

jobder application

Testing Report



April 22, 2018

Name: Adrian Rabbitt

Student Number: 14500447

Title: Jobder

Finish Date: 20/05/2018

Contents

[1. Approach 2**.**](#_Toc512164073)

[2. Testing methods 3](#_Toc512164076)

[2.1 The core of the testing framework is the "Test System". 3](#_Toc512164077)

[3. Android and Android testing Frameworks. 5](#_Toc512164078)

[3.1 Expresso 5](#_Toc512164079)

[3.2 JUnit. 6](#_Toc512164080)

[4. Testing of the Application. 8](#_Toc512164081)

[5. Functional UI Testing (Expresso). 28](#_Toc512164082)

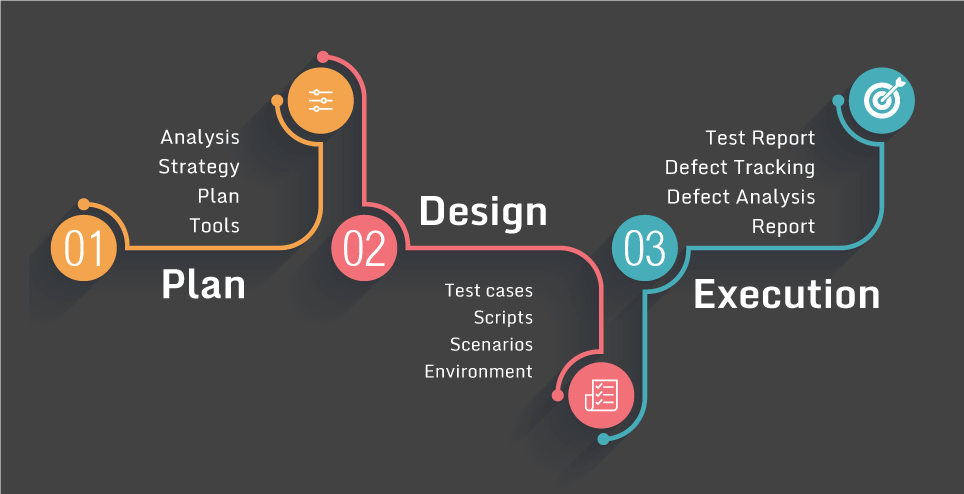
[6. Performance Issues and Resolution. 49](#_Toc512164083)

[7. Stress Testing in Android Using Monkey. 51](#_Toc512164084)

[8. Testing throughout the application. 61](#_Toc512164085)

[9. User Testing: 54](#_Toc512164086)

**JOBDER APPLICATION TESTING**



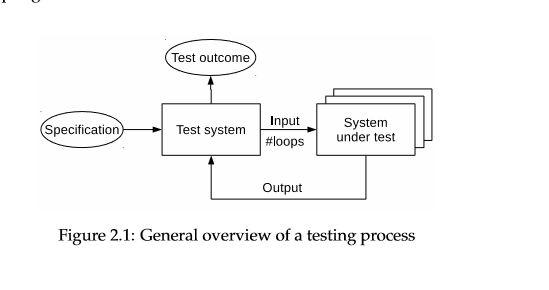
# Approach

During the Implementation of the application Jobder, I was faced with many issues during the development process, that I was able to overcome. I have recorded several tests within this document and have performed a stress test of the entire application, with the goal of ﬁnding errors that the functional tests did not ﬁnd. To perform the functional tests, I have used Google’s framework Espresso Junit and Monkey.

# Testing methods

The main reason I started software testing is to ﬁnd faults that can be ﬁxed to increase the reliability of the system. These faults may cause potential breaks within the code. The second category, functional errors, has more to do with the expected behaviour of the application the software runs. An example of a functional error is that images, buttons or other graphic elements are not displayed correctly, either at different times or on different devices. Another one might be that a text piece is displayed in the incorrect setting or to speciﬁc users that should not see the text Figure.

## The core of the testing framework is the "Test System".



**White, grey and black box testing.**

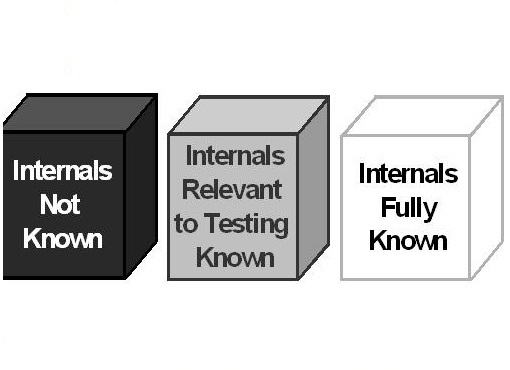
Software testing can be divided into three categories white, grey and black box testing. Black box testing is a method that mainly focuses on the functionality of the application without taking into its considerations of the applications internal functions. When implementing this test it can be applied to any level of software testing e.g. unit, integration system and acceptance testing. The main reason I am applying this test is for the following reasons.

* Incorrect or missing functions.
* Interface errors
* Errors in data structures or external database access
* Behaviour or performance errors

White box testing also known as glass box testing, this is nearly opposite to Black Box testing, where the tester has to know explicit knowledge of the internal workings of the system being tested. During my implementation of unit testing I will be using programming code where the outputs will be studied. The reasons why I am incorporating white box testing is to:

* Efficient in finding errors and problems
* Required knowledge of internals of the software under test is beneficial for thorough testing
* Allows finding hidden errors
* Programmers introspection
* Helps optimizing the code

Grey box testing is a strategy for software debugging in which the tester has limited knowledge of the internal details of the program. It receives its name from a simple combination OF white box and black box testing.



# Android and Android testing Frameworks.

The android platform is very popular with mobile technology currently holding the largest market for mobile applications. When creating an application for the android operating system it can be run on debugging mode using an android device or a hosted emulator running locally on your Virtual Machine (VM).

Android consists of a custom Linux Kernel in the bottom of the software stack, the android applications are usually written in in Java, the User Interface (UI) is expressed through XML where parts of the UI can be programmatically established. Each property in the android device are usually linked through id codes where a view is established, when undergoing tests using expresso I had to also link the objects through their unique ids using the testing platform.

The android platform is open source, which makes easy to create testing frameworks.

## Expresso

Expresso is a testing framework for android to make it easy to write reliable user testing interface testing to aid in functional testing, this platform is created and maintained by Google. Expresso automatically synchronises test actions with the UI, the framework also ensures the activity undergoing the test is started before the testcases are applied. A summary of Expresso can be split into 3 main components

* ViewMatchers
* ViewActions
* ViewAssertions

Where ViewMatchers are used to find the view in the current view hierarchy, View Actions allows to perform actions on the views and view Assertions allows the user to assert the state of view. The main reasons I applied Expresso to the application is

* Each build needs to be validated after code changes are made.
* Dependencies on remote servers and other workstations for testing slow down the process.
* Unit and functional tests need to be easy to execute from both an IDE and continuous integration perspective.
* Testing can occur on both emulators and real devices.
* Fast and Reliable Feedback to developers.

## JUnit.

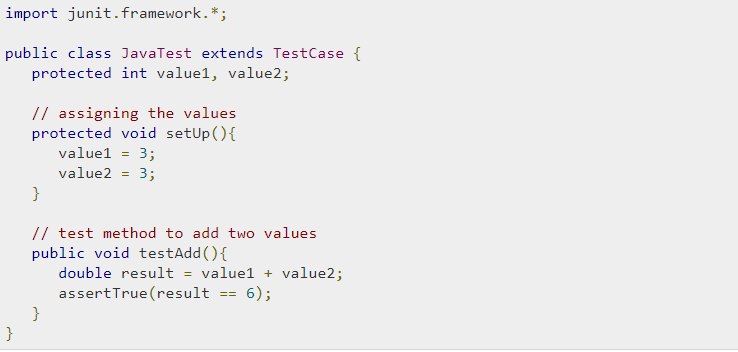
JUnit is a unit testing framework for the Java Language. It is a regression Testing framework used by android developers to create and execute unit tests in Java. The platforms main features include:

* Fixtures
* Test Suites
* Test runners
* JUnit classes

Fixtures is a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that which tests are run so that results are repeatable. It includes –

* SetUp() method, which runs before every test invocation.
* tearDown() method, which runs after every test method.

e.g.



**Test Suites.**

Test suite allows to tester to run a few unit tests at the one time here are examples referenced from <https://www.tutorialspoint.com/junit/junit_test_framework.htm> to show the purpose of the platform.



**Test Runners.**

Here is where the test cases are executed.

**JUnit Classes.**

These are the classes used for writing test cases.

Assert- Contains all assert methods.

TestCase -Test case involved.

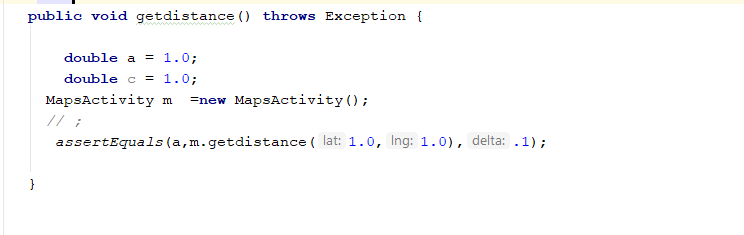
TestResult – Contains methods to collect the results.

# Testing of the Application.

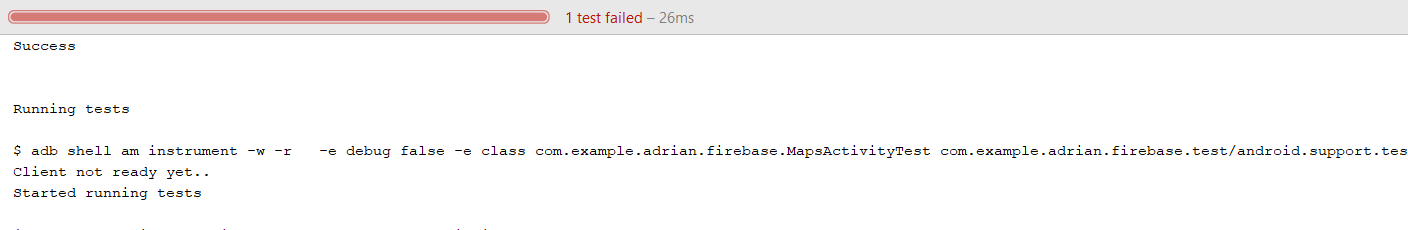
The following Testcases are created by JUnit that I adapted throughout the development process of the application Jobder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **1** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | Distance not equal to null | Positive Value | FAILED | This was a simple test that was expected to pass |
| 2 | Distance returns the approximate distance in km. | Distance is returned as 1.523 from coordinates 1 lat and 2 lng | FAILED | This result was to verify that the haversine formula was correctly implemented. |

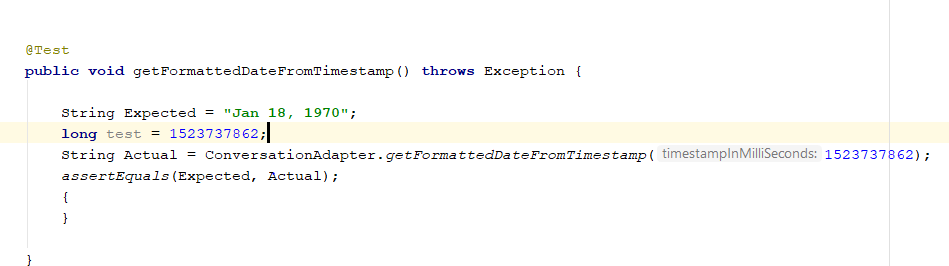
JUnit Test.



Visual Result



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **2** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | The getFormattedDateTimeStamp() will not return null. | String result will never be equal to null. | PASS |  |
| 2 | The correct date will be returned with regard to its miliseconds | When 1523737826 is passes in as parameter the date  “Jan 18, 1970 is returned. | PASS | The correct date was returned where the test was verified. |

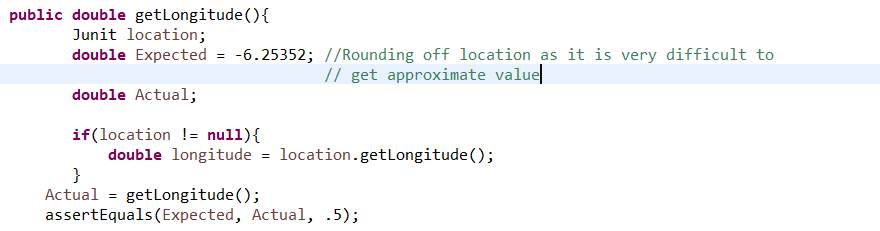
JUnit Testing

Visual Result

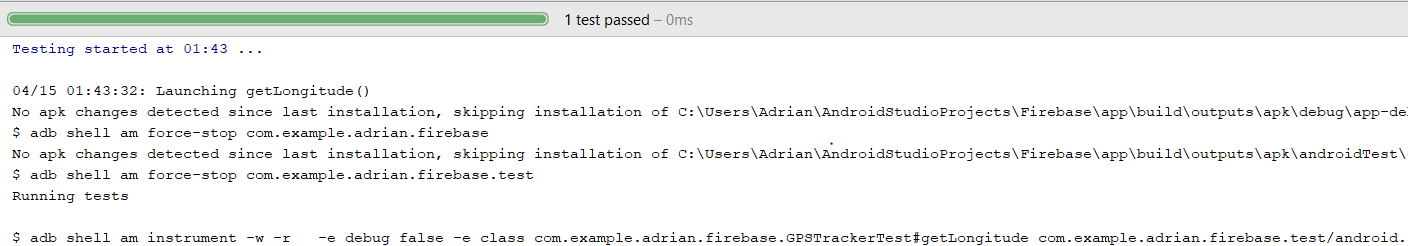


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **3** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | The value for Longitude will never be equal to null. | Longitude will be a whole number. | PASS |  |
| **2** | The result will be approximately correct according to X10-3 correctness. | Correct Location for Longitude will be returned. | **PASS** | The correct Location from the longitude values were returned. |

JUnit.

