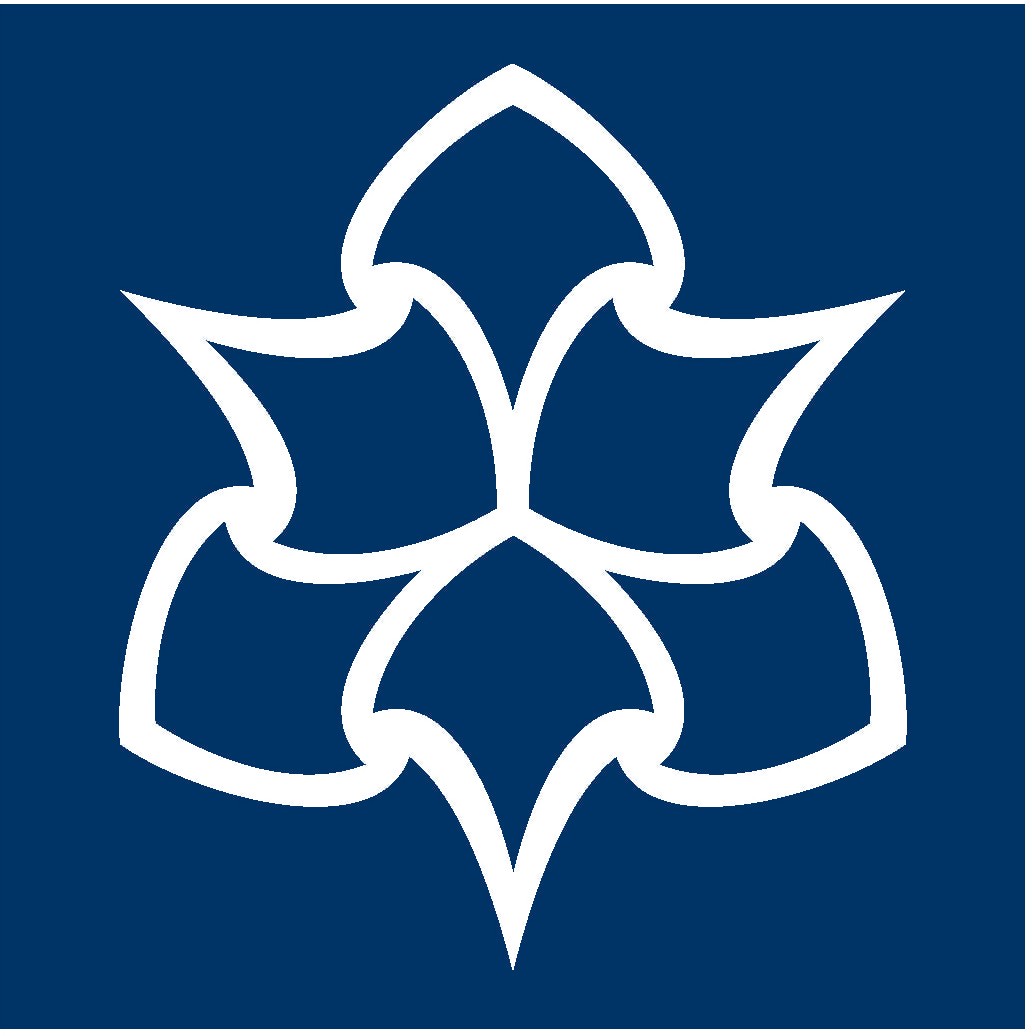
**Developing an IoT Android Application for Controlling and Monitoring a Smart Home System**

A REPORT SUBMITTED TO MANCHESTER METROPOLITAN UNIVERSITY FOR THE DEGREE OF BACHELOR OF SCIENCE IN THE FACULTY OF SCIENCE AND ENGINEERING



2021

By: Aubrey Monk

Supervisor: Dr Amna Eleyan

School of Computing, Mathematics and Digital Technology

**Contents**

[Declaration iv](#_Toc72709985)

[Acknowledgments v](#_Toc72709986)

[Abstract vi](#_Toc72709987)

[List of Figures vii](#_Toc72709988)

[List of Tables x](#_Toc72709989)

[Abbreviations xi](#_Toc72709990)

[1 Introduction 1](#_Toc72709991)

[1.1 Project Background 1](#_Toc72709992)

[1.2 Aim 3](#_Toc72709993)

[1.3 Objectives 3](#_Toc72709994)

[1.4 Report Structure 4](#_Toc72709995)

[2 Literature Review 5](#_Toc72709996)

[2.1 Benefits of Smart Home Technology 5](#_Toc72709997)

[2.2 Security Risks and Social Barriers of Smart Home Technology 6](#_Toc72709998)

[2.2.1 Security Risks 6](#_Toc72709999)

[2.2.2 Social Barriers 7](#_Toc72710000)

[2.3 Smart Home Hubs and Devices Currently on The Market 7](#_Toc72710001)

[2.3.1 Google Nest Hub Max 7](#_Toc72710002)

[2.3.2 Amazon Echo (4th Gen) 8](#_Toc72710003)

[2.3.4 Samsung SmartThings Hub v3 8](#_Toc72710004)

[2.3.5 August Wi-Fi Smart Lock 9](#_Toc72710005)

[2.3.6 Phillips Hue 9](#_Toc72710006)

[2.4 Programming Languages and Technologies 10](#_Toc72710007)

[2.4.1 Android applications 10](#_Toc72710008)

[2.4.2 APIs 11](#_Toc72710009)

[2.5 User Interface and Usability 11](#_Toc72710010)

[2.5.1 The Golden Rules 11](#_Toc72710011)

[2.5.2 Contrast, Colour, Size and Space 12](#_Toc72710012)

[2.6 Summary of Literature Review 13](#_Toc72710013)

[3 Design 14](#_Toc72710014)

[3.1 Development Methodology and Version control 14](#_Toc72710015)

[3.1.1 Agile Development Methodology 14](#_Toc72710016)

[3.1.2 Waterfall Development Methodology 14](#_Toc72710017)

[3.1.3 Version Control 15](#_Toc72710018)

[3.2 Overall System Design 16](#_Toc72710019)

[3.4 Android Application Design 18](#_Toc72710020)

[3.4.1 Framework and Environment 18](#_Toc72710021)

[3.4.2 Requirements 18](#_Toc72710022)

[3.4.3 Use Case Diagram 19](#_Toc72710023)

[3.4.4 Draft Wireframe 20](#_Toc72710024)

[3.4.5 Finalised Wireframe 24](#_Toc72710025)

[3.4.6 Flowchart 32](#_Toc72710026)

[3.5 Computer Client Design 33](#_Toc72710027)

[3.5.1 Language and Environment 33](#_Toc72710028)

[3.5.2 Class Diagram 33](#_Toc72710029)

[3.6 API Design 34](#_Toc72710030)

[3.6.1 Language and Environment 34](#_Toc72710031)

[3.6.2 Specification 34](#_Toc72710032)

[3.7 Database Design 37](#_Toc72710033)

[3.7.1 Language and Environment 37](#_Toc72710034)

[3.7.2 Entity Relationship Diagram 37](#_Toc72710035)

[4 Implementation 39](#_Toc72710036)

[4.1 API 39](#_Toc72710037)

[4.1.2 User Management 39](#_Toc72710038)

[4.1.3 Device Management 41](#_Toc72710039)

[4.2 Computer Client 42](#_Toc72710040)

[4.2.1 Main Class 43](#_Toc72710041)

[4.2.2 MQTT Client Class 43](#_Toc72710042)

[4.2.3 Callback Class 44](#_Toc72710043)

[4.2.4 Device Controller Classes 45](#_Toc72710044)

[4.3 Android Application 48](#_Toc72710045)

[4.3.1 Phidget Devices 48](#_Toc72710046)

[4.3.2 Screens and Components 49](#_Toc72710047)

[4.3.3 Testing 55](#_Toc72710048)

[5 Evaluation 79](#_Toc72710049)

[5.1 Evaluation of Project Aim 79](#_Toc72710050)

[5.2 Evaluation of Project Objectives 80](#_Toc72710051)

[5.3 Summary 82](#_Toc72710052)

[6 Conclusion 83](#_Toc72710053)

[6.1 Future Works 83](#_Toc72710054)

[References 84](#_Toc72710055)

[Appendices 87](#_Toc72710056)

# Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work. This work has been carried out in accordance with the Manchester Metropolitan University research ethics procedures and has received ethical approval number 2020-26247-20539.

Signed:

# Acknowledgments

I’d like to thank my project supervisor Dr Amna Eleyan for the support and guidance throughout this project and, for fantastic teaching throughout my time here.

I’d also like to thank: Dr David McLean, Connah Kendrick, Dr Bob Cherry, Marie Carrol, Stephen Gordon, Laurie Cooper, Dr Matthew Crossley, Dr Giovanni Masala, Dr Kristopher Welsh, Dr Mohammed Kaleem, Dr Matthew Shardlow, Dr Ryan Cunningham, Dr Nicholas Costen, Dr John Darby and Dr Ashley Williams for all being outstanding lecturers and tutors during my course and inspiring my passion for Computer Science.

Finally, I’d like to thank Luciano Gerber and Sandra Brooks for helping me through one of my toughest times here by offering tremendous guidance and support, I wouldn’t have made it this far without you.

My gratitude to you all for everything you have done.

# Abstract

There are many benefits to smart home technology and many related risks, this project addresses some these benefits and risks and aims to implement a product that will help with the further development of smart home technology.

The aim of this project is to create an Android application that allows a user to control or monitor smart devices within their home, the system must be cost-effective, efficient and easy to use, this is so that it is more accessible to a wider range of people than some of the systems and devices that are currently on the market.

# List of Figures

Figure 1: “The things home automation can do” (Source: Diy Doctor, 2020: online). 1

Figure 2: “Thirteen smart home technology benefits discussed by expert interview respondents” (Sovacool and Furszyfer Del Rio 2020). 2

Figure 3: “Prospective users’ perceptions of the purpose & benefits of smart home technologies” (Wilson et al., 2017) 5

Figure 4: “Google Nest Hub Max” (Source: Google Store, 2021: online). 7

Figure 5: “Amazon Echo (4th Generation)” (Source: Amazon, 2021: online). 8

Figure 6: “Samsung SmartThings Hub V3” (Source: Tom’s Guide, 2021: online). 8

Figure 7: “August Wi-Fi Smart Lock” (Source: The Verge, 2021: online). 9

Figure 8: “Phillips Hue Starter Kit” (Source: Amazon, 2021: online). 10

Figure 9.1: “Triadic Colour Structure” (Source: visual composer, 2021: online)) 12

Figure 9.2: “Split Complementary Colour Structure” (Source: visual composer, 2021: online)) 12

Figure 9.3: “Analogous Colour Structure” (Source: visual composer, 2021: online)) 13

Figure 10: “Agile Development Methodology” (Source: Synopsys, 2021: online). 14

Figure 11: “Waterfall Development Methodology” (Source: Wikipedia, 2021: online). 15

Figure 12: “First 2 Attempts at Overall System Design” 16

Figure 13: “Finalised Overall System Design” 17

Figure 14: “Lists of Android Application Requirements” 18

Figure 15: “Use Case Diagram of Android Application” 19

Figure 16: “Page 0 of Draft Wireframe” 20

Figure 17: “Page 1 of Draft Wireframe” 21

Figure 18: “Page 2 of Draft Wireframe” 21

Figure 19: “Page 3 of Draft Wireframe” 22

Figure 20: “Page 4 of Draft Wireframe” 22

Figure 21: “Page 5 of Draft Wireframe” 23

Figure 22: “Page 6 of Draft Wireframe” 23

Figure 23: “Wireframe: Notifications” 24

Figure 24: “Wireframe: Welcome Screen” 24

Figure 25: “Wireframe: Sign Up Screen” 25

Figure 26: “Wireframe: Login Screen” 25

Figure 27: “Wireframe: Home Screen” 26

Figure 28: “Wireframe: Lights Screen” 26

Figure 29: “Wireframe: Security Screen” 27

Figure 30: “Wireframe: Power Screen” 27

Figure 31: “Wireframe: Temperature Screen” 28

Figure 32: “Wireframe: Add Device Screen” 28

Figure 33: “Wireframe: Drawer” 29

Figure 34: “Wireframe: Settings Screen” 29

Figure 35: “Wireframe: Notification Settings Screen” 30

Figure 36: “Wireframe: Accessibility Settings Screen” 30

Figure 37: “Wireframe: Update User Info Screen” 31

Figure 37.1: “Analogous Colour Scheme of Application” 31

Figure 38: “Flowchart” 32

Figure 39: “Client Class Diagram” 33

Figure 40: “API Specification” 35

Figure 41: “API Specification: Add Device” 36

Figure 42: “API Specification: Code” 36

Figure 44: “Second Database ERD” 37

Figure 45: “Third Database ERD” 37

Figure 46: “API: User Routes” 38

Figure 47: “API: User Model, Create method” 39

Figure 48: “API: User Controller, Create method” 40

Figure 49: “API: Device Routes” 40

Figure 50: “API: Device Model, List method” 41

Figure 51: “API: User Controller, Create method” 41

Figure 52: “Computer Client Running” 42

Figure 53: “Client: Main Class” 42

Figure 54: “Client: MQTT Client” 44

Figure 55: “Client Callback: On Message Arrived” 45

Figure 56: “Motion Sensor Controller: Activate Sensor” 46

Figure 57: “Motion Sensor Controller: Check if Sensor Exists” 47

Figure 58: “Motion Sensor Controller: Create Sensor” 47

Figure 59: “Motion Sensor Controller: Check Sensor” 47

Figure 60: “Phidget Devices Used for Testing and Implementation” 48

Figure 61: “Application: Login Screen” 49

Figure 62: “Application: Login Component” 50

Figure 63: “Locks Screen: Lock/Unlock Function” 51

Figure 64: “Locks Screen: Check Lock Status Function” 52

Figure 65: “Locks Screen: Get Device List Function” 52

Figure 66: “Locks Screen: On Component Load” 53

Figure 67: “Locks Screen: User Interface” 54

Figure 68: “Application: List Devices Component” 55

# **List** **of** **Tables**

Table 1: “Requirements Traceability Matrix” 59

Table 2: “Test Cases” 78

# Abbreviations

IoT Internet of Things

MQTT Message Queuing Telemetry Transport

PC Personal Computer

API Application Programming Interface

SQL Structured Query Language

JS JavaScript

TCP/IP Internet Protocol Suite

USB Universal Serial Bus

DDoS Distributed Denial of Service

UI User Interface

ERD Entity Relationship Diagram

UUID Universally Unique Identifier

# 1 Introduction

## 1.1 Project Background

Smart home automation systems have become increasingly common in the last decade due to the arrival of the Internet of Things (Ali et al., 2020). The Internet of Things is an “interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless ubiquitous sensing, data analytics and information representation with Cloud computing as the unifying framework.” (Gubbi et al., 2013:1647) for IoT to be able to connect such a large number of devices it needs to have a flexible and layered architecture, the most basic model of IoT consists of a 3 layer architecture which includes the Perception, Network and Application layers (Choudhary and Jain 2016).

There are many different applications of IoT including wearables, agriculture, smart grids, hospitality, connected health, smart cities and more, but this project will be focused on its smart home automation applications. A smart home system is normally made up of IoT devices such as cameras, sensors, actuators, and appliances that can be accessed remotely, typically through a centralised hub or application (Rizvi et al., 2018) as shown in Figure 1 below.



Figure 1: “The things home automation can do” (Source: Diy Doctor, 2020: online).

For IoT devices to be able to transmit data over a network they need to use a certain protocol, the main protocol that will be used in this project is MQTT which is very lightweight and usually runs over TCP/IP and uses a publish/subscribe concept to transfer information through a broker.

Smart home systems can also increase the quality of life for many (Miah and Khan 2019), especially for certain groups of people such as the blind and visually impaired. For instance, if a blind or visually impaired person needed to use one of the devices or appliances in their home, lock/unlock the doors or control the temperature/lights they can do so using a voice command through a smart home control application; this cuts out the need for any physical interaction making the task significantly easier. Also, sensors such as motion and sound can be used to monitor the well-being of the inhabitant and detect if any assistance is needed. A huge benefit of this system is that it is much more cost efficient and favourable than having the support of a personal assistant (Rizvi et al., 2018). Figure 2 below shows thirteen main benefits of smart home technology.

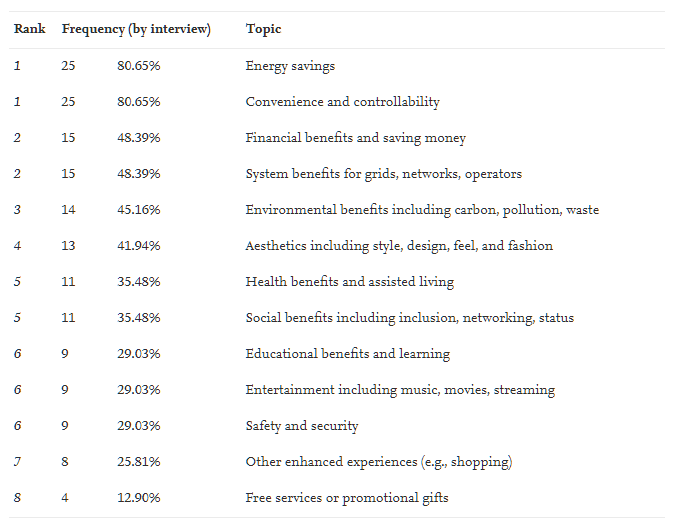


Figure 2: “Thirteen smart home technology benefits discussed by expert interview respondents” (Sovacool and Furszyfer Del Rio 2020).

Despite the numerous benefits of smart home technologies and systems they do come with many risks and social barriers, the main social barrier that is addressed within this project is the high cost of smart home systems and there accessibility towards a wider range of people.

The smart home devices used in this project are all Phidget devices which are inexpensive electronical components and sensors which can be connected to a computer via a USB connection. The project is made up of four main components: A React Native Android application, a JavaScript API, a MySQL relational database and a Java computer client. The Android application provides a user interface enabling a user to add their Phidget devices to the system and monitor/control the added Phidget devices. The Android application can communicate with a database via the API, the database used is a MySQL relational database running on the university’s ‘mudfoot’ server. This database can contain a user’s account information and connected Phidget devices, this means the system can support multiple users. The client runs on a computer that has all the Phidget devices connected to it, the Android application can communicate with this client through an MQTT broker, meaning that any Phidget devices connected can be controlled or monitored.

## 1.2 Aim

The aim of this project is to create an Android application that allows a user to control or monitor smart devices within their home, the system must be cost-effective, efficient and easy to use, this is so that it is more accessible to a wider range of people than some of the systems and devices that are currently on the market.

## 1.3 Objectives

* Find existing related works and mobile applications for controlling and monitoring smart home systems
* Identify user requirements and features
* Research and choose an appropriate software design methodology
* Research appropriate languages and technologies that can be used for Android applications, APIs, computer clients, databases, and voice recognition
* Research UI design that can be used for Android applications
* Research and choose appropriate smart home devices
* Design and Wireframe Android application
* Design database system
* Design API used to communicate with database system
* Design computer client that will be connected to the smart home devices
* Implementation of the Android application, database system, API and computer client and voice recognition
* Thoroughly test and evaluate the system and provide the results
* Conclude project with summary of the created application, the results produced and provide suggestions for further related works

## 1.4 Report Structure

* Chapter 1 – Introduction
* Chapter 2 – Literature Review
* Chapter 3 – Design
* Chapter 4 – Implementation
* Chapter 5 – Evaluation
* Chapter 6 – Conclusion

# **2 Literature Review**

## 2.1 Benefits of Smart Home Technology

There are numerous benefits of smart home technology, this section of the literature review will look at other research conducted towards identifying some of these benefits.

