
SOFTWARE DESIGN SPECIFICATION

for

Encost Smart Graph Project

Version 1.3

Prepared by: Student # 5
SoftFlux Engineer

SoftFlux

April 7, 2023

Contents

Contents	2
1. Introduction/Purpose	4
1.1. Purpose	4
1.2. Document Conventions	4
1.3. Intended Audience	4
1.4. Reading Suggestions	5
2. Scope	5
3. Extra/Specialised Requirements Specification	5
3.1. Licenses	5
3.2. Graph Styling	6
3.3. Users	6
3.3.1. Intended Users	6
3.3.2. User Preferences and Accessibility	6
3.4. Client Clarifications	6
4. Software Architecture	7
4.1. Use Case Diagram	8
4.2. Component Diagram	10
4.3. Process Flow Diagram	12
4.4. Other System Wide Design Decisions	13
4.4.1. Repetitive checks	13
5. Component Design	14
5.1. Class Diagrams	14
5.1.1. Navigation/Main Controller Architecture	14
5.1.2. User, User Roles, and Features	16
5.1.3. Login and Authentication	17
5.1.4. Graphing and handling Device Data	19
5.2. Sequence Diagrams	21
5.2.1. View Graph	21
5.2.2. Login	22
5.2.3. Loading a custom dataset	23
5.2.4. Calculating Summary Statistics	24
6. Screenshots	24
6.1. Console Prompts	25
6.1.1. Standard Input/Output console prompts	25
6.2. Calculate Device Summary Statistics	29

6.3.	Calculate Device Location Summary Statistics	30
6.3.1.	Number of households, devices, and average devices per household in each region	30
6.3.2.	Number of devices, number of households, and average number of devices per household by category by region	31
6.4.	Calculate Device Connectivity Summary Statistics	32
6.4.1.	Average, minimum, and maximum number of devices that Encost Router is connected to	32
6.4.2.	Average, minimum, and maximum number of Encost Hub/Con- trollers that a device is receiving commands from	33
6.4.3.	Average, minimum, and maximum number of devices that an En- cost Hub/Controller is sending commands to	33
6.5.	Mockup GraphStream Nodes	34
7.	Conclusion	36
A.	Appendix GraphStream Mockup SVG Icon Attributions	38

Revision History

Name	Date	Reason for Changes	Version
	23/03/2023	Added Purpose statement	1.1
	4/04/2023	Updated Architecture structure	1.2
	7/04/2023	Final draft completed	1.3

1. Introduction/Purpose

1.1. Purpose

The purpose of this document is to define the Software Design Specification (SDS) for the software system called Encost Smart Graph Project (ESGP). This document outlines the specialised requirements, software architecture, and detailed component design that will provide the basis for software construction. This document has been written based on the software requirements outlined within the Software Requirements Specification document version 1.0 dated February 7, 2023.

The content in this document shall form the base for implementing and subsequent testing of the software and it should be read in conjunction with the latest, approved Software Requirements Document.

1.2. Document Conventions

Unified Modelling Language(UML) diagrams are used to illustrate the architecture, design, and implementation of the proposed software system. Wherever possible the notation shall follow the current version of UML which is 2.0.

1.3. Intended Audience

This document is intended for Encost and SoftFlux project team members such as project managers, developer/programmers, software testers, end users, and requirements engineer.

Each intended audience type and suggested use for this document are outlined below:

- Project Managers - to oversee the details of the project in order to minimise risks.
- Developer/Programmers - to confirm, negotiate, and implement the proposed software solution.
- Software Testers - to identify issues that might come up in the testing phase.
- End Users - to provide feedback both good and bad on the design.
- Requirements Engineer - to ensure that the proposed solution satisfies the requirements. Wherever necessary, changes should be made or negotiated in consultation with affected parties.

1.4. Reading Suggestions

Before reading this document, it is recommended that readers be familiar with the software requirements as outlined in the Software Requirements Document.

2. Scope

The ESGP software system shall cover the provision of the Encost Smart Homes Dataset (ESHD) for Encost devices and the visualisation of these devices in a graph structure, following the agreed delivery requirements on:

- Data handling including input and output
- User login and feature management
- Visualising device data using a graph structure
- Interface management
- Software performance
- Security
- Software quality

3. Extra/Specialised Requirements Specification

3.1. Licenses

The usage of the GraphStream Java library must comply with the license obligations as outlined on the software publisher's website¹.

¹<https://graphstream-project.org/>

3.2. Graph Styling

Default color schemes and styling will be implemented for the graph styling. It is recommended that the client provide more brandcentric product photography, product descriptors, and definitive colour schemes that reinforce their brand image with the end users who will be using this system.

3.3. Users

3.3.1. Intended Users

The software shall be designed for two intended user types - Community users and Encost users. It is recommended that the software should be designed in such a way that other user types can be added or that access to features can be adjusted over time based on user type.

3.3.2. User Preferences and Accessibility

No allowances have been made for visually impaired users. For operability, the software primarily requires the use of a keyboard, mouse, and display device such as a computer screen for viewing the graph representation of the device data. The software interfaces shall display textual prompts in English using default styles and it is recommended that graphical representations utilise default colours, shapes, and sizes from the GraphStream library. Wherever possible styling and language should be used that is clear and concise to the user. There is no requirement for end users to change style or language preferences.

3.4. Client Clarifications

- Devices per household per region and devices per category are both to be averages.
- The file structure for the smart device data will remain the same, however, new device categories and types may be added over time.
- There is no preference to persist the filepath for custom datasets.
- A device will only ever have 1 device category.
- There is no specific location for the ESHD file to be stored in.
- Initialising of device objects using a Device class can be done at anytime.
- Malformed or invalid field values in custom datasets should be handled when the dataset is loaded and parsed and feedback passed to the user.
- The details for software maintenance are to be covered in a separate document.

- The software shall accept a .txt file with data that is comma separated. An informative warning message will be displayed to the user if they try to load data of an unrecognised format. Similarly, if a user tries to load a custom data-set that does not conform to the ESHD data format, then an informative warning will be displayed.
- Storing of Encost user's password and username information should utilise one-way hashing using the SHA-256 algorithm.

4. Software Architecture

4.1. Use Case Diagram

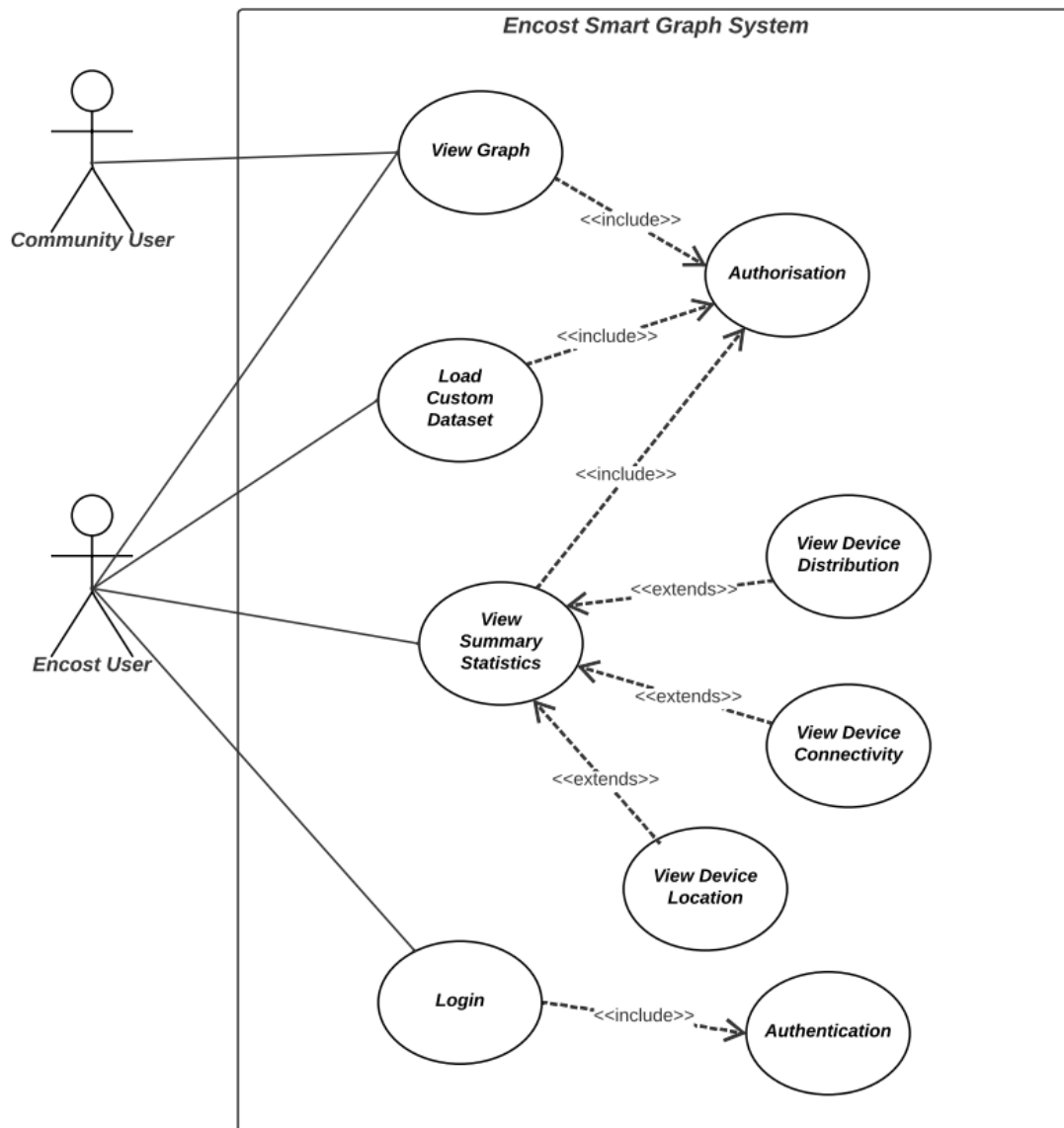


Figure 4.1.: Use Case Diagram

This use case diagram shows the high level view of the proposed system and how actors interact with the system.

