Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

Home Operating System

# Functional Specification

| | |
|---|---|
| **Project Name:** | PiLYNK Home OS |
| **Student Name:** | Keith Byrne |
| **Student ID:** | C00170460 |
| **Class Group:** | Software Development (KCSOF_B) |
| **Supervisor:** | Joseph Kehoe |
| **Document:** | Functional Specification |

## Table of Contents

1. Project Overview ..... 4
1.1 Introduction ..... 4
1.2 System Architecture Strategy Brief ..... 4
1.2.1 Communication Access Layer (Frontend) ..... 5
1.2.2 Control Service Layer (Backend) ..... 5
1.2.3 Command Layer (Backend Extension) ..... 5
1.3 System Architecture Expanded ..... 6
2. Functional Requirements ..... 7
2.1 Key Functional Points ..... 7
2.1.1 Control Device Power ..... 7
*2.1.1.1 Outlet Device Control ..... 7*
*2.1.1.2 Lighting Control ..... 7*
2.1.2 Device Grouping ..... 7
2.1.3 Cost Estimation ..... 7
2.1.4 Accounts ..... 8
2.1.5 Expansion ..... 8
2.1.6 Sensor Network Monitoring ..... 8
*2.1.6.1 Temperature ..... 8*
*2.1.6.2 Movement ..... 8*
2.2 Use Cases ..... 9
2.2.1 Use Case Diagram ..... 9
2.3 Brief Use Cases ..... 10
2.3.1 Login ..... 10
2.3.2 Logout ..... 10
2.3.3 Switch Device ..... 10
2.3.4 View Usage History ..... 11
2.3.5 View Temperature History ..... 11
2.3.6 View Cost Estimation ..... 11
2.3.7 Approve User Registration ..... 12
2.3.8 Request Registration ..... 12
3. Supplementary Specification ..... 13
3.1 Objectives ..... 13
3.2 Functionality ..... 13
3.2.1 System Activity Logging ..... 13

3.2.2 System Error Logging ....13
3.3 Usability ....14
3.3.1 Browser Compliance ....14
3.3.2 Design for Ease of Use ....14
3.4 Reliability ....15
3.5 Performance ....15
3.5.1 Simultaneous Users ....15
3.5.2 Database Access Response Time ....15
3.6 Supportability ....15
3.7 Security ....16
3.7.1 Storage ....16
3.8 Design Constraints ....16
3.8.1 Platform Requirements ....16
4. References ....17

# 1. Project Overview

## 1.1 Introduction

PiLYNK will be an all in one self-contained micro computing home operating system acting as privately hosted web server in tandem. The system can be installed on any premise with ease and little to no expert training is required. This will be of significant consideration in the physical design of the system in order to reduce installation efforts as much as possible.

Users will be able to access the system using a routine switch and link IP service such as NO-IP. This is due to the fact that most broadband providers don't offer static IP services to typical consumers and a cloud hosting solution would add more dependencies to the system. Thus, access will still be possible from anywhere in the world with a public network link and a web browser and potentially an Android application if feasible.

## 1.2 System Architecture Strategy Brief

The system will be divided into three separate layers to allow for all functional needs to be catered for. Design will be focused from a top down perspective with the user being the first point of contact and control being the last. A C3 (command, control and communicate) strategy will be adopted in dealing with the key concerns of controlling devices from integrated command protocols from users via the systems communication layer (front end).

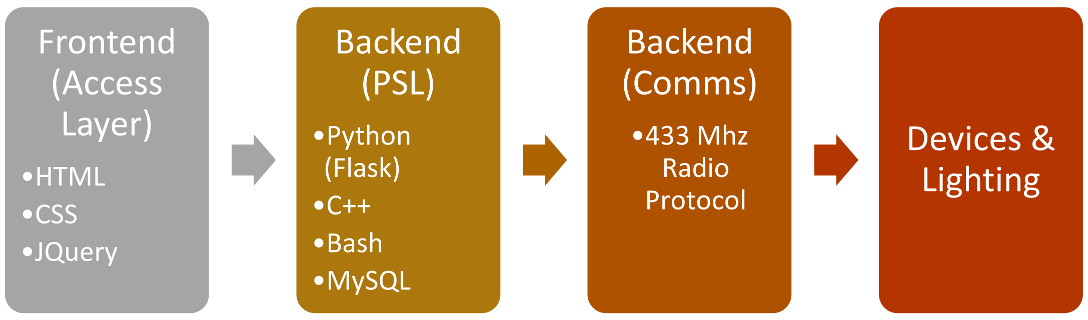


*Figure 1.a - Architectural Overview (Brief)*

### 1.2.1 Communication Access Layer (Frontend)

Present at this layer will be the principal GUI’s, either in the form of a web application or Android application. This will provide the primary system stakeholders a method of interaction with the system. The bulk of the work regarding the frontend is carried out in iteration two and iteration three.

