SOFTWARE MAINTENANCE DOCUMENT

for

Encost Smart Graph Project

Version 1.0

Prepared by: Joey Han
SoftFlux Engineer

SoftFlux

May 29, 2023

Contents

Revision History

Name	Date	Reason for Changes	Version
SoftFlux	29/05/2023	Software Maintenance and CI/CD Pipeline	1.0

1 Introduction/Purpose

1.1 Purpose

This document is Software Maintenance Document for the software Encost Smart Graph Project (ESGP) for the company Encost. The purpose of this document is to provide a way to show software maintenance of the ESGP, CI/CD pipeline and to track what maintenance have been done already.

1.2 Document Conventions

This document uses the following conventions:

- ESGP: Encost Smart Graph Project
- SDPD: Software Development Planning Document
- FSTP: Functional Software Test Plan
- SDS: Software Design Specification

1.3 Intended Audience and Reading Suggestions

This document is intended for any software maintainer or DevOps engineer. Here are the potential uses for each of the reader types:

- Software Maintainer and DevOps Engineer: Uses this document to track the progress and to see what has been newly implemented.

1.4 Project Scope

The ESGP is a software system with aims to enable the Encost smart devices to be visualised in a graph data structure from the command line. With both accessibility to Encost verified users and community users but with restricted access to the software capability. There will be no hardware integration required for the ESGP nor for the testing purposed of the software. SoftFlux is responsible to the software development and maintenance of the ESGP and the insurance that the software meets the requirements listed in the SDPD 2, FSTP 4 and the SDS 3.

2 Specialized Requirements Specification

- There are no specialized requirements specifications.

3 Maintenance

3.1 Updating GraphStream to latest version

3.1.1 User Story

- The client (Encost) has decided that the system should use the most up to-date version of GraphStream.

3.1.2 Problem/modification request

- Update GraphStream to the latest version (V2.0).

3.1.3 Problem/modification analysis

Maintenance Category: Perfective
Impact Analysis

- **Verify the problem:** The system is currently using GraphStream 1.3 which is not the latest version.
- **Implementation options:** Update the GraphStream to version 2.0 and change some of the methods used for GraphStream 1.3.
- **Effort:** 10-15 lines of code including testing code, need to learn some new methods created in the new version which throws away some of the old methods used.
- **Resources:** 2 hours which includes making changes and running all the tests it previously had.

3.1.4 Acceptance/rejection

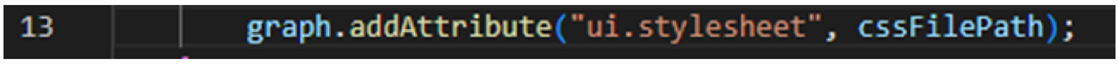
- Accepted.

3.1.5 Modification implementation

Changes made:

- Changed addAttribute method to setAttribute method in the GraphVisualiser java file. Seen in Figure 3.1 and Figure 3.2.

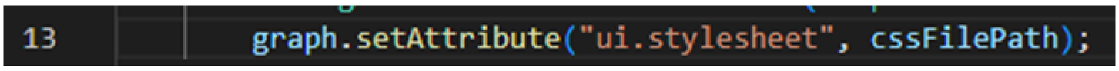
Before in GraphVisualiser java file:



```
13          graph.addAttribute("ui.stylesheet", cssFilePath);
```

Figure 3.1: Line 13 of GraphVisualiser java file using GraphStream version 1.3

After in GraphVisualiser java file:

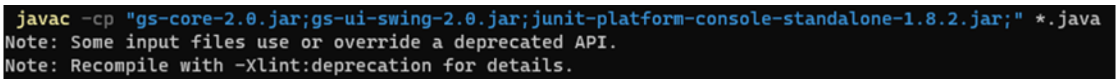


```
13          graph.setAttribute("ui.stylesheet", cssFilePath);
```

Figure 3.2: Line 13 of GraphVisualiser java file using GraphStream version 2.0

3.1.6 Maintenance review/acceptance

Compiling all the java files including GraphVisualiser java file:

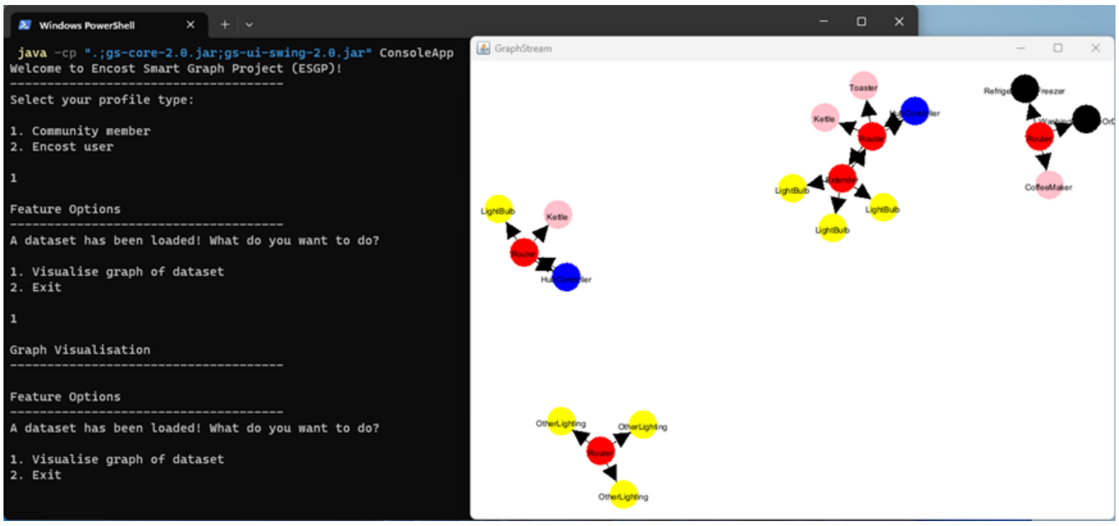


```
 javac -cp "gs-core-2.0.jar;gs-ui-swing-2.0.jar;junit-platform-console-standalone-1.8.2.jar;" *.java
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

Figure 3.3: Working GraphStream version 2.0 and JUnit 1.8.2 jar java compilation

- It shows no errors so that means all the methods used for graph stream is compatible with the latest version.

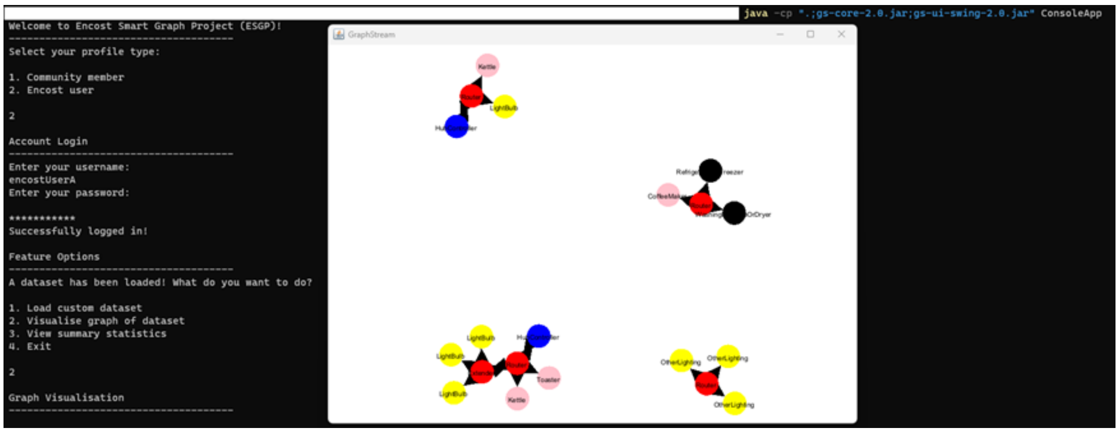
Community user graph visualisation



```
 java -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" ConsoleApp
Welcome to Encost Smart Graph Project (ESGP)!
-------------------------------------
Select your profile type:

1. Community member
2. Encost user

1

Feature Options
-------------------------------------
A dataset has been loaded! What do you want to do?

