

---

# DEVELOPMENT PLANNING DOCUMENT

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student 2  
SoftFlux Engineer

SoftFlux

May 19, 2023

# Contents

<b>1</b>	<b>Introduction/Purpose</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Document Conventions . . . . .	4
1.3	Intended Audience and Reading Suggestions . . . . .	5
1.4	Project Scope . . . . .	5
<b>2</b>	<b>Specialized Requirements Specification</b>	<b>5</b>
<b>3</b>	<b>Product Backlog</b>	<b>6</b>
3.0.1	Categorising Users . . . . .	6
3.0.2	ESGP Account Login . . . . .	7
3.0.3	ESGP Feature Options . . . . .	7
3.0.4	Loading Encost Smart Home Dataset . . . . .	7
3.0.5	Categorizing Smart Home Devices . . . . .	8
3.0.6	Building a Graph Data Type . . . . .	8
3.0.7	Graph Visualisation . . . . .	9
<b>4</b>	<b>Sprint Details</b>	<b>9</b>
4.1	Sprint #1 <May 10th - May 11th> . . . . .	9
4.1.1	Product Backlog Items . . . . .	9
4.1.2	Sprint Tasks . . . . .	10
4.1.3	Software Design . . . . .	10
4.1.4	Software Testing . . . . .	10
4.1.5	Sprint Task Completion . . . . .	11
4.2	Sprint #2 <May 12th - May 13th> . . . . .	11
4.2.1	Product Backlog Items . . . . .	11
4.2.2	Sprint Tasks . . . . .	11
4.2.3	Software Design . . . . .	11
4.2.4	Software Testing . . . . .	12
4.2.5	Sprint Task Completion . . . . .	12
4.3	Sprint #3 <May 14th - May 15th> . . . . .	13
4.3.1	Product Backlog Items . . . . .	13
4.3.2	Sprint Tasks . . . . .	13
4.3.3	Software Design . . . . .	13
4.3.4	Software Testing . . . . .	13
4.3.5	Sprint Task Completion . . . . .	13
4.4	Sprint #4 <May 16th - May 17th> . . . . .	14
4.4.1	Product Backlog Items . . . . .	14
4.4.2	Sprint Tasks . . . . .	14
4.4.3	Software Design . . . . .	15
4.4.4	Software Testing . . . . .	15

4.4.5	Sprint Task Completion . . . . .	15
4.5	Sprint #5 <May 18th - May 19th> . . . . .	15
4.5.1	Product Backlog Items . . . . .	15
4.5.2	Sprint Tasks . . . . .	15
4.5.3	Sprint Task Completion . . . . .	16
<b>5</b>	<b>Conclusion</b>	<b>16</b>

## List of Figures

4.1	Proof of the categorising users test passing . . . . .	10
4.2	Proof of the account login tests passing . . . . .	12
4.3	Proof of the display features tests passing . . . . .	12
4.4	Proof of the load dataset tests passing . . . . .	14
4.5	Proof of the categorising device tests passing . . . . .	14
4.6	Proof of the build graphs tests passing . . . . .	15
4.7	Proof of a successful GraphStream UI window . . . . .	16

# Revision History

Name	Date	Reason for Changes	Version

## 1 Introduction/Purpose

### 1.1 Purpose

The purpose of this document is to outline and specify the development planning and processes undertaken during development of the Encost Smart Graph Project (ESGP). This document specifies the software construction and use of SCRUM Agile methodologies used in the development period. The software construction will be completed in Java in conjunction with a GitLab repository for progress tracking. The testing will be completed using the Junit tests provided in the Functional Software Test Plan.

This development planning document is based off of the FSTP #4, and the SDS #3.

### 1.2 Document Conventions

This document uses the following conventions:

- ESGP: Encost Smart Graph Project
- SRS: Software Requirements Specification
- SDS: Software Requirements Specification
- FSTP: Functional Software Test Plan

- CLI: Command Line Interface
- REQ: Requirement
- SHA-256: Secure Hashing Algorithm 256
- ESHD: Encost Smart Home Dataset

### 1.3 Intended Audience and Reading Suggestions

This document is to be used as a tool to assist with the construction of the ESGP and document the process which was undertaken during development. Thus this document will be of use to developers, testers, and project managers for the following reasons.

