ELEN4009 - Software Engineering Project
Smart Home Power Management System

Front-End Pair: Daniel Weinberg (547937) and Alice Yang (597609)
Back-End Pair: Kanaka Babshet (678851) and Ari Croock (718005)

2016/04/11

First Prototype Design and Execution of a Smart Home Power Management System 2016/04/11

Ari Croock - 718005
Kanaka Babshet - 678851
Alice Yang - 597609
Daniel Weinberg - 547937

ELEN4009 - Software Engineering

School of Electrical & Information Engineering, University of the Witwatersrand, Johannesburg

Abstract: This paper details the design and implementation of the first prototype of a smart home power management system. The final project submission is composed of an application which can add a household device, and control its state, as well as a compilation of all the documentation written in the process of designing this application. Details of the experiences during the design, implementation, and testing, have been provided. All constraints and specifications are successfully met, and additional improvements can still be made in the future.

Key words: Design, Smart Home, Power Management, Internet of Things

1. INTRODUCTION

A smart home power management system allows users to control and supervise various electrical devices within the household in order to manage the overall power consumption. The system also automatically manages various devices without human interaction depending on the settings enabled in advance by the system user. This interaction and status display is done via a webpage accessible to all predefined household users.

1.1 Problem Statement

With the rapidly growing interest in new Internet of Things (IoT) technologies, networks consisting of these devices will become increasingly difficult to manage and control. Additionally, power consumption and monitoring will become a greater concern, especially in emerging markets such as South Africa.
Given this knowledge, surveys were conducted in order to define the specifications and the requirements of the Smart Home Power Management System.

The survey is structured in such a way that it enables users to provide useful information required by the software engineers. The information provided allows the software engineers to define the problem, sketch outlines for the best solution, and determine the constraints as well as resources needed for the system design.

In obtaining information from the potential users, the software engineers are able to get a precise overview of the requirements that the users are expecting from the system, as well as an insight of the system from the user's perspective.

The figures in Appendix A illustrate the results obtained from the survey.

1.2 Project Objectives

The aim of this project is to design the complete aforementioned system and implement some aspect of the system which is defined as the first prototype. The prototype implemented is made up of a scrollable webpage, with a page header, the addition and removal of a device, with corresponding specifications, viewing the devices connected, and toggling between the states of devices.

1.3 Stakeholders

Stakeholders are defined as the people, groups, or organisations that have interest, concern, or influence in an organisational project [1]. Figure 1 below illustrates the project stakeholders that will directly interact with the system as well as those external to it.

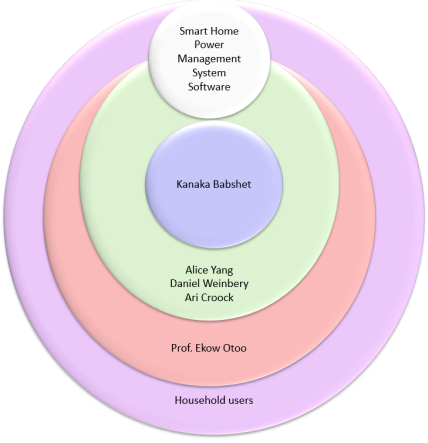

Figure 1 : Project stakeholder onion diagram

1.4 Report Breakdown

Section 2 details the software requirement specification, and is followed by section 3 which contains an extensive design document split into the relevant front- and back-end sections. Section 4 details the initial prototype implementation. Section 5 introduces the sprint planning and illustrates the work breakdown over the time designated for this project, while section 6 provides the sprint retrospective corresponding to section 5.

2. SOFTWARE REQUIREMENT SPECIFICATION

2.1 Purpose

This section details the system requirements specification (SRS) for the Smart Home Power Management System. The system design documentation will be developed from this information.

This project aims to provide a flexible software system which is able to remotely control and monitor IoT devices, as well as perform detailed power consumption diagnostics.

2.2 Project Scope

The project is a system that will allow for remote control, monitoring and automation of IoT devices on a Local Area Network (LAN). A client-server architecture will be used since this allows a back-end server to continuously manage devices while allowing for a client to connect on-demand. The front-end will initially be implemented as a web page for simplicity and compatibility with many existing devices (such as cellphones and personal computers).

The back-end provides functionality to control and monitor devices, and to log device power consumption. Additionally, the back-end will contain the web server used for interfacing with users.

The front-end will consist of a web-based user interface which will provide secure access to a device dashboard. The dashboard will provide remote control and configuration of devices, as well as access to power consumption data. Addition and removal of IoT devices will also be performed through the dashboard.

2.3 Project Overview

Figure 2 shows a data flow diagram of the system. Data collected from IoT devices is stored in a database and used to control other devices as defined by the user. A web interface allows the user to configure devices and to view the stored data.

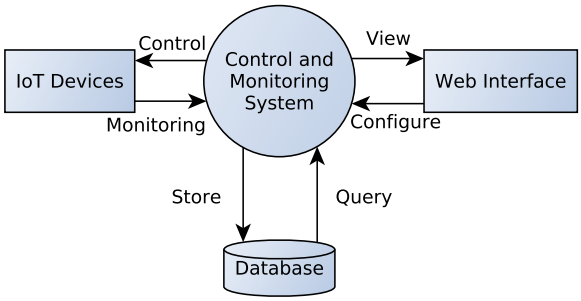

Figure 2 : Data flow diagram (DFD) of the system. All data flows are bidirectional

2.3.1 Assumptions:

- All of the IoT devices will be connected to the same LAN
- Only a small number of simultaneous users and devices will be connected at once

2.3.2 Constraints:

- The user interface must be accessible through a web browser
- All devices in the system require wireless connectivity

2.3.3 User Classes and Characteristics:

- Basic users who can configure basic devices and view power usage statistics
- Advanced users who can configure custom devices and export power usage statistics for further processing

2.3.4 Operating Environment: The following technologies will be used to implement the system:

- Django Web Framework (Python 3.5.1)
- PostgreSQL Database Management System (DBMS)
- Docker 1.8.2

Assuming that few users and devices will be connected to the system simultaneously (such as in a typical household), the system requirements are fairly relaxed:

- Any modern Microsoft Windows or GNU/Linux system that supports the technologies used
- A connection to the LAN, such as Ethernet or WiFi
- Low-cost hardware such as a Raspberry Pi

However, depending on the system configuration, an external storage solution may be needed for storing a large amount of power usage data.

2.3.5 Software Development Life-cycle: The system will be developed using an Agile software development methodology. In particular, the popular SCRUM framework will be used so that the system's requirements and design details can change quickly if need be.