- There are currently 2 user roles: 'Community Users' and 'Encost Users', however more user roles may be added in future.
- The functionality to view a graph is accessible to both user roles while more specialised functionality such as viewing summary statistics and loading a custom data-set are accessible to Encost users who are required to provide valid login and password credentials. Recommend that a user's access to the 3 primary features should be checked at every step. This supports a system-wide consistent approach and ensures access to features is enforced by user type.
- An authentication service is recommended to handle the process of authenticating an Encost user's login and password credentials that are provided through the login component. There are currently 3 user-types: community, encost-unverified, and encost-verified. It is recommended, that by default, the community user-type is assigned to all users upon initialising the software. This is so that a role is applied upon starting the application and the user is identified on entering the software. Further details of how this fits in the overall architecture of handling user roles and access privileges are explained in the detailed design on user roles in the next section.
- Encost-verified is assigned to a user after valid login credentials have been provided at login and remains active for the duration of the user session. It is recommended that roles are not remembered between sessions which is a security measure.
- An authorisation service is recommended to handle and enforce access to software features based on the role of the user. This layer of checking prevents unauthorised access to software features.

4.2. Component Diagram

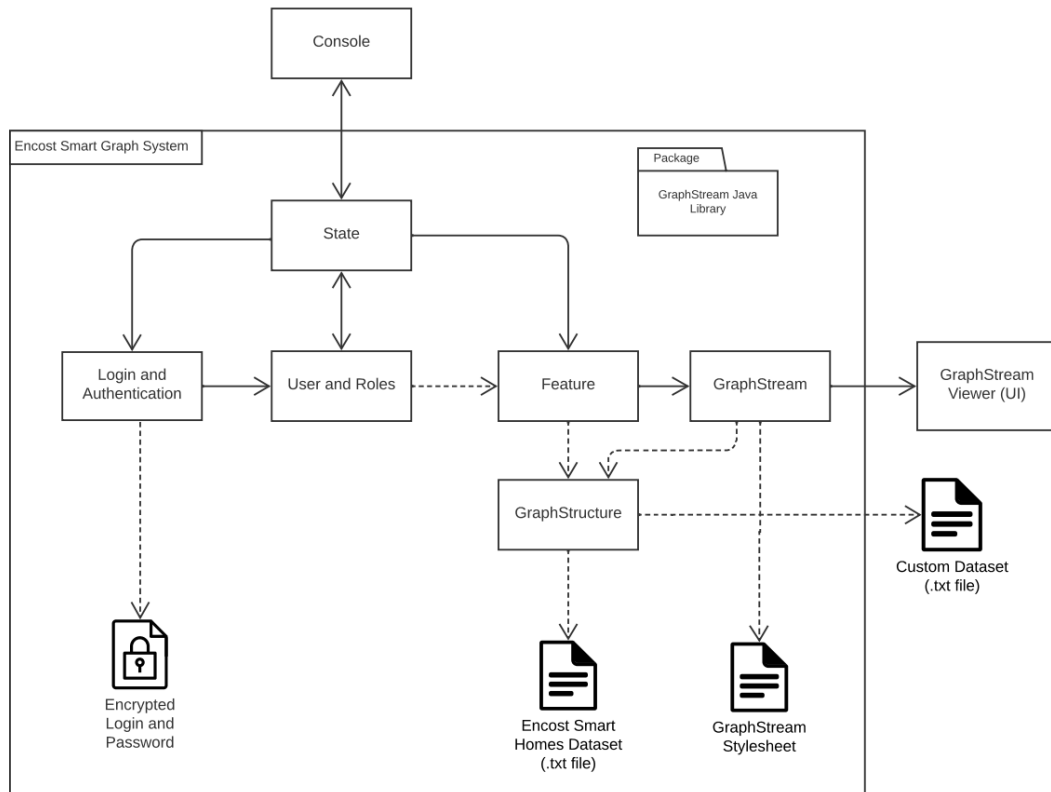


Figure 4.2.: Component Diagram

This component diagram shows the overall structure of the system from a component point of view.

- Users interact with the system using the console. This is the main mechanism for inputting commands to console prompts and receiving textual output. Console input and output is received and sent from and to the State component. The reason for this is to centralise the control of the application and enables sophisticated functionality to be implemented at the state level.
- There is a dependency to the GraphStream Java library which is an external package. More documentation about this product can be viewed on their website¹.
- A State component will handle the main control of the system and interfaces directly with the user. This component will be responsible for checking user input

¹<https://graphstream-project.org/>

and sanitising it to prevent users from sending input that might destabilise the system or for malicious purpose. Input checking methods should be used, such as regular expressions. Input can be checked for errors and feedback sent to the user to use correct input.

- The Authentication component will handle requests to authenticate Encost users who are required to provide valid login and password credentials through the console. These details will be authenticated against the encrypted login and password details that are saved in the system. One-way hashing with the SHA-256 algorithm is recommended. The reason for abstracting this component out is to implement the chain-of-responsibility design pattern. More details are outlined in the detailed component design section.
- The User and Roles component is responsible for managing a user's role and access privileges to the features. This enables other system components to check access privileges from one centralised point. There is a dependency on the Features component that encapsulates the functionality and attributes for each feature.
- Data is read in from the Encost Smart Homes Dataset situated in the system or a custom dataset (for Encost users only) that is situated elsewhere on the users machine. This data is used to calculate statistics and outputted to the console or it is used to generate visual graph structures that are displayed using the GraphStream Viewer functionality in a user interface(UI) window on the user's machine. A breakdown of the various components required to do this is outlined in the next section and reasoning for why they should be separated as such. There is a dependency to a stylesheet that holds all of the GraphStream styling attributes.

4.3. Process Flow Diagram

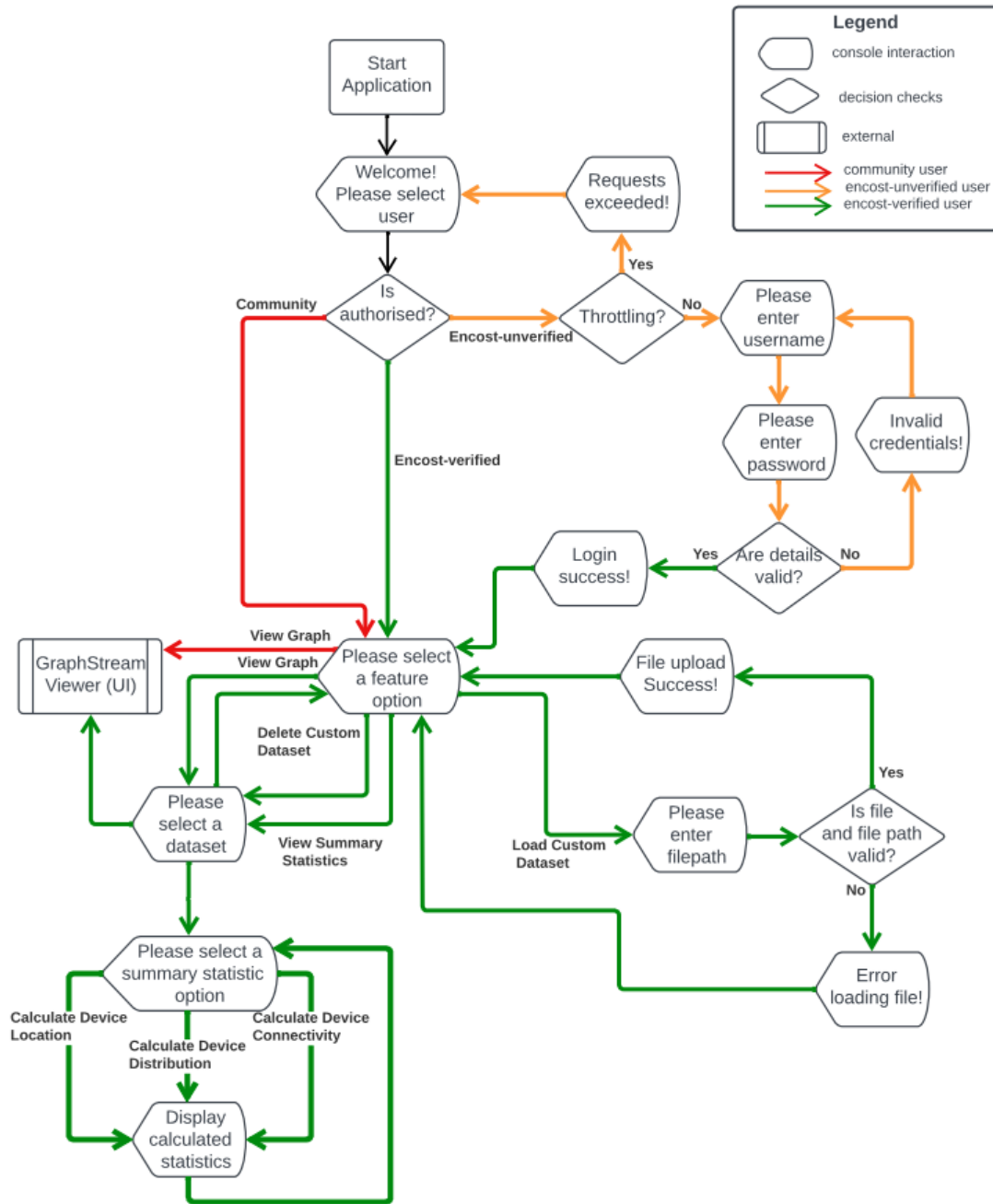


Figure 4.3.: Process Flow Diagram

This process flow diagram shows the main steps that occur in the software and illustrates the way a user might navigate and interact with the different software features.

