DEVELOPMENT PLANNING DOCUMENT

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student 3
SoftFlux Engineer

SoftFlux

May 20, 2023

Contents

Revision History

Name	Date	Reason for Changes	Version
SoftFlux	13/05/2023	Define product backlog	1.0
SoftFlux	14/05/2023	Completed community user sprint	1.1
SoftFlux	19/05/2023	Refactoring of user and device structures	1.2
SoftFlux	20/05/2023	Completed encost user functionality	1.3

1 Introduction/Purpose

1.1 Purpose

The purpose of this document is to outline the product backlog and sprint tasks for the development of the Encost Smart Graph Project (ESGP) software. Testing results are outlined with screenshots provided.

1.2 Document Conventions

SDS	The Software Design Specification document
SRS	The Software Requirements Specification document
ESHD	The Encost Smart Homes Dataset
ESGP	The Encost Smart Graph Project
ESH	Encost Smart Home
The test plan	The Functional Software Test Plan document

1.3 Intended Audience and Reading Suggestions

1.3.1 Intended Audience

- User: For users to understand the software design and ensure that the software meets the requirements.
- Developer: For clarifying software design and requirements and to ensure that the software meets those design specifications and requirements.
- Tester: For comparing software functionality to test cases.

1.3.2 Reading Suggestions

It is recommended that the reader be familiar with the Software Requirements Document, Software Design Specification Document, and the Functional Software Test Plan.

1.4 Project Scope

The scope of the project is to design software that enables the visualisation of Encost Smart Home Devices in a graphical interface and the output of specialised device statistics. The project encompasses a study of smart home devices in 100 New Zealand homes during the pierod April 2020 to April 2022. This software has been developed with the intention to be used by those participants in the study identified as 'community' users and Encost employees or approved agents referred to as 'encost' users. The software shall implement the requirements, design specification, and test plan as outlined in their respective documents.

2 Specialized Requirements Specification

- There is nothing in the SDS or test plan that outlines how each of the different components are to work together and how to get from one state to another. Therefore, a stateful architecture based on the state machine design pattern[1] is decided to hold and manage the transitions through different steps of interaction with the software.
- As per section 4.1.3 of the SRS, the user types "community", "encost-unverified", and "encost-verified" must be stored in the system. The SDS and test plan do not explicitly outline where these user types are to be stored. A User Class is decided to store this string attribute.
- The test plan states that device categories are derived from the device id. Instead, the device type is used to derive device category as the device id is unique to a device and will not enable device categories to be derived reliably.
- It is decided that the graph object will go under a class called GraphStructure. This class will perform the loading of the ESHD file, creation of devices, categorising of devices, building the graph data type, and visualising the GraphStream graph. The reason for this is because the test plan and SDS both specify that the graph object should go under the CommunityUser Class but it is not possible to implement such functionality given the architecture of the CommunityUser and EncostUser classes as outlined in the SDS.
- It is not possible to test the loading of files through JUnit. Therefore manual loading tests are carried out. Furthermore, the tests for testing file location unknown, file not found, and file wrong format as outlined in the test plan are inaccurate. It is only possible to test if a file or directory exists or not. Therefore, I have only tested for a file loading scenario where the directory or file cannot be located.
- It is not clear from the SDS and test plan whether the device category is derived from the device type, name, or id. The original requirements document provides a table of categories, types,

[1] https://refactoring.guru/design-patterns/state

and names. The SDS states the device should be an abstract class and that the categories inherit from this class and that the devices inherit from the category, while the test plan indicates that device categories are derived from the device ID. I have implemented exactly what the SDS outlines since deriving device categories from the unique device ID is not correct. The device class is not made abstract which enables the derviceDeviceCategory() method to be implemented.

- The graph data type, according to the SRS and SDS, holds the necessary details that are required in order to build the GraphStream object in the external library. Utilising the Graph-Structure class, it will load in the EHSD, reading line-by-line, creates a device for each line and adds that to an array. The device information is then looped through and for each device a node and edges are added to the GraphStream object. The test plan only has 1 JUnit test. Instead, what I have done is created unit tests for testing that each device type derives the correct device category.

3 Product Backlog

3.1 Software End Users

ESGP currently has two types of users:

- Community - users who opted in and participated during the data gathering exercise for smart device usage for 100 New Zealand homes between April 2020 and April 2022.
- Encost - users who are provided with credentials to login and use extra features of the software. They can be Encost employees or agents that Encost approve access to.

3.2 Features

The major features of the software are:

- Community user console interaction
- Encost employee user console interaction
- Viewing a graph representation of smart devices using the GraphStream functionality
- Calculating and viewing device summary statistics
- Loading and use of custom datasets

3.3 User Stories

3.3.1 Community Users

1. Tony Butters and Heron Ibex are a couple with 3 children under the age of 16 living in Waitakere, Auckland. They both work in demanding jobs in Auckland CBD and are very interested in how smart devices can improve their lifestyle and help save them time with household tasks. They are also curious if their 3 children can use the software in a fun and interactive way as an education tool and to encourage them to use the smart devices more. As a Community User, the household members will need to be able to access the basic graph visualisation and be able to distinguish between different device categories and types.

2. Blue Ferlet has been living in and around Wellington for the past 10 years where she rents. She is very tech savvy and has all the latest smart device gadgets. She fully embraces a digital lifestyle. She is quite interested in exploring the different smart devices available and the different types of common and unique household based smart device configurations. As a Community User, Blue will need to able to access the graph visualisation and be able to view and explore different configurations for households, and identify device connections.

3.3.2 Encost Users

1. Erin Dogbane is a Sales Executive working in the sales team at Encost. In her role she deals with a variety of different customer types such as homeowners, architects, electricians, real estate agents, merchants etc. She is always eager to sell more smart devices and she is enthusiastic about using the software as a tool to help her with the selling process. As an Encost user, Erin will need to be able to login to the software, access graph visualisations and calculated summary statistics, information of which will help her in her role.

