SOFTWARE DESIGN

for

Encost Smart Graph Project

Version 1.0

Prepared by: Student 3
SoftFlux Engineer

SoftFlux

April 4, 2024

Contents

Revision History

Name	Date	Reason for Changes	Version
Student 3	26/03/24	Starting SDS	1.0
Student 3	1/04/24	Adding to UI	1.1
Student 3	2/04/24	Fleshing out Diagrams with descriptions	1.2
Student 3	4/04/24	Final touches and proof read	1.3

1 Introduction/Purpose

1.1 Purpose

The purpose of this document is to provide details of the software design for the Encost Smart Graph Project (ESGP) and will explain its Software architectures and its components such as user interface and framework. It will showcase the capabilities, design and framework of the software. This document is for comparing and testing against the final product to see whether this software is successfully implemented.

1.2 Document Conventions

This document uses the following conventions:

ESGP: Encost Smart Graph Project

SDS: Software Design Specification

1.3 Intended Audience and Reading Suggestions

- **Software Developers:** who will use this document to structure and develop this software. The SDS will provide them with a guide for developing a solution that will fit the requirements.

- **Quality Assurance Engineers:** Testers and QA Engineers will use this SDS to create test plans and cases to ensure software implementation aligns with design specs and meets functional and non-functional SRS requirements.
- **Project Managers:** To make decisions about project planning, resource allocation, and risk management.

1.4 Project Scope

The Encost Smart Graph Project (ESGP) is a software system designed to visualise Encost's smart devices using a graph structure and provide summary statistics on device distribution, location, and connectivity when provided with the Encost Smart Homes Dataset.

ESGP will have the ability to process data and visualise it in a graph form for community users. Encost Users can be Authenticated allowing them more features such as loading custom datasets and viewing the statistics. The system will integrate with the GraphStream library and run on Windows 10 with Java 1.8.0 or higher. Maintenance will be handled by SoftFlux.

2 Software Architecture

2.1 Component Architecture

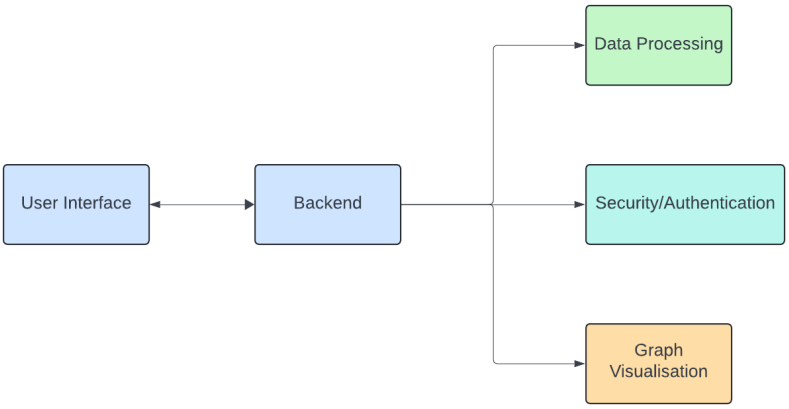

Figure 2.1: Component Diagram

The structured system shown in the component diagram is designed to ensure a clear separation of tasks and maintainability. The division into User Interface, Backend and Data Components aligns with software engineering principles, ensuring modularity and scalability. The User Interface Component handles user interactions, hiding away the logic from the backend Component, which handles the system's functionalities and interacts with the data, authentication and visualisation components. This enables easier maintenance, testing, and future developments, improving the overall quality and longevity of the software system.

2.2 Process Diagram

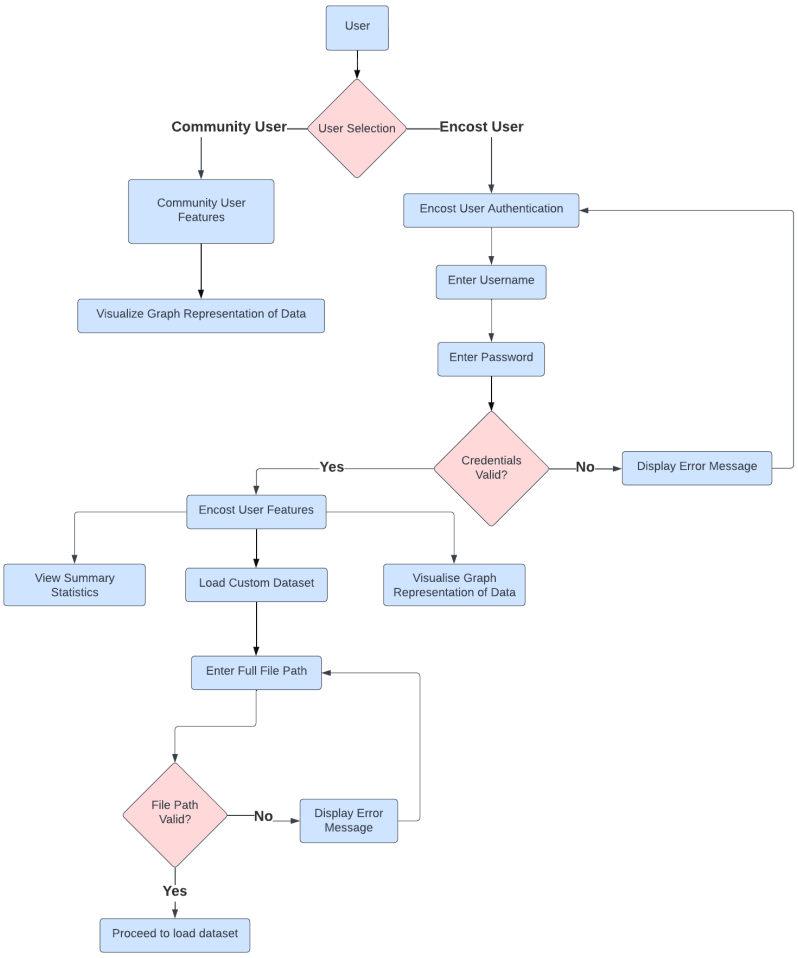

Figure 2.2: Process Diagram

The process diagram, shows the actions and choices users make when using the soft-

ware. Users are prompted to select their user type, either Community User or Encost User. Community Users are given a simpler path, with only the option to visualise the graph. Encost Users have to get authenticated before getting access to their features like loading custom datasets, graph visualisation, and viewing summary statistics. This gives an intuitive user experience, guiding users through each step suited to their needs and privileges within the system.