Smart home systems can make life significantly easier, more comfortable, and convenient. These systems are also able to provide peace of mind, for example if you’re away from home you can use these systems to monitor your house, for instance it could provide the ability to detect an intruder or provide help in case of an emergency, as you could potentially be notified when the fire alarm is activated and have the ability to unlock doors and call for help. Smart home technology also can provide assistance to the elderly and disabled that are living alone, it can be used to monitor health, contact carers/emergency services in case of an accident and also help with daily tasks that may be difficult to the user or even tasks that they may have forgotten to do (Robles and Kim 2010).

The first study found carried out multiple expert interviews and performed a frequency analysis of the benefits stated during these interviews, the results of which can be found in Figure 2 [Chapter 1.1]. The analysis results show that the most prominent benefits identified were energy savings and convenience/controllability. Others include safety and security as well as health benefits and assisted living. (Sovacool and Furszyfer Del Rio 2020).

Another study found carried out a national survey that characterised how the prospective users viewed the benefits of smart home technology; Figure 3 below shows these results.

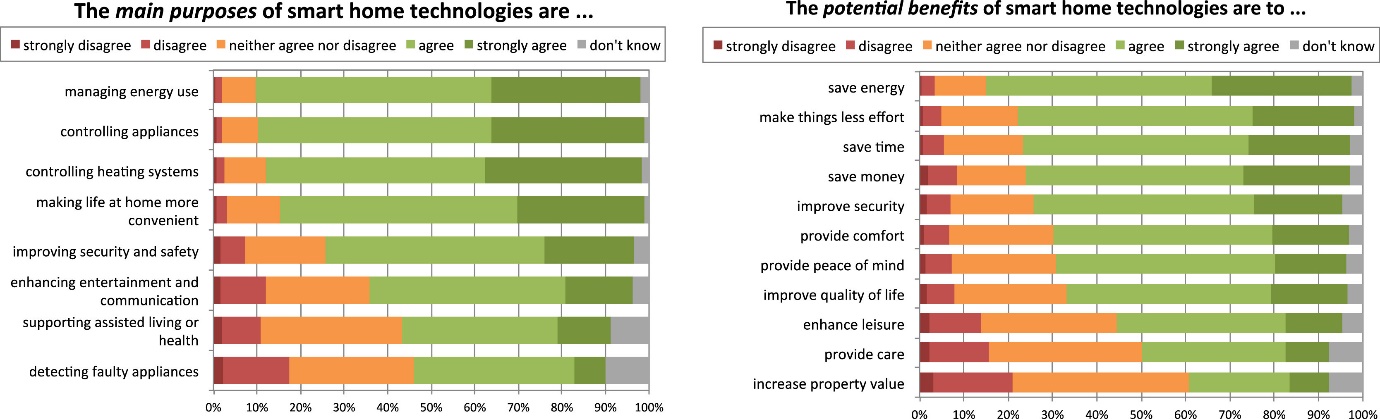


Figure 3: “Prospective users’ perceptions of the purpose & benefits of smart home technologies” (Wilson et al., 2017)

Again, we can see that some of the benefits identified by the participants include, managing/saving energy consumption and convenience, supporting assisted living and safety/security.

## 2.2 Security Risks and Social Barriers of Smart Home Technology

### 2.2.1 Security Risks

Network security is a major concern with smart home systems, private and personal information exists on two different mediums within a network, either it is stored physically on a hard disk or it can be transported across a network. Here are 5 known methods used to compromise this information:

* *Packet sniffers*
* *Password attacks*
* *IP Spoofing*
* *Man-in-the-middle attacks*
* *Distribution of sensitive internal information to external sources*

(Robles and Kim 2010).

Privacy is a major concern when it comes to this personal information, smart home systems could potentially have different devices, sensors and cameras that collect information. If someone obtained this data, it could be used for various malicious purposes. Another possibility is if a hacker gained access to an inhabitant’s passwords, they could use this to gain unwarranted access into the inhabitant’s home.

Another security threat is Distributed Denial of Service Attacks, a DDoS attack is a coordinated, large scale attack on a network’s resources or the availability of a systems services (Robles and Kim 2010). If an inhabitant relies on the services provided by their smart home, then a DDoS attack could provide major problems if the network or services are taken down for a long period of time.

### 2.2.2 Social Barriers

There are many different difficulties and social barrier when it comes to adopting smart home technology some of these include technical, conceptual and management issues (Balta-Ozkan et al. 2013). This section of the literature review will cover some of these in further detail.

Overall, the main reasons such barriers exist is due to the perceived risk of the consumer, normally stemming from uncertainty and possible negative consequences (Hong et al. 2020). A few perceived risks include performance risk, privacy risk, psychological risk, and financial risk. Consumer survey results show that financial risk is the main barrier for adopting smart home technologies (Hong et al. 2020). Another survey found also suggested that cost of these technologies posed as a barrier, multiple groups from the survey agreed that the adoption of this technology was only worthwhile for financially stable long-term home owners, excluding inhabitants of rented or council properties or those on a lower income (Balta-Ozkan et al. 2013).

## 2.3 Smart Home Hubs and Devices Currently on The Market

A smart home system typically comprises of multiple IoT devices normally connected to a central hub, however there are also standalone devices that do not requires a hub. This section of the literature review will cover existing smart home systems, hubs and devices that are currently on the market.

### 2.3.1 Google Nest Hub Max

The Google Nest Hub Max is a 10-inch smart display which supports Wi-Fi, Bluetooth, and Thread connections. From the display you can watch videos, play music, make video calls, and control any Google Home compatible devices. The Nest Hub Max retails at around £219 (Google 2021)



Figure 4: “Google Nest Hub Max” (Source: Google Store, 2021: online).

### 2.3.2 Amazon Echo (4th Gen)

The Amazon Echo is a smart speaker that acts as a smart home hub. It supports Bluetooth, Wi-Fi and Zigbee connections. The Amazon Echo uses Amazon Alexa as a voice assistant which allows the user to control smart devices through voice commands and configure complex routines and schedules. It also uses Zigbee radio which means it can connect to many different low power smart home devices such as sensors and lightbulbs. The Amazon Echo retails at around £89.99 (Amazon 2021)



Figure 5: “Amazon Echo (4th Generation)” (Source: Amazon, 2021: online).

### 2.3.4 Samsung SmartThings Hub v3

Samsung SmartThings Hub v3 is a smart home hub, it supports Samsung’s own smart home devices but also many others on the market allowing you to control, monitor and schedule these devices via an IOS or Android mobile application. The Samsung SmartThings Hub v3 retails at around £90.98 (Amazon 2021)



Figure 6: “Samsung SmartThings Hub V3” (Source: Tom’s Guide, 2021: online).

### 2.3.5 August Wi-Fi Smart Lock

The August Smart Lock is a battery powered lock that is controlled over a Wi-Fi connection. You are able to control the lock using the August mobile application and it also supports a very wide range of other hubs, platforms and devices including some of the previously mentioned smart home systems. The August Wi-Fi Smart Lock retails for around $249.99 (August 2021)



Figure 7: “August Wi-Fi Smart Lock” (Source: The Verge, 2021: online).

### 2.3.6 Phillips Hue

Phillips Hue is a range of smart lightbulbs and devices that support an extremely wide range of other platforms, smart hubs, and smart home systems. The reason Phillips Hue can support so many devices is because they have an open and accessible API allowing almost anyone to develop software for use with the Hue technology. There is also a Phillips Hue mobile application that allows for control of all the Phillips Hue devices over a Wi-Fi connection. The Phillips Hue Starter kit E27 retails for around £119.99 (Phillips 2021)



Figure 8: “Phillips Hue Starter Kit” (Source: Amazon, 2021: online).

## 2.4 Programming Languages and Technologies

### 2.4.1 Android applications

There are lots of different options when it comes to choosing a programming language for Android application development, some of the most popular languages include: C++, C#, Python, Corona, Java, Kotlin and JavaScript. The two that will be focused upon in this chapter are JavaScript and Java.

Java is an object orientated programming language and is the most popular language used for Android development. The benefits of Java are that it is high-level, has simple syntax, automatic memory management and has a large supporting community around it. However, a major drawback is that it takes a long time to develop in Java as it can have large and highly complex code.

JavaScript is used alongside many different frameworks when it comes to application development such as: Titanium, React Native, Meteor, Rachet and many other. The main framework that will be focused upon is React Native. React Native is an open-sourced framework created by Facebook and is used to develop applications for many different platforms. React native is easy to learn saving lots of time, its flexible in the sense that it is easy to upgrade and change code, it also offers high performance as applications written in React Native perform the same as a native application running on a specific platform, and unlike Java the code is not so large and complex as React is normally used for simpler applications.

### 2.4.2 APIs

An API or Application Programming Interface is a type of software that allows multiple applications to communicate with each other. In the case of this project it will be used for handling any request that the Android application has that requires use of the Database.

Popular languages for the development of APIs include: Java, Python, NodeJS (JavaScript runtime environment) and C#. This section will be focusing on Python and NodeJS for API development.

Python is easy to use and very popular within all kinds of academia, it is dynamically typed and very flexible. However, Python is an interpreted language meaning that speed can sometimes be an issue.

NodeJS is a JavaScript runtime environment that offers a sturdy and robust technology stack, it fast, scalable in nature and supports JSON seamlessly. However, the complex nature of callback functions in NodeJS can cause lots of issues and can take time to debug. NodeJS can often experience bottlenecks with tasks that require high computation.

## 2.5 User Interface and Usability

### 2.5.1 The Golden Rules

Ben Shneiderman set out 8 Golden rules for an applications usability, they are as follows

1. *Strive for consistency*
2. *Seek universal usability*
3. *Offer informative feedback*
4. *Design dialogs to yield closure*
5. *Prevent errors*
6. *Permit easy reversal of actions*
7. *Keep users in control*
8. *Reduce short-term memory load*

These rules provide a starting point when taking an applications usability into account, they aim to increase a user’s productivity, increase competence, and give as much control as possible over a system (Shneiderman. et al., 2016).

### 2.5.2 Contrast, Colour, Size and Space

Contrast is important to consider when designing a UI, its primary purpose is to generate interest. It is important to have contrast between the primary actions of a screen/page and the other elements. Colours are typically divided into two groups, warms colours and cool colours. Typically, warm colours tend to overpower cool colours meaning they are good to use for the main elements of a screen/page (Williams 2021).

When choosing a colour scheme there are a few different accepted compositions that can be chosen from:

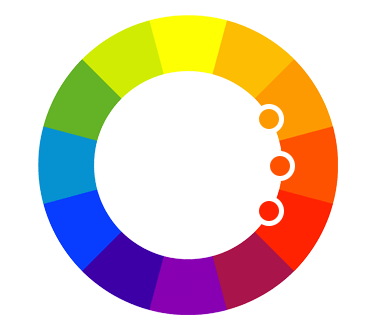
Triadic (Figure 9.1: “Triadic Colour Structure” (Source: visual composer, 2021: online)):



Split Complementary (Figure 9.2: “Split Complementary Colour Structure” (Source: visual composer, 2021: online)):



Analogous (Figure 9.3: “Analogous Colour Structure” (Source: visual composer, 2021: online)):



It is also important to consider the use of size and space when designing a UI, things emphasize or de-emphasize a specific elements of the UI whereas a good use of space can decrease visual noise and once again help the other main elements stand out (Williams 2021).

## 2.6 Summary of Literature Review

As revealed by this literature review there are multiple benefits of smart home technology that gives an incredible motivation into the development of this application and system. However, there are many risks and barriers to be considered such as security and most importantly financial risk for the consumer. Many of the products currently on the market would cost a large sum of money as you would need to purchase a smart home hub along with many smart devices. This project aims to deliver a smart home system that is secure and affordable. This will be done by taking security precautions during development of the system and to keep things affordable the project will be using inexpensive electronical components to act as the hub and smart home devices. Upon review, React Native will be chosen for the Android Applications development, Java will be chosen for the computer client (smart home hub) and NodeJS for the API. This project also aims to follow the “8 Golden Rules” [Chapter 2.5.1] for usability as closely as possible and hopes to create an efficient, easy to use and well thought out user interface that will allow a user to have full control over any smart devices in the home.

# 3 Design

The Design chapter covers all the tools, technology’s, languages, and design techniques that were considered before implementation of the system and the reasons behind these choices. This chapter also covers the development methodology and planning of the implementation required so that development could run as smoothly and efficiently as possible.

## 3.1 Development Methodology and Version control

The development methodology chosen for this project was the Waterfall methodology but with some aspects of the Agile methodology included at each stage of development so that if any bugs, problems or system changes arise occur they can be reviewed and dealt with easily. For version control of the system, GitHub was chosen.

### 3.1.1 Agile Development Methodology

The Agile methodology revolves around an iterative development of a system where features of the system are delivered incrementally as opposed to all at once, this means that features can be reviewed at each stage and feedback can be given. This type of development is low risk as if the requirements of the system change or bugs are found they can easily be dealt with before the system is finished.



Figure 10: “Agile Development Methodology” (Source: Synopsys, 2021: online).

### 3.1.2 Waterfall Development Methodology

The Waterfall development methodology consists of certain steps that are completed in a linear fashion. Each phase must be completed before the next one can begin. This methodology is very simple however, it means that it is hard to react to requirement changes, bugs or other problems that may occur during the development process.

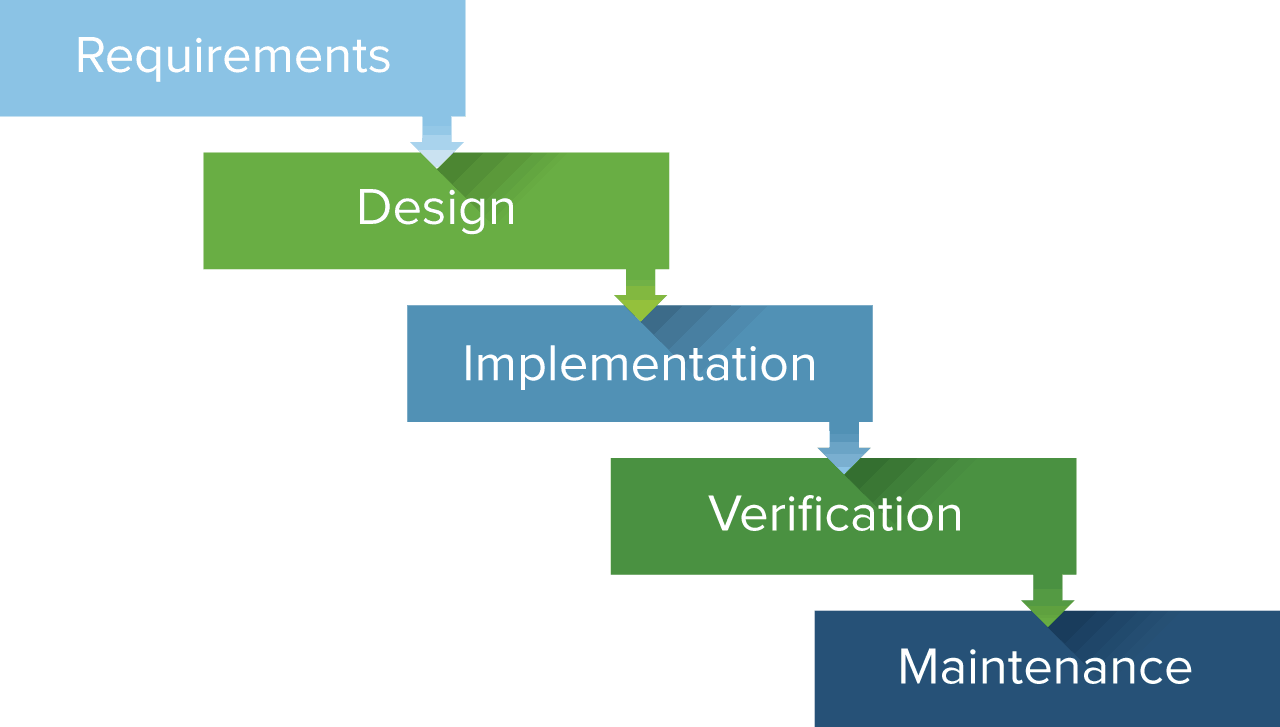


Figure 11: “Waterfall Development Methodology” (Source: Wikipedia, 2021: online).

### 3.1.3 Version Control

GitHub (<https://github.com/>) was the chosen platform for version control. GitHub is a platform for collaboration and version control of a project, you are able to commit your code to a repository hosted online, you can also create separate branches of the project that can be used for keeping bug fixes or certain features separate. You can access the project from any device by creating a pull request for a specific branch so that changes to the code can be made from anywhere. The repository for this project can be found here: <https://github.com/Aubrey-Monk/Smart-Home-System---3rd-Year-Project>

## 3.2 Overall System Design

The first major part of the design process to be considered was the overall design of the system, this was important as it led to choosing the appropriate technologies used in the system. It also gave insight into how each component interact with each other.

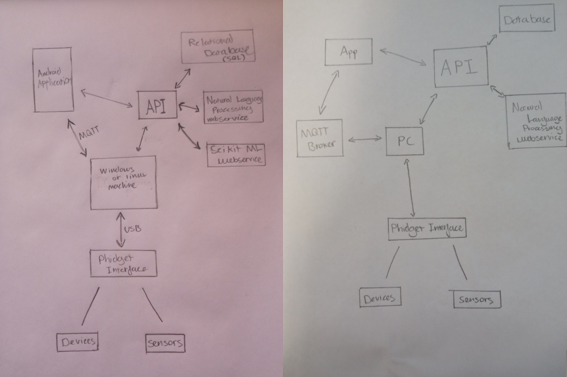


Figure 12: “First 2 Attempts at Overall System Design”

Figure 12 shows the first 2 diagrams created to illustrate overall system design, these were both drafts and came with issues and inconsistencies.

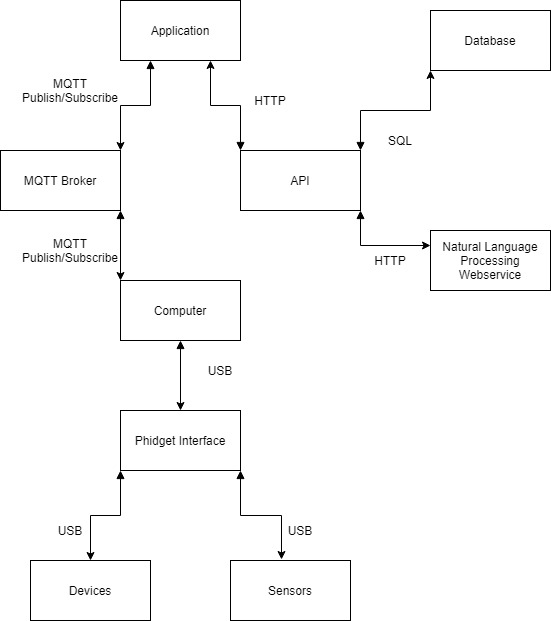


Figure 13: “Finalised Overall System Design”

Figure 13 shows the final diagram for the overall system design, the boxes show each individual component, the arrows show the direction of information travel and the labels represent the medium across which the information sent between the system is travelling.

## 3.4 Android Application Design

This section covers all the choices made and includes all the diagrams created when designing the Android Application.

### 3.4.1 Framework and Environment

The framework chosen to develop the application was React-Native, the application needs to be of little complexity to keep it efficient as possible, it also needs to be able to handle the API responses in a flexible manner. React-Native works well for both these purposes, it runs natively on the device so performance is great, and the UI options are extremely flexible. React code is quick to write and easy to learn which is also another great benefit of choosing this framework.

The development environment chosen was Visual Studio Code with the ESLint (Code Linter) and Prettier plugins, these were both used to improve code quality also, to reduce errors. The style guide chosen for the linter was Airbnb. Found here: <https://github.com/airbnb/javascript>

### 3.4.2 Requirements



Figure 14: “Lists of Android Application Requirements”

Figure 14 outlines the requirements of the application, it includes the core, sacrificial and out of scope requirements, this list was extremely useful in designing the rest of the application as it provided a basis of which features need to be included.

