SOFTWARE DESIGN SPECIFICATION

for

Encost Smart Graph Project

Version 1.3

Prepared by: Student # 2
SoftFlux Engineer

SoftFlux

April 6, 2023

Contents

List of Figures

List of Tables

Revision History

Name	Date	Reason for Changes	Version
	March 30, 2023	Device Classes are to be future proofed for potential modifications	1.1
	April 03, 2023	Routers per household assumption & error handling	1.2
	April 04, 2023	Correction to requirement section 4.10.3	1.3

1 Introduction/Purpose

1.1 Purpose

The purpose of this document is to ensure that the development, deployment and testing of the software by SoftFlux meets the requirements outlined by Encost. The software system must meet both the functional and non-functional requirements specified in the software requirements document. In doing so this document will describe the software architecture, component design and extra software requirements though figures, graphs tables and diagrams. This will aid the development of the Encost Smart Graph Project (ESGP) and be used for comparison and justification in the testing process.

1.2 Document Conventions

This document will use the following conventions:

ESGP: Encost Smart Graph Project
CLI: Command Line Interface
UI: User Interface

1.3 Intended Audience and Reading Suggestions

This document is intended for use by any individual user, developer, tester, or project manager. Each reader type has their own generalised individual uses for this document outlined below:

- User: A user may wish to use this document in order to understand the software application design and ensure that the software meets the requirement needs for their specific use case.
- Developer: A developer who may require clarification on the software design specifications or to ensure that the software itself is meeting the requirements and design specifications outlined in this document.
- Tester: A tester may require this document to be used as a comparison piece to the final software product they are testing or to create the testing parameters from the specifications outlined in this document.

1.4 Project Scope

This software design specification document is to bed used by the software development and testing teams at Softflux as a definition of the software design and requirements needed by Encost for their Encost Smart Graph Project (ESGP). 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.

2 Specialized Requirements Specification

2.1 Additional/Changed Requirements

The requirements below have been added after discussion with the client (Encost). The changes can also be seen within the revision history.

- The device class must allow future modifications to each specific device is Encost chooses to do so.
- Correction to the SRS section 4.10.3 requirement 3: The system should use the information stored in the graph data structure to calculate the AVERAGE number of devices that exist in each household, in each region in New Zealand.
- Correction to the SRS section 4.10.3 requirement 4: The system should use the information stored in the graph data structure to calculate the AVERAGE number of devices in each category, for each household, in each region in New Zealand.

2.2 Assumptions

These assumptions have been made during the design process and are outlined within their relevant sections. This section is only to be regarded as a summary.

- Each household is only to have 1 router. But not every device within the household may be connected to said router.

2.3 Nonfunctional Requirements

The ESGP is to be designed and implemented using an object-orientated design methodologies. The software must be capable of working offline and integrating with the GraphStream library. The users are assumed to be a user who had their information recorded by Encost between April 2020 to April 2022 and due to this the ESGP must be developed with a broad range of users in mind. Other users which will be making use of this software may require verification to be able to use specific features. Specifically if they are part of Encost they will have verified access to different features.

2.4 Product Requirements

The developed software must be compatible with the Windows 10 system and be developed in Java at a 1.8.0 version or higher. No use of hardware will be involved in the design, or development of the ESGP as outlined in the requirements document.

The naming conventions for this must follow the Java naming conventions:

- Classes: Should be nouns, mixed case with a capitalised letter for the beginning of each new internal word.
- Methods: Should be mixed cased verbs with a capitalised letter for the beginning of each new internal word.
- Variables: Should be mixed case with a lower case first work and a capitalised letter for the beginning of each new internal word. They shall be short but meaningful to increase readability and understanding of the code.
- Constants: Shall be upper case with an underscore '_' denoting the separation of each word.

3 Software Architecture

3.1 Component Architecture Diagram

The component architecture diagram aims to demonstrate the function of the ESGP software components and the relationships between these components within the system. It demonstrates how each component can interact with the connected components which is relevant to the development and testing of the ESGP.

The main component of the ESGP is housed within the application framework. This is the back-end component of the software where all communications between the systems connect to. All the device classes are stored within the application framework. The back-end will be able to communicate with the command-line interface (CLI) where all stimulus given by the user will be received and any responses required by the software will be output.

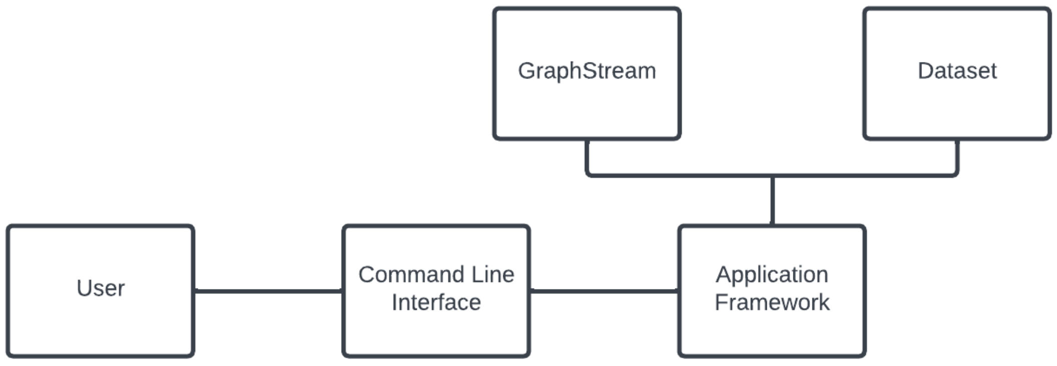


Figure 3.1: The ESGP component architecture diagram.

Below is a description of how multiple components may communicate to complete different functions within the application framework.

The user will be able to login to a verified user account from the CLI. If successful the back-end will communicate with the CLI to update the user-type to ”encost-verified”, where more features from the back-end will become available. Such as, if a verified Encost user chooses to load a custom dataset the back-end will take the stimulus from the CLI containing the full file path for the dataset, using this path the backend will attempt to read in the dataset.

A non-verified user, known as a ”community-user” will only be able to access the visualisation of data with the GraphStream. This feature is still available to verified users and incorporates the use of the GraphStream library in conjunction to the back-end to open a new UI window separate from the CLI to visualise the dataset information which the back-end has loaded in.

This component design has been chosen to be able to be deployed on a single offline system as requested in the software requirement document.

3.2 Deployment diagram

The deployment process of the ESGP is a straight forward process. Because the system has no hardware components and the UI is entirely through the CLI the entire deployment can be within a single device. There is no requirement or use of hardware or the internet for this project. The application can communicate with the dataset and the CLI in order to produce the functions required for the ESGP. The process for development is through object orientated programming (OOP) due to this each of the individual classes in section 4.1 are to be created in separate '.java' files. This way the application can use the classes in a clean and organised way using standard OOP conventions.