2.3 Deployment Diagram

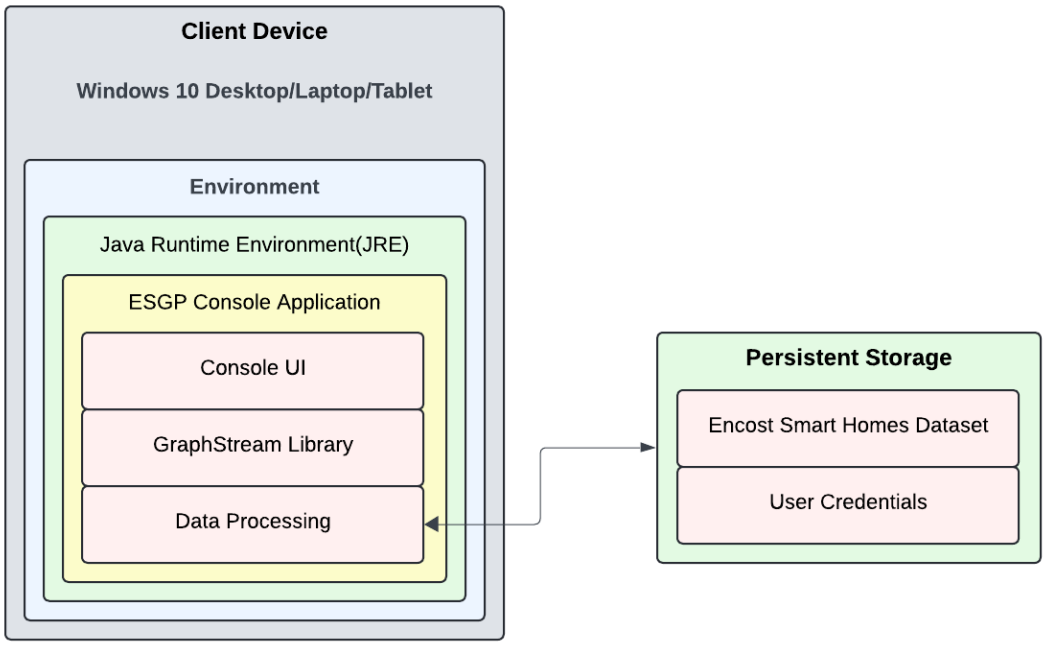

Figure 2.3: Deployment diagram

The diagram shows the interactions between the different parts of the system. The client device, which could be a Windows 10 desktop, laptop, or tablet, runs the ESGP application. This application is developed using Java and uses the GraphStream library to visualise the Encost Smart Homes Dataset to provide users with a graph of device connectivity. The sensitive data is local and encrypted, ensuring greater control of data privacy and security. Using Java allows for wider compatibility across different various devices. The design aims to prioritise data security, usability, and compatibility.

2.4 Software Design decisions

Originally it was thought to use a database to store all the persistent data. However, this was changed to local storage to ensure greater control over the security and for simplicity. The User Credentials will be encrypted and only decrypted during run time. The SRS doesn't explicitly state that custom datasets need to be stored within the persistent storage. So any custom data set uploaded will only remain in memory and not be stored for the next startup of the application.
To make sure Brute force can't be used to log in as an Encost user. The user only gets 10 login attempts before a timeout.

3 Component Design

3.1 User Interface

The User Interface Component focuses on simplicity and clarity. Since the ESGP application is console-based, the UI is text prompts displayed to the user in the command line. The design focuses on clear textual output, with neatly formatted information for readability. The user prompts and menus are made to guide users through the different functions. Error messages are descriptive, to help users troubleshoot issues. This design approach is a user-friendly experience that aligns with the software's functional requirements.

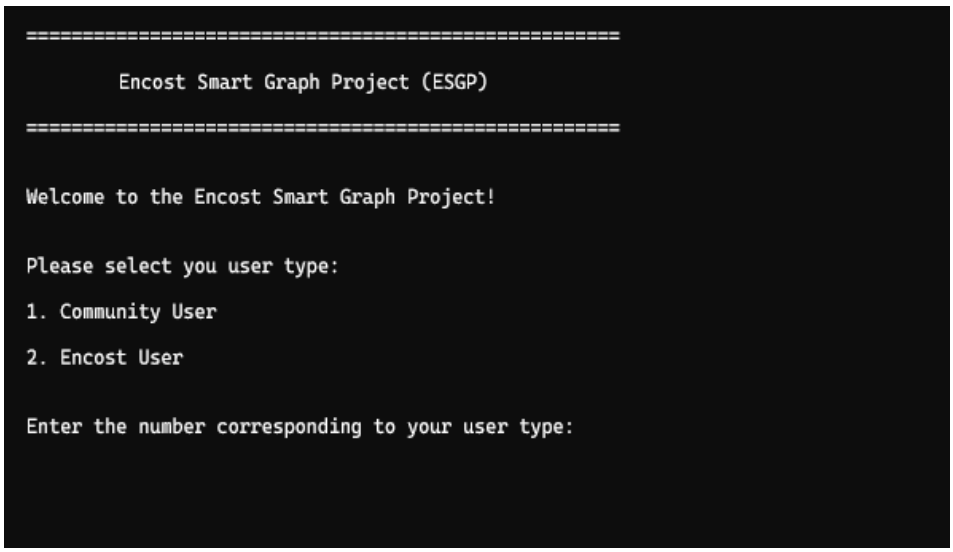

Figure 3.1: Opening Screen

This screen allows users to choose between "Encost User" or "Community User." Once the user selects their user type, they proceed to the next menu.

