DEVELOPMENT PLANNING DOCUMENT

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student 1
SoftFlux Engineer

SoftFlux

May 19, 2023

Contents

List of Figures

List of Tables

Revision History

Name	Date	Reason for Changes	Version
SoftFlux	09/05/23	Split backlog into sprints	1.0

1 Introduction/Purpose

1.1 Purpose

The purpose of this document is to provide the development plan for the Encost Smart Graph Project. This document covers the Product Backlog and the dates and progress of each Sprints.

1.2 Document Conventions

This document uses the following conventions:
ESGP: Encost Smart Graph Project
SRS: Software Requirements Specification
SDS: Software Design Specification
FSTP: Functional Software Test Plan
ESH: Encost Smart Homes

1.3 Intended Audience and Reading Suggestions

- Business owners - to see that the software is being created and getting value for their money
- Project manager - to see the progress in the software development and ensure the project is on track

- Developer - use the document to implement the software by reading user stories and sprint dates
- Tester - can see the progress made (as well as upcoming tasks to test for) and to see the tests it has passed

1.4 Project Scope

The FSTP chosen was #2, which was based off SDS #3.

The requirements to be implemented were all of those labelled *High Priority* in Version 1.1 of the SRS.

2 Specialized Requirements Specification

When a product does not relate to the user, the user story should be in the point of view of the Encost company.

Software design decisions have been discussed within the 'Software Design' section for each sprint.

In the SDS, it was specified that to exit the program, the user can click the exit button (X) on the window of the console. There is no requirement for any other exit method (for example through feature selection) so this was not implemented.

3 Product Backlog

3.1 Product #1: Categorising Users

User story: I want to be able to choose whether I am a community or encost-verified user.

The purpose of this is for the user to select whether they are a community or encost user. This involves displaying the user selection prompt to the user, reading and validating the user input, storing the user type, and checking against the JUnit tests.

3.2 Product #2: ESGP Account Login

User story: As an encost-verified user, I want to ensure that I can gain access to the secure software as simple as possible.

The purpose of this is for the user to enter their username and password to login. This involves displaying the prompts, reading the user input, validating the username/password pair, and checking against the JUnit tests. It also involves reading in username and passwords from a file as well as encrypting existing passwords and user inputted passwords.

3.3 Product #3: ESGP Feature Options

User story: I want to be able to view only the features that I can use as simple as possible.

The purpose of this is to display the appropriate features for each user type and to display the selected feature. This involves displays each user types prompts, reading and validating each user types inputs, and checking by going through each option to ensure correct input validation.

3.4 Product #4: Loading the ESH Dataset

User story: I want the software to preload data.

The purpose of this is to open and read the ESH Dataset and return an array of Device objects in the file. This involves creating two classes with methods to read in lines of the file to create a Device object, and checking against JUnit tests.

3.5 Product #5: Categorising Devices

User story: Encost wants the devices to recognise the different device categories for graph visualisation and statistics purposes.

The purpose of this is to assign the correct Device category to a device when the object is created. This involves creating an Enum, relating the type to the category, and checking against JUnit tests.

3.6 Product #6: Building Graph Data Type

User story: Encost wants the devices to be stored in a graph data structure to show the connections between devices.

The purpose of this is to create a graph data structure which holds all the Device objects and shows the connections between them. This involves creating a class with methods to add nodes (devices) and add edges (connections), and checking against JUnit tests.

3.7 Product #7: Graph Visualisation

User story: I want to be able to see all the devices in a clear way where I am able to see the connections between devices, the category of each device, and whether a device can send and/or receive commands.

The purpose of this is to implement the GraphStream library to visualise data using a UI graph. This involves adding devices as nodes to the GraphStream graph, adding connections between devices as edges, formatting the graph using CSS, as passing visual tests.

3.8 Product #8: Device Distribution

User story: I want to be able to see how many devices there are of each type and category in a clear and well formatted way.

As of 15/05/2023, this Product is no longer required in this software implementation.

4 Sprint Details

4.1 Sprint #1 <10/05/2023 to 11/05/2023>

4.1.1 Product Backlog Items

- Section 3.1: Categorising Users
- Section 3.2 ESGP Account Login
- Section 3.3 ESGP Feature Options

These were chosen as the products for the first sprint as they are the first steps leading to the other requirements. For example, feature options cannot be completed before categorising users is completed. This is due to the feature options being dependant on the user type which is determined through the categorising users requirement.

These were grouped together as they only deal with the login information, as opposed to dealing with the dataset and storing/manipulating data.