2.4 System Features

The smart home power management system is made up of various features which enhance the usability of the product, along with some additional features which simplify all related functionality.

2.4.1 System Feature 1 - Remote On/Off Switching: Switch any of your selected appliances on or off from any personal device connected to the home network.
Description: The user can control the power status of various appliances in the household. The appliances are connected independently, for which reason, multiple appliances can be controlled through the same interface, simultaneously.
Stimulus/Response Sequences: The user has to select the "Power Remote" tab on the home page to display the on and off options corresponding to each connection.

2.4.2 System Feature 2 - User Configurable Triggers: Personalise various situations which will trigger an action.
Trigger 1 - Motion Detection: Set appliances to switch on or off depending on surrounding motion.
Description: The user can control whether specific appliances must be configured with motion detection or not. For example, if motion detection is set for a certain set of lights, they will detect movement in the surrounding area and switch on, or switch off when there is no motion detected for a set period of time. If the user does not want an appliance or set of lights to be triggered by motion, the respective switch is "off" on the webpage.
Stimulus/Response Sequences: The user has to select the "Triggers" tab from the home page, and thereafter select the "Motion Detection" tab at the top left of the newly loaded page to display the corresponding settings. Figure 4 in Section 4 below illustrates this page.
Trigger 2 - Temperature Detection: Set appliance statuses to vary depending on the surrounding temperature.
Description: The user can control whether specific appliances must be configured with heat triggers or not. For example, if temperature detection is set for a specific appliance, it will detect a certain temperature in the surrounding area and switch on, switch off, or maintain state if it is too hot, too cold, or within the correct temperature range depending on the user choices. If the user does not want an appliance or set of lights to be triggered by the ambient temperature, the respective switch is "off" on the webpage.
Stimulus/Response Sequences: The user has to select the "Triggers" tab from the home page and thereafter select the "Temperature Detection" tab at the top of the newly loaded page, (second from the left), to display the corresponding settings.
Trigger 3 - Light Detection: Set appliance statuses to vary depending on the surrounding light.
Description: The user can control whether specific appliances must be configured with light triggers or not. For example, if light detection is set for a specific appliance, it will detect a certain ambient light level in the surrounding area and correspondingly vary, or maintain state depending on the user choices. If the user does not want an appliance or set of lights to be triggered by the ambient light levels, the respective switch is "off" on the webpage.
Stimulus/Response Sequences: The user has to select the "Triggers" tab from the home page and thereafter select the "Light Detection" tab at the top of the newly loaded page, (third from the left), to display the corresponding settings.
Trigger 4 - Time Scheduling: Select and adjust times for the different appliances to be on, off, or running on defined settings.
Description: The user can activate time scheduling for each device/set of lights independently connected to the smart home power management system. For example, the user can set the geysers to switch on from 6 am to 8 am and from 7 pm to 9 pm everyday. Note that the selection of time scheduling for appliances does not affect the user's ability to manually control the power via the system, or directly in person.
Stimulus/Response Sequences: The user has to select the "Triggers" tab from the home page and thereafter select the "Time Scheduling" tab at the top right of the newly loaded page to display the corresponding settings.

2.4.3 System Feature 3 - Power Consumption Monitoring: Monitor the power consumption by various appliances and sets of lights in the household.
Description: Given that this system is based on power management, the user can easily view detailed layouts of the total household power consumption data and analytics. The user will be able to view graphical statistics of the power consumption over the course of the day, as well as detailed statistics over a longer period of time (previous month, or year). This allows the user to compare the household power consumption with that of the usage if this system was not implemented.
Stimulus/Response Sequences: The webpage automatically starts on the "Home Status" tab upon loading. The user can view this page for a daily consumption summary, and can then click on the "Consumption Monitoring" tab seen on the left of the same page to view these analytics in greater detail, over a

longer period of time.

2.4.4 System Feature 4 - Energy Source Management: Control and manage the different energy sources in the household such as varying the usage between electricity and solar power.
Description: This system encourages the use of alternative, renewable energy sources. If the user has catered for these facilities, the system allows additional power volume management through the energy source management configuration.
Stimulus/Response Sequences: The user has to select the "Energy Source Management" tab on the left of the homepage to be directed to this page.

2.4.5 System Feature 6 - User Configurable Alerts: Choose which events should result in an immediate alert/notification on your system.
Description: Along with all the personalised settings configured on the system, the user is able to define situations in which alerts appear on the system and possibly directly to the user's smart device.
Stimulus/Response Sequences: The user has to select the "Alerts" tab on the left of the homepage to then configure these settings.

2.4.6 System Feature 7 - User Configurable Webpage Settings Define webpage settings in order to enhance the power management system experience.
Description: The user can add or remove devices to be controlled by the smart home power management system, alter the webpage appearance, manage users on the network, and edit security settings.
Stimulus/Response Sequences: The user must click on the "Settings" link to the right of the title on the home page to be able to configure all the required settings.

2.5 External Interface Requirements

The external interface is the interface which allows the user to interact with the system through a hardware device using software. The external interfaces consists of a user interface, hardware interface, software interface and communication interface.

2.5.1 User Interface - GUI: The user interface is how the user will interact with the system. The requirements for the design of user interface must be simple and easy to navigate. A simple format of representing all the devices connected to the Smart Home Power Management System is to display the features available for all devices on a dashboard. This can been seen in Figure 3.

Figure 4 is a simple illustration of the interface between the user the light detection triggers on the system. The layout of the tab is further divided to clearly illustrate the sub-systems which the user has access to. As seen from the figure, the switches on the triggers tab are designed in a graphical format that is simple for users to use.

Figure 5 is the settings page, in which it designed for the user to navigate through each sub-system. The *Add/Remove Device* feature is an important feature, as it is expected that the client may have to add or remove devices for future use. The settings page is set up similarly to the home page on the trigger tab, as user's familiarity to the website is of importance.

- Home Page Dashboard

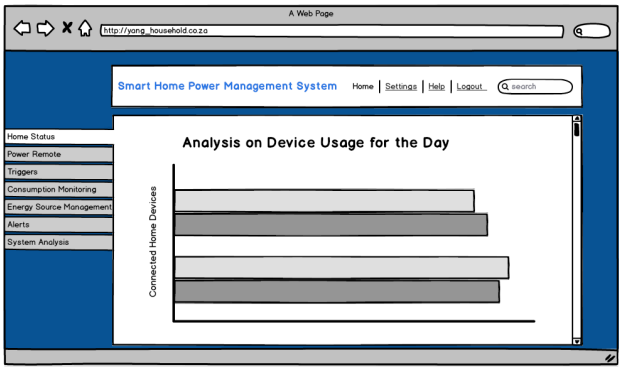

Figure 3 : Figure illustrating the home page the user will see once they have logged on to access the system.