### 3.4.3 Use Case Diagram

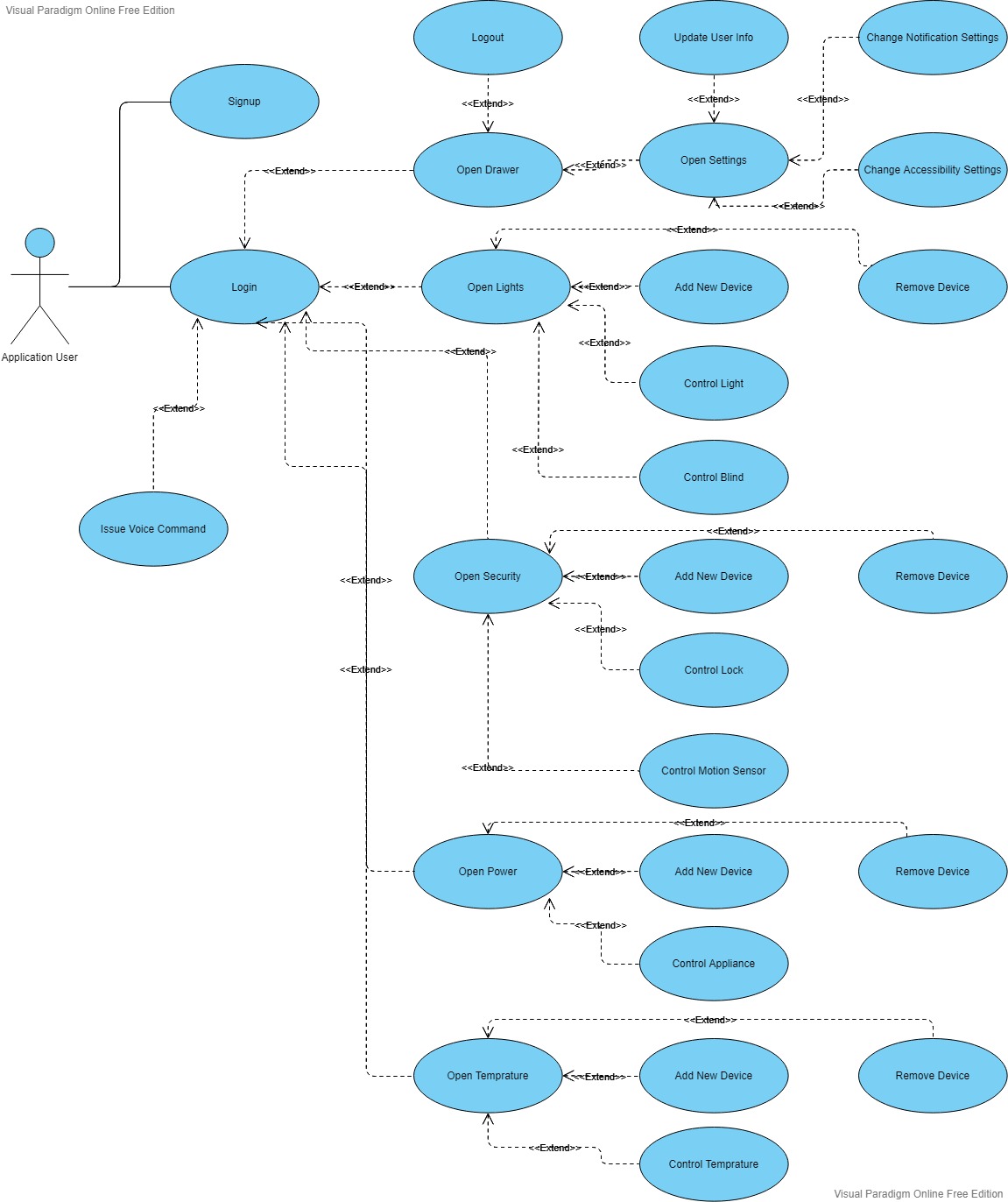


Figure 15: “Use Case Diagram of Android Application”

Figure 15 is a Use Case diagram of the application and is used to see every possible action a user can perform; it visually represents the user requirements.

### 3.4.4 Draft Wireframe

Below is the draft wireframe of the application, drafting the wireframe proved important as it allowed a visualisation of application and aided in refining the UI design. It also provided insight into the navigation structure of the application.

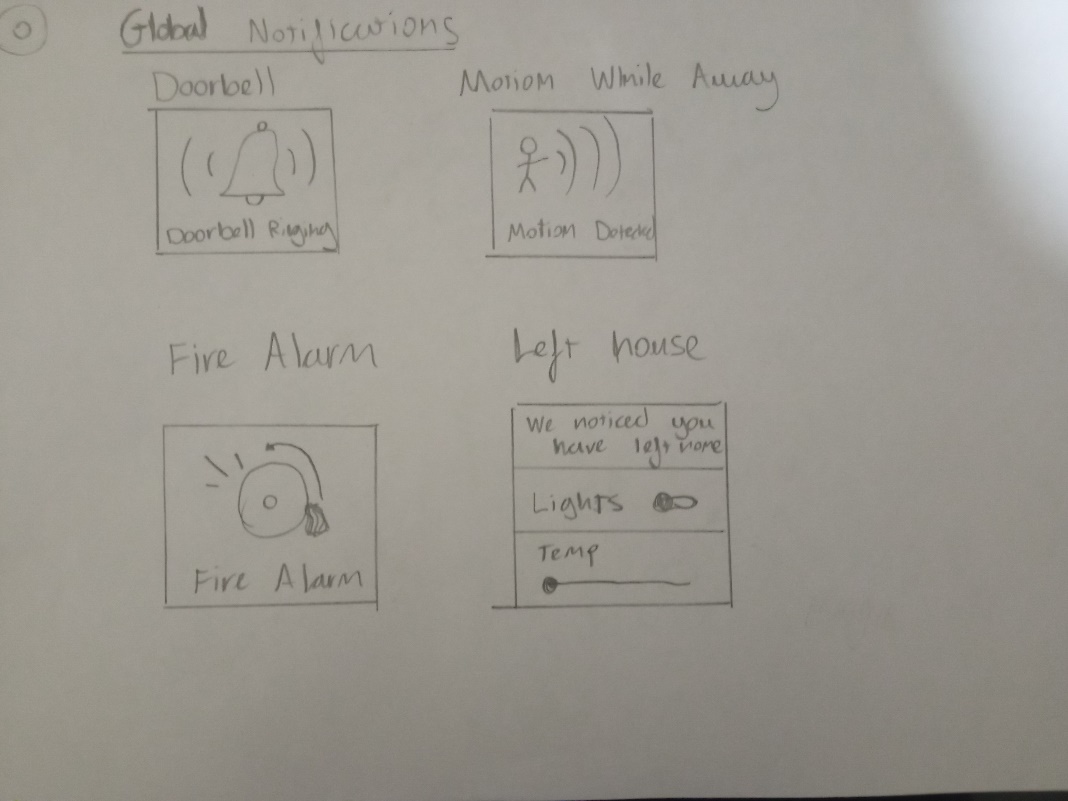


Figure 16: “Page 0 of Draft Wireframe”

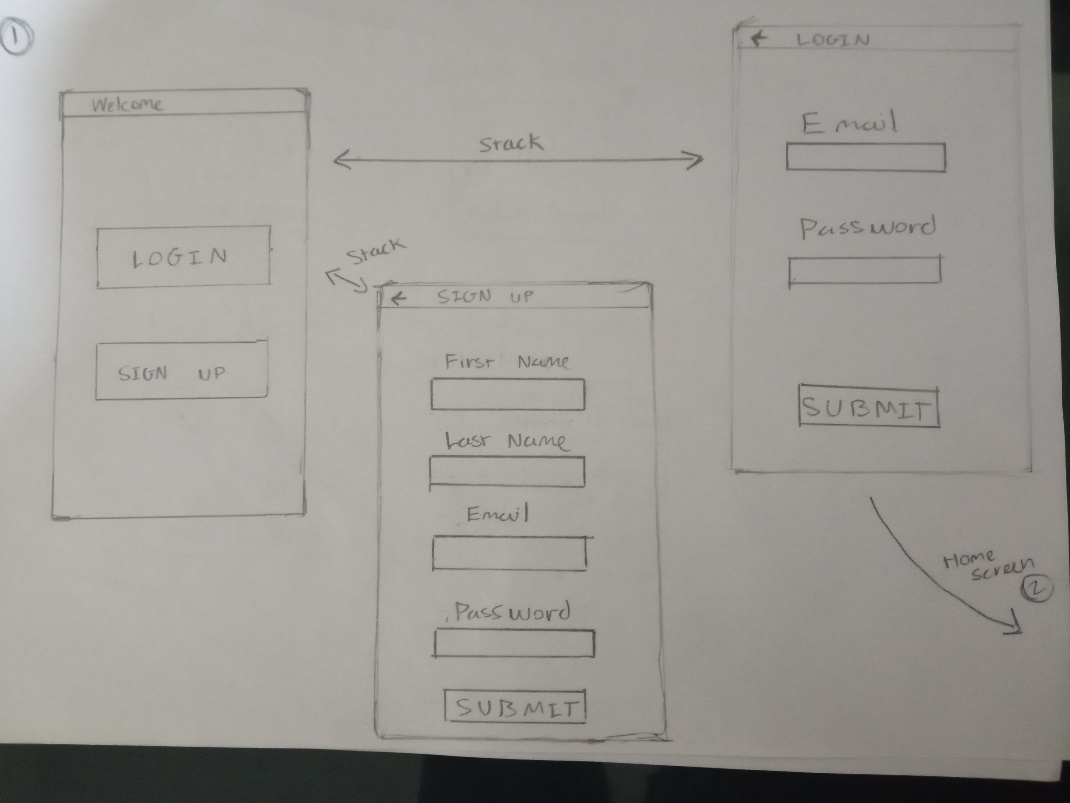


Figure 17: “Page 1 of Draft Wireframe”

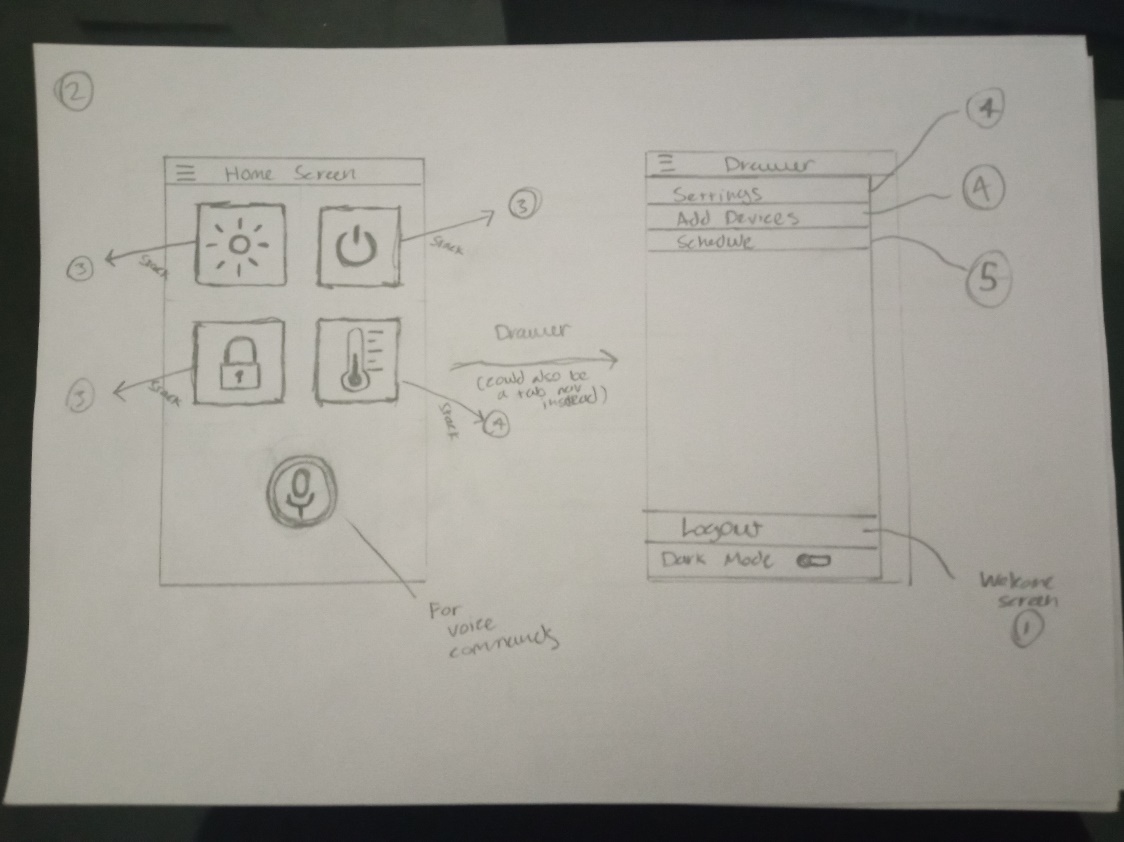


Figure 18: “Page 2 of Draft Wireframe”

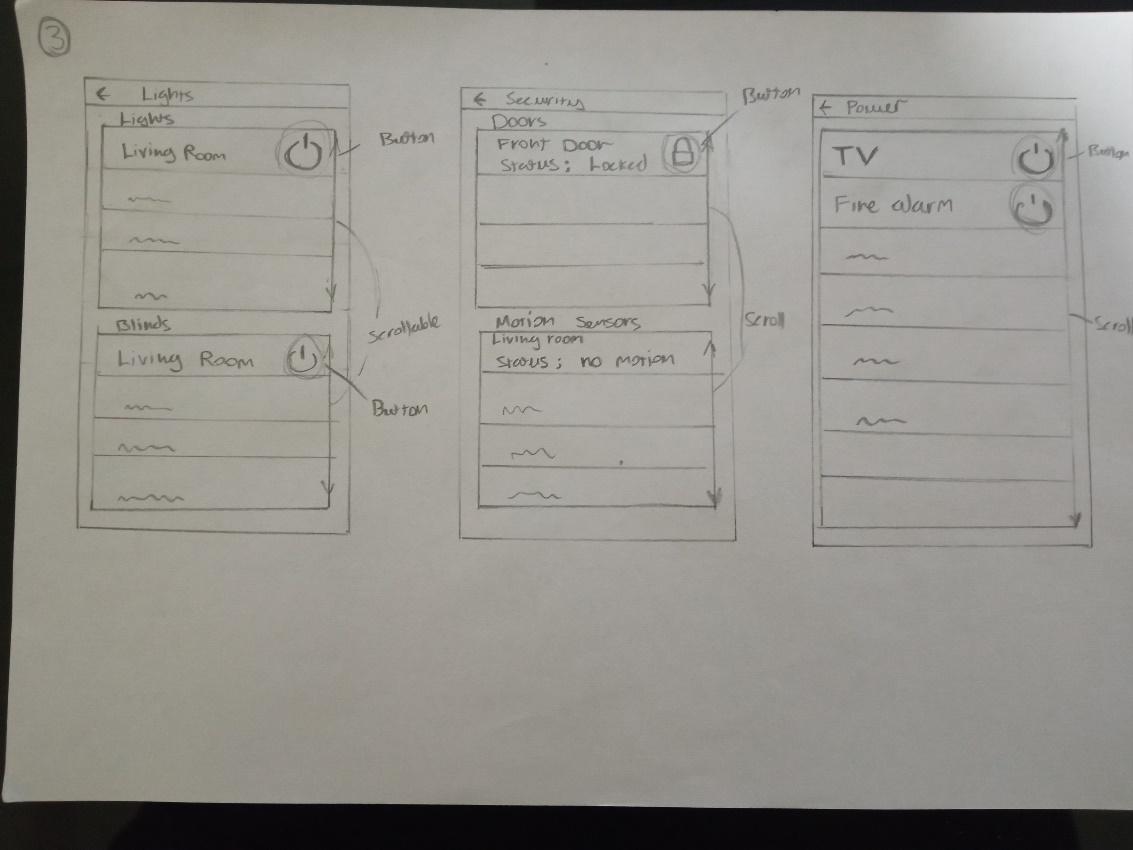


Figure 19: “Page 3 of Draft Wireframe”

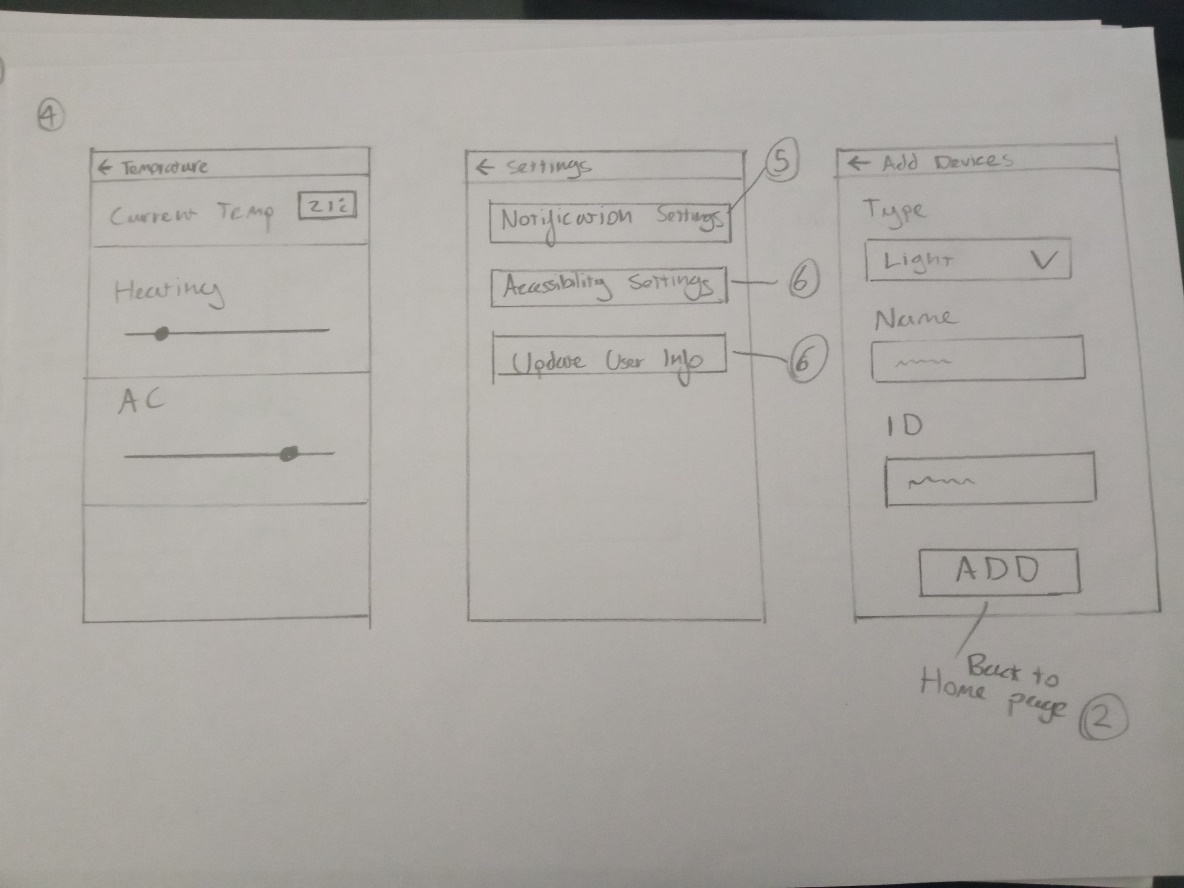


Figure 20: “Page 4 of Draft Wireframe”

