EISH: Energy Intrinsic Smart Home.
Software Requirement Specification

MonoToneID

April 2019

Contents

1 Introduction

In recent years, the Internet of Things ("IOT") devices have entered the consumer space with products such as SmartThings, Apple Homekit, etc that are able to give you control over your home devices and also provide you with information about your devices power consumption. Simultaneously, there is a global push towards renewable energy where consumers are able to generate and store their own energy.

One area that has not been fully explored is connecting the generation and consumption management of home automation systems. While solar panels and and backup batteries are able to measure consumption and control energy storage, they have no insight or control over the consumption, and while your smart devices can often measure and control their own consumption they cannot determine how much power is available at their disposal.

The Energy Intrinsic Smart Home Management System (EISHMS) is a software solution that aims to connect the generation and consumption management of home automation systems. EISHMS should be able to monitor a smart homes' energy generation capacity, while also monitoring the consumption of the energy. Additionally it will incorporate an intelligent system that will be able to measure and predict the energy generation capacity, while also learning the usage patterns of specific devices to prioritize devices and optimize energy consumption.

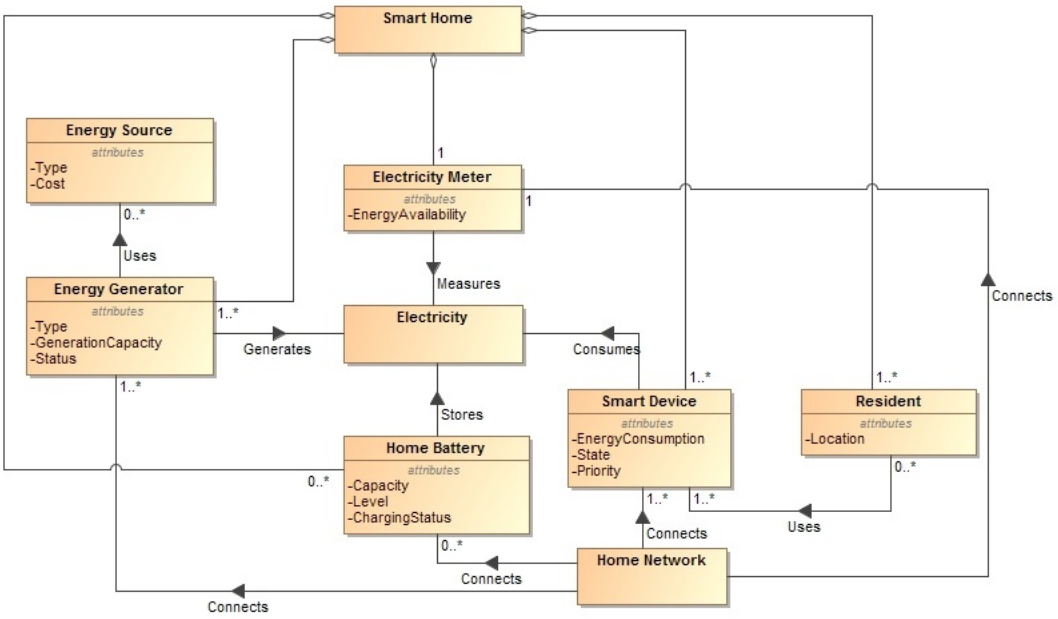
The EISHMS is particularly relevant in South Africa where Eskom is experiencing severe generation difficulty making the grid unreliable. Those who can afford it are investing in alternate energy generators (solar panels or diesel/petrol generators). EISHMS is giving users peace of mind knowing that you don't have to worry about manually managing your energy generators and devices, as it will do all that for you while ensuring efficient and optimised usage of energy.

The scope of the EISHMS will cover the management of energy generation and consumption of a single household that is connected to a Local Area Network (LAN).

Glossary

IOT	Internet Of Things
EISHMS	Energy Intrinsic Smart Home Management System
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
EISH	Energy Intrinsic Smart Home
UI	User Interface
MGS	Monitor Generation Subsystem
MCS	Monitor Consumption Subsystem
CGS	Configuration Generation Subsystem
EPS	Energy Prediction Subsystem
ESS	External Services Subsystem
CES	Cost Estimation Subsystem
LS	Learning Subsystem
MRS	Monitor Resident Subsystem
NS	Notification Subsystem
OS	Optimization Subsystem
DBMSS	DBMS Subsystem
UCS	Usage Controller Subsystem
DCS	Device Configuration Subsystem
HTTPS	Hypertext Transfer Protocol Secure

Domain Model



2 User Characteristics

2.1 Resident

It is assumed that a resident using the EISHMS resides within a smart home. Such a resident has a smart home with smart devices connected to the smart home. The resident will additionally have one or more methods of energy generation. The resident will interact with the system in order to monitor device consumption and energy generation from different energy generators connected to the smart home.