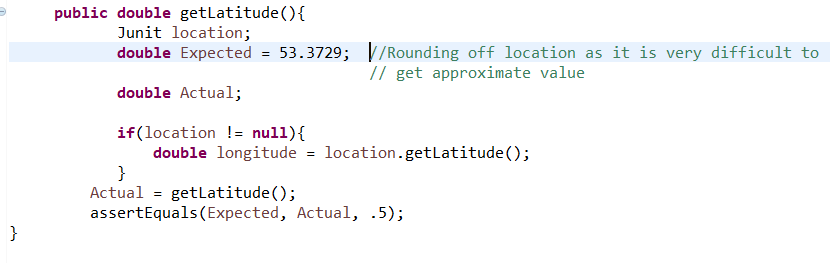


Visual Result.

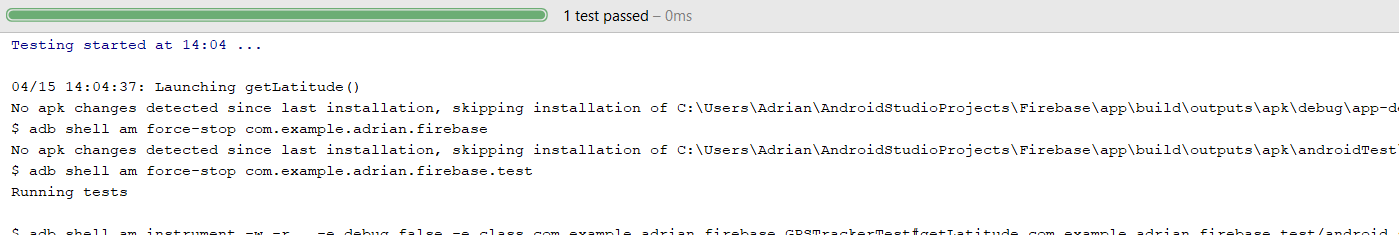


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **4** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | The value for Latitude will never be equal to null. | Latitude will be a whole number. | PASS |  |
| **2** | The result will be approximately be correct according to X10-3 correctness. | Correct Location for Latitude will be returned. | **PASS** | The correct Location from the latitude values were returned. |

JUnit.

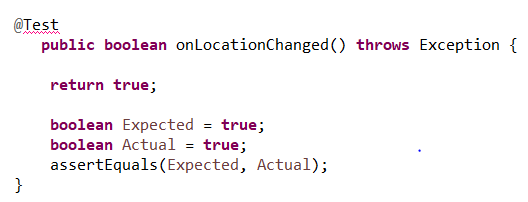


Visual Result.

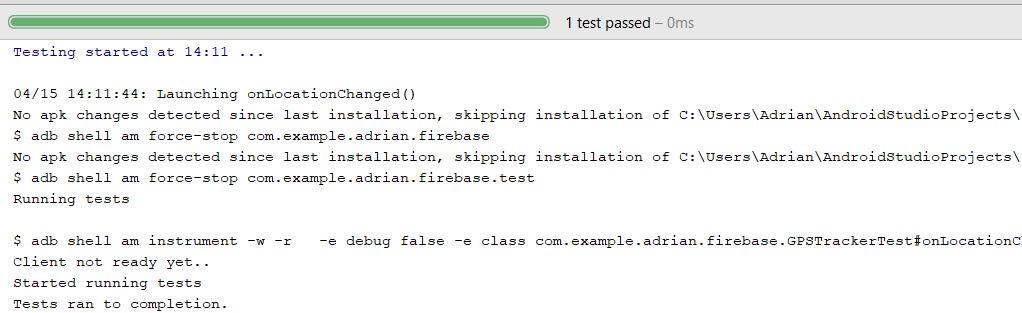


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **5** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | Location will be returned every time the location is changed | Change in location when function is executed is not equal to null. | PASS | Location returns correct every time function is executed |

JUnit

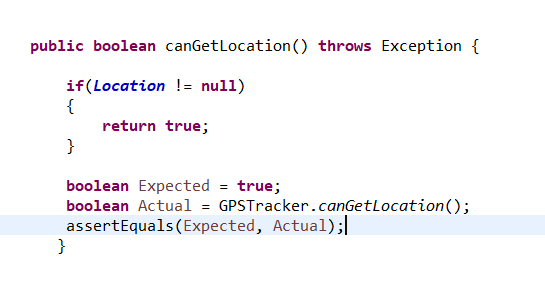


Visual Result

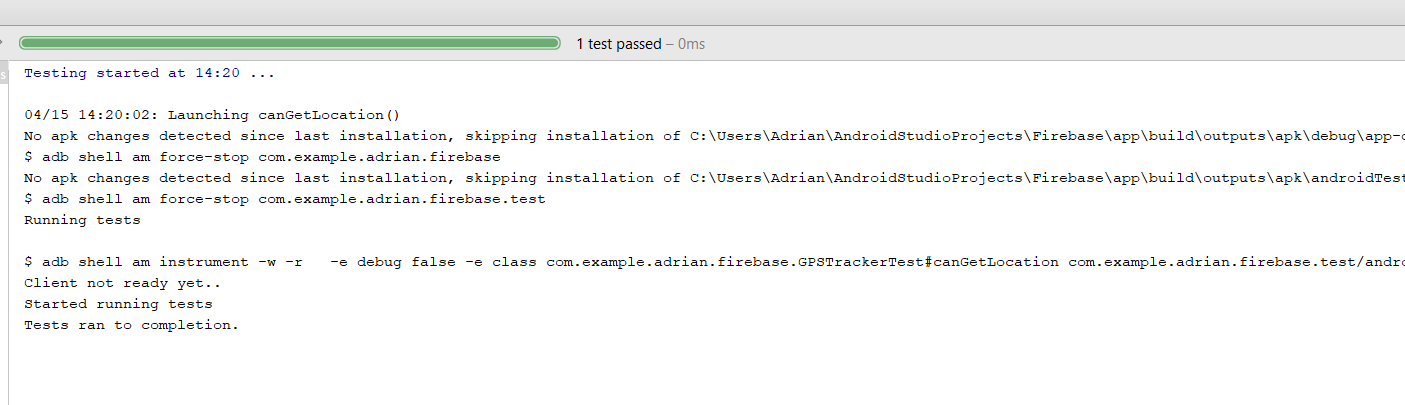


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **6** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the location is not equal to null the canGetLocation will return true when the function is executed. | canGetLocation will return true. | PASS | This ensured me that the function will return true once there is a Location available. |

JUnit.

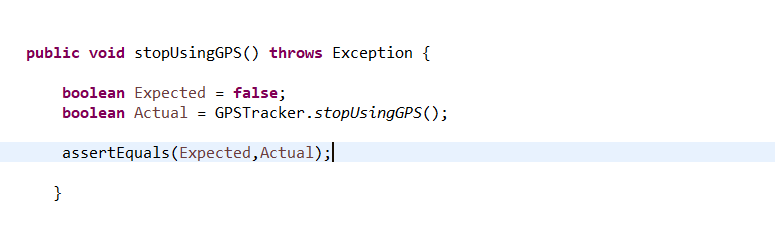


Visual Result.

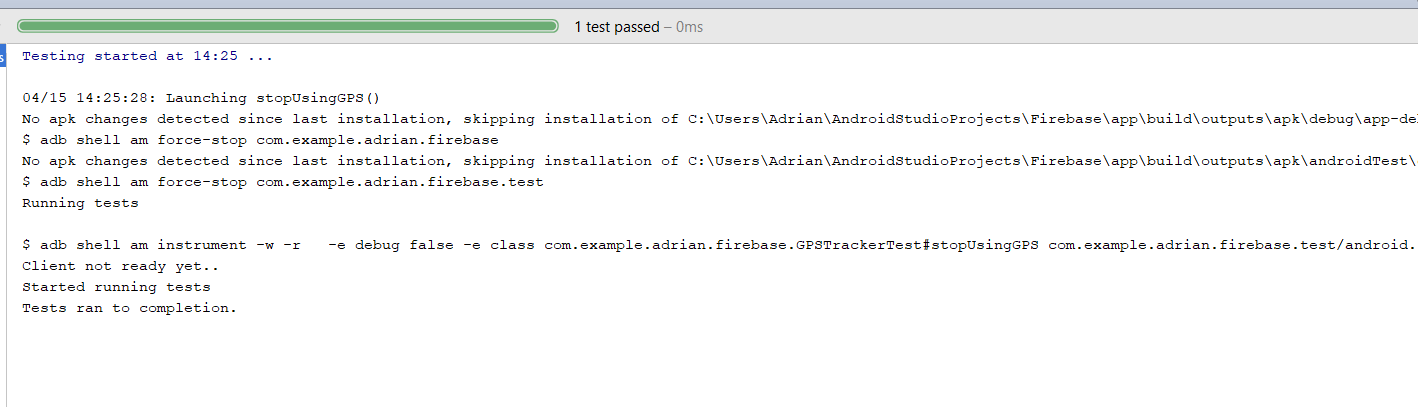


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **7** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the location services are in use the stopUsing GPS will return false as the GPS service is still in use. | When Location is in service the result of the function executed will return null. | PASS | This test was passed and will reduce the reduce percentage error of the application braking. |

JUnit.

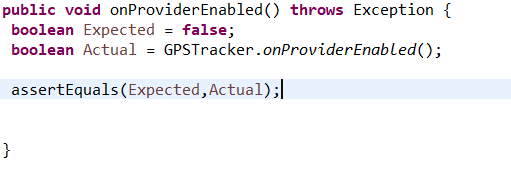


Visual Result.

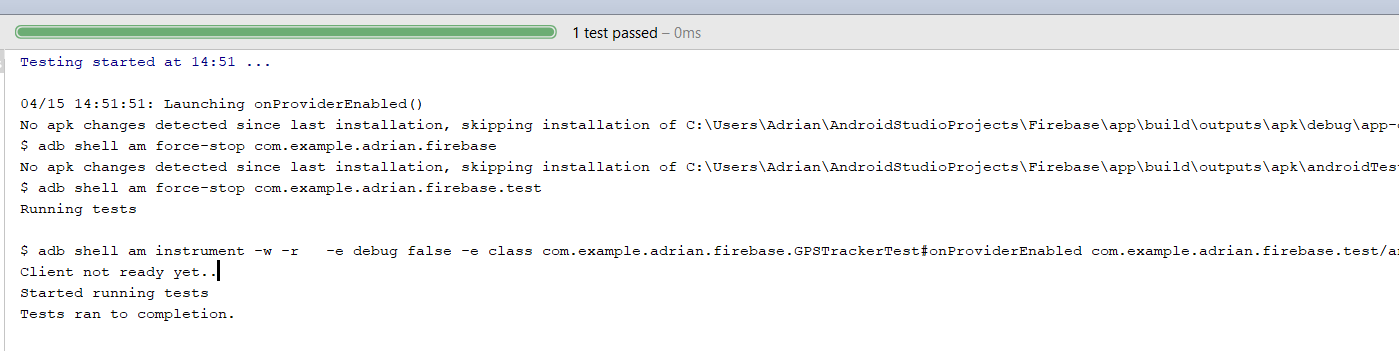


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **8** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When Location is on turned on the onProviderEnabled function will return false. | When Location is turned off the onProviderEnabled function will return false. | PASS | This test ensures there are no false signals sent by the device when there is no location breaking. |

JUnit.

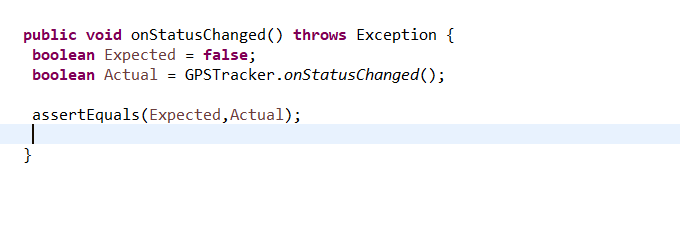


Visual Result.

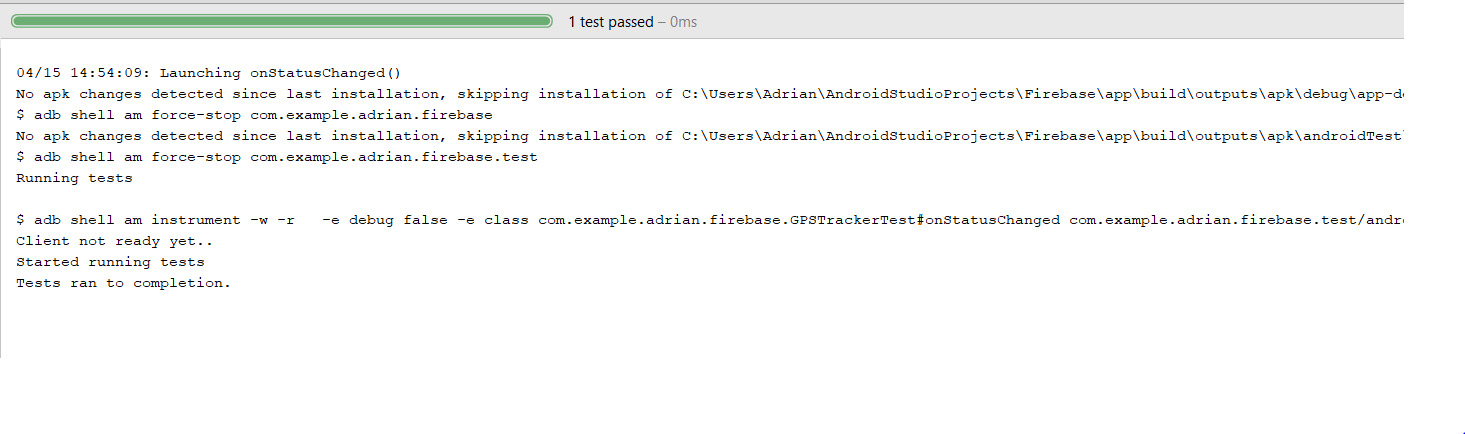


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **9** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the location is remaining in the same position, when executed the onStatusChanged function will return false. | Function will return false when the function is executed. | PASS | This will ensure that the location will not change when there is not change of displacement in the application. |

JUnit.