- Trigger Tab on Dashboard

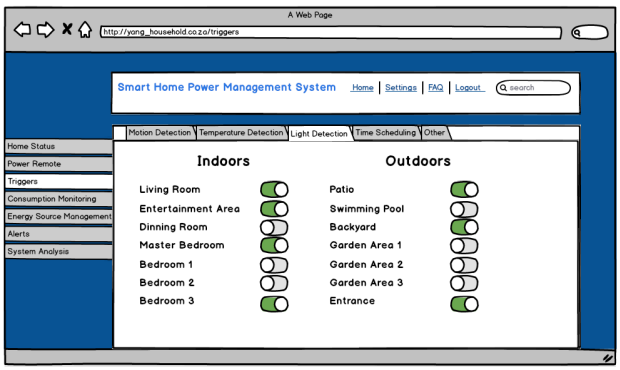

Figure 4 : Figure illustrating the triggers tab, to allow users to control switching of sensors in their home through the system.

- Settings Page

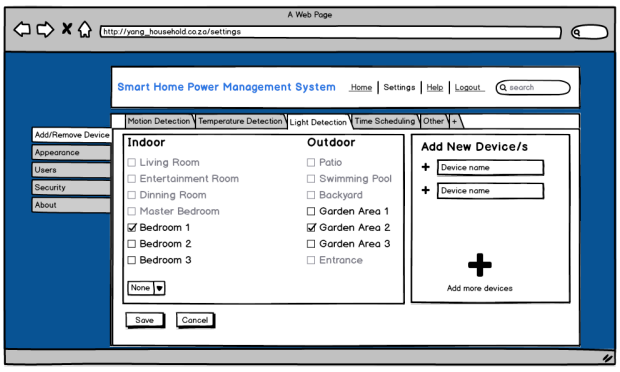

Figure 5 : Figure illustrating the settings page.

2.5.2 Hardware Interface: The user interface is implemented on a web based platform that is accessible through a browser. The hardware required to access the system can be either of the listed devices found in the list below:

- Large Desktops (1200 - pixels)
- Medium Desktops (992 - 1199 pixels)
- Small Devices (768 - 991 pixels)
- Extra Small Devices (max 767 pixels)

2.5.3 Communication Interface: The Smart Home Power Management System is a web-based application that requires a internet connection for the communication between the hardware devices, system and IoT devices. The main connection between the system and the IoT devices is through the user's home wireless connect (WiFi). For the hardware device to communicate with the system any type of internet connection will do.

2.6 Other Non-Functional Requirements

2.6.1 Performance Requirements: The smart home power system is an application and system that is designed for efficiency. This means that the application as well as the connecting household components need to respond quickly and in real time.

The application will most likely undergo several updates and changes for the user's benefit. The system is required to update on user command. Updates will include further improvements to the application as well as fixes for issues that arise in operation.

2.6.2 Safety Requirements: The application as well as the system components need to take certain safety concerns into account.

Due to the fact that the application is essentially controlling most of a home's appliances and electricity usage, it is important that it monitors everything it is controlling to prevent problems that may arise. The application needs to ensure that any device connected in the system does not reach a dangerous level of usage. In this case, the application should either switch off the device in question or notify the user that something is not functioning correctly and needs to be addressed.

2.6.3 Security Requirements: Due to the fact that the system controls a user's home, it therefore requires security considerations to be taken into account.

The system needs to ensure that only the user has access to the application to prevent external parties gaining control of the connected components within a house. This can be done with a signup, login and authentication process. The components within the house that the application connects with, need to be protected from external parties and therefore they must be explicitly authenticated.

2.6.4 Software Quality Attributes: The application is web based and needs to be user friendly. The functioning of the application needs to be simple so that no additional documentation or prior knowledge or experience is required.

3. DESIGN DOCUMENT

The purpose of this section is to provide details regarding the software development of the project.

3.1 Scope

The scope of the project is for a basic prototype of the proposed power management system. The focus of the system is to manage connected devices and their corresponding power usage data.

The scope of this prototype, however, is limited to the management and analytics portion of the complete system, not the inter-device communication aspect.

I. FRONT-END

3.2 Overview

The front-end of the power home management system is an integral component of the system as a whole. Without the presence of the front-end, the back-end becomes essentially irrelevant. The front-end allows the user to interact with the smart home system through a web-page platform. Within the front-end are graphical and functional features that make the connected devices easily controlled. The front-end

consists of a login page, an administration page, a settings page, a device details page, both individual and collective, as well as a dashboard page containing system and device diagnostics.

3.2.1 Purpose The purpose of the front-end is to provide a user friendly graphical user interface which allows the user to interact with the smart home power management system. It provides a means of communication between the front-end and the back-end of the system in order to display the relevant information or changes to the user. In addition it allows the user to control the devices remotely through the platform from any location.

3.2.2 Scope The implementation of the front-end is for a smart home power management system which monitors device usage and power consumption. It therefore allows the user the ability to alter power usage as well as configure alerts and manage the settings of the front-end. The front-end also allows the user to initiate remote triggering of connected devices such as lights and kitchen appliances. It is also designed to display the outcomes of triggering motion detectors and temperature sensors.

3.2.3 Definitions, Acronyms and Abbreviations

- **CSS** - Cascading Style Sheets which is a stylesheet language that sets up the appearance of the HTML [2].
- **HTML** - Hypertext Markup Language which is used to generate web pages and its objects form the basis of all websites [3].
- **Django** - A framework used to compile and render the HTML pages.
- **Bootstrap** - provides a html and css template for web development
- **Javascript** - A high level and dynamic programming language that is used with HTML and the web [4].

3.3 Use Cases

The use cases for the front-end are as given in the tables below.

Table 1 : Table illustrating the details of the use case for trigger switches

Use Case	**Trigger Switches**
Description	The goal of the trigger switching is to allow the users to manually control the connected devices' state through a web-page platform.
Actors	Users within a household
System Under Design	Web-page Platform for trigger switching
Preconditions	The user has successfully passed the authentication process via the login system.
Assumptions	It is assumed that the user has a basic understanding of how to access a web-page platform using technological devices.
Non-Functional Requirements	The switches are required to be instantaneously responsive when triggered. In addition the triggers must be able to tolerate system malfunctions. .
Frequency	The usage frequency of the trigger switches are on a hourly to daily bases
Priority	It is of priority, as it is the most accessed functionality in the Smart Home Power Management system. It is also of importance since the functionality allows the user to override the time scheduling settings.

