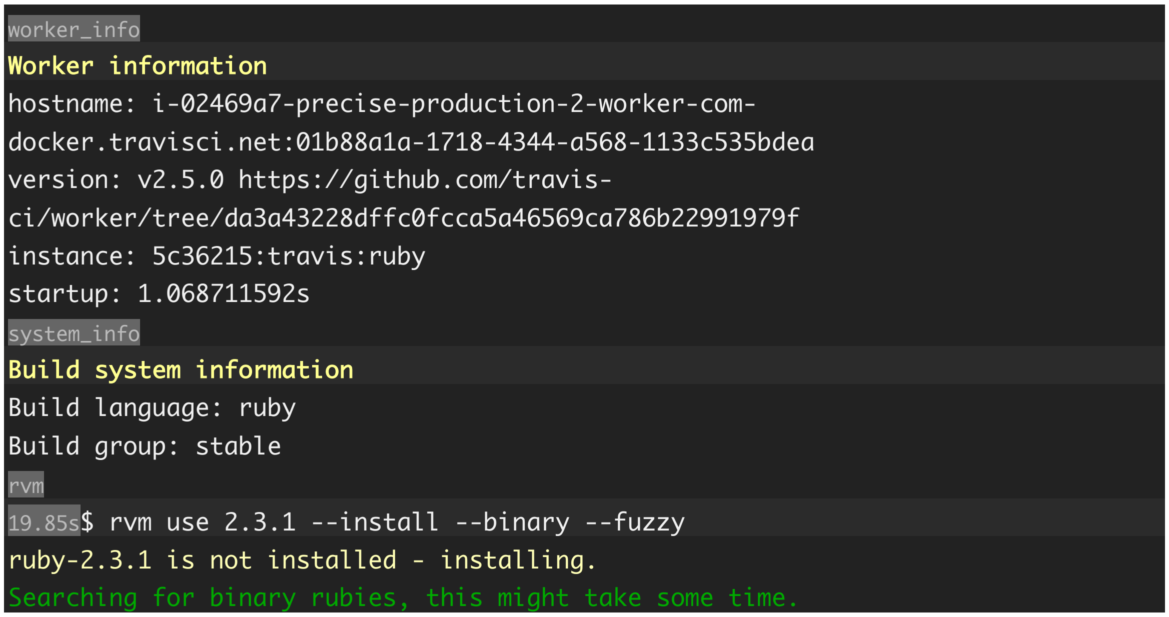
```
language: ruby
rvm:
  - 2.3.1
  - jruby-18mode
  - jruby-19mode
  - jruby-head
```

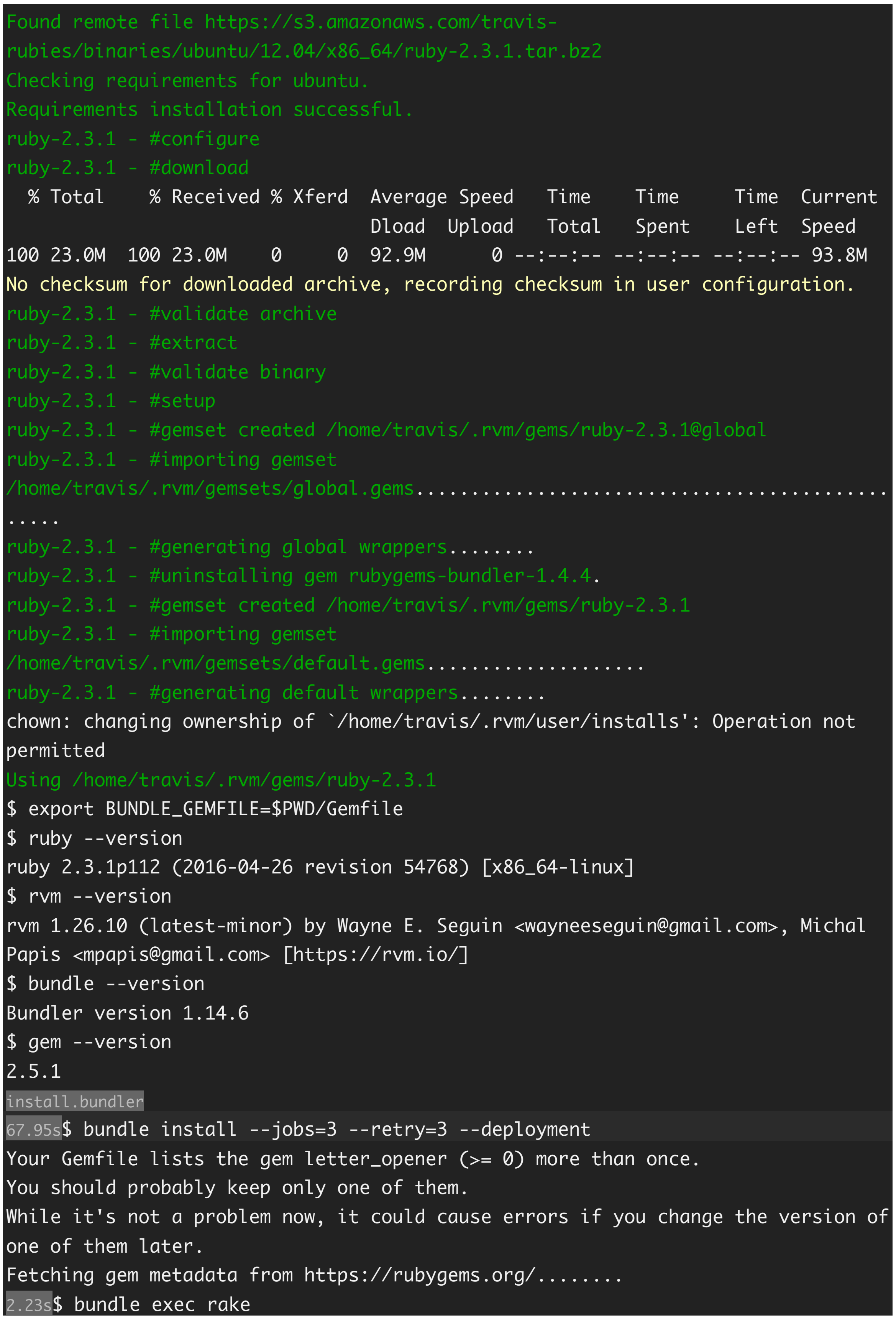
this file declares the ruby version as well as some additional info for the Travis build. Once this file is pushed to the remote. Whenever a commit is sent to the remote repository in Github Travis will run through the test suite, and if these tests pass it will notify the developer.

like the numbers controller this test

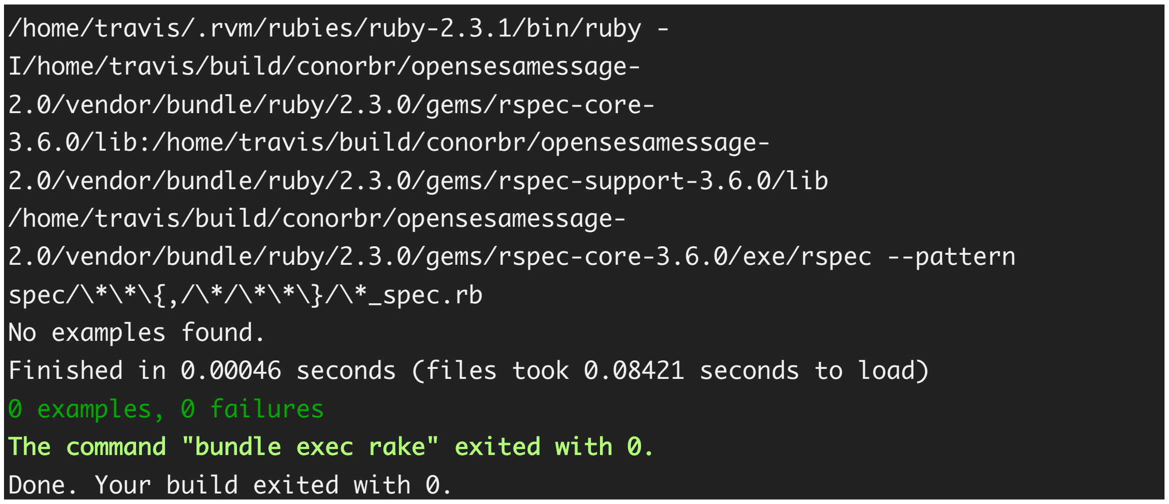

added test databse
conorbr committed 23 hours ago ✓ Success: The Travis CI build passed
7e1c038

