SOFTWARE DESIGN SPECIFICATION

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student # 3
SoftFlux Engineer

SoftFlux

April 7, 2023

Contents

Revision History

Name	Date	Reason for Changes	Version

1 Introduction/Purpose

1.1 Purpose

The purpose of this document is to specify the software design, architecture, and components for a software system called Encost Smart Graph Project (ESGP). This document will explain the architecture and components of the application, and what a typical user interaction would involve.

1.2 Document Conventions

This document uses the following conventions:

ESGP: Encost Smart Graph Project

1.3 Intended Audience and Reading Suggestions

This document is intended for anyone who will be developing or maintaining this application, and also the client who could be interested in the specific implementation of the application. This could be developers, testers, project managers, etc. Different readers could use this document as follows:

- Developer: The developer(s) who are developing this application should read this document in order to build the ESGP application as according to specifications so that anyone who is testing or maintaining the application can use this document to understand how the application works. Developer(s) who are maintaining the ESGP application should also read this document in order to better understand how the application works. This is so that they can ensure any changes that are made still fit within the specifications given in this document.

- Tester: Testers should use this document along with the Software Requirements Specification in order to test that the requirements of the application are met, and that they are met in the ways that are specified here.

- Project Manager: The project manager should utilise this document to understand the requirements of the application that they are either developing or maintaining. By reading this document they will be able to discuss the implementation with our clients. They will also be able to ensure that the developers are correctly implementing the agreed upon solution.

- Client: The client can use this document to understand the intended architecture and components that we, SoftFlux, are using to implement the requested solution. They should review this document so that if they have any issues with the specifics of our implementation they can inform us so that the issue can be addressed.

1.4 Project Scope

Encost is a start-up company in the Smart Home space. They manufacture various Smart Home and IoT items such as Wi-Fi routers, Smart Hubs and Controllers, Smart Light Bulbs, Smart Appliances, and Smart Whiteware. As a part of one of their larger objectives, optimising use of and connectivity between devices, SoftFlux has been hired to develop the base version of the ESGP software.

The ESGP software will be used to visualise Encost items and the connections between them utilising a graph data structure. The data set that Encost would like to visualise has been collected with assistance from energy companies and their users. Encost would also like to be able to visualise custom data provided that it is in the same format as the aforementioned data.

This software is intended to be available for both Encost users and members of the community that provided information about their smart devices to Encost. Community users would be able to view a visualisation of the Encost data set, while Encost users could also load custom data and view a summary of the data.

2 Specialized Requirements Specification

2.1 Storage of user information

The client has specified that the usernames and passwords for all users are to be stored locally. Because of this it is extremely important for the passwords to be stored in a way such that users can not freely view them.

The clients originally asked for the passwords to be encrypted, but they have specified that hashing is also an acceptable means of protecting the users' passwords. Therefore we will use hashing on all user passwords before they are made available to be downloaded by the users.

2.2 Anonymity of data

The client has specified that any custom data sets should be anonymised before use with this system. To fulfill this requirement, the identifying household ID should be processed as follows: split the region section and the number section. Leave the region section so that it can be used for calculating summary data, but hash the number portion so that the household ID is not identifiable.

2.3 Response Times

The client has specified that if the following times when running the application should be met. If any of these processes takes more than 1 second, an update should be printed to the user.

- the Encost Smart Homes Dataset should take no more than 10 seconds to load and process
- the graph visualisation should take no more than 5 seconds to display
- calculating device distribution should take no more than 10 seconds to load
- calculating device location should take no more than 10 seconds to load
- calculating device connectivity should take no more than 10 seconds to load

2.4 Source code backups

Backups of the source code should be made daily.