1. Visualise graph of dataset
2. Exit

1

Graph Visualisation
-------------------------------------

Feature Options
-------------------------------------
A dataset has been loaded! What do you want to do?

1. Visualise graph of dataset
2. Exit
```

Figure 3.4: Community graph visualisation

Encost user graph visualisation:



```
java -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" ConsoleApp
```

```
Welcome to Encost Smart Graph Project (ESGP)!
--------------------------------------
Select your profile type:

1. Community member
2. Encost user

2

Account Login
--------------------------------------
Enter your username:
encostUserA
Enter your password:

***********
Successfully logged in!

Feature Options
--------------------------------------
A dataset has been loaded! What do you want to do?

1. Load custom dataset
2. Visualise graph of dataset
3. View summary statistics
4. Exit

2

Graph Visualisation
--------------------------------------
```

Figure 3.5: Encost-verified graph visualisation

3.2 Calculating and displaying device distribution in string table format

3.2.1 User Story

- The Encost users have noticed that the summary statistics aren't working. They would like to be able to see a textual summary of the device distribution (SRS 4.9 Calculating Device Distribution).

3.2.2 Problem/modification request

- Investigate the summary statistics and add a textual summary of device distribution.

3.2.3 Problem/modification analysis

Maintenance Category: Corrective
Impact Analysis

- **Verify the problem:** The summary statistics is not implemented, and we must also implement textual summary of device distribution.
- **Implementation options:** Make a DataDistributionStatistics class java file which have a method to calculate the device distribution and another method to display it in a textual summary format.

- **Effort:** 200-400 lines of code including testing code, need to learn how to display device distribution in a textual summary format and how to calculate the device distribution.
- **Resources:** 12 hours to 24 hours which includes making changes and making new tests.

3.2.4 Acceptance/rejection

- Rejected, takes too much time and resources to do.

3.3 One household graph visualisation

3.3.1 User Story

- The Community users are having trouble telling which household is theirs in the graph visualisation. They would like the additional feature of being able to see a graph visualisation for their household only.

3.3.2 Problem/modification request

- Add a new feature for community users so that they can see their own household.

3.3.3 Problem/modification analysis

Maintenance Category: Perfective
Impact Analysis

- **Verify the problem:** The system currently only visualises all graphs made using the dataset. The new feature will let the user tell the system their household and it will display a graph with only that household devices.
- **Implementation options:** Create a new method called askUserForHousehold that will ask the user for their household, and it will get the current household devices from the current dataset and display it all in the graph.
- **Effort:** 100-250 lines of code, no new knowledge needed.
- **Resources:** 4-8 hours which includes making changes, running all the tests it previously had and doing manual tests.

3.3.4 Acceptance/rejection

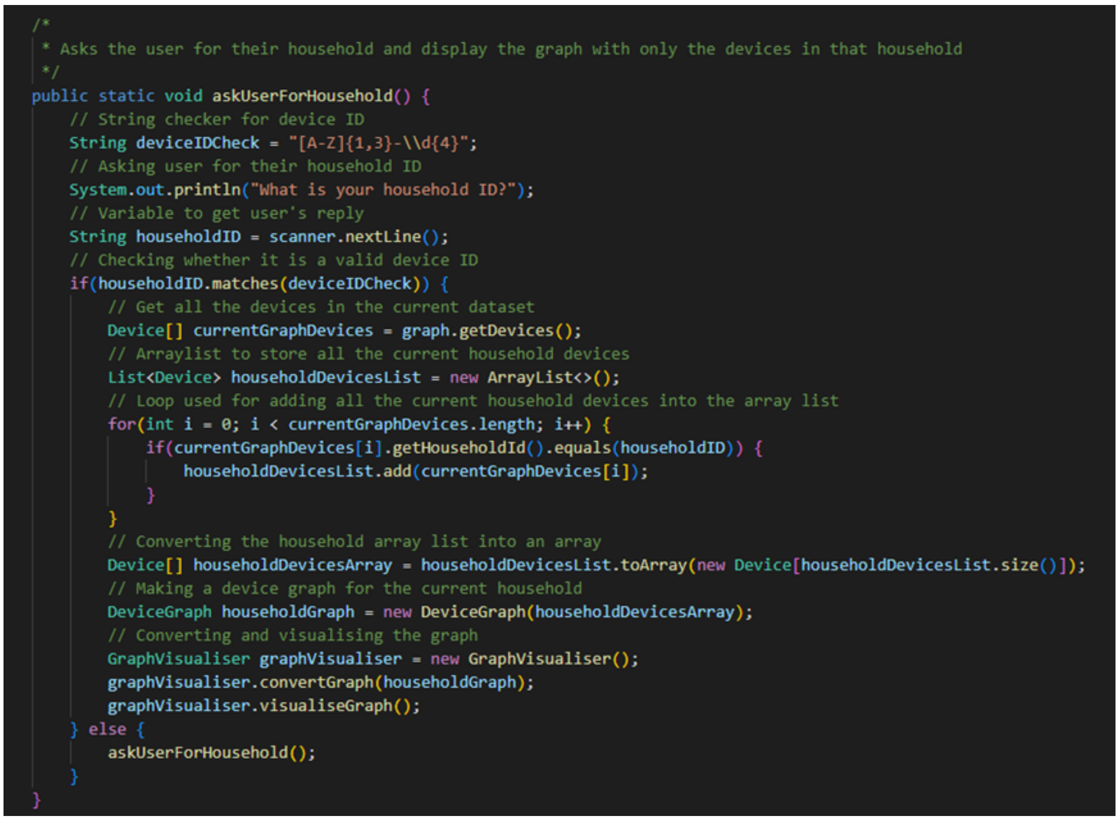
- Accepted.

3.3.5 Modification implementation

Changes made:

- Changed the input "2" to be used for the household graph and made input "3" the exit instead. Seen in Figure 3.7 and Figure 3.8.
- Added the option to visualise the current household and made the exit option number 3 instead of 2 for the optionsCommunity array. Seen in Figure 3.9 and Figure 3.10.

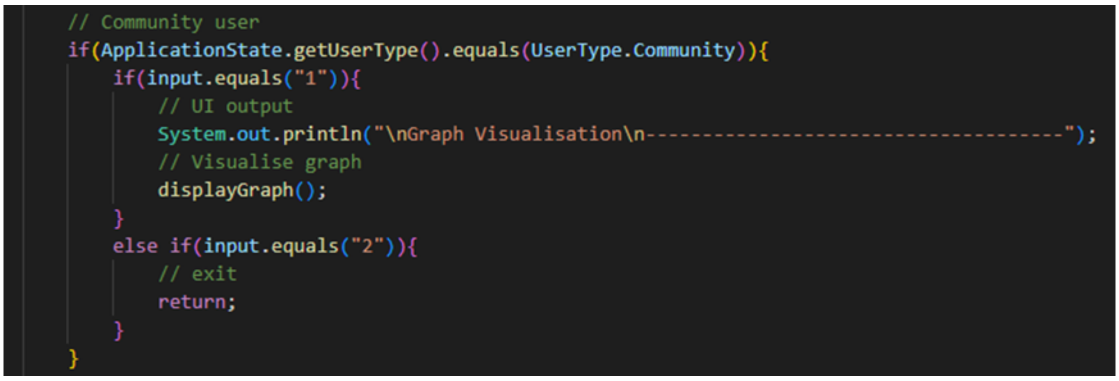
askUserForHousehold method:



```java
/*
 * Asks the user for their household and display the graph with only the devices in that household
 */
public static void askUserForHousehold() {
    // String checker for device ID
    String deviceIDCheck = "[A-Z]{1,3}-\\d{4}";
    // Asking user for their household ID
    System.out.println("What is your household ID?");
    // Variable to get user's reply
    String householdID = scanner.nextLine();
    // Checking whether it is a valid device ID
    if(householdID.matches(deviceIDCheck)) {
        // Get all the devices in the current dataset
        Device[] currentGraphDevices = graph.getDevices();
        // Arraylist to store all the current household devices
        List<Device> householdDevicesList = new ArrayList<>();
        // Loop used for adding all the current household devices into the array list
        for(int i = 0; i < currentGraphDevices.length; i++) {
            if(currentGraphDevices[i].getHouseholdId().equals(householdID)) {
                householdDevicesList.add(currentGraphDevices[i]);
            }
        }
        // Converting the household array list into an array
        Device[] householdDevicesArray = householdDevicesList.toArray(new Device[householdDevicesList.size()]);
        // Making a device graph for the current household
        DeviceGraph householdGraph = new DeviceGraph(householdDevicesArray);
        // Converting and visualising the graph
        GraphVisualiser graphVisualiser = new GraphVisualiser();
        graphVisualiser.convertGraph(householdGraph);
        graphVisualiser.visualiseGraph();
    } else {
        askUserForHousehold();
    }
}
```

Figure 3.6: A method that ask users for their household and display a graph with their household devices only

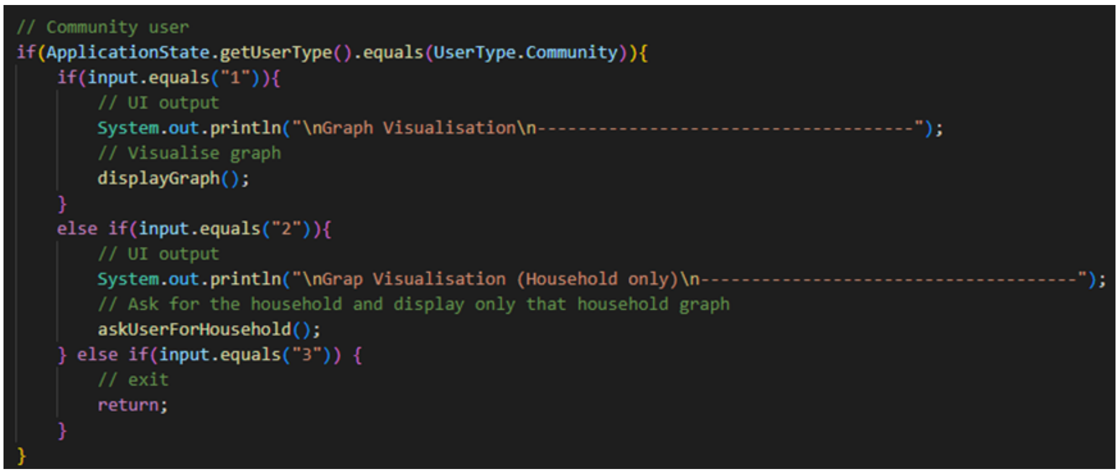
Before in ConsoleApp java file selectFeature method:



```
// Community user
if(ApplicationState.getUserType().equals(UserType.Community)){
    if(input.equals("1")){
        // UI output
        System.out.println("\nGraph Visualisation\n------------------------------------");
        // Visualise graph
        displayGraph();
    }
    else if(input.equals("2")){
        // exit
        return;
    }
}
```

Figure 3.7: selectFeature method in ConsoleApp before changes

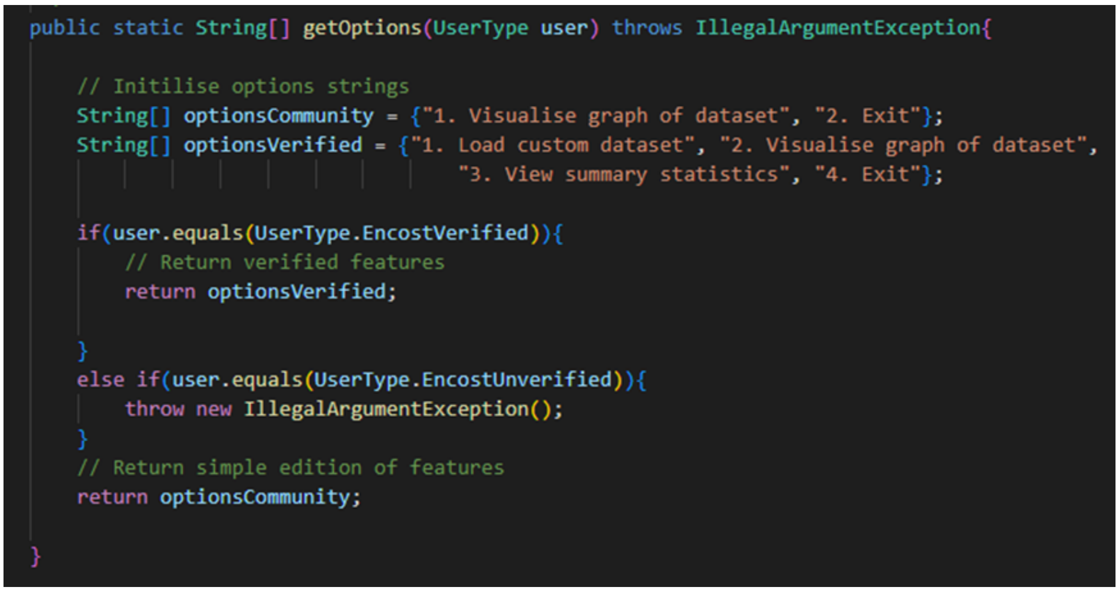
After in ConsoleApp java file selectFeature method:



```
// Community user
if(ApplicationState.getUserType().equals(UserType.Community)){
    if(input.equals("1")){
        // UI output
        System.out.println("\nGraph Visualisation\n------------------------------------");
        // Visualise graph
        displayGraph();
    }
    else if(input.equals("2")){
        // UI output
        System.out.println("\nGrap Visualisation (Household only)\n------------------------------------");
        // Ask for the household and display only that household graph
        askUserForHousehold();
    } else if(input.equals("3")) {
        // exit
        return;
    }
}
```

Figure 3.8: selectFeature method in ConsoleApp after changes

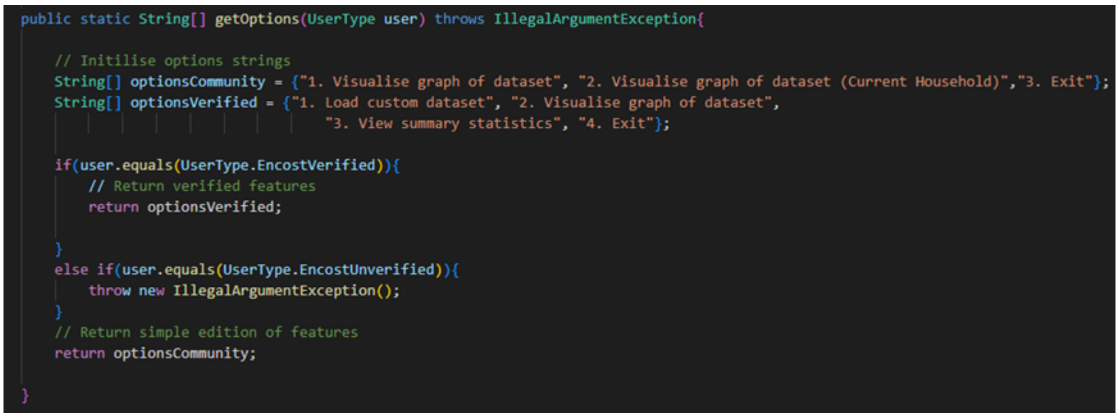
Before in ConsoleApp java file getOptions method:



```java
public static String[] getOptions(UserType user) throws IllegalArgumentException{

    // Initilise options strings
    String[] optionsCommunity = {"1. Visualise graph of dataset", "2. Exit"};
    String[] optionsVerified = {"1. Load custom dataset", "2. Visualise graph of dataset",
                                "3. View summary statistics", "4. Exit"};

    if(user.equals(UserType.EncostVerified)){
        // Return verified features
        return optionsVerified;