- At the 'Welcome! please select a user' step it is necessary to check the user role to determine which way they go. This is to support navigation of states. For example, if the user is already 'encost-verified' then the user should not be required to login again and will simply move forward to the feature option state.
- The filepath for the custom dataset is the fully qualified filepath for the location of the file. There will be a default location for this file in the directory structure.
- The throttling mechanism illustrated in the Encost-unverified flow is a preventative measure to slow down brute force attacks at the login step. It is recommended to implement such functionality. This is primarily a security measure.
- Recommend structuring the system so that Encost users can load multiple custom datasets, be able to choose which one they want to graph, and also navigate to the ESHD when required. The reason for this is a user experience enhancement. The consequence is that a restriction should be put on the space and memory whether that be number of files or total size of combined files. It is recommended that users are restricted to a maximum of 10 files and space of 50MB or some such size that does not severely impede performance or cause the application to crash.

4.4. Other System Wide Design Decisions

4.4.1. Repetitive checks

These features have not been included in the diagrams but it is recommended that they implemented with reasons outlined.

- Recommend adding 'cancel/go back' and 'exit the application' options in every step of the process. This is a user experience enhancement and stops the possibility of users being 'trapped' in screens.
- Recommend that input received from the console is sanitised and checked for conformity. Feedback is sent to the console if the input does not conform. This improves the user experience and also prevents malicious attacks. An additional check and rate limiting approach is recommended for instances where input is spammed to the console, not just brute force attacks, and appropriate feedback and measures are taken.
- Recommend authorisation is checked whenever state changes are initiated. This is to prevent users accessing features that they shouldn't and also enables the tailoring of specific language and messaging to be used for outputting to the console wherever appropriate.

- There are 5 non-functional performance requirements to check if requests are taking longer than 1second and displaying feedback. These checks are not shown on the process flow diagram as it will make the diagram too cluttered. A loading feedback message will be sent to the console to inform the user that something is taking longer to load.

5. Component Design

5.1. Class Diagrams

5.1.1. Navigation/Main Controller Architecture

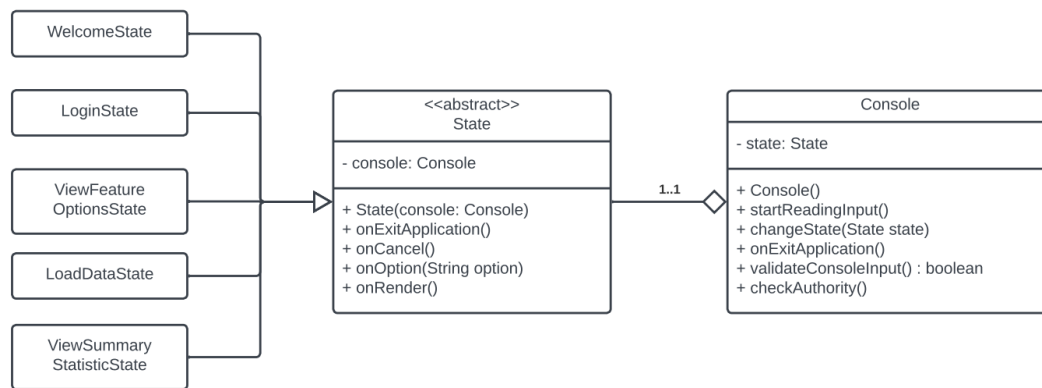


Figure 5.1.: Class Diagram for modelling the Navigation/Main Controller architecture

This class diagram illustrates the core functionality to handle the user navigation of the software and acts as the main controller structure for the various components that make up this software. The reasons that support this design are outlined below.

Please note that this diagram implements the 'State' design pattern. More information can be found at [refactoring.guru.com](https://refactoring.guru.com/design-patterns/state)¹.

- When the software is started, the `WelcomeState` is initiated.
- Recommend that users be able to navigate between the various states whether that is to navigate forward in the process flow by providing input to the console

¹<https://refactoring.guru/design-patterns/state>

prompts, backward where an option to go back is possible, or exit the application. Users should not be 'trapped' in screens.

- A user should only be able to access the login process flow if they are not already Encost-verified. Similarly, only users with the user type set to Encost-verified should be able to access the restricted feature options. This is a user navigability enhancement.
- The state design pattern enables the modularisation of the various features and states that the application can be in and allows for flexibility and customisation of functionality. For example, at the LoginState it is possible to utilise the Java `console.readPassword()` method which masks sensitive information input to the console prompt. This method can be implemented just while the LoginState is active and revert back to the standard Java `console.readLine()` method when transitioned into a different state.
- Common functionality such as exiting the application is centralised on the Console class and called from any of the stateful classes, thus eliminating duplication of code. Similarly, the `validateConsoleInput()` function can be utilised across all state classes to sanitise any console input. The `checkAuthority()` function is used for checking User roles to determine access privileges when different states are transitioned to.
- The `onRender()` method can be configured to print custom messages when the class is first rendered and enables flexibility in the different messaging requirements. Messages can be customised and tailored to the state that the program is in or events that occur within each state. It also enables messaging to be tailored to the user type where you might want to use jargon heavy language for Encost employees, while, for a community user you might want to use more explanatory language. This state structure will enable this flexibility.
- The `onOption()` method takes in the console input from the user. This method is the entry point for checking user input and executing different code branching based on what option has been provided by the user.

5.1.2. User, User Roles, and Features

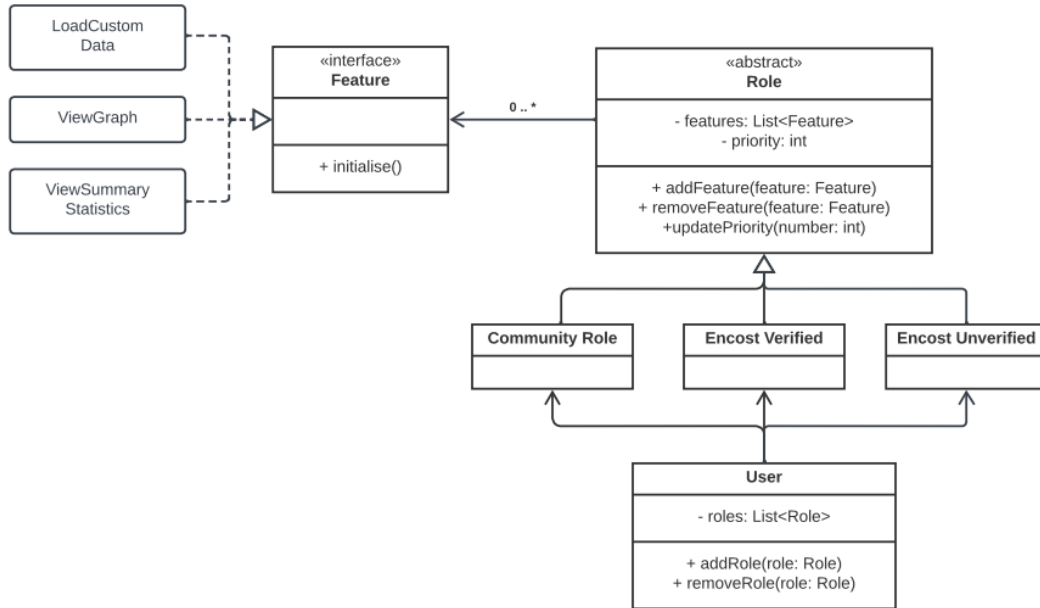


Figure 5.2.: Class Diagram for User, User Roles, and Features

This Class Diagram shows the suggested structure for Users, Roles, and Features. It is recommended to use a role and feature based approach. This enables more flexibility and decoupling of the design components and supports complex role structures where the management of more elaborate access privileges is desirable. Further details are outlined below.

- When the software is started a User object is created with the default role assigned. This is recommended to ensure that every user entering the software has a default role with least privileges. The default role in this case is the 'community' role but another role can easily be configured.
- A user can have 1 or many roles. By default, all users have the 'community' role. The above proposed structure, illustrated in the class diagram, reinforces the concept that roles can be added or removed to a user and is important for enforcing access restrictions for certain features. It also allows for the client to add or update roles over time and/or configure features associated with each role. If new features are added or removed they can be added or removed from a role.
- Centralising the user and the roles like this in one place allows for a users role to be checked from elsewhere in the code-base. It also enables maintenance in a single

place.