- Developer: To be used to track progress through the development of the ESGP using scrum methodologies. Can be used to identify and self assign tasks from the product backlog.
- Tester: Can be used to identify what may be ready for testing within the project and aid in identifying other testing which is required.
- Project Manager: As a tool for tracking the construction process and assigning tasks to team members. Will allow easy understanding of progress and identifying any project scheduling issues.

### 1.4 Project Scope

Encost is a new and emerging Smart Home development company. They manufacture a series of Smart Home and IoT solutions, including Wifi Routers, Smart Hubs and Controllers, Smart Light Bulbs, Smart Appliances, and Smart Whiteware. Encost is interested in investigating how their smart devices are being used and connected within households across New Zealand.

The ESGP is a software system with aims to enable the Encost smart devices to be visualised in a graph data structure from the command line. With both accessibility to Encost verified users and community users but with restricted access to the software capability. There will be no hardware integration required for the ESGP nor for the testing purposes of the software.

SoftFlux is responsible to the complete software construction of the ESGP and the insurance that the software meets the requirements by passing the tests outlined in the FSTP, developing the software to the standards outlined in the SDS using the Agile Scrum methodology outlined in this document.

## 2 Specialized Requirements Specification

During the implementation process of the ESGP there were three main changes which occurred:

- Due to an issue with the ESHD given for development the Device Types. When these issues were found the clients were notified by a development at SoftFlux. Due to the small development window the databases was modified to fit the requirement specifications rather than the ESGP developed to detect these issues. This is a possible further development step which will need to be addressed later on.
- Part way through the development of the ESGP the Clients changed the scope of the minimum viable product. The development of the device summary statistics was halted as the priority was changed from high to medium. Therefore, this part of the ESGP was not developed during this development period.
- There were small issues with the testing JUnit tests. These are possibly due to version issues and that the tests were built prior to any development so no compiling errors were able to be caught. When these errors were found they were noted within the code and the sprint write ups.

## 3 Product Backlog

### 3.0.1 Categorising Users

**Description:**

An enum within the ConsoleApp which indicates whether the user is a Community User or an Encost User.

**Requirements Covered:**

SRS section 4.1.3. REQ 1, 2, 3.

**Required Tests:**

FSTP section 3.1. Black-box unit test to validation of ApplicationState.

**User Stories:**

- Ada Lovelace is a community user that would like to try using the ESGP. She has never used the application before but was emailed information of how to access the program to use it as she was part of the 100 users who had information of their smart devices gathered between April 2020 and April 2022.
- Grace Hopper is a employee at Encost and has been asked to use the ESGP to ensure that it meets the requirements which the company gave to SoftFlux before development began. She has a verified Encost account she has been told to use.

### 3.0.2 ESGP Account Login

**Description:**

Allows Encost employees to login to access additional features. Successful verification will take the user to the verified features page. Verified user passwords are hashed with SHA-256 hashing.

**Requirements Covered:**

SRS section 4.2.3. REQ 1, 2, 3, 4, 5.

**Required Tests:**

FSTP section 3.2. Boundary values black-box test to test verifying the credentials of the users. Equivalence partition to ensure the correct response occurs after a successful or unsuccessful login attempt is made.

**User Stories:**

- Grace Hopper has been given a verified Encost login to use while checking out the ESGP. She has never logged into the ESGP yet and needs to access the system to ensure that all the requirements outlined by her team at Encost were met by the developers at SoftFlux.

### 3.0.3 ESGP Feature Options

**Description:**

Encost-Verified Users should be able to select whether they want to (a) load a custom dataset; (b) visualise a graph representation of the data; or (c) view summary statistics. A Community user shall only be able to (a) visualise a graph representation of the data.

**Requirements Covered:**

SRS section 4.3.3. REQ 1, 2.