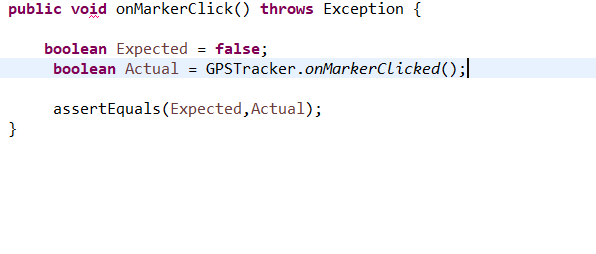


Visual Result.

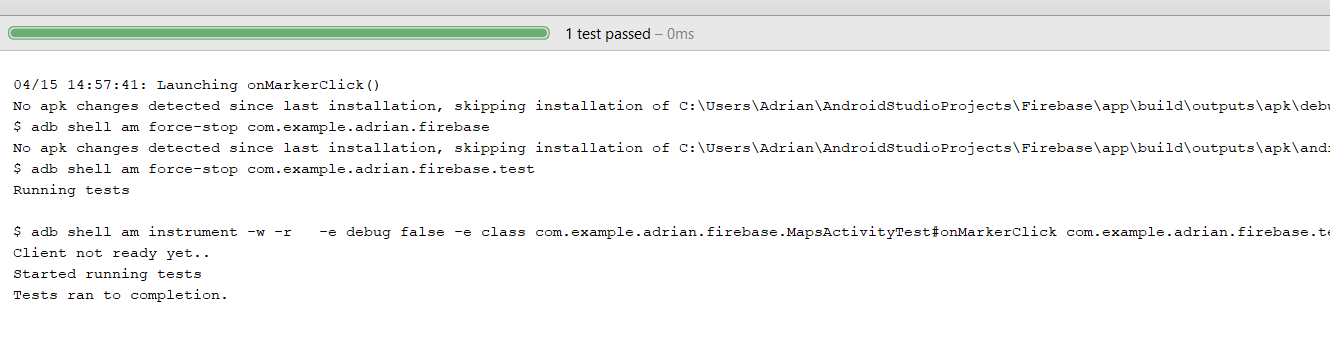


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **10** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the marker is clicked continue to next Activity | When the marker is clicked the process returns true | PASS | Marker returns True |

JUnit.

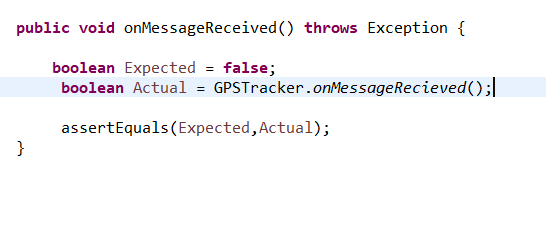


Visual Result.

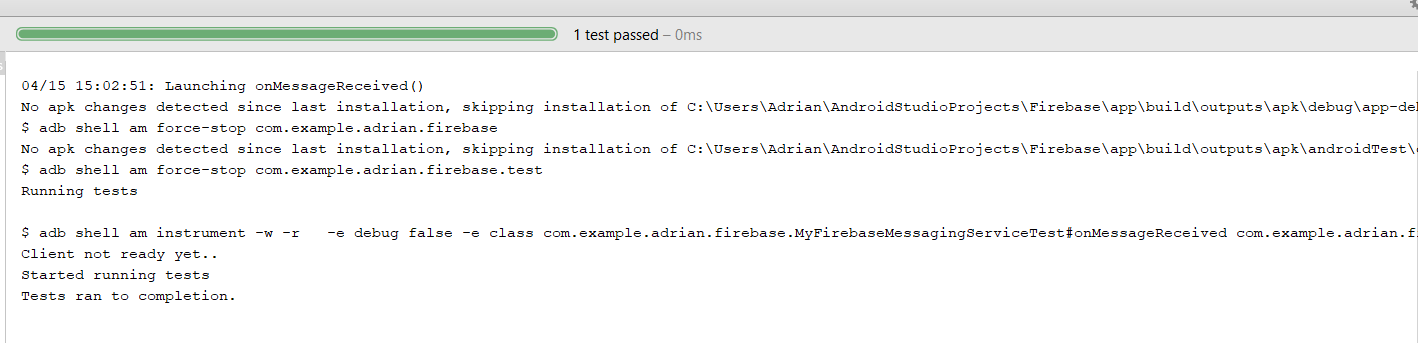


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **11** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the push notification is received the application will handle the incoming notification. | When the push notification is received the onMessageRecieved() function returns true. | PASS | The notifications will not block up and crash the application. |

JUnit.

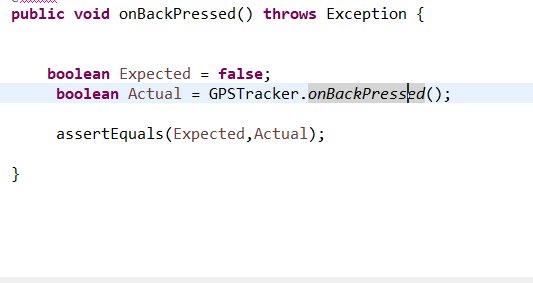


Visual Result.

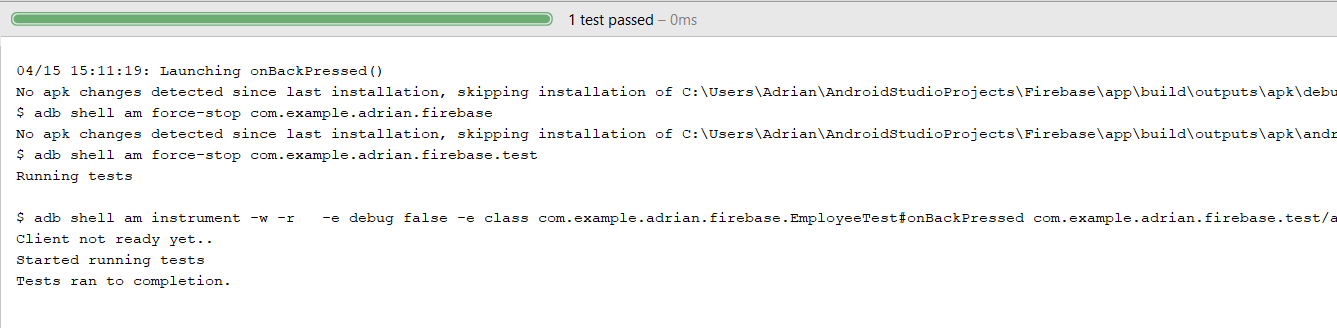


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **12** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the back button is pressed a return action is received. | onBackPressed will return true. | PASS | An action will be taken when the application returns. |

JUnit.

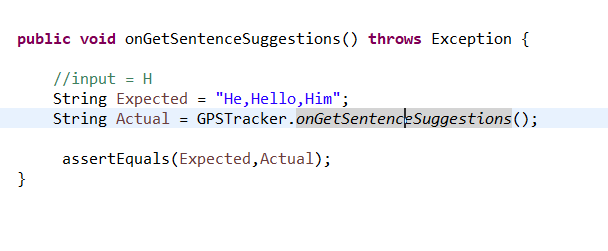


Visual Result.

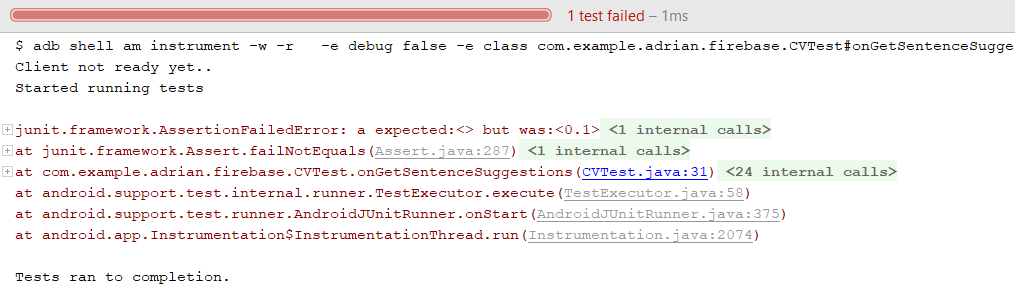


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **13** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | View all the sentence suggestion when the onGetSentanceSuggestions() function is called. | onGetSentanceSuggestions() returns true with the input as “H” with the expected output of “He,Hello,Him” | Fail | The output retuned false as it did nt consist of the String  “He,Hello,Him” |

JUnit.

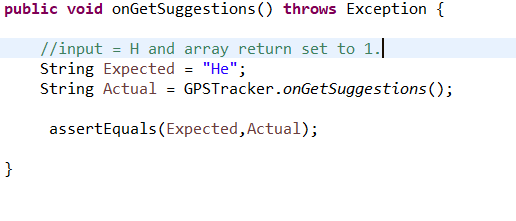


Visual Result.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **14** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | Get the first sentence suggestion that is returned with input “H”. | The first suggestion must be contain the String “He” when the onGetSuggestions() function is executed. | PASS | The output limited to 1 returned consisted of the String “he”. |

JUnit.

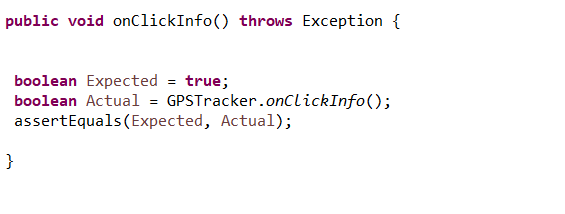


Visual Result.

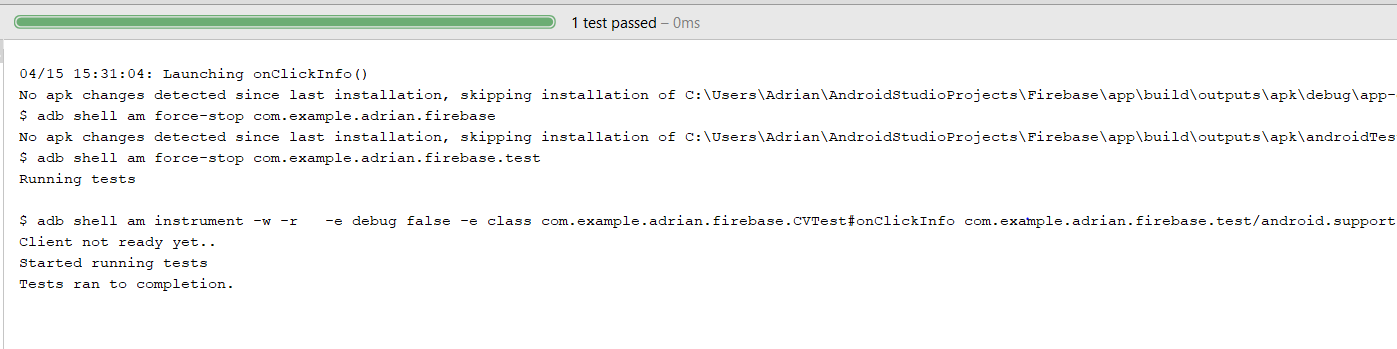


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **15** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the onClickInfo() function is called the output is returned as true. | The value of true is returned when the function is called. | PASS |  |

JUnit.

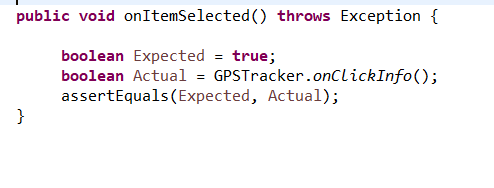


Visual Result.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **16** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | View Items when selected | OnItemSelected() returns true. | PASS | Function was implemented properly. |

JUnit.

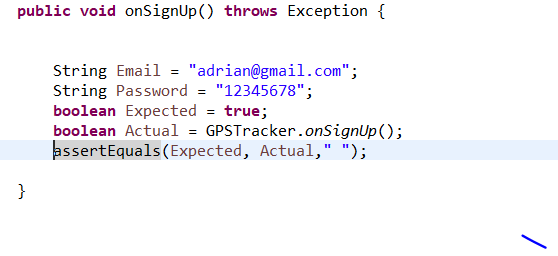


Visual Result.

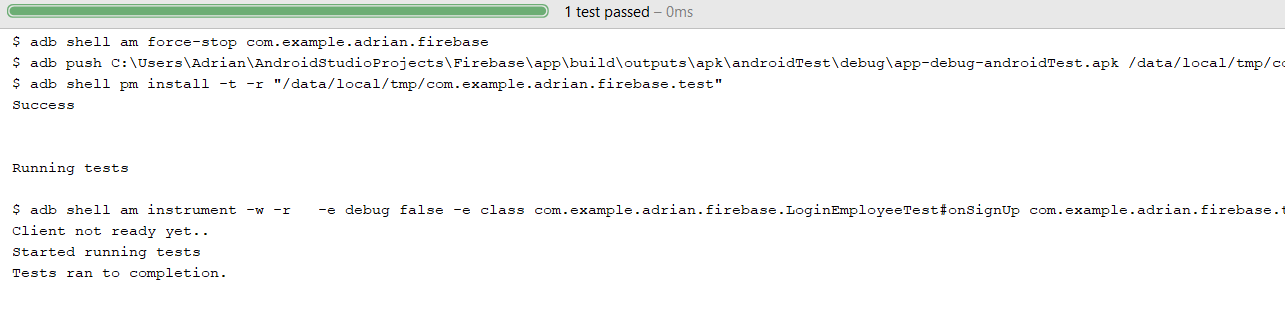


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **17** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When onSignUp() is executed the data will be inserted into the database. | When the correct parameters are passed in e.g valid gmail account. | PASS | The onSignUp() function is working correctly. |

JUnit.

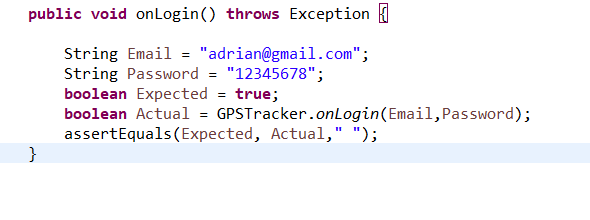


Visual Result.