2.2 Guest

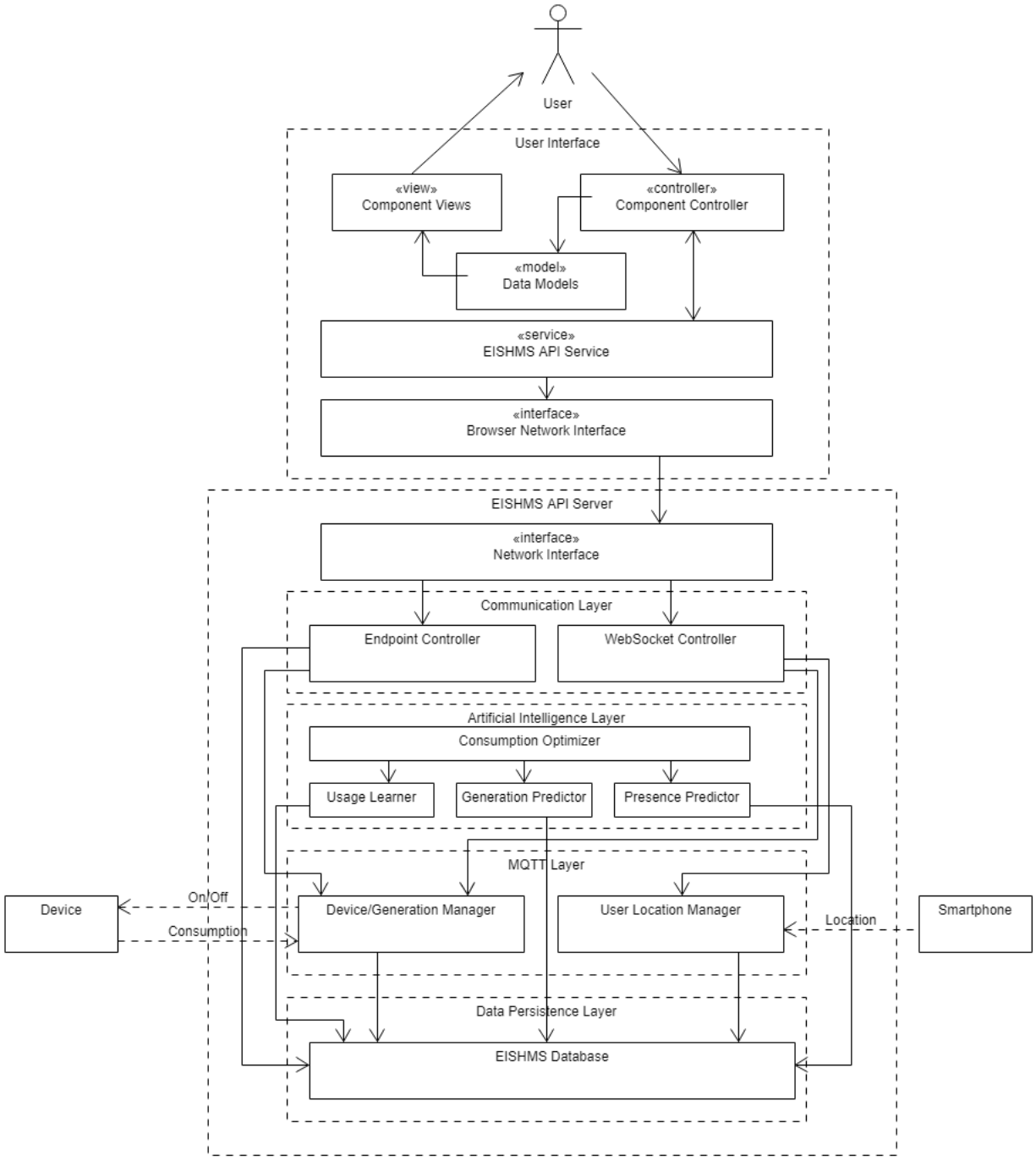
A Guest is a non-resident of the smart home who is present within the smart home and is granted limited privileges by the Resident. Such a guest will interact with UI and MCS subsystems to monitor their consumption.