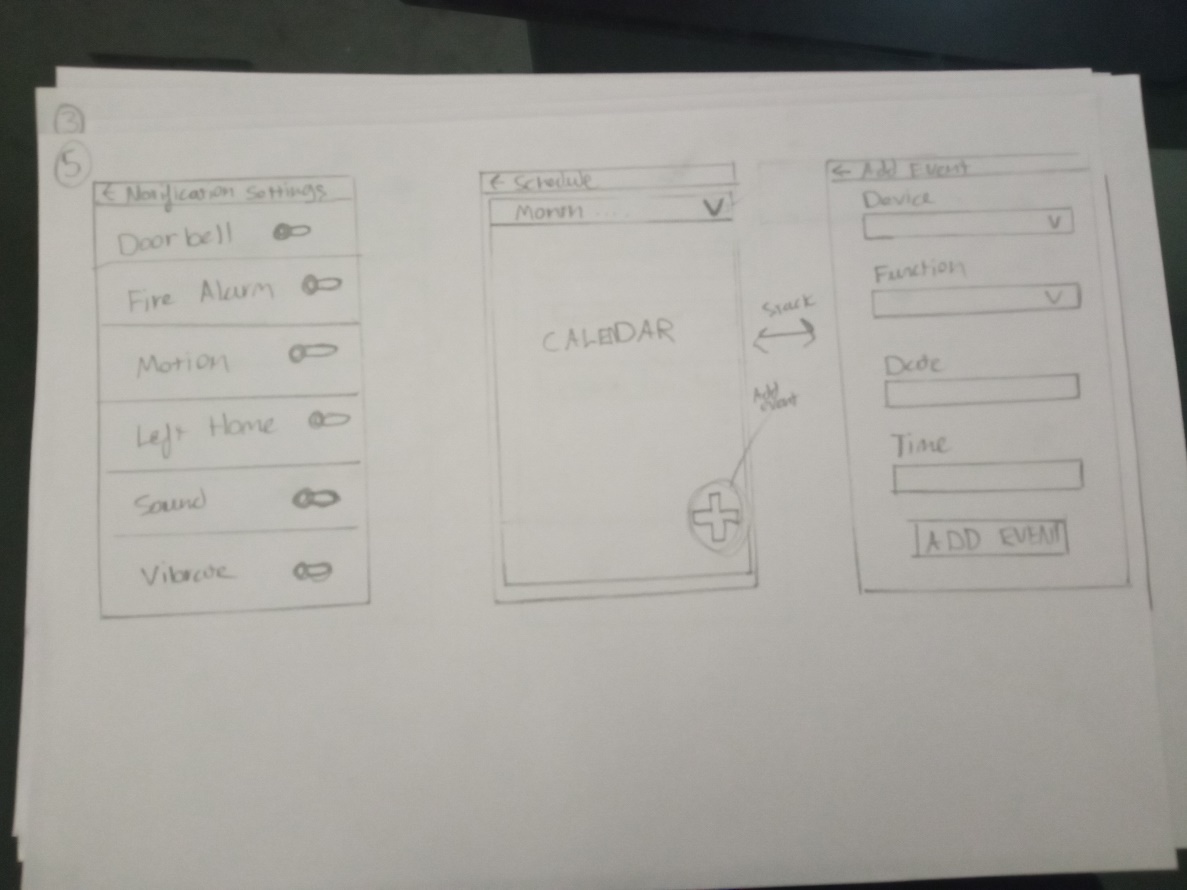


Figure 21: “Page 5 of Draft Wireframe”

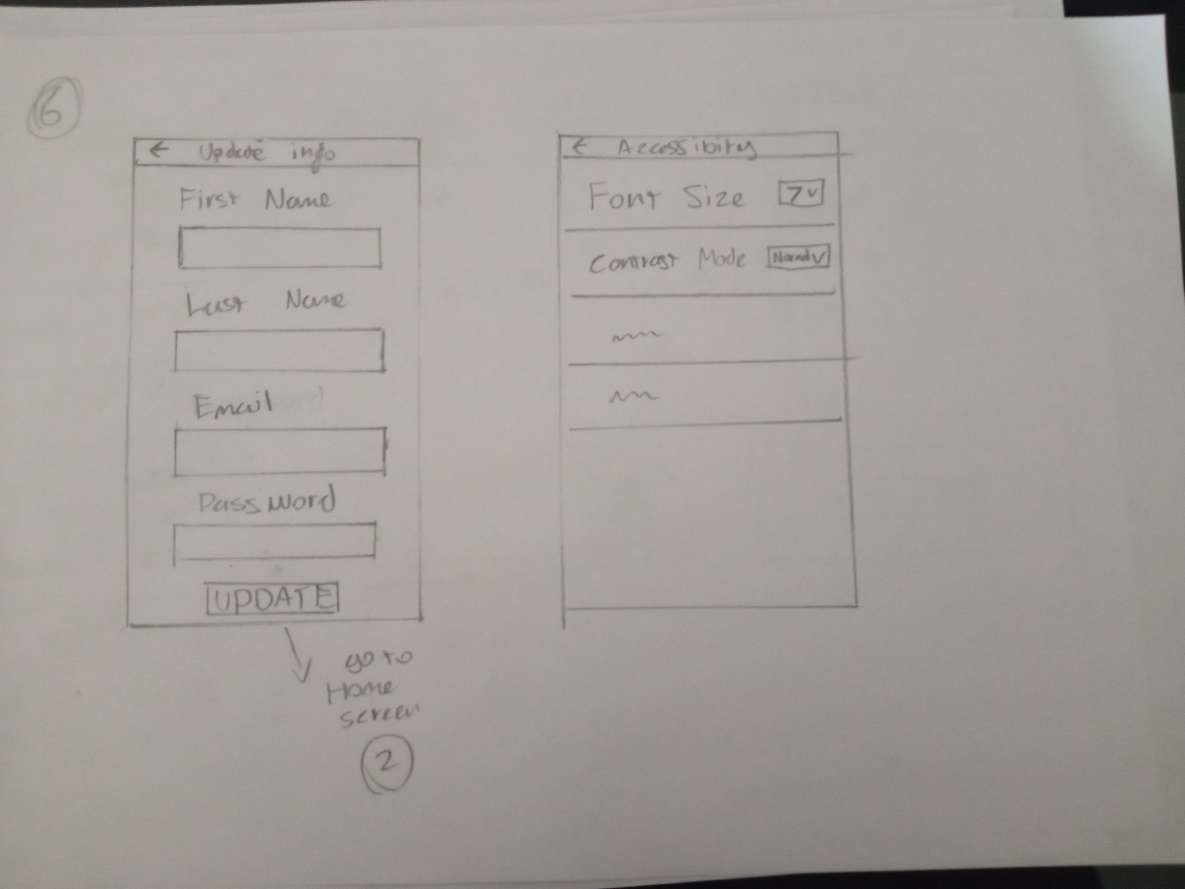


Figure 22: “Page 6 of Draft Wireframe”

### 3.4.5 Finalised Wireframe

Below is the finalised wireframe of the application, this wireframe is an improved version of the draft. This wireframe gave a clearer insight on which features weren’t necessary or would be too time consuming and it allowed for further refinement of the UI. Also, it provides valuable insight into how a user would interact with the interface and reduced time spent on implementation by providing the foundation of the applications design.

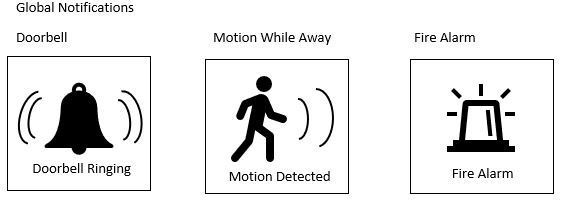


Figure 23: “Wireframe: Notifications”

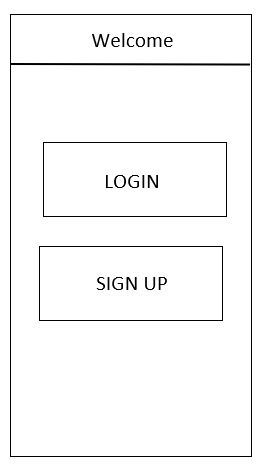


Figure 24: “Wireframe: Welcome Screen”

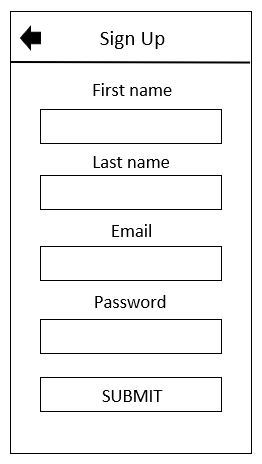


Figure 25: “Wireframe: Sign Up Screen”

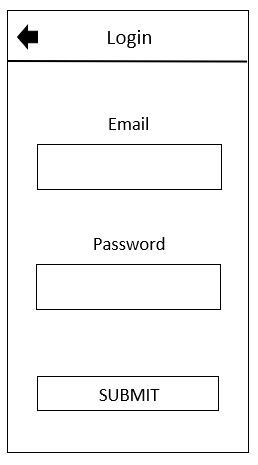


Figure 26: “Wireframe: Login Screen”

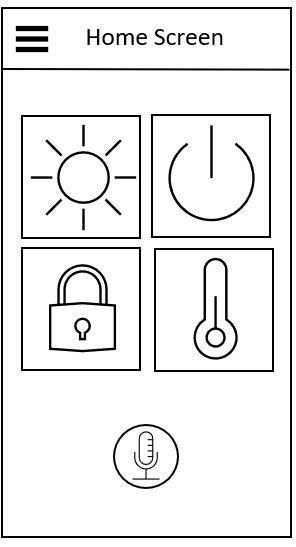


Figure 27: “Wireframe: Home Screen”

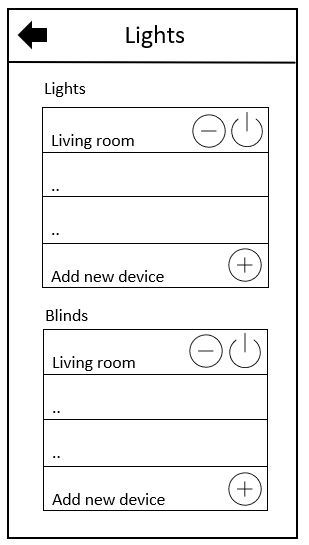


Figure 28: “Wireframe: Lights Screen”

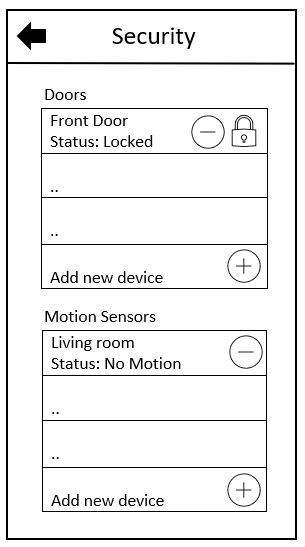


Figure 29: “Wireframe: Security Screen”

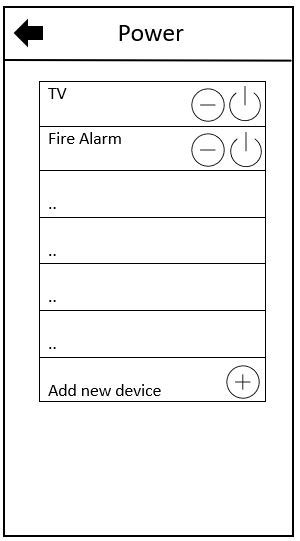


Figure 30: “Wireframe: Power Screen”

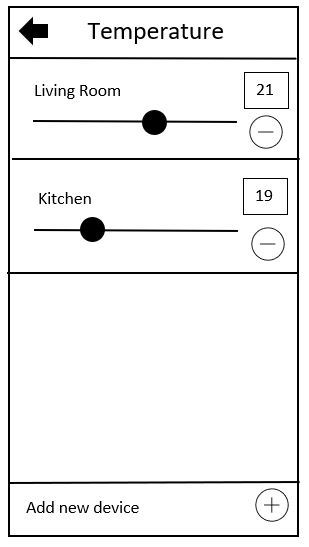


Figure 31: “Wireframe: Temperature Screen”

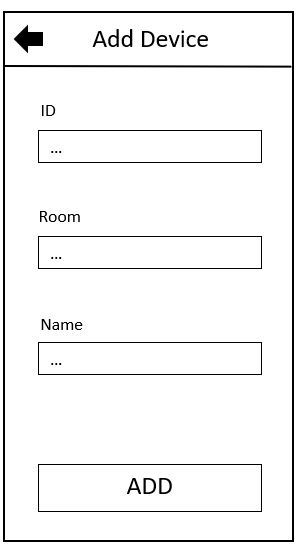


Figure 32: “Wireframe: Add Device Screen”

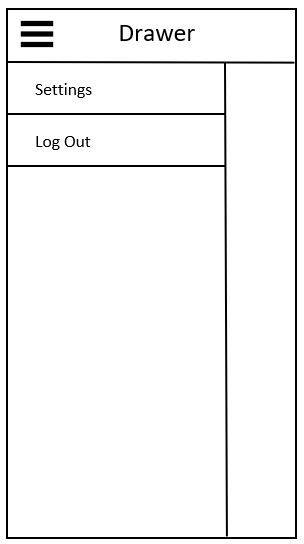


Figure 33: “Wireframe: Drawer”

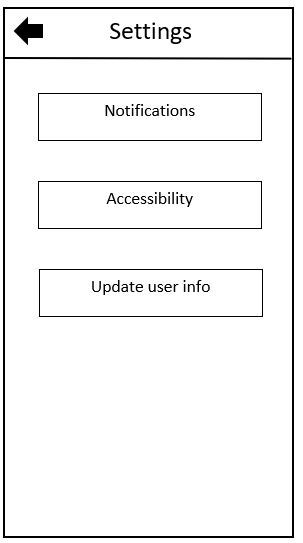


Figure 34: “Wireframe: Settings Screen”



Figure 35: “Wireframe: Notification Settings Screen”

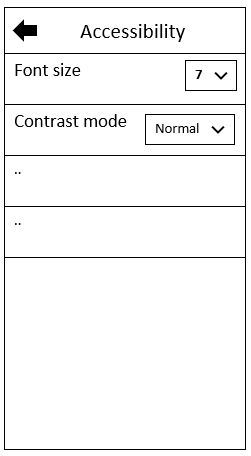


Figure 36: “Wireframe: Accessibility Settings Screen”

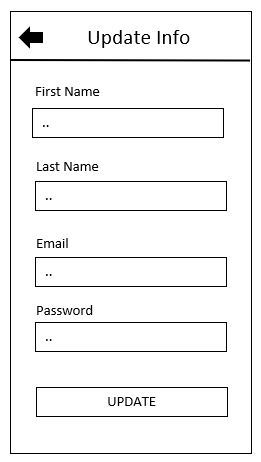


Figure 37: “Wireframe: Update User Info Screen”

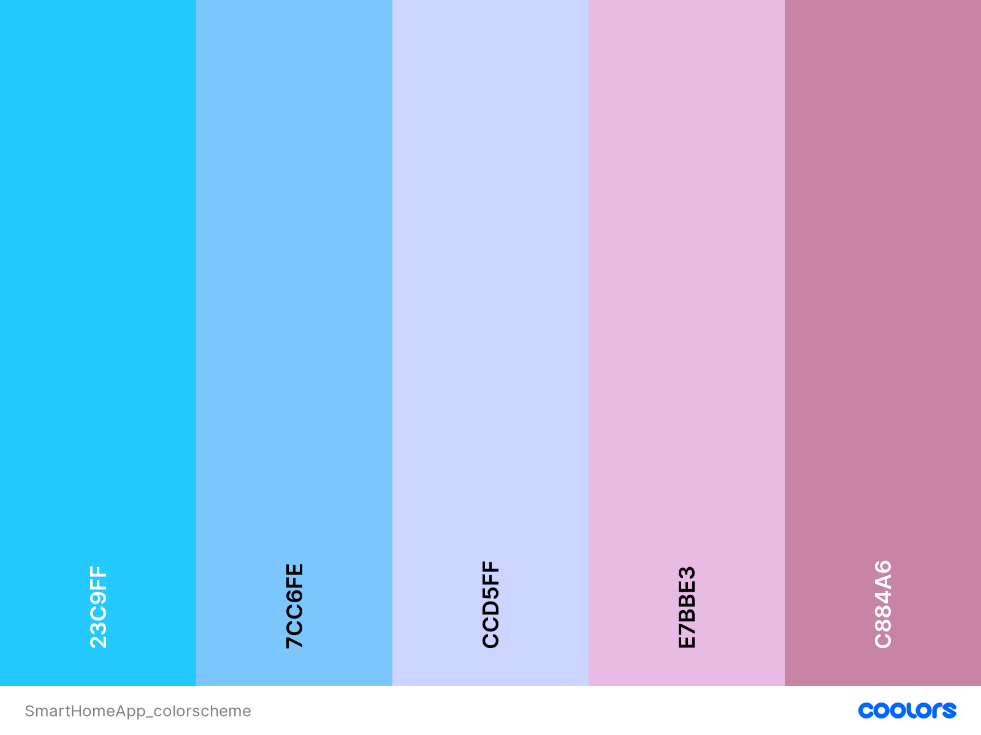


Figure 37.1: “Analogous Colour Scheme of Application”

### 3.4.6 Flowchart

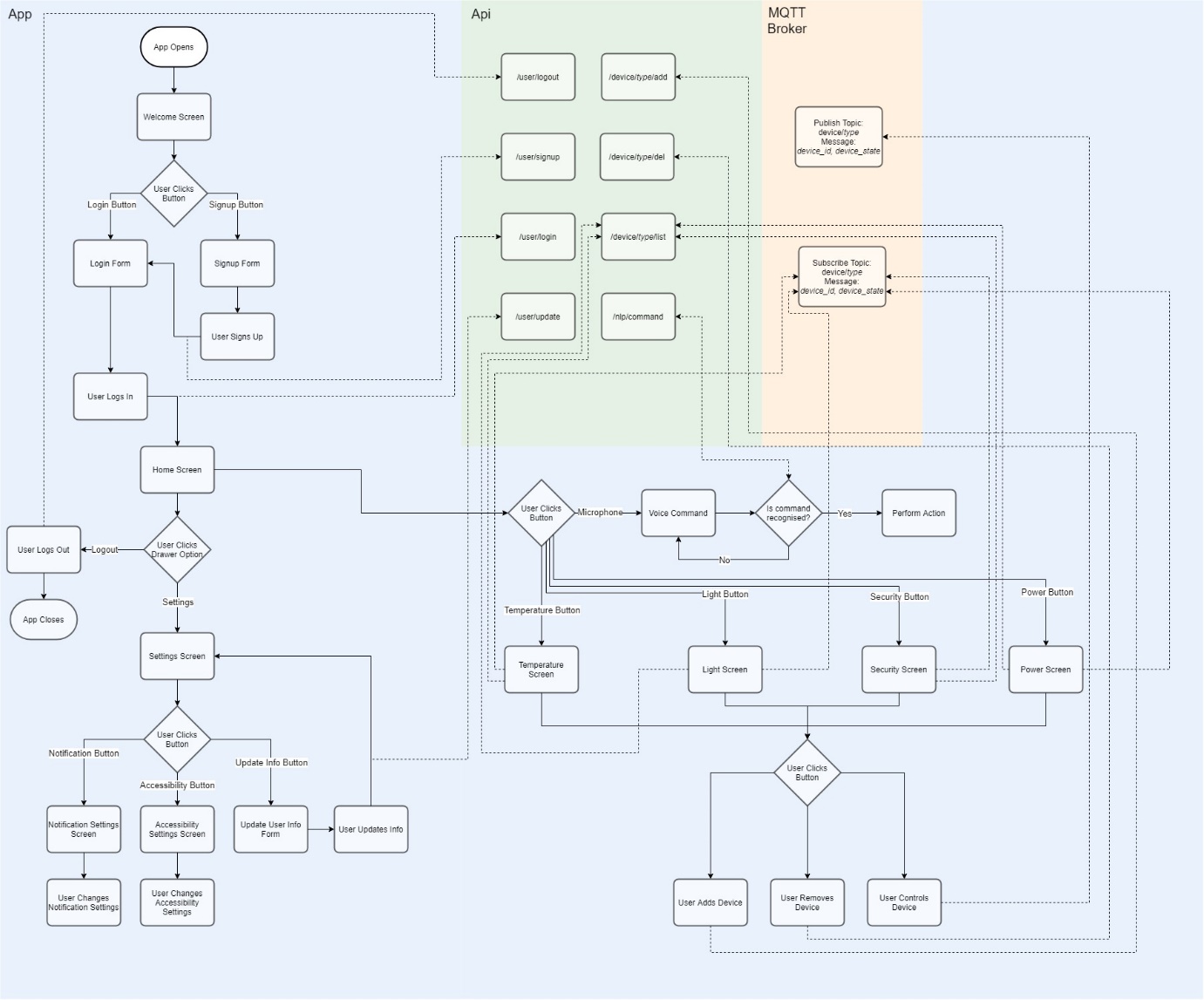


Figure 38: “Flowchart”

Figure 38 is a flowchart of the Android Application, it provides an overview of the entire app including all the points at which the API is called, the broker is subscribed too and when a message is published to the broker. The flowchart breaks down the logic of the system into easy to follow instructions and provides another point of view to the structure of the application, it turns the complexities of the application into a comprehensible visual aid that will be crucial in the implementation stage as it will save considerable amounts of time.

