SOFTWARE DESIGN SPECIFICATION

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student # 1
SoftFlux Engineer

SoftFlux

April 7, 2023

Contents

List of Figures

List of Tables

Revision History

Name	**Date**	**Reason for Changes**	**Version**
SoftFlux	24/03/2023	Added information for section 1	0.1
SoftFlux	31/03/2023	Added information for section 2	0.2
SoftFlux	05/04/2023	Added information for section 3	0.3
SoftFlux	07/04/2023	Added information for section 4	0.4
SoftFlux	07/04/2023	Added information for section 5	0.5

1 Introduction/Purpose

1.1 Purpose

This document is Software Design Specification for the software Encost Smart Graph Project (ESGP) for the company Encost, which has the purpose of optimising the use and connectivity between devices. This software allows the user to view the devices and their connections, provides graph visualisation of Encost's devices, as well as providing summary statistics about each device.

The purpose of this Software Design Specification is to provide software architecture, component and user interface designs to aid in the implementation and testing of the software.

1.2 Document Conventions

This document uses the following conventions:

ESGP: Encost Smart Graph Project

1.3 Intended Audience

- Developer: Uses this document to implement software.
- Tester: Uses this document to understand what the program should do, and what should be tested.

1.4 Scope

SoftFlux is only responsible for the base version of the software. This version is only in English and includes the functionalities:

- Handling input and output
- User login
- Graph visualisation
- Data processing and statistics

The Encost Smart Graph Project does not integrate with any hardware.

2 Specialized Requirements Specification

The software should allow both non-Encost (Community) and Encost members to login. Depending on the type of member, different features are available. Both types of members are able to view a graph representation of the Encost Dataset. This is viewed in a UI window and allows the user to see how the different devices are connected, along with their device category and whether they can send and/or receive data. Only Encost can view summary statistics. This provides information regarding device distribution, location, and connectivity for the dataset. Only Encost members are able to unload custom datasets, which they are then able to view the graph representation of it, as well as view it's summary statistics.

2.1 Functional Requirements

The Encost Smart Graph Project should be implemented with Java version 1.8.0 or higher using object-oriented solutions. It should be designed for operating system Windows 10. UTF8 encoding should be used.

Region names and codes are according to the ISO 3166-2 standard published by the ISO 3166 Maintenance Agency (ISO 3166/MA).

2.1.1 Error checking and handling

Where user input is required, should ensure that all forms of incorrect input are considered and caught. An appropriate error message must be displayed to the user stating the cause of the error. Error checking also needs to occur when reading in a custom dataset to ensure data is in the expected format.

2.1.2 Security

Ten username and password pairs are provided and should be stored within the application. Before storing, the **password must be encrypted**. When an Encost user enters their username and password, the entered password must be encrypted and then compared to the stored encrypted password to check validity.

Assuming that the Encost Dataset has anonymity. And that all custom datasets must be anonymised before using with the software.

2.2 Non-functional Requirements

2.2.1 Performance

The graph visualisation function should take no longer than 5 seconds to load. All other functions should take no longer than 10 seconds to load. In the case that a function takes longer than 1 second to complete, the user should be provided with feedback stating that the function is taking longer than expected.

2.2.2 User input

User should always know when they are/are not required to enter input. This should be clear, and the options should be easy to understand.

2.2.3 Console output

When required for the system to output to the console, data should be outputted in a clear and concise matter.

2.2.4 Software quality

- Data loss must be prevented as much as possible
- Source code should be backed up daily
- Textual layout must be clear and readable
- Language must be concise and easy to understand

2.3 Assumptions

- Each household only has one router
- Every device in a household is connected to the households router (or to an extender which is connected to the router)
- Hubs can send/receive to and from all the devices (which have sending/receiving capabilities) within the household

- A device must send and/or receive, a device cannot not send and not receive
- Wifi Extenders complement Wifi Routers. Assuming that devices can be connected to extenders which are connected to routers

3 Software Architecture

3.1 Component Diagram

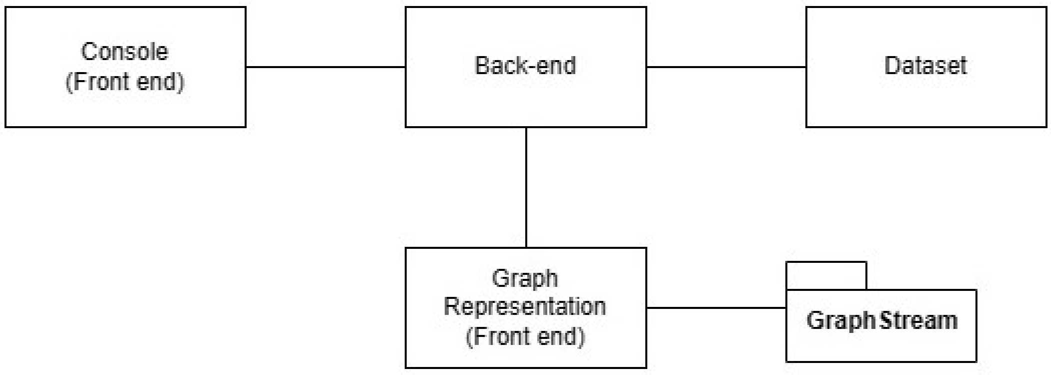


Figure 3.1: Component Diagram of the system

The component diagram shows the interactions between the different components of the system. There are two front end components to the system - one being a Console application, and the other being a UI graph representation. The UI graph representation implements the GraphStream library. Both of these front ends communicate with the backend to display information. The dataset is uploaded and stored within the backend of the application.