### 1.2.2 Control Service Layer (Backend)

This is where the Raspberry Pi and all integrated developments will become operational. The Raspberry Pi will be used to host a lightweight web server to complement the Pi’s lower power operating model.

### 1.2.3 Command Layer (Backend Extension)

This layer represents the lowest level component structure of the system. All radio communication devices will be present here. It has been decided that this layer be separated from the initial backend layer because of the nature of this portion of the system. Certain aspects can’t be abstracted and require a particular level of detail. Mainly due to the fact that a schematic is required for its construction and the Raspberry Pi’s GPIO sits at a lower layer than the control service layer itself. However, no software requirements exist at this layer, it is purely physical and also returns no data to layer two.

## 1.3 System Architecture Expanded

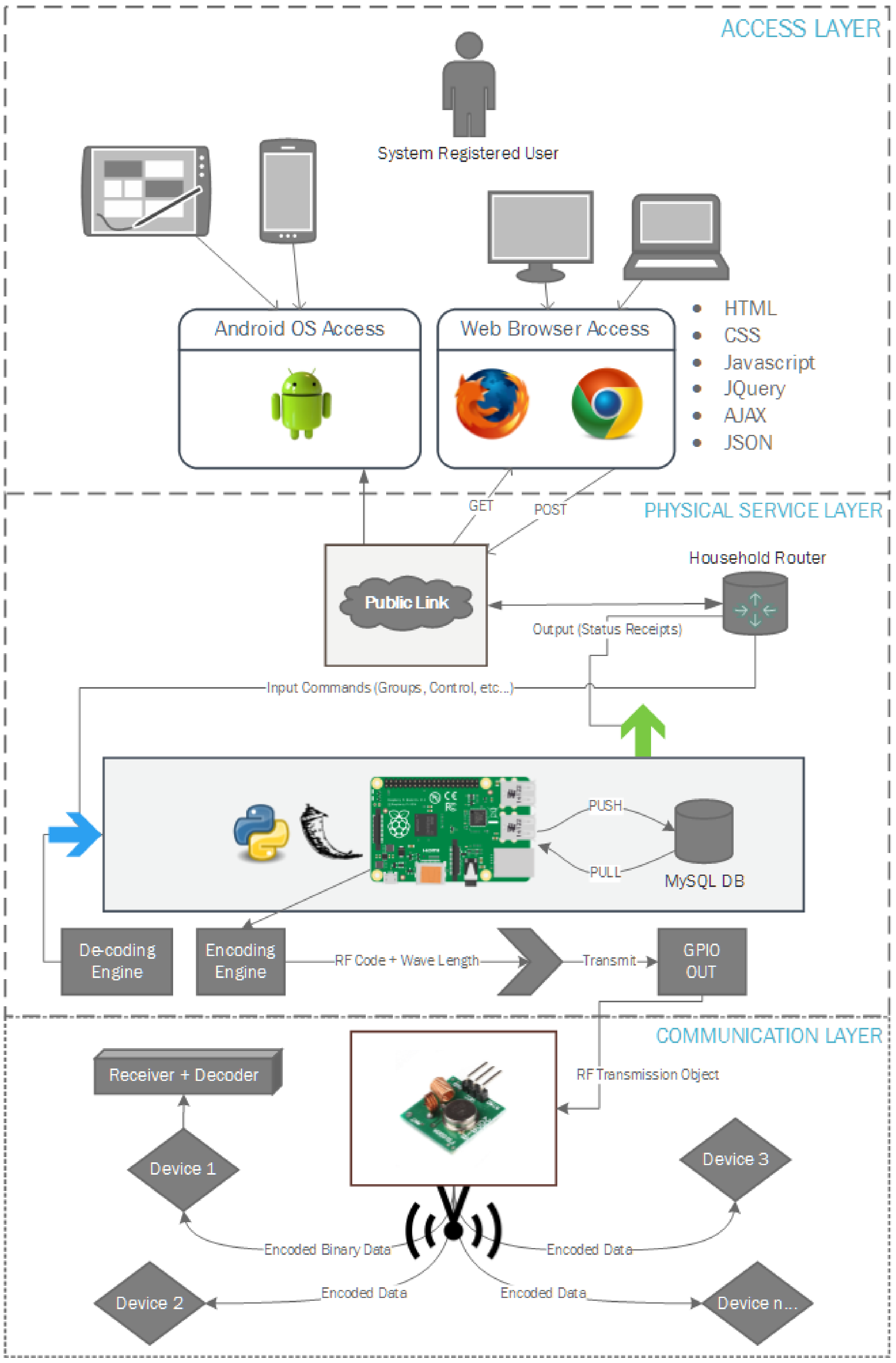


*Figure 1.b - System Architecture (Detailed)*

# 2. Functional Requirements

## 2.1 Key Functional Points

### 2.1.1 Control Device Power

This is, at its core, the most crucial functional point of the project. All other function points extend from this core feature. This will require all layers to operate effectively. In essence, it is the ability to control the flow of electrical current to a device such as a phone charger, television set, desk lamp, floor lamp, ceiling lights, etc. User should be able to interact with this functionality via a simplistic user interface.

#### *2.1.1.1 Outlet Device Control*

This introduces the notion of controlling the flow between the household mains socket and the device in question. This can be a somewhat dangerous measure, thus, the safest method is being considered as not all interested stakeholders of the system may own their own property. Interference with the outlets on a property is forbidden in most letting agreements.