2. Anton Pegas is the Commercial Business Manager at Encost. In his role he meets regularly with internal management at Encost and also externally with manufacturers, distributors, and big suppliers who sell the range of the products in retail outlets. He is interested in using the software primarily to extract the summary statistics which he thinks will be useful in decision making for driving the strategic goals of the organisation around product development, market segmentation, and planning. As an Encost user, Anton will need to be able to login to the software, primarily just to access the calculated summary statistics.

3.3.3 Encost Company View

These user stories reflect the view of Encost that do not necessarily have interaction with end users, but are critical for the functioning of the software.

1. Encost would like users to be able to view devices as nodes in the GraphStream UI, connections between nodes, be able to distinguish the send/receive capabilities of each device, and distinguish between the different device categories for each node. In order to fulfil this, the system will need to be able to:
    - Create device objects that hold information about devices
    - Identify and label each device with the category that it belongs to
    - Build a GraphStream structure that visually displays devices as nodes, connections as edges, send/receive capabilities, and other identifying information

4 Sprint Details

Length of time-boxed sprints is 3 hours.

4.1 Sprint #1 14/05/2023 - 14/05/2023

4.1.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.1.2 Sprint Tasks

1. (Development) Console prompt for user type at welcome screen
2. (Development) Receiving input from console at welcome screen and validation
3. (Development) User type is stored as community
4. (Development) Output of feature options for community user
5. (Development) Receiving input from console at feature options screen and validation
6. (Development) Creating device objects
7. (Development) Loading the Encost Smart Homes Dataset
8. (Development) Categorising Smart Home Devices
9. (Development) Building Graph Data Type
10. (Development) Visualising GraphStream graph

4.1.3 Software Design

none noted

4.1.4 Software Testing

Console Screenshots

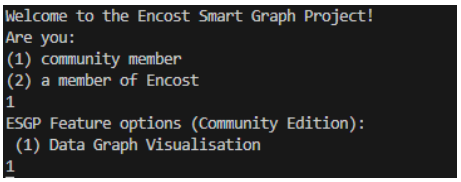

Figure 4.1: Screenshot of successful community user console interaction

```
Welcome to the Encost Smart Graph Project!
Are you:
(1) community member
(2) a member of Encost
x
Error: unrecognised input.
Welcome to the Encost Smart Graph Project!
Are you:
(1) community member
(2) a member of Encost
```

Figure 4.2: Screenshot of error input at Welcome Screen

```
ESGP Feature options (Community Edition):
(1) Data Graph Visualisation
x
Error: unrecognised input.
ESGP Feature options (Community Edition):
(1) Data Graph Visualisation
```