3.2 Deployment Diagram

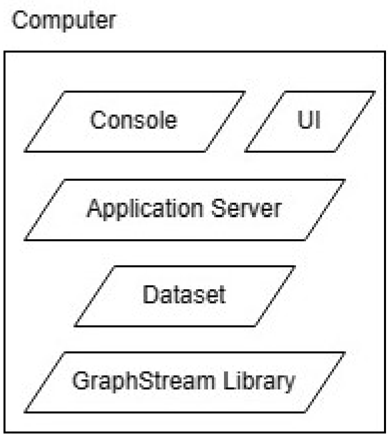


Figure 3.2: Deployment Diagram of the system

The deployment diagram shows the relationship between multiple systems. This software is entirely **stored locally on the device**. The software is not connected to the internet or any other network or device. There is no backup to another server. The dataset should be located locally, and there is no external database. This is shown on the diagram as everything being stored on just the one computer.

3.3 Flow Diagram

Figure 3.3 shows all the different valid paths/options through the entire software. This does not include what happens if the user enters invalid input (option does not exist). These errors should be appropriately handled.

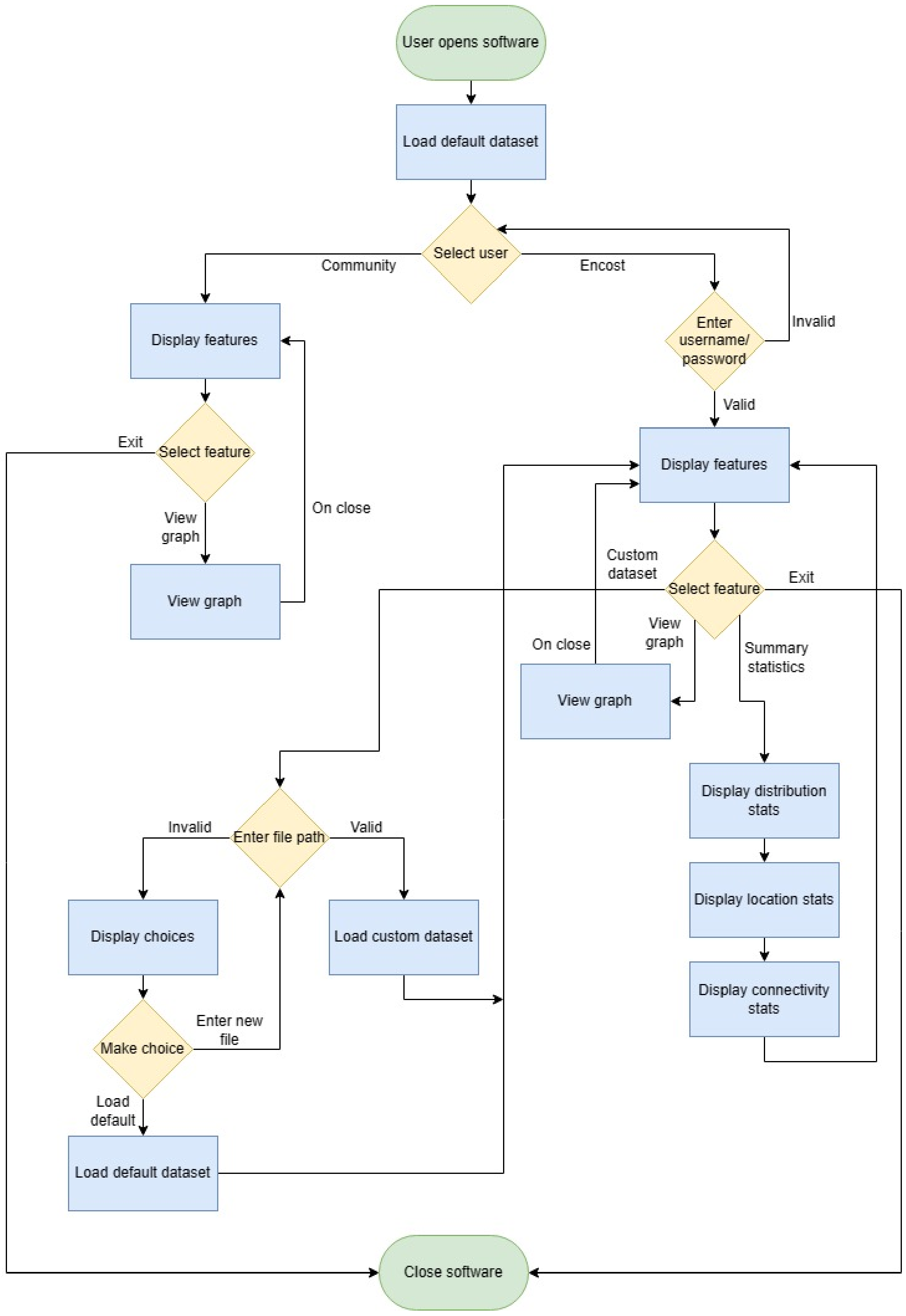


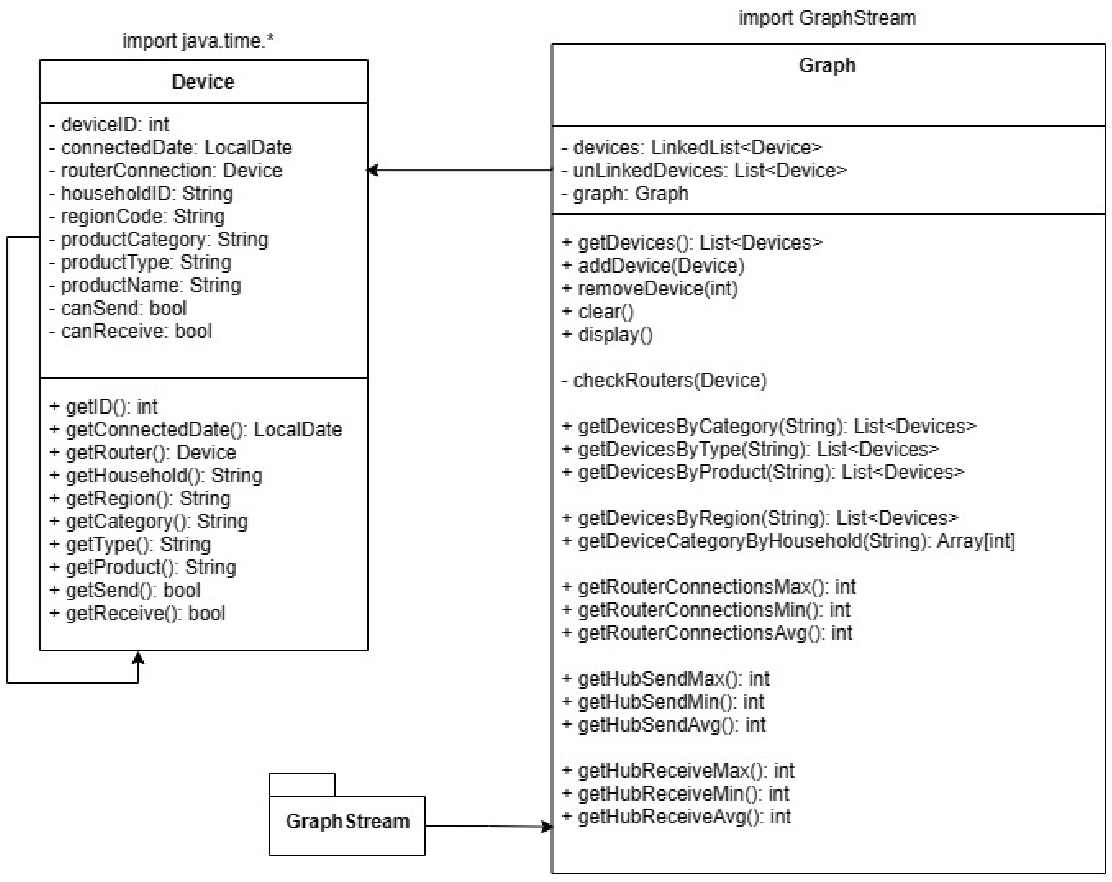
Figure 3.3: Flow diagram showing valid paths through the system

3.4 Frameworks

GraphStream version 1.3 - this framework is for building and visualising dynamic graphs. Some methods of this framework include: addNode (to add a device), addEdge (to add a connection between devices), and setAttribute (to allow formatting of nodes and edges).

4 Component Design

4.1 Class Diagram



import java.time.*
Device
- deviceID: int
- connectedDate: LocalDate
- routerConnection: Device
- householdID: String
- regionCode: String
- productCategory: String
- productType: String
- productName: String
- canSend: bool
- canReceive: bool
+ getID(): int
+ getConnectedDate(): LocalDate
+ getRouter(): Device
+ getHousehold(): String
+ getRegion(): String
+ getCategory(): String
+ getType(): String
+ getProduct(): String
+ getSend(): bool
+ getReceive(): bool
import GraphStream
Graph
- devices: LinkedList<Device>
- unLinkedDevices: List<Device>
- graph: Graph
+ getDevices(): List<Devices>
+ addDevice(Device)
+ removeDevice(int)
+ clear()
+ display()
- checkRouters(Device)
+ getDevicesByCategory(String): List<Devices>
+ getDevicesByType(String): List<Devices>
+ getDevicesByProduct(String): List<Devices>
+ getDevicesByRegion(String): List<Devices>