- A priority for a role allows for a hierarchical approach where Roles that have a higher priority value supersede those lower in the hierarchy. For example, if a user has multiple roles, then the roles are evaluated from the highest priority first, cascading down to the lowest priority. Generally, roles that are higher in the priority list will naturally inherit the features beneath them, but not in all cases. By using this separation of concerns it enables greater flexibility for the client should they wish to implement such functionality.
- An individual role can have 0 or many Features. Features correspond to the software features such as viewing a graph, loading custom data, or viewing summary statistics. The existence of a Feature in the Role signals that the user who has this Role is able to access this feature. If the feature does not exist, then the Role does not have access to it.
- The 3 main features (load custom data, view graph, view summary statistics) of the software are illustrated in the diagram. Decoupling the features like this enables them to be maintained easily if the functionality ever changes. It also means that if new features are added they can be inserted into the structure. The core functionality for the various features will be implemented within the individual class.

5.1.3. Login and Authentication

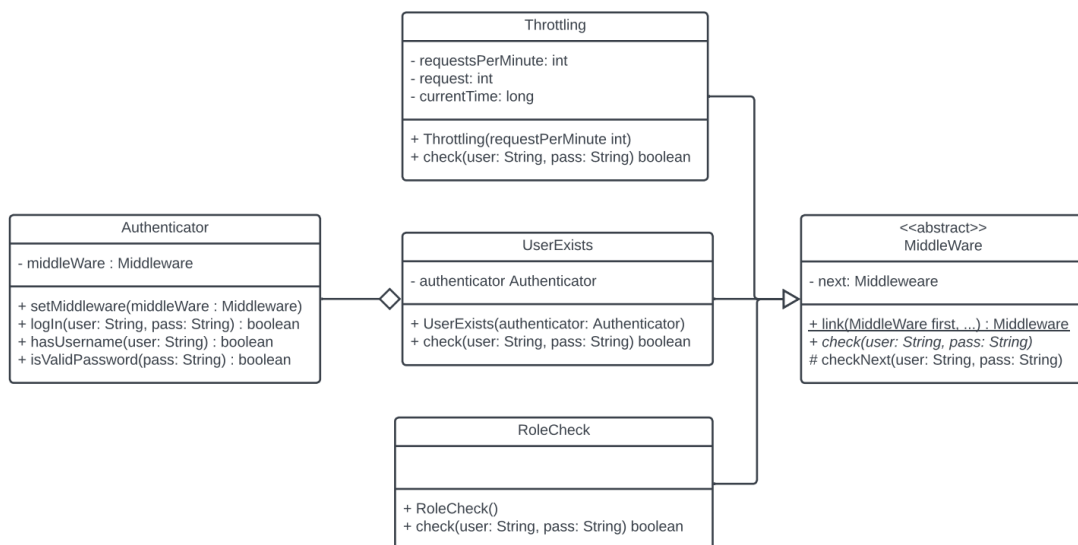


Figure 5.3.: Class Diagram for handling Login and Authentication

This Class Diagram shows the suggested structure for handling the Login flow and authentication of Encost users. Reasons for this recommended design are outlined below.

Please note that this diagram implements the 'Chain of Responsibility' design pattern. More information can be found at refactoring.guru.com².

- This design pattern enables requests to be passed along a sequential chain of handlers or middleware. This is ideal for the login flow which requires the sequential processing of the username, password, and resulting authentication.
- The Throttling middleware is responsible for minimising brute force attacks and will timeout the user if the threshold of attempts is reached within a specified timeframe. The timeout will expire after a defined time period has elapsed allowing the user to continue with more attempts. The enforcement of the timeout will be handled at the state component level.
- The UserExists middleware is primarily the code for handling the authentication of user credentials which is handled through a separate authenticator class. The authenticator class should implement the SHA-256 algorithm for checking user input against the encrypted credentials saved in the system directory.
- A final role check is added at the end of the chain to perform any special role based functionality, but this can be configured or removed if desirable.
- Users should have the option to escape from entering login and password credentials which will divert the user back to the Welcome state. This is so that they don't get trapped in an option that they entered into in error and are forced to complete the inputs in order to get out of it.

²<https://refactoring.guru/design-patterns/chain-of-responsibility>

5.1.4. Graphing and handling Device Data

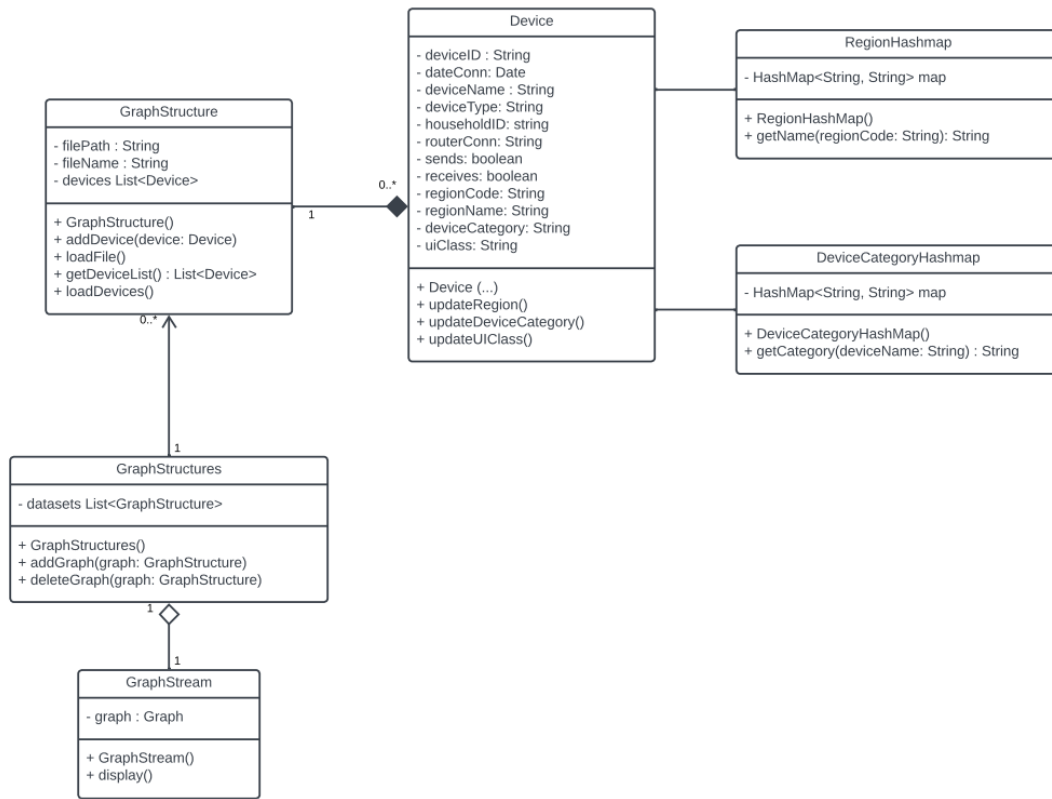


Figure 5.4.: Class Diagram for handling Graphing and Device data

This Class Diagram shows the suggested structure for the graphing functionality and handling of device data.

- The Device class attributes are taken from the sample dataset in the Software Requirements Document. Four additional attributes are derived from the data and saved to the Device class object:
 - `regionCode` - to record the 3 digit region code for the device.
 - `regionName` - the name of the region.
 - `deviceCategory` - the device category for a device name.
 - `uiClass` - this attribute is the string definition for the `GraphStream` class. It defines the colours and sprites attached to the device.
- The device category is derived from a hash map data structure that maps the 5 device categories to the device names. A hash map is used because it enables direct

lookup and will speed up the parsing of the data set. The values for initialising the hash map are hardcoded into the constructor. If these categories and devices change frequently, then a better approach would be to store them in an external file and read the file in as necessary. However, the impact of doing this on the end user and the frequency of updates should be considered.

- There are 17 regions according to the ISO-3166-2:NZ³ standard and it is very unlikely that these will change. For this reason, they are written into the code. The data for these regions is stored in a hashmap. Region names are referenced using direct lookup by the 3 digit code. The 3 digit code can be extracted from the 'Household-ID' field using a string split function. The reason for storing the name is because the user is more likely to know the names rather than the 3 digit codes when looking at the summary statistics.
- A device holds the configuration settings (sprites and colours) which will be passed to the GraphStream object for generating nodes and edges. This will enable distinguishing between different types of nodes. This has the added benefit of providing a more engaging experience for the users as they will be able to identify individual nodes in the graph structure easily. To provide more context, a light bulb sprite can be used to represent a lighting device and colours can be used to distinguish the category that the device belongs to. Mockups are provided in the screenshots section of this document.
- The GraphStructure object holds the filePath and fileNames of the dataset. When the software is initialised the ESHD will be read in and loaded and saved to the list situated in the GraphStructures class. If a custom file is loaded, then the filepath and filename for this file will be created and stored in the list also. This structure enables functionality to save multiple custom datasets. It also enables for checking whether the limit of 10 files or the 50MB allowable size has been reached.
- The GraphStream class is primarily for initiating and displaying the GraphStream object and its associated methods and features. It is separated into its own class to keep it decoupled from the rest of the software architecture. If in future the client wishes to change to a different graph library or implement more GraphStream product features, then the decoupling will minimise the impact on the overall system and means testing can be carried out in isolation. It also enables the configuring of graph settings such as background color and window size. If there is a desire to provide functionality for the user to configure their own settings whether they want to enable/disable a light or dark mode, then this decoupling will support this type of functionality. The GraphStream object will reference the stylesheet file saved in a designated directory within the system. Style settings can be easily reconfigured, added, or removed depending on the client or end user requirements.