2.3 System Administrator

System Administrator is responsible for the upkeep, configuration, and reliable operation of the EISHMS.

3 Architectural Design

3.1 Architectural Style



3.2 Architectural Style Description

The EISHMS is made of a custom architecture. The core architecture is made up of MVC architecture and within the controller of the MVC architecture, we have an n-tier component and event-driven component. The reason for choosing the above architecture is discussed below:

MVC

- Faster development process: MVC supports rapid and parallel development. One programmer can work on the view while another can work on the controller to create business logic of the web application.
- Support for asynchronous technique: MVC also supports asynchronous technique, which helps developers to develop an application that loads very fast.
- Modification does not affect the entire model: Modification does not affect the entire model because model part does not depend on the views part. Therefore, any changes in the Model will not affect the entire system.

N-tier

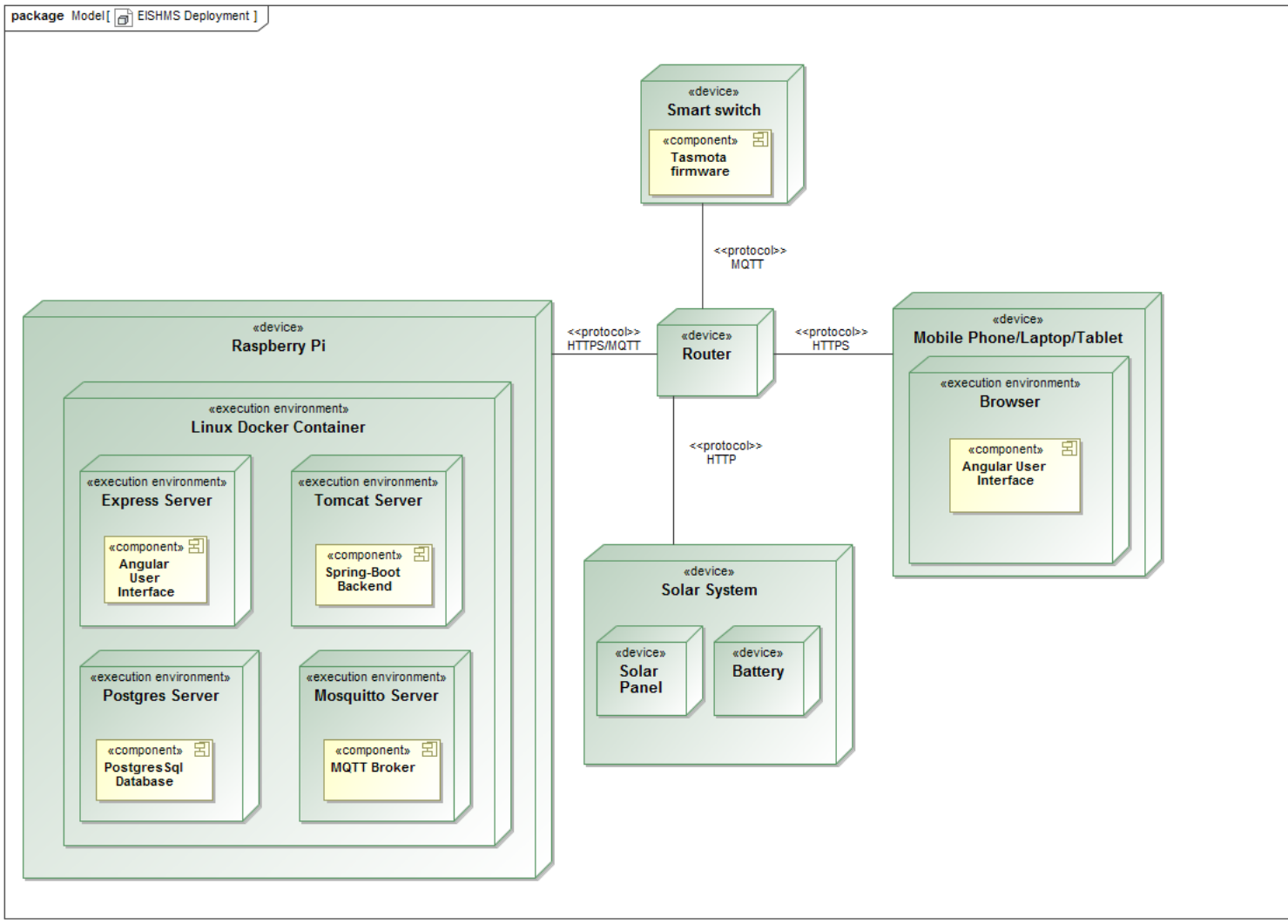
- Easier development of components: Developers can break the system up into different layers and work on them individually.
- low coupling and high cohesion of components: There is lower degree of dependency between components and components are lightweight so they do only what is required of them.