Table 2 : Table illustrating the details of the use case for the addition and removal of household devices

Use Case	**Addition and Removal of Devices**
Description	The purpose of the feature is to enable the user to add and remove devices on the system.
Actors	Administration user
System Under Design	Addition and Removal web-page platform
Preconditions	The user has passed the authentication procedure and they have administration rights.
Typical Flow of Events	1. User logins into the system 2. Accesses the administration web-page 3. Selects the *Add Device* option 4. Provides device details in the allocated input boxes 5. Once satisfied with the details of the devices, then select the *save* button to save.
Assumptions	There is at least one allocated administration user in the household.
Non-Functional Requirements	The addition and removal of the devices in the system is expected to be instant, to prevent any confusion to any of the users.
Frequency	The frequency of usage of the addition and removal feature is low, since the devices will be added during the installation of the system. If there are any device removals, this will only occur on occasion when the device/s are defective.
Priority	This is a high priority feature in the system, as this system is reliant on the ability of the administration user to add and remove devices. Without this feature, the system becomes redundant.

Table 3 : Table illustrating the details of the use case for the authentication procedure

Use Case	**Authentication Procedure**
Description	The aim of the authentication procedure is to ensure security on the system which results in preventing unwanted user access.
Actors	User that wishes to access the system
System Under Design	Authentication procedure
Preconditions	The user must have previously created an account on the system.
Typical Flow of Events	1. User accesses login page. 2. User provides valid username in the allocated input box. 3. User provides valid password in the allocated password input box. 4. Select *Login* button to submit details for validation.
Assumptions	It is assumed that the user has an already existing account. In addition it is assumed that once the user has logged into their account, they remain logged in.
Non-Functional Requirements	The procedure is required to hide the password provided by the user and allow immediate access to the user once the provided information is verified.
Frequency	Moderate usage due to the assumption that once the user has logged in, the user remains logged unless they are accessing the system from a new mobile or desktop device.
Priority	The authentication procedure has high priority, due to the fact that the user is required to log in to access the system. This is important in terms of security, which prevents unwanted users from accessing it.

Table 4 : Table illustrating the use case of settings procedure

Use Case	**Settings Procedure**
Description	The focus of the settings page is to enable the user to alter both the settings of the connected devices as well as the appearance of the web-page to the user's needs.
Actors	Household users
System Under Design	Settings Procedure
Preconditions	The user is required to have passed the authentication procedure by the logging into the system.
Typical Flow of Events	1. Accessing the settings page 2. Selecting either the settings for the connected devices or the appearance of the web-page. 3. The settings for the connected devices is performed by selecting the desired device that the user wishes to alter. 4. The appearance settings is opened by selecting the *Appearance* tab, in which this takes the user to the web-page that enables them to alter the settings to their customized needs. 5. For both the settings the user is required to select the *save* button to ensure that the alterations made are saved.
Assumptions	There are default settings implemented for each device during the system installation.
Non-Functional Requirements	The settings page is required to be easily accessible and understood by the user to ensure that it is user friendly.
Frequency	Moderate usage, since the settings will be performed during installation. It will only be accessed by the user should the user wish to adjust the settings due to changes done in the system.
Priorities	High priority, since this establishes the operations of each device. This includes timing, automating devices and power monitoring.

Table 5 : Table illustrating the use case of the device details page

Use Case	**Device Details Page**
Description	The goal of the device details page is to inform the user of the details of each individual connected devices. In addition it displays the power consumption of each connected devices, as well as the total power consumption in a graphical format.
System Under Design	Device Details Page
Preconditions	The devices are required to be connected to the system and the user must have passed the authentication procedure to have access the device details.
Assumptions	It is assumed that the devices are connected to the system, and that the user has an account.
Non-Functional Requirements	It is required to instantly display the device details in a method that is easily understood by the user.
Frequency	High usage, since the main reason for the Smart Home system is to manage the power consumption and save, the details device page is predicted to be accessed often to enable the user to analyse the power consumption in their home.
Priority	High priority, because this is a crucial component of the system, which entails the details of the each device and its power consumption.

3.4 *Design Overview*

The Design Overview illustrates the architecture of the front-end implementation of the system. It further places it into the context of external systems, which enables the reader of this document to understand the overall details of the front-end design.

3.4.1 Overall Architecture The overall architecture of the front-end is shown in Figure 3.4.1.

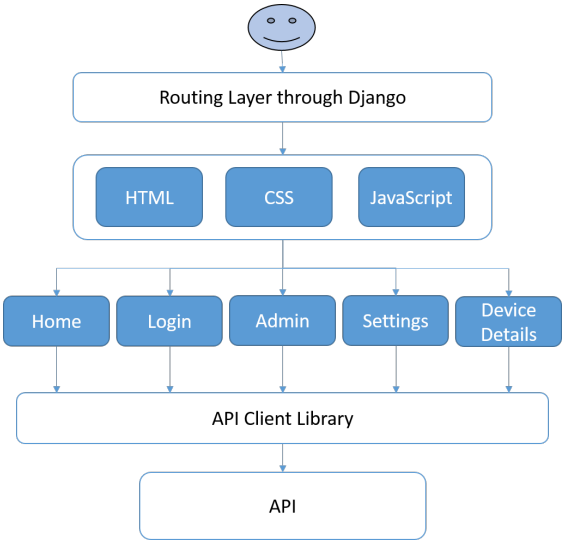

Figure 6 : Figure showing the front-end system architecture

3.4.2 System Interfaces An interface is a shared boundary across which two separate components exchange information

External User Interface Requirements:

1. *User interfaces*: The user interface enables the user to interact with they system using a graphical user interface (GUI). This can occur without any knowledge of the back-end functionality. The GUI is in the form of a web page which provides a platform for the user to communicate with multiple connected devices. The interface requires a digital device such as a mobile or desktop device which consists of a screen and keyboard. The GUI is designed in such a way as to create an easily navigable display to ensure it is user friendly.
2. *Software Interfaces*: The front-end is required to interact with the management system in order to obtain data from the database. This is done by using the supplied Django library namely, *django.shortcuts*, in which the functions `render()` and `get_object_or_404()` are used to link the back-end with the front-end HTML.
3. *Communication Interfaces*: The communication between the management system back-end and the front-end is performed using JavaScript. The connection enables the user to interact with the database by updating it with specific actions.