3 Software Architecture

3.1 Component Diagram

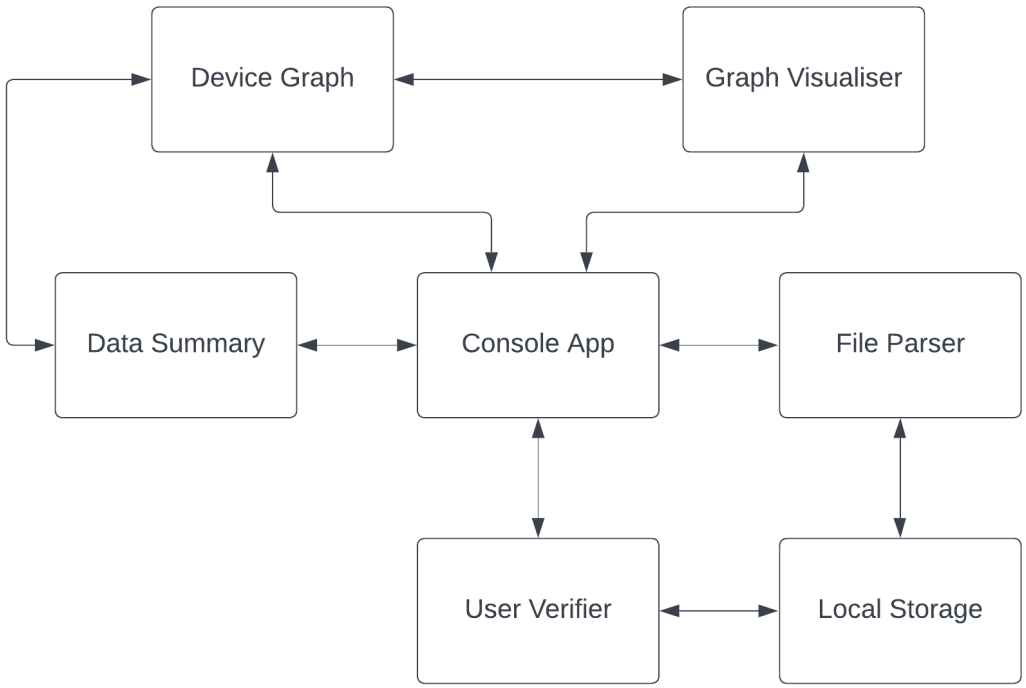

Figure 3.1: ESGP Application Component Diagram

The components in the ESGP application are centered around the console app which the user will interact with. This console app will call various helper components to fulfill tasks required of the application. At present, the tasks required are verifying the user, parsing files from local storage, visualising the graph using the GraphStream library, and calculating and providing a summary of the data. The way these components interact will be covered in more detail in the Components section.

The device graph is not a helper component. It is used to store the devices as nodes and the relationships between them as edges in a graph data structure. This is different from the GraphStream library graph, which is only used for visualisation purposes.