If the User chooses Community User the UI will show them the options they have which is limited to visualise Graph or navigation options.

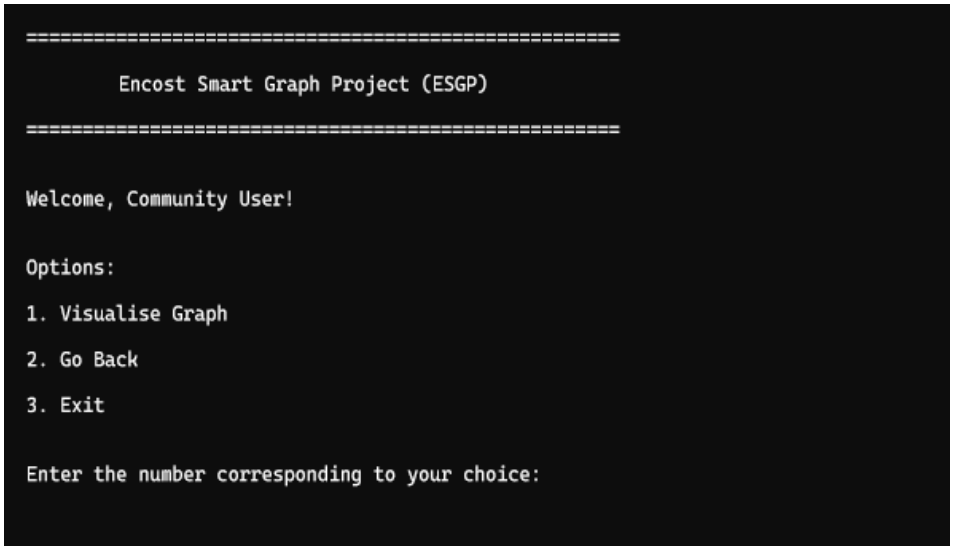

Figure 3.2: Community User Screen

Else User chooses Encost User they are shown the option to log in, go back or exit.

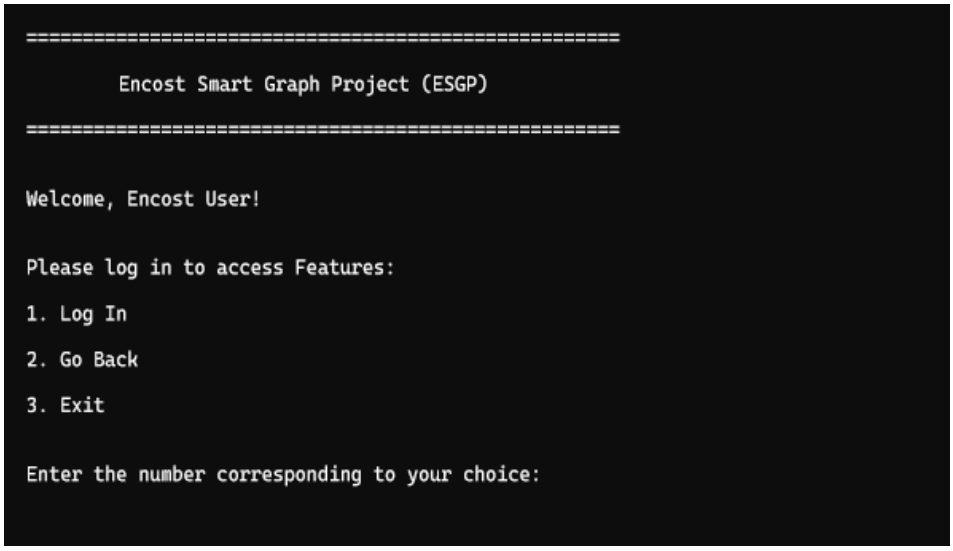

Figure 3.3: Encost User Screen

The user chooses to log in and provides their login details

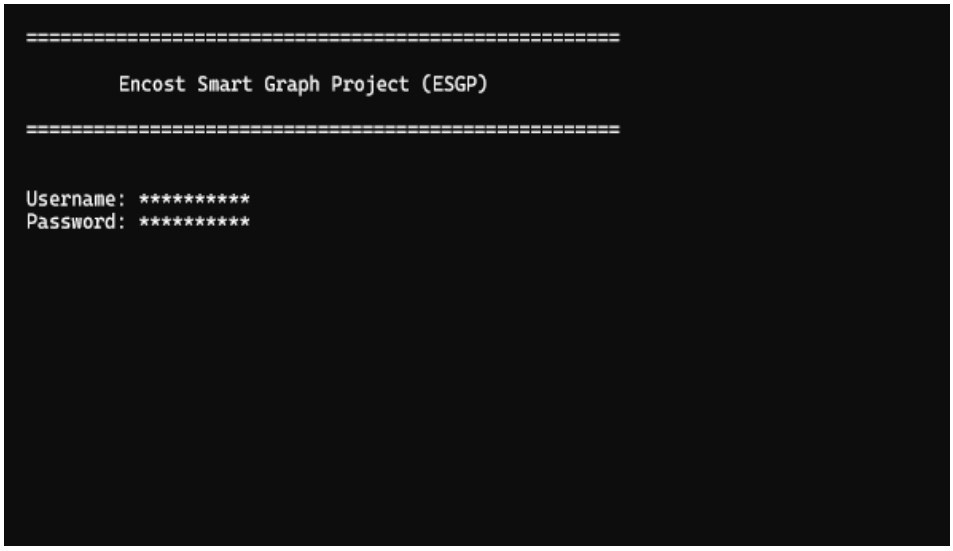

Figure 3.4: Encost User Log in Screen

Incorrect details provided the user gets an error message and is asked to retry showing how many tries left before timeout.