3.4.3 Constraints: The main constraint that is experienced in terms of front-end design is time which is a significant limiting factor. The user must be able to access the interface through a web browser which requires internet connectivity.

3.4.4 Assumptions: It is assumed that the interface is designed for users that have previous knowledge and experience in navigating through a webpage platform. The user us expected to know how to create an account as well as log into the account for future use. Individual assumptions regarding each use case have been discussed previously.

3.4.5 Dependencies: The front-end of the system is dependant on a single database management system in the back-end. Without the information that can be pulled from the database, the front-end becomes essentially useless.

3.5 *Front-End Relevant Modules*

The following describes the relevant modules for the front-end implementation. The modules consists of switch-buttons, a list of all device details, individual device details, and administration. The following modules are implemented using CSS, HTML, JavaScript and the Django framework.

3.5.1 HTML Module: The HTML module constructs the basic layout of how the information is displayed to the user in the front end. Although the HTML module is crucial in layout of the web-page, it is assisted by the CSS module. The HTML presents the information obtained from the database in a specific manner that can be easily understood by the user.

3.5.2 CSS Module: The CSS Module is an extension of the HTML module, in which it assists the HTML layout by arranging the information in an orderly manner that is appealing and can be easily navigable by the user. The CSS does not convey any information to the user, but it does highlight the importance of specific information that is presented by the HTML module.

3.5.3 JavaScript Module: JavaScript is a high level and dynamic programming language. The JavaScript module is also an extension of the HTML module, in which it enables the user to update the database dynamically when the user performs specific actions in the front-end.

3.5.4 Web-Page Layout: The web-page layout is implemented using the Twitter Bootstrap Creative Theme, in which the HTML and CSS modules are the base modules that dictate the presentation of the web-page. The theme is to ensure that the web-page is user friendly as well as appealing. The layout is simple, in which the home-page illustrates all the devices that are connected to the Smart Home Power System along with the toggle switches for each connected device.

3.5.5 Device Switch-Button: The purpose of implementing the switch-button is to allow users to control the linked devices through the platform. The switch-button has two states namely, *on* or *off*. It can only be one or the other and cannot be both. The implementation of the button will only show one state at a time, to eliminate any confusion to the user. In doing so, this allows the implementation to be user friendly. The on and off switch-button can be seen in Figures 7 and 8, respectively.

Figure 7 : Figure illustrating the switch-button for 'on' state

Figure 8 : Figure illustrating the switch-button for 'off' state

3.5.6 Device Details and Diagnostics: The details and the diagnostics of each connected device are illustrated on the details page, in which it informs the user of the power consumption of each device, as well as giving an overview of the operational times of the device. The purpose of the details and the diagnostics is to inform the user about the accumulated data measured in the system. This is achieved using a graphical format which contains easily understood information.

3.5.7 Settings: The settings page is implemented in such a way that it allows the user to easily navigate to the desired device. The settings page contains various settings that allows the user to adjust and modify the chosen device to their desired operational adjustments. These settings consist of adjusting the automated devices with a time schedule, modifying the intensity of specific devices, and allowing the user to trigger the devices.

3.5.8 Administration: The simplicity of the administration page is crucial as it is required to be easily understood by the user. The page allows the user to add and remove devices from the smart home power management system. To ensure that the page is easily understood, the conventional *add* and *remove* graphical figures are used. The instructions are clearly stated to prevent any confusion to the user. Aside from allowing the user to add and remove devices, the administration page has the functionality of adding and removing users to the system. The users known to the system are granted access to remotely use it to control the connected devices.

3.5.9 Login: The login page is designed to be simplistic, in which it only contains two input boxes for a username and password. The *Login* button is centred and explicit, in which it allows the user to easily fill in their details and login into the system. The login page is straight forward and conforms to the general format of other web-pages that require authentication.

3.6 Non-functional Requirements

3.6.1 Performance The front-end web page needs to be able to respond quickly and in real time. It is also likely that it will need to undergo several updates triggered by user command.

3.6.2 Safety In terms of safety, the front-end needs to be able to notify the user in the event that a connected device malfunctions. This allows the user to take action depending on the notification.

3.6.3 Security The front-end reinforces security of the system by having login and authentication procedures to prevent unwanted external interference from non-household members.

3.6.4 Software Quality Attributes The front-end needs to be user friendly in a way that the user does not require any supporting documentation or prior knowledge of the system to navigate through the web page.

3.6.5 Design Constraints The front-end should be built while taking full advantage of open source libraries and supporting software to avoid the need for licensing and additional costs.

II. BACK-END

3.7 Design Overview

This will give a brief introduction to the smart home power management system back-end design.

3.8 Definitions

1. **Django** - A high-level Python web framework designed to be scalable and easy to use
2. **MQTT** - A lightweight publish/subscribe messaging protocol commonly used in embedded devices [5]
3. **RabbitMQ** - A robust message broker framework that supports MQTT [6]
4. **Celery** - A distributed task queue for Django that uses RabbitMQ as its default message broker [7]

3.8.1 System Architecture: The system architecture will give an overall view of the entire back-end system designed.

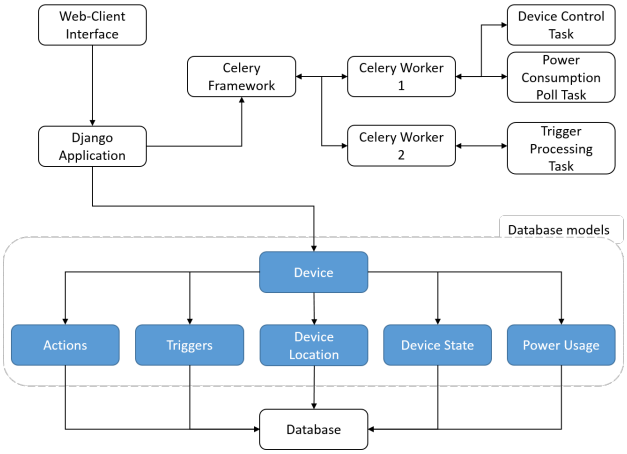

Figure 9 : System architecture diagram

3.8.2 Constraints: The main constraint that is also experienced in terms of back-end design is time, which is a significant limiting factor. Additionally, the network bandwidth may be constrained due to large amounts of data generated by many devices. A device which meets the system requirements set out in the Software Requirements Specifications.

3.8.3 Assumptions: It is assumed that all the connected devices will be on the same network and that they will implement the MQTT protocol. It is also assumed that a database will have to be significantly large due to all the data that these devices are generating.