**Required Tests:**

FSTP section 3.3. Black-box equivalence class test to verify that the correct options are displayed for each user.

**User Stories:**

- Ada Lovelace is a community user she would to be able to see her device through the Encost system. But she needs to leave for work soon so also needs to be able to quickly exit the ESGP to shut down her computer.
- Grace Hopper has successfully logged in to the ESGP. She now needs to check that she can access all the required features for her to complete her work for the day. It is 4pm and Grace finishes work soon so needs to be able to quickly check her access to the programs features.

### 3.0.4 Loading Encost Smart Home Dataset

**Description:**

The system should be able to read and process the Encost Smart Homes Dataset. This must be created using a FileParser. The dataset location must be stored in ENCOST\_DATASET\_FILEPATH.

**Requirements Covered:**

SRS section 4.4.3. REQ 1, 2, 3.

**Required Tests:**

FSTP section 3.4. Count verification Block-box boundary test. Ensure that all device data is correctly read in and the FileParser can handle new lines. Ensure the parser returns the correct objects and that the file parser can handle unexpected inputs.

**User Stories:**

- Grace Hopper has access to the ESHD as she works at Encost. She wants to be able to open the dataset from within the ESGP to ensure all of the data is loaded correctly. She must check that no data has been lost or edited during the development process. She has to also ensure all of the community users information has remained private and cannot be seen from within the program due to the privacy clause Encost has with their users.

### 3.0.5 Categorizing Smart Home Devices

**Description:**

The system should be able to categorize each Encost Smart Device into one of five categories. These categories will be used for the graph visualisation and summary statistics.

**Requirements Covered:**

SRS section 4.5.3. REQ 1, 2.

**Required Tests:**

FSTP section 3.5. Black-box equivalence partitioning and value verification. Ensures the device values are assigned to the appropriate fields. Also tests to ensure that the correct category is being returned for the device categoriser class.

**User Stories:**

- Encost would like users to be able to easily recognise the different device types as outlined in all of the provided SRS, SDS, and FSTP. This will the users to easily identify the devices within the graph visualisation.

### 3.0.6 Building a Graph Data Type

**Description:**

The system should be able to categorize each Encost Smart Device into one of five categories. These categories will be used for the graph visualisation and summary statistics.

**Requirements Covered:**

SRS section 4.6.3. REQ 1, 2, 3.

**Required Tests:**



FSTP section 3.6. Black-box equivalence partitioning, value verification and edge case testing. Ensure the deviceGraph assigns the internal device array correctly. That the neighbours of the device graph are calculated correctly. Ensures that invalid device ID's are handled correctly.

**User Stories:**

- Encost would like to ensure that the graphs created are correctly assigning the nodes and edges. They would also like to know that the internal device array information is correctly transferred into the graph.

### **3.0.7 Graph Visualisation**

**Description:**

The system should allow the user to view a visualisation of the graph data structure.

**Requirements Covered:**

SRS section 4.7.3. REQ 1, 2, 3, 4, 5.

**Required Tests:**

FSTP section 3.7. Testing the user interface is out of the scope for the FSTP.

**User Stories:**

- Ada Lovelace is a community user, the only feature available to her is the graph visualisation. She can see the ESHD graph using this feature but due to being a community user she has no knowledge of how the data should be represented. The graph visualisation must be clear and easy to understand to ensure that Ada Lovelace can make the most of the one feature available to her.

## **4 Sprint Details**

The sprints undertaken by the developers at SoftFlux while working on the ESGP will be of 2 day duration until the completion of the project.

### **4.1 Sprint #1 <May 10th - May 11th>**

#### **4.1.1 Product Backlog Items**

- Ticket section 3.0.1 - Categorizing Users
- Ticket section 3.0.2 - ESGP Account Login

## 4.1.2 Sprint Tasks

### Categorizing Users

- Set up initial ESGP framework. Includes linking to GitLab repository, empty classes and welcome message to console within the ConsoleApp.
- Set up ApplicationState class with get and set methods.
- Develop section 4.1.1 of the SDS, getting user version.
- Verify development with FSTP section 3.1.1 to validate the application state.

