NavUP System

Testing phase

Group name: Broadsword Data

Jacques Smulders - u15003087
Keegan Ferret - u15132847
Lesego Makaleng - u15175716
Linda Potgieter - u14070091
Lyle Nel - u29562695
Seonin David - u15063021

Contents

1 Introduction

This document specifies the steps taken to test the Gladius Implementation of the data subsystem. The data subsystem was required to return a devices location based on its MAC address. The location returned would be the x and y coordinates of the connected device (based on its MAC address) relevant to the Wi-Fi hotspot it is connected to at the time of request. Testing consisted of functional and non-functional requirement tests, with each test case having a description, severity level, a mark obtained out of 10 and a status which may either be Pass or Fail.

2 Functional Tests

2.1 Introduction

The functional requirements specify the behaviour of the system. It consists of the abilities that the system must have to be considered functional. The main purpose of the subsystem, whether it will be able to return the location (relevant to a connected Wi-Fi hotspot) of a device with a given MAC address, will be tested. Testing will also include error checking and handling for various components of the subsystem including the Aruba Location Engine, Apache Flink and Json queries. Each test case has a test description, severity level, obtained mark out of 10 and a status of either Pass or Fail.

2.2 Tests

Serial:F1
Title: Running system when connected to the ALE
Description: For this test, the system was queried with a valid mac address while connected to the ALE to see if a location is returned. The test proved succesful with the output shown in **Figure 1** produced.
Severity: N/A
Score: 10
Status: Pass

Serial:F2
Title: Running system when disconnected from the ALE
Description: For this test, the system was queried with a valid mac address while disconnected from the ALE to see if the system is capable of catching and displaying errors. A reasonably verbose error was displayed:

> *Error [Source: Socket Stream -¿ Map -¿ Sink: Unnamed (1/1)] INFO org.apache.flink.runtime.taskmanager.Task - Source: Socket Stream -¿ Map -¿ Sink: Unnamed (1/1) (ea9f7d77aad0d00674c883feb4176b12) switched from RUNNING to FAILED. java.lang.RuntimeException: Could not for-*

Figure 1: Location Returned

ward element to next operator

However, the program crashed after producing the error, causing the test to fail.
Severity: Critical
Score: 0
Status: Fail

Serial:F3
Title: Running Apache Flink with defined data sink
Description: For this test I ran Apache Flink code that I got from the Gladios Data team's repository. The system's defined data sink is:

text.map(new Mapper()).writeAsText ("C:/Users/rob/Documents/NetBeansProjects/COS301/test.txt");

This line will only work on one computer and did not run when we tested it. As can be seen in **Figure 2**, the reason why the file directory cannot be found is because it is only specified for one computer.

```
[Source: Socket Stream -> Map -> Sink: Unnamed (1/1)] INFO org.apache.flink.runtime.taskmanage
java.io.IOException: Mkdirs failed to create C:/Users/rob/Documents/NetBeansProjects/COS301
```

Figure 2: Wrong File Path

Severity: Critical
Score: 0
Status: Fail

Serial:F4
Title: Running Flink after changing data sink

Description: For this test I ran Apache Flink code but I changed the data sink to

```
text.map(new Mapper()).writeAsText(/"tmp/test.txt");
```

This is done so the system can run on any linux computer. After changing the data source the system runs without any error and returns a devices location in the text file, as can be seen in **Figure 3**.As a side note is shows that the result is the Longitude and Latitude but it is actually the x and y values relative to the access point.

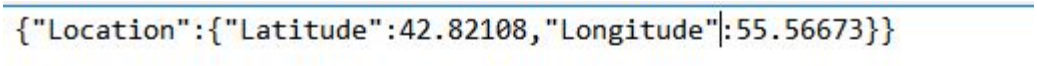

Figure 3: Output

Severity: N/A
Score: 8
Status: Pass

Serial:F5
Title: Running flink without starting the Job Manager
Description: This test will see if Apache Flink runs and returns the relevant information without starting the Job Manager. Flink does run without a Job Manager and returns the same result which can be seen in Figure 3 of the previous test.
Severity: N/A
Score: 10
Status: Pass

Serial:F6
Title: Does the current system stream data
Description: Currently Gladios Data only queried a devices mac address once. From this test the system return a devices location, which can be seen in **Figure 4**.
Severity: N/A
Score: 9
Status: Pass

