SOFTWARE DESIGN SPECIFICATION

for

Encost Smart Graph Project

Version 1.0

Prepared by: Ben Lee
SoftFlux Engineer

SoftFlux

June 11, 2023

Contents

Revision History

Name	Date	Reason for Changes	Version

1 Introduction/Purpose

1.1 Overview

The Encost Smart Graph Project (ESGP) hopes to improve connectivity and enable the visualisation of Encost devices. Encost has gathered information on their devices through energy companies and their users for 2 years. This information will provide the data necessary to visualise the Encost devices using a graph data structure.

1.2 Purpose

This Software Design Specification document's purpose is to act as an outline for developers and testers to follow and compare, it will ensure the requirements set in the Software Requirements Specification document are met and design choices are made with reason. To do this, a component diagram, process diagram and use case diagram will be leveraged to demonstrate how the components interact with each other as well as how users will navigate the application. It is important that every choice made has been given a clear and concise explanation, this will leave no room for ambiguities.

2 Specialized Requirements Specification

2.1 Functional Requirements

2.1.1 User Welcome

On the application start, the user will be greeted with a welcome on the console and prompted to input whether the user is a member of the community or a member of Encost, by selecting a corresponding input. The system should then store the user type, either "encost-unverified" or "encost-verified". Depending on the user type, the next prompt will be provided. For Community users, the prompt will be the ESGP Feature Options. For Encost members, they will be prompted with an account login.

2.1.2 Account Login

The user will be prompted to enter a Username on the console, and then be prompted to enter a password. The system will compare the inputs against the stored username and password pairs, this includes encrypting the password to compare against stored encrypted passwords. If the system doesn't find a match to the inputs, the user will be informed via the console that this information is incorrect and will be prompted to reenter the username and password. However, if the inputs are correct, the system will update the user type to "encost-verified" and provide the user with the ESGP Feature Options.

2.1.3 ESGP Feature Options

The user will be prompted via the console with a selection of features to pick from. Community Users can only choose to visualise the graph representation of the data. Encost Users have three features, loading a custom data set, visualising a graph representation of data, and viewing summary statistics, each corresponding to an input (ie. 1, 2 or 3). Depending on what input the user responds with, the system will prompt them for that corresponding feature.

2.1.4 Data-Set Requirements

The Encost Smart Homes Data-set file should be located in the system, the system should know this location. The system will then read line by line and extract relevant device information. Another option for "encost-verified" users is to load custom data sets. The system will prompt the user to enter the full file path. If the custom data set has an invalid format, the system will inform the user via the console that the data set is incompatible, and to either use the default data set or try another path. Otherwise, this file path should now be the file path used when the system visualises the graph representation of the data. The data set will be stored in the system as a .txt file, the first line of the data set will be the column titles separated by commas, every subsequent

line will be the device information separated by columns.
Here is an example:
Device ID,Date Connected,Device name,Device type,Household ID,Router Connection,Sends,Receives
EWR-1234,01/04/2022,Encost Router 360,Router,WKO-1234,-,Yes,Yes

2.1.5 Visualisation Requirements

When the user selects “visualising a graph representation of the data” from the ESGP Feature Options, the system must use the GraphStream library to output a graph on a UI window. The graph must show all connections between nodes(device objects) using edges(connections between objects). Nodes must be categorised into different devices, meaning one category’s nodes should be distinguishable from another category’s nodes. The graph should also illustrate the ability of the device to send and receive commands from other devices. All households should be represented in the graph.

2.1.6 Calculation Requirements

The system should use the information stored in the current data set to calculate any information needed in representing any summary statistics, see the System Requirements Document for any clarification on calculations. The system should output these figures to the console in a clear and concise manner.

2.2 Nonfunctional Requirements

2.2.1 Performance Requirements

Feedback on response times should always be given to the user if any process takes longer than 1 second. The Encost Smart Homes Data set, calculating device distribution, location and connectivity should all take no longer than 10 seconds to load and process. The graph visualisation, no more than 5 seconds to display.