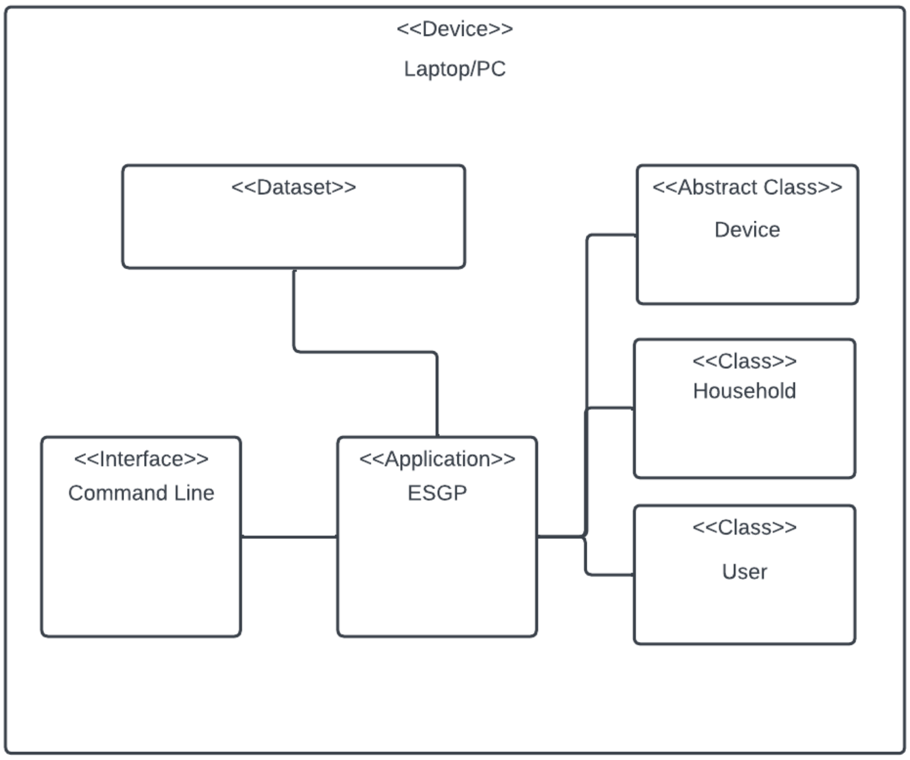


Figure 3.2: The ESGP deployment diagram.

3.3 Use Case Diagram

The use case diagram found in figure 3.3 describes the two simple use cases for the ESGP. There are two main actors within the ESGP use case story, the community user and the Encost verified user. There are five different use cases defined for this system: show features, account login, graph visuals, view summary statistics and load custom dataset. The UI design can be found in section 4.2 and the software functional requirements can be found in the requirements documentation.

As figure 3.3 shows the community user only has access to two use cases the graph visuals and show features section. Whereas, a Encost verified user is able to access all available use cases. The use case figure 3.3 also demonstrates how the use cases 'load custom dataset' and 'view summary statistics' derive information from another use case 'graph visuals' This is as the information which the graph object is to hold is the same information which these use-cases require in order to perform their functions. By having these use cases derive their information from the graph structure the code length will be minimised along with the code re-usability for the entire ESGP will increase.

3.4 Process Diagram

3.4.1 Dataset Read and Load

A verified Encost user is able to choose a custom dataset, or use the default Encost Smart Homes Dataset. A community user will only ever be able to use the default dataset. If the ESGP is unable to read and load in the custom dataset chosen it should give the option to use the default or allow to user to reattempt submitting a file location. A visual representation of this user interaction can be shown in section 4.2.4. The process diagram in figure 3.4 shows an outline of the process steps the system should undertake in order to read in a new custom dataset.

First the user should enter the file location of the new dataset. The ESGP should traverse the file system and find the dataset. If the file is not file the ESGP will return to ask the user to re-attempt entering the file name. If successfully the ESGP will read in and load the dataset information into the class structure outlined in section4.1. If the dataset if invalid again the system should return to give the user another attempt to give a valid dataset file. A successful read of the dataset should return the user to the 'encost-verified' features prompt as shown in figure 3.4. Visual aids of this process within the CLI are shown in section 4.2.

The default database is the 'Encost Smart homes Dataset'. The location of this dataset should be in a folder labeled '\datasets' which must be located in the same directory as the ESGP source program. The default database must be named 'ESH_dataset.csv'. With 'ESH' being short for Encost Smart Homes. All databases used by the ESGP must be a '.csv' formatted file.

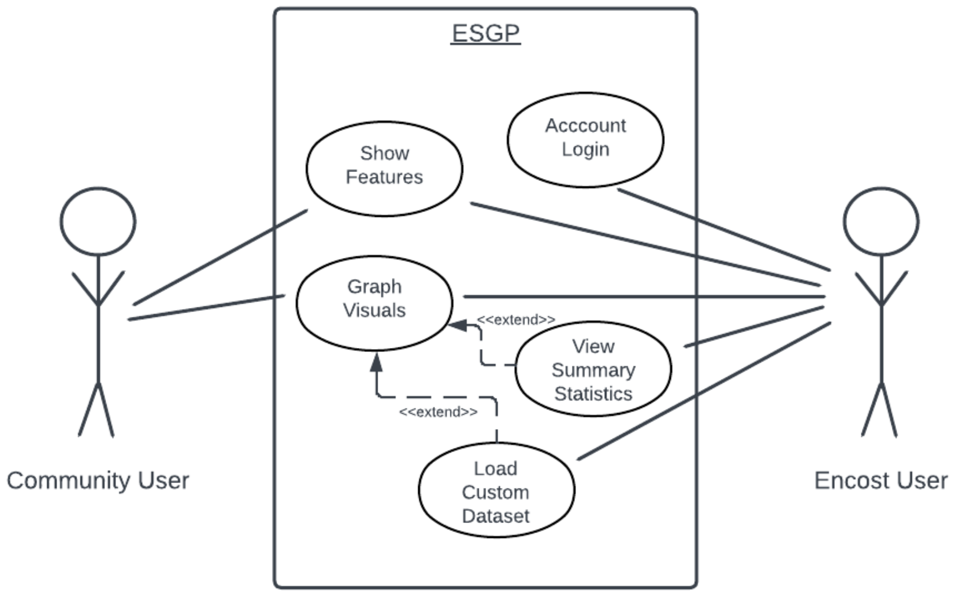


Figure 3.3: The ESGP use case diagram.

3.5 Sequence Diagram

3.5.1 User Verification Process

When a user selects to login in as a verified Encost user the ESGP will have a complete a process to verify the user and their login credentials. A UI design representation of this can be shown in section 4.2.2. However, the process which needs to be completed in order to verify a user is more complicated than this UI representation shows. In figure 3.4 the separation between the two user types can be seen through the ESGP process. For the specific encryption process for the passwords used in the ESGP for verification of users see section 5.1 security.