### ESGP Account Login

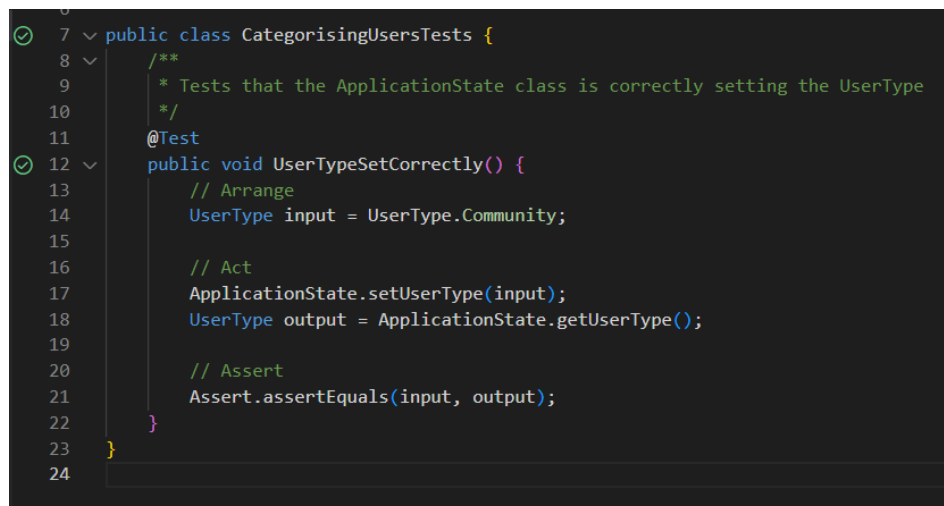
- Develop user verifier from section 4.3 of the SDS.
- Develop section 4.1.2 of the SDS, verifyUser function.
- Verify the development with test from section 3.2 of the FSTP.

## 4.1.3 Software Design

No software design aspects were included in this sprint.

## 4.1.4 Software Testing

Modifications to the imports of the CategorisingUsersTest.java were made in order to allow the test to run. No changes to the actual test code were made.



```
7 public class CategorisingUsersTests {
8     /**
9      * Tests that the ApplicationState class is correctly setting the UserType
10     */
11     @Test
12     public void UserTypeSetCorrectly() {
13         // Arrange
14         UserType input = UserType.Community;
15
16         // Act
17         ApplicationState.setUserType(input);
18         UserType output = ApplicationState.getUserType();
19
20         // Assert
21         Assert.assertEquals(input, output);
22     }
23 }
24
```

Figure 4.1: Proof of the categorising users test passing

Testing for the ESGP account login were not all completed. Of the given tests only 4 were successfully compiled to run.

### 4.1.5 Sprint Task Completion

Categorising users was successfully implemented and tested as shown in figure 4.1. This ticket has been closed as of the end of sprint 1.

ESGP account login was implemented into the ESGP. However, due to JUnit compilation errors the development was not fully verified. As of the closing of Sprint 1 four of the eight tests successfully compiled, ran and passed. These include:

- IncorrectCredentialsHaveLoginFailure()
- EmptyStringsReturnFalse()
- LoginSuccessPromptWhenSuccess()
- LoginFailurePromptWhenFailure()

## 4.2 Sprint #2 <May 12th - May 13th>

### 4.2.1 Product Backlog Items

Incomplete tickets from Sprint 1 section 4.1:

- Ticket section 3.0.2 - ESGP Account Login

Ensure the remaining tests from 'AccountLoginTest.java' successfully compile and run. Make modifications to account login code if required.

New tickets to be added into Sprint 2:

- Ticket section 3.0.3 - ESGP Feature Options

### 4.2.2 Sprint Tasks

#### ESGP Account Login

- Verify the development with test from section 3.2 of the FSTP.

#### ESGP Feature Options

- Create the empty classes required for when a user selects a feature. Not implementing the class features.
- Develop section 4.1.4 of the SDS, selecting a feature.
- Verify development with FSTP section 3.3 to validate the consoleApp class modifications.