3.2 Process Flow Diagram

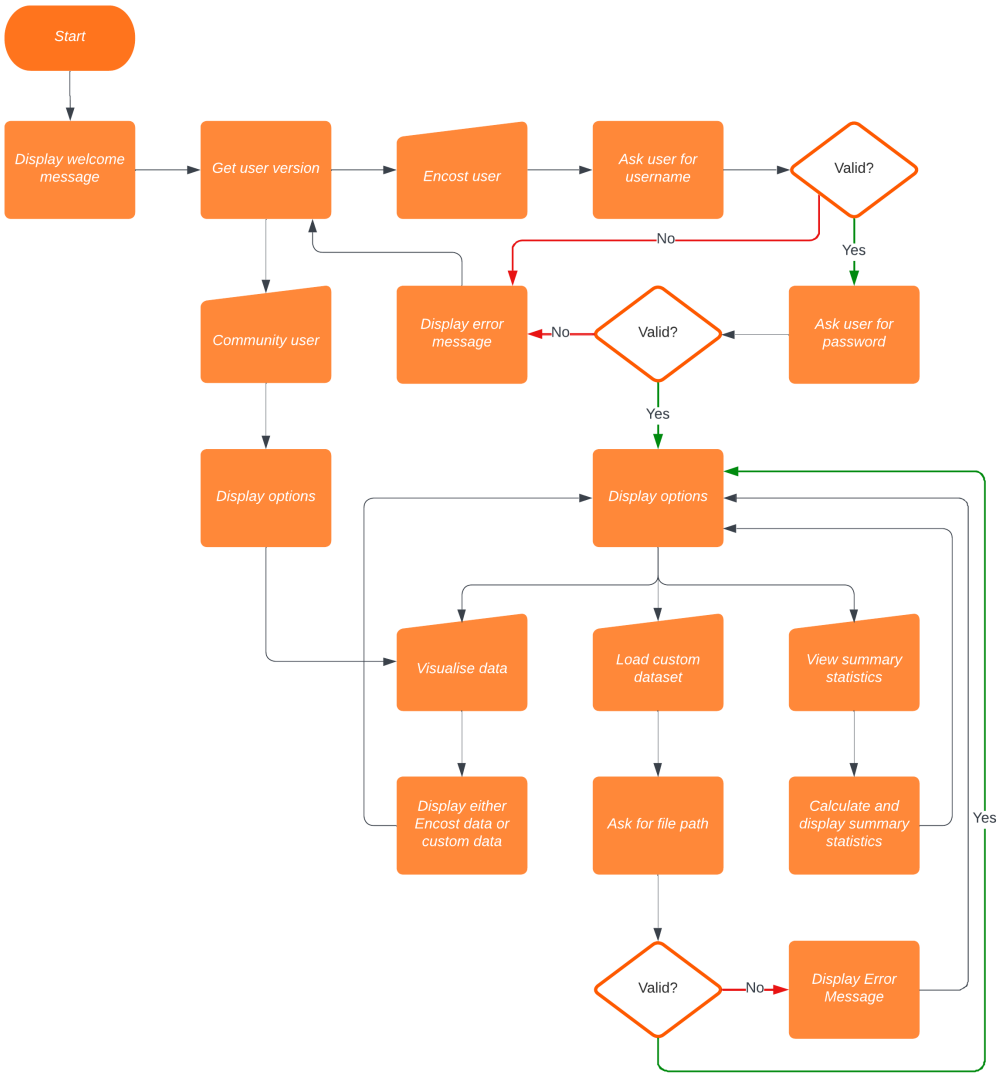

Figure 3.2: ESGP Application Component Diagram

The process flow diagram visualises what a typical user interaction would look like. The clients' have given several requirements in regards to the user interface and how the user should be able to interact with the application.

3.2.1 Getting user version

The first specification was to greet the user, and then to ask the user whether they would like to log in to a Encost user account or continue as a community user. If the user wishes to log in, they are first asked for their username. This is checked against the usernames in local storage. If the username is not valid, an error message will be displayed and the user will be asked again if they would like to log in or continue as a community user. If the username is valid, the same process is repeated for the user's password. The user will be returned to the option where they choose their version upon an invalid entry in order to prevent the user getting trapped. If the user simply made a mistake while inputting their username or password, there would be no problem, but if the user accidentally selected Encost user instead of community user, if they were stuck being asked for their username they would have no way to return to the version selection without restarting the application.

3.2.2 Displaying options

If the user successfully logs in or they choose the community user option they will be displayed the options available to them based on their version. If they are a community user, they will only be able to visualise the ESGP data set. If they are an Encost user, they will be able to visualise the ESGP data set, load a custom data set, or view statistics about the currently loaded data set.

3.2.3 User selects an option

If the user chooses the visualise data option, a new window will be opened to display a visualisation of either the Encost data set or a custom data set if the user has provided one. If the user chooses to load a custom data set, they will be asked for the file path. If it is valid, they will be displayed their options again, but with the load custom data set option indicating the currently supplied data set. If it is not valid, they will be shown an error message and will be displayed their options again. The reasoning for not asking for the file path a second time is the same as for the username and password. If the user chooses to view statistics these will be displayed to the user in the console.