    }
    else if(user.equals(UserType.EncostUnverified)){
        throw new IllegalArgumentException();
    }
    // Return simple edition of features
    return optionsCommunity;

}
```

Figure 3.9: getOptions method in ConsoleApp before changes

After in ConsoleApp java file getOptions method:

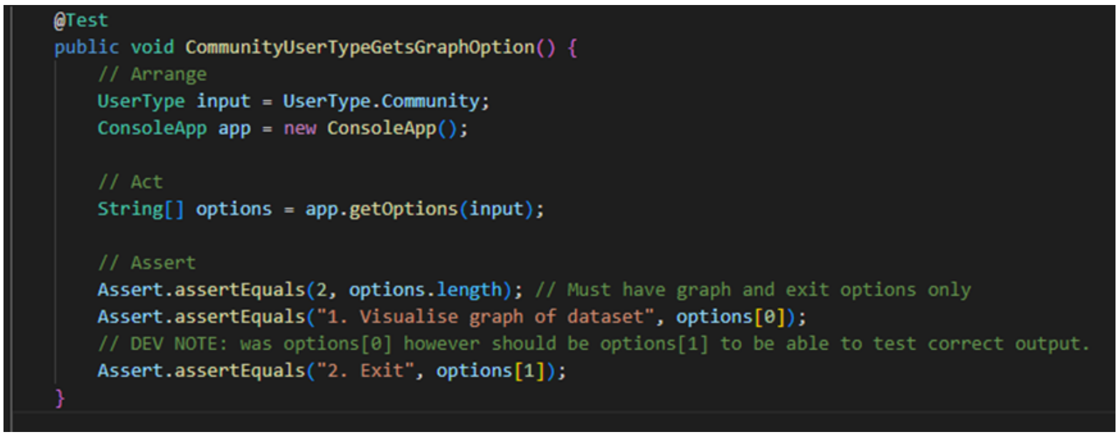


```java
public static String[] getOptions(UserType user) throws IllegalArgumentException{

    // Initilise options strings
    String[] optionsCommunity = {"1. Visualise graph of dataset", "2. Visualise graph of dataset (Current Household)","3. Exit"};
    String[] optionsVerified = {"1. Load custom dataset", "2. Visualise graph of dataset",
                                "3. View summary statistics", "4. Exit"};

    if(user.equals(UserType.EncostVerified)){
        // Return verified features
        return optionsVerified;

    }
    else if(user.equals(UserType.EncostUnverified)){
        throw new IllegalArgumentException();
    }
    // Return simple edition of features
    return optionsCommunity;

}
```

Figure 3.10: getOptions method in ConsoleApp after changes

3.3.6 Maintenance review/acceptance

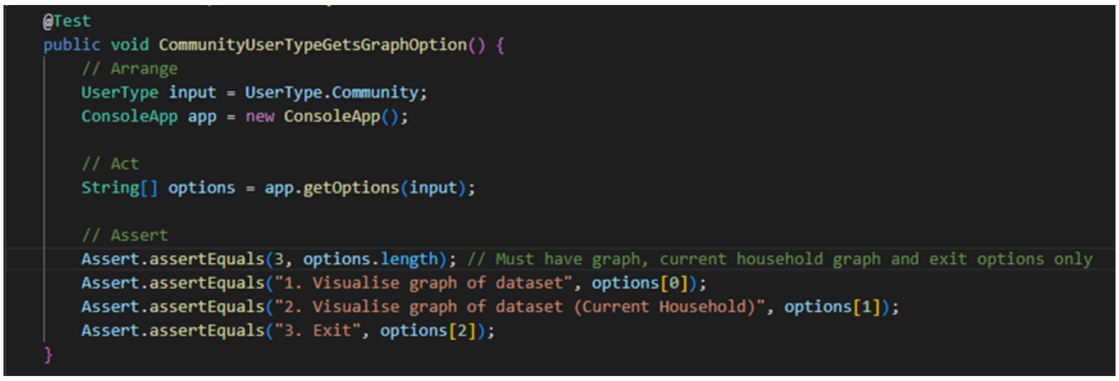
Before in FeatureOptionsTest java file:



```java
@Test
public void CommunityUserTypeGetsGraphOption() {
    // Arrange
    UserType input = UserType.Community;
    ConsoleApp app = new ConsoleApp();

    // Act
    String[] options = app.getOptions(input);

    // Assert
    Assert.assertEquals(2, options.length); // Must have graph and exit options only
    Assert.assertEquals("1. Visualise graph of dataset", options[0]);
    // DEV NOTE: was options[0] however should be options[1] to be able to test correct output.
    Assert.assertEquals("2. Exit", options[1]);
}
```

Figure 3.11: CommunityUserTypeGetsGraphOption test case in FeatureOptionsTest java file before changes

After in FeatureOptionsTest java file:

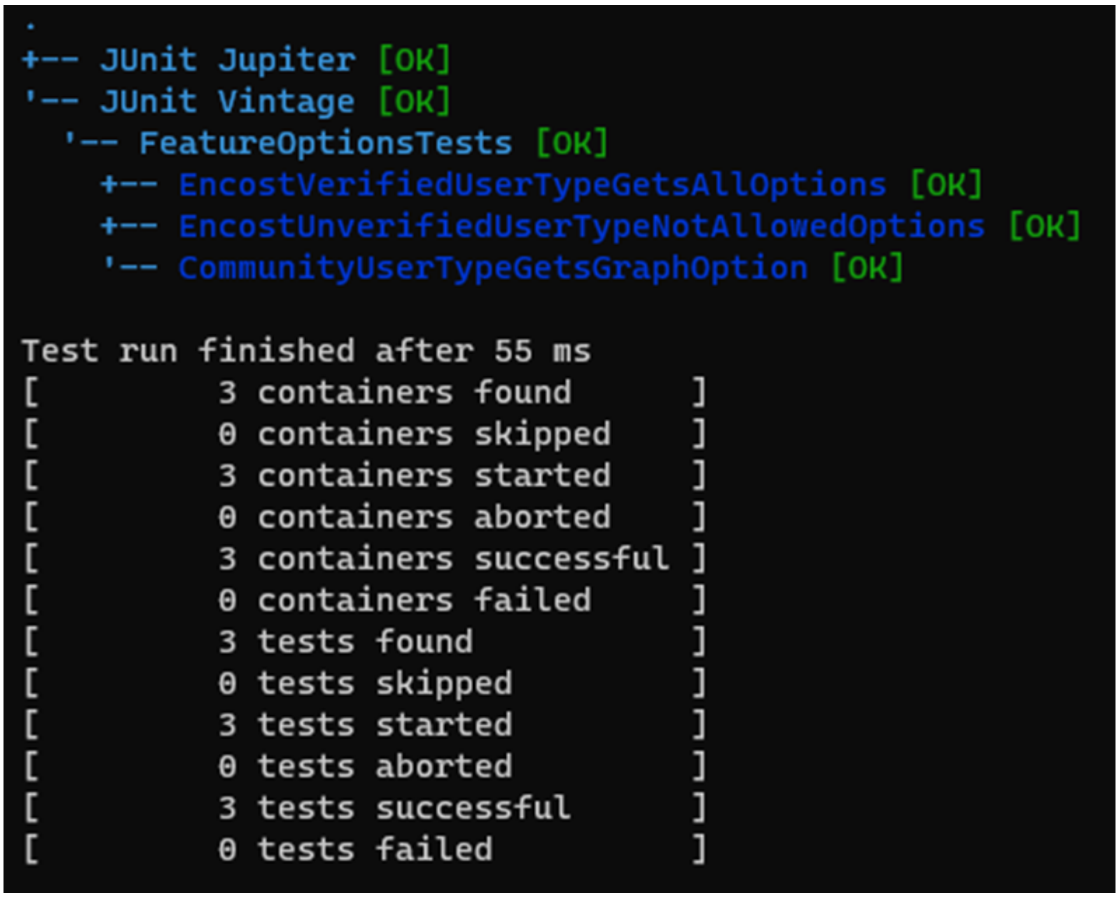