Travis-CI also gives a log of the build process, this allows you to run down through the testing process and find where the sequence is being interrupted. This is extremely useful for rapid development that has you adding functionality quickly. It gives peace of mind.


```
worker_info
Worker information
hostname: i-02469a7-precise-production-2-worker-com-
docker.travisci.net:01b88a1a-1718-4344-a568-1133c535bdea
version: v2.5.0 https://github.com/travis-
ci/worker/tree/da3a43228dffc0fcca5a46569ca786b22991979f
instance: 5c36215:travis:ruby
startup: 1.068711592s
system_info
Build system information
Build language: ruby
Build group: stable
rvm
19.85s$ rvm use 2.3.1 --install --binary --fuzzy
ruby-2.3.1 is not installed - installing.
Searching for binary rubies, this might take some time.
```



```
Found remote file https://s3.amazonaws.com/travis-
rubies/binaries/ubuntu/12.04/x86_64/ruby-2.3.1.tar.bz2
Checking requirements for ubuntu.
Requirements installation successful.
ruby-2.3.1 - #configure
ruby-2.3.1 - #download
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 23.0M  100 23.0M    0     0  92.9M      0 --:--:-- --:--:-- --:--:-- 93.8M
No checksum for downloaded archive, recording checksum in user configuration.
ruby-2.3.1 - #validate archive
ruby-2.3.1 - #extract
ruby-2.3.1 - #validate binary
ruby-2.3.1 - #setup
ruby-2.3.1 - #gemset created /home/travis/.rvm/gems/ruby-2.3.1@global
ruby-2.3.1 - #importing gemset
/home/travis/.rvm/gemsets/global.gems..........................................
.....
ruby-2.3.1 - #generating global wrappers........
ruby-2.3.1 - #uninstalling gem rubygems-bundler-1.4.4.
ruby-2.3.1 - #gemset created /home/travis/.rvm/gems/ruby-2.3.1
ruby-2.3.1 - #importing gemset
/home/travis/.rvm/gemsets/default.gems....................
ruby-2.3.1 - #generating default wrappers........
chown: changing ownership of `/home/travis/.rvm/user/installs': Operation not
permitted
Using /home/travis/.rvm/gems/ruby-2.3.1
$ export BUNDLE_GEMFILE=$PWD/Gemfile
$ ruby --version
ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-linux]
$ rvm --version
rvm 1.26.10 (latest-minor) by Wayne E. Seguin <wayneeseguin@gmail.com>, Michal
Papis <mpapis@gmail.com> [https://rvm.io/]
$ bundle --version
Bundler version 1.14.6
$ gem --version
2.5.1
install.bundler
67.95s$ bundle install --jobs=3 --retry=3 --deployment
Your Gemfile lists the gem letter_opener (>= 0) more than once.
You should probably keep only one of them.
While it's not a problem now, it could cause errors if you change the version of
one of them later.
Fetching gem metadata from https://rubygems.org/........
2.23s$ bundle exec rake
```



```
/home/travis/.rvm/rubies/ruby-2.3.1/bin/ruby -
I/home/travis/build/conorbr/opensesamessage-
2.0/vendor/bundle/ruby/2.3.0/gems/rspec-core-
3.6.0/lib:/home/travis/build/conorbr/opensesamessage-
2.0/vendor/bundle/ruby/2.3.0/gems/rspec-support-3.6.0/lib
/home/travis/build/conorbr/opensesamessage-
2.0/vendor/bundle/ruby/2.3.0/gems/rspec-core-3.6.0/exe/rspec --pattern
spec/\*\*\{,/\*/\*\*\}/\*_spec.rb
No examples found.
Finished in 0.00046 seconds (files took 0.08421 seconds to load)
0 examples, 0 failures
The command "bundle exec rake" exited with 0.
Done. Your build exited with 0.
```

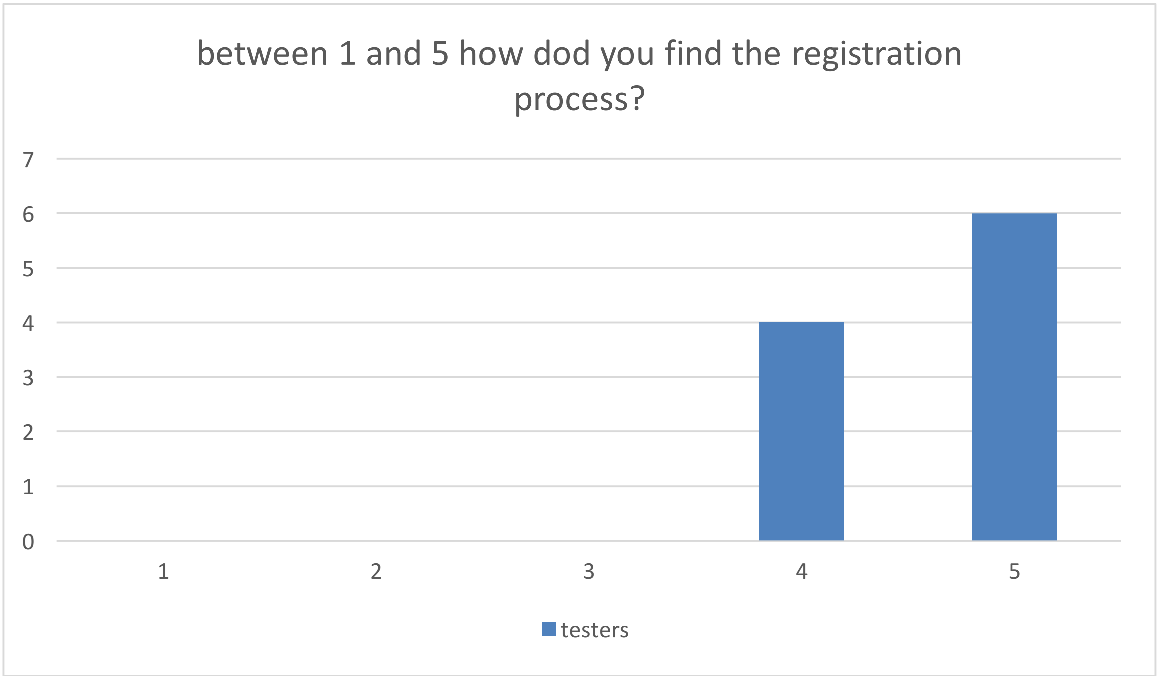
Customer testing

Usability testing was conducted for OpenSesamessage to see how quickly users could get setup with the service and operate its functionality. The aim for this project in terms of usability was to have users capable of gaining a comprehensive understanding of the system within five minutes
Number of participants: **Ten** *(age range 18 to 58)*
Tools used in Usability study:

- **Online Survey:** Created a survey based questionnaire, with questions related to improvements, problems and recommendations on the interfaces and the in-game controls
- **Consumer testing**: Beta version date was 05/04/2017

Usability Study:
All questions are rating based, Ratings below or at 2 are deemed problem areas, users that answer with this rating are asked and required to provide feedback as to what the problem or concern is
all questions were based on what tasks the user would be preforming within the life cycle of the product, tasks such as status check, opening and closing of the device and adding new users to the system as well as getting pictures.

First Question: on a scale of 1 to 5, how easily did you find creating an account on the website?

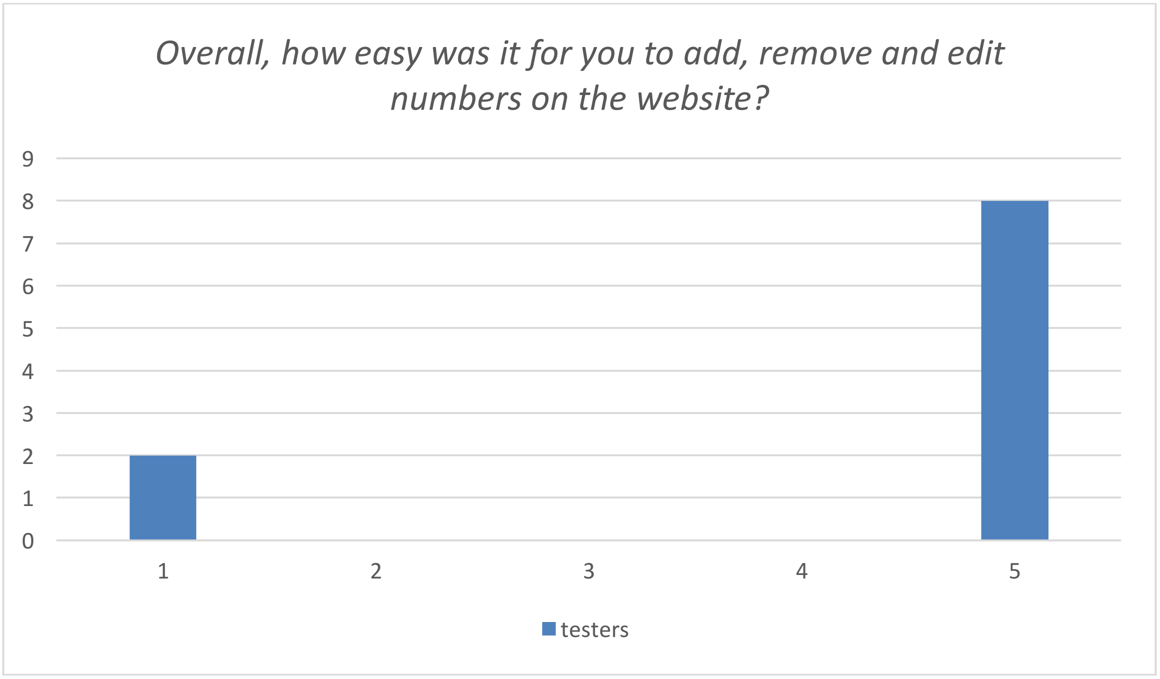


Collected User Reponses from those that answered with a rating of 3 or lower: *(Copied directly from survey response)*

- "I had trouble hitting the sign-up button on my phone"
    - device was iPhone 4s
- "Buttons are to small"
    - Lenovo tablet

Analysis & Solution
From the collected information, it was clear that smaller screen devices were having trouble navigating signup form. I altered the css of the web application to allow for larger buttons when the screen size dropped below a certain size. This seems to have remedied the issues users were having

Second Question: Overall, how easy was it for you to add, remove and edit numbers on the website?