Event-Driven

- Highly scalable and flexible: Event-driven architecture scales well to a very large number of users and allows new services to be added without breaking the streams of data that are currently deployed.
- Lower bandwidth constraints: Communication only occurs when an event occurs so network traffic is kept low.
- Easier monitoring of clients and error detection: Events act as heartbeats of clients/devices so when events are not occurring it means there might be a problem.

3.3 Deployment Model

The EISH system will be deployed primarily using Docker Containers which encompass the various components of the system. Within the primary Linux container the various internal containers will communicate internally using their docker container names as they are on the same network.
The Raspberry Pi will communicate and transmit data with the router using HTTP and MQTT protocols. HTTP will transmit web requests and messages and MQTT will transmit information related to devices and generators connected to the system. HTTP will also be used to communicate between the various external API's used and with the users device.



3.4 Technology Requirements

The following technologies will be used to implement the system:

User Interface Technologies

This system will use Angular to create a SPA that the user will interact with.
Angular is chosen due to its ability to code in modular blocks and data binding for real-time and dynamic data viewing [4].
Angular is well-documented and is supported by Google, which ensures the availability of resources and a network of people and forums to help where needed.

Communication Technologies

- **MQTT**: MQTT transfers data as a byte array and publish/subscribe model, which makes it perfect for resource-constrained devices and helps to save battery.MQTT Protocol is easy to use. It is essential when response time, throughput, lower battery and bandwidth usage are critical[5].
- **HTTPS**: Hypertext Transfer Protocol Secure is an extension of the Hypertext Transfer Protocol. It is used for secure communication over a computer network which is required in our local area network.

Implementation Technologies

SpringBoot will be the main technology used for the implementation of the controller of the system. It was chosen because of its easiness to create stand-alone production grade spring based applications. SpringBoot helps ease Java-based applications Development, Unit Testing and Integration Testing Process. This helps increase productivity and reduce time to market.

Hardware and Operating Systems

The Energy Intrinsic Smart Home system will overall run on Raspberry Pi or similar single-board computers, which will serve as lightweight home server within the users' home. This is to enable and easily guarantee security and availability of the system at all times. These devices are also generally small and affordable and provide free hosting
The system will run on a Linux Docker container and each component of the system is also encapsulated in a docker container. Docker is used for virtualization and acts an executable package that can be installed on any appropriate server or preferred platform. [7] The Linux Operating System is ideal as it also lightweight, open source, versatile and extensible. The libraries it has and supports make it easier to support various containers and/or software which the system encompasses.

4 Requirements

4.1 Use Cases

User Interface

UC1 Configuring energy generators and smart devices.

- **UC1.1** Add and remove energy generators/smart devices.
- **UC1.2** View available energy generators/smart devices.

UC2 View information about energy generators and smart devices.

- **UC2.1** View energy generation information of individual energy generators.
- **UC2.2** View energy consumption information of individual smart devices.

UC3 View system notifications.

UC4 View cost estimate report.

UC5 Prioritize smart devices.

UC6 Override system decision.

Monitor Generation Subsystem(MGS)

UC7 Monitor energy generated by individual energy generators.

UC8 Store data collected from individual energy generators.

UC9 Monitor energy level of the home battery.

Monitor Consumption Subsystem(MCS)

UC10 Monitor energy consumed by individual smart devices.

UC11 Store consumption data collected from individual smart devices.

Energy Prediction Subsystem(EPS)

UC12 Retrieve energy generation capacity of a certain energy generator.

UC13 Retrieve information that affects energy generation capacity from ESS.

UC14 Predict total amount of energy that will be generated.

External Services Subsystem(ESS)

UC15 Retrieve weather information from external sources.

UC16 Retrieve electricity cost information from external sources.

UC17 Retrieve diesel/fuel cost information from external sources.

UC18 Retrieve user location information.

UC19 Process information gathered to an acceptable format.