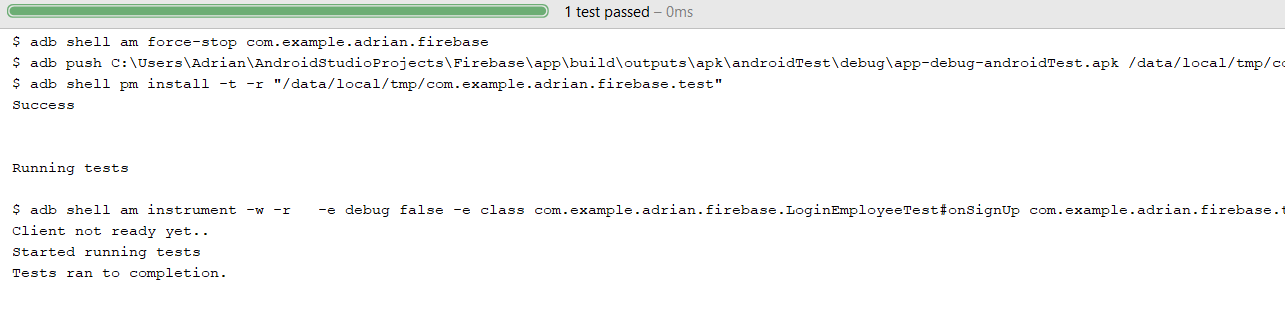


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **18** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the onLogin() function is excuted with the correct parameters the onLogin() function returns true. | onLogin() returns true. | PASS | The onLogin() function will return true with the correct parameters. |

JUnit.

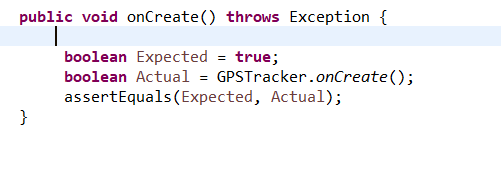


Visual Result.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **20** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | When the main onCreate() function is called in android all methods are executed. | All functions return true. | PASS |  |

JUnit.

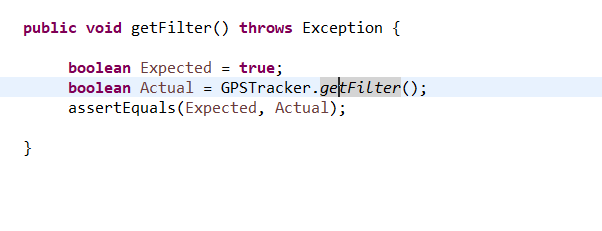


Visual Result.

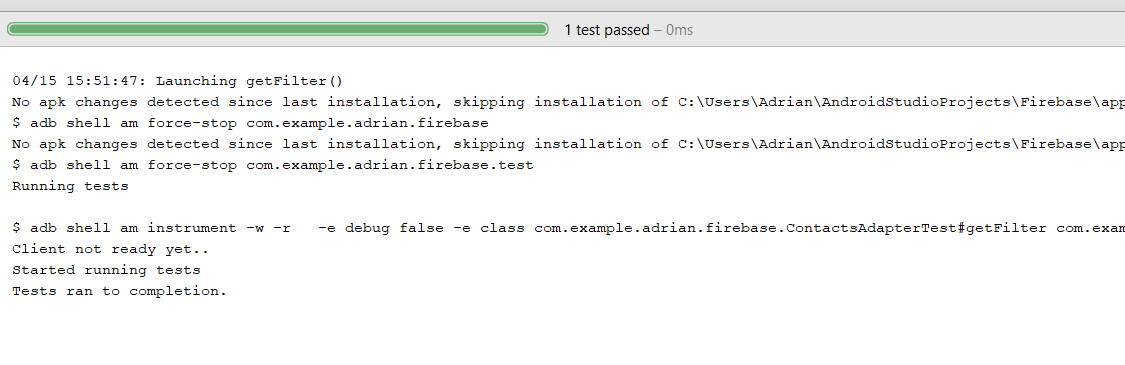


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case Id**  **21** | **Test case Description** | **Expected Result** | **Actual Result** | **Remarks** |
| 1 | The appropriate filter is returned when the getFilter() function is executed. | When the correct filter is received the function will return true. | PASS | The getFilter function is returning true. |

JUnit.



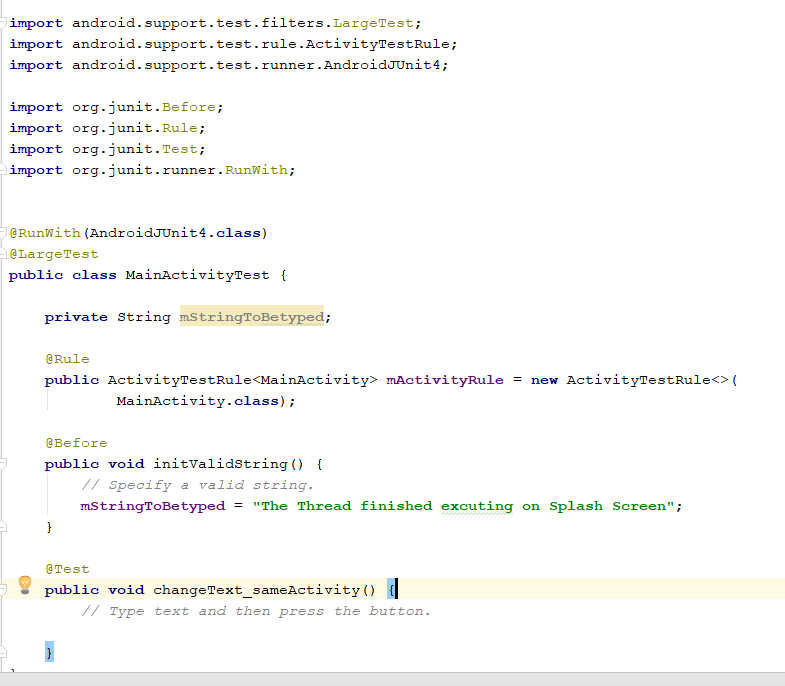
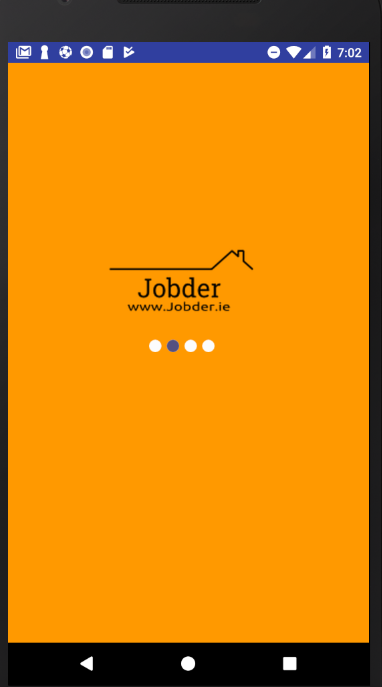
Visual Result.



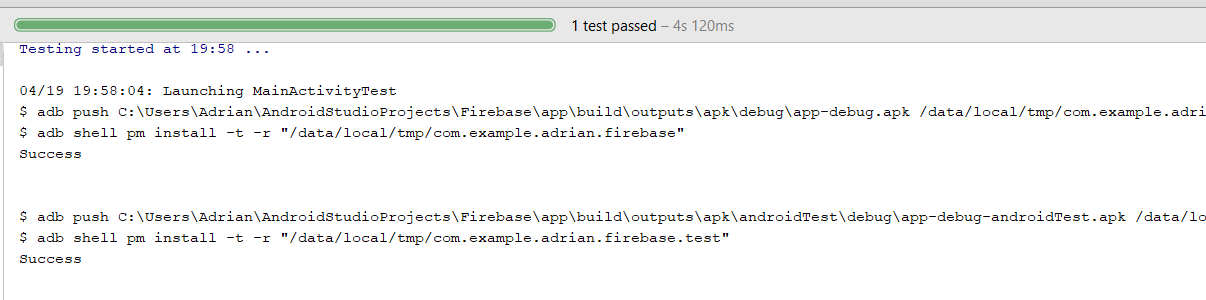
# Functional UI Testing (Expresso).

Here are samples of the following expresso tests I created to test the UI functionality of the application.

1. Testing The execution of splash screen.

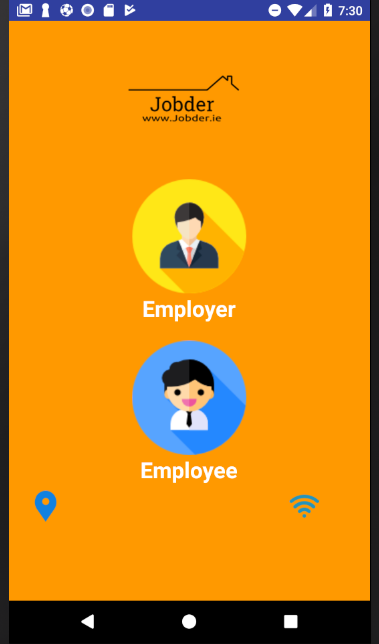
Test Result:



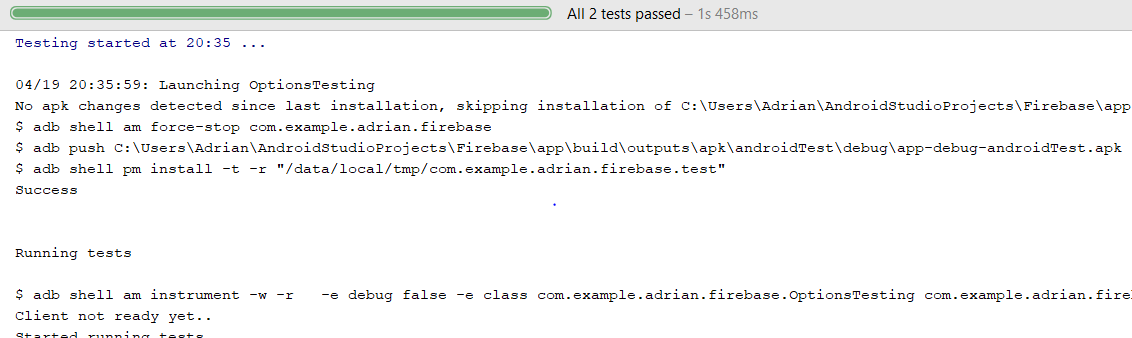
Conclusion:

When the application is Launched the Splash, screen runs correctly. The was very straight forward as the splash screen has no other dependencies and is set to a fixed time.

1. **Options Screen.**



**Test Result**

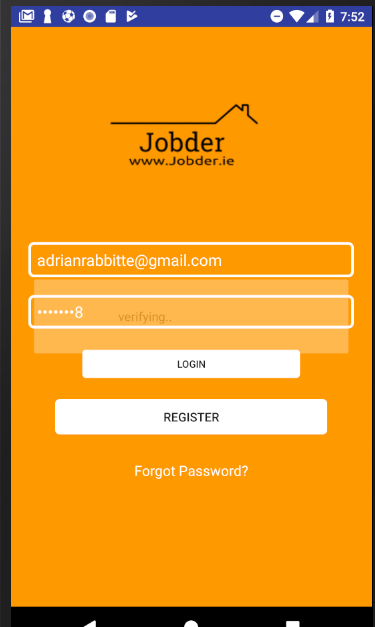


**Conclusion**

When the application is launched it is taken to the options page where the options page consists of two buttons as Image Views representing Employers and Employees. When the user clicks on their desired profession they are taken to their home screens of either the Employee and Employer respectively. Options page passed the Test as both buttons were operational when clicked.

**3)Login Employee Screen.**

**3)Login Employee Page.**

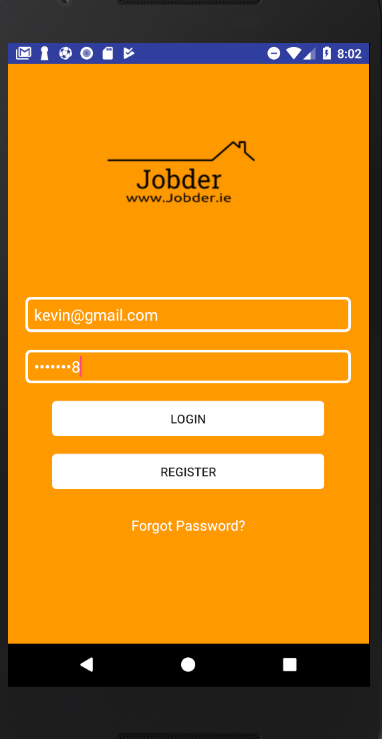


**Test Result**

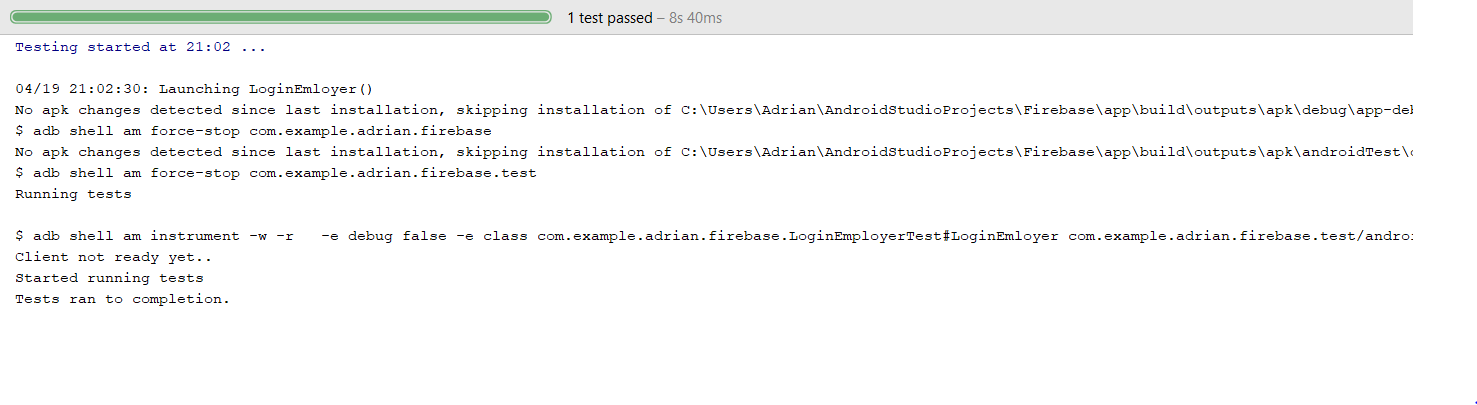


**Conclusion.**

When the user logins with the correct credentials the user is logged in successfully. This function is returned from the Firebase AUTH database.