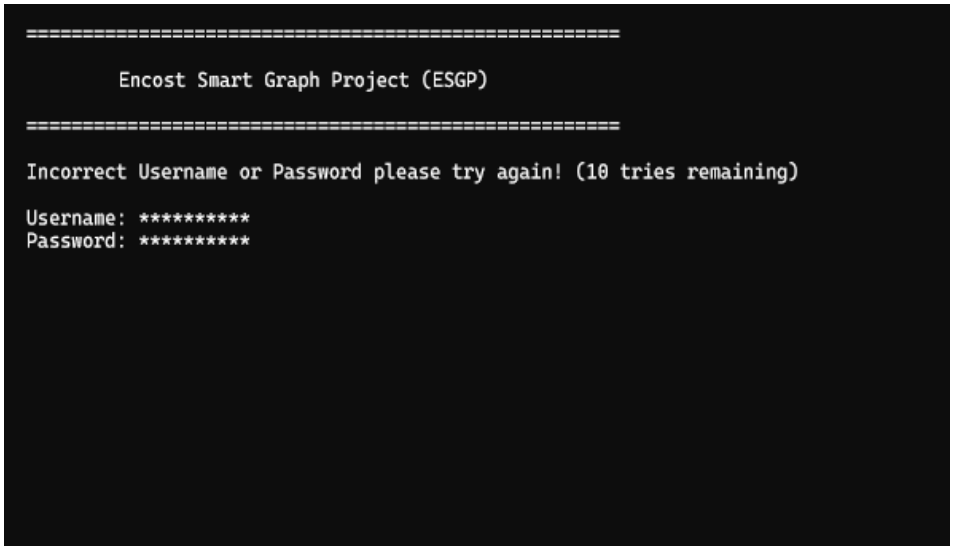

Figure 3.5: Encost User Log in Screen, Incorrect Login

Logged in Encost User, showing the features available to them

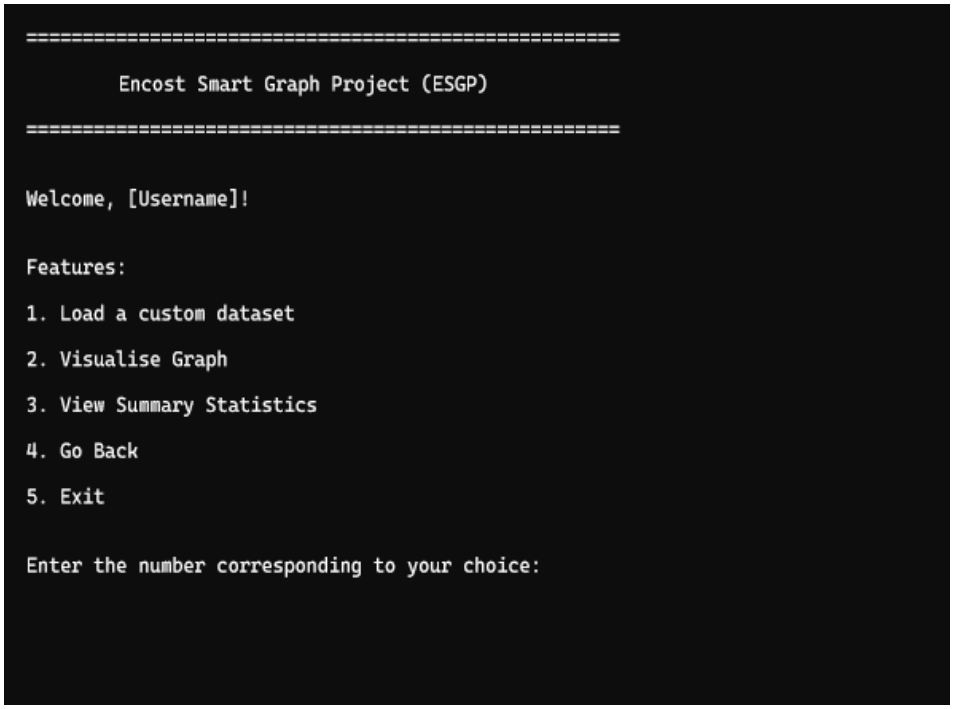

Figure 3.6: Encost User Features

The user chooses to view summary statistics

```
Summary Statistics:

1. Device Distribution:
   - Category 1: <Number of devices>
   - Category 2: <Number of devices>
   ...
   - Total Devices: <Total number of devices>

2. Device Location:
   - Region 1:
     - Total Devices: <Number of devices>
     - Devices by Category:
       - Category 1: <Number of devices>
       - Category 2: <Number of devices>
       ...
   - Region 2:
     - Total Devices: <Number of devices>
     - Devices by Category:
       - Category 1: <Number of devices>
       - Category 2: <Number of devices>
       ...
   ...
   - Total Regions: <Total number of regions>

3. Device Connectivity:
   - Average Devices Connected per Router: <Average number>
   - Min Devices Connected per Router: <Minimum number>
   - Max Devices Connected per Router: <Maximum number>
   - Average Devices Receiving Commands per Hub/Controller: <Average number>
   - Min Devices Receiving Commands per Hub/Controller: <Minimum number>
   - Max Devices Receiving Commands per Hub/Controller: <Maximum number>
   - Average Devices Sent Commands to per Hub/Controller: <Average number>
   - Min Devices Sent Commands to per Hub/Controller: <Minimum number>
   - Max Devices Sent Commands to per Hub/Controller: <Maximum number>

Press any key to go Back:
```