Figure 4.3: Screenshot of error input at Feature Options Screen

JUnit Tests Categorising Users

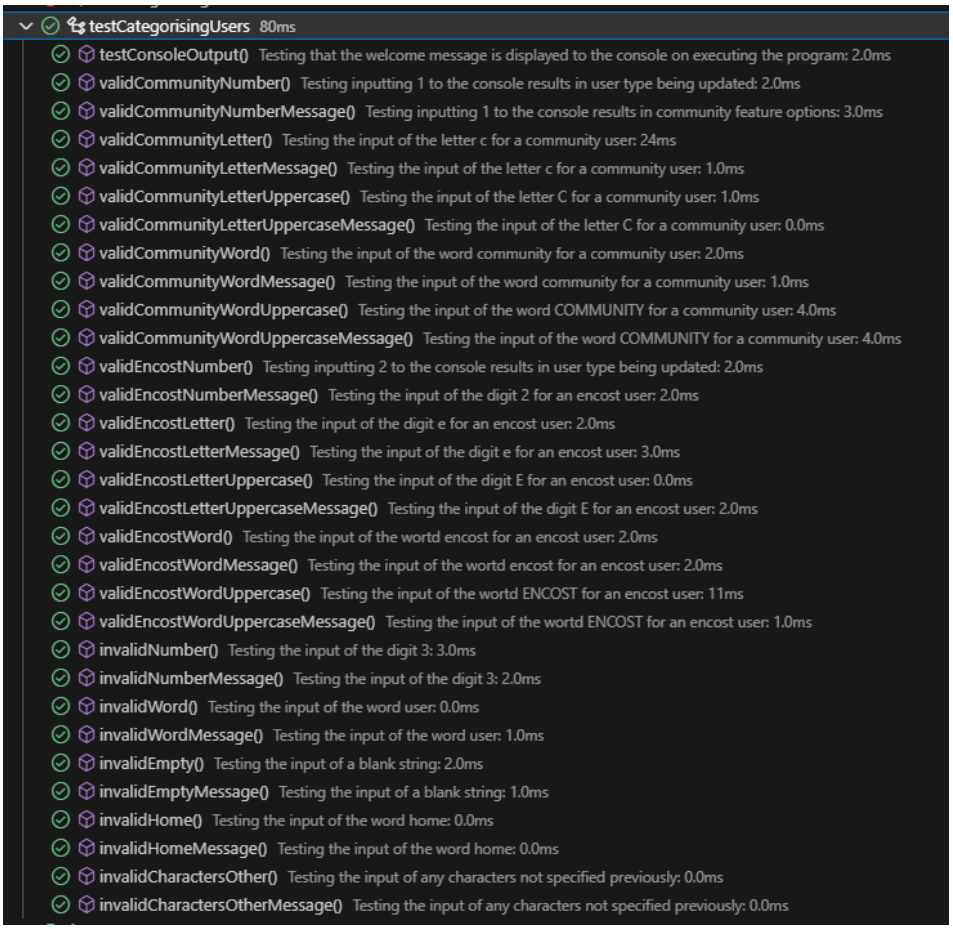

Figure 4.4: Screenshot of passed JUnit tests in Visual Studio Code

JUnit Tests Community Feature Options

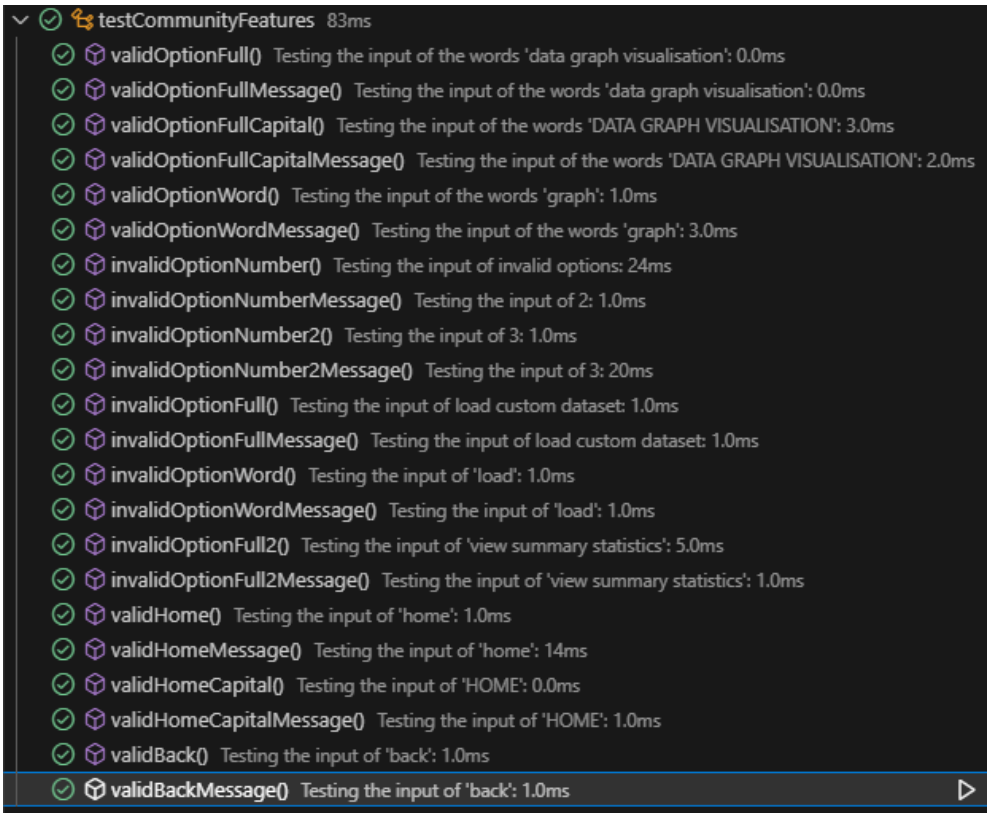

Figure 4.5: Screenshot of passed JUnit tests in Visual Studio Code

4.1.5 Sprint Task Completion

Tasks Completed

- (Development) Console prompt for user type at welcome screen
- (Development) Receiving input from console at welcome screen and validation
- (Development) User type is stored as community
- (Development) Output of feature options for community user
- (Development) Receiving input from console at feature options screen and validation

Tasks Rolled Over

- (Development) Creating device objects
- (Development) Loading the Encost Smart Homes Dataset
- (Development) Categorising Smart Home Devices
- (Development) Building Graph Data Type
- (Development) Visualising GraphStream graph

4.2 Sprint #2 14/05/2023 - 14/05/2023

4.2.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.2.2 Sprint Tasks

- (Development) Creating device objects
- (Development) Loading the Encost Smart Homes Dataset
- (Development) Categorising Smart Home Devices
- (Development) Building Graph Data Type
- (Development) Visualising GraphStream graph

4.2.3 Software Design

Table of device mappings

Device Category	Device types	Device Names
Encost Wifi Routers	Router, Extender	Encost Router 360, Encost Router Plus, Encost Wifi Range Extender 1.0, Encost Wifi Range Extender 2.0
Encost Hubs/-Controllers	Hub/Controller	Encost Smart Hub, Encost Smart Hub 2.0, Encost Smart Hub Mini
Encost Smart Lighting	Light Bulb, Strip Lighting, Other Lighting	Encost Smart Bulb B22 (white), Encost Smart Bulb B22 (multi colour), Encost Smart Bulb E26 (white), Encost Smart Bulb E26 (multi colour), Encost Strip Lighting (white), Encost Strip Lighting (multi colour), Encost Novelty Light (giraffe), Encost Novelty Light (lion), Encost Novelty Light (bear)
Encost Smart Appliances	Kettle, Toaster, Coffee Maker	Encost Smart Jug, Encost Smart Whistling Kettle, Encost Smart Toaster (2 slice), Encost Smart Toaster (4 slice), Encost Smart Coffee Maker, Encost Smart Coffee Maker Mini, Encost Smart Coffee Maker Pro
Encost Smart Whiteware	Washing Machine/Dryer, Refrigerator/Freezer, Dishwasher	Encost Smart Washer, Encost Smart Washer Pro, Encost Smart Dryer, Encost Smart Dryer Pro, Encost Smart Refrigerator, Encost Smart Freezer, Encost Smart Refrigerator/Freezer Combo, Encost Dishwasher, Encost Dishwasher Pro

Figure 4.6: Screenshot of the device mappings table taken from the SRS document

4.2.4 Software Testing

Console screenshots of error messages for file loading

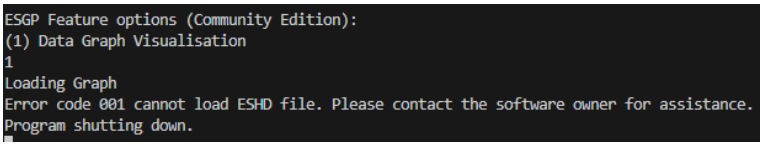

Figure 4.7: Screenshot of error message cannot load ESHD file

JUnit Tests Graph Data Type

Note: checkCategories() manually failed by the tester as the written test was a duplication of testing carried out on the categorising devices test suite.