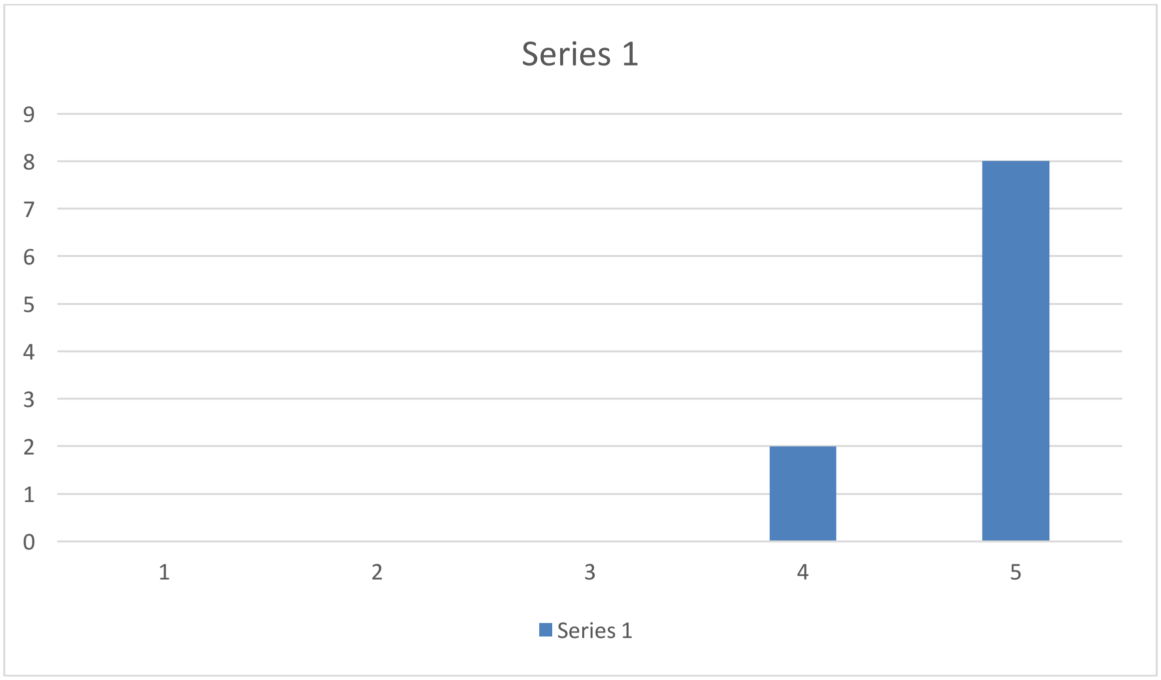


Collected User Reponses from those that answered with a rating of 3 or lower:

- "I got an error message when adding my phone number"
- "I couldn't save my phone number"

Analysis & Solution
from the collected information, the error was rather vague, however looking back over the logs I discovered that when users entered a phone number with non-numerical characters such as '+' or '-' it would cause errors with the saving of the numbers. To address this I added some parsing code to get the number and alter it into such a way that I would be accepted by both the rails web application and the Raspberry pi device this question more than any thought me of the importance of testing, because I created this application, I never considered that users would enter numbers with non-numerical characters.

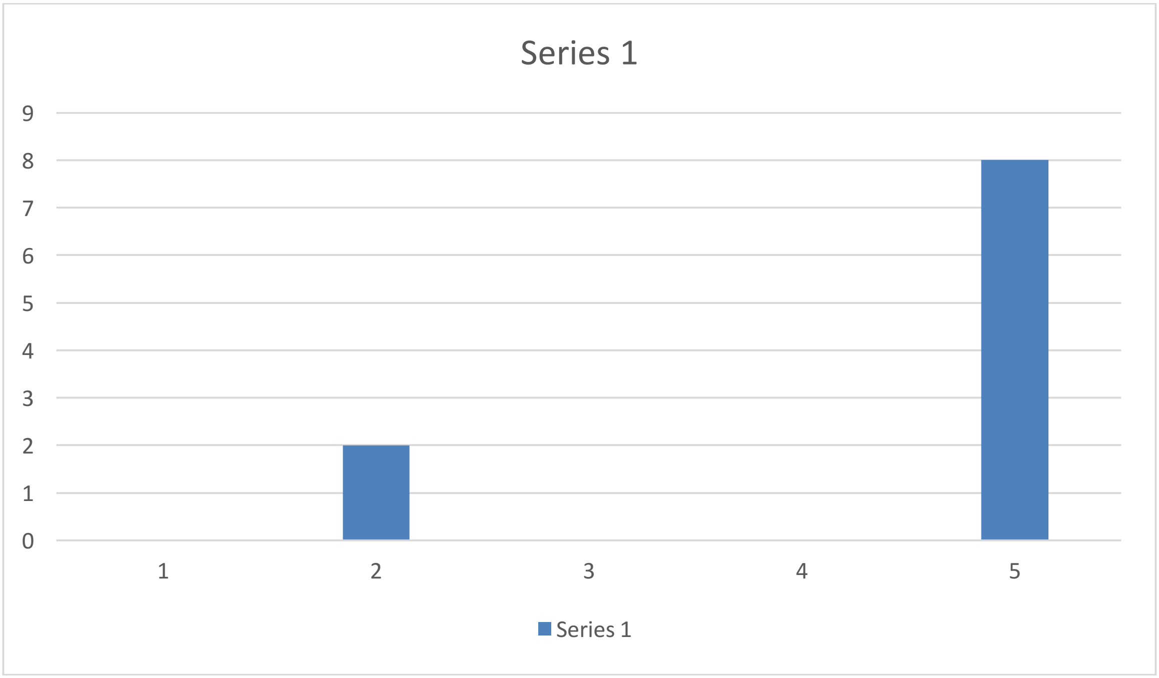
Third Question: overall how did you find the open function for this project?



Collected User Reponses from those that answered with a rating of 3 or lower:
None, however one participant experience trouble registering for telegram. The two-step verification failed, and they had to request another text message

Remarks
No solution needed, I'm happy with the result because I had usability in mind when I was deciding to use this functionality

Fourth Question: how likely would you be to implement a system like this in your home?



Collected User Reponses from those that answered with a rating of 3 or lower: *(Copied directly from survey response)*

- *"I'm not sure I would trust hackers"*
- *"I keep a key in the porch"*

Analysis & Solution

This survey question gave a better understanding of the potential consumers for this product. The users that seemed interested in the product and showed interest in implementing it in their home were of a younger age group. The people who scored lower were older and were unsure of technology and relying on encryption in their home. They worried about potential hackers. This is a view I was not able to change.

13 Market research

Online survey

This survey was carried out during the project, its aim was to get a better understanding of the market and what consumers would be interesting in with a product of this type. The questionnaire was 10 questions long and was designed to gauge the users gender age and their openness to a pro

The online survey: was carried out using the survey tool typeform.com. this allowed me to get the survey out to as many people as possible. The survey consisted of 10 simple questions, aimed to get a better understanding of the product and its features. The survey was carried out by a total of 178 people. To get a good spread of user data I asked friends, family and posted to online forums.