+ getDeviceCategoryByHousehold(String): Array[int]
+ getRouterConnectionsMax(): int
+ getRouterConnectionsMin(): int
+ getRouterConnectionsAvg(): int
+ getHubSendMax(): int
+ getHubSendMin(): int
+ getHubSendAvg(): int
+ getHubReceiveMax(): int
+ getHubReceiveMin(): int
+ getHubReceiveAvg(): int
GraphStream

Figure 4.1: Class diagram showing Device and Graph classes

4.1.1 Device Class

Purpose: To hold information about individual devices

Scope: All attributes of this class are private scope which can only be accessed through the public get methods. Only methods for this class are get methods for all of the private attributes. There are no setting methods as the user should not be able to change any of the device values. There are no private methods required.

Attributes:

- deviceID - the unique ID for a device
- connectedDate - date the device was connected
- routerConnection - links to the device object of the router this device is connected to. If the device is a router, this should remain null
- householdID - the id of the household the device is part of
- regionCode - the ISO 3166-2 code for the region the household is part of
- productCategory - the overall category the device is part of (Encost Wifi Routers, Encost Smart Lighting, etc.)
- productType - the type of device (router, extender, light bulb, etc.)
- productName - the specific name of the product (Encost Router 360, Encost Smart Bulb B22 (white), etc.)
- canSend - boolean value of whether the device can send commands
- canReceive - boolean value of whether the device can receive commands

Note: the region code can be extracted as part of the household ID, and the product category can be found using a look-up table using the type.

4.1.2 Graph class

Purpose: To hold the devices in a graph data structure to show connections

Scope: All attributes are private. Contains both public and private methods; where public methods return data used in summary statistics as well as the graph visualisation, and the private methods aid in the LinkedList formation. This class implements the GraphStream Framework.