Cost Estimation Subsystem(CES)

UC20 Retrieve cost information from ESS.

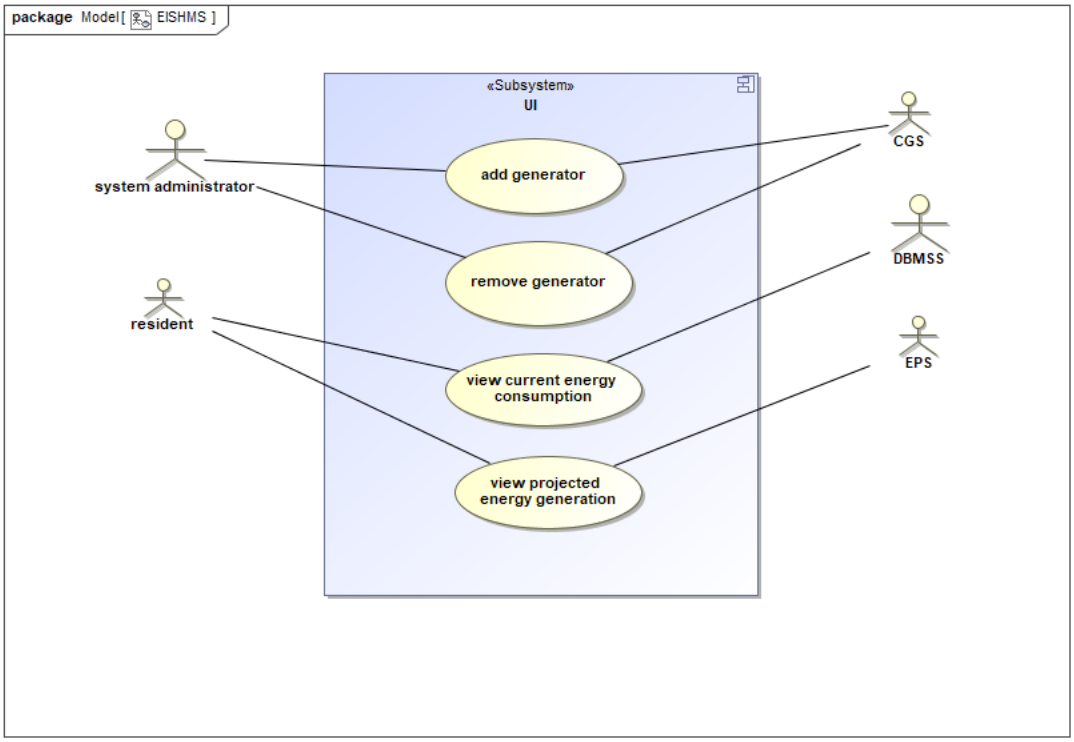
UC21 Estimate how much it will cost to use a certain energy source.

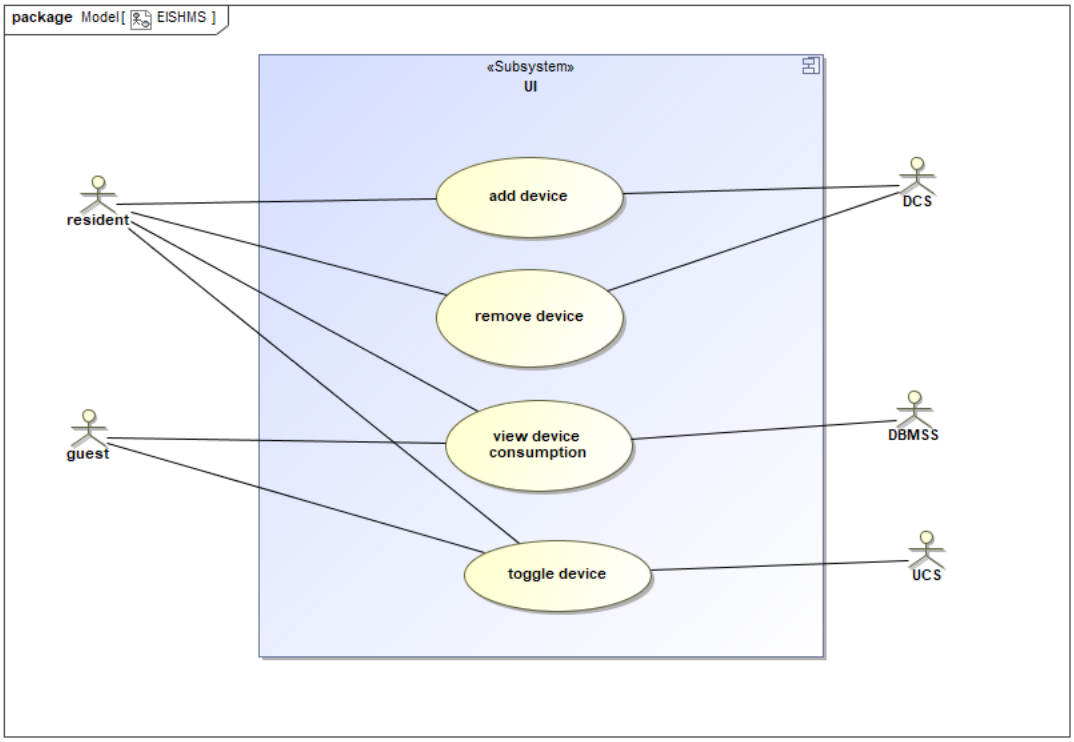
UC22 Generate a report for different costs.