## 3.5 Computer Client Design

### 3.5.1 Language and Environment

The language chosen for the computer client is Java, the object-orientated nature of this language will prove incredibly useful when it comes to implementing this system as each Phidget device will require similar code to operate so the reusable code that objects provide is a perfect candidate, Java is also multithreaded which will be important when operating multiple Phidget devices at once. The development environment chosen was eclipse as it is standard for Java development, prettier was also used to format the code to ensure more readable code and higher standard of code quality.

### 3.5.2 Class Diagram

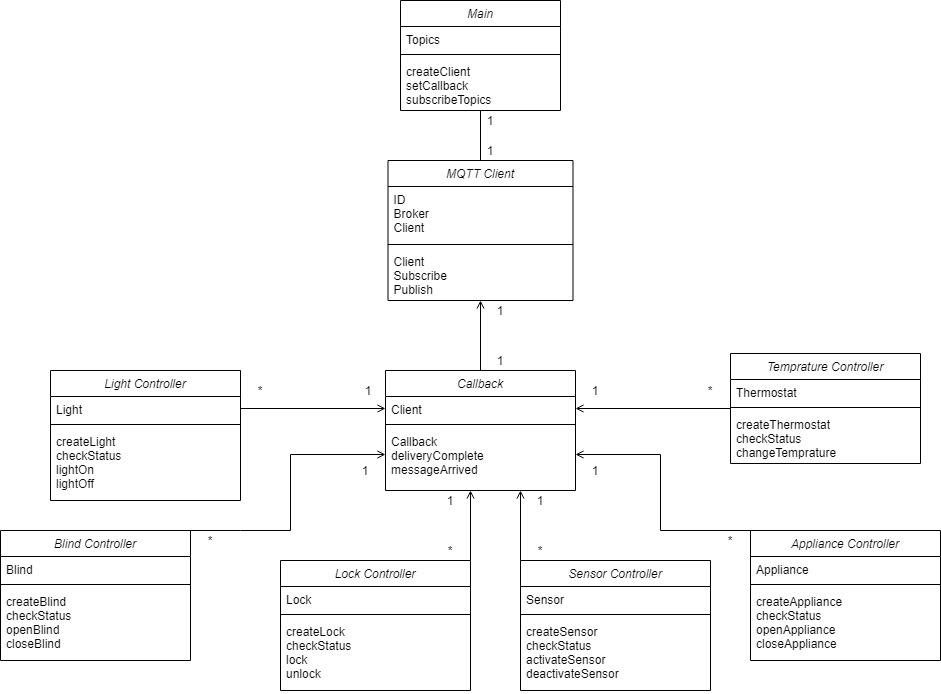


Figure 39: “Client Class Diagram”

Figure 39 shows the proposed classes and class relationships of the computer client, a rough outline of the properties and methods are also included for each class, the class diagram provides a visual model of the structure of the system. During implementation the class diagram can be used to easily translate the structure of the system into code.

## 3.6 API Design

### 3.6.1 Language and Environment

JavaScript was the chosen language for the API, more specifically NodeJS, a JavaScript runtime environment. NodeJS was chosen mainly for its high performance and its mellow learning curve. Another benefit of NodeJS it is extremely scalable meaning it will be easy to add features to the API in the future. The development environment chosen was again Visual Studio Code along with the prettier and ESLint plugins. Again, these plugins were both used to improve code quality also, to reduce errors. The style guide chosen was Airbnb. Found here: <https://github.com/airbnb/javascript>

### 3.6.2 Specification

When developing an API, creating an API specification is an extremely useful and versatile tool, it acts as design, documentation, and a way to standardize data exchange between web services within the system. The API specification created in this project was made using Swagger and was based off the CoffiDa API specification (Mainly the user management portion) created by Dr Ashley Williams (Williams, A. 2021) which provided a brilliant starting point, template and gave insight into the development of the API. The full specification created for this project can be found here: <https://app.swaggerhub.com/apis/Aubrey-Monk/SmartHomeApp/1.0.0> Below a few screenshots have also been included, the JSON of the full specification has also been included in the product submission inside the ‘SmartHomeAppAPI’ folder.

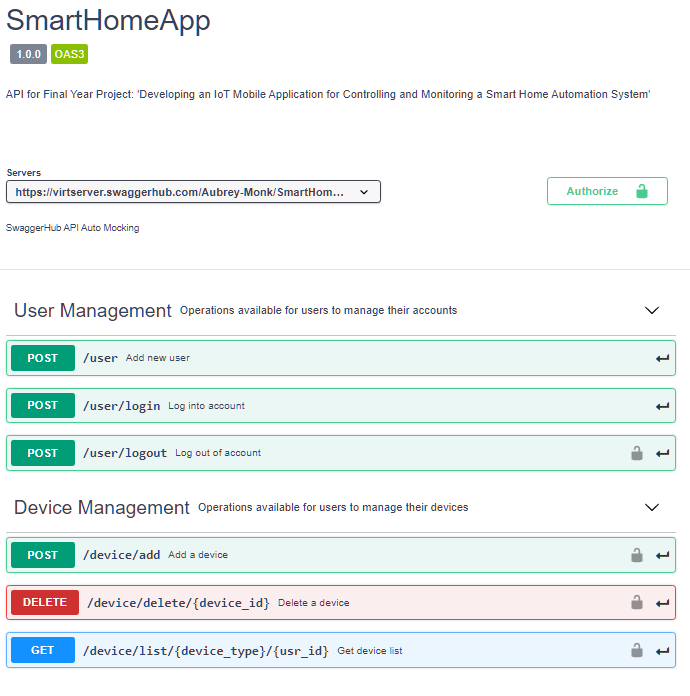


Figure 40: “API Specification”

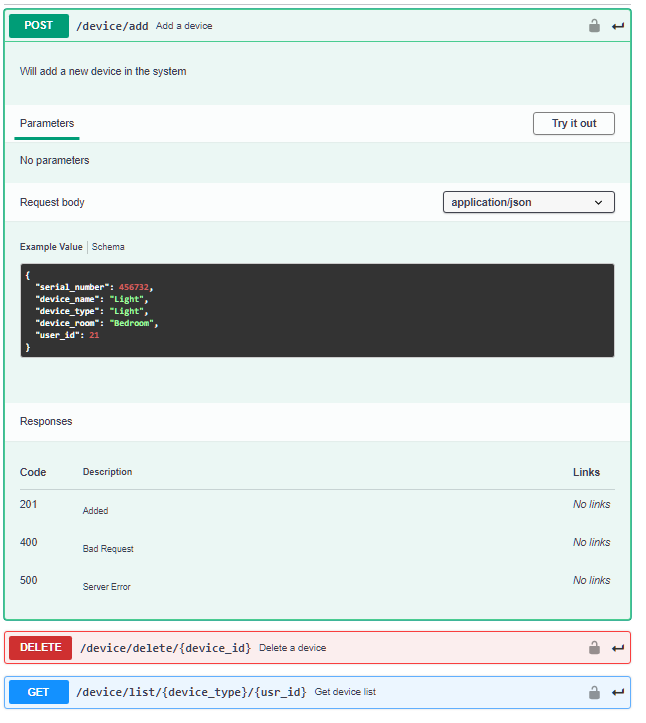


Figure 41: “API Specification: Add Device”

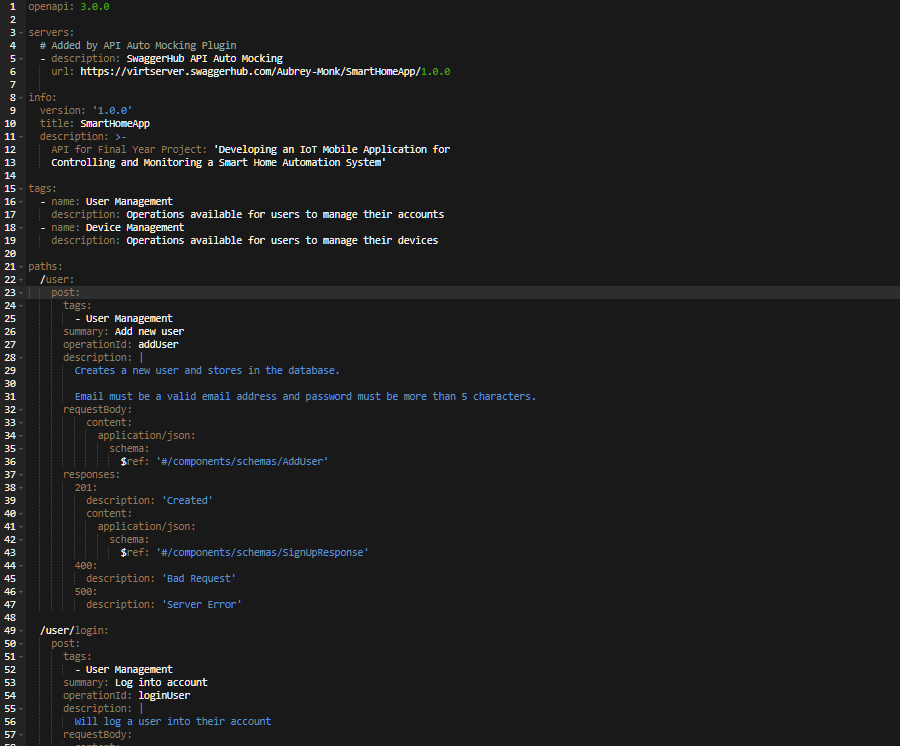


Figure 42: “API Specification: Code”

## 3.7 Database Design

### 3.7.1 Language and Environment

The database required for this project only needs to consist of 2 tables making it extremely simple, it will be written in SQL and hosted on the university’s MySQL “mudfoot” server. MySQL Workbench will be used to access the server and create the database tables.

### 3.7.2 Entity Relationship Diagram

Below are a few entity relationship diagrams created for the database system, they show the overall structure and relationship of the tables within the database system, this makes it very easy to implement the database.

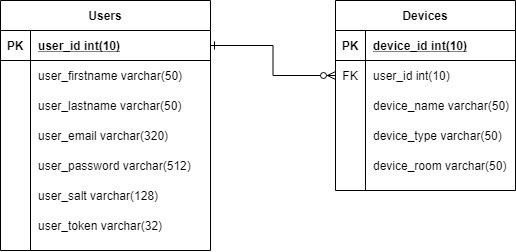


Figure 43: “First Database ERD”

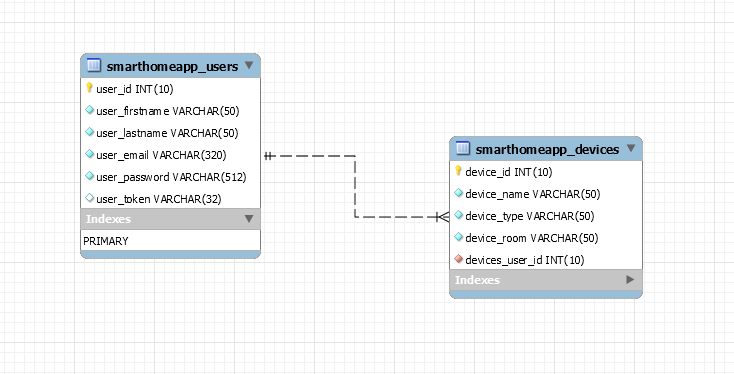


Figure 44: “Second Database ERD”

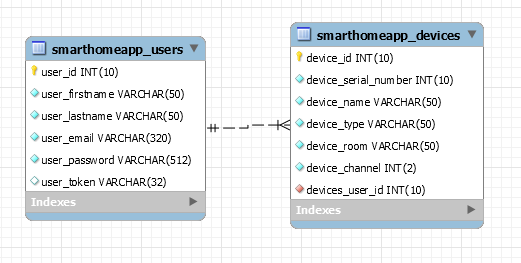


Figure 45: “Third Database ERD”

# 4 Implementation

This chapter of the report covers the implementation of the three main system components the API, the computer client, and the Android Application. These components are described in rough order of implementation.

## 4.1 API

The API is responsible for handling any requests the applications sends that requires use of the database, it is divided into two main sections, the user management section, and the device management section. Both sections include, a controller, model, and routes JS file. The *User Management* portion of the API was primarily based off the CoffiDa Server by Dr Ashley Williams (with their permission) (Williams, A. 2021). The API is only implemented to a working standard so the whole system can function, certain authentication and validation features were not included due to time constraints.

### 4.1.2 User Management

#### 4.1.2.1 User Management Routes

This portion of the code sets out the routes for all the endpoints of the user management section and the type of request (in this case just POST requests). Additionally, it links the correct method from the controller for each endpoint.

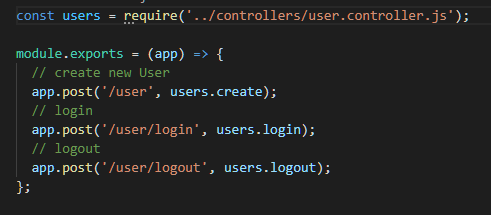


Figure 46: “API: User Routes”

#### 4.1.2.2 User Management Model

The User Model handles all database queries, below is an example of one of the methods from the from the model, the create method.

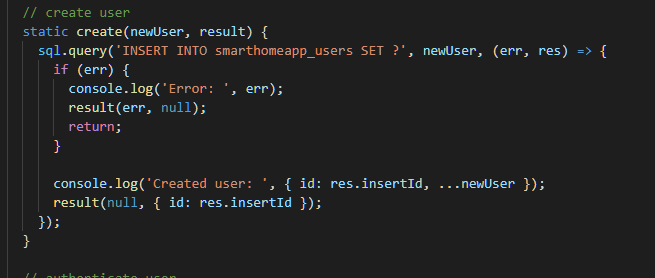


Figure 47: “API: User Model, Create method”

In Figure 47, a query is performed to insert the user into the database, if unsuccessful an error is returned, else if the query is successful the query results are returned.

#### 4.1.2.3 User Management Controller

The User Controller handles all User Management requests and returns the appropriate responses. Below is an example of one of the function from the controller, the create user function.

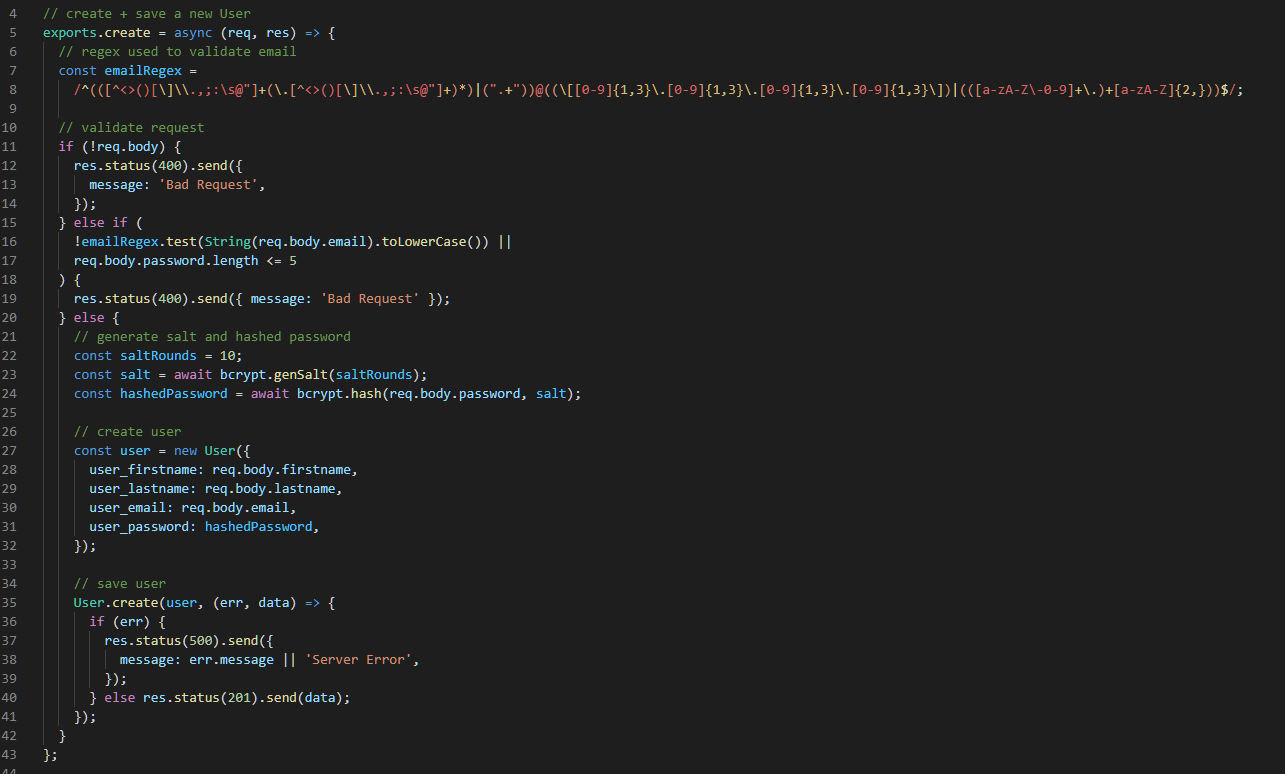


Figure 48: “API: User Controller, Create method”

In Figure 48, the request body is validated, if the request body is empty or invalid a ‘400 – Bad Request’ response is returned, if the request body is valid the password is hashed (for security so it is not stored as plaintext in the database), the user is created using the constructor from the user model, a user consists of: first name, last name, email and password fields. Then, the ‘create’ method from the User Model is called, if an error is returned from this method then a ‘500 – Server Error’ response is returned, if the user was successfully inserted into the database then the result is returned with the code 200.

### 4.1.3 Device Management

#### 4.1.3.1 Device Management Routes

This portion of the code sets out all the endpoints of the user management and their type of request (POST, GET and DELETE requests). Additionally, it links the correct method from the controller for each endpoint.

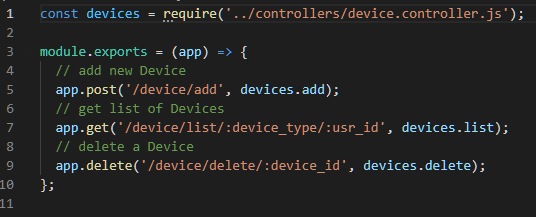


Figure 49: “API: Device Routes”

#### 4.1.3.2 Device Management Model

The Device Model handles all database queries, below is an example of one of the methods from the from the model, the list method.