³https://en.wikipedia.org/wiki/ISO_3166-2:NZ

5.2. Sequence Diagrams

5.2.1. View Graph

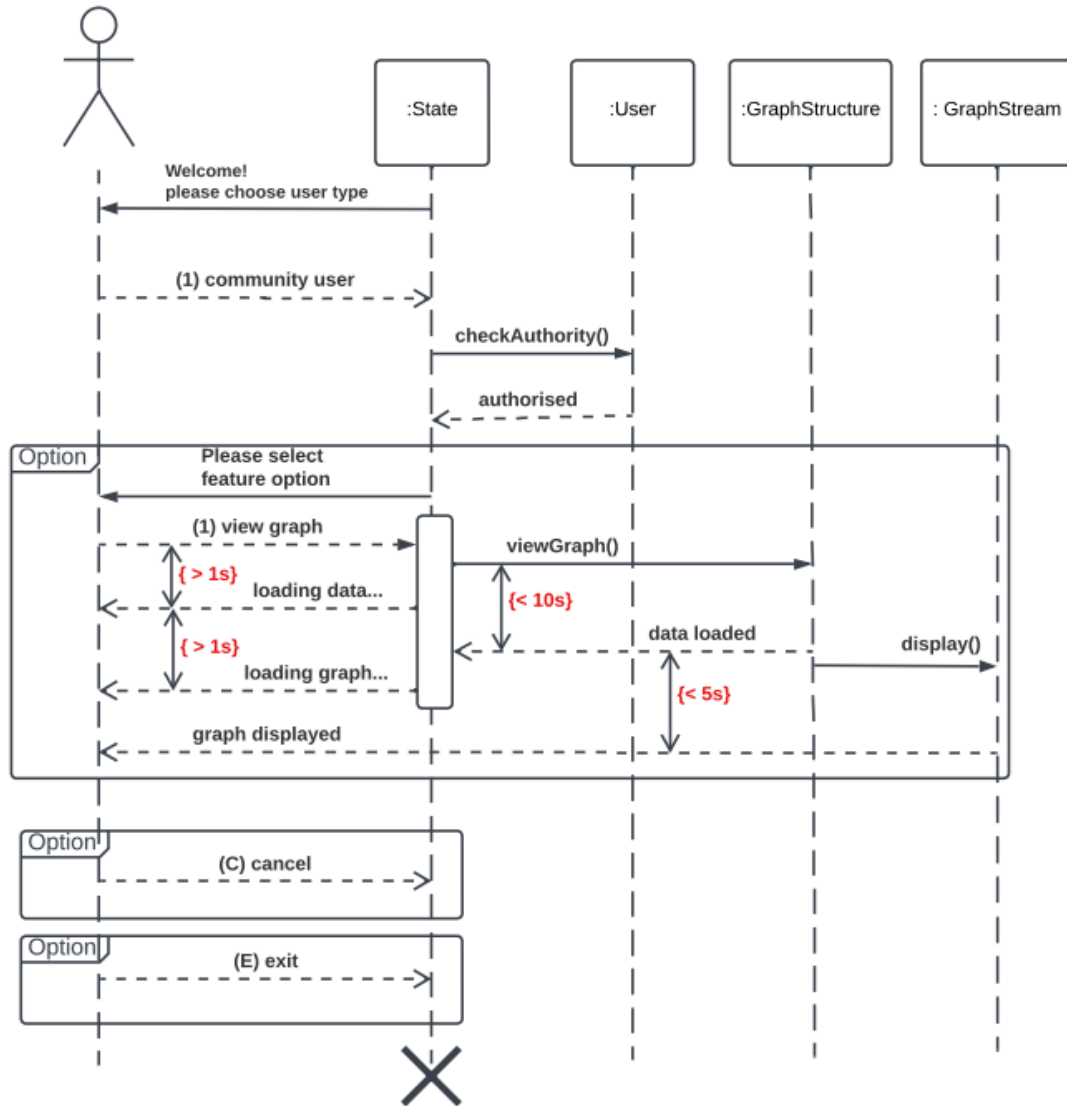


Figure 5.5.: Sequence diagram for viewing a graph

This Sequence Diagram shows message flows for viewing a graph along with the performance timing constraints as per the non-functional requirements.

- If a delay of more than 1second is experienced either with loading a dataset or

with loading a graph then a message should be sent to the console to let the user know that the information is loading.

- Data should take no more than 10seconds to load and a graph should take no longer than 5seconds to load.
- The steps to view a graph are the same for an Encost user.