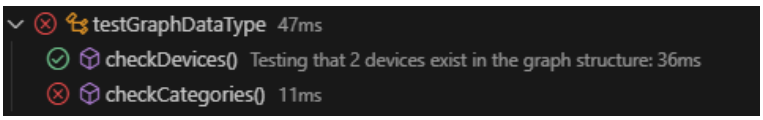

Figure 4.8: Screenshot of passed JUnit tests in Visual Studio Code

Basic implementation of GraphStream UI

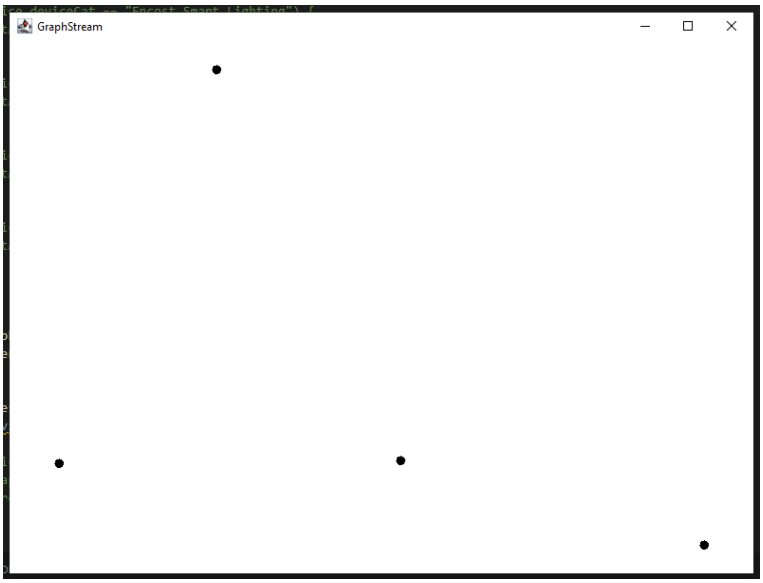

Figure 4.9: Screenshot GraphStream UI

4.2.5 Sprint Task Completion

Tasks Completed

- (Development) Creating device objects
- (Development) Loading the Encost Smart Homes Dataset
- (Development) Categorising Smart Home Devices
- (Development) Building Graph Data Type
- (Development) Visualising GraphStream graph

Tasks Rolled Over

None

4.3 Sprint #3 14/05/2023 - 14/05/2023

4.3.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.3.2 Sprint Tasks

1. (Refactor) Device structure to align more closely with the SDS specification
2. (Development) JUnit tests for new device structure

4.3.3 Software Design

Device Class Diagram

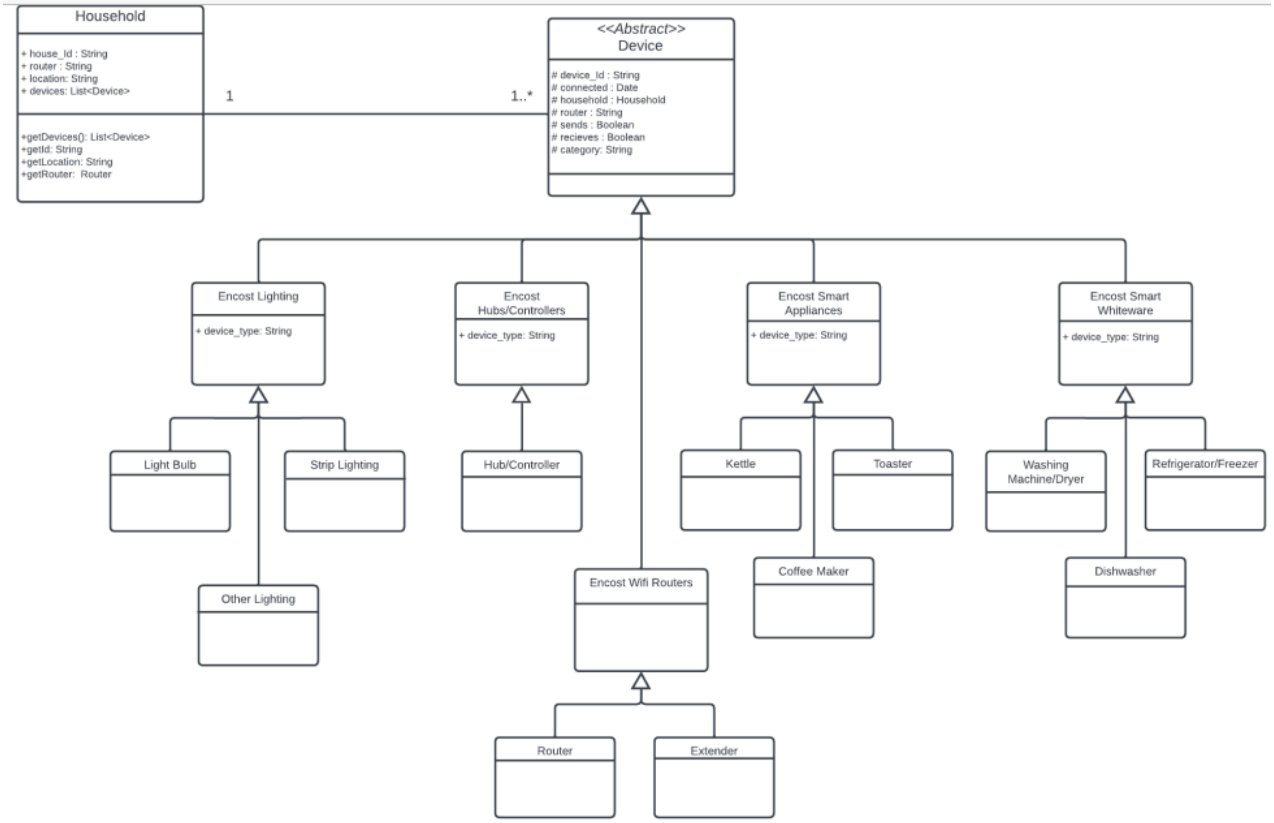

Figure 4.10: Screenshot of the device design structure taken from the SDS document