**4)Login Employer Screen**

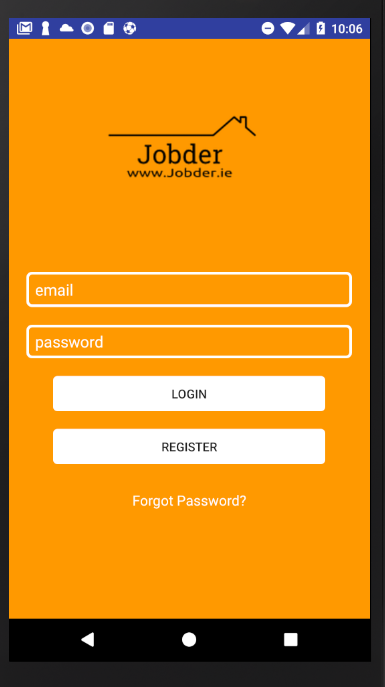
**Test Result**



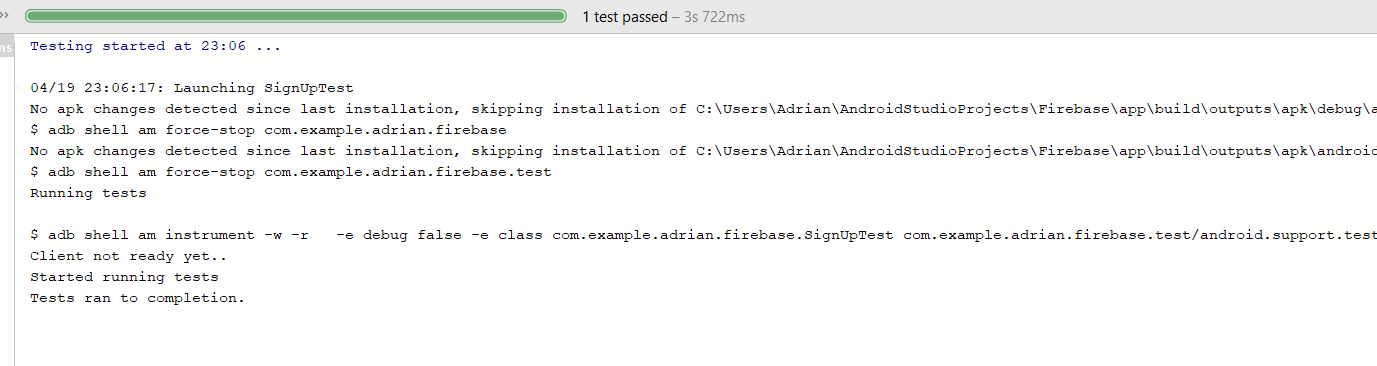
**Conclusion**

When the user logins with the correct credentials the user is logged in successfully. This function is returned from the Firebase AUTH database. When I tested the login function with [adrianrabbitte@gmail.com](mailto:adrianrabbitte@gmail.com) it failed as [adrianrabbitte@gmail.com](mailto:adrianrabbitte@gmail.com) was not a registered Employer.

5) Signup Button.

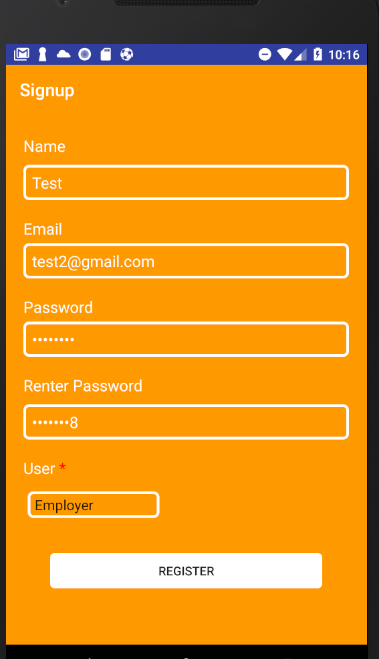
Test Result.



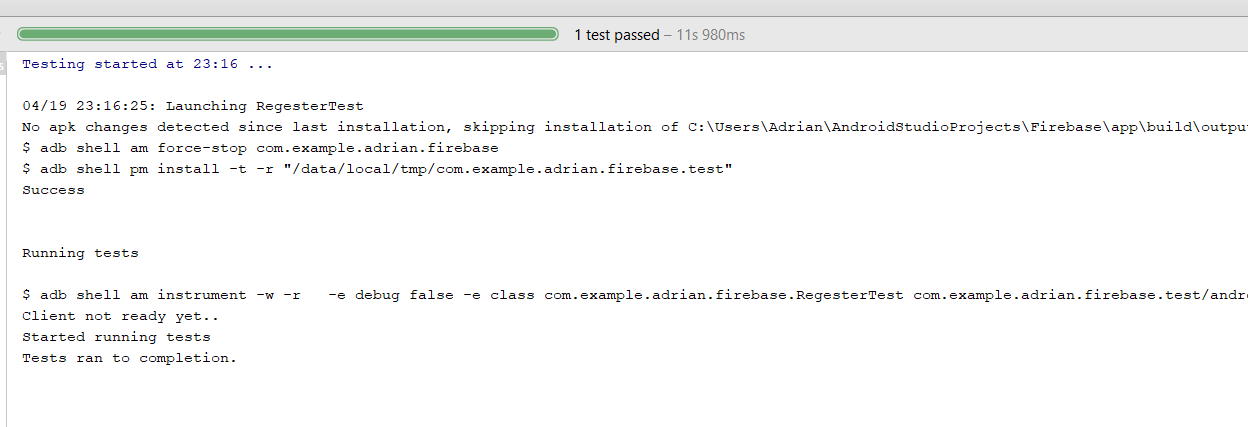
Conclusion.

When the Register button is clicked it brings you to the Register activity page where the user has the chance to sign up. This works correctly.

6) Register User



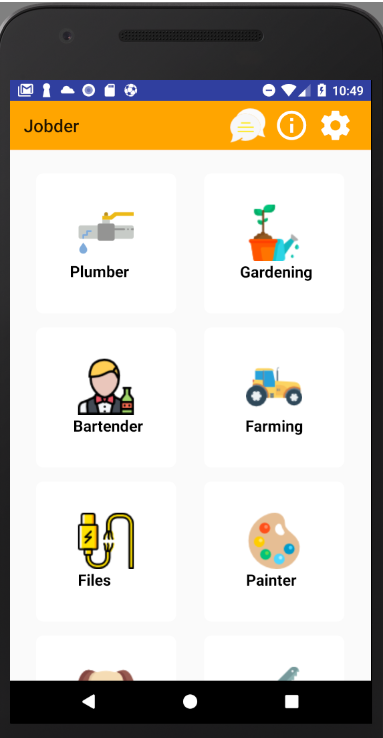
Test Result



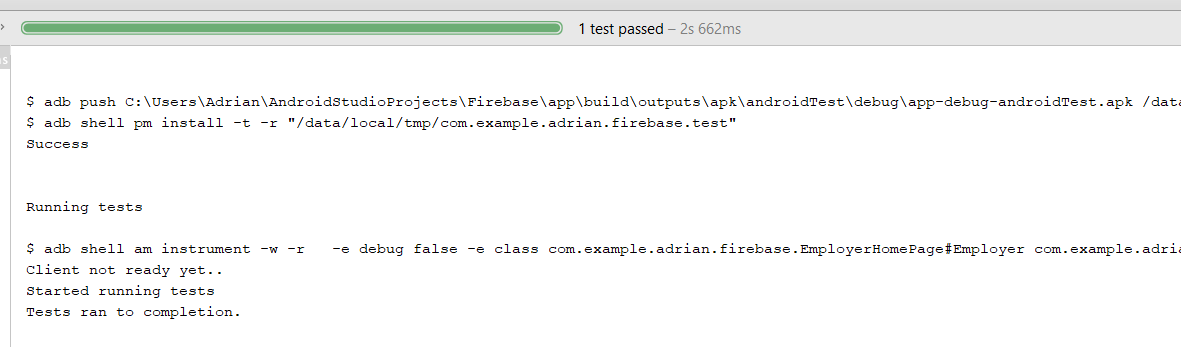
Conclusion.

When all fields are filled in the signup function registers the user successfully on the Firebase Server.

7) Employer Home Page.

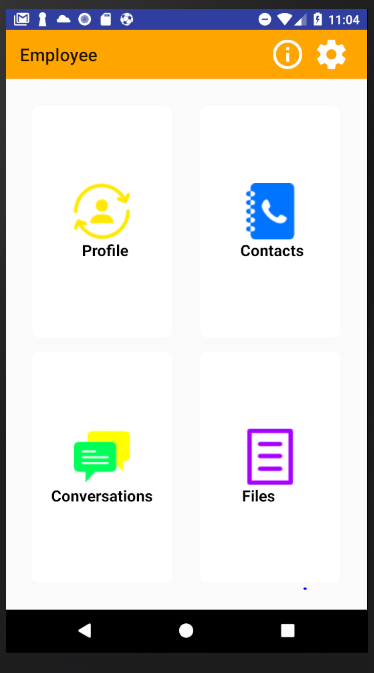
Test Result:



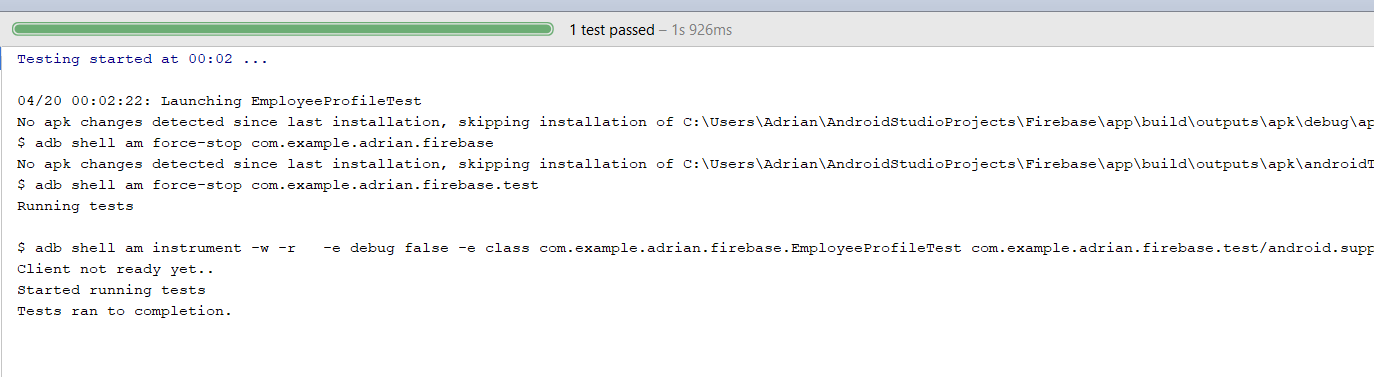
**Conclusion.**

In the Employer Home screen, I tested all the relative buttons, messages, info and settings. I also tested the Job search of plumber where I am brought to google Maps Api. Where the users are displayed whose profession is plumbing.

8) Employee Home Page.



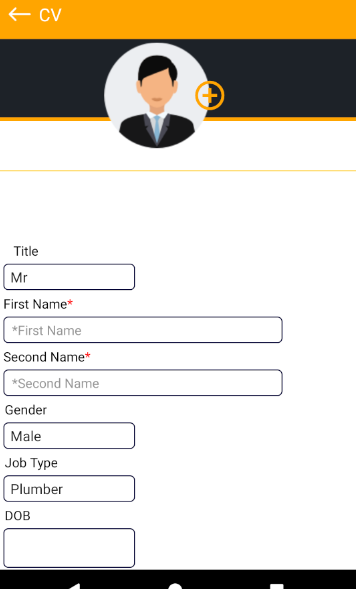
Test Result.



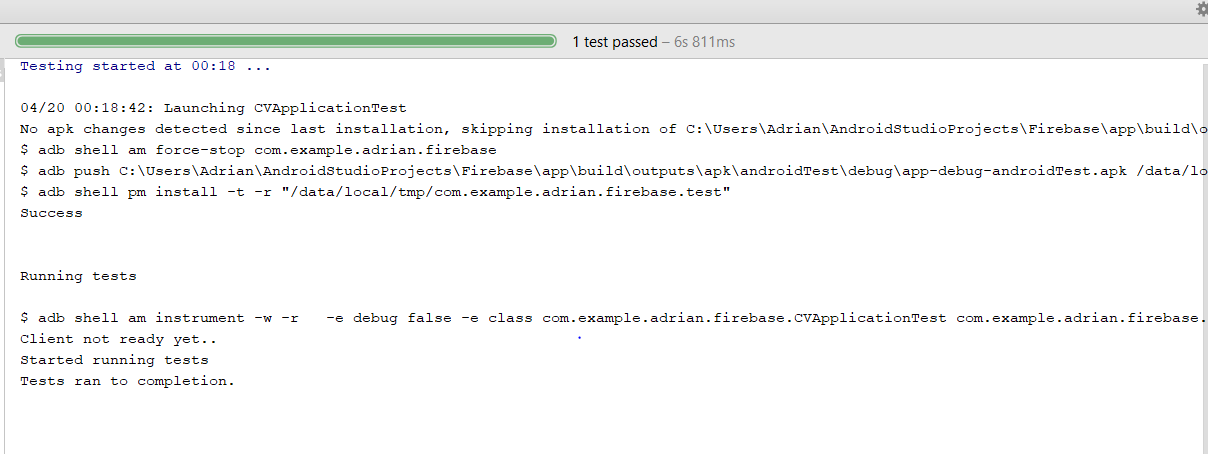
**Conclusion.**

Here all the functional buttons on the Employee Home Page were tested using expresso, all buttons are working correctly with each test passed.

9) CV Application Test.