5.2.2. Login

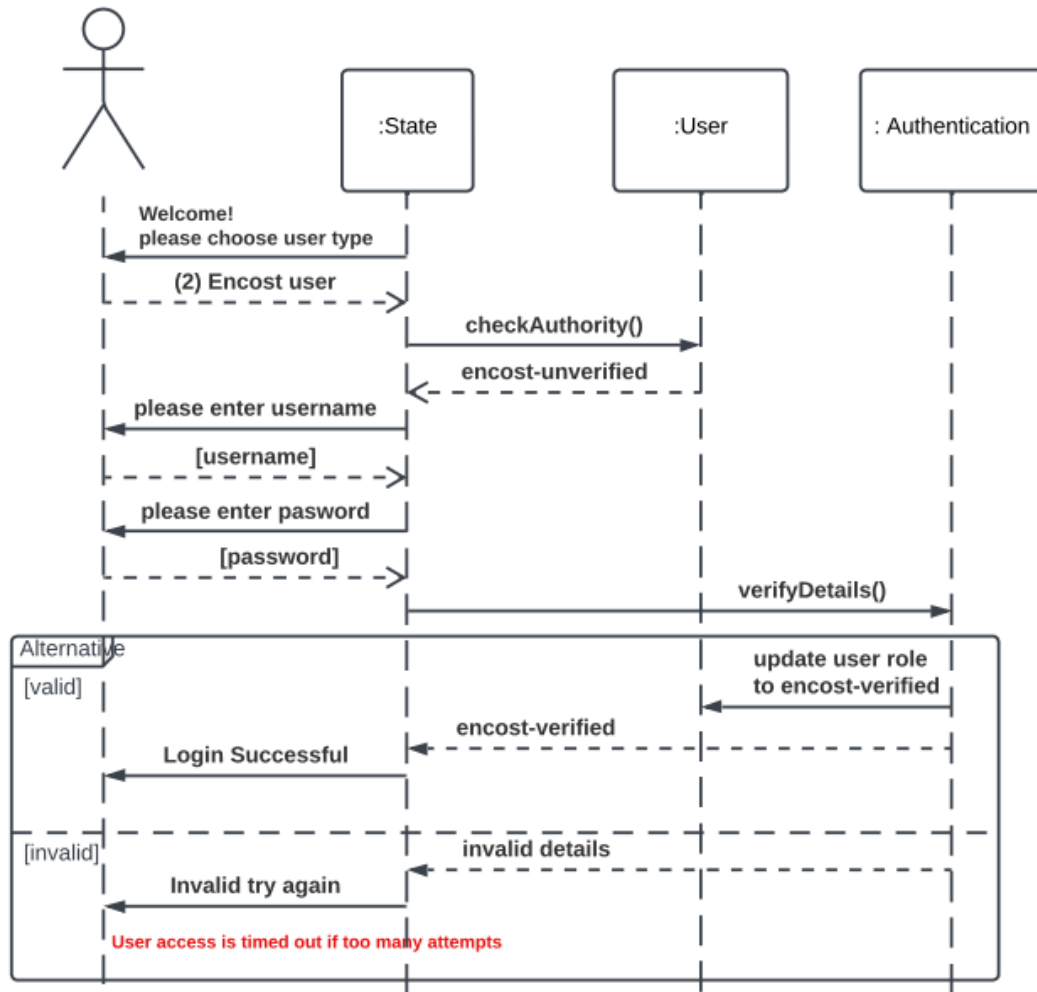


Figure 5.6.: Sequence diagram for the login process

5.2.3. Loading a custom dataset

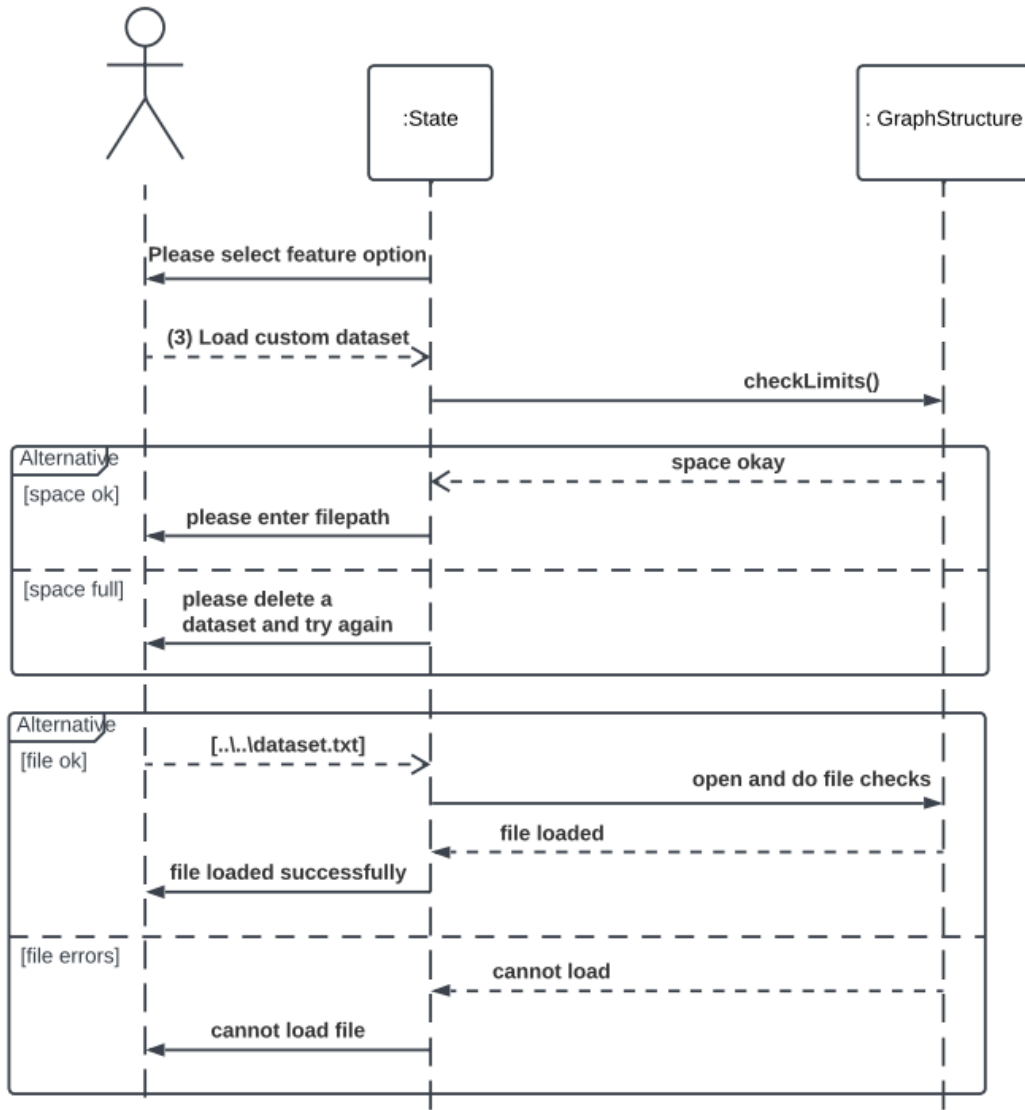


Figure 5.7.: Sequence diagram for loading a custom dataset

- A check is made to ensure that the user has not reached the maximum of 10 files or 50MB space. If they have reached the limit, then they are informed to delete a dataset to make room. The reason for this is to prevent the system being overloaded and crashing.
- If a file cannot be loaded, then an informative message is sent to the user to advise them of what went wrong. This is so that they can troubleshoot and rectify the

error quicker.

5.2.4. Calculating Summary Statistics

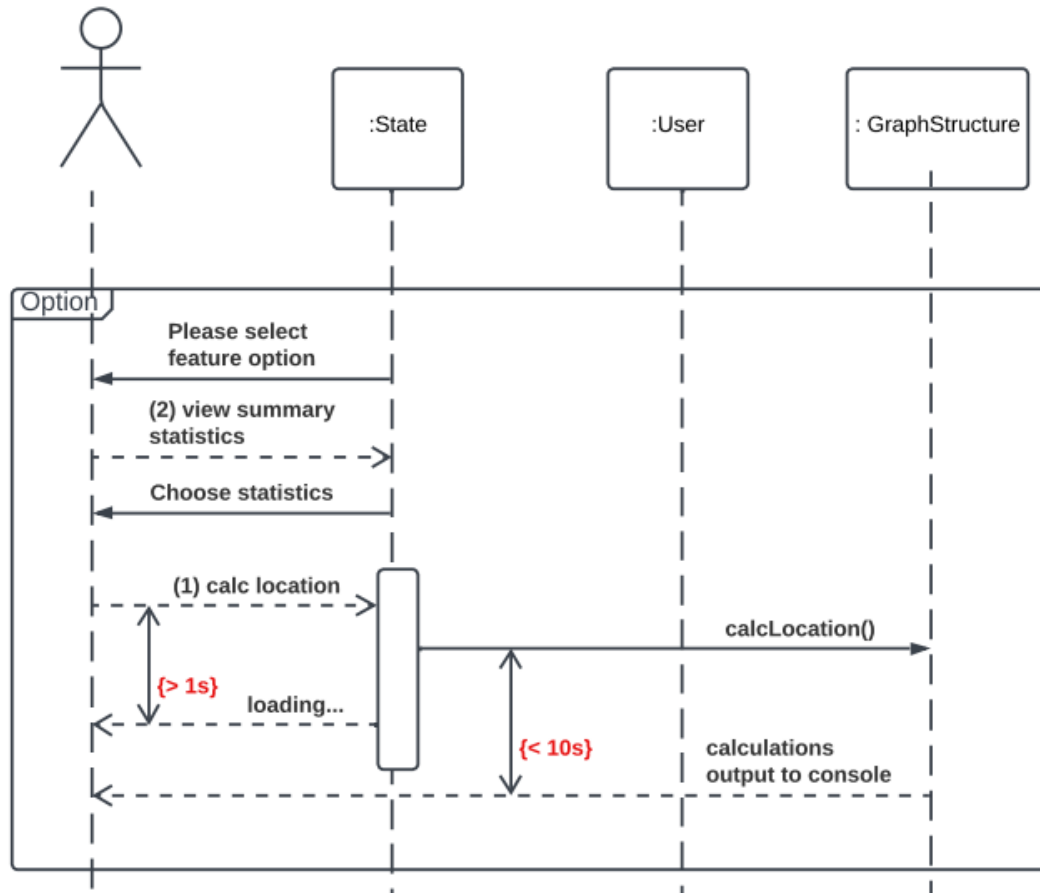


Figure 5.8.: Sequence diagram for calculating summary statistics

- The method and timings for the other 2 reports that aren't shown on this diagram are the same.

6. Screenshots

6.1. Console Prompts

6.1.1. Standard Input/Output console prompts

These screenshots represent the messaging that should be output to the console.

Welcome

```
Welcome!  
Please select a user from the below list or exit the application:  
(1) Community  
(2) Encost  
(E) Exit Application
```

Figure 6.1.: Mockup for the Welcome screen.

Community User Feature Options

```
Community User Feature Options  
Please select an option from the below list, exit or cancel:  
(1) View Graph  
(C) Cancel  
(E) Exit Application
```

Figure 6.2.: Mockup for the Community User Feature Options section.

Encost Unverified User Username prompt

```
Encost Un-Verified User Login  
Please enter your username or esc:
```

Figure 6.3.: Mockup for the Encost Unverified User Username prompt

Encost Unverified User Password prompt

```
Encost Un-Verified User Login
Please enter your password or esc:
```

Figure 6.4.: Mockup for the Encost Unverified User Password prompt

Encost Unverified Successful Login

```
Successful Login! redirecting you to Feature Options
```

Figure 6.5.: Mockup for the Encost Unverified Successful Login

Encost Unverified Invalid Credentials

```
Successful Login! redirecting you to Feature Options
The username and/or password that you have entered is not correct or is not found. Please try again.
```

Figure 6.6.: Mockup for the Encost Unverified Invalid Credentials

Encost Unverified Too Many Login Attempts

```
You have exceeded the amount of login attempts. Please wait a few minutes and try again.
```

Figure 6.7.: Mockup for the Encost Unverified Too Many Login Attempts

Encost Verified User Feature Options

```
Encost Verfied User Feature Options
Please select a feature option from the below list, exit or cancel:
(1) View Graph
(2) View Summary Statistics
(3) Load Custom Dataset
(C) Cancel
(E) Exit Application
```

Figure 6.8.: Mockup for the Encost Verified User Feature Options

Encost Verified User Summary Statistics Options

```
Encost Verified User Summary Statistic Options
Please select a summary statistic option from the below list, exit or cancel:
(1) Calculate Device Location
(2) Calculate Device Distribution
(3) Calculate Device Connectivity
(C) Cancel
(E) Exit Application
```

Figure 6.9.: Mockup for the Encost User Summary Statistics Options

Encost Verified User Filepath Upload

```
Encost Verified User Load Custom Dataset
Please enter the full filepath for the file, exit or cancel:
(C) Cancel
(E) Exit Application
```

Figure 6.10.: Mockup for the Encost User Filepath Upload

Encost Verified User Filepath upload errors

```
Unable to locate the specified file.
Cannot load the specified file. Please check the format.
File too large. Cannot load.
File limit of 10 reached. Please delete a file and retry.
File space limit of 50MB reached. Please delete a file and retry.
```

Figure 6.11.: Mockup for the Encost Verified User Filepath upload errors

Encost Verified User choose a filepath to view graph or calculate summary statistics

```
Encost Verified User View Graph
Please select a dataset from the below list, exit or cancel:
(1) Encost Smart Homes Dataset
(2) mycustomdataset.txt
(3) anotherdataset.txt
(4) yetanotherdataset.txt
(5) profitablecustomers.txt
(6) specialdataset.txt
(C) Cancel
(E) Exit Application
```

Figure 6.12.: Mockup for the Encost Verified User choose a filepath to view graph or calculate summary statistics

Encost Verified User Delete a dataset

```
Encost Verified User Delete Custom Dataset
Please select a dataset from the below list to delete, exit or cancel:
(1) mycustomdataset.txt
(2) anotherdataset.txt
(3) yetanotherdataset.txt
(4) profitablecustomers.txt
(5) specialdataset.txt
(C) Cancel
(E) Exit Application
```

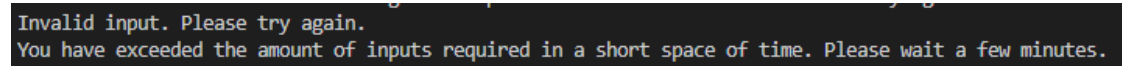
Figure 6.13.: Mockup for the Encost Verified User Delete a dataset

Encost Verified User Exit application

```
You have selected exit. All data will be lost. Are you sure?
(N) No
(Y) Yes
```

Figure 6.14.: Mockup for the Encost Verified User Exit application

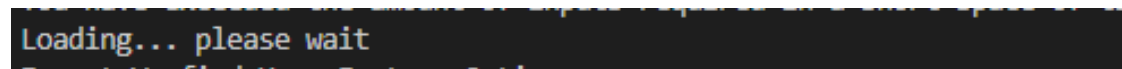
Invalid Inputs

A black rectangular box with white text. The text reads: "Invalid input. Please try again." followed by "You have exceeded the amount of inputs required in a short space of time. Please wait a few minutes." on the next line.