For both valid and invalid login attempts by a user the sequences which the ESGP application undertakes is the same. As shown in figure 3.5 the first steps of communication between the user and the applications initialises the login sequence. The second

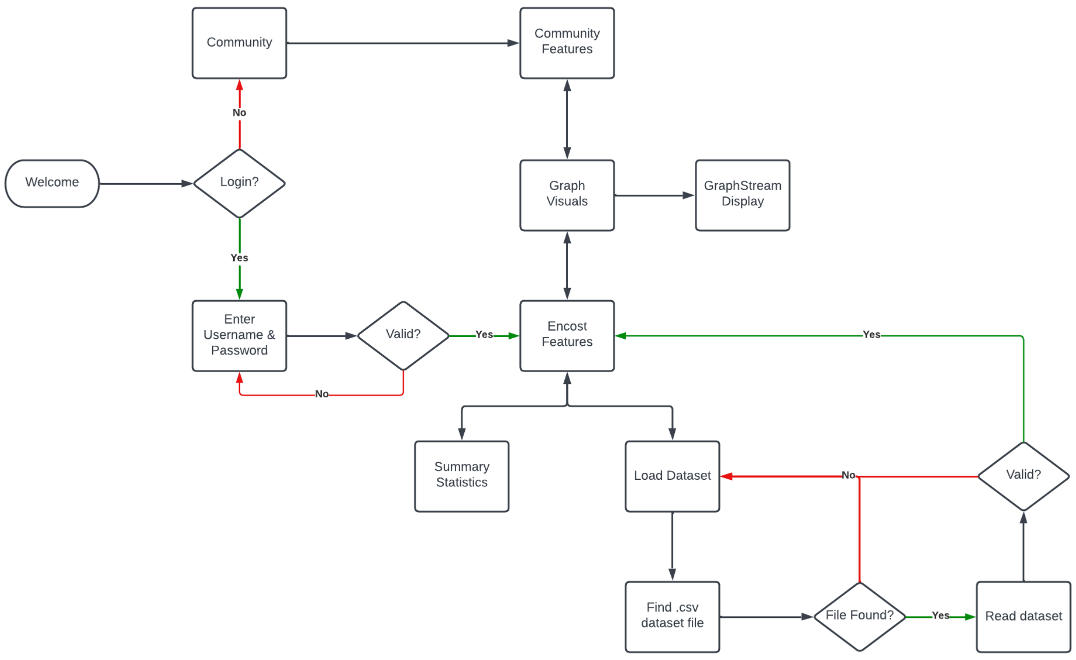


Figure 3.4: The ESGP process diagram.

round of input from the user gives the system the username and password of the user though the CLI. Throughout this process the password is not shown as shown in figure 4.7. The ESGP must then validate this information (both username and password must match those in the system). This validate input process uses the Java.security package as outlined in section 5.1 security. If the password and username match an encrypted pair within the system then the response can reflect this success and inform the user through changing the prompt to the features page. This should display to the user 'encost-verified edition' as shown in section 4.2. If this validation process is unsuccessful again the system must reflect this failure and respond to the user by changing the prompt to as for the user to re-enter their details. This system must also allow users to 'back' out of this option and user the community edition.

4 Component Design

4.1 Individual Classes

4.1.1 Users

Super-Class: Community-User

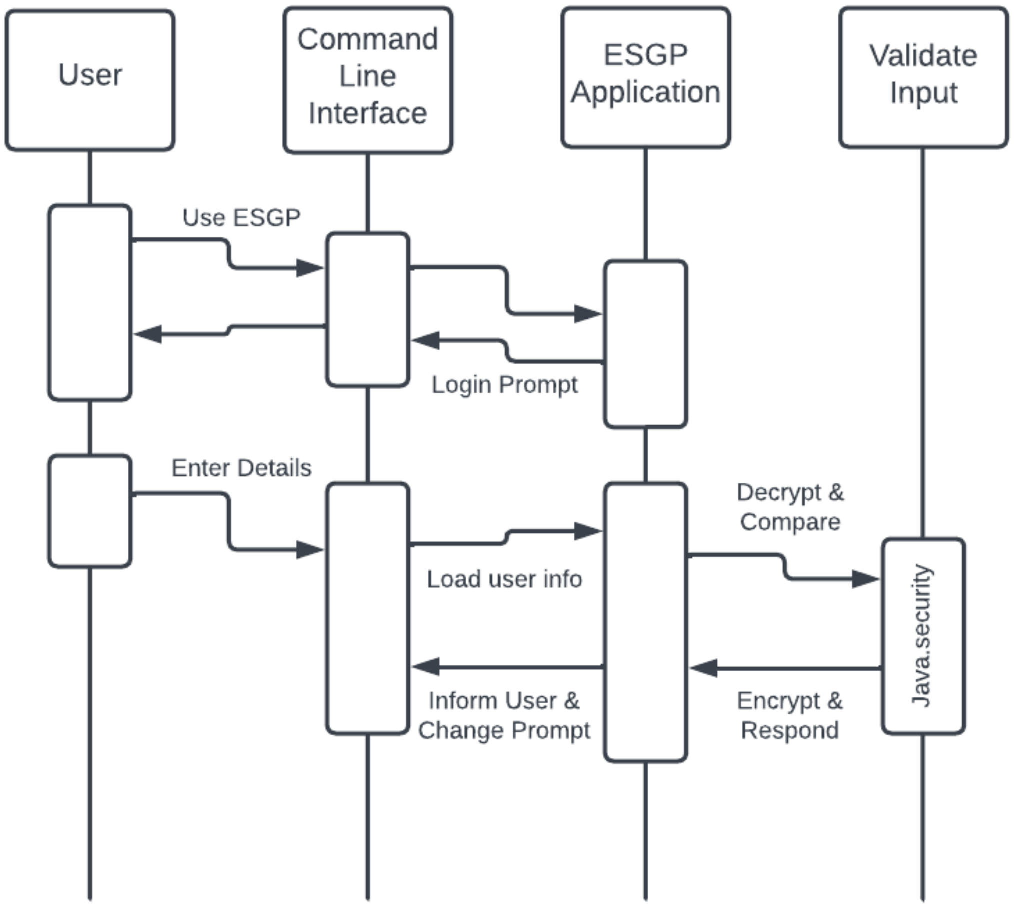


Figure 3.5: The ESGP login sequence diagram.

The community user is classes as the parent class to the verified user. This is because the only function held within the community user is graphVisual(), this function is also required by the verified encost-user. As per the requirements the community user hold no further information at this point.

Function: GraphVisuals()

Using a graph data structure with the stored Encost Smart device objects a new UI window will open and show a graph which will visualise nodes and edges to display the connections between each device.

Sub-Class: Encost-User

The sub-class of the community user class is the 'encost-user', or verified user.