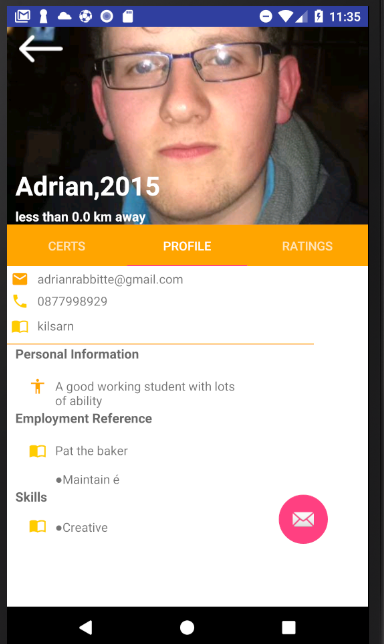
Test Result.



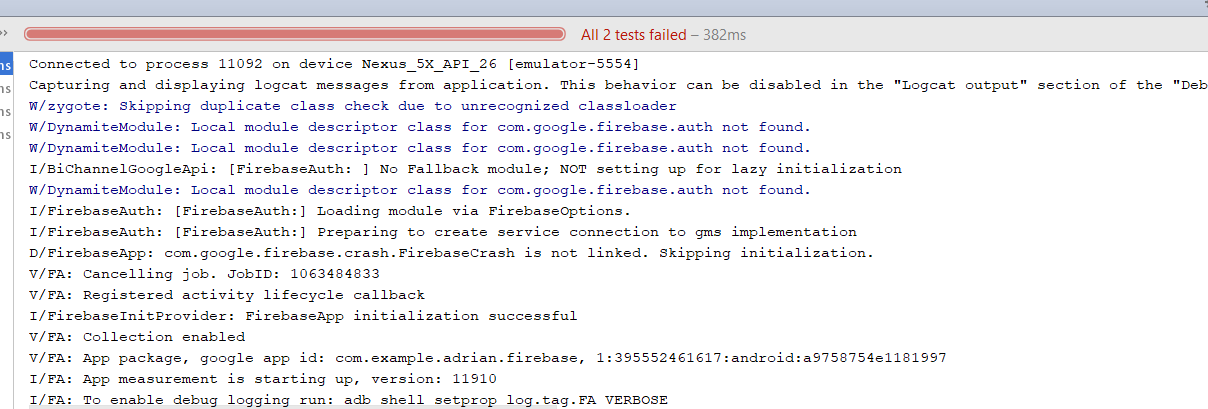
Conclusion.

The functionality of the CV activity was successful as when all fields were filled in it was successfully uploaded to the firebase server

10)Profile

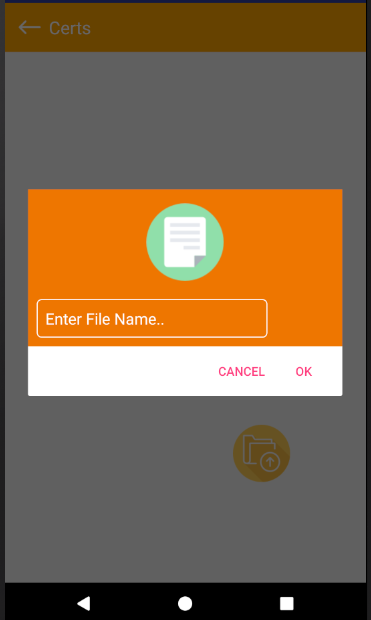
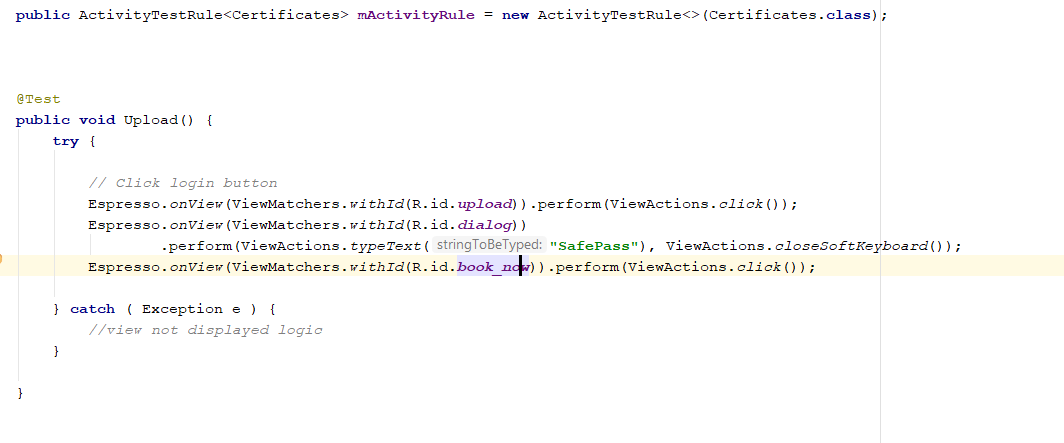
Test Result.



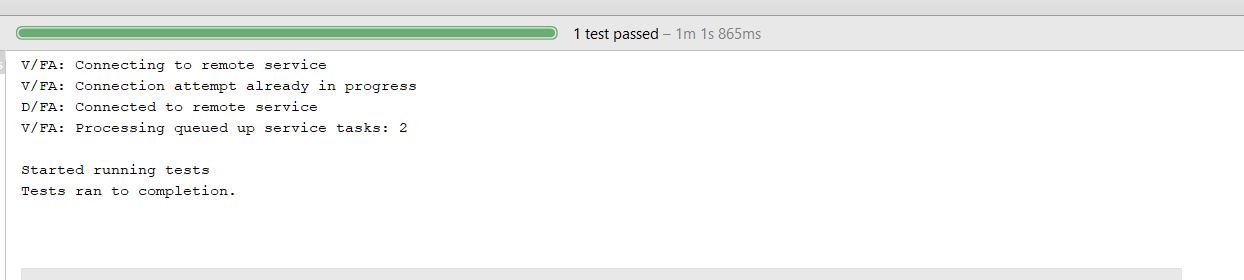
**Conclusion.**

When performing actions of swipe left and swipe right in the android application, I got a profound error. After studying this problem I realised that the context of the page viewer was established under Main2Activity.class by replacing this with getApplicationContext() got the current context of the application and allowed the page viewer to slide right and left.

11) Uploading Images from Camera Roll.



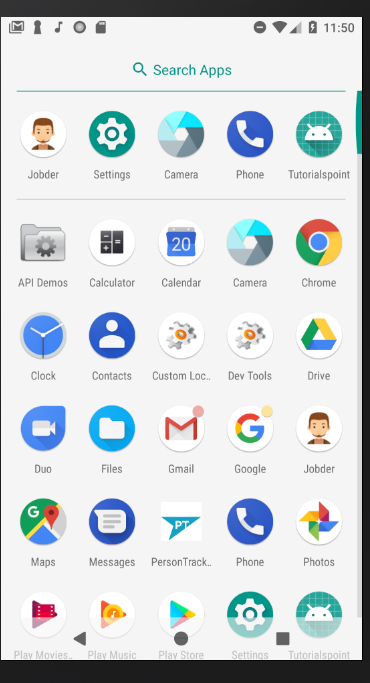
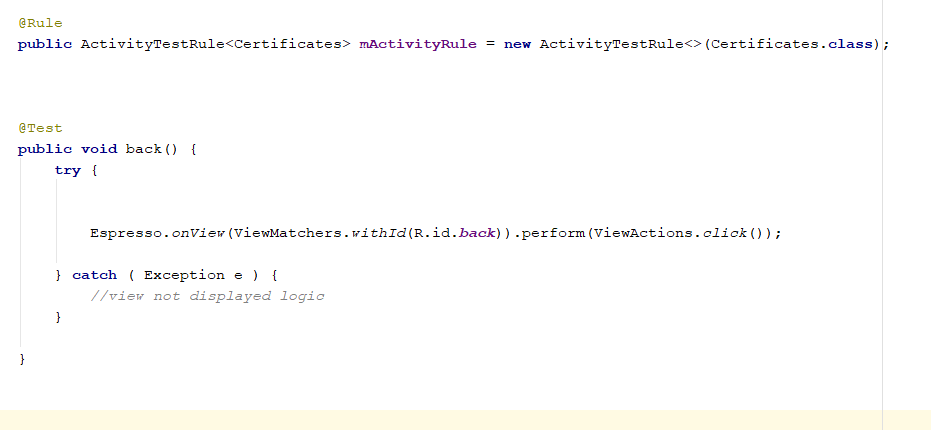
Test Result:



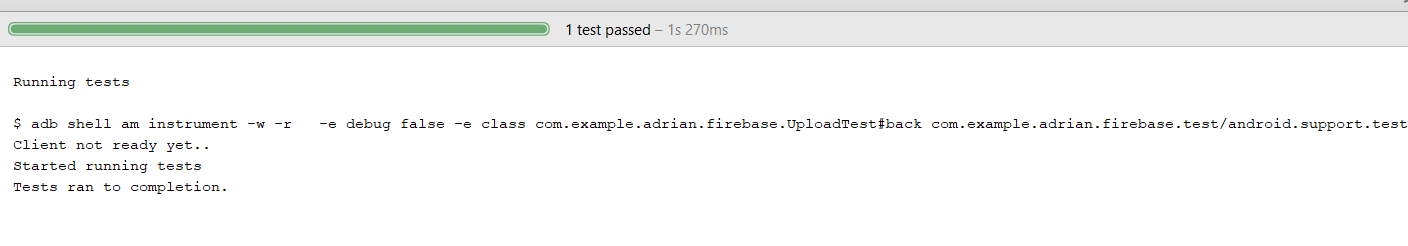
**Conclusion.**

The functionality of the upload button is implemented where the user can upload a file. The implementation is working correctly, and the user is able to upload.

12) Back Button for Each Activity.



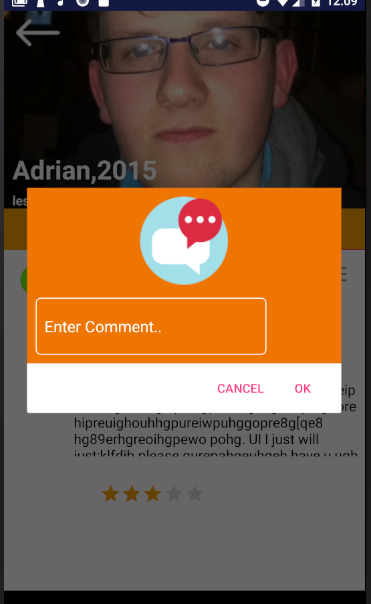
**Test Result.**



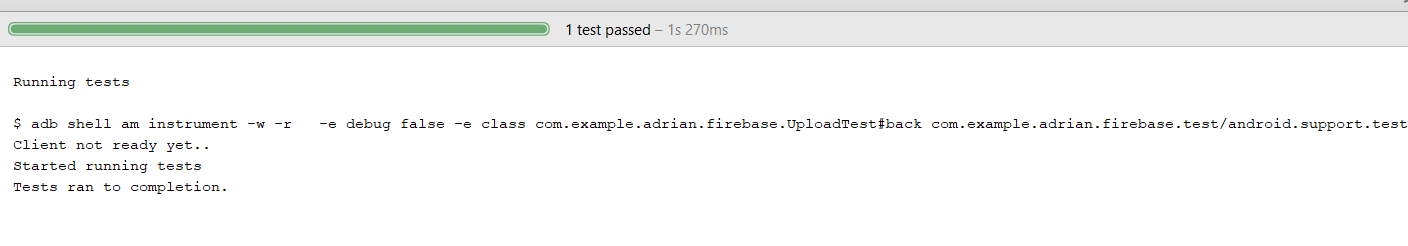
Conclusion.

When The back button is pressed on certain Activities the application is killed returning to its previous state.

**13) Upload Rating for Employee.**



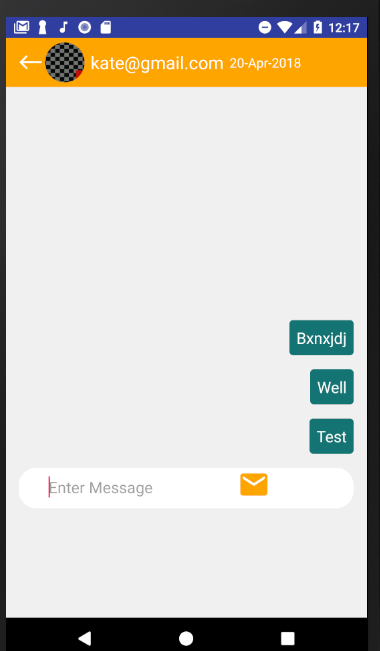
**Test Result:**



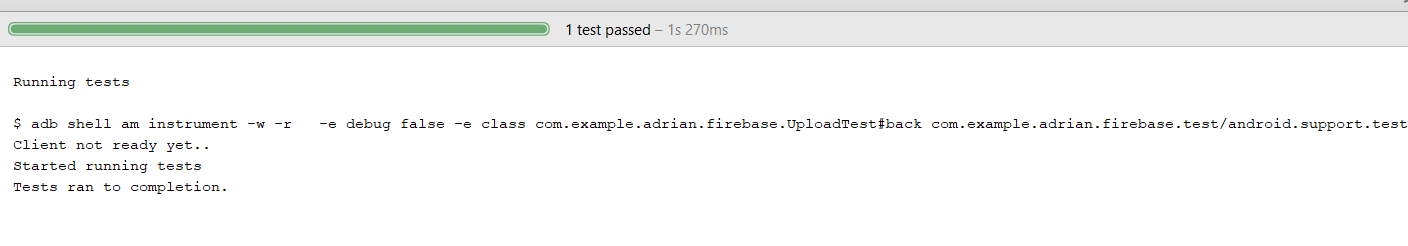
Conclusion:

When the Employer creates a review a rating pop up dialog appears where the user can enter their details.