Invalid input. Please try again.
You have exceeded the amount of inputs required in a short space of time. Please wait a few minutes.

Figure 6.15.: Mockup for the Invalid Inputs

Loading message for when it takes more than 1second to load

A black rectangular box with white text. The text reads: "Loading... please wait".

Loading... please wait

Figure 6.16.: Mockup for Loading message for when it takes more than 1second to load

6.2. Calculate Device Summary Statistics

Note: the numbers in these screenshot are made-up. They are not real. Where applicable, the combining of reports is to enhance the user experience and to simplify report output whilst adhering to requirements.

Calculate Device Distribution Summary Statistics

Note: This report combines 2 requirements.

```

-----
Device Distribution - number of devices by device category and device type
Filename: encost_dataset.txt
-----

```

DEVICE CATEGORY	DEVICE TYPE	COUNT
Wifi Router	Router	45
Wifi Router	Extender	32
TOTAL	-	77
Hub/Controller	Hub/Controller	64
Smart Lighting	Light Bulb	236
Smart Lighting	Strip Lighting	16
Smart Lighting	Other Lighting	16
TOTAL	-	268
Smart Appliance	Kettle	08
Smart Appliance	Toaster	28
Smart Appliance	Coffee maker	12
TOTAL	-	48
Smart Whiteware	Washing Machine	16
Smart Whiteware	Refrigerator	24
Smart Whiteware	Dishwasher	56
TOTAL	-	96

```

-----

```

Figure 6.17.: Mockup screen for device distribution summary statistics

6.3. Calculate Device Location Summary Statistics

6.3.1. Number of households, devices, and average devices per household in each region

Note: This report combines 3 requirements.

```

-----
Device Location - average number of devices per household by region
Filename: encost_dataset.txt
-----

```

REGION	NUMBER HOUSEHOLDS	NUMBER DEVICES	AVG DEVICES/HOUSEHOLD
AUK Auckland	22	262	11.91
BOP Bay of Plenty	4	48	12
CAN Canterbury	6	22	3.66
CIT Chatham Islands	2	11	5.5
GIS Gisborne	0	0	0
WGN Greater Wellington	15	189	12.6
HKB Hawke's Bay	0	0	0
MWT Manawatū-Whanganui	2	22	11
MBH Marlborough	8	64	8
NSN Nelson	1	8	8
NTL Northland	3	56	18.66
OTA Otago	6	28	4.66
STL Southland	0	0	0
TKI Taranaki	0	0	0
TAS Tasman	10	29	2.9
WKO Waikato	18	98	5.4
WTC West Coast	2	6	3

```

-----

```

Figure 6.18.: Mockup screen for Number of households, devices, and average devices per household in each regions

6.3.2. Number of devices, number of households, and average number of devices per household by category by region

Note: This report combines 2 requirements.

```

-----
Device Location - average number of devices for each category by household, by region
Filename: encost_dataset.txt
-----

```

REGION	DEVICE CATEGORY	NUM. HOUSEHOLDS	NUM. DEVICES	AVG DEVICES/HOUSEHOLD
AUK Auckland	Wifi Router	22	22	1
AUK Auckland	Hub/Controller	22	22	1
AUK Auckland	Smart Lighting	22	72	3.27
AUK Auckland	Smart Appliance	22	85	3.86
AUK Auckland	Smart Whiteware	22	120	5.45
BOP Bay of Plenty	Wifi Router	4	4	1
BOP Bay of Plenty	Hub/Controller	4	4	1
BOP Bay of Plenty	Smart Lighting	2	12	6
BOP Bay of Plenty	Smart Appliance	1	2	2
BOP Bay of Plenty	Smart Whiteware	3	4	1.3
CAN Canterbury	Wifi Router	6	6	1
CAN Canterbury	Hub/Controller	6	6	1
CAN Canterbury	Smart Lighting	5	28	5.6
CAN Canterbury	Smart Appliance	2	2	1
CAN Canterbury	Smart Whiteware	0	0	0
CIT Chatham Islands	Wifi Router	2	2	1
CIT Chatham Islands	Hub/Controller	2	2	1
CIT Chatham Islands	Smart Lighting	2	6	3
CIT Chatham Islands	Smart Appliance	2	4	2
CIT Chatham Islands	Smart Whiteware	0	0	0
GIS Gisborne	Wifi Router	0	0	0
GIS Gisborne	Hub/Controller	0	0	0
GIS Gisborne	Smart Lighting	0	0	0
GIS Gisborne	Smart Appliance	0	0	0
GIS Gisborne	Smart Whiteware	0	0	0
WGN Greater Wellington	Wifi Router	15	15	1
WGN Greater Wellington	Hub/Controller	15	15	1
WGN Greater Wellington	Smart Lighting	12	98	8.1
WGN Greater Wellington	Smart Appliance	14	48	3.4
HKB Hawke's Bay	Wifi Router	0	0	0
HKB Hawke's Bay	Hub/Controller	0	0	0
HKB Hawke's Bay	Smart Lighting	0	0	0
HKB Hawke's Bay	Smart Appliance	0	0	0
...				
...				
...				

Figure 6.19.: Mockup screen for Number of devices, number of households, and average number of devices per household by category by region

6.4. Calculate Device Connectivity Summary Statistics

6.4.1. Average, minimum, and maximum number of devices that Encost Router is connected to

Note: This report combines 2 requirements.


```

-----
Device Connectivity - average number, minimum, and maximum number of devices that an Encost Router is connected to
Filename: encost_dataset.txt
-----

```

DEVICE NAME	AVG NUMBER OF DEVICES	MINIMUM	MAXIMUM
Encost Router 360	16	6	24
Encost Router Plus	8.4	4	22
ENCOST WIFI RANGE EXTENDER 1.0	6.1	8	16
Encost Wifi Range Extender 2.0	5.9	2	11

```

-----

```

Figure 6.20.: Mockup screen for Average, minimum, and maximum number of devices that Encost Router is connected to

6.4.2. Average, minimum, and maximum number of Encost Hub/Controllers that a device is receiving commands from

Note: This report combines 2 requirements.

```

-----
Device Connectivity - average number, minimum, and maximum number of Encost Hub/Controllers that a device is receiving commands FROM
Filename: encost_dataset.txt
-----

```

DEVICE NAME	AVG NUMBER HUB/CONTROLLERS
Encost Router 360	1.3
Encost Router Plus	1
ENCOST WIFI RANGE EXTENDER	0
Encost Wifi Range Extender 2.0	1
Encost Smart Hub	1.2
Encost Smart Hub 2.0	2.3
Encost Smart Hub Mini	1
Encost Smart Bulb B22 (white)	1.4
Encost Smart Bulb B22 (multi colour)	1
Encost Smart Bulb E26 (white)	0
Encost Smart Bulb E26 (multi colour)	0
Encost Strip Lighting (white)	1
Encost Strip Lighting (multi colour)	1
...	
...	
...	

```

-----

```

Figure 6.21.: Mockup screen for Average, minimum, and maximum number of Encost Hub/Controllers that a device is receiving commands from

6.4.3. Average, minimum, and maximum number of devices that an Encost Hub/Controller is sending commands to

Note: This report combines 2 requirements.

```
-----
Device Connectivity - average number, minimum, and maximum number of devices that an Encost Hub/Controllers is sending commands TO
Filename: encost_dataset.txt
-----
```

DEVICE NAME	AVG NUMBER DEVICES
Encost Smart Hub	15.6
Encost Smart Hub 2.0	12.2
ENCOST SMART HUB MINI	6.1

```
-----
```

Figure 6.22.: Mockup screen for Average, minimum, and maximum number of devices that an Encost Hub/Controller is sending commands to