Attributes

- Username

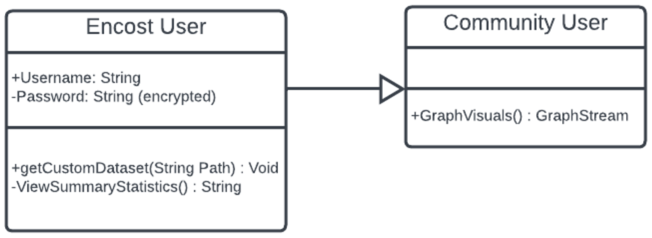

Encost User
+Username: String
-Password: String (encrypted)
+getCustomDataset(String Path) : Void
-ViewSummaryStatistics() : String
Community User
+GraphVisuals() : GraphStream

Figure 4.1: The ESGP user class diagram.

The verified user must be able to sign in to a valid verified account, using a username. This attribute is where this information will be stored. It is a Java string variable named 'username'.

- Password

  The verified user must be able to sign in to a valid verified account, with a password.This attribute is where this information will be stored. It is a Java string variable named 'password'. This password is an encrypted variable. The specifications for the encryption of this password will be found in section 5.1.

Functions

- GetCustomDataset(String path)

  A verified user will be able to load in a custom dataset for the ESGP to load. A path to the dataset must be passed in as a string. The application framework will attempt to open, read and process the dataset for the user.

- ViewSummaryStats()

  This function allows the verified user to view the summary statistics on the device distribution, connectivity and location. This is returned to the CLI as a table format.

4.1.2 Household

Each household which is held within a dataset gets loaded into the household class. This is where all information about a specific household that had its information gathered is kept.
Attributes

- House_id

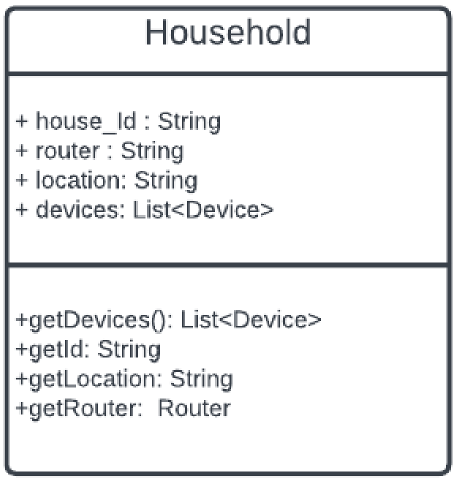


Figure 4.2: The ESGP household class diagram.

A Java string identifier for each individual household. This must be unique for each separate household. Example: 'WKO-1234'.

- Router

  A Java string to identify the router device by id which the house is using. Example: 'EWR-1234'. Assumes that each households only has one router.

- Location

  The region of which the household it located in. Used for summary statistics and house ID which includes the location.

- Devices

  A list of device objects which are connected within the household.

Functions

- GetDevices()

  Returns a list of all devices in the household. Each device is written as a Java string as the device id.

- getId()

  Returns the household id as a Java string.

- getLocation()

  Returns a Java string of the households regional location. Example: 'WKO'.

- getRouter()

  Returns the router id which is being used in the household as a Java string.

4.1.3 Device

The device class is an abstract class which holds all the common information which each device requires. The device class structure is a multi-level class structure to maximise the future proofing capabilities of the ESGP.

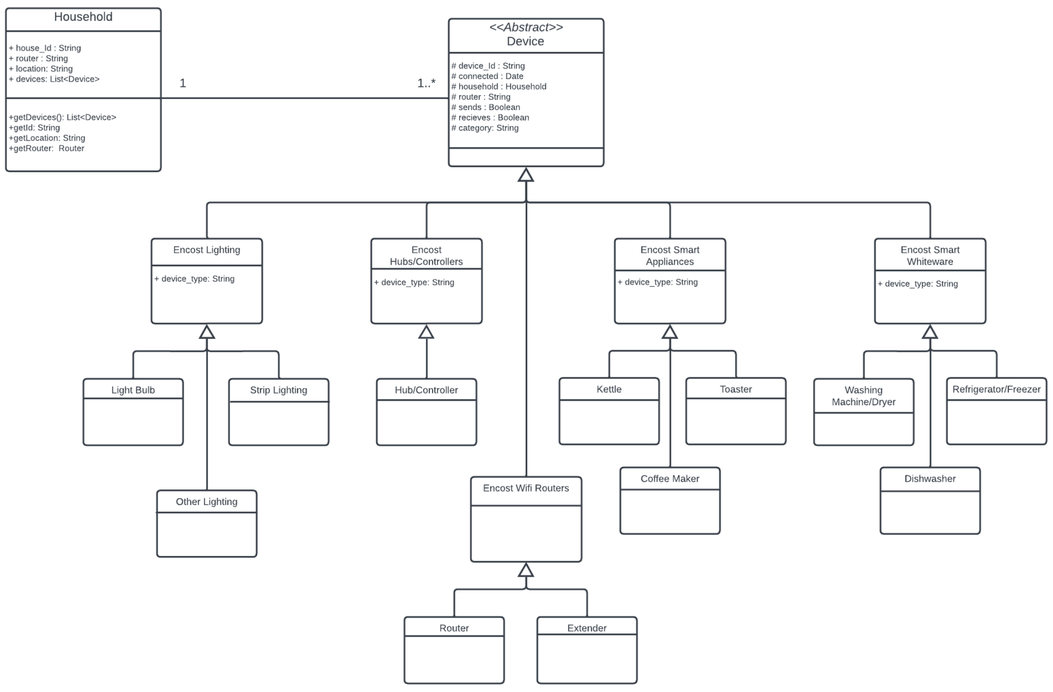


Figure 4.3: The ESGP device class diagram.

Attributes

- Device_id

  The devices unique Java string id. Example: 'EWR-1234'.

- Connected

  The date which the device was connected to the household router. A Java Date object.

- Router

  The households router which the device is connected to. Assumed to only be one per household. Example: 'EWR-1234'.

- Household

  The id of the household which the device is in. Stored as a Java string. Example: 'WKO-1234'.

- Sends

  A boolean object of whether the device can send information out to a router or other Encost device.

- Receives

  A boolean object of whether the device can receive information out to a router or other Encost device.

- Category

  Which device category the device is stored in. These categories are the Encost device categories as specified in the requirements documentation. They are listed below as sub classes of the Device class in section 4.1.3.

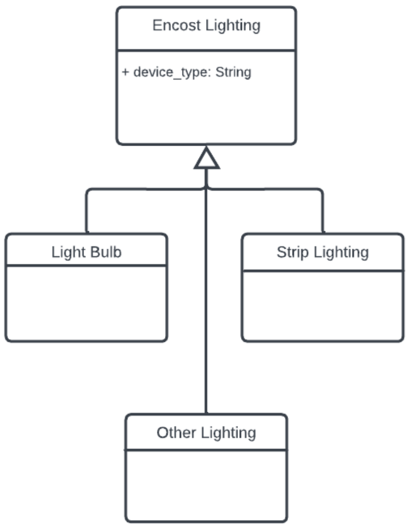
Each device category has its own tree of sub-classes called types listed as bullet points below. This is a formatting decision to display the sub-classes of each category. Currently the device types are empty classes. However, Encost has communicated that they wish to be able to expand individual information of each type or category to hold information which is not relevant to other types. By designing with this in mind the ESGP will be future proofed for Encosts' growth. This is shown in the device class diagram in figure 4.3.