2.2.2 Security Requirements

User Authentication is necessary for any Encost employees, Ten usernames and password pairs will be provided, and all passwords should be encrypted and stored in the application. Any password entered by an attempted login should be encrypted and compared against the stored encrypted passwords. The Encost Smart Home Data-set must be anonymised before use, this same restriction applies to any custom data sets used in the system. UTF8 should be used to encode any communication.

2.2.3 Compatibility Requirements

The software will be a local console application available on Windows 10 and must be developed through Java(version 1.8.0 or higher) with an object-oriented solution in mind. ESGP will integrate with GraphStream(version 1.3), GraphStream is a Java library for building and visualising dynamic graphs. It will be used to build and visualise a graph of Smart Home products and their connection and communication.

2.2.4 Usability and Reliability Requirements

The textual layout must be clear and readable. Language must be concise and easy to understand. Users must be able to follow any prompt or error easily. Data loss must be prevented, as well as backups of the source code should be made daily.

3 Software Architecture

3.1 Component Architecture

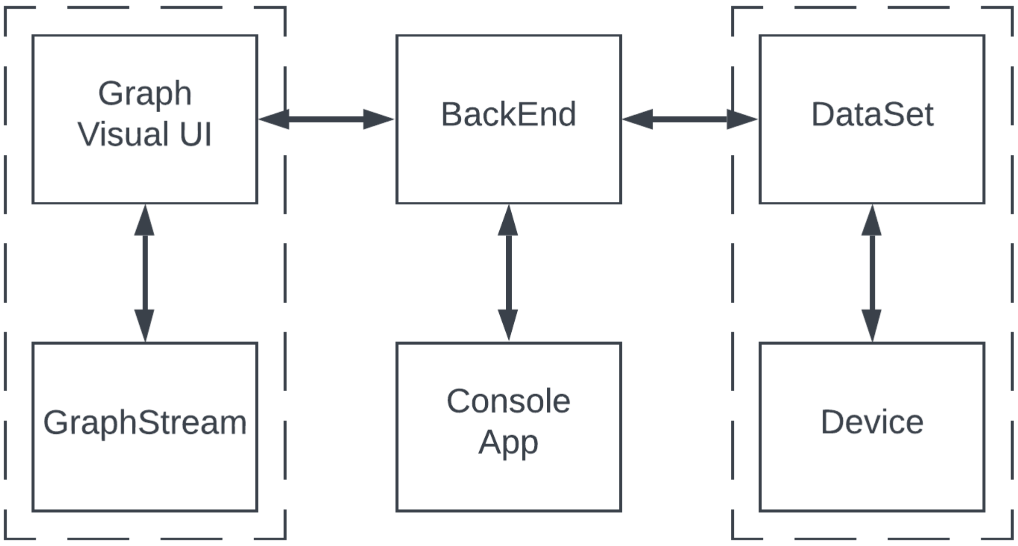

Figure 3.1: Component Diagram

Figure 3.1 shows that all components in the System rely on the back-end to send, receive and process information.
The console application sends and receives information from the back-end, and the back-end is responsible for passing information across the components. The back-end will process inputs from the user across the console application, the first exchange will be

sending a welcome message and user type option, depending on what the user inputs, the back-end will send the next step, either being the Encost member login or the ESGP Feature Options for community members. During the Encost member login, the back-end is responsible for checking the given username and password against the stored username and password pairs, the password will also need to be encrypted as the stored passwords are all encrypted. When the back-end displays the ESGP Feature Options, it will have the responsibility of going to the correct option when selected. It would then have to communicate with the data set component, to either set a new file or use the default data set and then translate the data set into the appropriate objects with necessary information. If the option was to visualise a graph, it would then send all data set information to the GraphStream Java library, and a UI window will appear allowing the user to view the graph. If the option was to see a statistics summary, the back-end will get the data set information, and calculate this information into device distribution, device location and device connectivity statistics.
The reason for this diagram is to show the importance of the back-end component as it impacts the rest of the components. The back-end must be efficient and know where information is coming from, where to send it and what calculations need to be done.