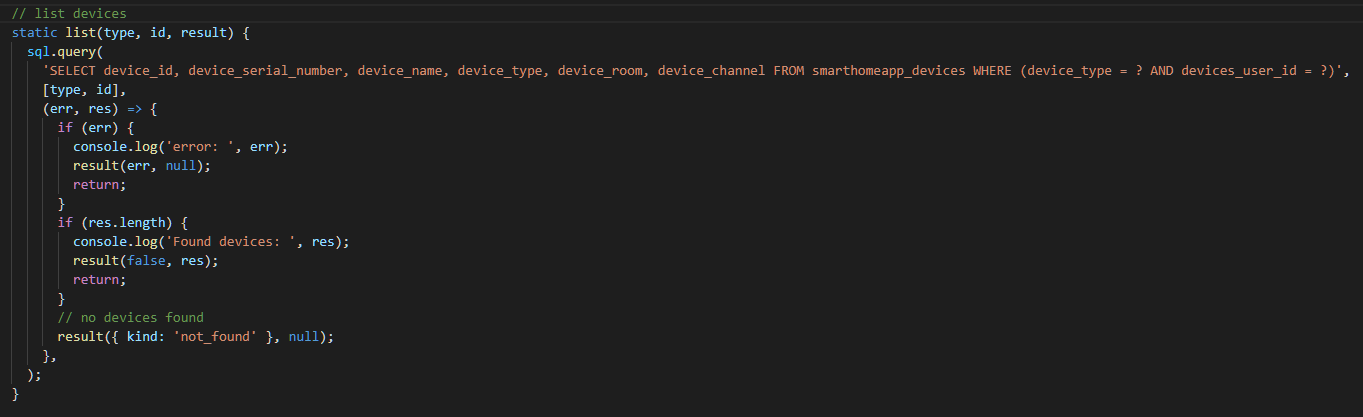


Figure 50: “API: Device Model, List method”

In Figure 50, a query on the is performed to select a specific device from the database given the *device\_type* and *device\_ID,* if any error occurs while preforming that query an error returns, if a device is found the device is returned and if no device is found *null* is returned.

#### 4.1.3.3 Device Management Controller

The Device Controller handles all Device Management requests and returns the appropriate responses. Below is an example of one of the functions from the controller, the list function.



Figure 51: “API: User Controller, Create method”

In Figure 51, the request body is checked, if the body is empty then a response of ‘400 – Bad Request’ is returned, if the body is not empty the ‘list’ class from the Device Model is called, passing a *device\_type* and *device\_ID.* If the method returns an error of type “not\_found” a response of ‘404 – No devices found’ is returned any other error returns a response of ‘500 – Server Error’. If the device is returned successfully then it sent as the response with the code 200.

## 4.2 Computer Client

The client runs on any computer that supports Java, Phidget devices first must be connected to the computer via USB then the client ran. The client’s job is to receive MQTT messages that are used to operate the Phidget devices and also send MQTT messages with the required status of any Phidget devices that the application needs.

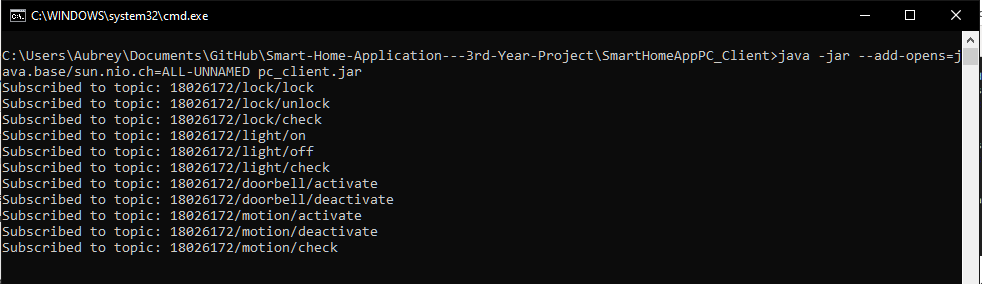


Figure 52: “Computer Client Running”

The Client is made up of seven different Java classes, this section will provide a brief overview of each class and give a few code examples.

### 4.2.1 Main Class

The Main or ‘SmartHomeAppPC\_Client’ class is the entry point of the code and must be ran first to start the client.

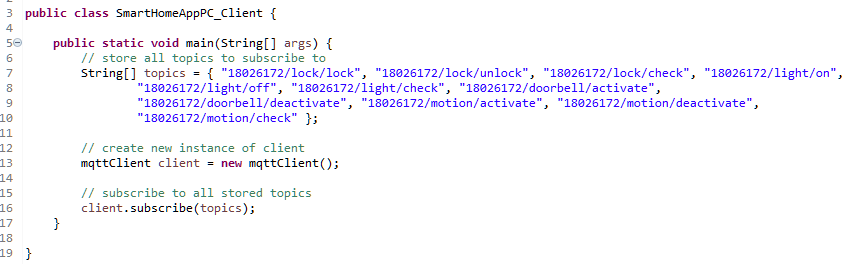


Figure 53: “Client: Main Class”

In Figure 53, the required MQTT topics are stored as an array, then an instance of the client is created and the subscribe method called passing all the required topics.

### 4.2.2 MQTT Client Class

The ‘mqttClient’ class utilizes the Eclipse Paho MQTT library, it is responsible for making a connection with the MQTT Broker, in this case a public MQTT Broker is used due to time constraints. The client class is also responsible for making subscriptions to the necessary topics.



Figure 54: “Client: MQTT Client”

In Figure 54, a UUID is generated for the client ID and the broker URL is stored as a String. The constructor class mqttClient() creates a new client object, sets the callback functions and makes a connection to the broker. There is also a subscribe method which iterates through each topic in the passed in topic array and makes a subscription to each one.

### 4.2.3 Callback Class

The callback function is used for handling any messages that arrive from the broker from the subscribed topics, first it outputs any received message to the console along with a timestamp then it checks the topic of the message and performs different functions depending on the topic. Below is an example of how it handles all the topics related to the locks.



Figure 55: “Client Callback: On Message Arrived”

Whenever a message is received that falls under the ‘check’ category, the message will contain all the serial numbers/channels of the device that’s state needs to be checked, these will be separated by a ‘-‘. The message is split and stored into a string array, the array is then iterated over and each device state is checked individually, a string of all the states separated by ‘-‘ is built and then a message containing this string is published to the broker the application then receives this message and uses the information to update the states of certain devices.

### 4.2.4 Device Controller Classes

There are 4 classes implemented for the control and monitoring of devices, these the lightController, lockController, motionSensorController and doorbellController classes.

These classes are responsible for executing the required code to control a device after a message has been received or returning the appropriate state information back to the callback function so a message can be published.

In each of these classes, the instances of the devices are stored in an Array List, excluding the doorbell as only one doorbell can be used at a time. Whenever a device needs to be operated the list if first checked to see if the device already exists, if the device does not exist a new instance is created and added to the list.



Figure 56: “Motion Sensor Controller: Activate Sensor”



Figure 57: “Motion Sensor Controller: Check if Sensor Exists”

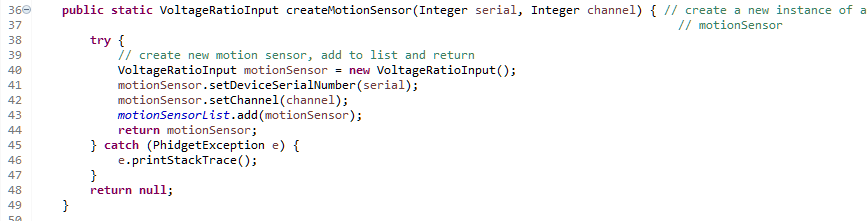


Figure 58: “Motion Sensor Controller: Create Sensor”

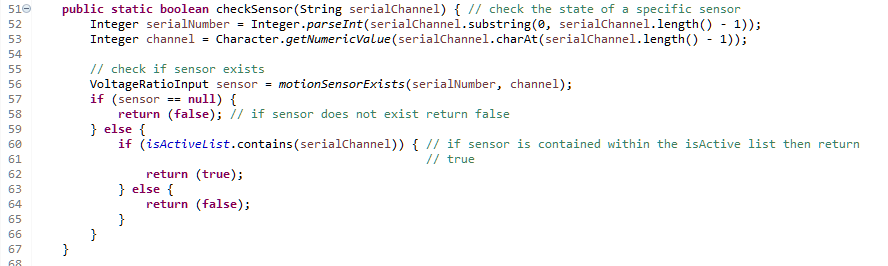


Figure 59: “Motion Sensor Controller: Check Sensor”

The function ‘checkSensor’ in Figure 59 is responsible to checking the current state of a specific motion sensor (active or inactive) and returning that so a message can be published containing states of the sensors.

## 4.3 Android Application

The Android application was the main focus of this project. The purpose of the Android application is to provide a user interface for controlling and monitoring Phidget devices, these Phidget devices represent devices that could be used in a smart home system.

### 4.3.1 Phidget Devices

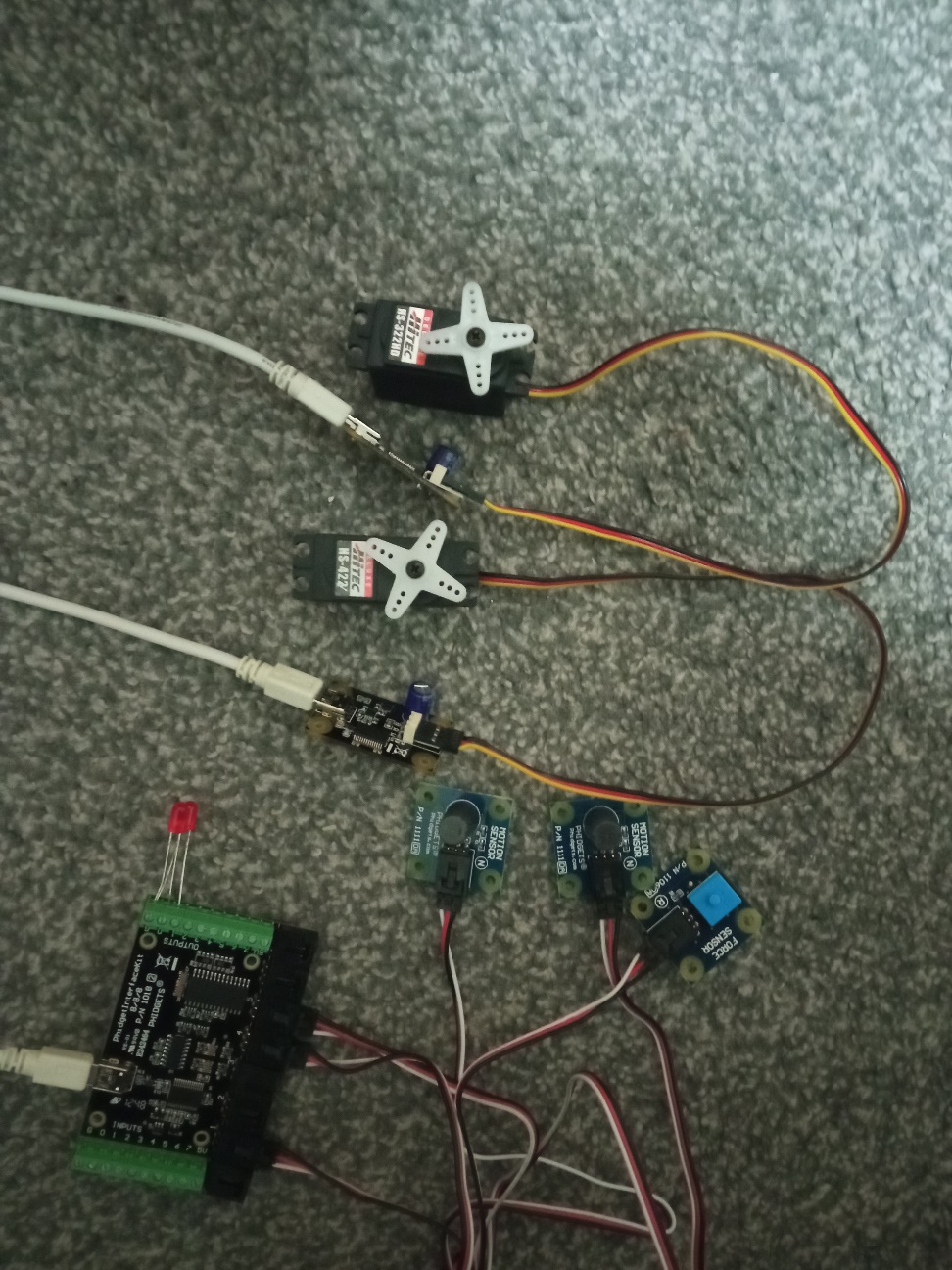


Figure 60: “Phidget Devices Used for Testing and Implementation”

Figure 60 Shows the device set up used for this project during the implementation and testing phase.

* A servo motor attached to a Phidget Advanced Servo was used as a lock.
* A light emitting diode (LED) attached to a Phidget Interface Kit was used as a light.
* A motion sensor attached to the Phidget Interface Kit was used for detecting motion.
* A force sensor attached to the Phidget Interface Kit was used as a doorbell.

### 4.3.2 Screens and Components

The application itself consists of two notable parts, the screens, and the components. The screens provide the user interface and handle interactions with the interface. The components are mainly responsible for sending request to the API and returning the responses. Below are a few examples and explanations of how they work.

#### 4.3.2.1 Login Screen

The login screen takes the user email and password as text inputs, validates them and then calls the login component to send the request to the API, if the request returns with a success the user is navigated to the Home Screen.

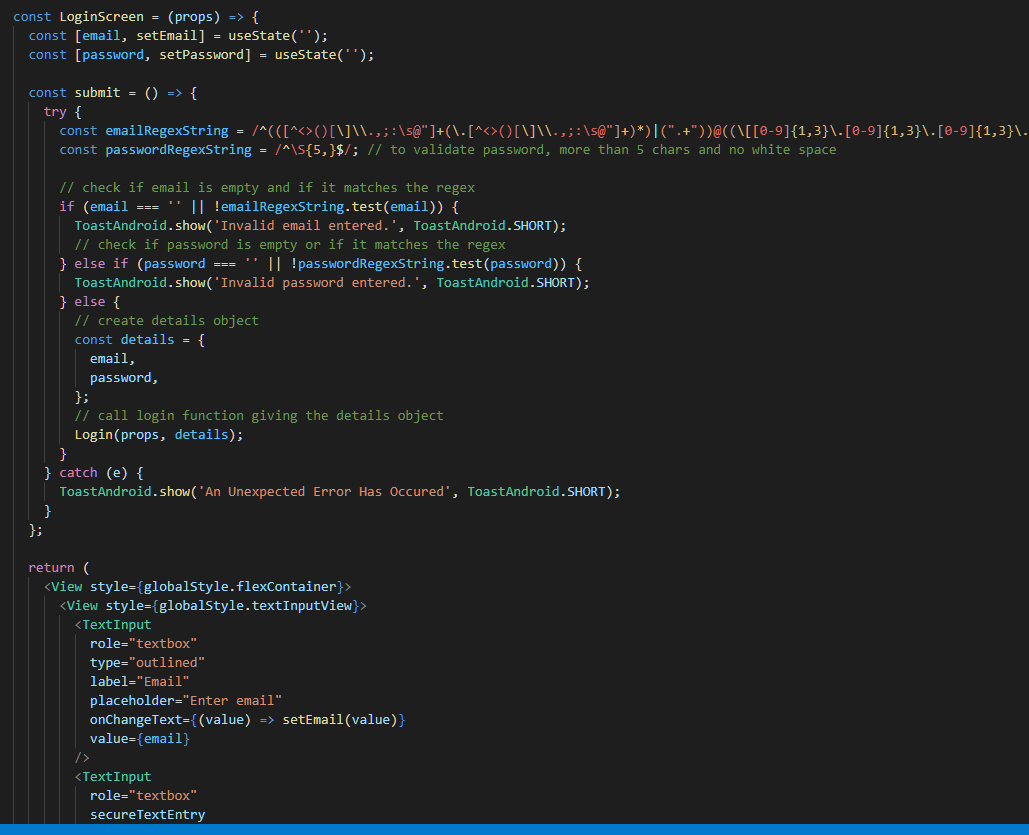


Figure 61: “Application: Login Screen”

#### 4.3.2.2 Login Component

The Login Component sends a POST request to the API with a user’s login details and then handles the responses appropriately. If successful, the response is returned and the users session token and ID are store within async storage on the device.



Figure 62: “Application: Login Component”

#### 4.3.2.3 Locks Screen

The Locks Screen handles user interaction with any lock devices. A user can add any amount of new locks from this screen by clicking the add device button, they are then taken to the add device screen which handles this interaction. Upon load of this screen the List Devices Component is called, if it returns successfully all the locks statuses are then checked by publishing there serial number to the MQTT Broker, the client handles this then returns the state of the locks and the interface is updated. If a user clicks on the padlock icon depending on the state of the lock a message is published to the MQTT broker which is again handled by the client and the lock is rotated accordingly. A user is also able to click the delete button which call the Delete Device Component and the device is then removed from the interface and the database.

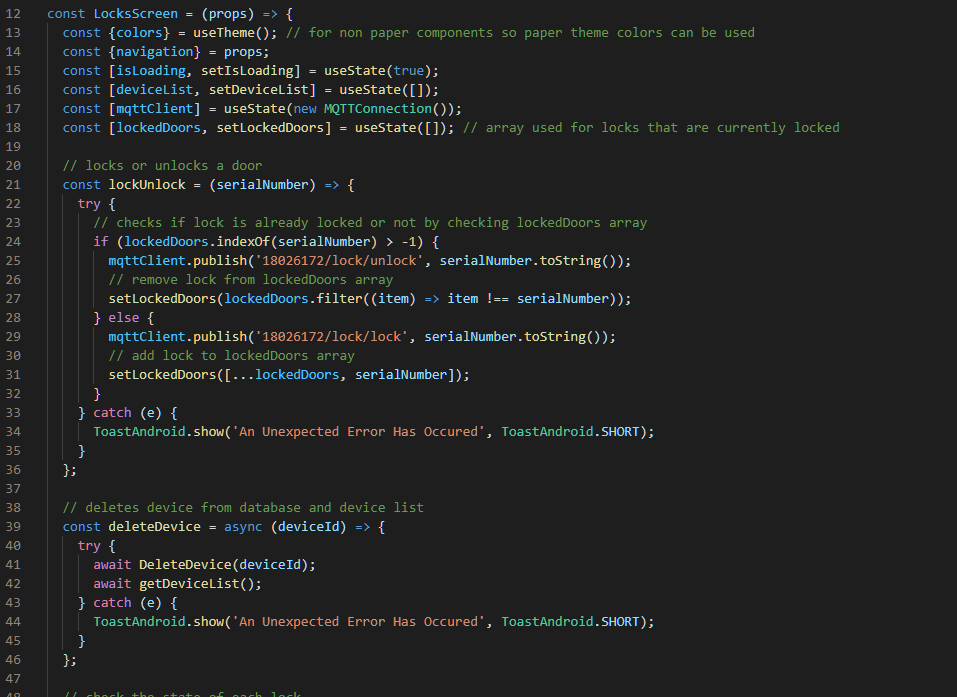


Figure 63: “Locks Screen: Lock/Unlock Function”