Sub-Class: Encost Lighting
Attribute: Type
Which device type the object is from the Encost lighting class. Currently the list of lighting types available includes:

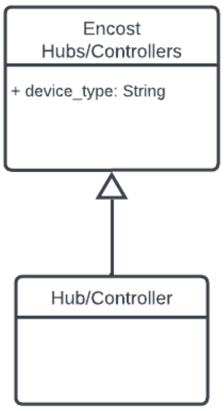
- Light Bulb
- Strip Lighting
- Other Lighting

Sub-Class: Encost Hubs/Controllers
Attribute: Type
Which device type the object is from the Encost hubs/controllers class. Currently the list of hub types available includes:

- Hubs/Controllers

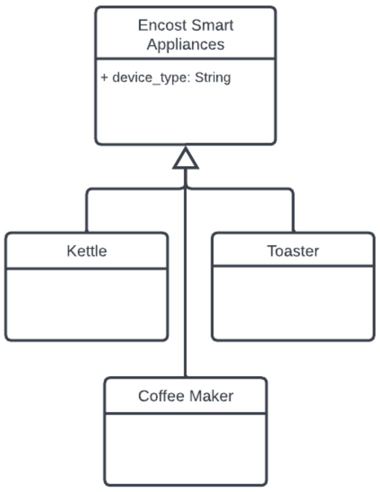


(a) The ESGP lighting class diagram.

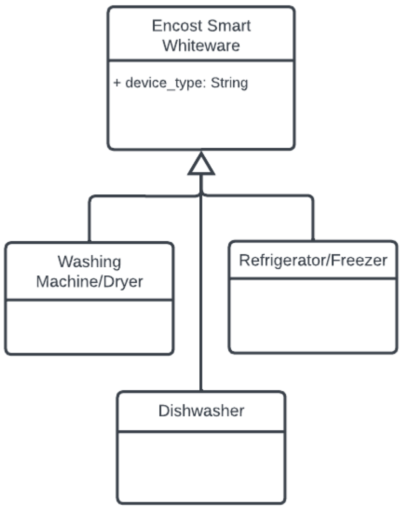


(b) The ESGP hubs & controllers class diagram.

Figure 4.4: The ESGP device class diagram zoomed in.



(a) The ESGP smart appliances class diagram.



(b) The ESGP smart whiteware class diagram.

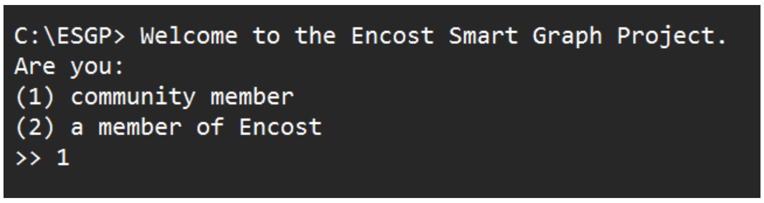
Figure 4.5: The ESGP device class diagram zoomed in.

Sub-Class: Encost Smart Appliances
Attribute: Type
Which device type the object is from the Encost smart appliances class. Currently the list of smart appliances types available includes:

- Kettle
- Toaster
- Coffee Maker

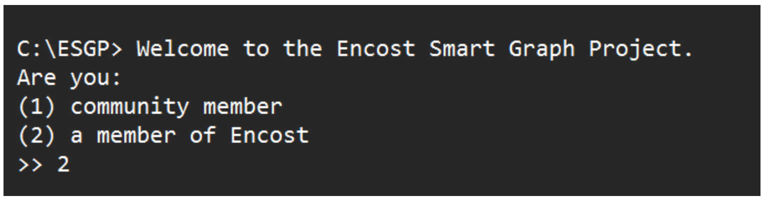
Sub-Class: Encost Smart White-ware
Attribute: Type
Which device type the object is from the Encost smart white-ware class. Currently the list of smart white-ware types available includes:

- Washing machine/Dryer
- Refrigerator/Freezer
- Dishwasher



```
C:\ESGP> Welcome to the Encost Smart Graph Project.
Are you:
(1) community member
(2) a member of Encost
>> 1
```

(a) The ESGP community user.



```
C:\ESGP> Welcome to the Encost Smart Graph Project.
Are you:
(1) community member
(2) a member of Encost
>> 2
```

(b) The ESGP verified user.

Figure 4.6: The ESGP welcome prompts.

4.2 User Interface

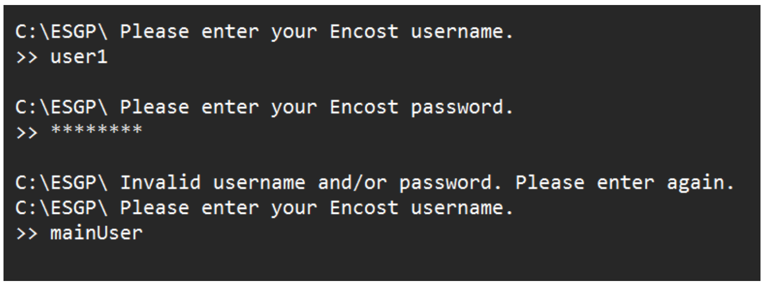
The user interface for the ESGP is entirely based within the command line interface (CLI). This ensures simplicity of the program and fast deployment times as an entire interface does not have to start up for a user to interact with the ESGP.

4.2.1 Welcome Prompt

When the ESGP starts up within the users CLI the welcome prompt is the first message which is shown. As shown in figure 4.6 the user is welcomed to the program and asked which type of user they are. The prompt when asking which type of user specified 1, or 2 to represent community members and Encost members respectively. Due to the possible input which users may enter other acceptable inputs include (c, community, e, encost), all with lower-case conversion implemented.

4.2.2 ESGP Account Login

If the user selects to login to an encost verified account then the CLI with display prompt 2, ESGP account login. The user will be prompted to enter their encost verified username first, then their password. The password will be converted to '*' as output to the CLI to follow standard password entering conventions and for security reasons. The entered detailed will then be checked by the ESGP backend to see if both the login details are valid. The encryption process of the passwords are outlined in section 5.1. If the user is not successful with login the result will be displayed by the CLI as an appropriate error message and the user will be prompted again as shown in 4.7. If the login is successful the ESGP backend must change the user type to 'encost-verified' and move onto the next prompt.