4 Component Design

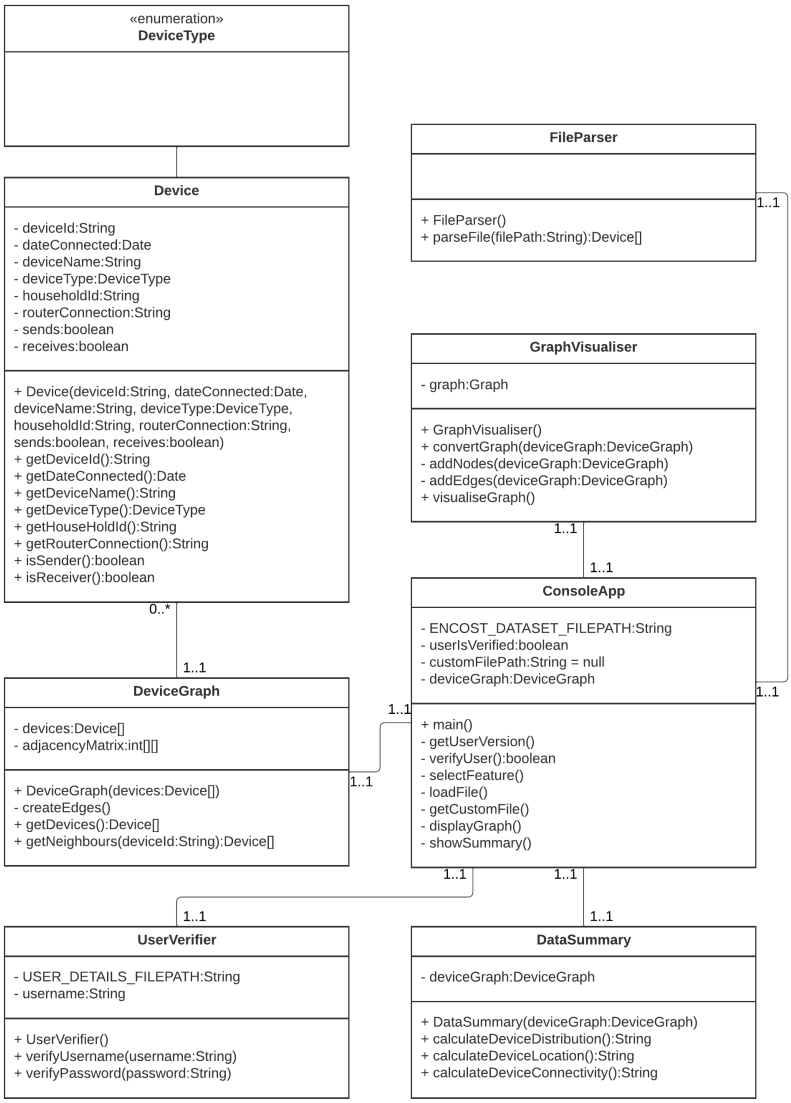

Figure 4.1: ESGP Application Class Diagram

4.1 Console App

The console app is the main component for this application. The console app will manage all interaction with the user. The console app's main function will proceed through the process detailed above in the process flow diagram, repeating display options until the user closes the application.

4.1.1 Getting user version

The getUserVersion function will ask the user to select a version in a loop until either they successfully log in and are verified or they enter as a community user and the loop is broken out of. The verifyUser function will be called if the user opts to log in to an Encost user account.

4.1.2 Verifying Encost user's details

The verifyUser function will create a UserVerifier object which will check the username, and then check the password. If either is invalid false will be returned to getUserVersion to indicate verification was unsuccessful. If both checks pass then true will be returned to getUserVersion.