```java
@Test
public void CommunityUserTypeGetsGraphOption() {
    // Arrange
    UserType input = UserType.Community;
    ConsoleApp app = new ConsoleApp();

    // Act
    String[] options = app.getOptions(input);

    // Assert
    Assert.assertEquals(3, options.length); // Must have graph, current household graph and exit options only
    Assert.assertEquals("1. Visualise graph of dataset", options[0]);
    Assert.assertEquals("2. Visualise graph of dataset (Current Household)", options[1]);
    Assert.assertEquals("3. Exit", options[2]);
}
```

Figure 3.12: CommunityUserTypeGetsGraphOption test case in FeatureOptionsTest java file after changes

- Changed the assertEquals for option 2 to visualise for current household and made the exit option number 3.

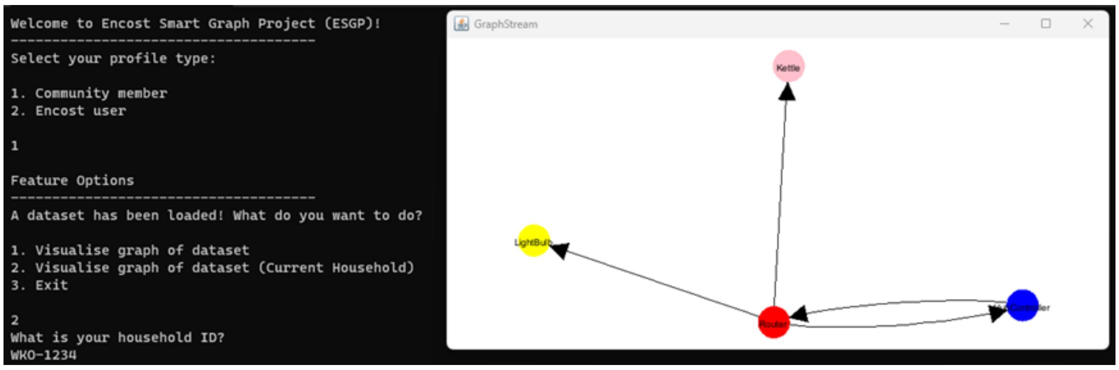
FeatureOptionsTest JUnit test:



```
.
+-- JUnit Jupiter [OK]
'-- JUnit Vintage [OK]
  '-- FeatureOptionsTests [OK]
    +-- EncostVerifiedUserTypeGetsAllOptions [OK]
    +-- EncostUnverifiedUserTypeNotAllowedOptions [OK]
    '-- CommunityUserTypeGetsGraphOption [OK]

Test run finished after 55 ms
[         3 containers found      ]
[         0 containers skipped    ]
[         3 containers started    ]
[         0 containers aborted    ]
[         3 containers successful ]
[         0 containers failed     ]
[         3 tests found           ]
[         0 tests skipped         ]
[         3 tests started         ]
[         0 tests aborted         ]
[         3 tests successful      ]
[         0 tests failed          ]
```

Figure 3.13: FeatureOptionsTests JUnit tests

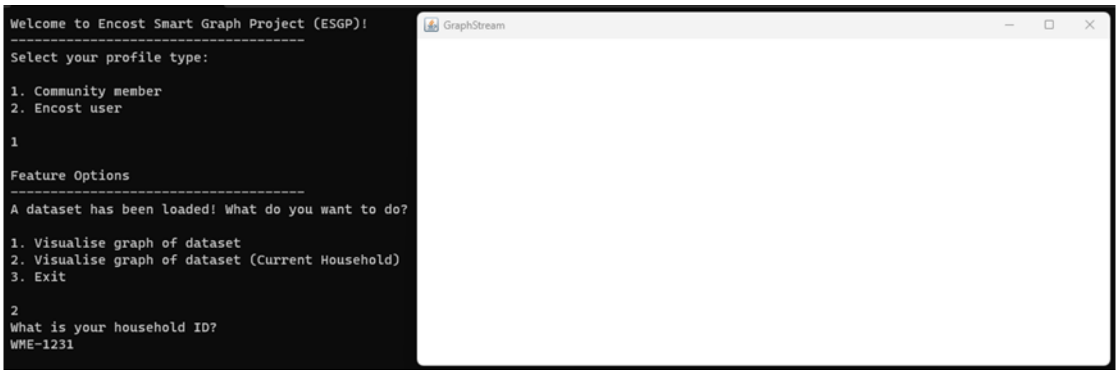
Existing household graph visualisation:



```
Welcome to Encost Smart Graph Project (ESGP)!
--------------------------------------
Select your profile type:

1. Community member
2. Encost user

1

Feature Options
--------------------------------------
A dataset has been loaded! What do you want to do?

1. Visualise graph of dataset
2. Visualise graph of dataset (Current Household)
3. Exit

2
What is your household ID?
WKO-1234
```

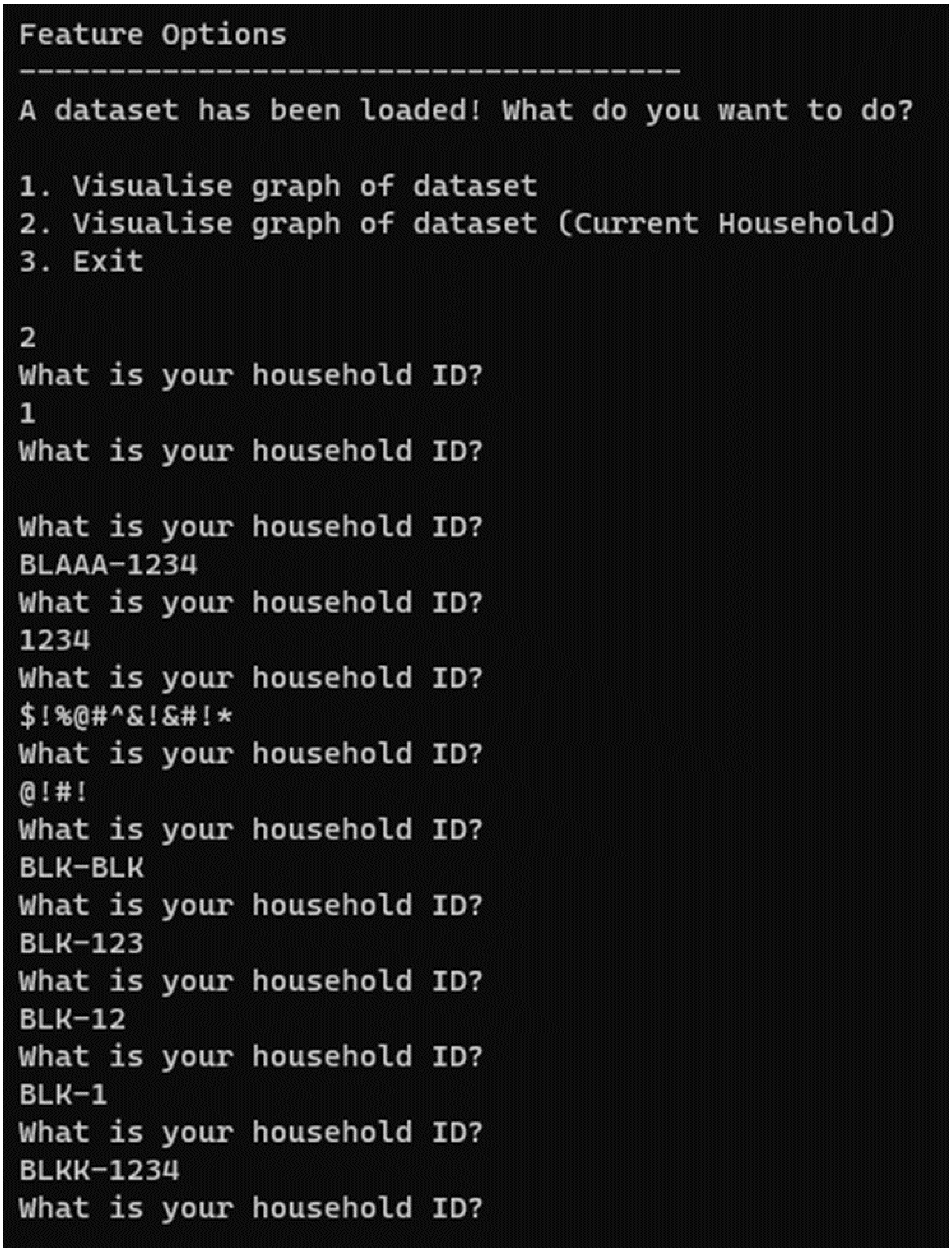
Figure 3.14: Household only graph visualisation when user input exist in the dataset

Non-existing household graph visualisation:



```
Welcome to Encost Smart Graph Project (ESGP)!
--------------------------------------
Select your profile type:

1. Community member
2. Encost user

1

Feature Options
--------------------------------------
A dataset has been loaded! What do you want to do?