### 4.2.3 Software Design

All design within this sprint following the UI design outlined in section 2.5 of the FSTP.

#### 4.2.4 Software Testing

Some slight modifications were required to allow the tests to run. The junit.jupiter imports were unable to run. However, these were just changed to a standard junit equivalence. The requirements for some exception handling tests meant that the `verifyUser` method and the `Verifier` constructor were required to throw errors in order for those tests to pass. Because of this they were required to be within try-catch structures in the testing methods. As shown in figure 4.2 all tests successfully passed.

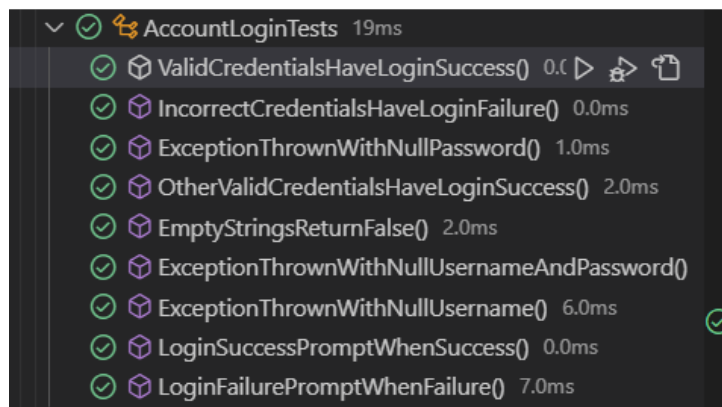


Figure 4.2: Proof of the account login tests passing

As with all other junit test the feature options tests required modifications to the imports in order to run correctly. A mistake with also found in the community features test were 'options[0]' was used for all options asserts meaning that the tests would not pass. This was change so that the correct index was used for each assert.

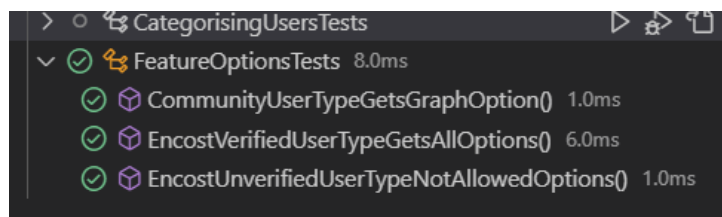


Figure 4.3: Proof of the display features tests passing

#### 4.2.5 Sprint Task Completion

Each goal within sprint 2 was completed in time with all JUnit test passing. All code was uploaded to the GitLab repository within the sprint time frame.

## **4.3 Sprint #3 <May 14th - May 15th>**

### **4.3.1 Product Backlog Items**

New tickets to be added into Sprint 3:

- Ticket section 3.0.4 - Loading Encost Smart Home Dataset
- Ticket section 3.0.5 - Categorising Smart Home Devices

### **4.3.2 Sprint Tasks**

#### **Loading ESHD**

- Implement the Device class attributes.
- Implement the FileParser class.
- Verify the class implementations using the junit test for the file parsing.

#### **Categorising Smart Home Devices**

- Implement the Device class methods.
- Implement the DeviceCategory enumerators.
- Fully implement the deviceCategoriser class.
- Verify the device class and methods using the provided tests.

### **4.3.3 Software Design**

Software requirements update:

During this sprint there was a requirement update released by the clients Encost. Calculating the device distribution is no longer considered to be a high priority requirement. This change has been reflecting in section 4.9 of the SRS (version 1.2).

### **4.3.4 Software Testing**

Proof of the dataset loading test and categorising devices tests can be seen in figure 4.4 and figure 4.5.

### **4.3.5 Sprint Task Completion**

Every task outlined for both tickets Loading ESHD and Categorising Smart Home Devices were completed within the sprint time frame. All tests passed for both tickets before sprint completion. No items from sprint 3 remain within the sprint backlog.