Serial:F7
Title: Can the system stream and handle 4000 requests

Figure 4: Correct Mac Address

Description: For this test case a for loop is used and queries 2 mac addresses 2000 times. The code can be seen in Figure 5. The system only returns the location of one mac address and does not do anything after that. From this I, can conclude that the Gladios Data module cannot take a data stream in nor produce an output data stream.

```
for (int i=0;i<2000;i++)
{
    temp = data.getLocation("58:48:22:a7:84:6b");
    t = data1.getLocation("40:b8:37:bd:f5:98");
}
```

Figure 5: Stream Test

Severity: Critical
Score: 0
Status: Fail

Serial:F8
Title: Querying system with a malformed mac address
Description: For this test, the system was queried with a malformed mac address while connected to the ALE to see if a location is returned. All though the system did not crash, it did return an empty location (see **Figure 4**) object instead of some kind of exception to inform the caller that the mac address was invalid.
Severity: N/A
Score: 7

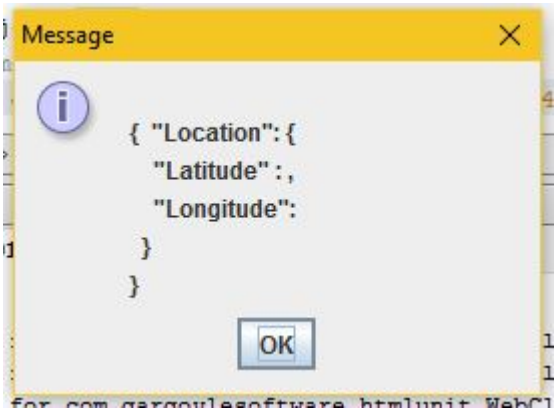

Figure 6: Empty Location Returned

Status: Pass

Serial:F9
Title: Querying system with a mac address which is not connected to the network
Description: For this test, the system was queried with a valid mac address, which would have not been visible on the ALE server, to see if a location is returned. All though the system did not crash, it did return an empty location (see **Figure 5**) object instead of some kind of exception to inform the caller that the mac address was invalid.

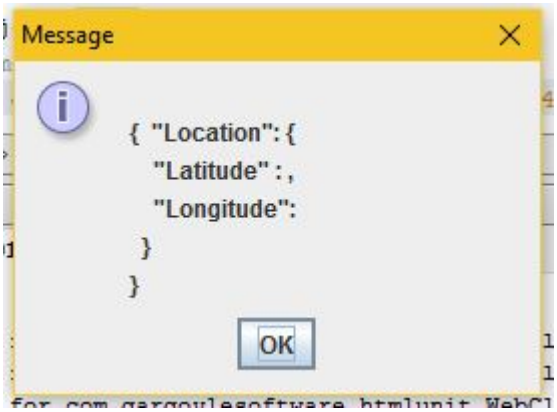

Figure 7: Empty Location Returned

Severity: N/A
Score: 7
Status: Pass

3 Non-Functional Tests

3.1 Introduction

The non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviours. The non-functional requirements consist of requirements which do not affect the behaviour of the system such as security of sensitive content, implementation of concurrency mechanisms for performance as well as efficacy and ease of the deployment frameworks implemented. Each test case has a test description, severity level, obtained mark out of 10 and a status of either Pass or Fail.

3.2 Tests

Serial:NF1
Title: Security: Aruba Credentials
Description: Take note of lines 45-46 in ClientLoginWeb.java. The Aruba credentails are embedded in the source code on a public repository, which is a big security risk.
Severity: Critical
Score: 0
Status: Fail

Serial:NF2
Title: Ease of deployment: Parametric settings
Description: Settings such as the hostname and credentials are hardcoded instead of a settings file or program arguments.
Severity: Low
Score: 0
Status: Fail

Serial:NF3
Title: Deployment Framework
Description: Maven was used to manage dependencies. This means the project can be compiled into a single jar file for execution by external entities.
Severity: Low
Score: 10
Status: Pass

Serial:NF4
Title: Concurrency
Description: The project fails to write to the same file concurrently. No concurrent mechanisms are put in place.

Severity: Low
Score: 0
Status: Fail