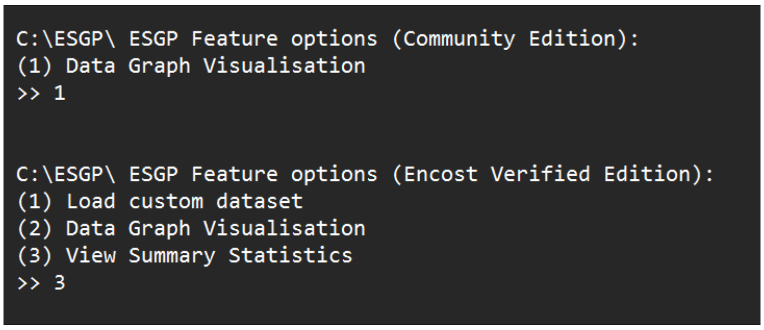


```
C:\ESGP\ Please enter your Encost username.
>> user1

C:\ESGP\ Please enter your Encost password.
>> ********

C:\ESGP\ Invalid username and/or password. Please enter again.
C:\ESGP\ Please enter your Encost username.
>> mainUser
```

Figure 4.7: The ESGP login process.

4.2.3 ESGP Feature Options

For either the community user or Encost verified user the feature options prompt acts as a main menu for the ESGP. The user type (community or encost-verified) must be displayed with the main prompt of the feature option to clearly communicate which edition the user is on. This is as the features available to the community user is different to the verified as shown in 4.8. Again due to the multiple ways a user may attempt to select a feature the ESGP must be able to recognise multiple user inputs. The number which labels each feature is a valid input as shown in 4.8 but also the sub-strings from within each line. The specific labeling for the features is chosen so that for each possible feature there are no overlapping sub-strings. A user may enter ’custom dataset’ or ’view summary’ and the ESGP must recognise these are options 1 and 3 respectively.

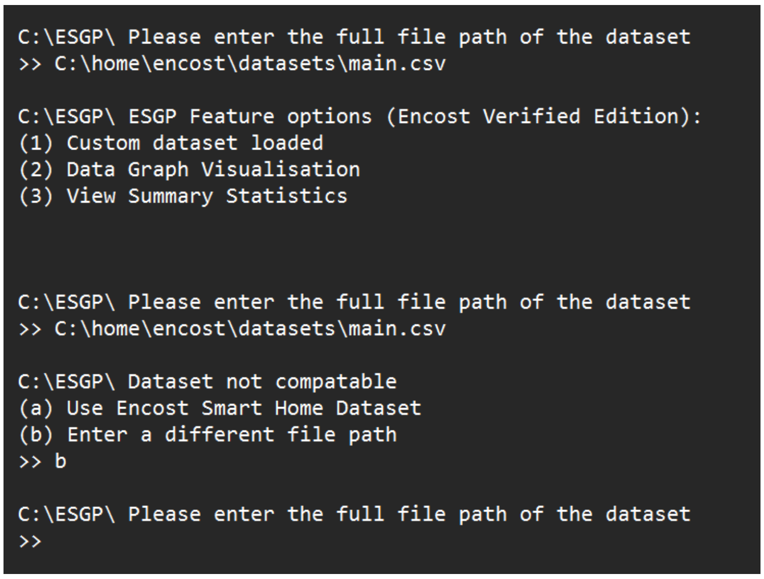


```
C:\ESGP\ ESGP Feature options (Community Edition):
(1) Data Graph Visualisation
>> 1


C:\ESGP\ ESGP Feature options (Encost Verified Edition):
(1) Load custom dataset
(2) Data Graph Visualisation
(3) View Summary Statistics
>> 3
```

Figure 4.8: The ESGP feature options.

4.2.4 Load Custom Dataset

The load custom dataset prompt must ask for a file path to the dataset. The ESGP must verify that the file path leads to a valid dataset. If it is a valid dataset then the ESGP will load the dataset and the CLI will output feature options again except this time option 1 now displays ’Custom dataset loaded’ to indicate success as shown in 4.9. If the dataset is not found or cannot be loaded by ESGP the user will be asked to either enter a new file path or use the default Encost Smart Home dataset.

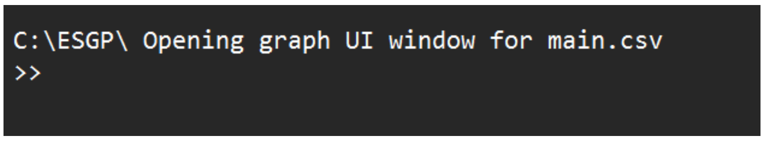


```
C:\ESGP\ Please enter the full file path of the dataset
>> C:\home\encost\datasets\main.csv

C:\ESGP\ ESGP Feature options (Encost Verified Edition):
(1) Custom dataset loaded
(2) Data Graph Visualisation
(3) View Summary Statistics



C:\ESGP\ Please enter the full file path of the dataset
>> C:\home\encost\datasets\main.csv

C:\ESGP\ Dataset not compatable
(a) Use Encost Smart Home Dataset
(b) Enter a different file path
>> b

C:\ESGP\ Please enter the full file path of the dataset
>>
```

Figure 4.9: The ESGP load custom dataset process.

4.2.5 View Graph Visualisation

If the user is to select view graph visualisation then the CLI output indicates that a new graph UI window is opening as shown in 4.10. The ESGP will open a UI window displaying a GraphStream object of the dataset.



```
C:\ESGP\ Opening graph UI window for main.csv
>>
```

Figure 4.10: The ESGP open graph visualisation.

The graphStream UI window will hold the graph of the devices which are found within the database. All devices within a single household may be connected to the one household which is associated with that house. However, some devices may not be connect to a router or directly to a router (possible to a hub or other device first). In order to increase readability the households must be clustered. Meaning that all devices from a single household must be grouped together and separate from another households devices.

The arrows used to connect the devices to the router need to use the graph stream arrow on the default shape and size. The direction of the arrow will indicate the flow of information to or from a device. In figure 4.11 an example graph is showing four households with different devices connected to one another and arrows which indication information flows. This will keep to standard connections on UI and ensure that the users are able to easily understand the diagram. This is important as the graphing visuals function of the ESGP is available to community user as well as the Encost verified users. Because the users which have access to this portion of the program are expected to have a varying level of understanding of technical knowledge or of the Encost products capabilities by following common conventions within the diagram the program will be simpler for all users to understand.