3.8.4 Dependencies: The back-end of the system is dependant on the RAM, harddrive, as well as CPU of the smart home server as well as on the supply of electricity to the household.

3.9 Back-End Relevant Modules

- **Django Application:** This section is the main application responsible for serving the front end and controlling the database interactions. It uses the Django framework in order to implement this functionality. Django uses Python classes to define models in an Object-Relational Mapping, corresponding to tables in a database. These models will be described in detail in the following section. By defining these models in Python code, Django automatically provides an administration interface and abstracts away low-level database operations.
- **Database Models:** For more details regarding the database models and relationships, see Figure 10.
  - *Device Model:* This model represents a singe device. It keeps track of relevant device information, including name, IP address, location, and state. It is the core model of the database system. Two devices cannot have the same name and location attributes. Additionally, no two devices can have the same IP address
  - *Device Location:* This model represents the location in the household where a device can be found. A location can store many devices, but a device can only be in one location
  - *Device State:* This model represents the various power states that a device can be in. It is initially populated with the values "on" and "off," however the user still has the ability to add more depending on device capability
  - *Triggers:* This model represents the various triggers that the system can incorporate, namely, motion, temperature, and light It stores an input device ID, an output device ID, and a string of Python code which will provide the conditions for the trigger to fire as well as the action to apply.
  - *Actions:* This model represents the various actions that can be taken by the system when a trigger is fired. Any number of Triggers can invoke the same Action
  - *Power Usage:* This model represents the power usage of the household devices at points in time. It stores power usage data for certain time intervals for each device, allowing time-series analysis of device power usage
- **Celery:** This is an asynchronous task queuing framework that supports scheduling and real-time operation. The Django application uses Celery in order to manage long running tasks such as trigger monitoring as well as controlling and monitoring devices. Using the default RabbitMQ message broker, and the MQTT messaging proto-

col, Celery communicates with Internet of Things devices over the network.
– *Celery Worker 1:* This worker will handle tasks requested by the user, such as changing the device state, or polling a device for power usage data. These tasks are typically executed and completed quickly.
– *Celery Worker 2:* This worker will periodically execute the trigger processing task by fetching all the triggers from the database and executing those actions which correspond to the trigger that was flagged. This task may run for an extended period of time, hence the need for the second worker.

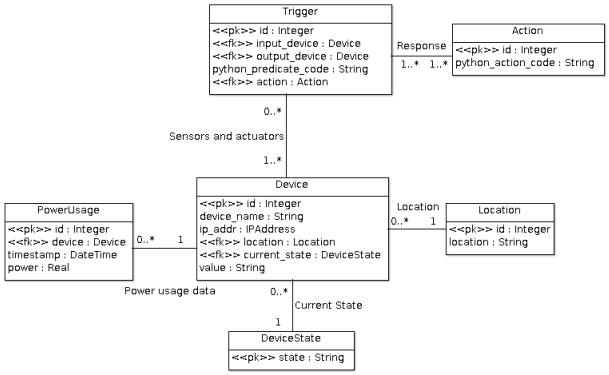

Figure 10 : The entity relationship diagram showing the relationships between database models as well as the primary and foreign keys for each model

3.10 Non-Functional Requirements

3.10.1 Performance Requirements: The smart home power system is an application and system that is designed for efficiency. This means that the application as well as the connecting household components need to respond quickly and in real time.

The application will most likely undergo several updates and changes for the user's benefit. The system is required to update on user command. Updates will include further improvements to the application as well as fixes for issues that arise in operation.

The lightweight MQTT protocol is chosen to minimise network traffic and processing requirements so that more devices can be added without performance degradation.

3.10.2 Safety Requirements: The application as well as the system components need to take certain safety concerns into account.

Due to the fact that the application is essentially controlling most of a home's appliances and electricity usage, it is important that it monitors everything it is controlling to prevent problems that may arise. The application needs to ensure that any device connected in the system does not reach a dangerous level of usage. In this case, the application should either switch off the device in question or notify the user that something is not functioning correctly and needs to be addressed.

3.10.3 Security Requirements: Due to the fact that the system controls a user's home, it therefore requires security considerations to be taken into account.

The system needs to ensure that only authorised users have access to the application to prevent external parties gaining control of the connected components within the house. This can be done with a signup, login and authentication process. The components within the house that the application connects with, need to be protected from external parties and therefore they must be explicitly authenticated.

3.11 Supplementary Documentation

3.11.1 Tools used to create diagrams:

- System overview diagrams: Microsoft PowerPoint
- UML diagram: ArgoUML
  http://argouml.tigris.org
- Burndown chart: Burndown for TRELLO
  https://www.burndownfortrello.com

4. PROTOTYPE IMPLEMENTATION

This section documents the implementation of the first prototype of the Smart Home Power Management System.

4.1 Key Modules Selected for Project Illustration:

A few class modules have been written using Django, a high-level Python Web framework, in conjunction with a Bootstrap Theme, a powerful front-end framework, for the first prototype:

1. Scrollable web-pages, with a page header and overall device information
2. The addition and removal of device/s, along with corresponding specifications
3. The adding and removing of user account/s in the system
4. Viewing the connected devices along with its details
5. Toggling between the individual device's state
6. Enable the user to log in into their user account

4.2 Testing:

Simple tests have been written and conducted along the development process of this prototype:

4.2.1 Front-end: There are 2 sets of testing in the front-end:

1. Manual user testing
2. Automatic, predefined written/coded tests

Manual User Testing: Every possible page and button combination have been manually tested.

Written tests:

1. Testing that the homepage successfully loads when it is called
2. Check that the details page of a specific device loads when it is called
3. Test that the device list page loads successfully
4. Check that the login page loads successfully

4.2.2 Back-end: The tests that have been written to demonstrate the functionality of the back-end are:

1. Check that a device can be added to the database
2. Test that the system does not allow datatypes which do not fit the device specifications. For example, a user cannot enter a device IP address as "3498347", but instead it must be an IPv4 or IPv6 address
3. Check that a certain home location cannot contain two devices with the same names. For example, a kitchen cannot contain "kettle 1" and "kettle 1"
4. Test that no two devices can have the same IP address

5. SPRINT PLANNING

All work for the prototype development was conducted as a well collaborated team. Upon receiving information complying to the deliverables of this lab, a three-hour sprint planning meeting was called in order to conclude the following:

1. The product backlog
2. The sprint goals
3. The action plan of how to achieve this goal by the target date (4 April 2016)