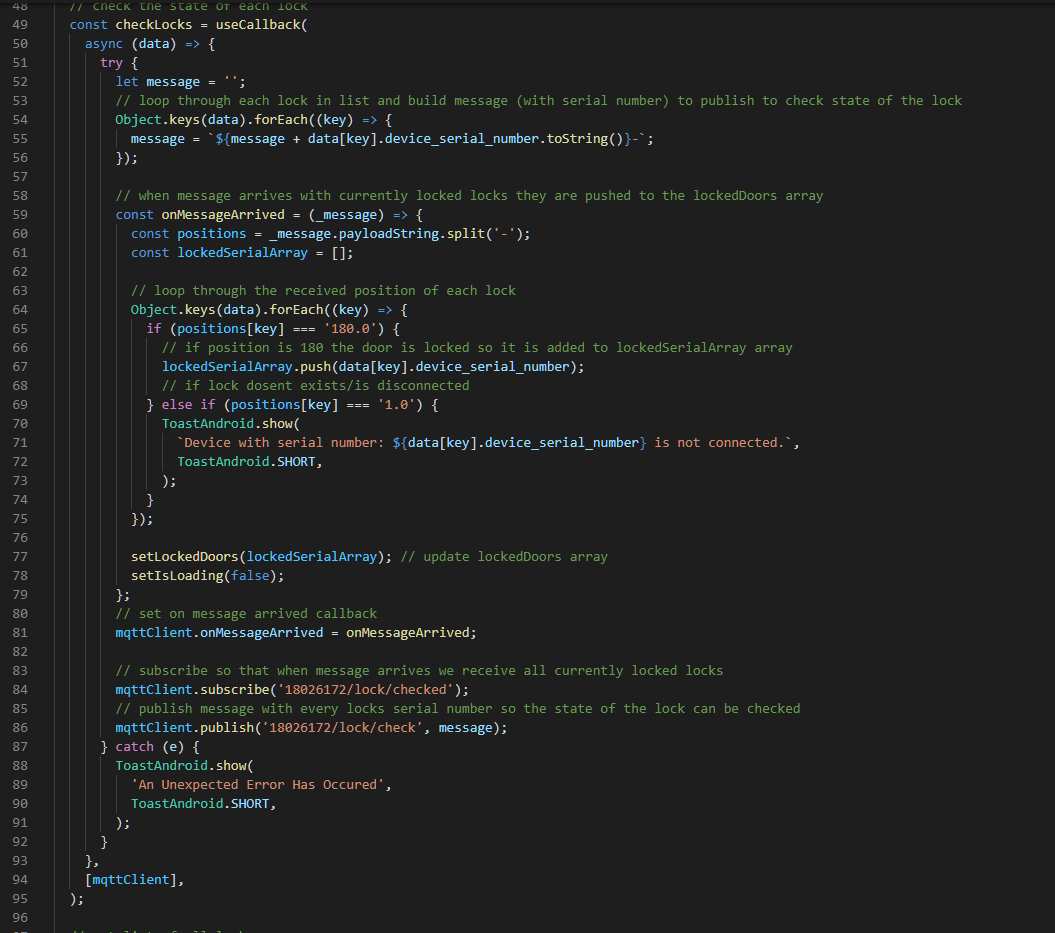


Figure 64: “Locks Screen: Check Lock Status Function”

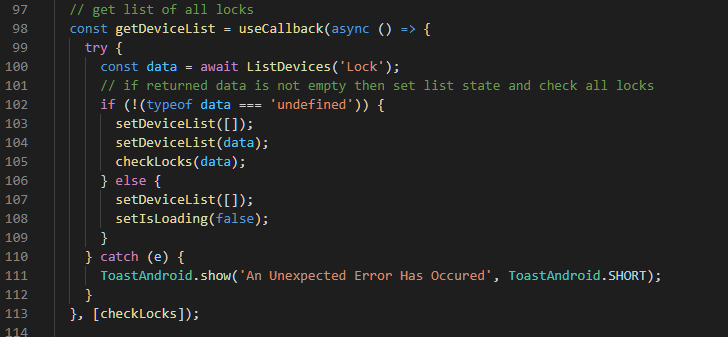


Figure 65: “Locks Screen: Get Device List Function”

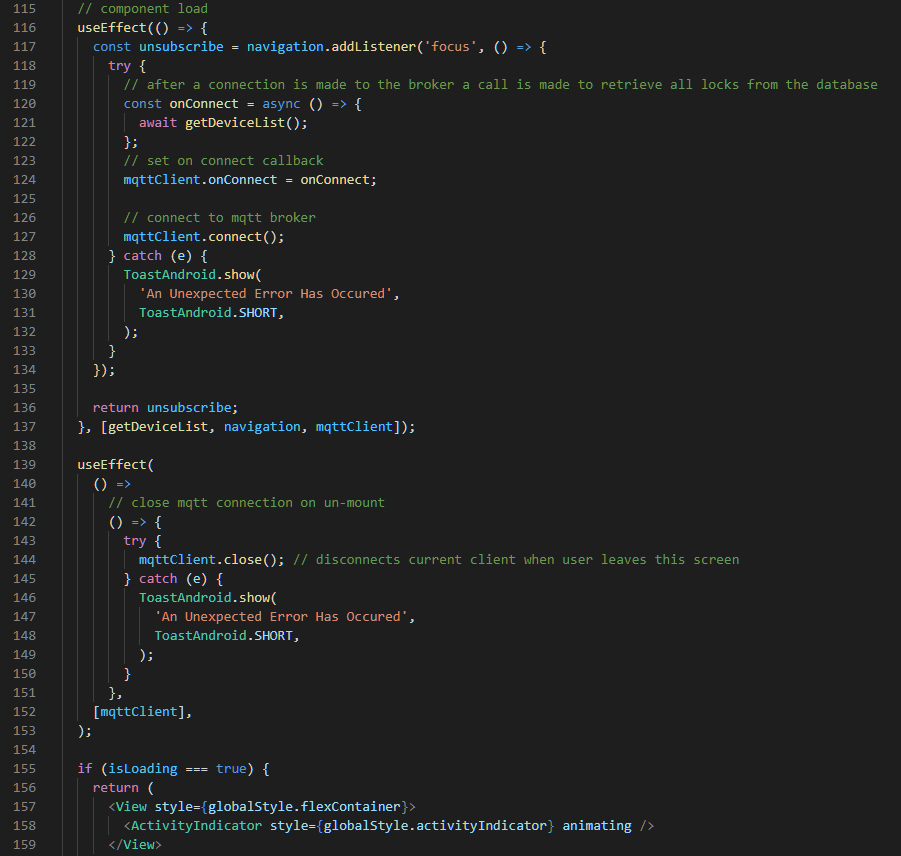


Figure 66: “Locks Screen: On Component Load”

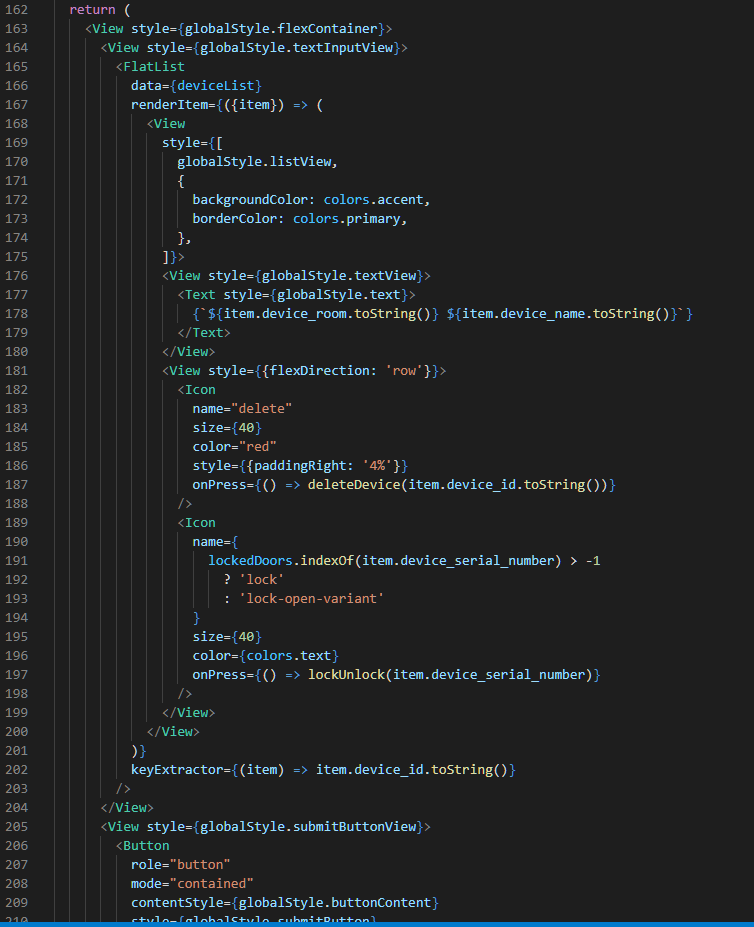


Figure 67: “Locks Screen: User Interface”

#### 4.3.2.4 List Devices Component

The List Devices Component sends a GET request to the API with the requested devices type and the Users ID and then handles the responses appropriately. If successful, the device list is returned.

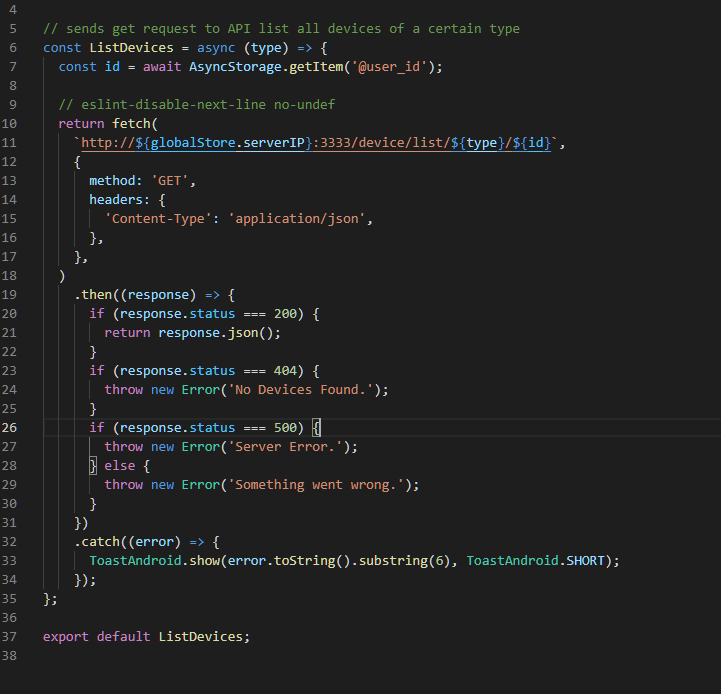


Figure 68: “Application: List Devices Component”

### 4.3.3 Testing

This section provides an analysis of which requirements were achieved and also discusses the testing strategy of the application.

#### 4.3.3.1 Analysis of Requirements

Figure 14 outlined the requirements of the application, below is a color-coded list of which requirements were achieved, partially achieved or not at all achieved by the project.

Core Requirements

* API
* SQL Database
* Add/Remove Phidget Devices
* MQTT Broker
* Voice Recognition
* Natural Language Processing
* Signup
* Login
* Logout
* Control Lights
* Control/Monitor Temperature
* Lock/Unlock Doors
* Control Blinds
* Monitor Motion Sensors
* Monitor Doorbell

Sacrificial Requirements

* Accessibility Features
* Accessibility Settings
* Notification Settings
* Update User Info
* Monitor Fire Alarm
* Control Power to Sockets

Out of Scope Requirements

* Is User Home
* Schedule

To conclude 73% of the Core Requirements were implemented. However, 0% of the Sacrificial and Out of Scope requirements were implemented. Accessibility features were only partially implemented with the inclusion of a dark mode setting.

#### 4.3.3.2 Testing Implemented Requirements

This section covers the testing of the Add/Remove Phidget Devices, Signup, Login, Logout, Control Lights, Lock/Unlock Doors, Monitor Motion Sensors and Monitor Doorbell requirements. For these, a requirements traceability matrix along with multiple test cases will be used for overall testing of the system.

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Requirement Description | Testcase ID | Status |
| 1 | Signup | 1,2,3,4,5,6,7,8,9 | 1 - Pass 2 - Pass 3 - Pass 4 - Pass 5 - Pass 6 - Fail 7 - Fail 8 - Pass 9 - Pass |
| 2 | Login | 10,11 | 10 - Pass 11 - Pass |
| 3 | Logout | 12 | 12 - Pass |
| 4 | Add light | 13,14,15,16,17,18,19,20,21,22 | 13 - Pass 14 - Pass 15 - Pass 16 - Pass 17 - Pass 18 - Pass 19 - Pass 20 - Pass 21 - Pass 22 - Pass |
| 5 | Turn On Light | 23, 24 | 23 - Pass 24 - Fail |
| 6 | Turn Off Light | 25, 26 | 25 - Pass 26 - Fail |
| 7 | Remove Light | 27 | 27 - Pass |
| 8 | Add Lock | 28,29,30,31,32,33,34,35,36,37 | 28 - Pass 29 - Pass 30 - Pass 31 - Pass 32 - Pass 33 - Pass 34 - Pass 35 - Pass 36 - Pass 37 - Pass |
| 9 | Lock | 38, 39 | 38 - Pass 39 - Fail |
| 10 | Unlock | 40, 41 | 40 - Pass 41 - Fail |
| 11 | Remove Lock | 42 | 42 - Pass |
| 12 | Add Sensor | 43,44,45,46,47,48,49,50,51,52 | 43 - Pass 44 - Pass 45 - Pass 46 - Pass 47 - Pass 48 - Pass 49 - Pass 50 - Pass 51 - Pass 52 - Pass |
| 13 | Activate Sensor | 53,54 | 53 - Pass 54 - Fail |
| 14 | Test Sensor | 55,56 | 55 - Pass 56 - Pass |
| 15 | Deactivate Sensor | 57,58 | 57 - Pass 58 - Fail |
| 16 | Remove Sensor | 59 | 59 - Pass |
| 17 | Add Doorbell | 60,61,62,63,64,65,66,67,68,69 | 60 - Pass 61 - Pass 62 - Pass 63 - Pass 64 - Pass 65 - Pass 66 - Pass 67 - Pass 68 - Pass 69 - Pass |
| 18 | Activate Doorbell | 70,71 | 70 - Pass 71 - Fail |
| 19 | Test Doorbell | 72,73 | 72 - Pass 73 - Pass |
| 20 | Deactivate Doorbell | 74, 75 | 74 - Pass 75 - Pass |
| 21 | Remove Doorbell | 76 | 76 - Pass |

Table 1: “Requirements Traceability Matrix”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TestCase # | Test Case | Test Steps | Test Data | Expected Result |
| 1 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click Signup | firstname="John" lastname="Smith" email="test@email.com" password="password123" | Signup successful |
| 2 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click Signup | firstname=" " lastname="Smith" email="test@email.com" password="password123" | Signup unsuccessful |
| 3 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname=" " email="test@email.com" password="password123" | Signup unsuccessful |
| 4 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname="Smith" email=" " password="password123" | Signup unsuccessful |
| 5 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname="Smith" email="test@email.com" password=" " | Signup unsuccessful |
| 6 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="123" lastname="Smith" email="test@email.com" password="password123" | Signup unsuccessful |
| 7 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname="123" email="test@email.com" password="password123" | Signup unsuccessful |
| 8 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname="Smith" email="123" password="password123" | Signup unsuccessful |
| 9 | Verify Signup | 1) Got to signup page 2) Enter firstname 3) Enter lastname 4) Enter email 5) Enter password 6) Click signup | firstname="John" lastname="Smith" email="test@email.com" password="123" | Signup unsuccessful |
| 10 | Verify Login | 1) Got to login page 2) Enter email 3) Enter password 4) Click login | email="test@email.com" password="password123" | Login successful |
| 11 | Verify Login | 1) Got to login page 2) Enter email 3) Enter password 4) Click login | email="" password="" | Login unsuccessful |
| 12 | Verify Logout | 1) Go to home screen 2) Open drawer 3) Click logout |  | Logout successful |
| 13 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Light" deviceroom="Bedroom" | Device added successfully |
| 14 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber=" " channel="0" devicename="Light" deviceroom="Bedroom" | Device not added |
| 15 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel=" " devicename="Light" deviceroom="Bedroom" | Device not added |
| 16 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename=" " deviceroom="Bedroom" | Device not added |
| 17 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Light" deviceroom=" " | Device not added |
| 18 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="123" channel="0" devicename="Light" deviceroom="Bedroom" | Device not added |
| 19 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="123" devicename="Light" deviceroom="Bedroom" | Device not added |
| 20 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="123" deviceroom="Bedroom" | Device added successfully |
| 21 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Light" deviceroom="123" | Device added successfully |
| 22 | Verify Adding Light | 1) Go to light screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="1234567" channel="0" devicename="Light" deviceroom="Bedroom" | Device not added |
| 23 | Turn On a Connected Light | 1) Go to light screen 2) Click light icon to turn on light |  | Light turned on |
| 24 | Turn On an Un-Connected Light | 1) Go to light screen 2) Click light icon to turn on light |  | User notified of disconnected light |
| 25 | Turn Off a Connected Light | 1) Go to light screen 2) Click light icon to turn off light |  | Light turned off |
| 26 | Turn Off an Un-Connected Light | 1) Go to light screen 2) Click light icon to turn off light |  | User notified of disconnected light |
| 27 | Remove Light | 1) Go to light screen 2) Click on delete button |  | Light removed |
| 28 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="0" devicename="Lock" deviceroom="Bedroom" | Device added successfully |
| 29 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber=" " channel="0" devicename="Lock" deviceroom="Bedroom" | Device not added |
| 30 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel=" " devicename="Lock" deviceroom="Bedroom" | Device not added |
| 31 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="0" devicename=" " deviceroom="Bedroom" | Device not added |
| 32 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="0" devicename="Lock" deviceroom=" " | Device not added |
| 33 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="123" channel="0" devicename="Lock" deviceroom="Bedroom" | Device not added |
| 34 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="123" devicename="Lock" deviceroom="Bedroom" | Device not added |
| 35 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="0" devicename="123" deviceroom="Bedroom" | Device added successfully |
| 36 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="305311" channel="0" devicename="Lock" deviceroom="123" | Device added successfully |
| 37 | Verify Adding Lock | 1) Go to Lock screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="1234567" channel="0" devicename="Lock" deviceroom="Bedroom" | Device not added |
| 38 | Lock a Connected Lock | 1) Go to Lock screen 2) Click Lock icon to turn on Lock |  | Servo rotates to pos 180 |
| 39 | Lock an Un-Connected Lock | 1) Go to Lock screen 2) Click Lock icon to turn on Lock |  | User notified of disconnected Lock |
| 40 | Unlock a Connected Lock | 1) Go to Lock screen 2) Click Lock icon to turn off Lock |  | Servo rotates to pos 0 |
| 41 | Unlock an Un-Connected Lock | 1) Go to Lock screen 2) Click Lock icon to turn off Lock |  | User notified of disconnected Lock |
| 42 | Remove Lock | 1) Go to Lock screen 2) Click on delete button |  | Lock removed |
| 43 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Sensor" deviceroom="Bedroom" | Device added successfully |
| 44 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber=" " channel="0" devicename="Sensor" deviceroom="Bedroom" | Device not added |
| 45 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel=" " devicename="Sensor" deviceroom="Bedroom" | Device not added |
| 46 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename=" " deviceroom="Bedroom" | Device not added |
| 47 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Sensor" deviceroom=" " | Device not added |
| 48 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="123" channel="0" devicename="Sensor" deviceroom="Bedroom" | Device not added |
| 49 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="123" devicename="Sensor" deviceroom="Bedroom" | Device not added |
| 50 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="123" deviceroom="Bedroom" | Device added successfully |
| 51 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Sensor" deviceroom="123" | Device added successfully |
| 52 | Verify Adding Sensor | 1) Go to Sensor screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="1234567" channel="0" devicename="Sensor" deviceroom="Bedroom" | Device not added |
| 53 | Activate Connected Sensor | 1) Go to Sensor screen 2) Click Sensor icon to activate Sensor |  | Sensor activated |
| 54 | Activate an Un-Connected Sensor | 1) Go to Sensor screen 2) Click Sensor icon to deactivate Sensor |  | User notified of disconnected Sensor |
| 55 | Test activate sensor | 1) Wave hand over activated sensor |  | Notification appears |
| 56 | Test deactivated sensor | 1) Wave hand over deactivated sensor |  | Nothing Happens |
| 57 | Deactivate a Connected Sensor | 1) Go to Sensor screen 2) Click Sensor icon to activate Sensor |  | Sensor deactivated |
| 58 | Deactivate an Un-Connected Sensor | 1) Go to Sensor screen 2) Click Sensor icon to deactivate Sensor |  | User notified of disconnected Sensor |
| 59 | Remove Sensor | 1) Go to Sensor screen 2) Click on delete button |  | Sensor removed |
| 60 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Doorbell" deviceroom="Bedroom" | Device added successfully |
| 61 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber=" " channel="0" devicename="Doorbell" deviceroom="Bedroom" | Device not added |
| 62 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel=" " devicename="Doorbell" deviceroom="Bedroom" | Device not added |
| 63 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename=" " deviceroom="Bedroom" | Device not added |
| 64 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Doorbell" deviceroom=" " | Device not added |
| 65 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="123" channel="0" devicename="Doorbell" deviceroom="Bedroom" | Device not added |
| 66 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="123" devicename="Doorbell" deviceroom="Bedroom" | Device not added |
| 67 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="123" deviceroom="Bedroom" | Device added successfully |
| 68 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="316668" channel="0" devicename="Doorbell" deviceroom="123" | Device added successfully |
| 69 | Verify Adding Doorbell | 1) Go to Doorbell screen 2) Click add device 3) Enter serialnumber 4) Enter channel 5) Enter devicename 6) Enter deviceroom | serialnumber="1234567" channel="0" devicename="Doorbell" deviceroom="Bedroom" | Device not added |
| 70 | Activate Connected Doorbell | 1) Go to Doorbell screen 2) Click Doorbell icon to activate Doorbell |  | Doorbell activated |
| 71 | Activate an Un-Connected Doorbell | 1) Go to Doorbell screen 2) Click Doorbell icon to deactivate Doorbell |  | User notified of disconnected Doorbell |
| 72 | Test activate Doorbell | 1) Press activated Doorbell |  | Notification appears |
| 73 | Test deactivated Doorbell | 1) Press deactivated Doorbell |  | Nothing Happens |
| 74 | Deactivate a Connected Doorbell | 1) Go to Doorbell screen 2) Click Doorbell icon to activate Doorbell |  | Doorbell deactivated |
| 75 | Deactivate an Un-Connected Doorbell | 1) Go to Doorbell screen 2) Click Doorbell icon to deactivate Doorbell |  | User notified of disconnected Doorbell |
| 76 | Remove Doorbell | 1) Go to Doorbell screen 2) Click on delete button |  | Doorbell removed |

Table 2: “Test Cases”

Upon analysis of Figure 1, out of 76 tests, 67 passed, giving it an 88% pass rate. The two reasons tests failed was because of either faulty validation or unimplemented error handling. It is to be noted that letting other users trial the system and receiving feedback on the system would be another great way to test it, this has not been possible however due to *Covid 19 restrictions,* and the fact that each user would need the appropriate hardware required to test the system.