Attributes:

- devices - linked list holding all the devices as Device objects, showing the connections between the routers and it's connected devices
- unLinkedDevices - list of devices whose router does not exist in the linked list. These are later added when the router is added or the graph is displayed
- graph - GraphStream graph object holding the appropriate device data as nodes and edges

Methods:

- getDevices() - returns a list of all the devices
- addDevice(Device) - adds the Device object to the linked list. If the device is a router, add it and loops through unlinked devices to check if they are connected to that router. Else, check if its router is in the graph; if so, add the appropriate link, else add the device to the unlinked device list **SEE FIGURE FLOWCHART**
- removeDevice(Device) - removes a Device from the LinkedList
- clear() - removes all the data/clears the lists and graph to allow for custom dataset upload
- display() - displays the graph representation of the data using the GraphStream library
- checkRouters(Device) - private method which loops through the unlinked devices to check whether its router is the Device passed in, so that it can be added to the LinkedList
- getDevicesByCategory(String)/getDevicesByType(String)/getDevicesByProduct(String) - returns a list of Devices where all the Devices are or are part of the passed in category/type/product
- getDevicesByRegion(String) - returns a list of Devices which are part of households in the passed in region, where the passed in region follows the ISO 3166-2 code
- getDeviceCategoryByHousehold(String) - returns an int Array showing the number of devices in each category, where each position indicates one of the categories in the household passed in
- getRouterConnectionMax()/getRouterConnectionMin()/getRouterConnectionAvg() - returns the max/min/avg number of devices routers are connected to
- getHubSendMax()/getHubSendMin()/getHubSendAvg() - returns the max/min/avg number of devices that hubs can send commands to
- getHubReceiveMax()/getHubReceiveMin()/getHubReceiveAvg() - returns the max/min/avg number of devices that hubs can receive commands from

4.2 Loading data

The default dataset is the Encost Smart Homes Dataset. This dataset contains anonymous devices around New Zealand within the time frame April 2020 - April 2022 This should be stored locally, with the location known. Each line of this CSV dataset contains one device. Each line should then be split to extract the information which a Device object can be created from (see Figure 4.1). If the user uses a custom dataset, the format of this CSV must be the same as the default dataset format. Appropriate error checking and handling should still occur. The format of this CSV is as follows:

device ID, date connected, device name, device type, household ID, router ID, can send, can receive
Example 1: EWR-1234,01/04/22,Encost Router 360,Router,WKO-1234,-,Yes,Yes
Example 2: ELB-4567,01/04/22,Encost Smart Bulb B22 (multi colour),Light bulb,WKO-1234,EWR-1234,No,Yes

The router ID contains the device ID of the router it is connected to. The household ID starts with the 3 letter region code followed by a hyphen, then a unique number ID to distinguish between households.

After the Device object is created, the object should get added to the Graph object which holds the devices in a graph data structure. This enables the connections (between the router and devices) to be stored. These are stored in a Linked List acting as an Adjacency List. An issue which may occur is if a device is added to the graph structure before its router is added. Figure 4.2 shows the process of inserting the Device object into the graph data structure.

Note: the default dataset should be read, processed, and stored when the software is opened (before the user selects their user type).

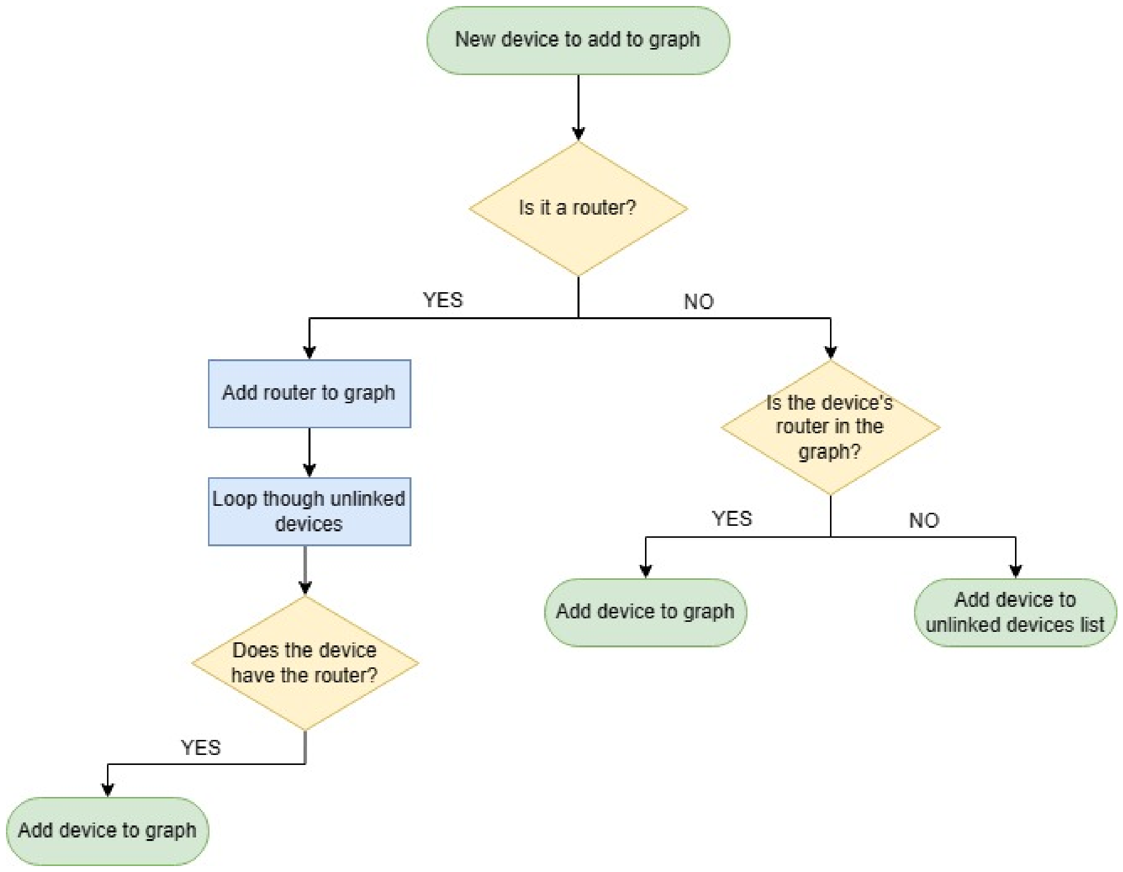

New device to add to graph
Is it a router?
YES
NO
Add router to graph
Loop though unlinked devices
Does the device have the router?
YES
Add device to graph
Is the device's router in the graph?
YES
NO
Add device to graph
Add device to unlinked devices list