Figure 3.7: Summary Stats Screen

The user chooses to view graph

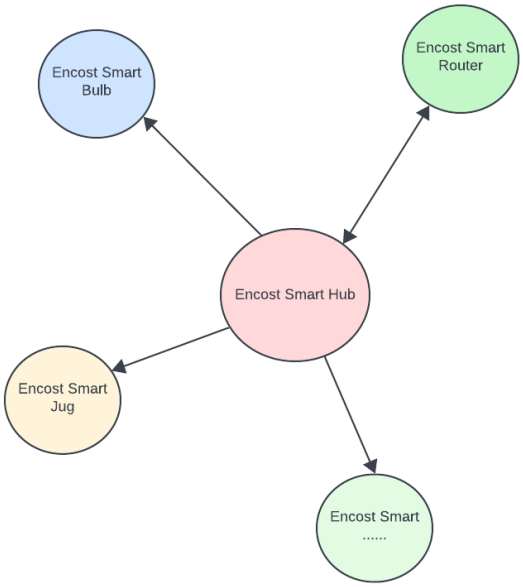

Figure 3.8: User Interaction with Graph Visualisation

Graphstream creates a new window for their graphs. Therefore this part of the system doesn't use a console UI. The graph visualises the connection between the nodes(Encost Devices) and their connectivity, as well as which devices can send/receive based on the arrow's direction. The colours are used to represent the device category.

The User Chooses to load a custom dataset

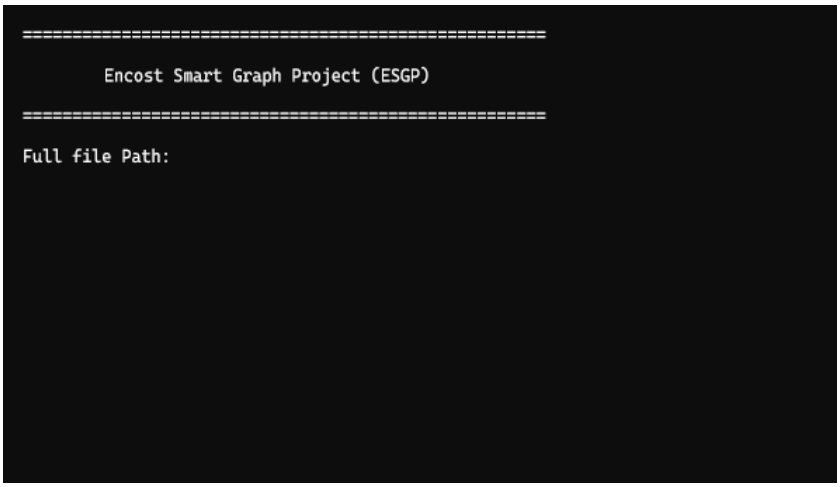

Figure 3.9: User Interaction loading Custom Dataset

The application will prompt the user for the full file path.

If the file type, format or location is incorrect the user will get an error corresponding to which was incorrect. In the image, it shows all 3 however only 1 will be shown depending on the error type.

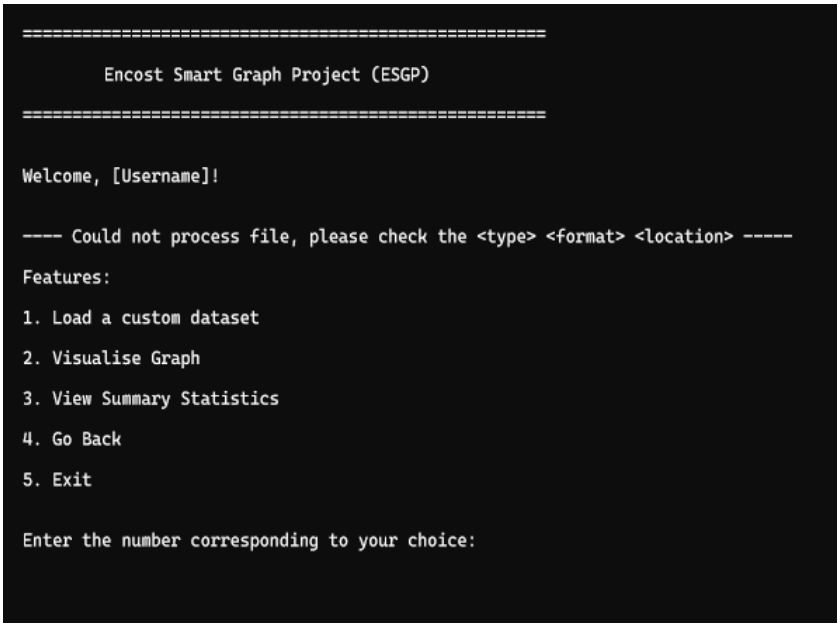

Figure 3.10: User Interaction Incorrect Custom Dataset

In part 5 of the SRS it was stated that should a feature take longer than a second to load the User should be made aware.