1. Visualise graph of dataset
2. Visualise graph of dataset (Current Household)
3. Exit

2
What is your household ID?
WME-1231
```

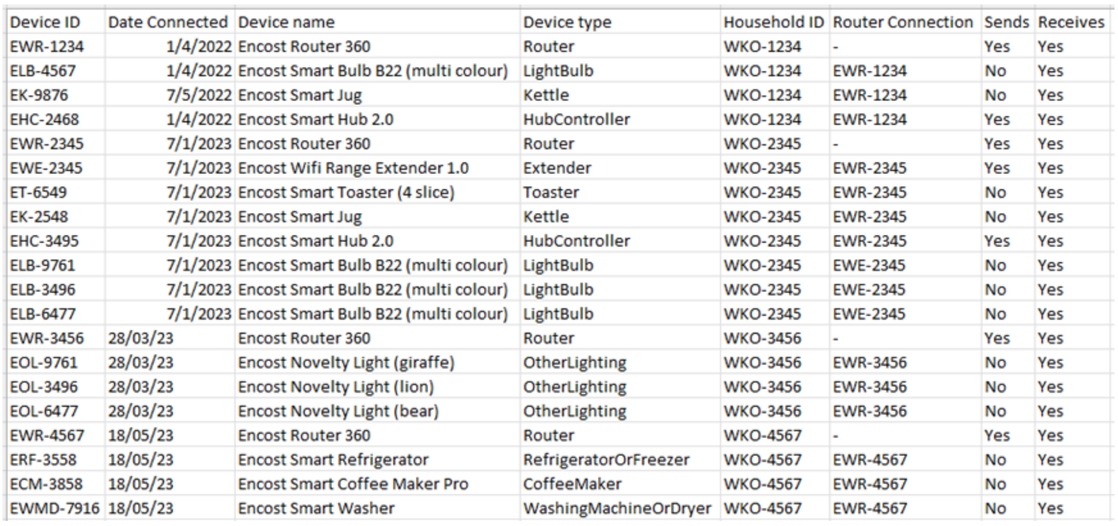
Figure 3.15: Household only graph visualisation when user input doesn't exist in the dataset



```
Feature Options
-------------------------------------
A dataset has been loaded! What do you want to do?