4.1.3 Loading a data set

The loadFile function will create a FileParser object and use it to parse either the Encost data set, which is located by the constant ENCOST_DATASET_FILEPATH, or the custom data set which is located by the variable customFilePath. Which to use will be determined by whether or not customFilePath is null or not. The array of Device objects will be returned by FileParser if the information in the file is valid, otherwise null will be returned. If not null, this information will passed to the DeviceGraph constructor, which will return a new DeviceGraph to be stored in deviceGraph.

4.1.4 Selecting a feature

The selectFeature function will be displayed to the user, with the options depending on whether or not the user is an Encost user or not. This will be determined by checking the userIsVerified variable. It will call one of displayGraphics, getCustomFile, or showSummary depending on user input. This function will be repeated until the user closes the app.

4.1.5 Displaying a graph

The displayGraph function will create a new GraphVisualiser object which will visualise the either the Encost data set or a custom data set as a graph data structure. This will be described in more detail in section 4.6.

4.1.6 Getting a custom file

The getCustomFile function will prompt the user to enter a file path. If it is a valid file path, loadFile will be called to replace deviceGraph with the information from the new data set.

4.1.7 Showing summary of statistics for data

The showSummary function will create a DataSummary object which can be used to calculate device distribution, location, and connectivity. The strings returned by these functions will then be printed to the user in the console.

4.2 Local Storage

Local storage will be used for accessing two types of data: user data, and data to be graphed. The user data and the Encost data set will be downloaded with the main app and their file paths will be stored as constants. Custom data sets can also be used by Encost users.

4.2.1 User data

The user passwords will be hashed so that users can not find the user data file and enter an Encost user's details to access privileged features.

4.3 User Verifier

The UserVerifier component will be used for verifying details when a user wants to verify themselves as an Encost user. The location where user details are stored can be found in the constant USER_DETAILS _FILEPATH. When a username is verified the username is stored so that the password can be verified as belonging to the correct user.

4.4 File Parser

The FileParser component will be used to process a data set file and convert each line into a Device object. It will check the validity of the file and return an array of Device objects if it is valid, otherwise it will return null.

4.5 Device Graph

The DeviceGraph component is the graph data structure that the client requested for storing all of the Encost Smart Device Objects. This graph will not be used for the visualisation, however the details getDevices and getNeighbours functions will be used to create graphs using the GraphStream Graph class.

4.6 Storing edges

The devices will be stored in an array. The edges will be stored in a 2D array adjacency matrix where the position of a device in the devices array matches it's position in the adjacency matrix. For example, adjacencyMatrix[1][0] = 1 would mean that the device in position [1] of the devicesArray sends data to the device in position[0] of the devicesArray. This does not mean that the opposite is true, as devices can send and not receive and vice versa.

4.7 Graph Visualiser

The GraphVisualiser component will use GraphStream to visualise a data set's devices and connections between them.

4.7.1 Converting a Device Graph

The convertGraph function will take a DeviceGraph object and convert it into a GraphStream Graph. It will first loop through all the devices in the DeviceGraph and add them as nodes. Then it will loop through all devices again, this time using the DeviceGraph function getNeighbours to add edges for each device.