As is required, the product backlog was first created in order to decide which of these tasks will be transferred to the sprint backlog for the latest product increment. The product backlog drawn up on 24 March 2016 is as shown below (ordered by priority):

- As a user I would like to be able to add/remove a device
  - Create form to add new device
  - Create list of devices which can be disabled/removed
- As a user, I would like to be able to manually switch on and off appliances remotely
  - Create page header menu
  - Create sidebar and container
  - Create page tabs
- As a user I would like to view which devices are using power, and how much they are using
  - Create dummy data on backend
  - Create basic power usage graph for each device
- As a user I would like to personalise various situations which will trigger an action
  - Create list of different triggers
  - Create trigger settings page
  - Create form to choose triggers
- As a user I need to choose which events should result in an immediate alert or notification on my system
  - Logic to monitor status of different subsystems
  - Configure how alert will be sent to user and connect accordingly
- As a user it is convenient for me to be able to define webpage settings in order to enhance the power management system experience
- As a developer I want to be able to easily run the server using Docker
  - Set up docker
  - Configure docker to run server and database

Upon this discussion, with two weeks remaining till project submission, a sprint planning meeting was held in order to discuss and plan the work to be performed in the first sprint which would have a timebox of one week. A sprint backlog was drawn up on Trello (a software collaboration tool). The sprint backlog is a basic list of the tasks that must be implemented by the team in order to deliver a functional product increment at the end of that sprint [8].

The screenshot below illustrates the items from the product backlog that were moved to the sprint backlog on this day.

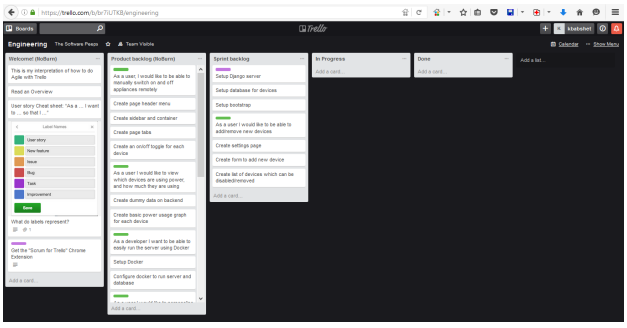

Figure 11 : Screenshot of the Trello page drawn on the first day of the first sprint for the initial prototype implementation

After the first week, a second sprint planning was held for the following week-long sprint. The screenshot below illustrates the new items that were now moved to the next sprint backlog.

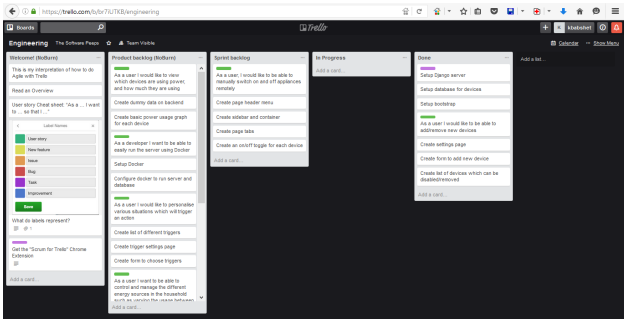

Figure 12 : Screenshot of the Trello page drawn on the first day of the second sprint of the for the initial prototype implementation

Tasks were delegated and the product was worked on and shared amongst the team using GIT, while sprint retrospective meetings were held once every week on Thursdays.

As was decided at the commencement of this project, Alice Yang and Daniel Weinberg have been assigned to the front-end development, whereas Ari Croock and Kanaka Babshet have been assigned to the back-end development.

6. SPRINT RETROSPECTIVE

The burndown charts for Sprint 1 and 2 are shown in Figure 13 and Figure 14 . As can be seen, the actual time spent on the planned tasks exceeded the estimated time overall. This can be attributed to a number of reasons, namely time constraints and inexperience with the chosen software frameworks. Over time the scrum master and software engineers will become better at estimating time required for tasks, and will gain experience with relevant frameworks, making routine tasks quicker and easier to perform.

The team should continue having productive and efficient sprint planning and sprint retrospective meetings at the beginning and end of each sprint.

Due to time constraints and other projects being worked on simultaneously, daily scrum meetings could not be held (meetings were held roughly every two days). Preferably these meetings should take place daily in order to encourage communication and help reduce the time spent on solving technical issues. The scrum master should also encourage continuous feedback during the sprint as ideas may be forgotten by the time the retrospective meeting occurs.

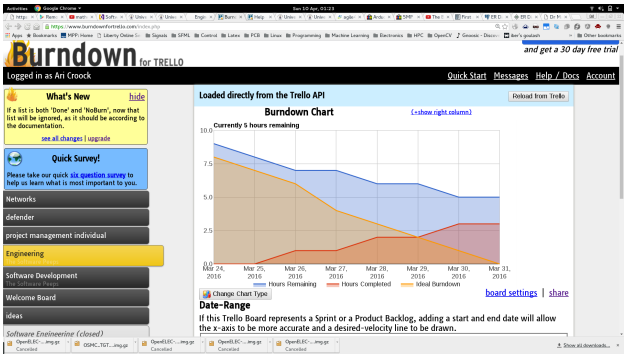

Figure 13 : Burndown chart for Sprint 1: 24 March 2016 - 31 March 2016

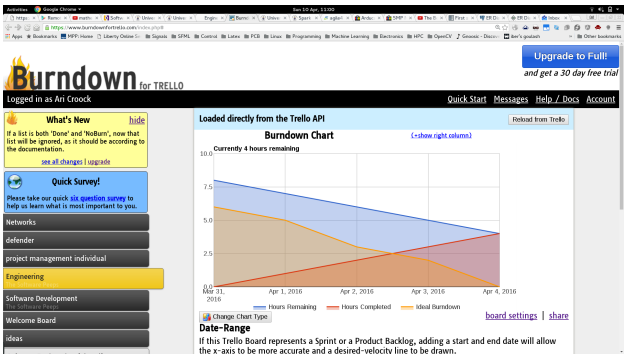

Figure 14 : Burndown chart for Sprint 2: 31 March 2016 - 4 April 2016

REFERENCES

[1] BusinessDictionary, "stakeholder," 2016. [Online]. Available: http://www.businessdictionary.com/definition/stakeholder.html

[2] w3schools, "Css tutorial," 2016. [Online]. Available: www.w3schools.com/css/

[3] ——, "Html tutorial," 2016. [Online]. Available: www.w3schools.com/html/

[4] ——, "Javascript tutorial," 2016. [Online]. Available: www.w3schools.com/js/

[5] "Mqtt version 3.1.1," OASIS, 2016. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[6] "Rabbitmq," Pivotal, 2016. [Online]. Available: https://www.rabbitmq.com/

[7] "Celery: Distributed task queue," Celery Project, 2016. [Online]. Available: http://www.celeryproject.org/

[8] L. Felix, "9 tips for creating a good sprint backlog," 2016. [Online]. Available: https://www.scrumalliance.org/community/articles/2009/march/9-tips-for-creating-a-good-sprint-backlog

Appendix A

A survey was conducted prior to implementation and design of the smart home power management system. The figures below illustrates the results corresponding to every question.