Figure 4.2: Flow diagram showing how devices should be added to the graph

4.3 Look-up dictionaries

A look-up dictionary is needed to show the connection between a devices category and its type. The category of the device needs to be stored within the Device object, but only the type is provided. Therefore a dictionary is required to find the category from the type. Table 4.1 is an excerpt of the Encost products. It shows that the dictionary would have the relationship between 'Router' and 'Wifi Routers' as well as 'Extender' and 'Wifi Routers'.

Table 4.1: Encost Products

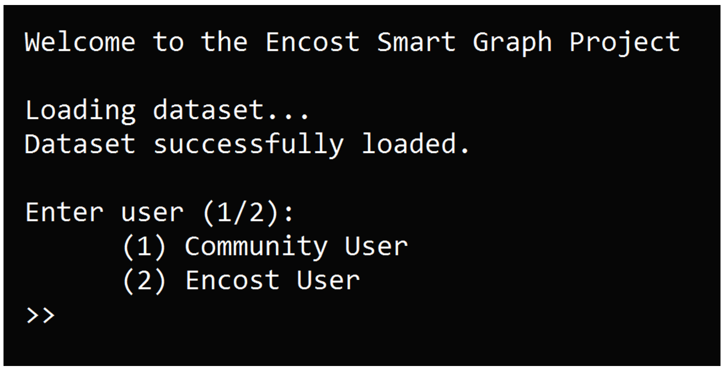
Category	Type	Product
Encost Wifi Routers	Router, Extender	Encost Router 360, Encost Router Plus, Encost Wifi Range Extender 1.0, Encost Wifi Range Extender 2.0
Encost Hubs/-Controllers	Hub/Controller	Encost Smart Hub, Encost Smart Hub 2.0, Encost Smart Hub Mini

Another dictionary should be provided for the region codes and their English name. Household regions are categorised through 3 letter codes. In the summary statistics, to make is clearer for the user, the name of the region should be displayed along with the code. This means a dictionary is needed to link 'WKO' to 'Waikato' and 'BOP' to 'Bay of Plenty'.

4.4 User Interface

4.4.1 Welcome and login

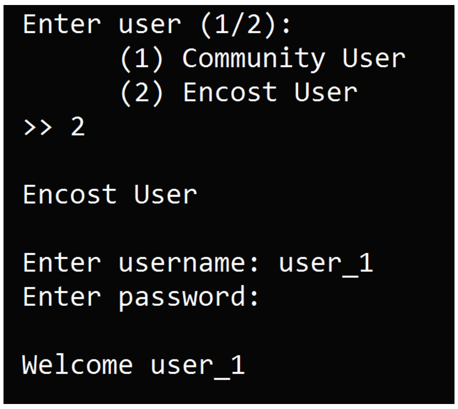
A welcome prompt (see Figure 4.3) should be displayed to the Console. It should show the two different types of members. The user should enter a number corresponding to their member type. The need for input should be made clear to the user.



```
Welcome to the Encost Smart Graph Project

Loading dataset...
Dataset successfully loaded.

Enter user (1/2):
      (1) Community User
      (2) Encost User
>>
```

Figure 4.3: Welcome prompt

If the Encost user has been selected, the user should be prompted to enter their username and password (see Figure 4.4). If unsuccessful, the user should be made aware.



```
Enter user (1/2):
      (1) Community User
      (2) Encost User
>> 2

Encost User

Enter username: user_1
Enter password:

Welcome user_1
```

Figure 4.4: Logging in as an Encost User

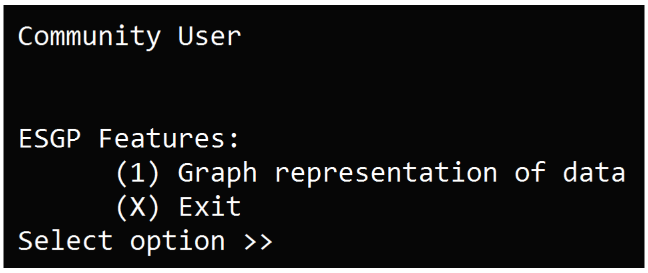
Note: the default dataset should be loaded before the user options are displayed. This should display a message to the user.

4.4.2 Features

Depending on the user type selected from Section 4.4.1, different features should be displayed for the user.

- **Community user** (see Figure 4.5) - only has the option to view the graph representation of the data
- **Encost user** (see Figure 4.6) - has the option to; load a custom dataset, view the graph representation of the data, or view summary statistics of the dataset

After being displayed, the user should be prompted to enter the feature number to view. The need for input should be made clear to the user.



```
Community User


ESGP Features:
      (1) Graph representation of data
      (X) Exit
Select option >>
```