Lightwave RF offer pre built solutions to this using inline radio controlled relays integrated into a standard socket wiring solution. Deconstruction of the gang sockets is necessary in this case. Belkin WeMo offer the safer, external option of inline socket interrupts. Which is the basis of this project’s control method also, at a much lower cost, however.

#### *2.1.1.2 Lighting Control*

Controlling ceiling lights offers a more difficult set of problems than standard plug and control devices. Very little third party solutions exist to allow for this. Nevertheless, during extensive research it was discovered that a set of integrated lighting socket interrupts similar to the wall socket interrupts are available.

Philips Hue specialises in this area of home automation, although most of their solutions are controlled via a locked down control bridge.

### 2.1.2 Device Grouping

Users can request that several devices be bound together into a node group. This will allow macro control of a set of devices without individual interaction. This has uses from parental control to total household switch off.

### 2.1.3 Cost Estimation

Deployed with the system should be an industry researched estimation list. This will allow users to assign estimate metric control to a device, for example, the typical wattage of a household floor lamp. These metrics can be expanded manually and the internal calculation processor will output cost estimations.

#### 2.1.4 Accounts

Since the system will be publically accessible via the web, it is crucial that account security be held in the highest regard. For this, this set of key functionality points will be flagged as a priority one development consideration.

#### 2.1.5 Expansion

It would be ideal to allow for the system to be expanded post-installation by a typical user. Consideration will be given to this however, deploying a test set of devices is a bigger priority. The greatest complication with this is the varying sets of radio standards available. Some devices operate on completely different protocols than the given set with this project.