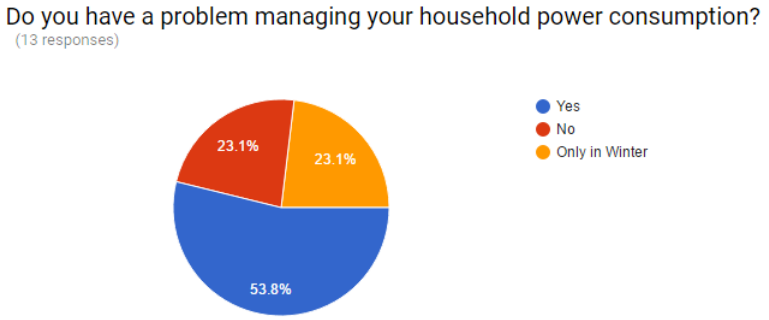

Figure 15 : Pie chart illustrating the results for Question 1 of the smart home power management survey

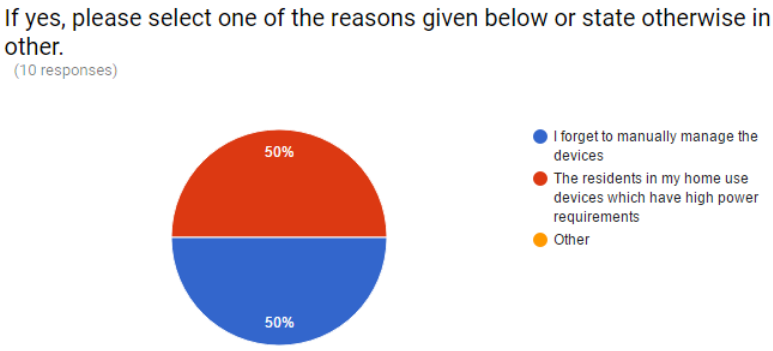

Figure 16 : Pie chart illustrating the results for Question 2 of the smart home power management survey

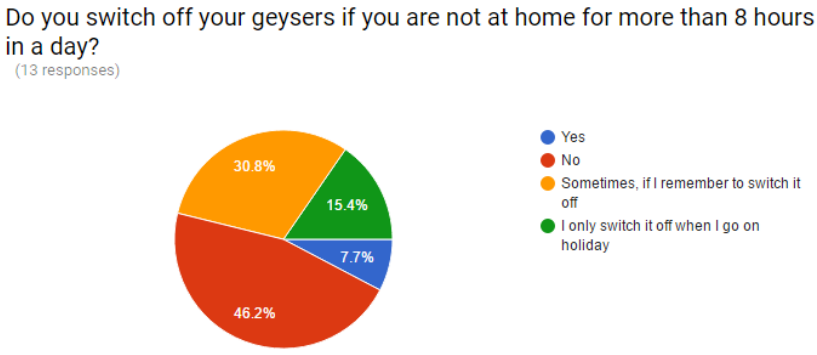

Figure 17 : Pie chart illustrating the results for Question 3 of the smart home power management survey

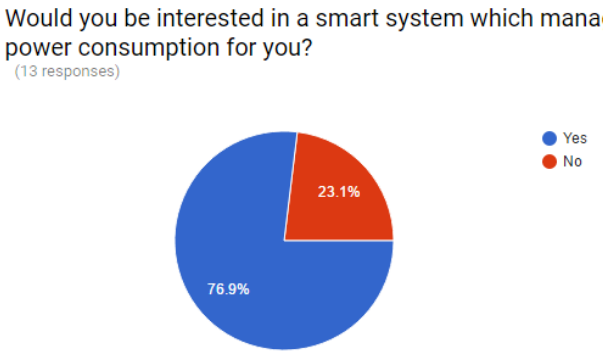

Figure 18 : Pie chart illustrating the results for Question 4 of the smart home power management survey

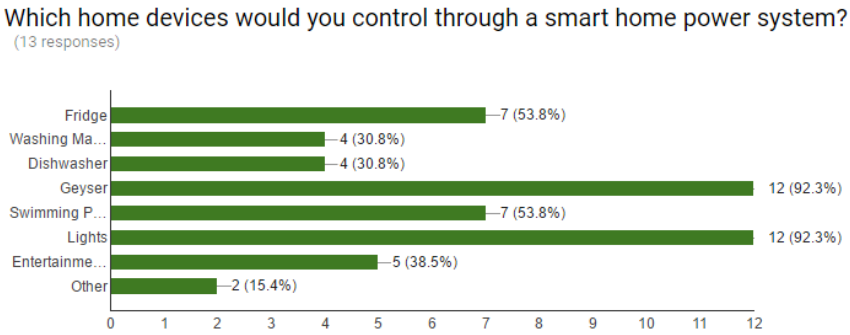

Figure 19 : Bar graph illustrating the results for Question 5 of the smart home power management survey

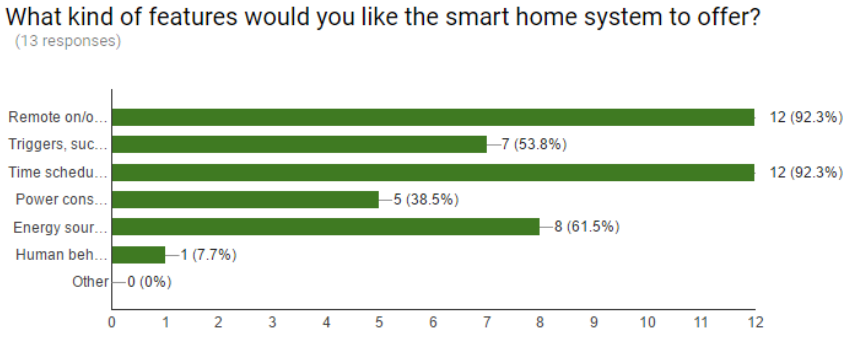

Figure 20 : Bar graph illustrating the results for Question 6 of the smart home power management survey