Figure 4.5: Features available for community users

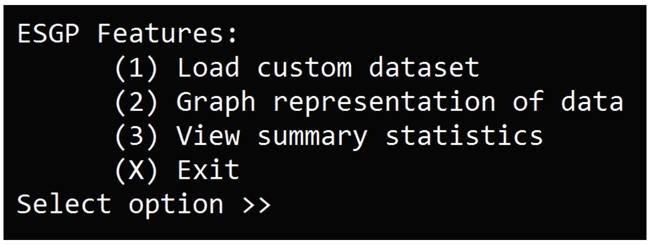
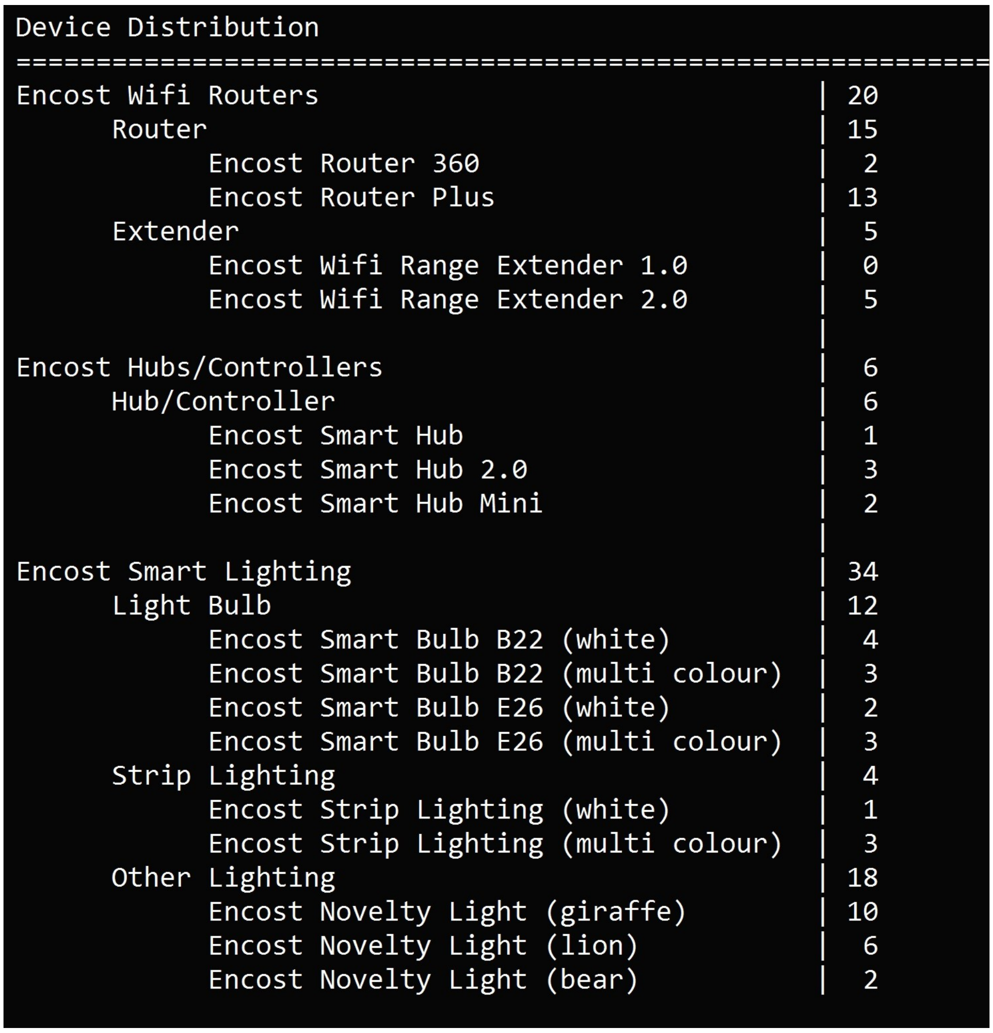

```
ESGP Features:
      (1) Load custom dataset
      (2) Graph representation of data
      (3) View summary statistics
      (X) Exit
Select option >>
```

Figure 4.6: Features available for Encost users

4.4.3 Summary Statistics

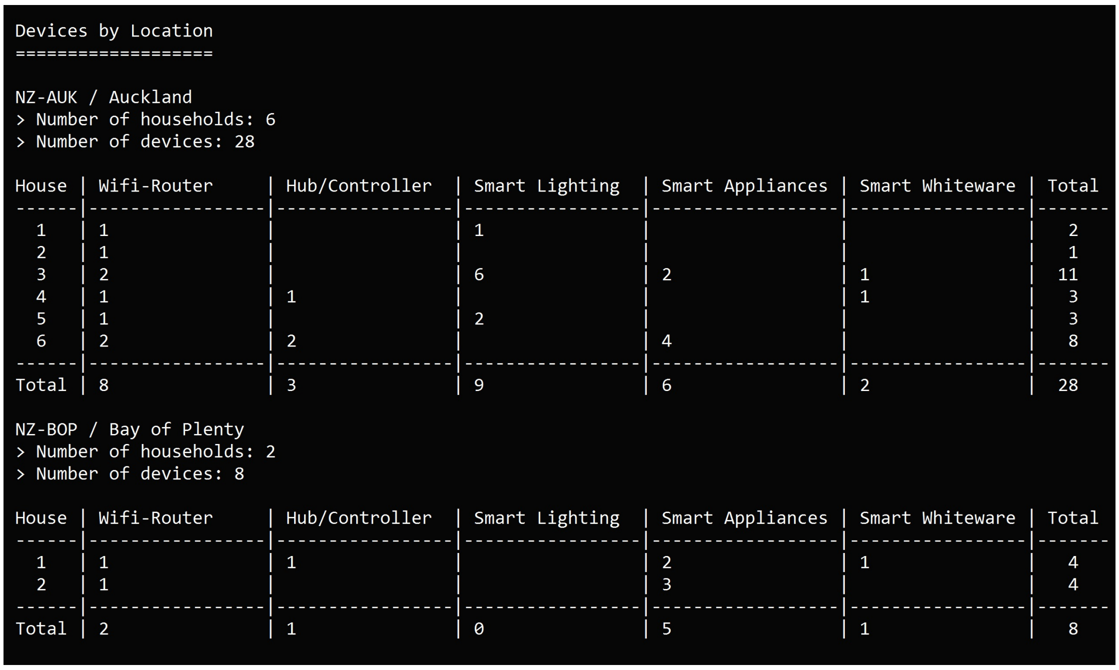
This feature outputs statistics about the Encost devices to the console.