#### 2.1.6 Sensor Network Monitoring

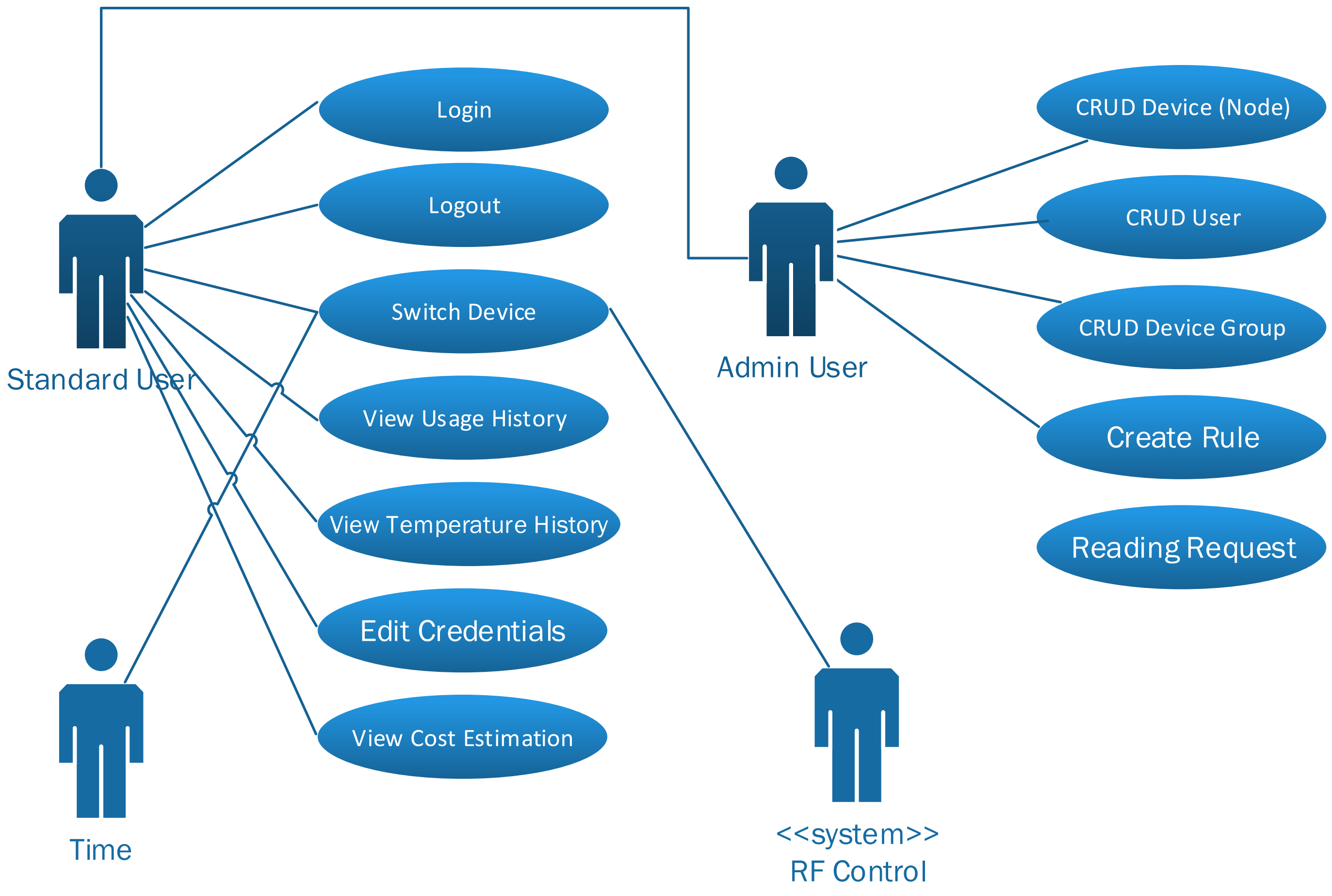
##### *2.1.6.1 Temperature*

After extensive research into sensors compatible with the Raspberry Pi's GPIO pins, it was concluded that integrating sensors would be relatively simplistic and consideration has been given for a temperature monitor relay system. This offers the user the ability to obverse current and past temperature readings using a simple module such as the DHT22 Digital Temperature & Humidity Sensor. Compact devices such as this use a capacitive humidity sensor and a thermistor to measure the surrounding air, and outputs a digital signal on the data pin to the Raspberry Pi's GPIO input pin.

##### *2.1.6.2 Movement*

## 2.2 Use Cases

### 2.2.1 Use Case Diagram



## 2.3 Brief Use Cases

### 2.3.1 Login

| **CASE CATEGORY:** | **SECURITY** |
|---|---|
| **NAME:** | Login |
| **ACTORS:** | Standard User, Administrative User, System |
| **PRECONDITIONS:** | The web application has successfully loaded from the web server and the initial login screen has been successfully loaded. |
| **ACITIVTY:** | This use begins when the user opens the PiLYNK web page. The system listens for the user login username and password. Validations carried out on the client. |
| **CONSEQUENCE:** | Once confirmed on the back end, the system displays the home page. System logs activity. |

### 2.3.2 Logout

| **CASE CATEGORY:** | **SECURITY** |
|---|---|
| **NAME:** | Logout |
| **ACTORS:** | Standard User, Administrative User, System |
| **PRECONDITIONS:** | The user has been successfully logged in. |
| **ACITIVTY:** | This use case begins when a user has completed all intended actions on the front end of the system. The user clicks the logout button and asked for confirmation of this action. |
| **CONSEQUENCE:** | The user session is destroyed and the user must log in again in order to return to the system dashboard. System logs activity. |