# 5 Evaluation

This chapter of the report aims to compare the product implemented to the original aim and objectives and determine whether the project was successful.

## 5.1 Evaluation of Project Aim

**Breakdown and Evaluation of Project Aim:**

1. ***Control and Monitor Smart Devices***

The aim of being able to control and monitor smart home devices has been partially achieved, the system met 73% of the core requirements. Out of these core requirements, six of them were directed at the controlling and monitoring smart home devices and four have been successfully implemented and tested, these include: controlling lights, locking/unlocking doors, monitoring motion sensors, monitoring doorbells. However, it can be argued that the Phidget devices used weren’t fully capable of acting as smart devices, and that they are only representing or emulating them.

1. ***Cost Effective***

If you were to buy a fully working smart home system currently on the market it would costs hundreds or even thousands of pounds, a hub and multiple devices would have to be purchased which could prove very expensive, as shown in chapter 2.3. If a Raspberry Pi was to be used to run the computer client, the estimated cost of the hub would be around £30 for the Raspberry Pi and £55 for the Phidget Interface Kit totaling at £85. The price of each device also needs to be considered; the Phidget devices are relatively inexpensive coming it at around £5-20. This does mean that the overall system would be much cheaper. On the other hand, this project has not considered installation costs and costs of putting the product onto the market, this means that to a consumer it is inconclusive whether the system would be cost effective or not.

1. ***Efficient***

It can be said that some aspects of the application and system are very efficient due to the low complexity and careful choices of the languages, frameworks, and protocols. However, this was not actually tested so becomes inconclusive.

1. ***Easy to Use and Accessible***

The application interface itself proves relatively simple, well thought out and consistent, it also follows the *8 Golden Rules [Chapter 2.5]*, so it would be agreed that the application itself is easy to use. However, installation of the system could prove very challenging to some and adding more devices than what has been currently implemented would require programming knowledge. The system also only includes one accessibility feature which is dark mode and does not include voice recognition making it not as accessible as it was set out to be.

## 5.2 Evaluation of Project Objectives

This section states each project objective and evaluates whether the objective has been met or not.

1. *Find existing related works and mobile applications for controlling and monitoring smart home systems*

This objective has been partially met and is somewhat covered in Chapters 2.1, 2.2 and 2.3.

1. *Identify user requirements and features*

This objective has been met and was covered in Chapter 3.4.2.

1. *Research and choose an appropriate software design methodology*

This object has been met and briefly covered in Chapter 3.1.

1. *Research appropriate languages and technologies that can be used for Android applications, APIs, computer clients, databases, and voice recognition.*

This objective was partially met and can be found in Chapter 2. However, research of voice recognition was not covered at all.

1. *Research UI design that can be used for Android applications*

This objective has been met and was covered in Chapter 2.5.

1. *Research and choose appropriate smart home devices*

This objective has been met research was conducted in Chapter 2.3 and Phidget devices were chosen to be used as the smart home devices.

1. *Design and Wireframe Android application*

This objective has been met and was covered in Chapter 3.4.

1. *Design database system*

This objective has been met and was covered in Chapter 3.7.

1. *Design API used to communicate with database system*

This objective has been met and was covered in Chapter 3.6.

1. *Design computer client that will be connected to the smart home devices*

This objective was partially met, it was covered in Chapter 3.5. The client was designed, but it was not designed thoroughly and included many floors in the class diagram, this meant it was hard to follow the design which led to the client being rushed during implementation.

1. *Implementation of the Android application, database system, API and computer client and voice recognition*

This objective has been partially met, everything was successfully implemented other than the voice recognition and the API and Client were not up to a high standard as they were rushed. Another major downfall is that the system is not secure at all, security was taking into consideration during research but was not implemented at all.

1. *Thoroughly test and evaluate the system and provide the results*

This objective has been met and is covered in Chapter 4.3.3.

## 5.3 Summary

The product does hit a great deal of the requirements (73%) with a high test pass rate (88%). Some of the products aim and most of the objectives were also achieved. Despite this success, some features were left out and there are floors with the API and Client. Furthermore, security was not taken into consideration and the majority, if not all the accessibility features were not implemented.

# 6 Conclusion

Initially the project background, aims and objective were set, then thorough research was conducted on smart home technology including its benefits and limitations, current products on the market, relevant programming languages, relevant technologies, user interface design and, application usability. This gave a great starting point for the design of the product, the design was meticulous and well thought out, this aided greatly with the implementation of the project. The implementation included an Android application, API, Database system and computer client which all function to a good standard. Thorough testing of the product was then conducted, and it yielded great results. Lastly a critical evaluation was completed which shed light on some of the drawbacks of the product and some of the missed features, even so this project was a great success.

## 6.1 Future Works

There are many things that still needing to be implemented and some extra research to be carried out, these are the future works that would be suggested:

* **Voice Recognition**

Voice recognition would be a great addition to the product and the structure is already in place given the nature of the API and the overall design of the system. Research into natural language processing would be a good starting point for its inclusion.

* **Accessibility features**

There is also a huge opportunity to add some accessibility features to this product, research of appropriate features would have to be carried out.

* **New devices**

It is very easy to add new device to the current system, in terms of programming the API and Database is already set up to handle almost any type of device and the only change to the client would be to add an additional device controller.

* **Improving security**

There are many security risks with smart home technology, and they were barely considered during this project, if the product was to one day be commercially viable, the security of the system would play a big role in this. Extra research would need to be carried out next before taking some extra security measures.

# References

Ali, Y. M. et al. (2020) “Voice Command Intelligent System (VCIS) for Smart Home Application using Mel-frequency cepstral coefficients and linear prediction coefficients”. *In International Conference on Semiconductor Materials and Technology (ICoSeMT 2019).* 1535 012008. Penang, Malaysia. 29-30 April 2019. IOP Publishing Ltd. pp. 1-9. [Online] [Accessed on 9th October 2020] <https://iopscience.iop.org/article/10.1088/1742-6596/1535/1/012008>

Gubbi, J. et al. (2013) “Internet of Things (IoT): A vision, architectural elements, and future directions”. *Future Generation Computer Systems*. Volume 29, Issue 7, September 2013. Pp. 1645-1660. [Online] [Accessed on 16th October 2020] DOI: [10.1016/j.future.2013.01.010](https://doi.org/10.1016/j.future.2013.01.010)

Choudhary, G. Jain, AK. (2016) “Internet of Things: A survey on architecture, technologies, protocols and challenges”. *In 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE).* Jaipur, India. 23-25 December 2016. IEEE. [Online] [Accessed on 16th October 2020] <https://ieeexplore.ieee.org/abstract/document/7939537>

Rizvi, S. et al. (2018) “A Smart Home Appliances Power Management System for Handicapped, Elder and Blind People”. *In* *2018 4th International Conference on Computer and Information Sciences (ICCOINS).* Kuala Lumpur, Malaysia. 13-14 August 2018. IEEE. [Online] [Accessed on 11th October 2020] <https://ieeexplore.ieee.org/abstract/document/8510595>

Miah, J and Khan, RH. (2019) “Service Development of Smart Home Automation System: A Formal Method Approach”. *In* *CIIS 2019: The 2nd International Conference on Computational Intelligence and Intelligent Systems*. Bangkok, Thailand, November 2019. Association for Computing Machinery. pp. 161-167. [Online] [Accessed on 8th October 2020] <https://dl.acm.org/doi/10.1145/3372422.3372437>

Sovacool, B. K. and Furszyfer Del Rio, D. D. (2020) “Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies” *Renewable and Sustainable Energy Reviews.* Volume 120, March 2020, 109663. [Online] [Accessed on 20th May 2021] <https://doi.org/10.1016/j.rser.2019.109663>

Wilson, C. et al. (2017) “Benefits and risks of smart home technologies” *Energy Policy*. Volume 3, April 2017, Pp. 72-83. [Online] [Accessed on 20th May 2021] <https://doi.org/10.1016/j.enpol.2016.12.047>

Robles, R. J. Kim, T (2010) “A Review on Security in Smart Home Development” *International Journal of Advanced Science and Technology*. Volume 15, February 2010, Pp. 13-21. [Online] [Accessed on 21st May 2021] <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.178.1685&rep=rep1&type=pdf>

Balta-Ozkan, N. et al. (2013) “Social barriers to the adoption of smart homes” *Energy Policy.*  Volume 63, December 2013, Pp. 363-374. [Online] [Accessed 21st May 2021] <https://doi.org/10.1016/j.enpol.2013.08.043>

Hong, A. et al. (2020) “What will be the possible barriers to consumers’ adoption of smart home services?” *Telecommunications Policy.*  Volume 44, Issue 2, March 2020, 101867. [Online] [Accessed 21st May 2021] <https://doi.org/10.1016/j.telpol.2019.101867>

Shneiderman, B. et al. (2016) *Designing the User Interface: Strategies for Effective Human-Computer Interaction* Sixth Edition, Pearson.

Williams, A. (2021) Manchester Metropolitan University *Week 2 UI-UX Design* Lecture from Mobile Applications Development Unit.

Williams, A. (2021) *CoffiDa Server* Manchester Metropolitan University [Online] [Accessed 22nd May 2021]<https://github.com/ash-williams/coffida_server>

Google. Google Store. (2021) *Nest Hub Max* [Online] [Accessed on 19th May 2021] <https://store.google.com/gb/product/google_nest_hub_max?hl=en-GB>

Amazon. (2021) *Echo (4th Generation)* [Online] [Accessed on 19th May 2021] <https://www.amazon.co.uk/dp/B08HB1TWCN/ref=s9_acsd_al_bw_c2_x_2_t?pf_rd_m=A3P5ROKL5A1OLE&pf_rd_s=merchandised-search-3&pf_rd_r=WCQNT922NX0ZFP7GD2N9&pf_rd_t=101&pf_rd_p=f33be48f-28a1-401a-b098-d34dd3c5d68d&pf_rd_i=14100223031>

Amazon. (2021) *SmartThings Hub (V3)* [Online] [Accessed on 19th May 2021] <https://www.amazon.co.uk/Samsung-GP-U999SJVLGEA-SmartThings-Hub-2018/dp/B07H9G68DW>

August. (2021) *August Wi-Fi Smart Lock* [Online] [Accessed on 19th May 2021] <https://august.com/products/august-wifi-smart-lock>

Phillips. (2021) *Hue Starter Kits* [Online] [Accessed on 19th May 2021] <https://www.philips-hue.com/en-gb/p/hue-white-and-colour-ambiance-starter-kit-e27/8718696685754#faq>

# Appendices

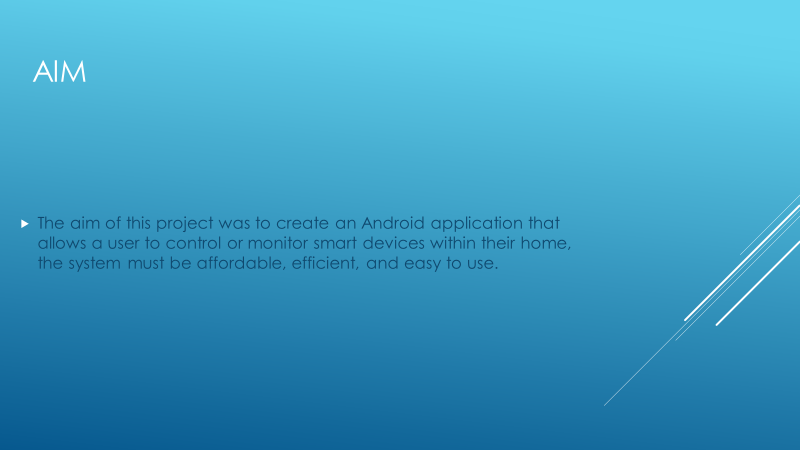
*Appendix A:*

**

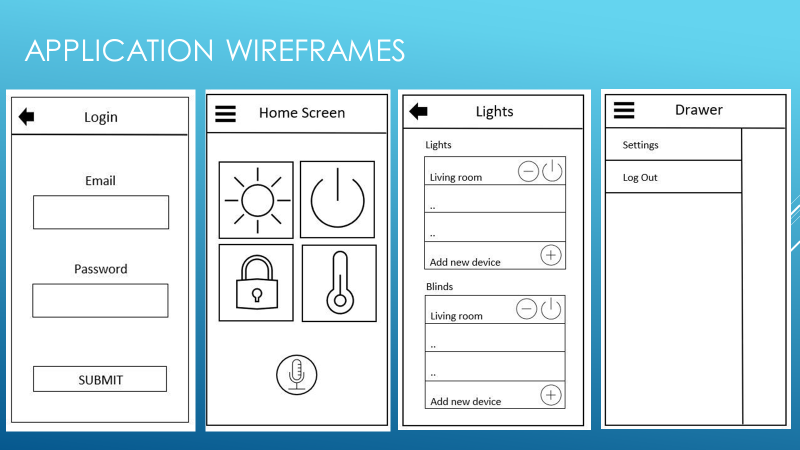
**

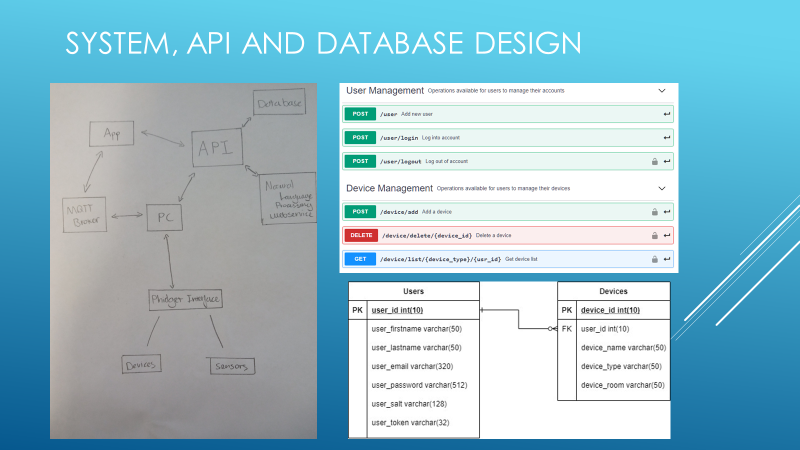
**

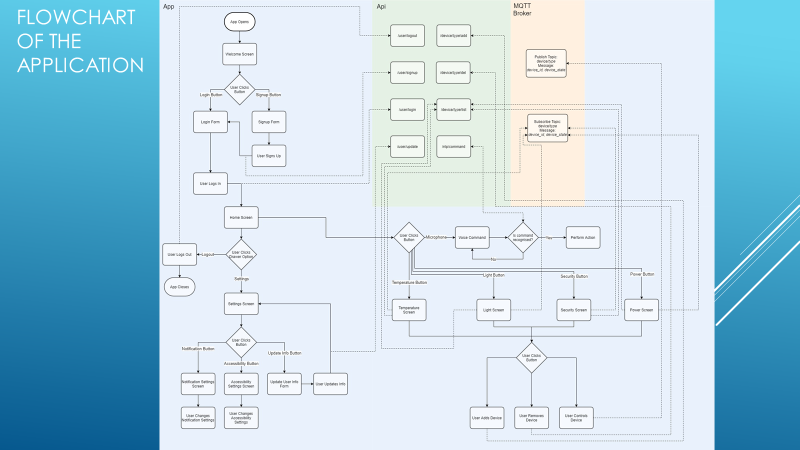
**

**

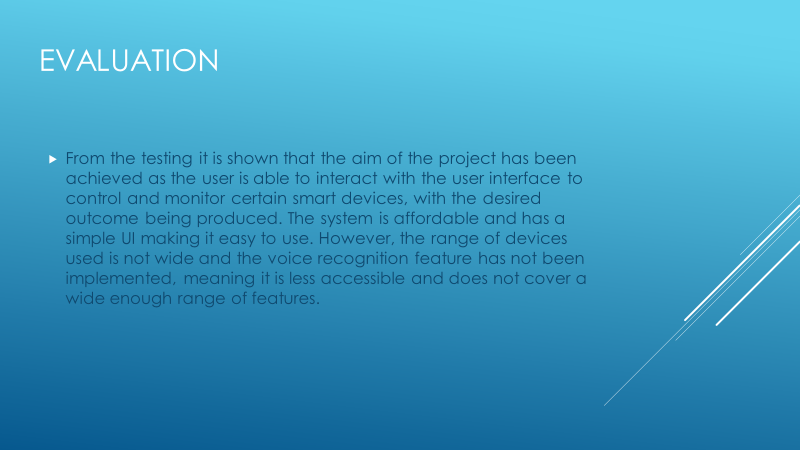
**

**

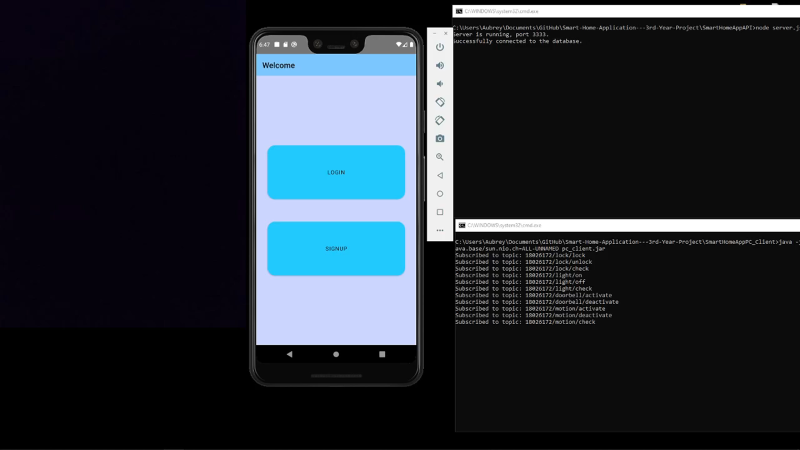
**

**

**

**

**

**

**