1. **Device Distribution** (see Figure 4.7) - Information should be written to the Console listing the quantity of devices each Encost device category, type, and product exist in the dataset.
2. **Device Locations** (see Figure 4.8) - Information should be written to the Console listing the number of devices and households per region (refer to section 2.1). This should be shown as well as the number of devices in each category in each household.
3. **Device Connectivity** (see Figure 4.9) - Information should be written to the Console including the min/avg/max devices Encost Wifi Routers are connected to, Encost Hubs/Controllers that Encost Smart Devices can receive from, and Encost Hubs/Controllers that Encost Smart Devices can send to.



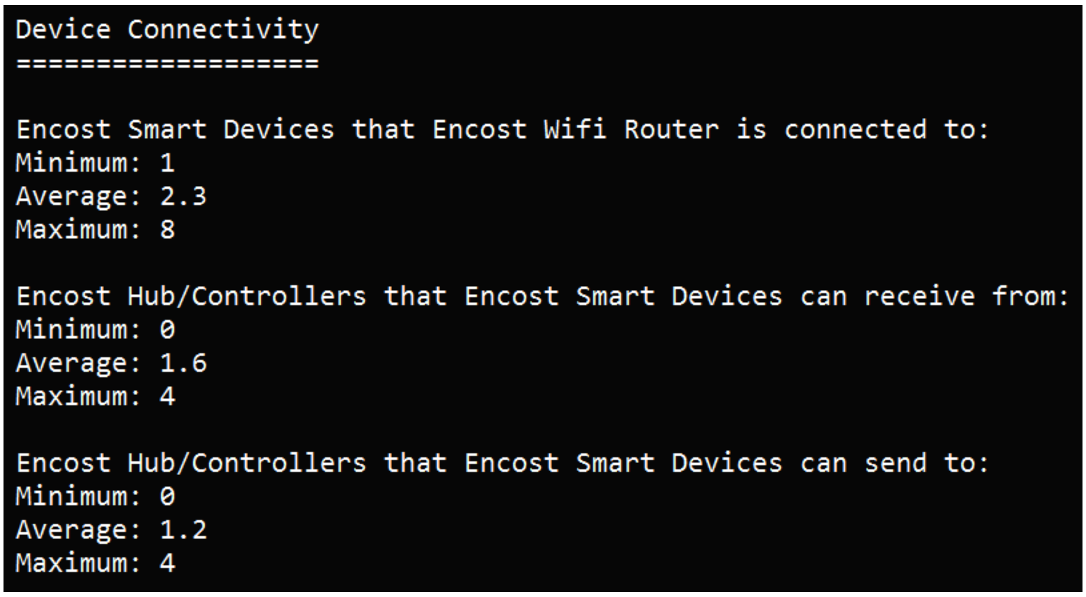
```
Device Distribution
=========================================================================
Encost Wifi Routers                                 | 20
      Router                                        | 15
            Encost Router 360                       |  2
            Encost Router Plus                      | 13
      Extender                                      |  5
            Encost Wifi Range Extender 1.0          |  0
            Encost Wifi Range Extender 2.0          |  5
                                                    |
Encost Hubs/Controllers                             |  6
      Hub/Controller                                |  6
            Encost Smart Hub                        |  1
            Encost Smart Hub 2.0                    |  3
            Encost Smart Hub Mini                   |  2
                                                    |
Encost Smart Lighting                               | 34
      Light Bulb                                    | 12
            Encost Smart Bulb B22 (white)           |  4
            Encost Smart Bulb B22 (multi colour)    |  3
            Encost Smart Bulb E26 (white)           |  2
            Encost Smart Bulb E26 (multi colour)    |  3
      Strip Lighting                                |  4
            Encost Strip Lighting (white)           |  1
            Encost Strip Lighting (multi colour)    |  3
      Other Lighting                                | 18
            Encost Novelty Light (giraffe)          | 10
            Encost Novelty Light (lion)             |  6
            Encost Novelty Light (bear)             |  2
```

Figure 4.7: Summary statistics - device distribution



```
Devices by Location
===================

NZ-AUK / Auckland
> Number of households: 6
> Number of devices: 28

House | Wifi-Router      | Hub/Controller   | Smart Lighting   | Smart Appliances  | Smart Whiteware  | Total
------|------------------|------------------|------------------|-------------------|------------------|-------
  1   | 1                |                  | 1                |                   |                  |   2
  2   | 1                |                  |                  |                   |                  |   1
  3   | 2                |                  | 6                | 2                 | 1                |  11
  4   | 1                | 1                |                  |                   | 1                |   3
  5   | 1                |                  | 2                |                   |                  |   3
  6   | 2                | 2                |                  | 4                 |                  |   8
------|------------------|------------------|------------------|-------------------|------------------|-------
Total | 8                | 3                | 9                | 6                 | 2                |  28

NZ-BOP / Bay of Plenty
> Number of households: 2
> Number of devices: 8

House | Wifi-Router      | Hub/Controller   | Smart Lighting   | Smart Appliances  | Smart Whiteware  | Total
------|------------------|------------------|------------------|-------------------|------------------|-------
  1   | 1                | 1                |                  | 2                 | 1                |   4
  2   | 1                |                  |                  | 3                 |                  |   4
------|------------------|------------------|------------------|-------------------|------------------|-------
Total | 2                | 1                | 0                | 5                 | 1                |   8
```

Figure 4.8: Summary statistics - device locations



```
Device Connectivity
===================

Encost Smart Devices that Encost Wifi Router is connected to:
Minimum: 1
Average: 2.3
Maximum: 8

Encost Hub/Controllers that Encost Smart Devices can receive from:
Minimum: 0
Average: 1.6
Maximum: 4

Encost Hub/Controllers that Encost Smart Devices can send to:
Minimum: 0
Average: 1.2
Maximum: 4
```

Figure 4.9: Summary statistics - device connectivity

4.4.4 Graph representation

Both Community and Encost Users are able to view the graphical representation of the data. This should be implemented using the GraphStream library (see Section 3.4). All nodes/devices should be displayed.