### 2.3.3 Switch Device

| **CASE CATEGORY:** | **CONTROL** |
|---|---|
| **NAME:** | Switch Device |
| **ACTORS:** | Standard User, Administrative User, System, RF Controller |
| **PRECONDITIONS:** | 1. User has been successfully logged in and directed to the dashboard.<br>2. A device has been setup with the system. |
| **ACITIVTY:** | This use case begins when a user wishes to switch a currently linked device on or off indiscriminately. The system will feedback the current status and signal the RF Controller to activate or disable the current to the device. |
| **CONSEQUENCE:** | The device will switch. Time rules will still remain active on the device. System logs activity. |

### 2.3.4 View Usage History

| CASE CATEGORY: | REPORTING |
|---|---|
| NAME: | View Usage History |
| ACTORS: | Standard User, Administrative User, System |
| PRECONDITIONS: | 1. User has been successfully logged in and directed to the dashboard.<br>2. Historical records exists. Reports will not be permitted with no data. |
| ACITIVTY: | This use begins when the admin user wishes to view the activity log of the system. This log will contain historical user-system communication transactions. |
| CONSEQUENCE: | System logs activity. |

### 2.3.5 View Temperature History

| CASE CATEGORY: | REPORTING |
|---|---|
| NAME: | View Temperature History |
| ACTORS: | Standard User, Administrative User, System |
| PRECONDITIONS: | 1. User has been successfully logged in and directed to the dashboard.<br>2. System has been active for more than five to ten minutes. |
| ACITIVTY: | This use case relates to the reporting of temperature sensor relay information. A user selects to view the *general* report. From this menu, users can select filtered views, for example, daily, weekly and monthly temperature reports. |
| CONSEQUENCE: | System logs activity. |

### 2.3.6 View Cost Estimation

| CASE CATEGORY: | REPORTING |
|---|---|
| NAME: | View Cost Estimation |
| ACTORS: | Standard User, Administrative User, System |
| PRECONDITIONS: | 1. User has been successfully logged in and directed to the dashboard.<br>2. At least one device has been active during the system lifecycle. |
| ACITIVTY: | This use case relates to the reporting of cost estimation. A user selects to view the *general* report. From this menu, users can select filtered views, for example, daily, weekly and monthly cost reports. The system generates these reports based on predetermined estimate values. |
| CONSEQUENCE: | System logs activity. |

### 2.3.7 Approve User Registration

| CASE CATEGORY: | SECURITY |
|---|---|
| NAME: | Approve User Registration |
| ACTORS: | Administrative User, System |
| PRECONDITIONS: | 1. Admin user has been successfully logged in.<br>2. A user has requested system access (Register use case dependency) |
| ACITIVTY: | This use begins when an admin user receives an account access request from a non-registered user. The admin user has the right to review and deny or approve the request be it from a family member or other member of the same household. |
| CONSEQUENCE: | The user requesting access gets accepted or denied with the system and receives email notification of the process. |

### 2.3.8 Request Registration

| CASE CATEGORY: | SECURITY |
|---|---|
| NAME: | Request Registration |
| ACTORS: | Standard User, System |
| PRECONDITIONS: | Standard user credentials do not exist |
| ACITIVTY: | This use case begins when a non-registered user wishes to access the system. The user enters their details via application form. The form is submitted to the system and then the decision is forwarded to the principal administrative user for approval or rejection. |
| CONSEQUENCE: | The user must wait until they have been approved before proceeding to any future stages of the system. This period is purely at the discretion of the principal administrator. |

# 3. Supplementary Specification

## 3.1 Objectives

The purpose of this section of the document is to define supplementary requirements of the PiLYNK system. This **Supplementary Specification** lists the requirements that are not readily captured in the use cases of the use-case model. The Supplementary Specifications and the use-case model together capture a complete set of requirements on the system.

## 3.2 Functionality

This application places a heavy focus on responsive user interaction with the system and the precise, reliable and reportable control of devices around the home. After all, the system is responsible for reducing the cost of living but also to be safe.

Supplementing the base functionality covered in the Use Cases, this section contains a list of functional considerations that are common to more than one use case.

### 3.2.1 System Activity Logging

All core system activity will be logged. This is in order to monitor trends in activity so the system may suggest improvements. This is however a post project submission consideration.