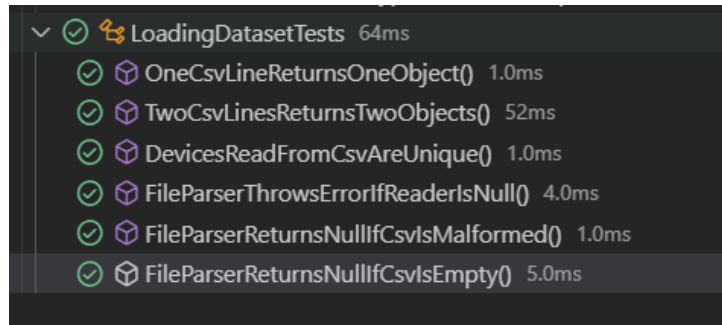


Figure 4.4: Proof of the load dataset tests passing

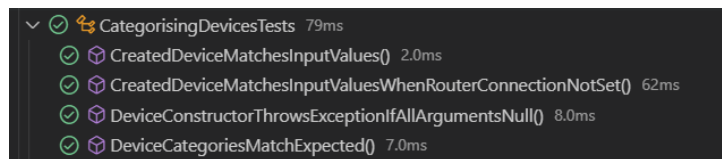


Figure 4.5: Proof of the categorising device tests passing

## 4.4 Sprint #4 <May 16th - May 17th>

### 4.4.1 Product Backlog Items

New tickets to be added into Sprint 4:

- Ticket section 3.0.6 - Building a Graph Data Type
- Ticket section 3.0.7 - Graph Visualisation

### 4.4.2 Sprint Tasks

#### Building a Graph Data Type

- Implement the deviceGraph class.
- Ensure edges and node of the graph are correctly created.
- Verify the class implementations using the junit test the deviceGraph object pass.

#### Graph Visualisation

- Implement the GraphVisualiser class.
- Ensure ensure the deviceGraph is correctly converted to a Graph Stream graph.
- Manually verify that the graph stream graph matches what is outlined within the SDS. No tests are outlined in the FSTP due to it being out of scope.

### 4.4.3 Software Design

Slight modifications to the ESHD were required in order for the graph stream objects to successfully compile and run. Some of the device types were considered as typos. Due to the client considerations these were modified rather than error caught for this stage in development. These modifications were as follows:

- “Light bulb” - “LightBulb”
- “Hub/ Controller” - “HubController”

### 4.4.4 Software Testing

Proof of the build graph data type can be seen in figure 4.6.

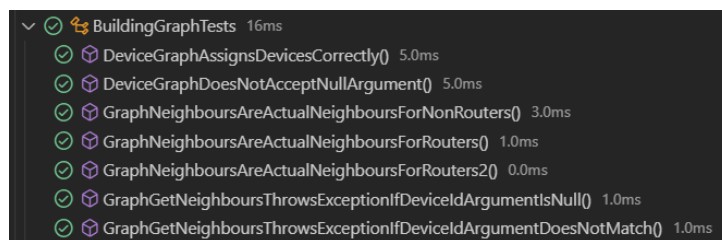


Figure 4.6: Proof of the build graphs tests passing

### 4.4.5 Sprint Task Completion

All tasks assigned during sprint 4 were completed on time and to the required level. The tests for the building graph datatype successfully pass and as shown in figure 4.7 the graph virtualisation was completed successfully. No tasks need to be rolled over into sprint 5.

## 4.5 Sprint #5 <May 18th - May 19th>

### 4.5.1 Product Backlog Items

No product backlog items are to be added to this sprint as no product backlog items remain.

### 4.5.2 Sprint Tasks

- Ensure code quality levels are within SoftFlux standards.
- Clear git repository of unnecessary files for production at Encost.
- Produce and attach java-docs for the ESGP.