1. Visualise graph of dataset
2. Visualise graph of dataset (Current Household)
3. Exit

2
What is your household ID?
1
What is your household ID?

What is your household ID?
BLAAA-1234
What is your household ID?
1234
What is your household ID?
$!%@#^&!&#!*
What is your household ID?
@!#!
What is your household ID?
BLK-BLK
What is your household ID?
BLK-123
What is your household ID?
BLK-12
What is your household ID?
BLK-1
What is your household ID?
BLKK-1234
What is your household ID?
```

Figure 3.16: Household only graph visualisation when user input isn't a device ID

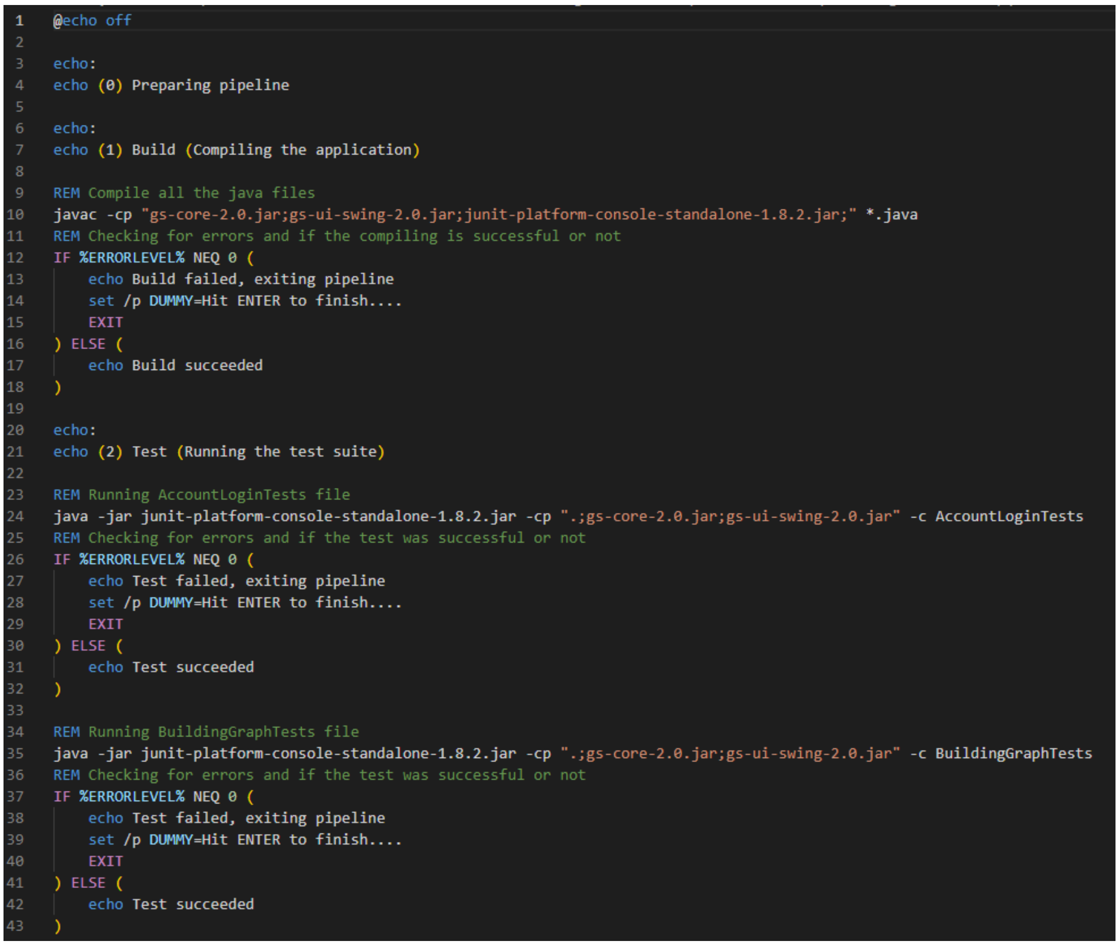
- If the user inputs anything that isn't a device ID, the system will keep asking for their household ID like in the figure below. Device ID must have 1-3 upper case letters first and a dash then followed by 4 digits.



Device ID	Date Connected	Device name	Device type	Household ID	Router Connection	Sends	Receives
EWR-1234	1/4/2022	Encost Router 360	Router	WKO-1234	-	Yes	Yes
ELB-4567	1/4/2022	Encost Smart Bulb B22 (multi colour)	LightBulb	WKO-1234	EWR-1234	No	Yes
EK-9876	7/5/2022	Encost Smart Jug	Kettle	WKO-1234	EWR-1234	No	Yes
EHC-2468	1/4/2022	Encost Smart Hub 2.0	HubController	WKO-1234	EWR-1234	Yes	Yes
EWR-2345	7/1/2023	Encost Router 360	Router	WKO-2345	-	Yes	Yes
EWE-2345	7/1/2023	Encost Wifi Range Extender 1.0	Extender	WKO-2345	EWR-2345	Yes	Yes
ET-6549	7/1/2023	Encost Smart Toaster (4 slice)	Toaster	WKO-2345	EWR-2345	No	Yes
EK-2548	7/1/2023	Encost Smart Jug	Kettle	WKO-2345	EWR-2345	No	Yes
EHC-3495	7/1/2023	Encost Smart Hub 2.0	HubController	WKO-2345	EWR-2345	Yes	Yes
ELB-9761	7/1/2023	Encost Smart Bulb B22 (multi colour)	LightBulb	WKO-2345	EWE-2345	No	Yes
ELB-3496	7/1/2023	Encost Smart Bulb B22 (multi colour)	LightBulb	WKO-2345	EWE-2345	No	Yes
ELB-6477	7/1/2023	Encost Smart Bulb B22 (multi colour)	LightBulb	WKO-2345	EWE-2345	No	Yes
EWR-3456	28/03/23	Encost Router 360	Router	WKO-3456	-	Yes	Yes
EOL-9761	28/03/23	Encost Novelty Light (giraffe)	OtherLighting	WKO-3456	EWR-3456	No	Yes
EOL-3496	28/03/23	Encost Novelty Light (lion)	OtherLighting	WKO-3456	EWR-3456	No	Yes
EOL-6477	28/03/23	Encost Novelty Light (bear)	OtherLighting	WKO-3456	EWR-3456	No	Yes
EWR-4567	18/05/23	Encost Router 360	Router	WKO-4567	-	Yes	Yes
ERF-3558	18/05/23	Encost Smart Refrigerator	RefrigeratorOrFreezer	WKO-4567	EWR-4567	No	Yes
ECM-3858	18/05/23	Encost Smart Coffee Maker Pro	CoffeeMaker	WKO-4567	EWR-4567	No	Yes
EWMD-7916	18/05/23	Encost Smart Washer	WashingMachineOrDryer	WKO-4567	EWR-4567	No	Yes

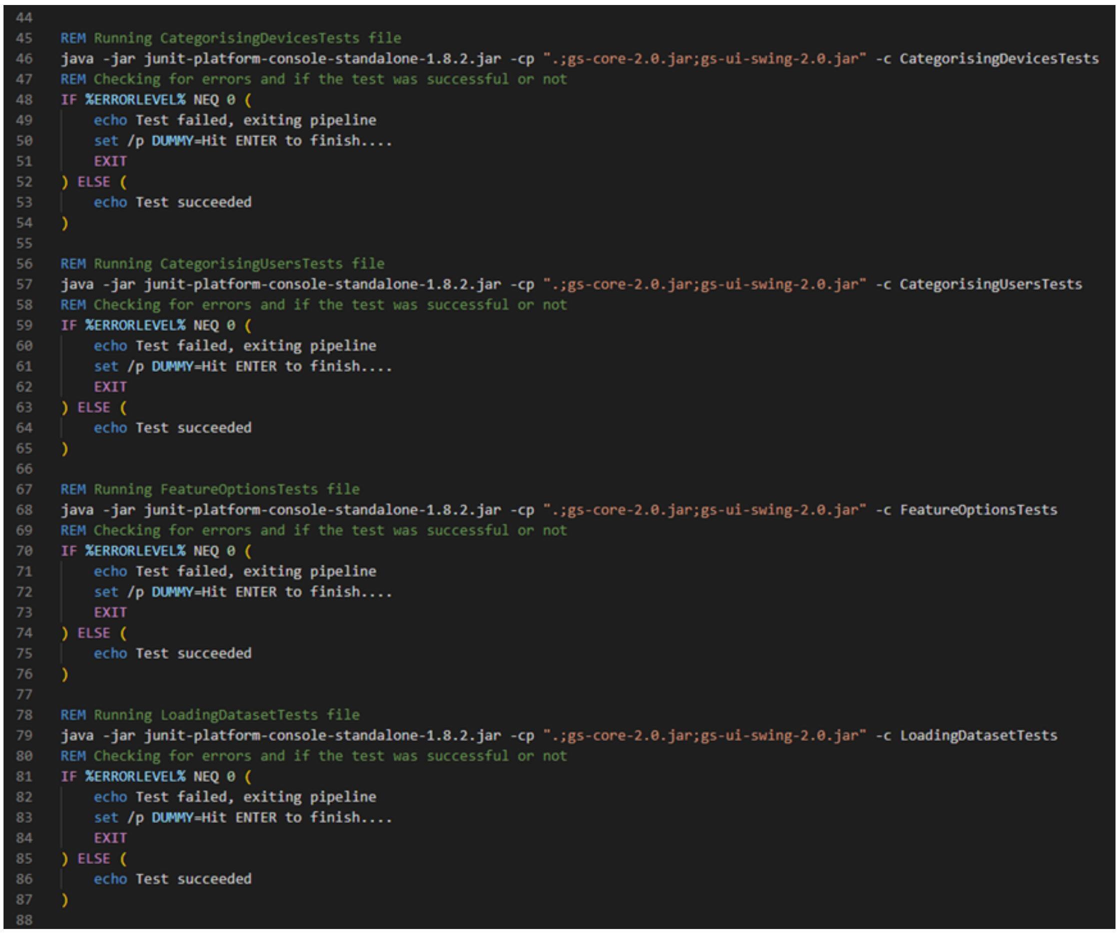
Figure 3.17: Dataset used for the graph visualisation

4 CI/CD Pipeline



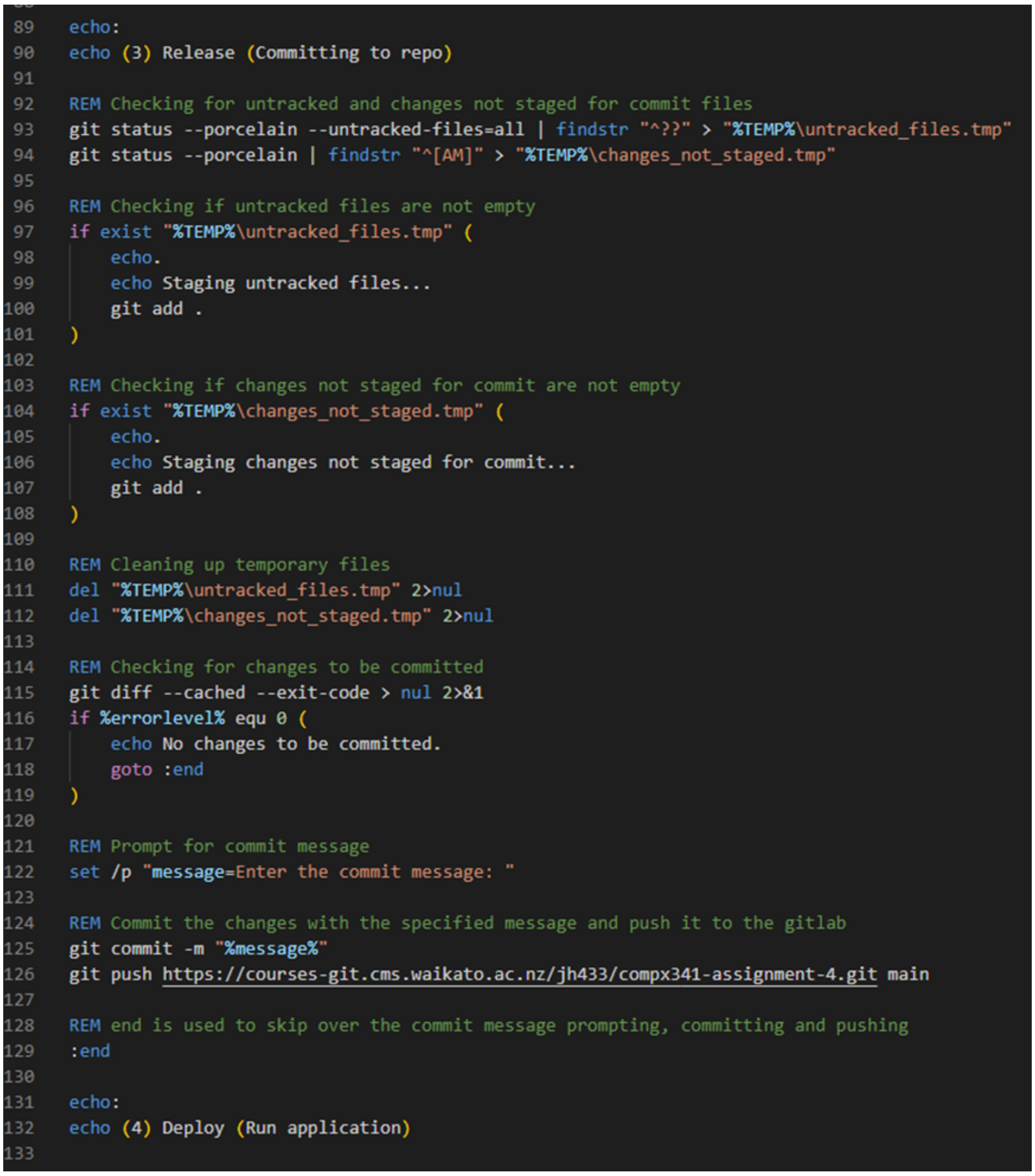
```
@echo off