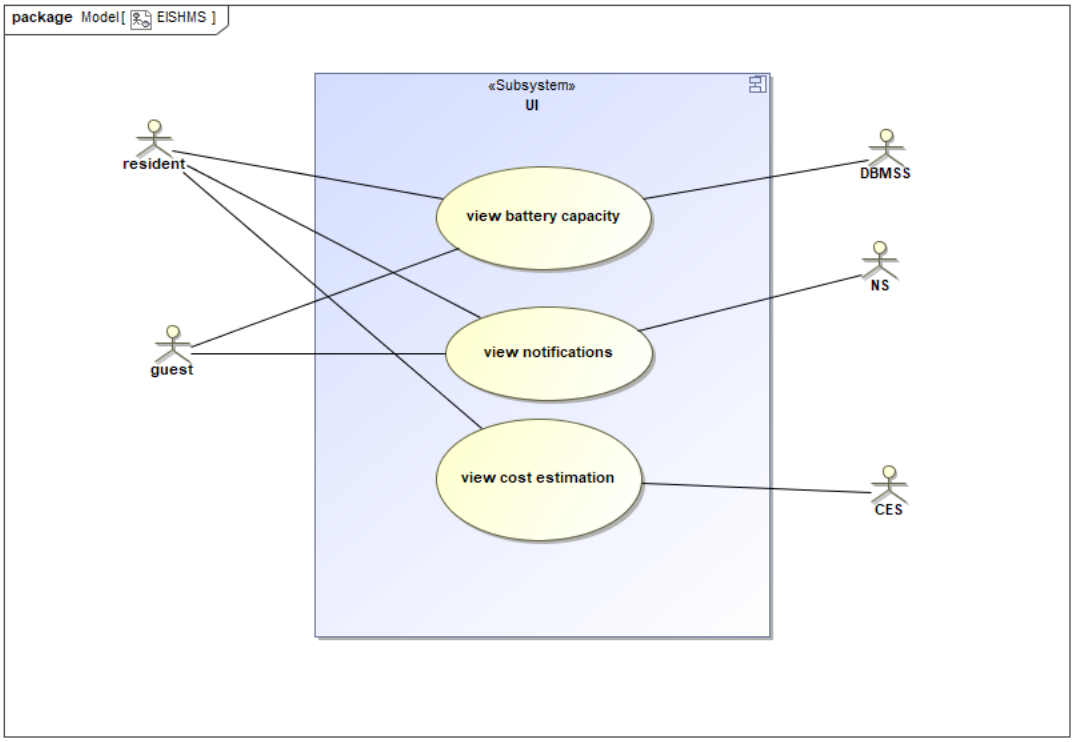
Monitor Resident Subsystem(MRS)

UC23 Retrieve user location information from ESS.