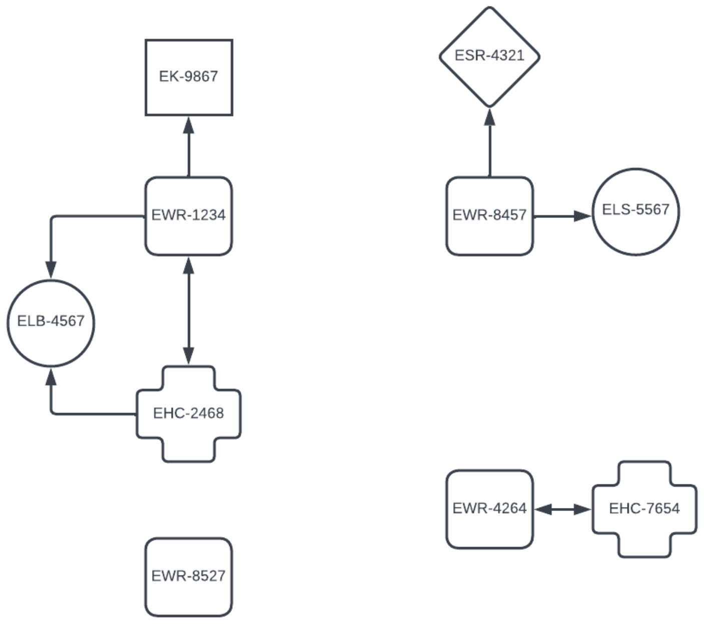


Figure 4.11: Example of Graph Stream Visualisation

A node within the graph stream object can be a cross, circle, box or diamond. As there are four separate device types then one should be assigned to each node style. A rounded square will be used to identify the Encost Routers specifically. The specific matching of device type and node shape can be shown in table 4.1. This will increase the clarity for users to be able to identify each object within the graph. The device categories will be identified by the device ID. For example a an Encost Kettle ID will always start with 'EK' then the identifying digits as per the Encost naming conventions. Because a kettle is a smart appliance it will be a square labeled 'EK-****'. This enables a user to identify the specific device, type and its category from the graph. In figure 4.11 a full representation of the device ID labels are shown in addition to the device shape representations, arrows for data flows and separation of household within a single dataset.

4.2.6 Summary Statistics

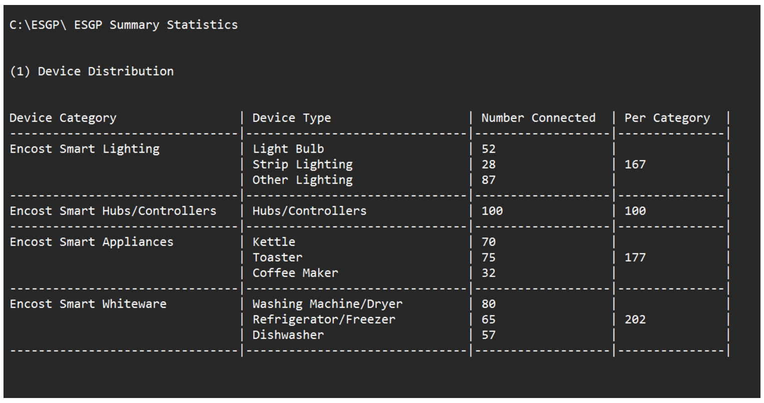
Viewing the summary statistics is the largest single output to the CLI which the ESGP will complete. After completing the calculations on device distribution, device location

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

Table 4.1: Device categories and node shapes.

and device connectivity as outlined in the requirements documentation tabular forms of this information will be displayed to the user. As shown in figure 4.12 all the table summaries are created with '-' as horizontal dividers and '|' as vertical dividers.

Device distribution calculates the number of connected devices per device type and per category. These numbers should be calculated by the backend of the ESGP and the GraphStream object. An example output is shown in figure 4.12.



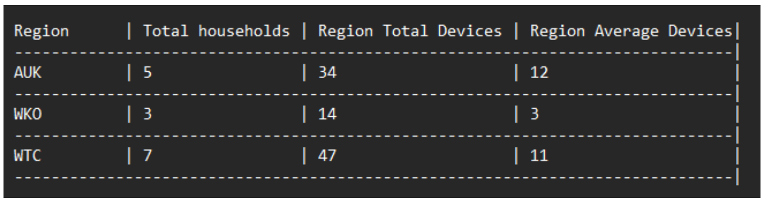
```
C:\ESGP\ ESGP Summary Statistics


(1) Device Distribution


Device Category                 | Device Type                  | Number Connected  | Per Category  |
--------------------------------|------------------------------|-------------------|---------------|
Encost Smart Lighting           | Light Bulb                   | 52                |               |
                                | Strip Lighting               | 28                | 167           |
                                | Other Lighting               | 87                |               |
--------------------------------|------------------------------|-------------------|---------------|
Encost Smart Hubs/Controllers   | Hubs/Controllers             | 100               | 100           |
--------------------------------|------------------------------|-------------------|---------------|
Encost Smart Appliances         | Kettle                       | 70                |               |
                                | Toaster                      | 75                | 177           |
                                | Coffee Maker                 | 32                |               |
--------------------------------|------------------------------|-------------------|---------------|
Encost Smart Whiteware          | Washing Machine/Dryer        | 80                |               |
                                | Refrigerator/Freezer         | 65                | 202           |
                                | Dishwasher                   | 57                |               |
--------------------------------|------------------------------|-------------------|---------------|
```

Figure 4.12: The ESGP device distribution summary.

The device location tables must show a variety of different statistics as shown in the 3 tables in figure 4.13. First, using all region locations as outlined in ISO 3166-2 standard published by the ISO 3166 Maintenance Agency (ISO 3166/MA) all devices in that region must be calculated. Along with listing the total households per region and average number of devices for that region. An example of this is shown in figure 4.13 (a). This example is a smaller than the output which should be developed as there are 17 different listed regions for the ISO 3166 regions.

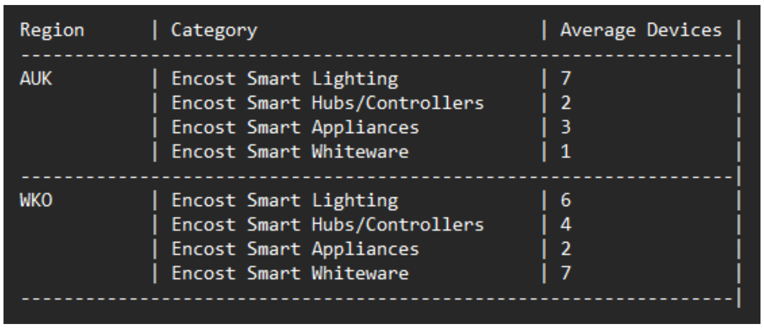
The second table for the device location summary statistics is to then display the average number of devices per category, per household. This table again should be ordered by region to allow it to be easier reading for the user as shown in figure 4.13 (b). This again will be the same tabular format for the 3rd location statistic which shows the number of devices by category in each region. An example of this is shown in figure 4.13 (c).

The final summary statistics tables gives the statistics of device connections. Each



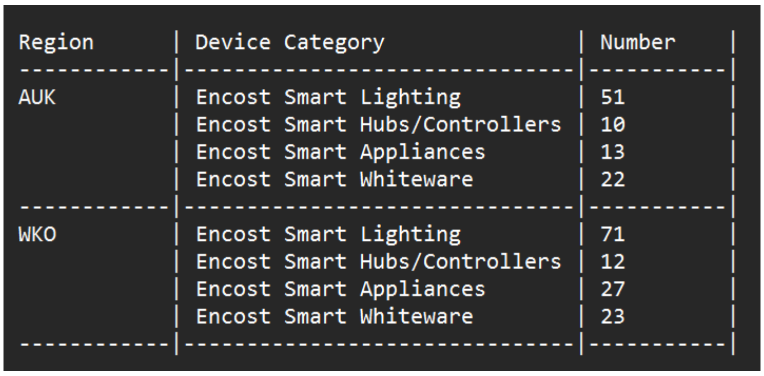
```
Region      | Total households | Region Total Devices | Region Average Devices|
------------------------------------------------------------------------------|
AUK         | 5                | 34                   | 12                    |
------------------------------------------------------------------------------|
WKO         | 3                | 14                   | 3                     |
------------------------------------------------------------------------------|
WTC         | 7                | 47                   | 11                    |
------------------------------------------------------------------------------|
```

(a) Devices per household, by region.



```
Region      | Category                           | Average Devices |
-------------------------------------------------------------------|
AUK         | Encost Smart Lighting              | 7               |
            | Encost Smart Hubs/Controllers      | 2               |
            | Encost Smart Appliances            | 3               |
            | Encost Smart Whiteware             | 1               |
-------------------------------------------------------------------|
WKO         | Encost Smart Lighting              | 6               |
            | Encost Smart Hubs/Controllers      | 4               |
            | Encost Smart Appliances            | 2               |
            | Encost Smart Whiteware             | 7               |
-------------------------------------------------------------------|
```

(b) Devices per household, by category.