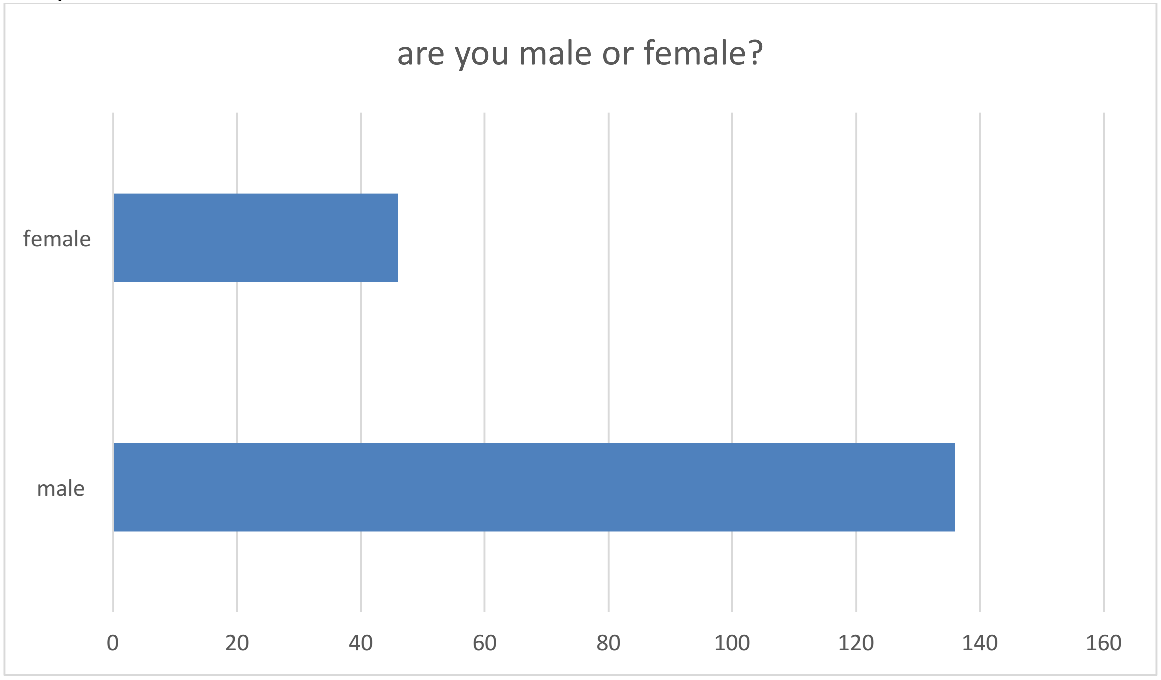
Reason for online survey: because of the nature of the project getting a good understanding of what people want, and the concerns they have over implementing a technology such as this is paramount. The best way to gauge what the public want and how they feel entrusting their home and belongings to a smart device is to ask, so I created this survey.

Questions and results

This section will go through each question, the amount of people that answered the question, the results and the conclusion.

Question 1:

are you male or female?



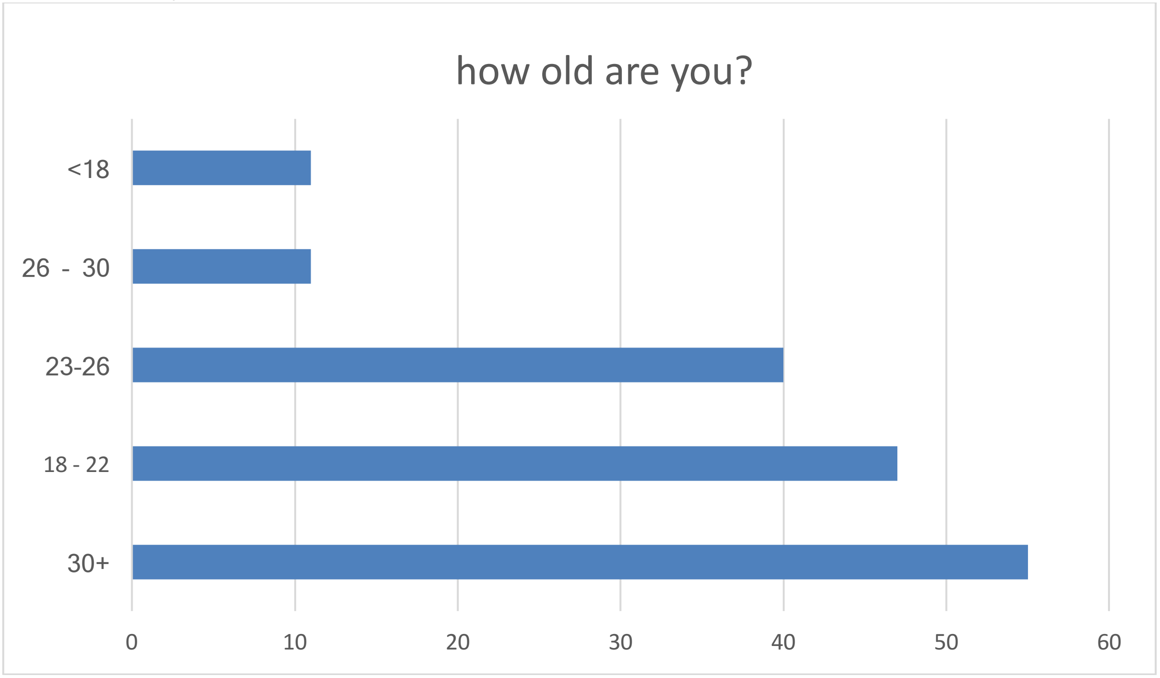
number of people answered:
176 out of 178

results: this question is intended to get a better understanding of the demographics of potential users. The results show that the people who were interested in taking the survey were predominantly male

conclusion: the data shows people who were answering the questions were mainly mail. This needs to be considered when reading the data. However, this does mean I can differentiate different answers based on male and female later in the data analysis.

Question 2:

How old are you?



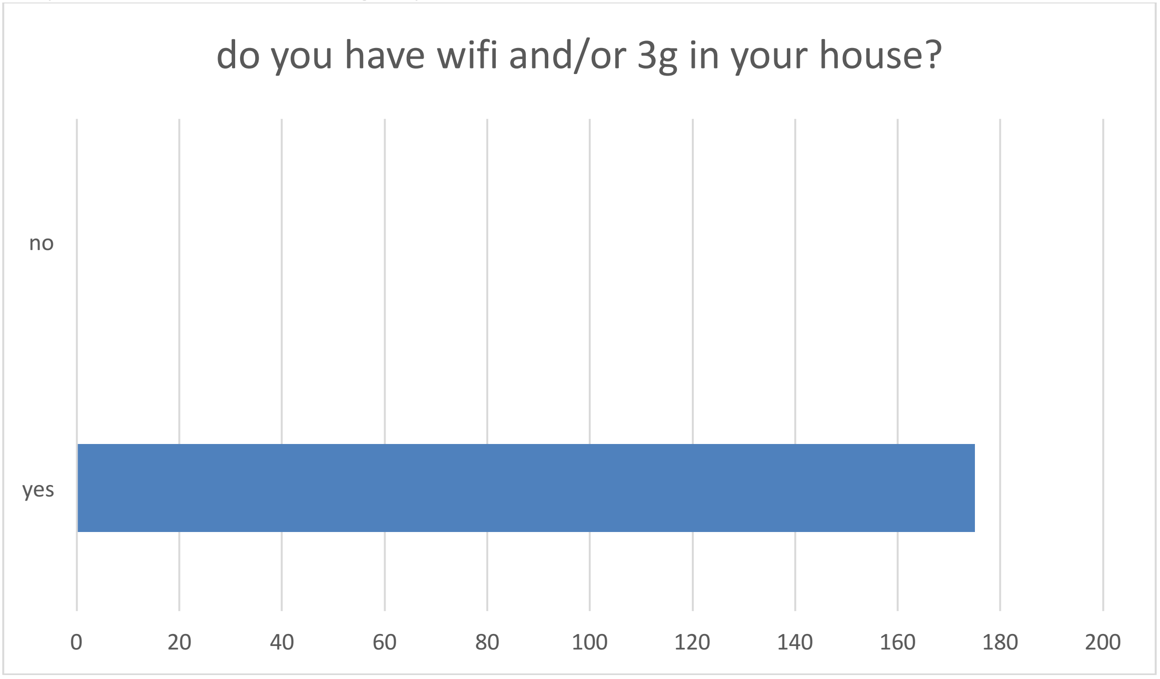
number of people answered:
177 out of 178

results: the results of this question show that the users who took this survey were generally within the age group of over 30 years old.

conclusion: the data shows that the people interested in taking this survey and thus the product were people over the age of 30. People of this age generally own their own homes, so would be the target market. This is the intended audience for the survey and the results therefore are a good indication of what the target market is interested in.

Question 3:

do you have Wi-Fi and/or 3g in your house?



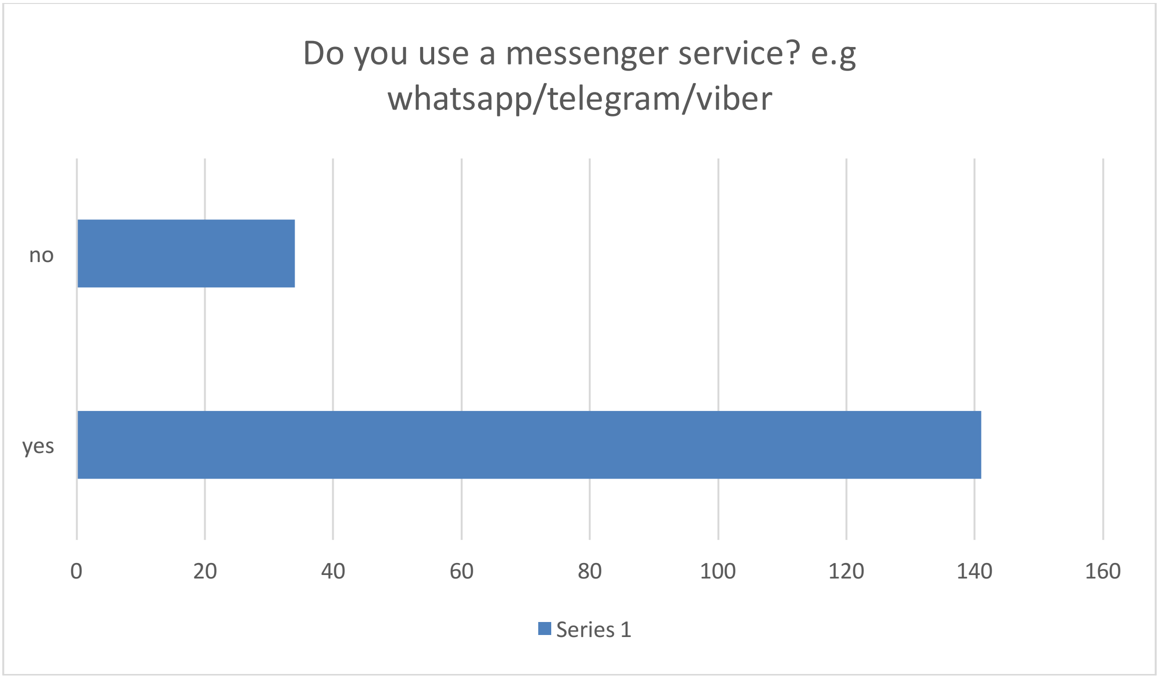
number of people answered:
175 out of 178

results: this question was both surprising and unsurprising. It revealed that everyone who answered this question had some form of internet connectivity

conclusion: the device requires internet connectivity to operate as the messages are sent over the internet. Having internet access in the home means that users can have the raspberry pi connected to the internet, while being within range of Wi-Fi/3g to be able to message the device