3.2 Process Diagram

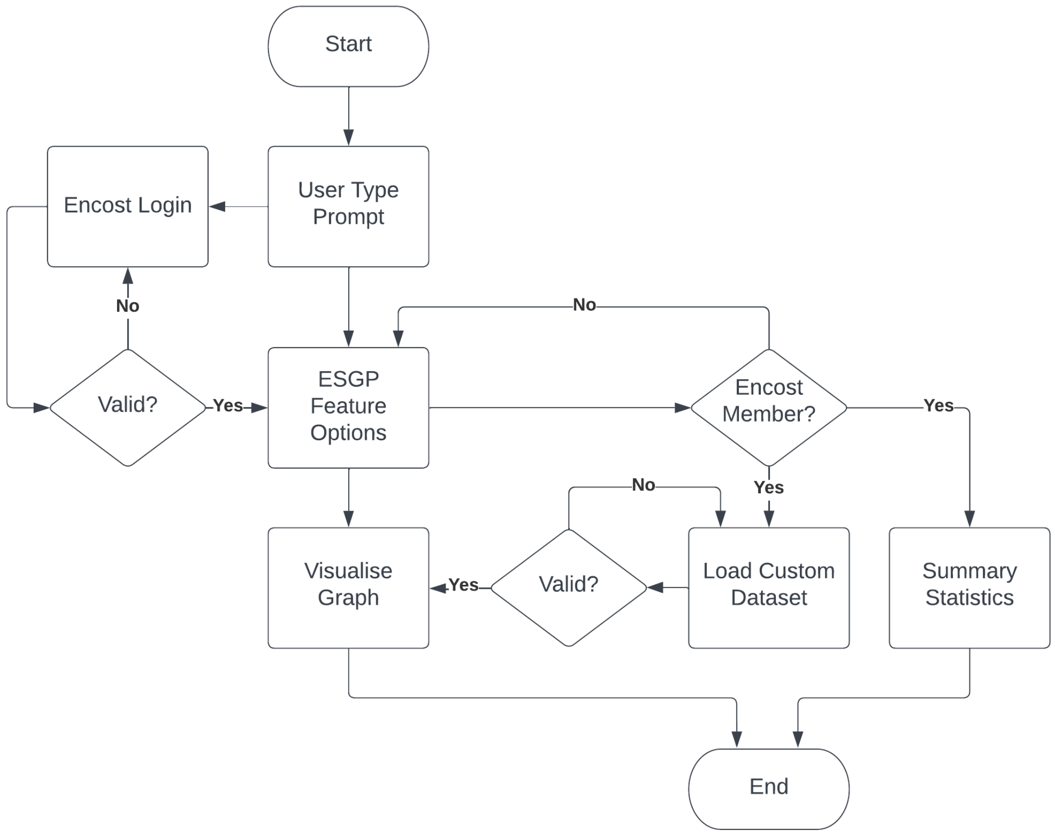

Figure 3.2: Process Diagram

Figure 3.2 shows how the process of the system will use checks to allow the user to progress through.
When the user starts the application they will be prompted to select their user type. If the user indicates they are an Encost member they will need to input a username or password, if this is incorrect they will be asked again to input their username and password. If this is correct or the user is a community member, the ESGP Feature Options will be displayed, Encost members will be able to load a custom data set and get a device statistics summary. the custom data set will be used to visualise the graph, Encost and community member will both be able to visualise the data set in a graph representation.
This diagram visualises how each function is necessary for the system to progress the user through to the end of the process. Each function is controlled by the back-end, whether that be the information calculated and displayed or the Options presented to each user type, for example.