6.5. Mockup GraphStream Nodes

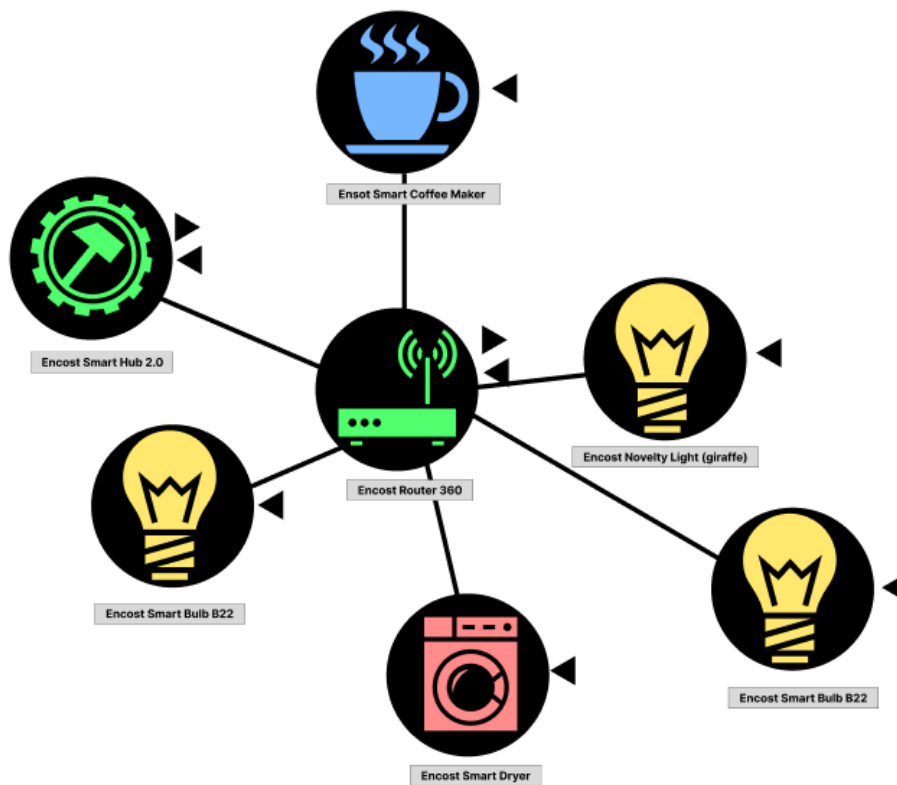


Figure 6.23.: Mockup of a simple household with few devices

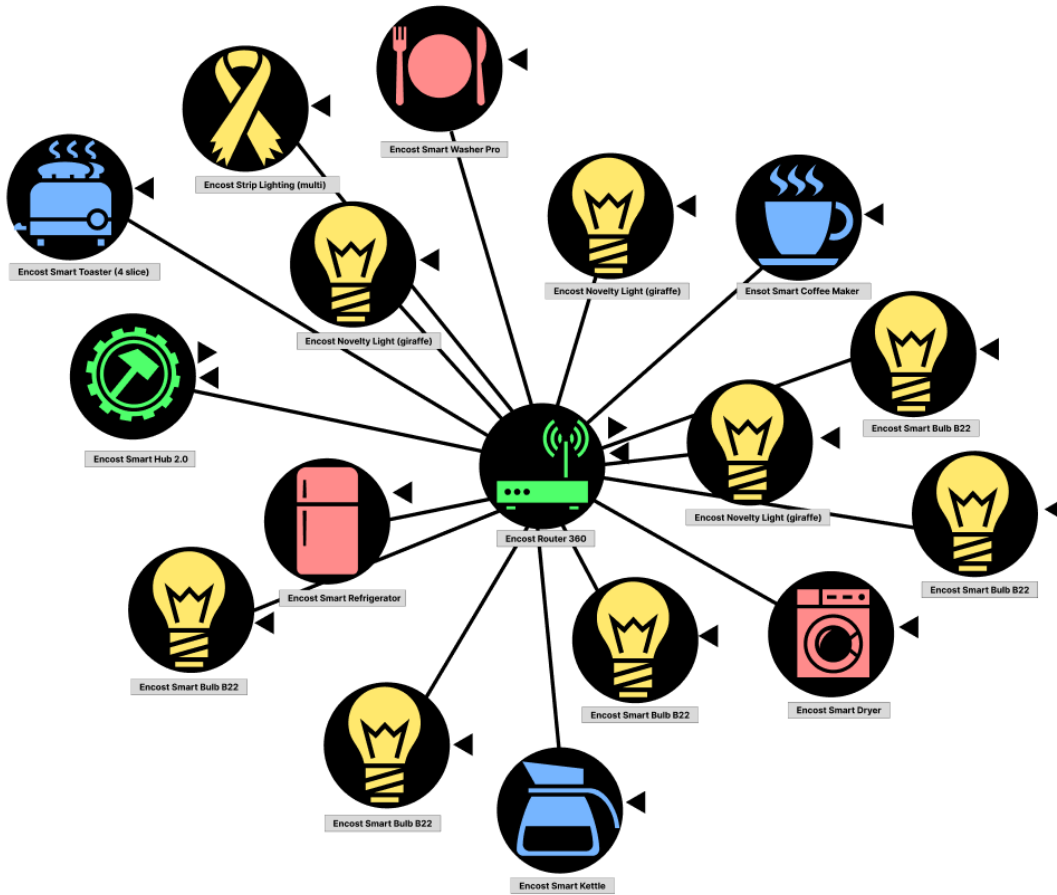


Figure 6.24.: Mockup of complex household with many devices

These screenshots show mockups of some GraphStream node and edge structures, made using a design tool, Figma¹ and SVG icons obtained from game-icons.net². Icon attributions are listed at the end of this document in Appendix A. The mockups are for illustrative purposes only to demonstrate the capabilities of the GraphStream functionality. More information regarding GraphStream functionality is available on their website graphstream-project.org³.

- Icons are used to distinguish between the different device types, while colours are used to distinguish between the different categories. Yellow for lighting, blue for appliances, red for whiteware, and green for routers and controllers. It is possible to use actual product photography instead of SVG icons or more brandcentric visuals

¹<https://www.figma.com/>

²<https://game-icons.net/>

³<https://graphstream-project.org/>

based on the client and end user requirements. It is possible to configure the colour scheme also. Icons correspond to sprites in the GraphStream terminology.

- Inward facing arrows are used to indicate that the device is capable of receiving communication, while outward facing arrows are used to indicate that the device is capable of sending communication. More appropriate iconography can be used otherwise based on the client requirements. These icons correspond to sprites in the GraphStream terminology.
- Nodes are labelled with the device name. Device IDs, dates, or other attributes can be used also. Textual labelling corresponds to the `ui.label` GraphStream terminology.

Final Note: Even though the GraphStream styling 'mimics' the Cascading Style Sheet (CSS) infrastructure which is commonly utilised in website design, the library does not implement any animation or more sophisticated features associated with CSS. Therefore, it is important to point out that there are limitations on what visual effects can be achieved through this library. The extent of these capabilities are outlined in the extensive API documentation ⁴.

⁴<https://graphstream-project.org/doc/API/>

7. Conclusion

This software design document implements an object-oriented design complete with software design patterns and typical object-oriented programming techniques. The GraphStream Java Library has been leveraged to fulfil the visual requirements of the software with a decoupled approach so as to separate this external library from the core functionality of the software. Recommended approaches to styling have been made with users in mind and within the bounds of the GraphStream product.

Section 3 regarding User Interface Requirements are implemented in this design utilising the programming design pattern, 'state machine'. A user role structure is implemented and a separate login flow that implements the programming design pattern, 'chain-of-responsibility'. This approach provides flexibility and a finer degree of control over access privileges while achieving the requirements on user categorisation and ESGP account login.

A graph structure and device class are implemented and utilise hashmaps to speed up aspects of data loading. This structure ensures that the functional requirements for loading data, categorising device data, calculating summary statistics, and building and visualising a graph are achieved. Allowing users to upload more than 1 custom dataset in a session is recommended and the required implementation details to achieve this have been outlined and described.

In terms of the nonfunctional requirements, providing feedback to users for delays has been implemented. Good coding practices should be followed to ensure that information is loaded well within the 5 and 10 second thresholds. Security features have been implemented including throttling of brute force attacks during login, cleansing and checking user input, throttling of spamming input to the console, recommending the use of the SHA-256 algorithm for encryption, using a user and role structure to enforce access privileges, and using the `console.readpassword()` method to mask out sensitive console input. All of these measures are utilised with the aim of ensuring system stability, protecting information, and protecting the integrity of the system. To support software quality clear and concise language has been used throughout for console output and error checking and messaging has been implemented to provide informative feedback. Cancel and Exit functionality has been implemented to prevent users being trapped in screens.

A. Appendix GraphStream Mockup SVG Icon Attributions

Use of the icons are licensed under Creative Commons 3.0¹.

- Light Bulb icon by Lorc <https://lorcblog.blogspot.com/> sourced from <https://game-icons.net/1x1/lorc/light-bulb.html>.
- Coffee Cup icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/coffee-cup.html>.
- Toaster icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/toaster.html>.
- Fridge icon by Caro Asercion <https://game-icons.net/> sourced from <https://game-icons.net/1x1/caro-asercion/fridge.html>.
- Meal icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/meal.html>.
- Ribbon icon by Lorc <https://lorcblog.blogspot.com/> sourced from <https://game-icons.net/1x1/lorc/ribbon.html>.
- Coffee Pot icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/coffee-pot.html>.
- Wifi Router icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/wifi-router.html>.
- Gear Hammer icon by Lorc <https://lorcblog.blogspot.com/> sourced from <https://game-icons.net/1x1/lorc/gear-hammer.html>.
- Washing Machine icon by Delapouite <https://delapouite.com/> sourced from <https://game-icons.net/1x1/delapouite/washing-machine.html>.

¹<https://creativecommons.org/licenses/by/3.0/>