Question 4:

Do you use a messenger service? e.g whatsapp/telegram/viber



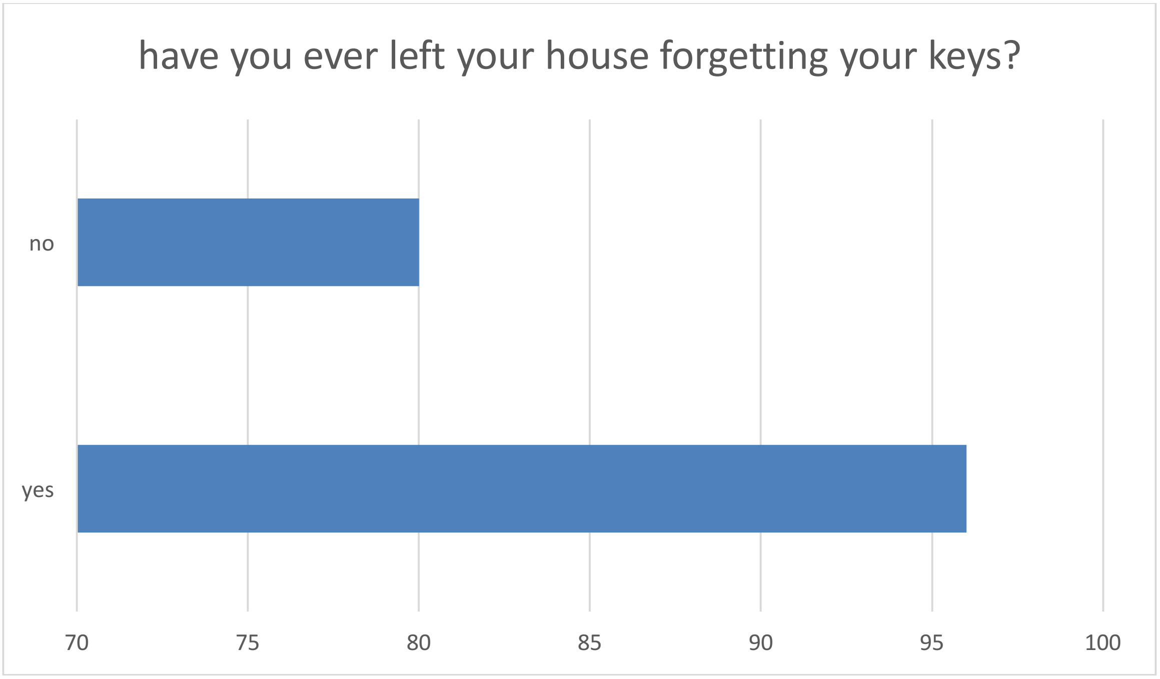
number of people answered:
175 out of 178

results: the results of this question shows that the target market for this product have some experience using a popular messenger service.

conclusion: this is an important result as it shows that most users have experience using a messenger service. The importance of this result comes from the importance of people ease of use of the service. The device is operated by sending messages to the device using the messenger service telegram. If users have experience with a messenger service then the learning requirement for the devices' operation is severely reduced

Question 5:

have you ever left your house forgetting your keys?



number of people answered:
176 out of 178

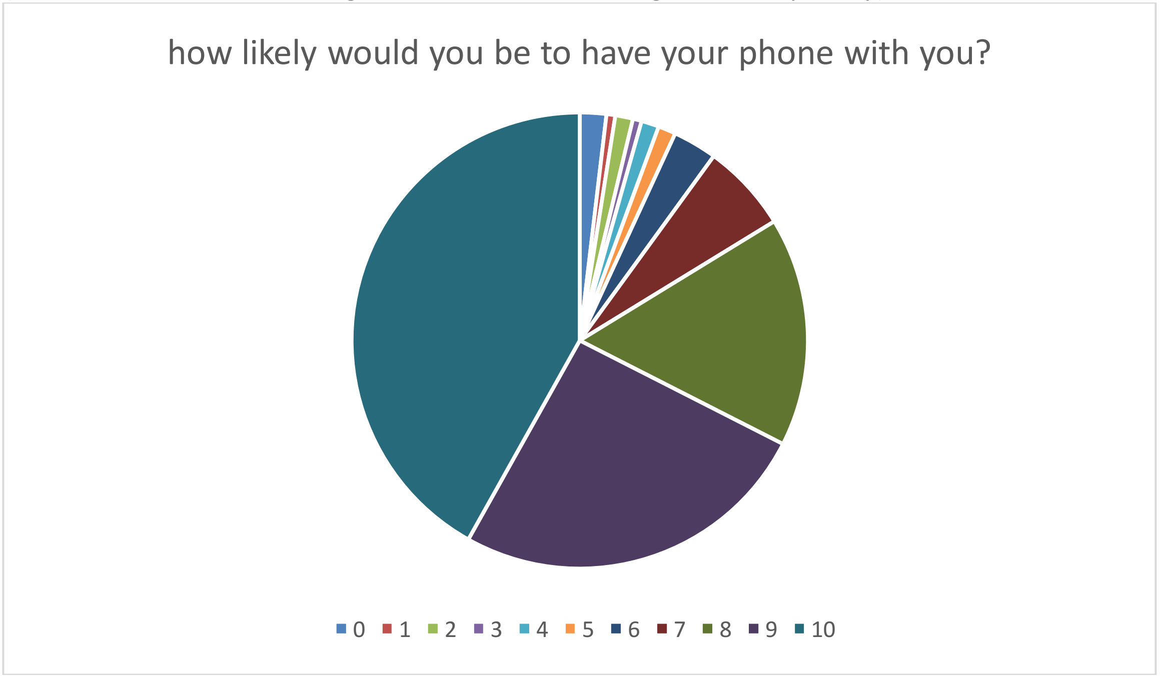
results: The majority of people who have taken this survey at some stage left their house forgetting their keys

conclusion: this question showed that the majority of people have experienced the problem that this product aims to solve. This is what the consumer has left their house forgetting their keys, meaning they either must get a locksmith, wait for someone with a key, or call someone in the house to let them in. with OpenSesamessage this is avoided by having the ability to just message the device and inlock the door

Question 6:

if yes, how likely would you be to have your phone with you?

(on a sale of 1 – 10. 1 being not at all, and 10 being extremely likely)



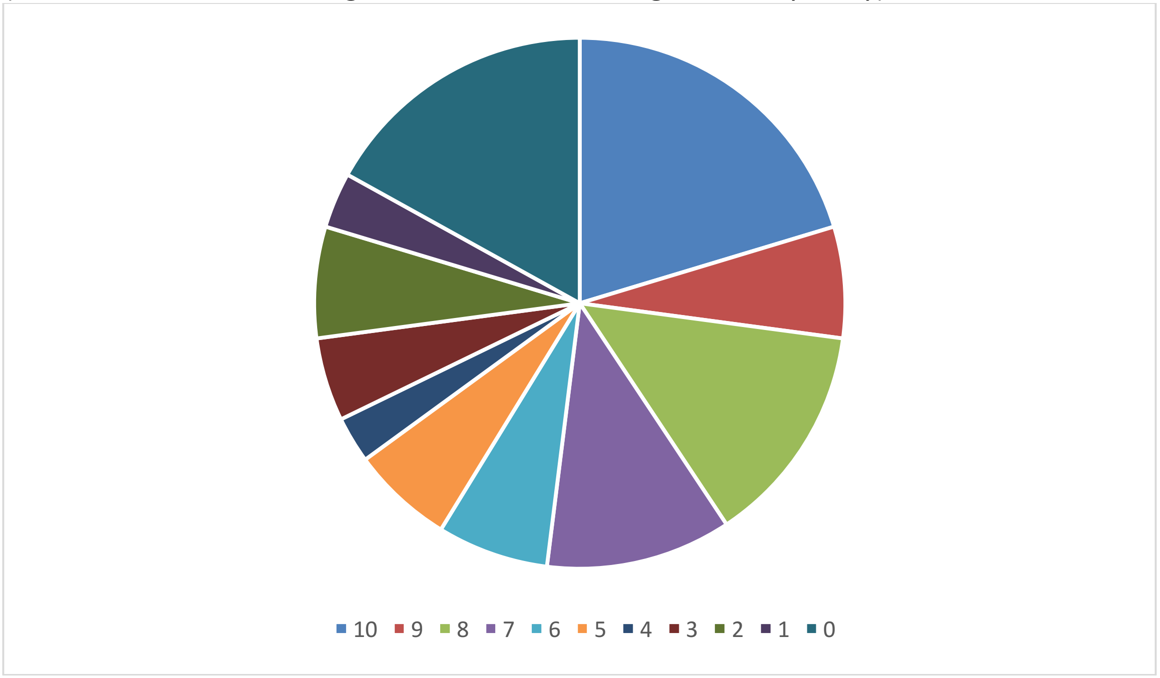
number of people answered:
158 out of 178