**14) Users Sending Messages.**



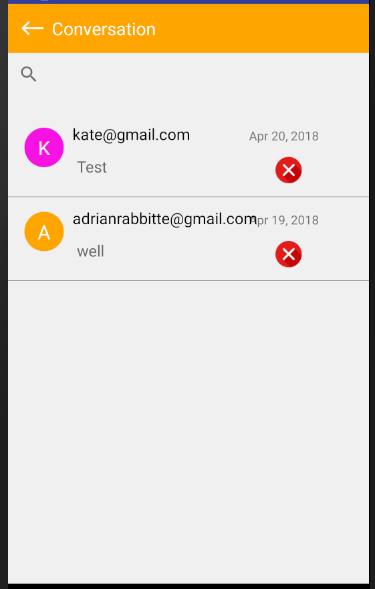
**Test Result**



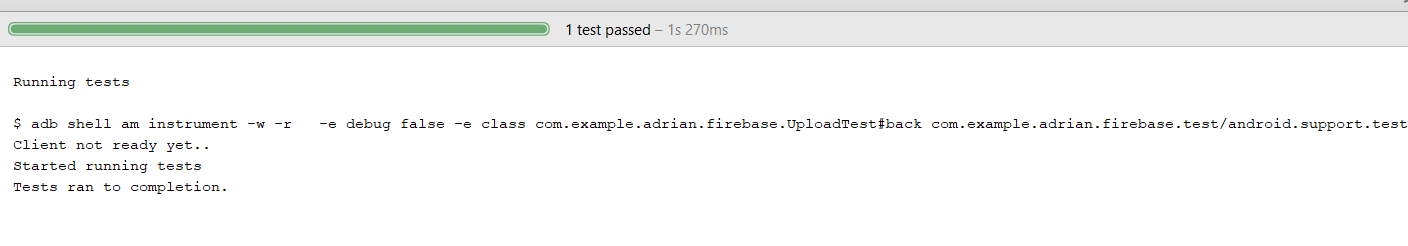
Conclusion:

This test validates the functionality of the messaging in jobder where the user can enter a simple message.

**15) All Conversations appear for User.**



**Test Result**



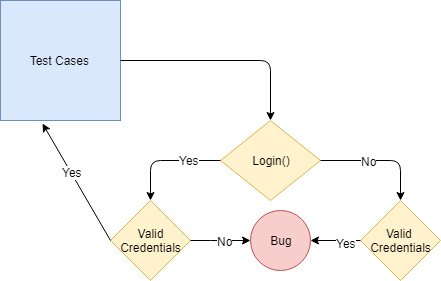
Conclusion

When the Employer and Employee view their test, the conversations will be loaded without the applications crashing.

**Different Test Cases Applied with Expresso.**

**Login**

When testing my login in the android application using Expresso I followed several steps when entering different testcases. Here is an example a data flow diagram that I used to find errors during the testing cycle.



The Test Cases that I used for the Login were as follows:

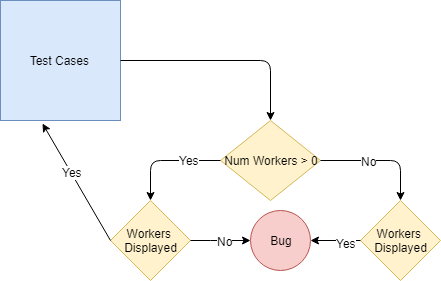
1. Valid Email and Valid Password
2. Invalid Name and Valid Password
3. Valid Name and Invalid Password
4. Invalid Name and Invalid Password

Results:

Test Case 1,2 and 4 returned true, However I was getting a bug with Test Case 3 as a value with an invalid password was registering as true. This was due to the password length which was easily resolved by removing an edit Text and replacing with a correct Password field.

**Show Workers (e.g. Plumbing)**

When the Employer Clicks on a field of work e.g. Plumbing they will be brought to a google Maps Activity where the workers will be shown as long as the Number of workers are greater than 0.



**Test Cases:**

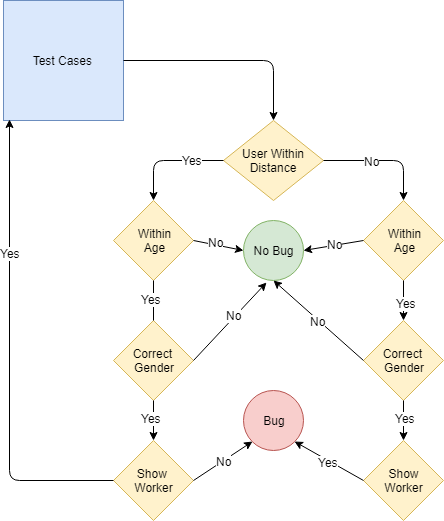
1. Plumbers = 2
2. Gardening = 3
3. Farming = 1
4. Painter = 0

**Results:**

Test Case 1,2 and 3 returned true. When test case 4 was executed the application still went to the maps activity. After studying the code, it would be very difficult to avoid this due to the structure of reference. Instead when a field of employment is clicked on and there are no workers available a toast message is created to display to the user that there are no available workers.

**Maps**

When an Employee clicks on a field of employment they are brought to the google maps activity where the workers are displayed. However, it is the responsibility of the Maps Activity to display the workers according to the Employers Settings and the Employees Settings. E.g. Distance, Show Profile and Gender.



These Test cases are run against the following settings.

{age = < 40, distance < 60 km, Gender = Male}

**Test Cases:**

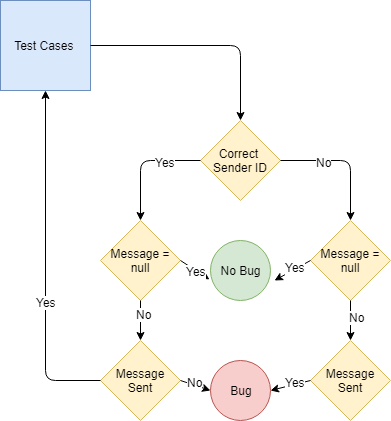
1. Male,30,20km
2. Female,20,15km
3. Male,50,10km
4. Female, 30,100km

**Results:**

Test Cases 1,2,3,4 returned true, false, false and true respectively this returned correct for the testcases. The functionality is much more descriptive but breaking the system down into Yes/No results made it easier to test.

**Send Message**

When a User of Jobder wishes to send a message, they can do so by simply typing into the interface of the chat application Activity. To provide correct communication there had to be several tests passed.



**Test Cases:**

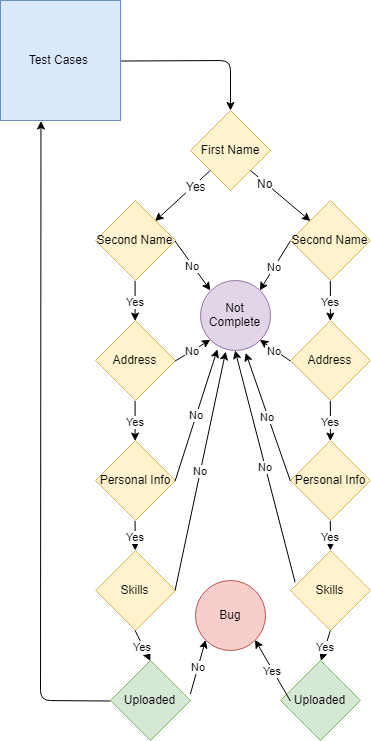
1. Valid Sender ID, Message
2. Valid Sender ID, Null
3. Invalid Sender ID, Null
4. Invalid Sender ID, Messag

**Results:**

All the test cases past the tests leaving proving that there are no invalid or null messages sent. However, for Test Case 3 there was no account as there was no user registered in the database and the system was trying to employ two null objects.

**Uploading CV**

When the user uploads a CV to the System there are fields that are mandatory and optional. For the layout of the application Jobder these mandatory fields must not contain null values. The Fields that must be filled in the application are First Name, Second Name, Address, Personal Description and Skills.



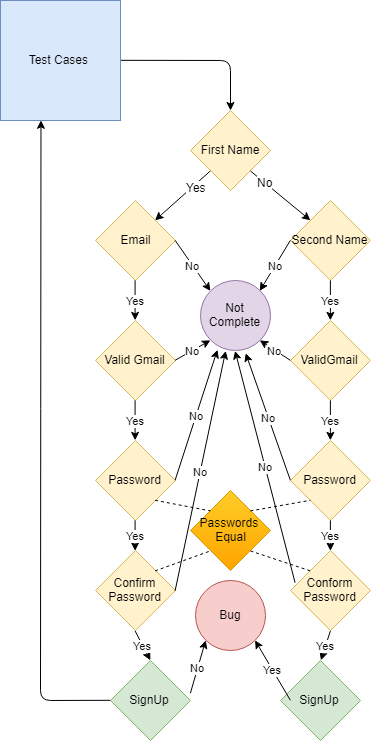
**Test Cases:**

1. First Name, Second Name, Address, Personal Info, Skills
2. Null, Second Name, Address, Personal Info, Skills
3. First Name, Null, Address, Personal Info, Skills
4. First Name, Second Name, Null, Personal Info, Skills
5. First Name, Second Name, Address, Null, Skills
6. First Name, Second Name, Address, Personal Info, Null
7. Null, Null, Null, Null, Null
8. **Test Results:**

Test case 5 was the only Test that failed this was a very simple mistake that I created before uploading the info in the edit Text, I was checking that if the edit Text was empty do not proceed with the upload however the hint in the textbox cancelled this error out.

**Sign Up**

When a User of Jobder wishes to sign up the must complete four fields Name, Email, Password and Conform Password. All fields must be filled in to create a successful login in the firebase NoSQL database. When the user enters their password, they must be the exact same in both fields (Password and confirm Password)



**Test Cases:**

1. Name, Email, Password, Renter Password
2. Null, Email, Password, Renter Password
3. Name, Null, Password, Renter Password
4. Name, Email, Null, Renter Password
5. Null, Null, Null, Null

**Test Results:**

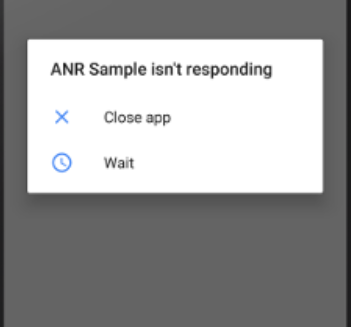
All the test cases were passed as expected as I had already performed user testing during the development process.

# Performance Issues and Resolution.

**Application Not Responding (ANR)**

This was an encountered error that I was receiving throughout the development process of the application. This happens in the android application when the main UI thread of the app is blocked or there is too much processes trying to run. It was not until mid-way through the development process that I started to see this. The reason I received this error was when my personal phone which was a Huawei p8 Lite which had a Octa-core 1.2 GHz Cortex-A5 and 3 GB of Ram which had more than enough power to run the application which broke. I then ran the application on an old Samsung galaxy although it was a powerful phone due to its age it was not able to handle all the process running on the main UI thread which caused the application to display an application not responding (ANR).

e.g. of an ANR.



**Resolution.**

The main areas that I was receiving ANR was when I was downloading data (especially images) and uploading files from the actual devices. This was because the application was trying to overcome a number of processes at a time. The most effective way to overcome this problem was a Worker Thread. The best way to create a worker thread is with AsyncTask class. This extends AsyncTask and implements the doInBackGround() method to perform the work. However by just using ONActivityResult() stopped the thread until the data was available which eliminated the ANR.

**OOM (Out Of Memory Error)**

The main reason for receiving OOM error is due to Memory Leaks. This happens when a program to release discarded memory causing impaired performance or Failure. The main cause of Memory Leaks is the Context of the object. Every Application has a registered activity which can be established by getApplicationContext() here relevant information of that activity is stored.

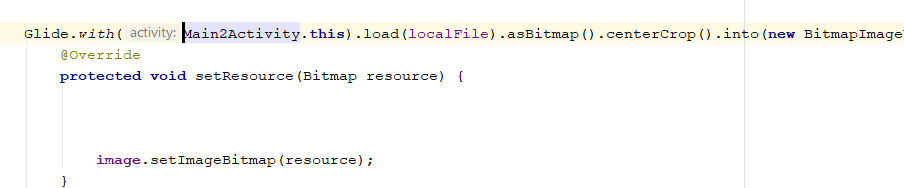
Here is an example of the memory measure of the application when there is an OOM.

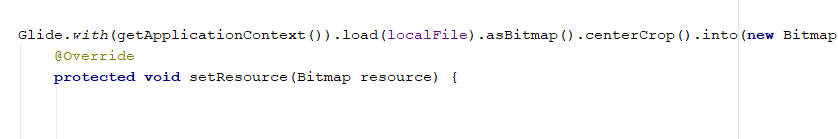


**Resolution.**

To avoid memory leaks I avoided passing context objects into activities and replace the actual Context name with getApplicationContect().

e.g.





# Stress Testing in Android Using Monkey.

Monkey is a program that runs on the Emulator or Device, the program generates random pseudo streams that of user’s events such as clicking touches and gestures. I am using this to stress test the application as it will give a good indication of all the percentage of actions that can be performed on the application. Monkeys main features include:

* Basic Configuration options.
* Operational Constraints.
* Event types and Frequency.
* Debugging options.

Here is the command which is launched inside platform tools.

adb shell monkey -p com.example.adrian.firebase -v 500

(sumerised)

adb = Android Debug Bridge

shell = The Interface on the Device.