4.3.4 Sprint Task Completion

Tasks Completed

None

Tasks Rolled Over

- (Development) JUnit tests for new device structure
- (Refactor) Device structure to align more closely with the specification

4.4 Sprint #4 19/05/2023 - 19/05/2023

4.4.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.4.2 Sprint Tasks

1. (Refactor) Device structure to align more closely with the specification
2. (Development) JUnit tests for new device structure

4.4.3 Sprint Task Completion

Tasks Completed

None

Tasks Rolled Over

- (Refactor) Device structure to align more closely with the specification
- (Development) JUnit tests for new device structure

4.5 Sprint #5 20/05/2023 - 20/05/2023

4.5.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.5.2 Sprint Tasks

1. (Refactor) Device structure to align more closely with the specification
2. (Development) JUnit tests for new device structure

4.5.3 Software Testing

JUnit Tests Categorising Devices

Note: checkingData() manually failed by the tester as the written test was not implemented because the test was inaccurate.

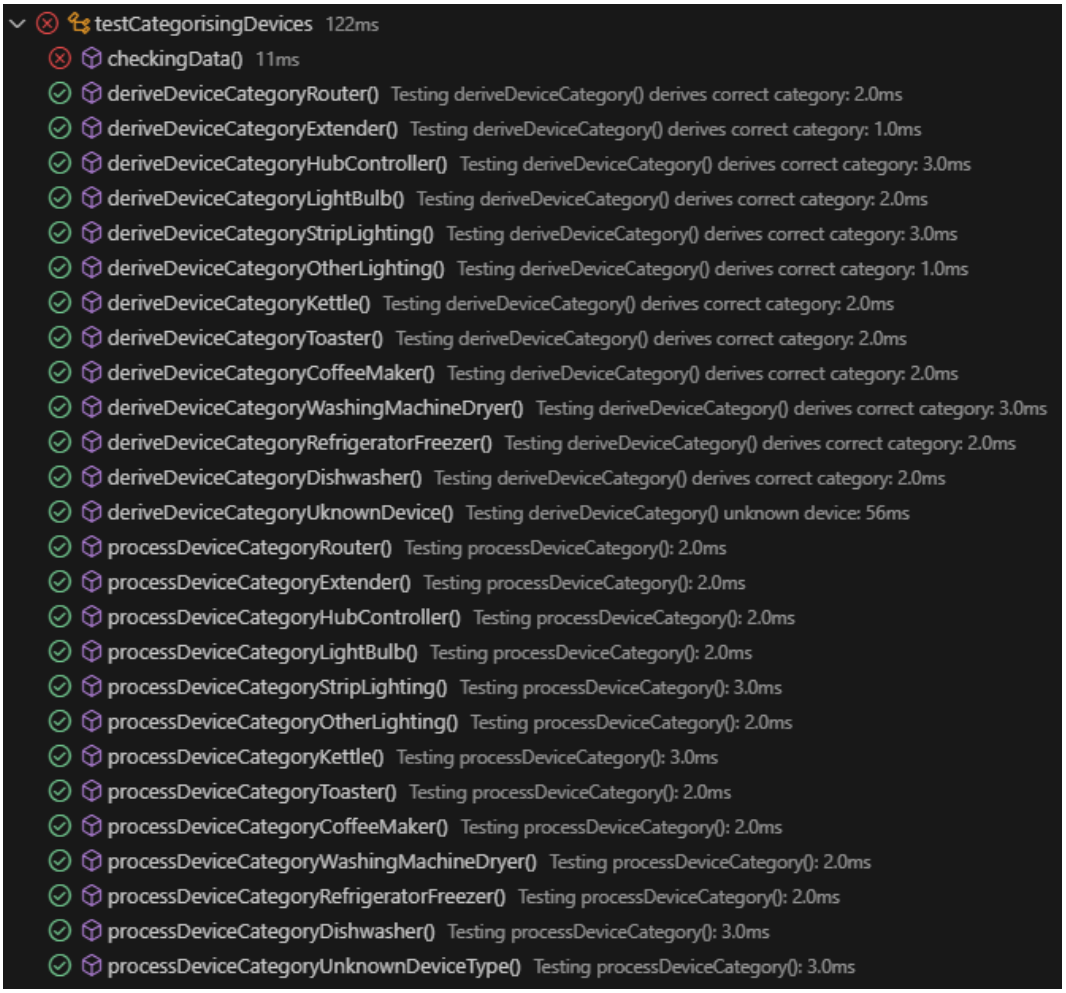

Figure 4.11: Screenshot of passed JUnit tests in Visual Studio Code

JUnit Tests Loading Default Dataset

Note: The tests at the top of the file have been manually failed by the tester as it is not possible to test file loading mechanisms in JUnit without making significant alternations to the software structure. The 4 following tests are manually failed because they have been rewritten by more comprehensive tests.