results: this data is for a follow up question, the data clearly shows that the likelihood of a person having their phone with them in the event of them forgetting their keys is quite likely, most people have answered either 10, 9 or 8 for this question. The average answer being 8.66

conclusion: this question further re-enforced the idea for this project. this device is operated using a smartphone. With users having access to their smartphone they can then use the device to access their home without the need for their keys.

Question 7:

how likely would you be to use a device to enter your home using your smartphone? (on a sale of 1 – 10. 1 being not at all, and 10 being extremely likely)



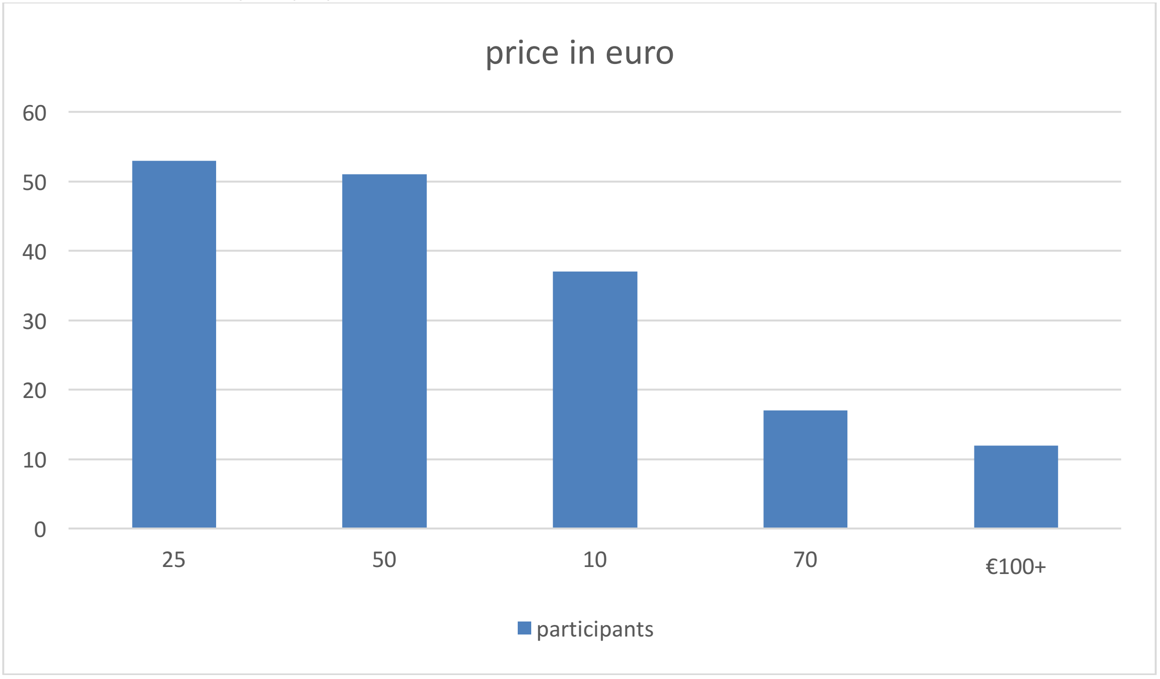
number of people answered:
176 out of 178

results: this data shows how likely a person taking this survey would be to have a device to allow entry to their home via smartphone. This data is scattered, a high percentage (17%). However, most the data scores 5 or higher.

conclusion: with the main number of people scoring 5 accounted for 64% of the people taking this survey. This shows that while there are people who are not interested, there are people willing to get this device, there are however clearly certain concerns

Question 8:

how much would you pay for a device such as the one described above?



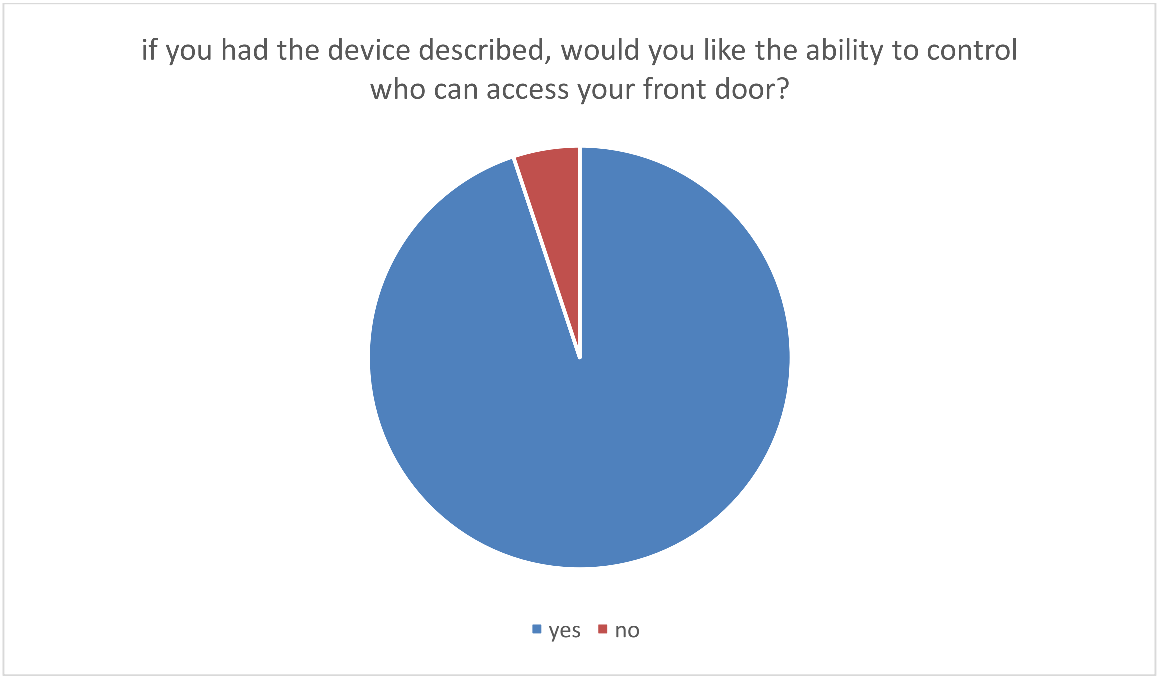
number of people answered:
170 out of 178

results: the participants showed interest in the product at a price ranger under €70

conclusion: this result shows that the participants would be willing to spend approximately €70 for a device such as the one this project is based. The initial research for this project showed that there are products on the market that perform a similar function, but at a price point of €150+. This further assures that the project would have an audience with consumers with a total cost of approximately 50 euro.

Question 9:

if you had the device described, would you like the ability to control who can access your front door?



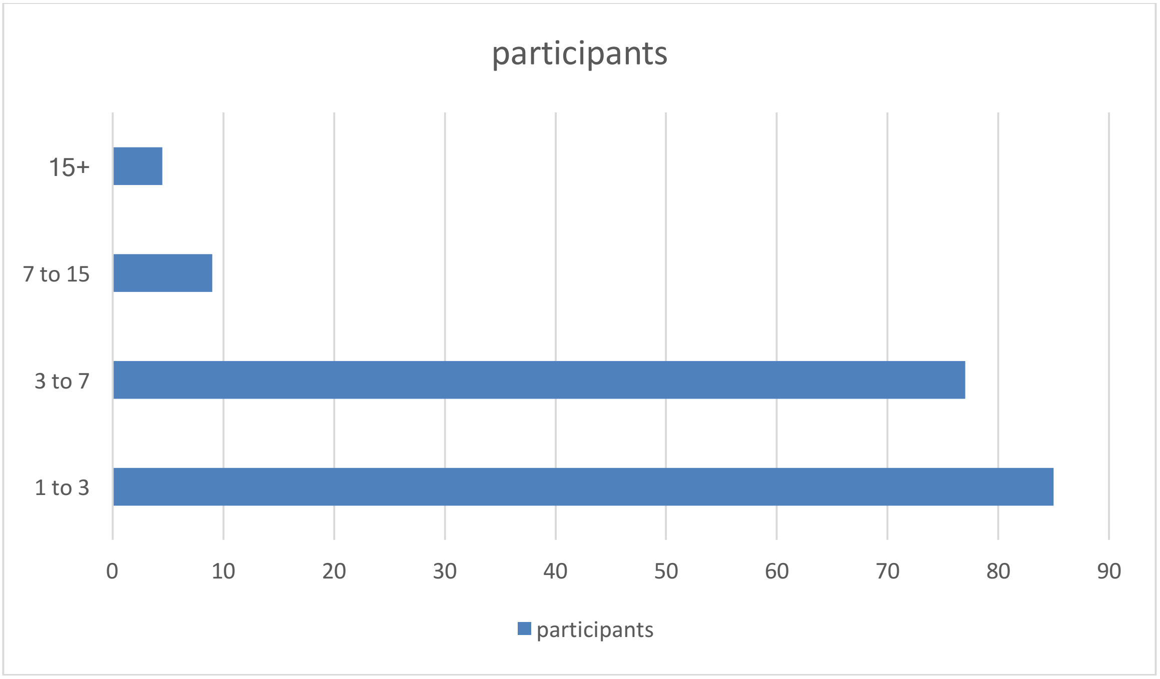
number of people answered:
176 out of 178