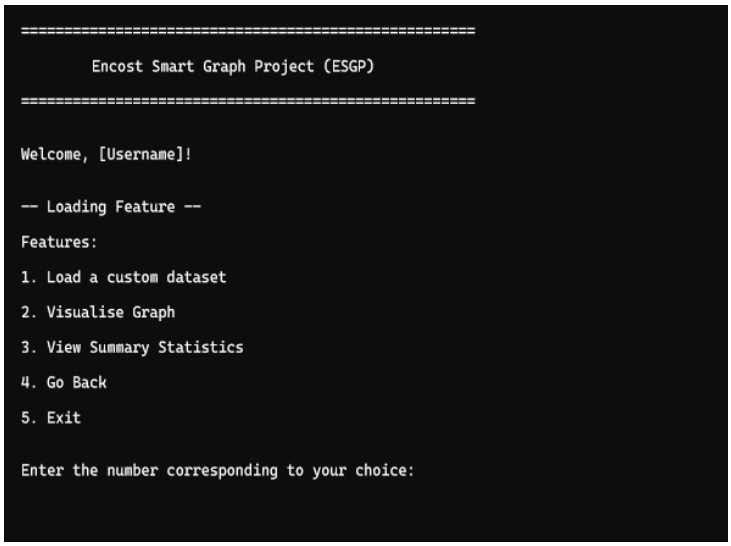

Figure 3.11: User Interaction with waiting for a feature

If a feature takes longer than a second to load they will be told that it is loading

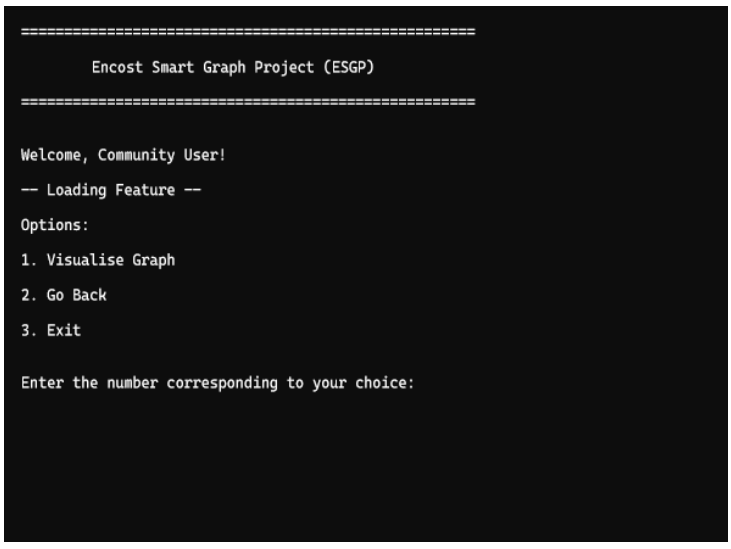

Figure 3.12: User Interaction with waiting for a feature

Which is the same for the community user.

3.2 Use Case Diagram

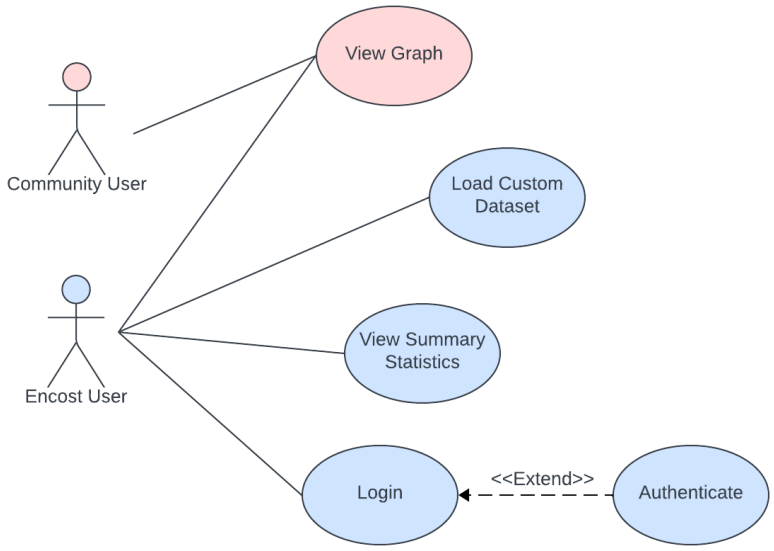

Figure 3.13: Use Case Diagram

The use case diagram shows the functionalities of the ESGP system for both Community Users and Encost Users. Community Users are able to view the graph representation of the Encost Smart Homes Dataset. Encost Users, after logging in securely, have the additional capabilities to load custom datasets, view graph visualisations, and access summary statistics.