The system activity messages will include a text description of the error, date time of the activity event, user details and the event sub system description. All system activity reports will be stored in a separate Audit Control Database with access granted **only** to the principal system administrator.

### 3.2.2 System Error Logging

All exceptional system behaviour will be logged. This is to allow the system administrator to understand the cause of critical system failure in order to diagnose the problem.

The system error messages will include a detailed text description of the error, date time of the event, which user triggered the sequence to cause the event and which section of the server was responsible for throwing the exception.

## 3.3 Usability

It has been heavily considered that not all system users will be computer literate adults. Most of these uX interaction related issues will be teased out during the core design phase of the project. This section lists all of those requirements that relate to, or affect, the usability of the system.

### 3.3.1 Browser Compliance

Development consideration cannot be given to all currently used browsers available. Google Chrome and Mozilla Firefox will be the focus point of the design.

### 3.3.2 Design for Ease of Use

The user interface of the PiLYNK System will be designed for ease-of-use and shall be appropriate for both computer-literate and limited literacy users. No training will be required to use the system.

## 3.4 Reliability

This section lists the reliability attributes of the system.

The system should be available 24 hours a day, 7 days a week. Downtime should only occur when external forces are involved, principally regarding power outages and maintenance on the premises. Nevertheless, the system should be able to automatically recover and restart server operations once power has resumed. A maximum of 10% downtime per year should be the achieved.

## 3.5 Performance

The performance characteristics of the system are outlined in this section.

### 3.5.1 Simultaneous Users

The system should be able to support a number of users at one time. However, due to the typical bandwidth of home broadband, this should not be strained extensively. Subsequently, a maximum count of ten users will be enforced in the system.

### 3.5.2 Database Access Response Time

Due to certain CPU limitations on the Raspberry Pi, database read and writes may be slow if the buffer is overloaded. Due to this, certain capacity thresholds may be implemented to allow for successful albeit slow right over many unsuccessful writes. One possible consideration for this is a timeout counter on device control.

## 3.6 Supportability

The system will not initially be heavily concerned with supporting minor changes. Because of the number of integrated technologies ranging from Python, C++, HTML, CSS, etc. Any updates may cause dependency related compatibility failures. Hence, major system upgrades will be handled on a board by board basis rather than exclusively any particular module. Upgrades will be carried out on a dormant system board rather than a live system.

Newly developed, tested and stable iterations will be available via GitHub for product owners. This will be a simple pull, update, restart server process with written assistance and troubleshoots for each revision.

## 3.7 Security

### 3.7.1 Storage

Due to the nature of the system having the ability to control devices in the home and potentially create fire hazards, a strong and secure user presence and persistent storage is required. To tackle this issue, the system will be developed with encrypted username and password storage using the Python Werkzeug library, specifically the SHA1 hashing algorithms.

## 3.8 Design Constraints

This section lists any design constraints on the system being built.

### 3.8.1 Platform Requirements

As the system will be running Python/Flask micro web framework as the server technology. Development will be focused around Python version 2.7 and continuously the latest Flask releases. C++ version will be the default version included on the Raspberry Pi B+ model.

The server operating system will be the Debian based Raspbian:

- **Wheezy**: Kernel 3.18
- **Released**: May 2015

## 4. References

[1] WeMo, “WeMo,” 1 October 2015. [Online]. Available: http://www.wemo.com/.

[2] Belkin, “Belkin WeMo Products,” 20 September 2015. [Online]. Available: http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/.

[3] L. W. R. PLC, “Light Wave RF,” Light Wave RF, 5 May 2015. [Online]. Available: http://lightwaverf.com/. [Accessed 2 October 2015].

[4] ADT, “ADT Security Specialists,” 2015 ADT LLC DBA ADT Security Services, 2015. [Online]. Available: http://www.adt.com/. [Accessed 1 October 2015].

[5] Philips, “Meet Hue,” Philips, 2015. [Online]. Available: http://www2.meethue.com/en-gb/about-hue/what-hue-does/. [Accessed 2 October 2015].

[6] Philips, “Philips Apple HomeKit,” Philips, 2015. [Online]. Available: http://www2.meethue.com/en-gb/friends-of-hue/apple-homekit/. [Accessed 2 October 2015].

[7] A. Fruit, “Raspberry Pi B+ Model,” Ada Fruit, 2015. [Online]. Available: https://www.adafruit.com/products/1914. [Accessed 5 September 2015].