4.1.2 Sprint Tasks

- Section 3.1
  - Setup ConsoleApp class
  - Display prompt

  - Validate user input
  - Store user type
  - Check against tests
- Section 3.2
  - Setup UserVerifier class
  - Encrypt passwords
  - Store usernames and encrypted passwords
  - Display prompt
  - Compare/validate user input
  - Check against tests
- Section 3.3
  - Display prompt for community
  - Validate community input
  - Display prompt for encost
  - Validate encost input
  - Check against tests

4.1.3 Software Design

Made a change regarding checkUserType(char input) function as mentioned in Section 3.1.1 in the FSTP. This function instead takes a string input and returns a boolean value rather than a string. The boolean returned is true if the input is valid and false if the input is not valid. This method will also change the state of the class scope userVerified variable.

Following the description in the FSTP rather than the SDS, to verify username and password, only one method is created which has both the username and password as a parameter and returns a boolean of whether they are a valid pair. This is implemented rather than having one method to verify the username and one method to verify the password.

Following the SDS, the passwords are being encrypted through hashing. MD5 was chosen for the hashing method.

The usernames along with the encrypted passwords are stored in a text file. This can then be read by the UserVerifier class at the time of verification to prevent storing the information internally.

In the SDS, the Class Diagram shows that the program should run through the main() method within the ConsoleApp class. This was changed to run through the main() method in a new class called ESGP. This was implemented as it prevents all the methods and attributes in the ConsoleApp form having to be static. The main() method in the ESGP class only creates a ConsoleApp object which then runs the software.

4.1.4 Software Testing

Figure 4.1 and Figure 4.2 show that Section 3.1 and Section 3.2 have passed all the appropriate tests included in the Functional Test Suite.

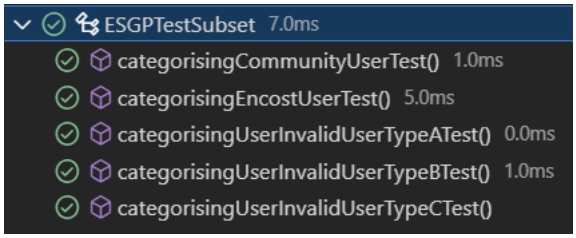

Figure 4.1: Passes all tests for Section 3.1

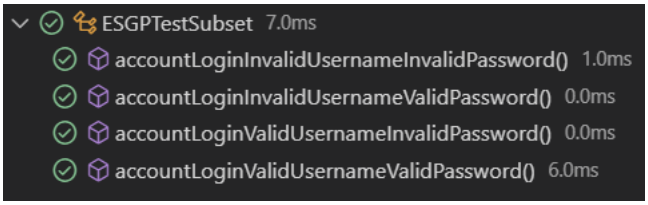

Figure 4.2: Passes all tests for Section 3.2

No JUnit tests were providing for checking Section 3.3. The following visual requirements were met during testing:

- Community members had 1 option displayed
- Encost members had 3 options displayed
- Only "1" was valid input for community user
- "1", "2", and "3" were valid inputs for encost user
- All other inputs were invalid

4.1.5 Sprint Task Completion

Table 4.1 shows the status of each product at the end of the sprint. All products were completed within the sprint, and none were carried onto the next sprint.

Product	Completion Date
Section 3.1	10/05/2023
Section 3.2 without encryption	10/05/2023
Section 3.3	10/05/2023
Section 3.2 with encryption	11/05/2023

Table 4.1: Completion dates of each product within sprint 1

4.2 Sprint #2 <11/05/2023 to 13/05/2023>

4.2.1 Product Backlog Items

- Section 3.4: Loading the ESH Dataset
- Section 3.5: Categorising Devices
- Section 3.6: Building Graph Data Type

These products were chosen as they all deal with the internal storage and manipulation of data. All of these products had nothing additional displayed to the user, all progress was in the backend. All three of these products are required to display the results of the data through graph visualisation and summary statistics (in the next sprint). The graph cannot be viewed if the data is not in the correct format and definitely cannot be viewed if the data has not been loaded.

4.2.2 Sprint Tasks

- Section 3.4
  - Create Device class
  - Create FileParser class
  - Determine file path to pass in
  - Check against tests
- Section 3.5
  - Create DeviceType Enum
  - Create a link between type and category
  - Check against tests
- Section 3.6
  - Create DeviceGraph class
  - Check against tests

4.2.3 Software Design

The DeviceType Enumeration in the SDS was renamed to DeviceCategory. A DeviceCategory attribute, called deviceCategory, was also added to the Device class. This is to represent both the type and the category of the device separately. Within the Device class, a dictionary was created to show the relationship between the String deviceType and DeviceCategory deviceCategory.