3.3 Use Case Diagram

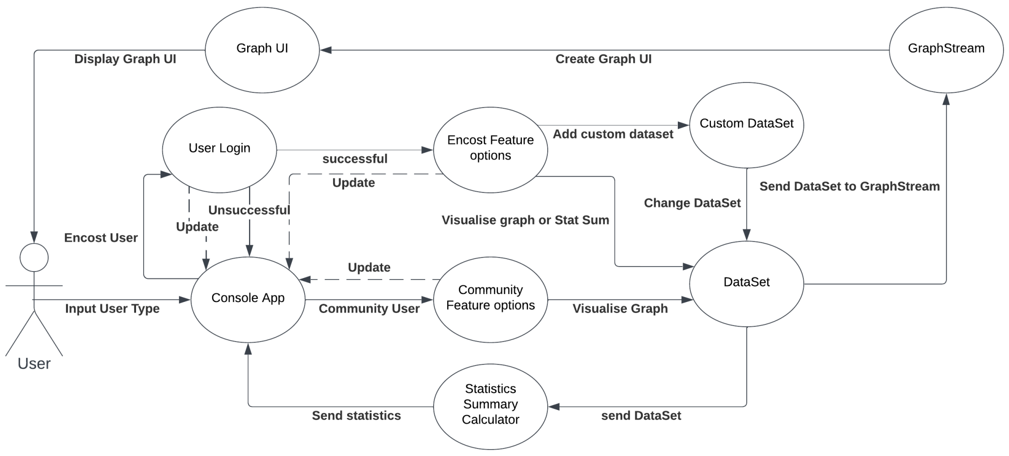

Figure 3.3: Use Case Diagram

Figure 3.3 shows the scope of the system and the interaction between the system and it's users.
When a user starts with the Console App they will be given prompts for where they would like to go. The first prompt will be for the user to indicate what user type they are, this will update the console with one of the following, a Encost login prompt or the Encost Feature Options, it will also save the which user-type in the system. A community user will be sent straight to the options prompt, but only be given the option to visualise the data set. Whereas an Encost member will be given the options, visualise the data set, use a custom data set and get Statistics summary. If the option to use a custom data set is chosen then the pathway to the data set will be changed. Much like in Figure 3.1 when Visualise the data set is chosen the GraphStream Java library is used to create a graph UI window that is displayed to the user. The last option will do calculation on the data set and display the results on the Console.
This graph depicts how the back-end will allow Users to see information over the console application and how they will be shown the Graph via a UI window. It's important we see how a user would use the system as it allows us to create restraints and limits where the user can go and what they can see. The user should not be able to see what happens in the back-end, they will only be shown what the system updates the console with.