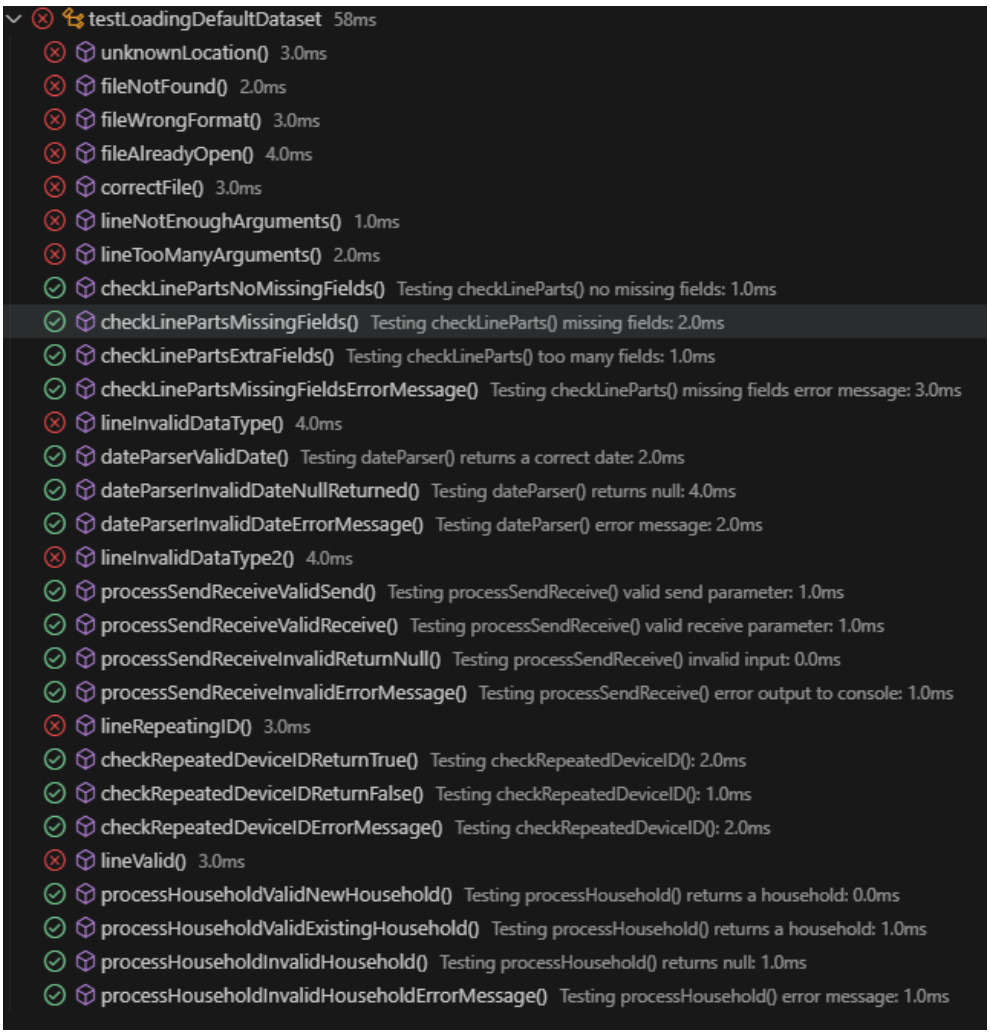

Figure 4.12: Screenshot of passed JUnit tests in Visual Studio Code

4.5.4 Sprint Task Completion

Tasks Completed

- (Development) JUnit tests for new device structure
- (Refactor) Device structure to align more closely with the specification

Tasks Rolled Over

None

4.6 Sprint #6 20/05/2023 - 20/05/2023

4.6.1 Product Backlog Items

Section 3.3.1 for community users, user stories 1 and 2

4.6.2 Sprint Tasks

Refactored the code and JUnit tests for users. This is so that the developed software aligns more closely with what is in the SDS. JUnit tests are developed for the corresponding code.

1. (Refactor) Community and Encost user classes to align more with SDS
2. (Refactor) JUnit tests for community user categorisation and feature options

4.6.3 Software Design

User Class Design

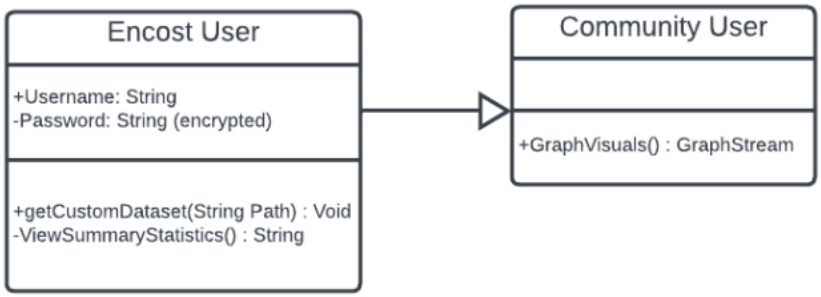

Figure 4.13: Screenshot of class diagram for design of EncostUser and CommunityUser classess from the SDS document

4.6.4 Sprint Task Completion

Tasks Completed

1. (Refactor) Community and Encost user classes to align more with SDS
2. (Refactor) JUnit tests for community user categorisation and feature options

Tasks Rolled Over

None

4.7 Sprint #7 20/05/2023 - 20/05/2023

4.7.1 Product Backlog Items

Section 3.3.3 from ESGP View, user story 1

4.7.2 Sprint Tasks

1. (Development) Distinguish send/receive capabilities for each device in the graph
2. (Development) Distinguish device categories for each device in the graph
3. (Development) Edges are represented in the graph
4. (Development) All nodes are represented in the graph

4.7.3 Software Design

Table of shape mappings to device category

Type	Node Shape
Encost Lighting	Circle
Encost Hubs\Controllers	Cross
Encost Smart Appliances	Square
Encost Smart Whiteware	Diamond
Encost Routers	Rounded Square

Figure 4.14: Screenshot of the device shape mappings to device category taken from the SDS document