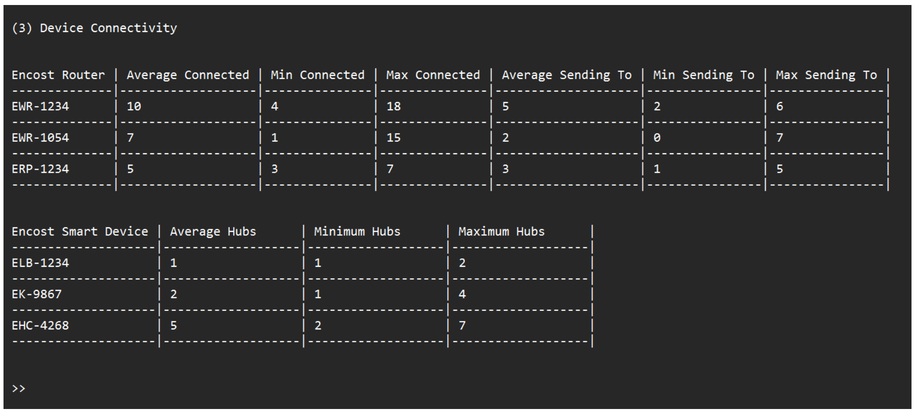
```
Region      | Device Category               | Number     |
------------|-------------------------------|------------|
AUK         | Encost Smart Lighting         | 51         |
            | Encost Smart Hubs/Controllers | 10         |
            | Encost Smart Appliances       | 13         |
            | Encost Smart Whiteware        | 22         |
------------|-------------------------------|------------|
WKO         | Encost Smart Lighting         | 71         |
            | Encost Smart Hubs/Controllers | 12         |
            | Encost Smart Appliances       | 27         |
            | Encost Smart Whiteware        | 23         |
------------|-------------------------------|------------|
```

(c) Devices per region, by category.

Figure 4.13: The ESGP device location summaries.

Encost router found within the data set will be listed alongside its average, minimum and maximum connections to other devices. It will also display the average, minimum and maximum devices the router is sending information to. This information can be taken from the GraphStream object and displayed in a format like shown in figure 4.14.

The second table shows the average, minimum and maximum number of Encost hubs/-controllers which a device is connected to. Each device in the dataset must be connected to at least 1 of the Encost hubs as there must be one router per household the statistics were taken from. This assumption is outlined in section 4.1.3 Attribute: Router.



```
(3) Device Connectivity


Encost Router | Average Connected | Min Connected | Max Connected | Average Sending To | Min Sending To | Max Sending To |
--------------|-------------------|---------------|---------------|--------------------|----------------|----------------|
EWR-1234      | 10                | 4             | 18            | 5                  | 2              | 6              |
--------------|-------------------|---------------|---------------|--------------------|----------------|----------------|
EWR-1054      | 7                 | 1             | 15            | 2                  | 0              | 7              |
--------------|-------------------|---------------|---------------|--------------------|----------------|----------------|
ERP-1234      | 5                 | 3             | 7             | 3                  | 1              | 5              |
--------------|-------------------|---------------|---------------|--------------------|----------------|----------------|


Encost Smart Device | Average Hubs      | Minimum Hubs      | Maximum Hubs      |
--------------------|-------------------|-------------------|-------------------|
ELB-1234            | 1                 | 1                 | 2                 |
--------------------|-------------------|-------------------|-------------------|
EK-9867             | 2                 | 1                 | 4                 |
--------------------|-------------------|-------------------|-------------------|
EHC-4268            | 5                 | 2                 | 7                 |
--------------------|-------------------|-------------------|-------------------|


>>
```

Figure 4.14: The ESGP device connectivity summary.

5 Other Design Specifications

5.1 Security

As per the requirement specification document each user password must be encrypted and decrypted as part of the user verification process. It is also a requirement that the ESGP be developed using Java. Due to these two requirements the encryption process for passwords will be using the Java secure hashing techniques from the 'java.security' library package. Specifically the MD5 hashing technique. This will allow the encryption and decryption of the passwords securely and allow easy and fast verification for user login.

5.2 Error Checking & Handling

5.2.1 User Errors

Users only have access to the CLI to interact with the ESGP. This means that in terms of error handling for user error only incorrect CLI commands need to be handled. This means that when users are choosing specific features or loading a custom dataset the ESGP needs to ensure these input are valid before using them. The user will be given multiple valid inputs for a response to the CLI prompts. As outlined in section 4.2.3 this may include sub-strings of the feature, the number of the feature, or 'c/community' for the user identification. This will decrease the likelihood of an invalid input passing to the application framework. If any invalid inputs are found the CLI must inform the user that their input is valid and re-prompt the user.

If a user is to incorrectly enter a prompt into the CLI, for example a community member selects to login to an Encost verified account when they weren't meant to. To implement a simple error handling point for this type of user error, at any point within the ESGP if a user enters 'back' the CLI will return to the previous prompt. Otherwise the user can type 'home' to return to the feature options page of the respective edition (community or encost-verified) they are using.

5.2.2 Dataset loading errors

Simple error handling needs to be used for when the application framework loads in a dataset. This is simply to ensure that the dataset is in the correct format, like used in the Encost smart homes dataset. If the dataset chosen by the user is not in this format or if the application framework cannot load the dataset an error message needs to be displayed to the user. This is to inform them of the error. Then the CLI must allow the user to choose a new dataset or use to default dataset. A visual example of this error handling can be seen in section 4.2.4, figure 4.9.

6 Conclusion

The Encost Smart Graph Project is to be designed, implemented, tested and maintained by Softflux as a command line interface program. This document is a software design specification which covered the purpose, product functions, required development formats, design details, user interface and security specifications. This document also outlined the object orientated design method to be used and how the classes are to interact and connect with one another within the application framework. This is all in order to inform and outline the implementation process. This information is also to be used as a testing comparison after the ESGP has been developed. This document follows the requirements outlined in the software requirements specification for the Encost Smart Graph project document.