results: this question shows that the participants answered strongly in favour of having control over who has access to the lock. The participants answered with a majority of 95%

conclusion: the results of this question showed a 95% majority of people wanted to control who has access to the device. This is a security feature that has been implemented because of a high popularity of this question.

Question 10:

how many people, friends and family, would you allow to access your home?



number of people answered:
174 out of 178

results: the results of this question show the number of people a user of this service would have associated with their device. Most users would have between 1 and 7 people

conclusion: the results showed that the majority of users would not have more than seven users associated with an account. However, the device can facilitate an infinite number of users provided there is sufficient storage. This question was primarily to give an insight into how people would use OpenSesamessage

14 survey conclusion

the survey showed that people would be interested in a cost-effective home security system. However, based on the data the participants had some reservations. Further investigation is needed for a fully comprehensive consumer opinion, however it would seem that some people are unsure whether they are prepared to trust technology with access to their home. This investigation gave me a great insight into what people wanted in terms of this project's scope. It also provided me with ideas for functionality.

15 Conclusions

Milestones and Hurdles

Milestones where set by myself in two ways:
Self-imposed: these milestones had very little in terms of time requirements, but more of a learning and understanding standpoint:

- Design level, I have a great interest in front end web development and I hope to peruse a future in it. I feel this project allowed me to get a better understanding of web development in terms of user experience and styling
- Web development, I have a huge interest in the intricacies of web development and the plethora of tools available to developers today. The creation of functionality is of huge interest to me.
- IOT or internet of things development. This field allows for the creation of fun and unique devices and tools. I enjoyed working with the raspberry pi and already have plans to maintain this project and countless projects for the future

Although these milestones did not have deadlines I worked toward these milestones throughout the project, and hoped to excel in these.

Required: these milestones unlike the ones outlined above were a requirement for the project. these requirements added structure to the project and created achievable goals to strive for through the course of the project

- Ensuring monthly logs of development progress were kept. This allows for a timeline for the project
- Ensure a working prototype for the mid-point presentation.

These milestones made up the framework on which the entire OpenSesamessage project was based.

The Main **Hurdles** that where encountered:
IOT: the IOT device was a great hurdle, I had little to no experience in working with the raspberry pi entering this project, and it proved to be quite a substantial learning curve. The Linux operating system has great support communities that allowed me to research the development process.

The working with devices such as motors was new territory for me, working with GPIO pins and handling external power sources proved difficult. It showed me the modular nature of the IOT development process, I learned when one problem is encountered there is always another module that can allow for the requirements to be met.

telegram: telegram as a service allows the use of their cli, however with this project I was not creating functionality for its intended use. But from the raw tools provided by the messenger service I could use the service to suit this project. I had trouble setting up communication between the raspberry pi, mobile devices and the web application. However, with the use of the provided methods I could accomplish cross platform communication.

new languages: this project required me to implement new programming languages. When it comes to new languages there is always a learning curve, however with the languages used such as **Lua** and **bash script** there is always a teething period, which did end up being time consuming

time hurdles: this project began with another goal, I was I agreement to implement a front-end framework with a start-up company unituition.com. the goal was to implement react.js into the pre-existing website. However, after my midpoint presentation my project supervisor pointed out the loss in marks I was suffering with no database implementation among other things. After much consideration, I chose to take a new direction. I chose to create my own ruby on rails application and implement and iot device

16 Further Development & Research

Further development opportunities:

- **New devices**: I would like to continue development on this project by adding new devices, such as a light that can be triggered when the door is opening
- **Further explore security:** because of the nature of the project I would like to explore the security aspect of this project further. I have plans to implement a brute force attack protocol. When an unrecognised device makes too many requests within a certain time frame the device would close connections
- **Improve OpenSesamessage:** this can take many forms, but some ideas I have are cleaning up of the code further, adding more user requests that the device can handle, further exploring the ruby on rails web application
- **Product design:** I would like to design and develop a 3d printed case for the device
- **Active Release:** prepare a full copy of the device code and product list, and write up documentation for the device. I would like to allow people to use and create their own OpenSesamessage device and use it.

Research opportunities for OpenSesamessage:

- **User Feedback:** Use feedback provided from users to better understand issues or take advice on what should be implemented going forward.
- **User testing:** I plan to release the code under the MIT licence as an open source project and see how users of the service implement the device and the feature they request or add themselves to the device.

17 Closing Statement

On further development and research for OpenSesamessage I aim to continue development, this project has become a great hobby and has grown my interest in IOT development. It has increased my knowledge in the areas of web development, IOT development and design.
I would like to apply myself to a future in areas such as this, and am considering a masters in the field. I aim to specialise my knowledge in the area if IOT development and continue development of OpenSesamessage

18 References

"3.6.1 Documentation". *Docs.python.org*. N.p., 2017. Web. 8 May 2017.

"15.3. Time — Time Access And Conversions — Python 2.7.13 Documentation". *Docs.python.org*. N.p., 2017. Web. 8 May 2017.

"Active Record Associations — Ruby On Rails Guides". *Guides.rubyonrails.org*. N.p., 2017. Web. 8 May 2017.

"Active Record Basics — Ruby On Rails Guides". *Guides.rubyonrails.org*. N.p., 2017. Web. 8 May 2017.

"Active Record Migrations — Ruby On Rails Guides". *Guides.rubyonrails.org*. N.p., 2017. Web. 8 May 2017.

"Active Record Validations — Ruby On Rails Guides". *Guides.rubyonrails.org*. N.p., 2017. Web. 8 May 2017.

"Document". *Core.telegram.org*. N.p., 2017. Web. 8 May 2017.
"GPIO: Raspberry Pi Models A And B - Raspberry Pi Documentation". *Raspberrypi.org*. N.p., 2017. Web. 8 May 2017.

"Lua: Documentation". *Lua.org*. N.p., 2017. Web. 8 May 2017.

"Razzpisampler". *Razzpisampler.oreilly.com*. N.p., 2017. Web. 8 May 2017.

"Rspec Documentation". *Rspec.info*. N.p., 2017. Web. 8 May 2017.

"Using A Standard USB Webcam - Raspberry Pi Documentation". *Raspberrypi.org*. N.p., 2017. Web. 8 May 2017.

"Welcome To RPIO'S Documentation! — RPIO 0.10.0 Documentation". *Pythonhosted.org*. N.p., 2017. Web. 8 May 2017.

"YARD - A Ruby Documentation Tool". *Yardoc.org*. N.p., 2017. Web. 8 May 2017.

19 Appendix

Monthly Journals

Reflective Journal

Student name: Conor Breen
Programme: BSc in computing
Month: September
You don't have to follow the suggested format. These sub-headings and questions below may help you to get the most out of this journal, but you are free to modify as you see fit. Through this journal, you demonstrate that you are engaged with the process and that you can identify what you need to do or change to progress and succeed in this project.
Upload one journal every month. Expected word count 300 words (of you own words).

My Achievements

This month I was able to settle on a project for my final year. I met with a company called Unituition to discuss possible projects that I could do for them. After our meeting, we settled on a project. My project will consist of me implementing a front-end framework on their site. This is no small task; it involves completely redesigning the entire front end of their site which is written in basic CSS. I have chosen React.js to implement. React is a front-end framework developed by Facebook and is most notable used in the previous example as well as Instagram. It improves responsiveness of sites such by moving html elements into a 'components' file. This allows responsive rendering of html elements on the fly. As well as that is improves upon site efficiency making the pages' load faster and allows for quicker ajax calls to be performed.

My Reflection

This month I have implemented the framework into the site by using the react-rails gem. This gem generates the necessary files needed for ruby on rails to properly interact with the react component files. However, there have been some setbacks. Because the site has been created using mainly CSS classes written when needed. The application.css file contains thousands of lines of code that will need to be loaded every time the site renders a new page. Cleaning up the legacy code will play a huge part in this project.

Making proper use of bootstrap should make the site more efficient, however I have had trouble when using ruby code in the component.js.jsx files.