Screenshot Graph Mockup

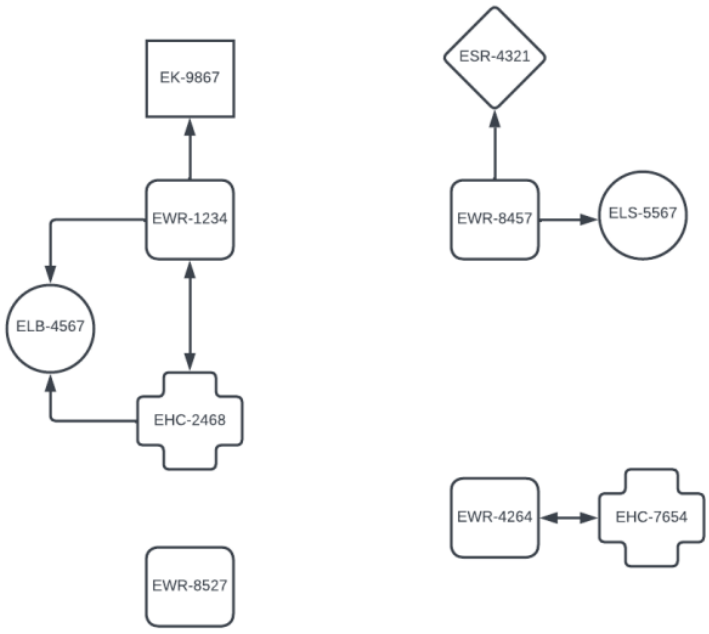

Figure 4.15: Screenshot of the graph UI mockup taken from the SDS document

4.7.4 Software Testing

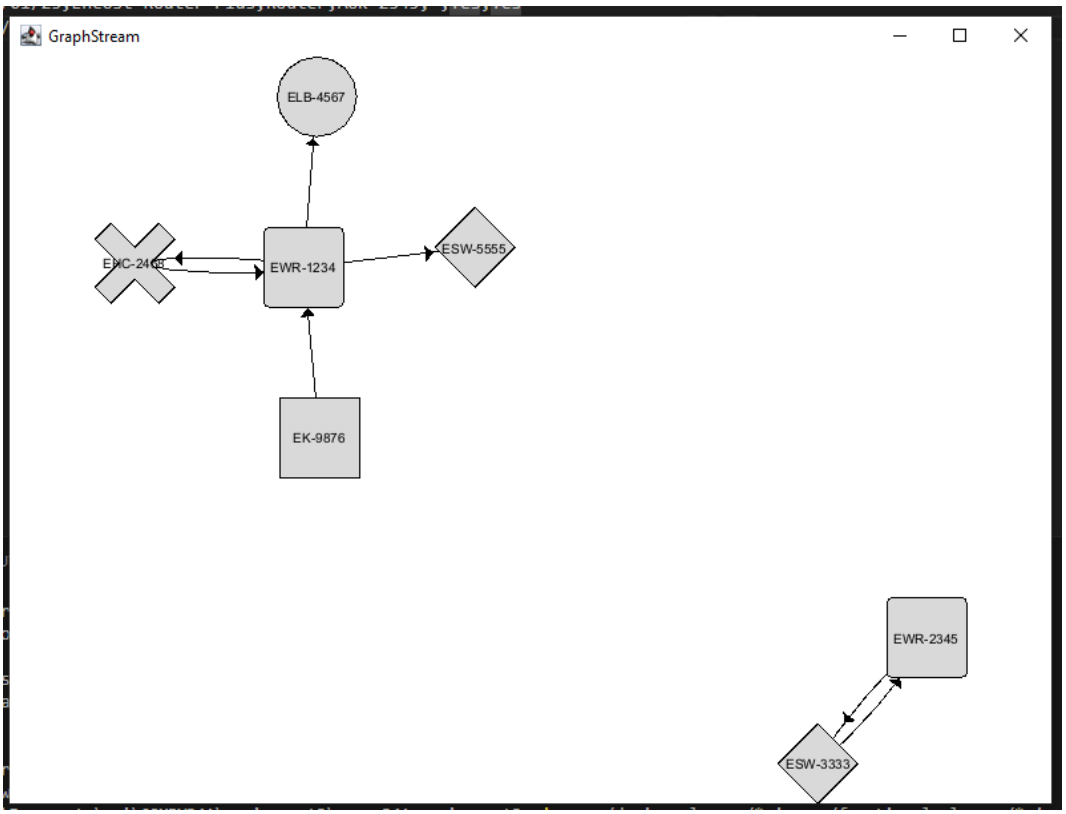

Figure 4.16: Screenshot of graphstream UI

4.7.5 Sprint Task Completion

Tasks Completed

1. (Development) Distinguish send/receive capabilities for each device in the graph
2. (Development) Distinguish device categories for each device in the graph
3. (Development) Edges are represented in the graph
4. (Development) All nodes are represented in the graph

Tasks Rolled Over

None

4.8 Sprint #8 20/05/2023 - 20/05/2023

4.8.1 Product Backlog Items

Section 3.3.2 for encost users, user stories 1 and 2

4.8.2 Sprint Tasks

1. (Development) Receiving input from console at welcome screen and validation
2. (Development) Request login as Encost user
3. (Development) User type is stored as encost-unverified
4. (Development) Handle failed login attempts
5. (Development) Implement MD5 password hashing algorithm
6. (Development) User type is stored as encost-verified
7. (Development) Output of feature options for encost user
8. (Development) Receiving input from console at feature options screen and validation
9. (Development) View Encost summary statistics options
10. (Development) Calculate and view device distribution summary statistics

4.8.3 Software Design

None noted.