3.3 Message Sequence Diagrams

For a more visual understanding please refer to 3.1

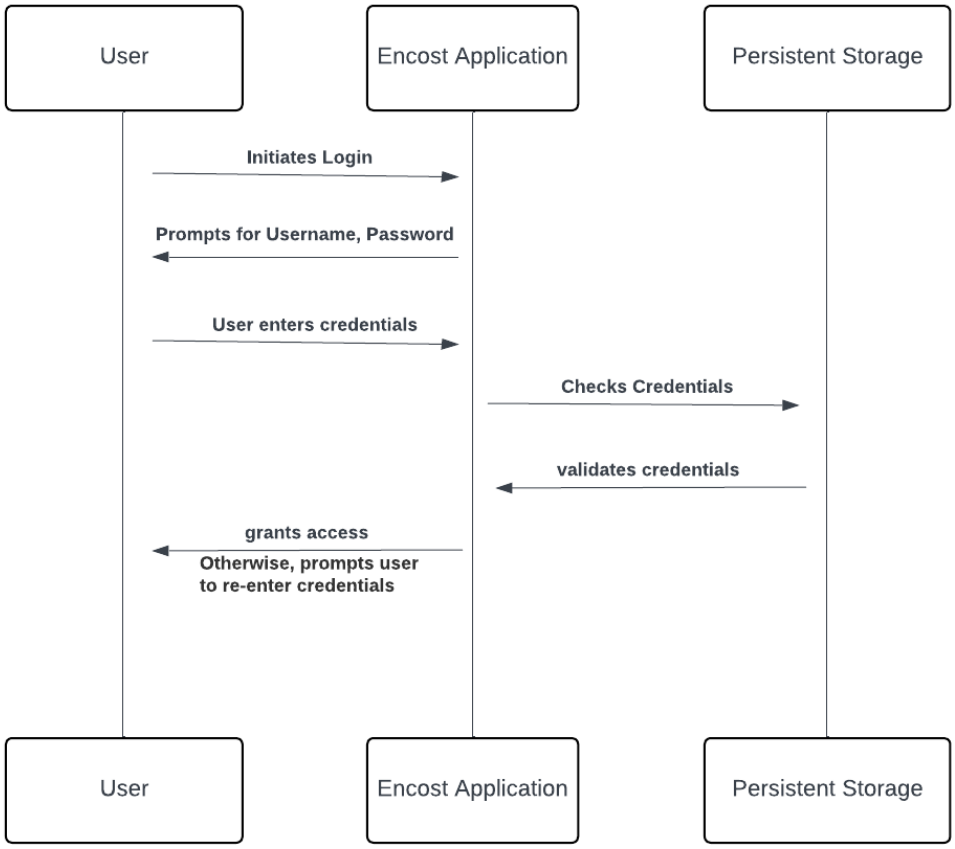

Figure 3.14: User Interaction with Login Screen

This diagram shows the flow of interaction between the User, Application and data. The user initiates login and the application gets the credentials from the user and validates it against the user/password key pairs stored in the persistent storage and either grants access, or prompts re-entry.

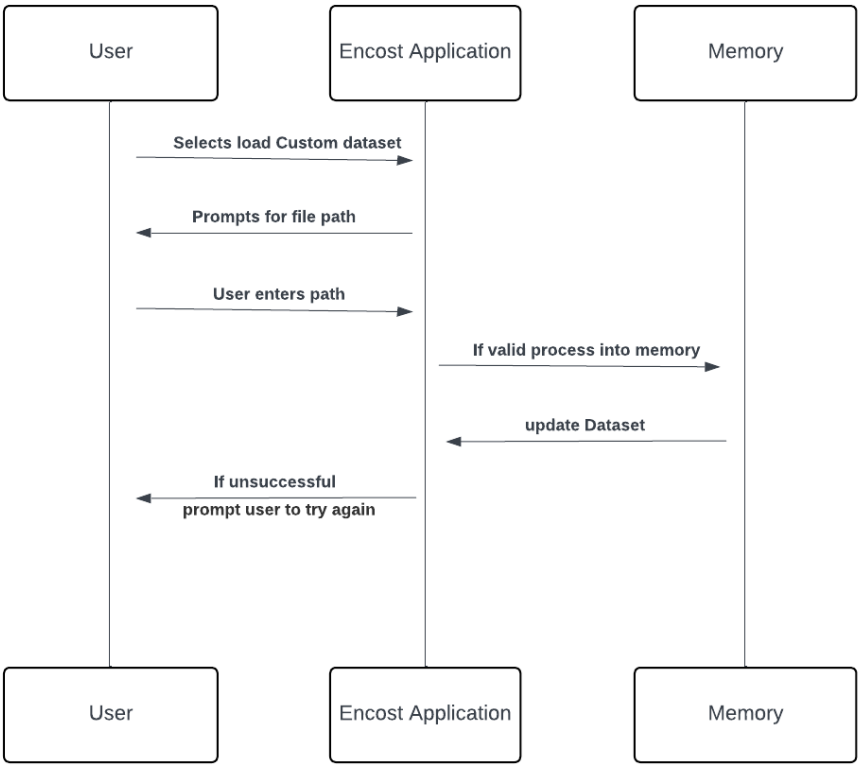

Figure 3.15: User Interaction with Loading Dataset

This Message sequence diagram depicts the flow of data when the user wants to load a custom dataset. The system will prompt for a full file path, The application will attempt to read the file and if any errors occur it will prompt the user otherwise on successful read it will return to the feature menu with no errors.

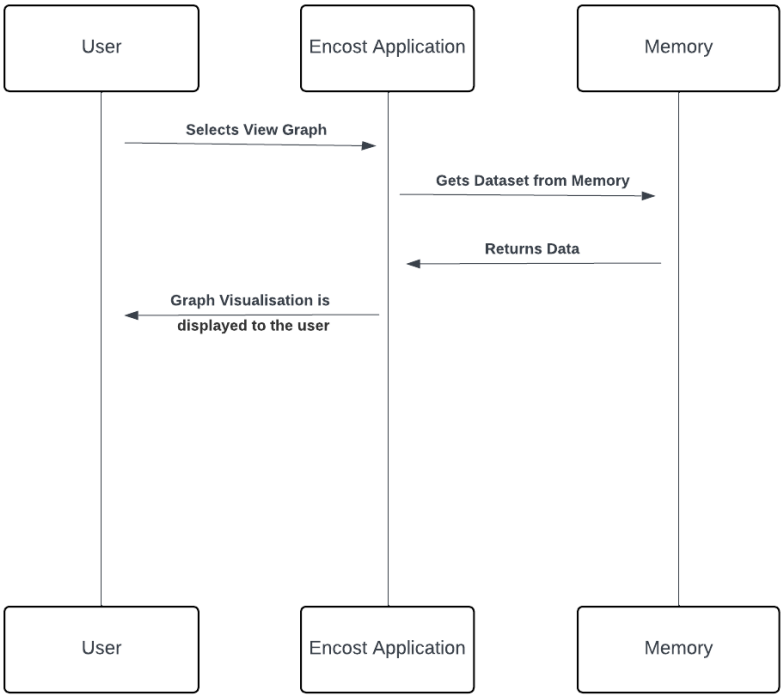

Figure 3.16: User Interaction with Graph Visualisation

The diagram shows the flow of interaction when the user chooses to view the graph. The application doesn't need to read in the data from the persistent storage as it does this on load and when uploading a custom dataset. The data is stored in memory. Not depicted in the diagram but if the application takes longer than 1 second a loading prompt will appear 3.12