4 Component Design

4.1 Detailed Design

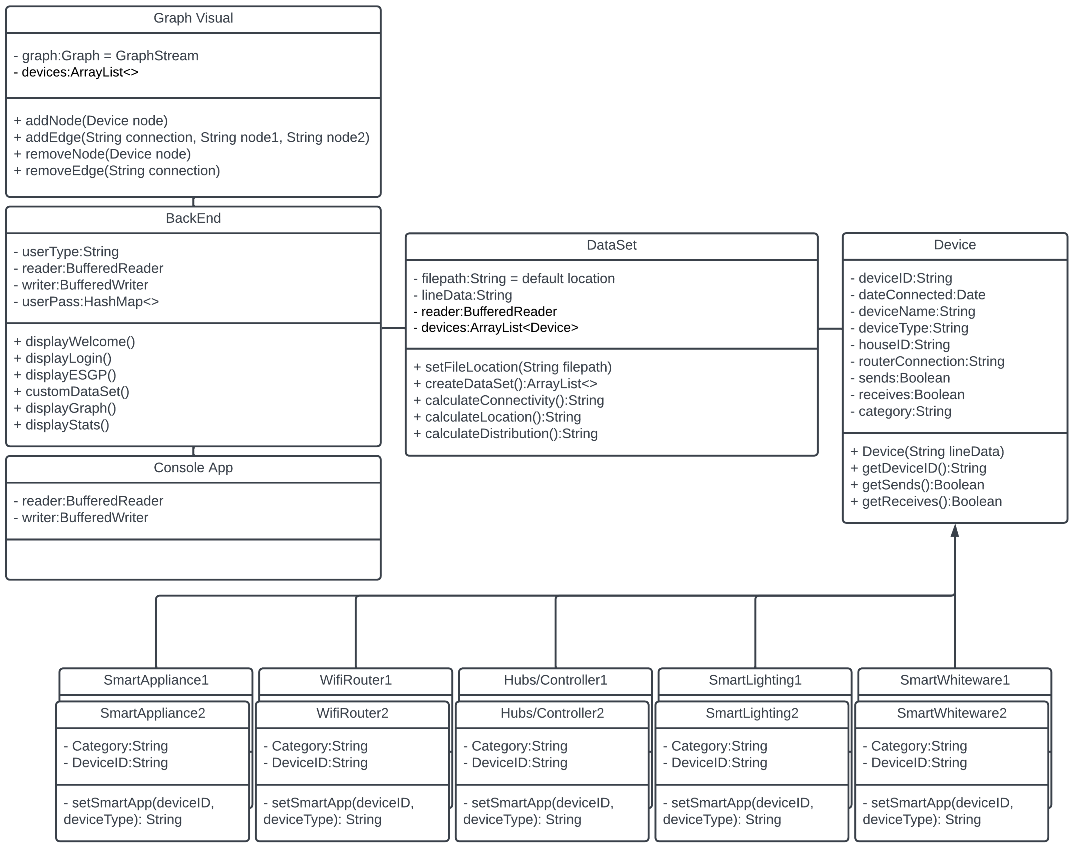

Figure 4.1: Class Diagram

4.1.1 Console Application

Description

The console application will act as an entry point for the program. Console application will be used to display information for interactions from the user for the back-end, the interactions will inform the back-end what it's next actions will be.

Attributes

reader

Creates a BufferedReader to read from standard input from the back-end.

writer
Creates a BufferedWriter to write to standard output to the back-end.

Implementation
The console application will call methods from the back-end, this will inform the back-end to send text across the standard i/o, the user will respond by writing to the standard i/o and the corresponding response will determine what the next action is. The back-end is responsible for knowing which action to perform, see figure 3.3 to clarify what actions should be performed accordingly. This is to ensure all responses come from a single place, so information is clear and the system is consistent.

4.1.2 back-end

Description
The back-end is responsible for communicating any given responses from the user to the rest of the system. It is responsible for creating the getting the data set file and creating the Device objects. It will use functions from other classes to generate any needed information.

Attributes
userType
Store the user type selected by user.

graph
Creates a graph instance from the GraphStream.

devices
Store an array list of the devices from the data-set.

userPass
This hash map will hold all the stored username and encrypted password pairs to compare against the user login.

Functions
displayWelcome()
A public method that displays a welcome message to the user and prompts the user-type to be selected.

displayLogin()
A public method that displays the Encost login prompt if the user selected the Encost user type, this login prompt will ask for a password and username, the password will be encrypted and compared against stored username and password pairs.

displayESGP()

A public method that displays the ESGP Feature Options for the user, the options will be different depending on user type, see figure 3.2 for options available for both user types.

customDataSet()
A public method that displays a prompt for the user to enter a new file path for a custom data-set, this will now be the new file destination for the data-set component to use, by calling the setFilelocation function in DataSet. Figure 3.2 illustrates that the file must have a valid format.

displayGraph()
A public method that displays a UI window for the user to view the graph via the Graph Visualisation component, it will use it's functions to create Nodes and edges from the devices array list.

displayStats()
A public method that displays a summary statistics to the standard i/o for the user, it will perform the necessary calculations on the devices array list, to get the statistics for display.

Implementation
The back-end will be designed as a singleton, this is to make sure there is only ever a single instance of the class. The console application will use the back-end functions to get prompts, depending on the responses that come from the user, the back-end must act accordingly, it will use functions from other classes to create the data-set, calculate summary statistics, get a list of device objects from data-set class, and use the functions from Graph Visualisation to create and display the graph to the user. This will ensure all information is passed and checked in this component, this will ensure consistency and prevent loss, invalid or unwanted changes to data.

4.1.3 Data-Set

Description
Data set component is used to create a collection of devices from a given file containing information of devices. It must formatted correctly and have the proper variables. It is also used to set a custom data set and calculate the summary statistics.

Attributes
filepath
This has the default file path location of the Encost Smart Homes Dataset, it can be updated with the setFileLocation function.

lineData
Stores each line of data from the data set file.

reader
Creates a BufferedReader to read from the data set file.

devices
Stores all the created devices from the data set in an array list.