4.8.4 Software Testing

Console Screenshots

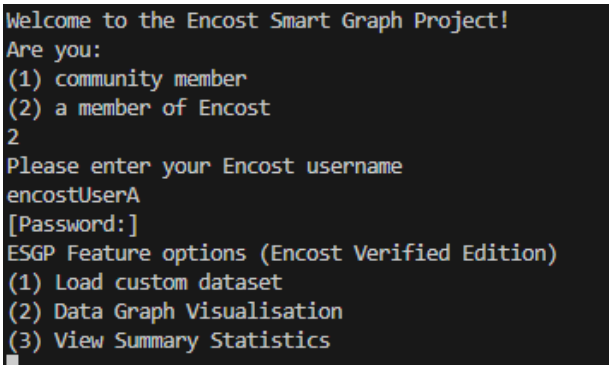

Figure 4.17: Screenshot of successful login

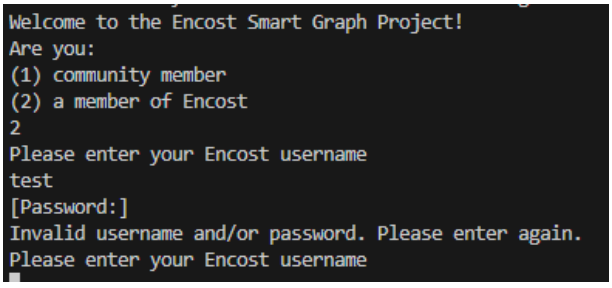

Figure 4.18: Screenshot of unsuccessful login

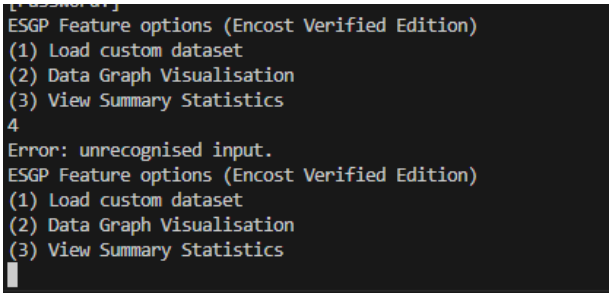

Figure 4.19: Screenshot of invalid input

JUnit Tests Encost Features

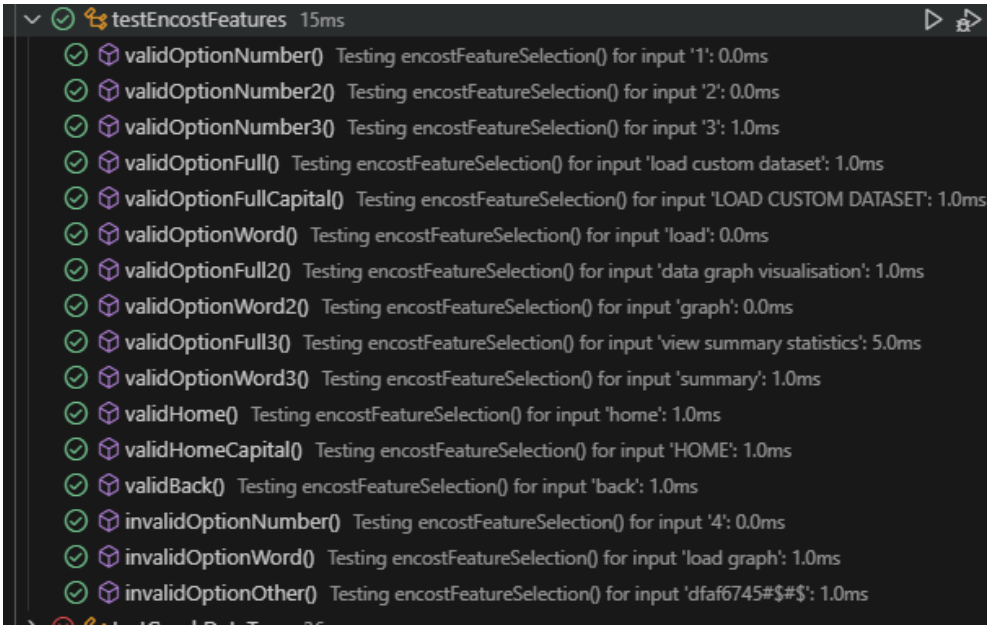

Figure 4.20: Screenshot of passed JUnit tests in Visual Studio Code

JUnit Tests Encost Login

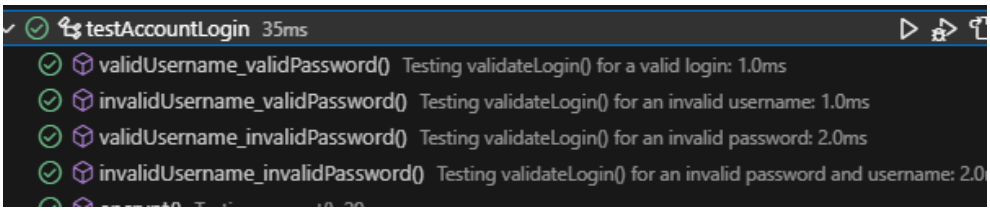

Figure 4.21: Screenshot of passed JUnit tests in Visual Studio Code

4.8.5 Sprint Task Completion

Tasks Completed

1. (Development) Receiving input from console at welcome screen and validation
2. (Development) Request login as Encost user
3. (Development) User type is stored as encost-unverified
4. (Development) Handle failed login attempts
5. (Development) Implement MD5 password hashing algorithm
6. (Development) User type is stored as encost-verified
7. (Development) Output of feature options for encost user
8. (Development) Receiving input from console at feature options screen and validation

Tasks Rolled Over

Note: these tasks were not completed as they are not high priority items.

1. (Development) View Encost summary statistics options
2. (Development) Calculate and view device distribution summary statistics

5 Conclusion

In summary, the ESGP software system has two primary user types, community users and encost users who interact with the software through a console interface. The GraphStream library is utilised to provide a graph representation of Encost Smart Home Devices in a UI window. The Encost user features to view summary statistics and load custom datasets has not been implemented as they are not high priority items. Development of the software has followed a test-drive approach that utilise predominately JUnit automated tests and manual testing where necessary. Password encryption uses the MD5 algorithm as stipulated in the SDS. Every effort has been made to follow the SDS and test plan documents. The specialised requirements section of this document outline any changes or decisions that were made during the development process.