UC24 Check whether user is present in the smart home.




package Model [EISHMS]
«Subsystem»
UI
add device
remove device
view device consumption
toggle device
resident
guest
DCS
DBMSS
UCS


package Model [EISHMS]
«Subsystem»
UI
resident
guest
view battery capacity
view notifications
view cost estimation
DBMSS
NS
CES

4.1.1 Functional Requirements

The EISH Management System must fulfill the following functional requirements:

R1 The system must display the energy consumption and generation information of the smart house

R2 The system must be able to display notifications

R3 The system must monitor available capacity from individual energy generators

R4 The system must monitor total energy capacity/storage

R5 The system must monitor energy consumption of smart devices

R6 The system must configure EISHMS for various energy generators

R7 The system must provide the ability to switch between energy generators

R8 The system must predict how much energy will be produced by the energy generators

R9 The system must predict how much energy will be stored

R10 The system must request information from external services

R11 The system must standardise information from external services

R12 The system must aggregate information from external services

R13 The system must provide cost estimates for different energy sources

R14 The system must learn resident's usage patterns of the various devices

R15 The system must produce a priority list of resident devices

R16 The system must detect absence of resident(s) from smart home

R17 The system must detect presence of resident(s) within the smart home

R18 The system must detect which resident(s) is/are within the smart home

R19 The system must restrict device usage

R20 The system must adjust device usage

R21 The system must switch between energy generators

R22 The system must notify resident(s) when energy availability is constrained

R23 The system must notify resident(s) when switching energy source

R24 The system must gather information from required subsystems

R25 The system must suggest device usage based on priorities

R26 The system must decided which devices are deactivated according to battery capacity

R27 The system must decided which devices are de-/activated according to resident(s) preference

R29 The system must allow updates of smart device(s)

R30 The system must allow removal of smart device(s)

R31 The system must allow addition of new generator(s)

R32 The system must allow updates of generator(s)

R33 The system must allow removal of generator(s)

R34 The system must configure EISHMS for various smart device(s).

4.2 Subsystems

4.2.1 EISH Management System (EISHMS)

4.2.2 User Interface (UI)

User Interface (UI) is responsible for giving the user insight into the generation and consumption management subsystems.

R1 The system must display the energy consumption and generation information of the smart house

R1.1 UI must be able to display energy generation data

R1.2 UI must be able to display energy consumption data

R2 UI must be able to display notifications

4.2.3 Monitor Generation Subsystem (MGS)

Monitor Generation Subsystem (MGS) is responsible for observing, checking and keeping continuous record of energy generator and smart home battery.

R3 MGS must monitor available capacity from individual energy generators

R4 MGS must monitor total energy capacity/storage

4.2.4 Monitor Consumption Subsystem (MCS)

Monitor Consumption Subsystem (MCS) is responsible for observing, checking and keeping continuous record of energy consumption of smart devices.

R5 MCS must monitor energy consumption of smart devices

4.2.5 Configuration Generation Subsystem (CGS)

Configuration Generation Subsystem (CGS) is responsible for the arrangement of different energy generators.

R6 CGS must configure EISHMS for various energy generators

R7 CGS must provide the ability to switch between energy generators

4.2.6 Energy Prediction Subsystem (EPS)

Energy Prediction Subsystem (EPS) is responsible for predicting energy generation and storage.

R8 EPS must predict how much energy will be produced by the energy generators

R9 EPS must predict how much energy will be stored

4.2.7 External Services Subsystem (ESS)

External Services Subsystem (ESS) is responsible for collecting the required information,aggregating it and ensuring that this information conforms to a format.

R10 ESS must request information from external services

R11 ESS must standardise information from external services

R12 ESS must aggregate information from external services

4.2.8 Cost Estimation Subsystem (CES)

Cost Estimation Subsystem (CES) is responsible for roughly calculating costs related to different energy sources.

R13 CES must provide cost estimates for different energy sources

4.2.9 Learning Subsystem (LS)

The Learning Subsystem (LS) is responsible for acquiring knowledge about usage patterns and priorities of devices.

R14 LS must learn resident's usage patterns of the various devices

R15 LS must produce a priority list of resident devices

4.2.10 Monitor Resident Subsystem (MRS)

The Monitor Resident Subsystem (MRS) is responsible for observing, checking and keeping continuous record of resident(s) within the smart home.

R16 MRS must detect absence of resident(s) from smart home

R17 MRS must detect presence of resident(s) within the smart home

R18 MRS must detect which resident(s) is/are within the smart home

4.2.11 Usage Controller Subsystem (UCS)

The Usage Controller Subsystem (UCS) is responsible for controlling smart devices based on information received from OS.

R19 UCS must restrict device usage

R20 UCS must adjust device usage

R21 UCS must switch between energy generators

4.2.12 Notification Subsystem (NS)

The Notification Subsystem (NS) is responsible for sending notifications to resident(s) via push notifications.