It was not made clear where the DeviceGraph is made, I have decided to do this before the feature selection loop. This then means that it is loaded and can be used for the graph visualisation and the summary statistics without needing to be loaded then or reloaded everytime.

4.2.4 Software Testing

Figure 4.3, Figure 4.4, and Figure 4.5 shows that all three products have passed the relevant tests included in the Functional Test Suite.

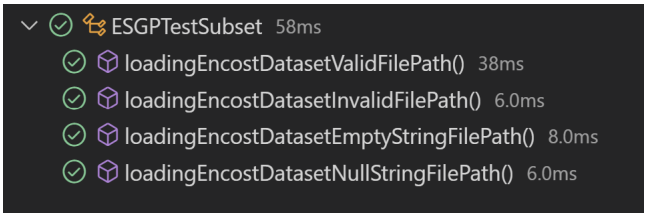

Figure 4.3: Passes tests for Section 3.4

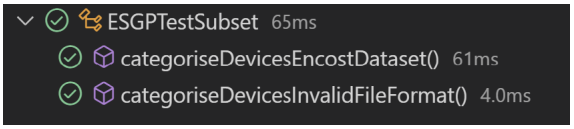

Figure 4.4: Passes tests for Section 3.5

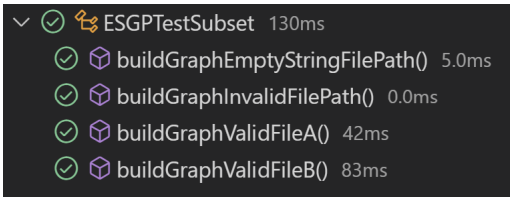

Figure 4.5: Passes tests for Section 3.6

4.2.5 Sprint Task Completion

Table 4.2 shows the status of each product at the end of the sprint. All products were completed within the sprint, and none were carried onto the next sprint.

Product	Completion Date
Section 3.4	12/05/2023
Section 3.5	12/05/2023
Section 3.6	12/05/2023

Table 4.2: Completion dates of each product within sprint 2

4.3 Sprint #3 <13/05/2023 to 16/05/2023>

Sprint 3 was intended to be from 14/05/2023 to 16/05/2023, but due to the early completion of Sprint 2, the start date of Sprint 3 was pushed forwards to 13/05/2023 while the end date remains the same.

As of 15/05/2023, the software requirements were changed and Section 3.8 Device Distribution does not need to be implemented at this stage.

This means that the only Product in this sprint is Section 3.7.

4.3.1 Product Backlog Items

- Section 3.7

This product was chosen for this sprint as it displays the results from sprint 2. It is the last product left to implement.

4.3.2 Sprint Tasks

- Section 3.7
  - Create GraphVisualiser Class
  - Add devices to graph as nodes
  - Format the graph nodes
  - Add connections to graph as edges
  - Check against tests

4.3.3 Software Design

In the SDS, the graph visualisation UI shows that a two way relationship (the device can both send and receive) is shown by a two ended arrow. This was implemented using the GraphStream library as 2 separate, directed lines.

4.3.4 Software Testing

Testing was not completed within this sprint.

4.3.5 Sprint Task Completion

Table 4.3 shows the status of each product at the end of the sprint. The only product in this sprint was half completed. The product was implemented and passed visual tests, but had not passed JUnit tests. The testing part of this product is being carried onto the next sprint.

Product	Completion Date
Section 3.7 without tests	15/05/2023
Section 3.7 testing	In progress

Table 4.3: Completion dates of each product within sprint 3

4.4 Sprint #4 <17/05/2023 to 21/05/2023>

The intention of this Sprint is to provide additional time before the date of software completion to allow for Backlog items to take longer than expected, or other unforeseen circumstances.

As of 12/05/2023, an extension was provided resulting in the software due date being extended from 19/05/2023 to 21/05/2023. The result of this was the extension of this sprints end date from 19/05/2023 to 21/05/2023.

4.4.1 Product Backlog Items

The testing of Section 3.7 was not completed in the previous sprint and is being carried onto this sprint.