Functions
setFileLocation(String filepath)
A public method that sets the filepath for the data set.

createDataSet()
A public method that creates all the device objects using the device component, then returns them in an array list of all the devices.

calculateConnectivity()
A public method that calculated the connectivity between devices in the data set and returns the result as a string to display on the standard i/o.

calculateLocation()
A public method that calculated the location of devices in the data set and returns the result as a string to display on the standard i/o.

calculateDistribution()
A public method that calculated the distribution between devices in the data set and returns the result as a string to display on the standard i/o.

Implementation
Data set will implement a builder design pattern, this is used because this component will be used to create a data set from a file, this data set will be split into a number of attributes and sent back into an array list as device objects. This ensures that creation of any device is controlled by a director. All functions must be accessible via the back-end. This is important as the back-end will use these Functions to to display information to the user.

4.1.4 Device

Description
Graph Visualisation UI Window is a User Interface that is displayed to the user when they select the Graph Visualisation option from ESGP Feature Options.

Attributes
deviceID
Stores the ID of the device, extracted from the file line data.

dateConnected
Stores the date the device was connected, extracted from the file line data.

deviceName
Stores the name of the device, extracted from the file line data.

deviceType
Store the device type, extracted from the file line data.

houseID
Stores the house the device belongs to, extracted from the file line data.

routerConnection
Stores the router connection of the device, extracted from the file line data.

sends
Stores if the device can send information, extracted from the file line data.

receives
Stores if the device can receive information, extracted from the file line data.

category
Stores the category of the device, extracted the device subclass that the device type belongs to.

Functions
Device(String lineData)
A public method that create a device object using the lineData from the data set file and splitting it into the component variables.

getDeviceID()
A public method that extracts the device id from a device object and returns it as a string.

getSends()
A public method that extracts if a device can send information from a device object and returns it as a boolean.

getReceives()
A public method that extracts if a device can receive information from a device object and returns it as a boolean.

Implementation

The device component will use an abstract factory design pattern, this allows Device to create families of related objects for the categories of the devices using inheritance. A new device object will be created for each line in the data set file, this ensures encapsulation, which allows access control and prevents unwanted changes.

4.1.5 Graph Visualisation UI Window

Description
Graph Visualisation UI Window is a User Interface that is displayed to the user when they select the Graph Visualisation option from ESGP Feature Options.

Attributes

graph
Creates a graph instance from the GraphStream.

devices
Store an array list of the devices from the data-set.

Functions

addNode(String node)
A public method that adds a node to the graph, that represents a device.

addEdge(String connection, String node1, String node2)
A public method that adds a Edge to the graph that represents a connection between two devices.

removeNode(String node)
A public method that removes a given node from the graph

removeEdge(String connection)
A public method that removes an edge from the graph with a given connection.

Implementation
Graph Visualisation UI Window will use the GraphStream library to display the graph with the data-set, using nodes for devices and edges for connections. This component will use singleton pattern, this ensures that there is only one instance of the class, this will help avoid inconsistencies and will allow for global access. This is important because the graph should only be created once.

Dependencies
The Graph Visualisation UI Window will depend on GraphStream to create and delete nodes and connections for all devices, and display the graph to the user.

4.2 Off-the-shelf Technologies

4.2.1 GraphStream

GraphStream is a Java library for building and visualising dynamic graphs. This will be utilised to build the graphs of Encost's Smart Home devices. GraphStream is designed to handle any change that will occur over time ensuring the addition or removal of any devices to and from the data set are handled correctly and effectively. GraphStream's visualisation options are ideal for what information we want to be displayed on the graph, such as each category of device needs to be distinguishable on the graph, as well as the need for edges to display the connection and communication between devices.

4.2.2 Git

Git enables version control, this allows developers to track previous versions of source code and reverse any changes when necessary. Git also enables collaboration for developers, it means developers can work on the same code files simultaneously.

5 Conclusion

In conclusion, the Encost Smart Graph Project (ESGP) is a software system that has been designed carefully to give a visualisation of Encost's devices using a graph data structure.

Throughout this document, were detailed plans for the development of ESGP and the reasoning behind any design decisions. There has been a focus on functionality, reliability, speed and data integrity, and this document should give confidence that we have satisfied our objectives. If followed correctly this document should provide value to Encost and its customers.