R22 NS must notify resident(s) when energy availability is constrained

R23 NS must notify resident(s) when switching energy source

4.2.13 Optimization Subsystem (OS)

The Optimization Subsystem (OS) is responsible for optimizing consumption and costs of energy within the smart home.

R24 OS must gather information from required subsystems

R25 OS must suggest device usage based on priorities

R26 OS must decided which devices are deactivated according to battery capacity

R27 OS must decided which devices are de-/activated according to resident(s) preference

4.2.14 DBMS Subsystem (DBMSS)

DBMS Subsystem (DBMSS) is responsible for maintaining a database for the energy generators and smart devices.

R28 DBMSS must allow addition of new smart device(s)

R29 DBMSS must allow updates of smart device(s)

R30 DBMSS must allow removal of smart device(s)

R31 DBMSS must allow addition of new generator(s)

R32 DBMSS must allow updates of generator(s)

R33 DBMSS must allow removal of generator(s)

4.2.15 Device Configuration Subsystem (DCS)

Device Configuration Subsystem (DCS) is responsible for the configuration of different smart device(s).

R34 DCS must configure EISHMS for various smart device(s).

5 Non-Functional Requirements

5.1 Quality Requirements

Q1 Performance

- **Q1.1** The system must respond in no longer than 3 seconds
- **Q1.2** The Mosquitto MQTT broker running on a 1Gb RAM environment of the raspberry pi can handle up to 15000 devices [2]. Therefore, the system should be able to handle at least 80 devices interactions without any noticeable system latency.

Q2 Availability

- **Q2.1** The system must be available as long as the home server is online.
- **Q2.2** Failure of one subsystem should not lead to failure of the whole system, so there should be an API interface to ensure low coupling among classes.

Q3 Reliability

- **Q3.1** The system must produce accurate cost estimations by utilising parametric estimation methods.
- **Q3.2** The systems AI model should utilise online machine learning to make its predictions and optimisations.

Q4 Scalability

- **Q4.1** South African households on average contain an average of 13.8 appliances [3], so the system should be able to handle at least 14 additional devices that the user chooses to add throughout the lifetime of the system.

Q5 Testability

- **Q5.1** The systems backend component must be testable using Unit and Integration tests through the use of JUnit testing framework.
- **Q5.2** The systems frontend component must be testable using Unit and Integration tests through the use of Karma and Jasmine testing frameworks.
- **Q5.3** The system must implement Continuous Integration through the use of Travis CI.

Q6 Security

- **Q6.1** The system must use role-based access control to gain access to the system's user interface.
- **Q6.2** The system must make use of mosquitto's SSL functionality to secure communication between the broker and clients.[6]

5.2 Constraints

The EISH system has the following constraints:

- The system operates on and within a internal home network
- The system only monitors and controls devices that have been both added onto the systems interface/ database and configured to the the appropriate physical device/ smart switch
- The system will not be involved in any configuration of devices or generators in the smart home
- The system will be deployed to docker containers
- The system will primarily focus on using a solar panel as a form generation

6 Use Case to Subsystem Traceability Matrix

	UI	MGS	MCS	CGS	EPS	ESS	CES	LS	MRS	UCS	NS	OS	DBMSS	DCS
R1.1	X													
R1.2	X													
R2	X													
R3		X												
R4		X												
R5			X											
R6				X										
R7				X										
R8					X									
R9					X									
R10						X								
R11						X								
R12						X								
R13							X							
R14								X						
R15								X						
R16									X					
R17									X					
R18									X					
R19										X				
R20										X				
R21										X				
R22											X			
R23											X			
R24												X		
R25												X		
R26												X		
R27												X		
R28													X	
R29													X	
R30													X	
R31													X	
R32													X	
R33													X	
R34														X
Q1.1	X													
Q1.2													X	
Q2.1	X												X	
Q2.2	X											X	X	
Q3.1							X							
Q3.2												X		

Q4.1	X												X	
Q5.1		X	X	X	X	X	X	X	X	X	X	X	X	X
Q5.2	X													
Q5.3	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Q6.1	X													
Q6.2									X					X

References

[1] The Magpi Magazine https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

[2] Scalagent http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers−v1−1.pdf

[3] Scientific.Net https://www.scientific.net/AMM.672-674.2165

[4] Why Use Angular https://thinkmobiles.com/blog/why-use-angularjs/

[5] Why MQTT https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105

[6] DZone https://dzone.com/articles/mqtt-security-securing-a-mosquitto-server

[7] Docker Container https://www.docker.com/resources/what-container