echo:
echo (0) Preparing pipeline

echo:
echo (1) Build (Compiling the application)

REM Compile all the java files
javac -cp "gs-core-2.0.jar;gs-ui-swing-2.0.jar;junit-platform-console-standalone-1.8.2.jar;" *.java
REM Checking for errors and if the compiling is successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Build failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Build succeeded
)

echo:
echo (2) Test (Running the test suite)

REM Running AccountLoginTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c AccountLoginTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)

REM Running BuildingGraphTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c BuildingGraphTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)
```

Figure 4.1: CI/CD pipeline part 1



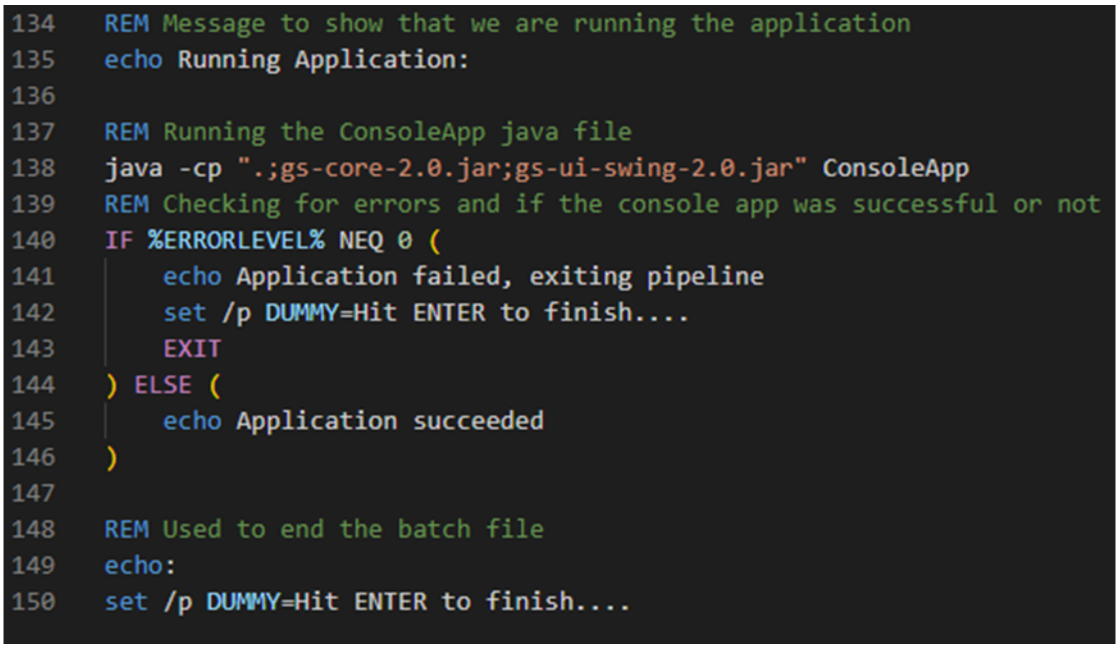
```
REM Running CategorisingDevicesTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c CategorisingDevicesTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)

REM Running CategorisingUsersTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c CategorisingUsersTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)

REM Running FeatureOptionsTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c FeatureOptionsTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)

REM Running LoadingDatasetTests file
java -jar junit-platform-console-standalone-1.8.2.jar -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" -c LoadingDatasetTests
REM Checking for errors and if the test was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Test failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Test succeeded
)

```

Figure 4.2: CI/CD pipeline part 2



```
echo:
echo (3) Release (Committing to repo)

REM Checking for untracked and changes not staged for commit files
git status --porcelain --untracked-files=all | findstr "^??" > "%TEMP%\untracked_files.tmp"
git status --porcelain | findstr "^[AM]" > "%TEMP%\changes_not_staged.tmp"

REM Checking if untracked files are not empty
if exist "%TEMP%\untracked_files.tmp" (
    echo.
    echo Staging untracked files...
    git add .
)

REM Checking if changes not staged for commit are not empty
if exist "%TEMP%\changes_not_staged.tmp" (
    echo.
    echo Staging changes not staged for commit...
    git add .
)

REM Cleaning up temporary files
del "%TEMP%\untracked_files.tmp" 2>nul
del "%TEMP%\changes_not_staged.tmp" 2>nul

REM Checking for changes to be committed
git diff --cached --exit-code > nul 2>&1
if %errorlevel% equ 0 (
    echo No changes to be committed.
    goto :end
)

REM Prompt for commit message
set /p "message=Enter the commit message: "

REM Commit the changes with the specified message and push it to the gitlab
git commit -m "%message%"
git push https://courses-git.cms.waikato.ac.nz/jh433/compx341-assignment-4.git main

REM end is used to skip over the commit message prompting, committing and pushing
:end

echo:
echo (4) Deploy (Run application)

```

Figure 4.3: CI/CD pipeline part 3



```
REM Message to show that we are running the application
echo Running Application:

REM Running the ConsoleApp java file
java -cp ".;gs-core-2.0.jar;gs-ui-swing-2.0.jar" ConsoleApp
REM Checking for errors and if the console app was successful or not
IF %ERRORLEVEL% NEQ 0 (
    echo Application failed, exiting pipeline
    set /p DUMMY=Hit ENTER to finish....
    EXIT
) ELSE (
    echo Application succeeded
)

REM Used to end the batch file
echo:
set /p DUMMY=Hit ENTER to finish....
```

Figure 4.4: CI/CD pipeline part 4

My CI/CD pipeline has 5 portions. The preparing in the pipeline is just used to start the pipeline. The building in the pipeline is used to compile all the java files and check whether it succeeds or not. The test in the pipeline is used to run all tests and checks whether they all succeed or not. The release in the pipeline is used to commit to the repository. The pipeline first checks whether any files need to be added. If there is any files needed to be added, it will add it. After it adds it, it will ask the user for the commit message. Once the user enters the commit message, the pipeline will commit and push the files into the repository. The last portion is the deploying. The pipeline just runs the application and let the user use the application. It will show a message if it fails or succeed after the user exits the application.

5 Conclusion

With the maintenance, the GraphStream for the application is now the latest version and there is a new feature which allows community user to give their household ID and it will display a graph visualisation with only the devices from the household ID stated. The CI/CD pipeline is now able to prepare, build, test, release and deploy.