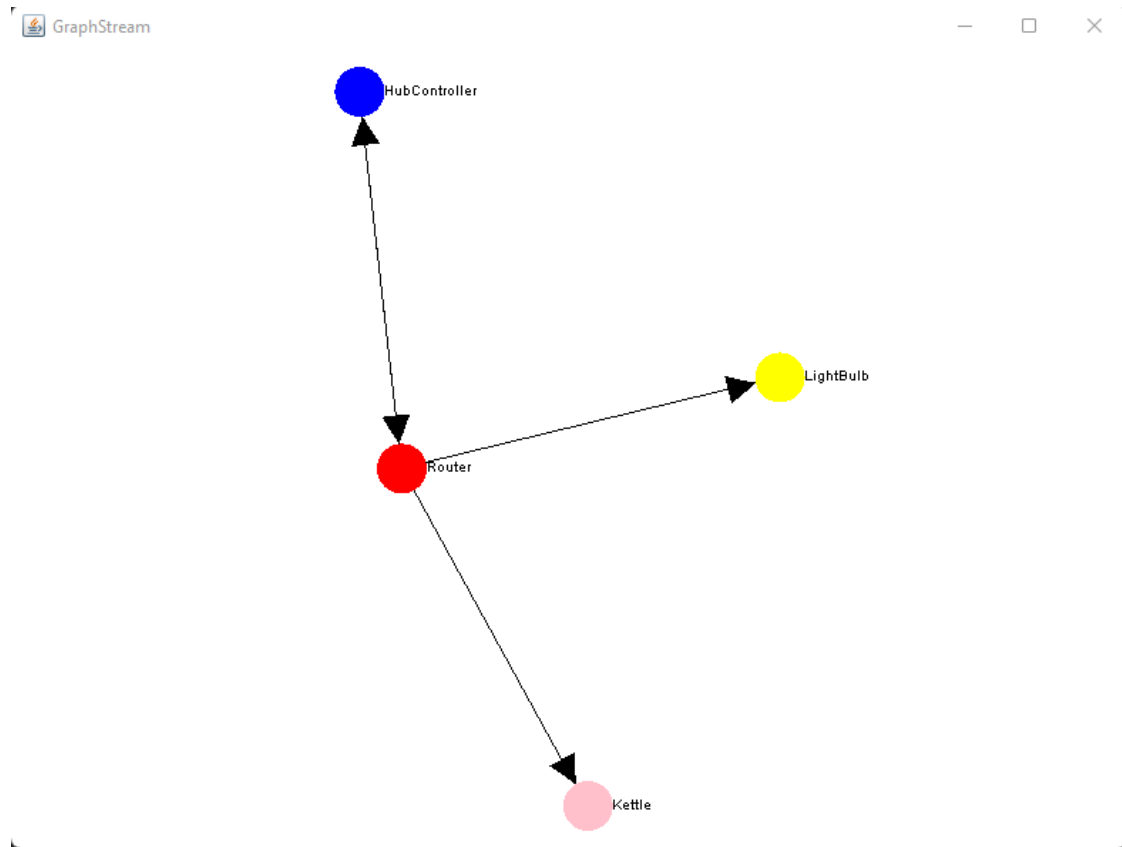


Figure 4.7: Proof of a successful GraphStream UI window

- Review documentation for the ESGP to ensure all information is up to date and ready for maintenance.

### 4.5.3 Sprint Task Completion

The git repository for the ESGP was tidied of unnecessary files (\*.class) files. Javadocs were completed for all the methods throughout the program. A new method was added in the ConsoleApp.java file to communicate to the user if a particular feature is not implemented due to the changes in the project scope.



## 5 Conclusion

The implementation of the minimum viable product of the ESGP was implemented across 5 two day sprints. During these sprints there were 5 high priority tickets which were completed. All tests outlined by the FSTP #4 were successfully run against the ESGP and passed. All the passed test were uploaded to this document as figures to show the progress of the implementation, a summary of this can be seen in the list of figures. The project was continuously uploaded to a GitLab repository across 19 commits. All methods and classes have appropriate JavaDoc commenting and comply with the SoftFlux coding standards. Due to the current scope of the ESGP some of the features found on the select features page for the Encost verified users are not implemented, a design decision was made by the developer to communicate this to the user and return to the select features page.