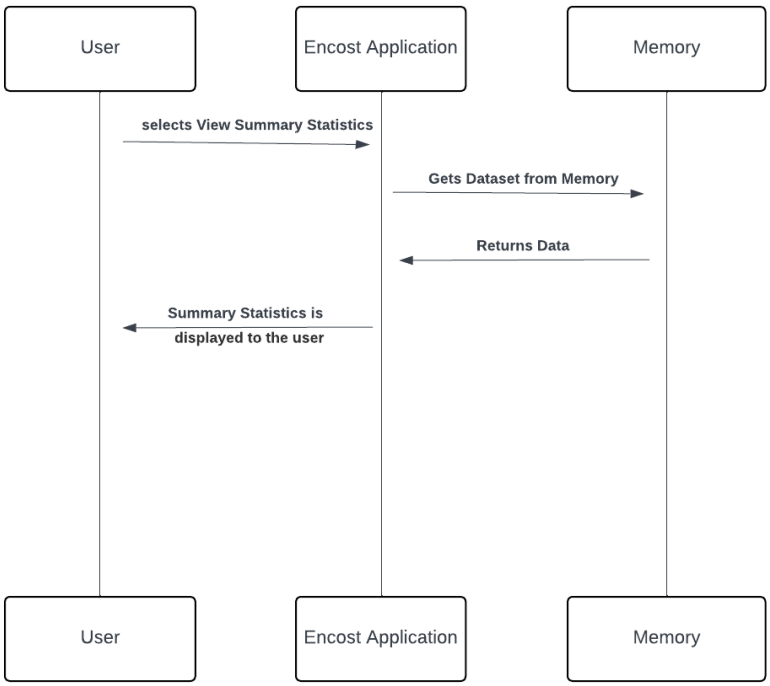

Figure 3.17: User Interaction with View Summary Statistics

The diagram shows the flow of interaction when the user chooses to view the statistics. The application doesn't need to read in the data from the persistent storage as it does this on load and when uploading a custom dataset. The data is stored in memory. Not depicted in the diagram but if the application takes longer than 1 second a loading prompt will appear 3.12

3.4 Class Diagram

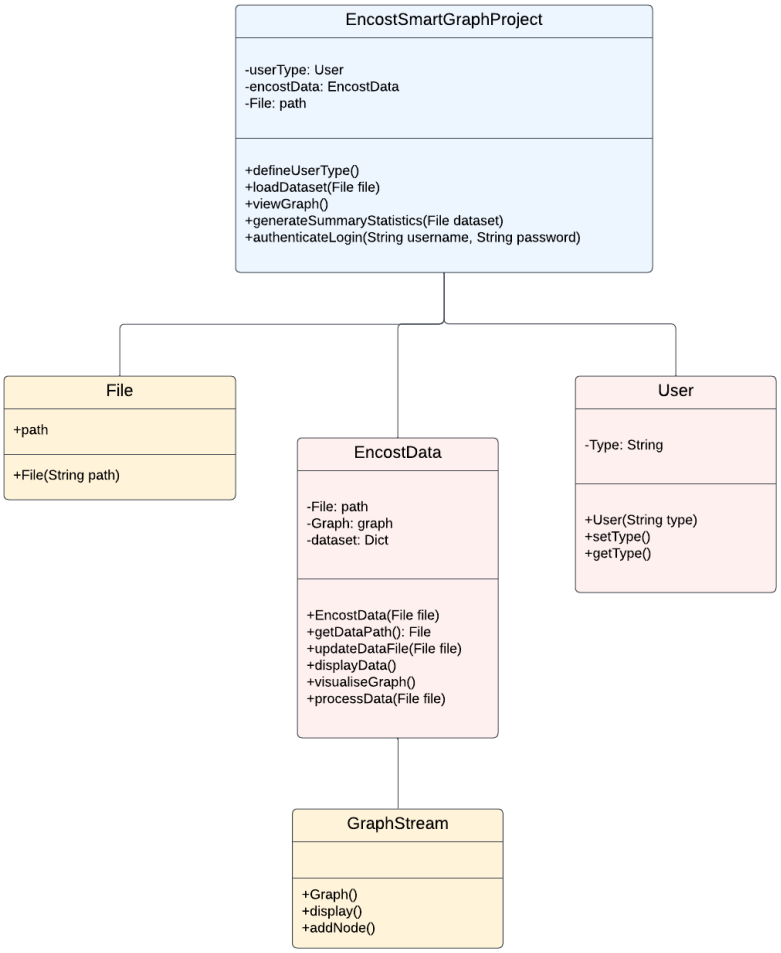

Figure 3.18: Class Diagram

The class Diagram shows the interaction between the classes and objects. The EncostSmartGraphProject Class handles the main logic; navigating through UI and handling user interaction. The EncostData class handles the features available to the users. Such as displaying the graph, summary statistics and processing the dataset into memory.

The user object contains information about the user, specifically the type of user they are. This is for modularity, allowing easier future expansion.

The File and Graph libraries are built-in/developed by other programmers and will be implemented into the solution.

4 Conclusion

The Encost Smart Graph Project (ESGP) is a software system that visualises Encost's smart devices and provides summary statistics. It allows Encost Users to load custom datasets and view statistics and graphs. Community Users can only visualise graphs. The system design is modular, with a separation of tasks into User Interface, Back-end and Data Components. The Component, Process, and Deployment diagrams show its structure, interactions and deployment. Design decisions are made with usability, security, and simplicity in mind, meeting the specified requirements.