Intended Changes

Next month I will work on the user dashboard trying to clean up the interface as well as make it more intuitive. This involves rewriting tables, ruby calls to the database as well as ajax calls that confirm bookings.

Supervisor Meetings

Date of Meeting: n/a
Items discussed: n/a
Action Items: n/a

Reflective Journal

Student name: Conor Breen
Programme: BSc in computing
Month: October

My Achievements

This month saw slow progress with the project. Reactjs is proving simple in some areas, and cripplingly difficult in others. Because I'm using ruby on rails, I was able to avail of the react-rails gem, a package for rails users that allows easy install of dependencies, and with a couple lines of command line input I had the necessary component.js folder, the require tree and the compiler setup and ready to go. Initially I began coding in JavaScript to create and render new Reactjs components. This however raised a new challenge, JavaScript code that generates html is not very easy to read. And in a project with multiple developers I made the decision to change from JavaScript to jsx. This syntax resembles xml, so it is inherently easier for other developers to read.

Because my project is building a frontend framework on top of an already established, and constantly changing website, there has been some challenges in terms of version control. Having met with the lead developer, we agreed on branching out from the master branch, and having my project run parallel to the main project. From there, this branch will act as my master branch, allowing me to branch from it and merge code without the fear of altering something in the main site.

My Reflection

This month has proved rather difficult in terms of project work. Working only one day a week on a project of this scale has me worried about the time frame. My other modules are requiring more and more of my time and I fear that the added workload will hinder just how much work I am able to do. However, this month I wrote the code to allow react components make calls to the database, pulling user information. And populating tables with only minimal input, while generating the rest of the tables based my pre definitions. This means for a more optimised app

Intended Changes

Next month I will work heavily on active record, as well as creating the react equivalent of for loops, to loop through associations between students and tutors, populating both landing pages.

Supervisor Meetings

Date of Meeting: n/a
Items discussed: n/a
Action Items: n/a

Reflective Journal

Student Name: Conor Breen

Programme: BSc in Computing

Month: November

This month saw a slight lull in my projects progression. November saw a considerable increase in my time required for other modules. However, I did manage to overcome the problems I was experiencing earlier in the month. I created a component to handle all of the database calls relating to appointments on the site. rather than using a ruby method to loop through the database records, this new jsx component creates the table's skeleton and with the raw database information generates the html code to display the information to the user in tables. beyond this i hope to add additional functionality to this page, but more importantly to move on to a new task.

with the onslaught of in class assessments, project due dates and my presentation for my final year project to prepare for I'm finding myself more and more short of time. however with careful planning i feel that i can keep this project on track

My Achievements

As mentioned above i did make progress improving the efficiency of the user dashboard pages, by reducing the amount of server side code required to pull appointment information to display to the user

Although ephemeral, I am happy with this progress however I do feel I need to get a considerable amount covered over the Christmas break. I do, justifiably so have reservations on how productive i will prove to be over the break, but i remain optimistic

My Reflection

this month thought me to maintain my timekeeping better than i have been doing previously, and with he help on waffle.io i have created a SCRUM board to keep track of the backlog of tasks, tasks in progress and tasks that i have completed

Supervisor Meetings

Reflective Journal

Student name: Conor Breen
Programme BSHCIOT4
Month: December

Month Overview

After my project mid-point presentation Adrianna suggested it would be best for a change in direction, so I have decided to make my own ruby on rails application and implement front end features into that. Also, if possible add an IOT element. December. proved difficult for project work, social obligation meant that progress slowed somewhat.
However, although slow progress was made. I continued development on the ruby on rails website, the site now has its entire scaffold. A user model for the admin users of the service, as well as a sub user model for the people users add to the service. This is a security feature as it allows to user to grant access to new people using OpenSesamessage, while giving them the option to remove this privilege rather easily by logging into their account and removing them. However, this functionality does not exist, and for the month coming I plan on implementing telegram into the rails website via the use of a gem

My Achievements

This month was focused heavily on the development of ruby on rails, having worked on it for the last months, I felt comfortable programming in it now. Making changes and adding functionality. I am also keeping a list of functionalities that should be implemented in the hopes that time will allow. I have substantial plans for next month.

My Reflection

In reflection, I guess I knew that December would be a slow month, it would mean family events and work parties and every form of social obligation or final year project hindrance imaginable. However, December did act as a way to blow off steam. I enter January optimistic

Intended Changes

Work further on the raspberry pi, implementing a messenger service and finding a way to parse incoming information and use it to control the raspberry pi, and motor. Further information on how this all works will make itself apparent next month when further research is carried out

Supervisor Meetings

Date of Meeting:
Items discussed: a possible change in direction for my project, which will be tough but not impossible.
Action Items:

Reflective Journal

Student name: Conor Breen
Programme BSHCIOT4
Month: January

Month Overview

This month saw great progress in terms of programming. I have chosen the messenger service telegram to handle requests and interpret messages as commands. With the introduction of telegram into the project I have been able to solve the problem of interoperability. This project is now, be default multi-platform. I have chosen a programming language called Lua to parse the incoming messages and based on their content execute commands. For example, the message hello to the system will return a message hello #{user_name}. this is the simplest of the commands, but in the next coming months I will use messages to control os commands with the system, executing python code that I have written.

My Achievements

This month's achievements have put me on track for this project. I have purchased a stepper motor which should arrive shortly. With this motor, I can write a python program to activate it, in the hopes that it would open a lock on a door solely from a message on a popular messenger service. I also, have plans to implement a webcam to allow the admin user to receive image messages of who is entering the house.

My Reflection

This month saw great progress, especially over last month. However, there is much to do. I need to configure the motor with the door lock. This means writing a program to handle this functionality, as well as possibly creating a functionality for taking photos and sending them back to the admin user via multimedia message through telegram.

This month saw a mild disaster with the project code, I.e. ruby on rails code. The database got corrupted by fault of my own when testing migrating to AWS. Fortunately, I had my code on Github, so I was able to salvage the vast majority of it by forking my repository and creating a new one. OpenSesamessage-2.0

Intended Changes

Next month I hope to establish the stepper motor functionality and have users be able to trigger this action via a message to “open”. More works needs to be done on the website. I need to implement a front-end framework into the new website.

Supervisor Meetings

Date of Meeting: 08/02/17
Items discussed: getting a revised project documents to Adrianna. Because of the new direction, the project has taken
Action Items:

Reflective Journal

Student name: Conor Breen
Programme BSHCIOT4
Month: February

Month Overview

This month was a difficult one. I have growing fears over the timeline of this project, based on the workload that has fallen upon us in the IOT stream. With no final exams, we have a substantial amount of project work. However, I did make progress with the rails website, mainly in terms of styling. I am going to have to make time for the project report as time seems to be rapidly slipping through my fingers.

My Achievements

Progress with the website, I need to begin looking into hosting. I have purchased the domain name openssesamessage.com and plan to host it online. I will probably choose Digital Ocean as they provide affordable hosting packages for small scale projects, as well as a certain amount of hosting credit for students.

My Reflection

In reflection, I would like to have gotten more done. However, with the circumstances of college project work outside of the final year project this was impossible. However, I have hopes that next month I can make further progress with the project. in an upside to the project I have found that using git has saved me once again, reverting back through commits has meant that I haven't lost huge amounts of code and progress based on poor programming choices. I have learned that programming when tired and in the master branch is recipe for disaster.

Intended Changes

Next month I hope to continue on with this project making steady progress. I have had problems with the stepper motor and have ordered a new one. This new one will have increased torque over the other and should be better suited for the job

Supervisor Meetings

Date of Meeting: 17/01/17
Items discussed: a time plan was drawn out to better manage time in the project, also I need to get the documents discussed last month as time restraints have gotten in the way.
Action Items:

Reflective Journal

Student name: Conor Breen
Programme BSHCIOT4
Month: March

Month Overview

With the deadline approaching fast it has become apparent that time is a commodity now. The rapid approach of the project deadline has become shrouded by the onslaught of projects from other modules. I have growing fears that my final year project will have its quality lessened by the work load. However, this month did see some progress. I have the website styled and the stepper motor working on command through the telegram messenger service. What's required now is a webcam to take photos and send them back upon request and to setup security features.

My Achievements

The achievements this month are in the programming sense. The motor works, with python code running by commanding the GPIO pins, the website is better styled and the user accounts are up and running. However, I still need to establish communication between the web application and the IOT device. This I hope will be possible through the use of a telegram gem. Gems are packages in ruby that allows wrappers for certain API's. with the use of a suitable gem I hope that I will be able to establish communication between the two devices.

My Reflection

It is coming more and more apparent that time is short, obligations outside the scope of this college project have been creeping in and I have growing concerns over the final report. So far, a substantial amount of my project work has been programming. And where I am comfortable programming, I fear that writing technical documents is my Achilles heel.

Intended Changes

Next month I aim to focus more on the final report. The daunting size of it requires a large time commitment and next month will be very busy finishing module projects.

Supervisor Meetings

Date of Meeting: 14/03/17
Items discussed: further time management, and features
Action Items:

Other Material Used

Any other reference material used in the project for example evaluation surveys etc.