4.4.2 Sprint Tasks

- Section 3.7
  - Create methods for testing
  - Complete JUnit tests

4.4.3 Software Design

The JUnit tests for these appeared to not work due to compiling issues with the GraphStream library. To be able to run the ESGP, GraphStream must be included in the command line to both compile and run the software. But as the JUnit tests are being ran through Visual Studio Code, these lines stating it is compiled with GraphStream is not included and therefore it cannot run tests when GraphStream is used. Instead, the

methods required to test were still included, and they are being manually tested through outputting the values to the console as well as performing visual tests on the UI graph window.

4.4.4 Software Testing

Figure 4.6 shows the visual result from adding data from two households into the dataset. These following requirements were tested and passed:

- The device type appears for each node
- The node colour relates to the device type
- The arrows represent the send/receive of each device
- Devices within the same household are grouped together
- Devices are connected to only one router

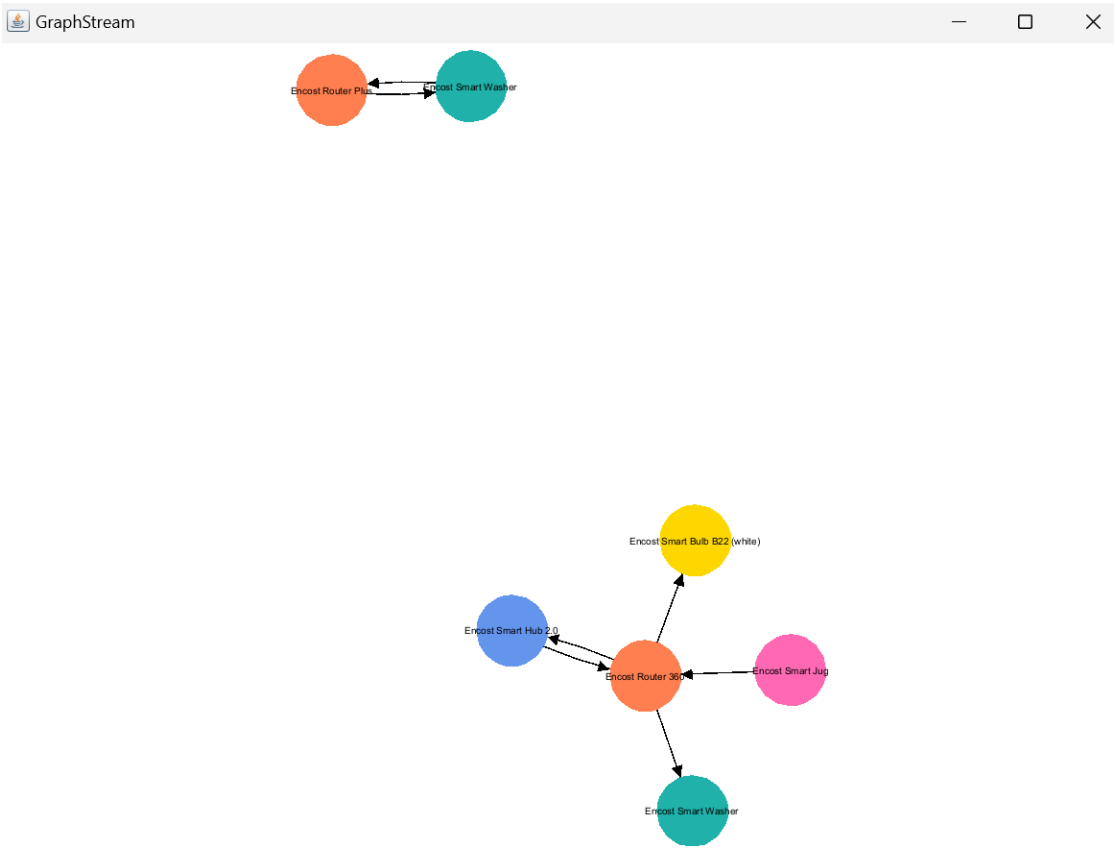

Figure 4.6: Passes visual test for Section 3.7

4.4.5 Sprint Task Completion

Table 4.4 shows the status of each product at the end of the sprint. The only product (which was testing) was completed, and therefore all requirements for the software were completed within the timeframe.

Product	**Completion Date**
Section 3.7 testing	17/05/2023

Table 4.4: Completion dates of each product within sprint 4

5 Conclusion

The ESGP created in this Development Planning Document has the features:

- Categorise a user (community or encost)
- Allows encost user to login
- Shows relevant features for each user type
- Loads ESH dataset
- Finds the device category from its type
- Creates a graph data structure t store devive data
- Visualises the data as a graph using the GraphStream library

All required features were implemented within the time frame, and all features passed the relevant tests (JUnit/visual).