The graph should show the relationship between the devices, as well as a defined way to see the category of each device as well as whether the device can send and/or receive. This should be complemented with a key, informing the user what the different colours and shapes represent (see Figure 4.10).

The colours used for the different categories should be easily distinguishable.

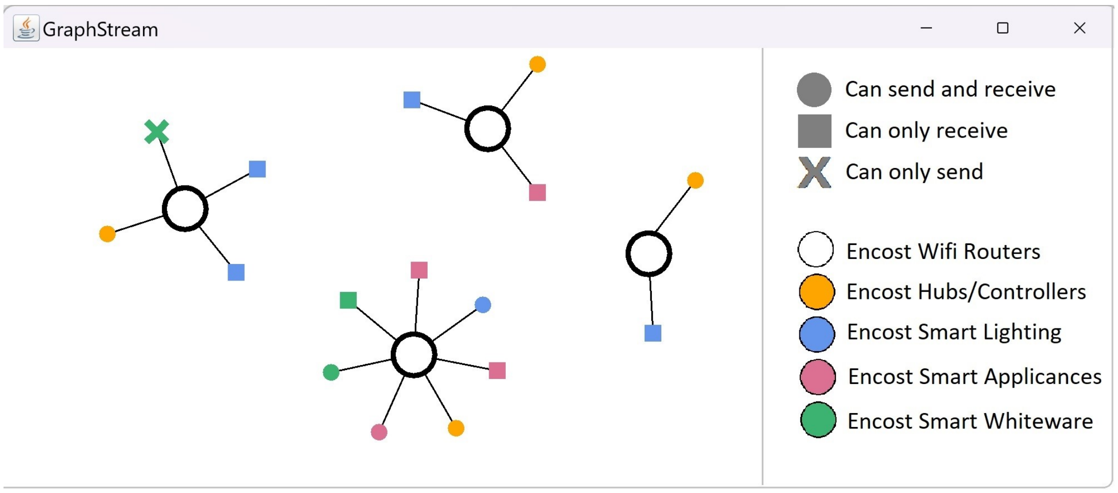
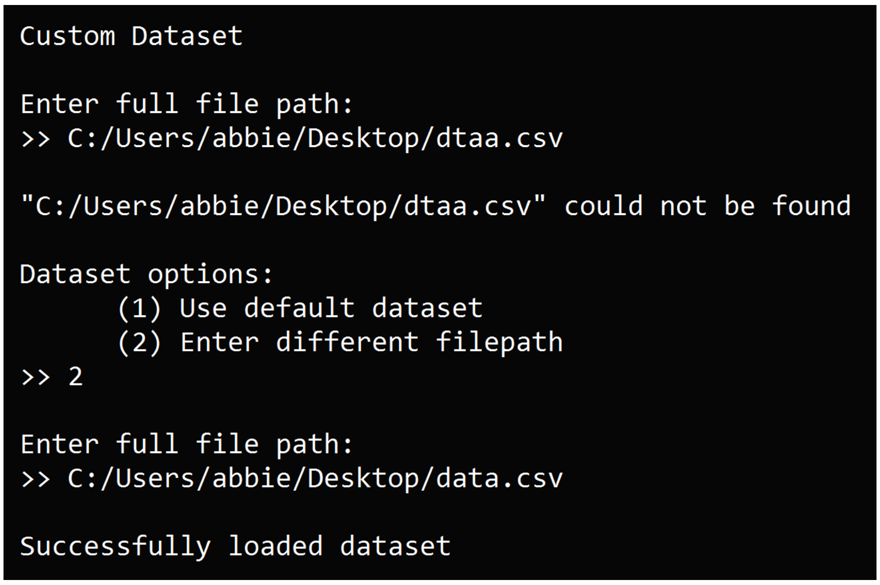


Figure 4.10: Device data represented as a graph

Note: this could be improved for accessibility reasons (such as colourblindness and vision impairment) by using icons instead of colours and shape.

4.4.5 Loading custom dataset

The user is prompted to enter the full path of their custom dataset. If the file path is invalid, the user should be informed and two options should be displayed; use the default dataset, or enter a different file path. If chosen the default dataset, or a valid file is selected, a message should be shown stating that the dataset has been loaded. Figure 4.11 for this process. If the custom dataset contains some invalid lines, an appropriate message should be displayed to the user informing them of this. The custom dataset should still be used.



```
Custom Dataset

Enter full file path:
>> C:/Users/abbie/Desktop/dtaa.csv

"C:/Users/abbie/Desktop/dtaa.csv" could not be found

Dataset options:
      (1) Use default dataset
      (2) Enter different filepath
>> 2

Enter full file path:
>> C:/Users/abbie/Desktop/data.csv

Successfully loaded dataset
```

Figure 4.11: Uploading a valid and invalid custom dataset

Note: the format of the custom dataset needs to be the same as the default dataset (see Section 4.2)

4.5 User Type Requirements

The users type should be stored as a global variable within the main program. This should be stored as 'community' or 'encost-verified'. This value is changed depending on the user selection (see Figure 4.3). For security, before displaying Encost only features, this variable should be checked.

5 Conclusion

This Software Design Specification document has covered the software architecture, component and user interface design, as well as non-functional requirements for the Encost Smart Graphs Program for the company SoftFlux.