4.7.2 Graph Visualisation Example

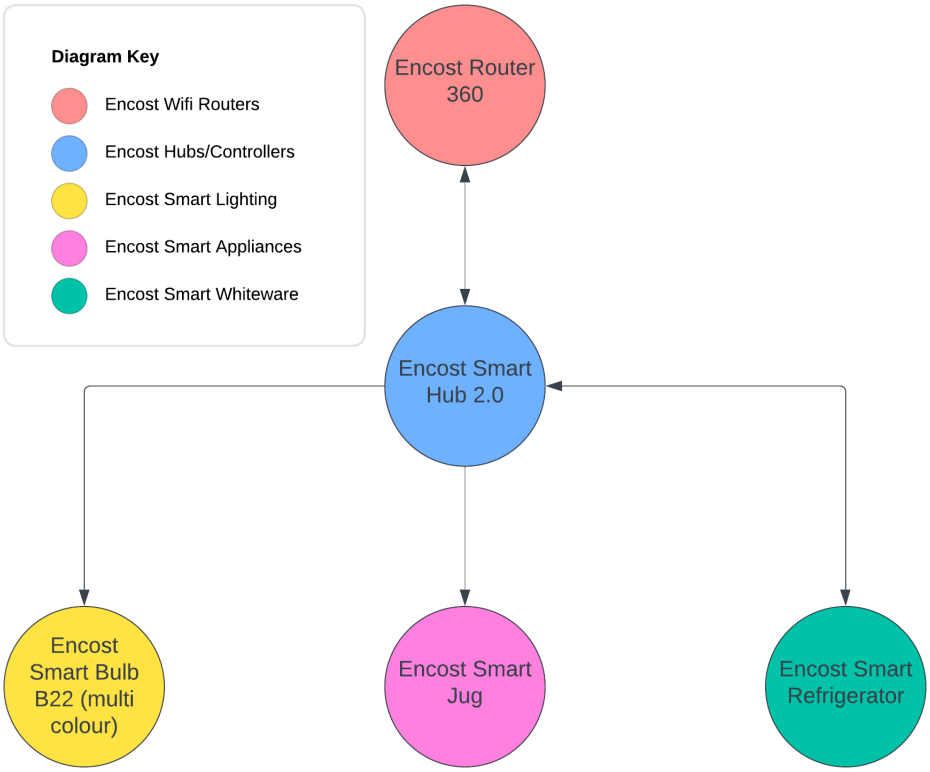

Figure 4.2: ESGP Application Graph Visualisation Example

The above diagram displays what the visualisation should look like. Different colours will differentiate between types of device. An arrow indicates data being sent. In this example, Encost Router 360 and Encost Smart Hub 2.0 both send data to each other, however, Encost Smart Bulb B22 (multi colour) does not send data to Encost Smart Hub 2.0, as there is no arrow pointing to Encost Smart Hub 2.0.

4.8 Data Summary

The DataSummary component will receive a DeviceGraph object and use it to calculate the distribution, location, and connectivity summary statistics for the data set.

4.8.1 Calculating device distribution

The calculateDeviceDistribution function calculates the number of devices that exist in each device category. The number of devices that exist for each device type should also be calculated. These figures are then converted to a string which can be returned to ConsoleApp to be printed to the console.

4.8.2 Calculating device location

The calculateDeviceLocation function will calculate various statistics using the ISO 3166-2 standard published by the ISO 3166 Maintenance Agency (ISO 3166/MA)[1]. These statistics that are calculated are:

- the number of devices that exist in each region in New Zealand
- the number of devices that exist in each household, in each region in New Zealand
- the number of devices in each category, for each household, in each region in New Zealand
- the number of devices in each category, for each region in New Zealand

These figures are then converted to a string to be returned to ConsoleApp and printed to console.

4.8.3 Calculating device connectivity

The calculateDeviceConnectivity function will use the DeviceGraph to calculate the following statistics:

- the average number of devices that an Encost Wifi Router is connected to
- the minimum number and maximum number of devices that an Encost Wifi Router is connected to
- the average number of Encost Hubs/Controllers that an Encost Smart Device is receiving commands from
- the minimum number and maximum number of Encost Hubs/Controllers that an Encost Smart Device is receiving commands from
- the average number of Encost Smart Devices that an Encost Hub/Controller is sending commands to
- The minimum number and maximum number of Encost Smart Devices that an Encost Hub/Controller is sending commands to

[1] `https://en.wikipedia.org/wiki/ISO_3166-2:NZ`

These figures are then converted to a string to be returned to ConsoleApp and printed to console.

5 Conclusion

This document should be used in combination with the Software Requirements Specification for developing and maintaining the ESGP application as requested by our clients, Encost.