Monkey = testing tool

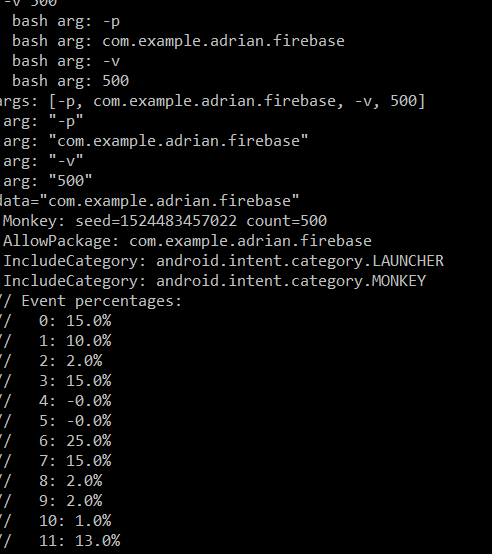
adb shell monkey -p com.example.adrian.firebase -v 500 = My application package

v = verbose method

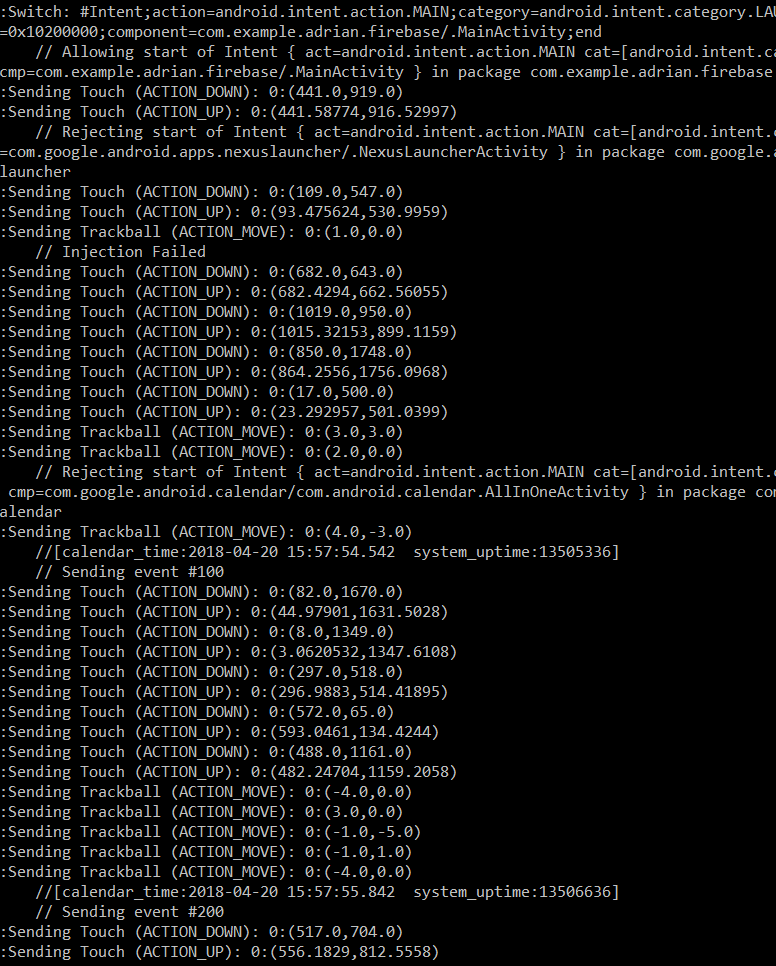
500 = number of test events.

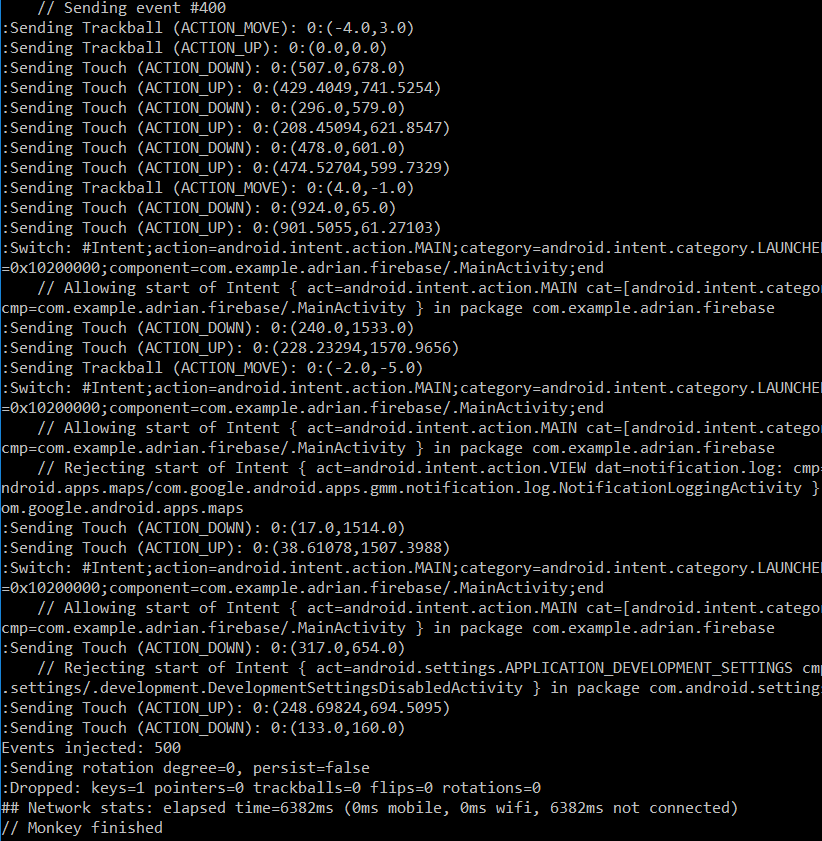
On the following page shows the Results of the Stress Test executed.

1. Launching



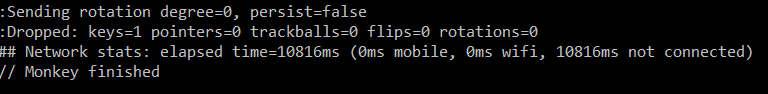
1. Results





**Analysis**

After I preformed the Stress test using monkey it was successful. The results show the different actions that the pseudocode code preformed with the emulator without the application crashing. I also underwent this test when the application was logged in as the Employer and the Employee respectively.

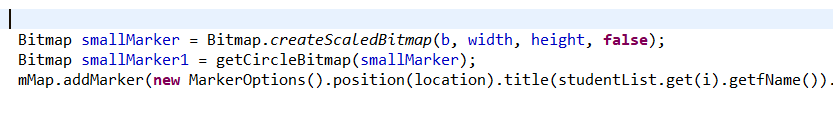
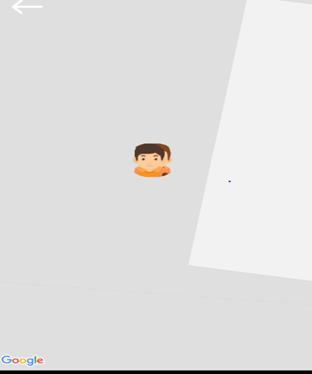


The Stress test stops when the application data access is removed. This is because a lot of the applications sources are dependent on a third party (The Server).

I felt that the overall result was not fully accurate for my application as I was not able to stress test different events such as the number of users using the application and the number of profiles displayed in the same area without the application crashing. Therefore, I created my own small java script to display more than 1 user in the exact Location without the application braking.

The map is hosted by google maps API which is displayed using a specified google key. The images(emojis) are displayed on the map as bit images which are received and filtered from different users within the database according to their individual latitude and longitude values.

I created a simple for loop for values less than and set the location as a static value of latitude = 34.1786998 and longitude = -86.6154153. This enabled the application to place two bitmaps in the exact same location.

****

# User Testing:

I used future users of the application to test the android application Jobder. I ranged my users from different professions and different ages.

Here are the following instructions that were giving to the user to test the application.

The Following instructions are for Employees

|  |  |
| --- | --- |
| **Task** | **Task Description** |
| 1 | Open the application and Sign Up as an Employee. |
| 2 | Login with unique User Name and Password. |
| 3 | Open Profile. |
| 4 | Upload a Profile Picture. |
| 5 | Fill in all fields. |
| 6 | Save Profile. |
| 7 | Open Files |
| 8 | Upload relevant Documents. |
| 9 | Delete Selected Documents. |
| 10 | Open Conversations. |
| 11 | View Conversations. |
| 12 | Select Conversation. |
| 13 | Create Message. |
| 14 | Send Message. |
| 15 | Select Info (View Information of the application. |
| 4 | Select Settings |
| 5 | Set Distance Range |
| 6 | Set Notifications (ON/OFF) |
| 7 | Set Visibility (On/OFF) |

These set of instruction displays the purpose of the application and different possibilities in range of user for the Employee side of the application. By following these sets of instructions, the user will have tested all the main functionality of the application. The Employees settings is difficult judge as this is a very dominant feature to determine the visibility of the application.

Here are the List of Users for the Employee Application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test User | Name | Age | Gender | Occupation |
| 1 | Shane Shiles | 21 | Male | Student |
| 2 | Dermot Rabbitt | 24 | Male | Student |
| 3 | Colm Brady | 26 | Male | Plumber |
| 4 | Joanne Rabbitte | 21 | Female | Gardening |
| 5 | Sean Mckeogh | 28 | Male | Electrician |
| 6 | Daniel O Reilly | 27 | Male | Farmer |
| 7 | David Shields | 39 | Male | Builder |
| 5 | Sarah Rabbitte | 24 | Female | Hairdresser |
| 6 | Padraig Tierney | 22 | Male | Farmer |

Here are the following instructions that were giving to the user to test the application.

The Following instructions are for Employees

|  |  |
| --- | --- |
| **Task** | **Task Description** |
| 1 | Open the application and Sign Up as an Employee. |
| 2 | Login with unique User Name and Password. |
| 3 | Click Field in search of employment. e.g. Plumbing |
| 4 | Select User |
| 5 | View Profile. |
| 6 | View Certificates. |
| 7 | View Ratings. |
| 8 | Create Rating. |
| 9 | Submit Rating. |
| 10 | Open Message. |
| 11 | Create Message. |
| 12 | Send Message. |
| 13 | Back to Home Screen. |
| 14 | Open Conversations. |
| 15 | Select Conversations. |
| 16 | Create Message. |
| 17 | Send Message. |
| 18 | Delete Conversations |
| 19 | Open Settings. |
| 20 | Filter Gender. |
| 21 | Filter Distance. |
| 22 | Filter Age |
| 23 | Re Visit Home Page and View Jobs. |

These set of instruction displays the purpose of the application and different possibilities in range of user for the Employer side of the application. By following these sets of instructions, the user will have tested all the main functionality of the application.

Here are the List of Users for the Employer Application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test User | Name | Age | Gender | Occupation |
| 1 | Conor Madden | 25 | Male | Office Worker |
| 2 | Eamonn Rabbitte | 52 | Male | Farmer |
| 3 | Kevin Rielly | 26 | Male | Plumber |
| 4 | Thomas Kiernan | 40 | Male | Farmer |
| 5 | James Rielly | 26 | Male | Office Worker |
| 6 | Enda O Rielly | 25 | Male | Electrician |
| 7 | Enda Rabbitte | 24 | Male | Student |

**User Feed Back:**

**Employees:**

|  |
| --- |
| **ID:**1  **Name:** Dermot Rabbit  **Age:** 24  **Occupation:** Student |
| **Issues:**  **Comment:**  Could not sign in with email address, was getting error saying Please use correct Email.  UI very simple and easy to navigate, Settings very easy to understand highlighting the important features. |

|  |
| --- |
| **ID:2**  **Name:** Shane Shields  **Age:** 21  **Occupation:** Student |
| **Issues:** Back Click from Info Page brings me to Employer Page  **Comment:**  Very easy to navigate through Employee Page. |

|  |
| --- |
| **ID:3**  **Name:** Colm Brady  **Age:** 26  **Occupation:** Plumber |
| **Issues:** No Issues  **Comment:**  Can the Employee add users to the system? |

|  |
| --- |
| **ID:4**  **Name:** Joanne Rabbitt  **Age:** 21  **Occupation:** Gardening |
| **Issues:** No Issues  **Comment:**  Can the distance bar be set further than 100km? |

|  |
| --- |
| **ID:5**  **Name:** Sean Mckeogh  **Age:** 28  **Occupation:** Electrician |
| **Issues:** No Issues  **Comment:**  If the user need to upload a pdf, can he? |

|  |
| --- |
| **ID:6**  **Name:** Daniel O Reilly  **Age:** 27  **Occupation:** Farmer |
| **Issues:** Can’t login with Email  **Comment:**  Won’t let me login with user name as I don’t have an email. |

|  |
| --- |
| **ID:7**  **Name:** David Shields  **Age:** 39  **Occupation:** Builder |
| **Issues:** No Issues  **Comment:**  Very Easy to navigate. |

|  |
| --- |
| **ID:8**  **Name:** Sarah Rabbitt  **Age:** 24  **Occupation:** Hair Dresser |
| **Issues:** No Issues  **Comment:**  Very Easy to navigate. |

|  |
| --- |
| **ID:9**  **Name:** Padraig Tierney  **Age:** 22  **Occupation:** Farmer |
| **Issues:** Cant Sign Up.  **Comment:**  Could not sign up as I have no email address. |

**User Feed Back:**

**Employers:**

|  |
| --- |
| **ID:1**  **Name:** Conor Madden  **Age:** 25  **Occupation:** Office Worker. |
| **Issues:** No Issues  **Comment:**  Very easy application if in search of workers. |

|  |
| --- |
| **ID:2**  **Name:** Eamonn Rabbitt  **Age:** 52  **Occupation:** Farmer |
| **Issues:** Can’t login with Email  **Comment:**  Won’t let me login with user name as I don’t have an email. |

|  |
| --- |
| **ID:3**  **Name:** Kevin Reilly  **Age:**26  **Occupation:** Plumber |
| **Issues:** No Issues  **Comment:**  Easy to establish ratings. |

|  |
| --- |
| **ID:4**  **Name:** Thomas Kiernan  **Age:** 40  **Occupation:** Farmer |
| **Issues:** No Issues  **Comment:**  When a user registers in that area will it update automatically. |

|  |
| --- |
| **ID:5**  **Name**: James Rielly  **Age:** 26  **Occupation:** Office Worker |
| **Issues:** Cant Sign Up.  **Comment:**  Wont let me use my current Email address. |

|  |
| --- |
| **ID:6**  **Name**: Enda O Reilly.  **Age:** 25  **Occupation:** Office Worker |
| **Issues:** No Issues  **Comment:**  Very easy to use application. |

|  |
| --- |
| **ID:6**  **Name**: Enda Rabbitt.  **Age:** 24  **Occupation:** Office Student |
| **Issues:** No Issues  **Comment:**  Very easy to use application, simple Interface. |

**Analysis of User Cases:**

From the User cases the main issue that was highlighted was the Sign up. There was 5 people out of the 16 unable to login, this was due to with two having no email address and three with email address from different domains. For the Sign-up using Jobder the email address must be a Gmail account e.g. “[xxxxxxxx@gmail.com](mailto:xxxxxxxx@gmail.com) where a different address e.g. [xxxxxxxx@hotmail.ie](mailto:xxxxxxxx@hotmail.ie) would not be applicable. As the application stands it is to late to change this feature, so to overcome this problem I have clearly stated in the login headings that the account must be Gmail.

From the overall feedback the User Interface of the application is very positive receiving very effective results from the user testing. Eight people clearly stated that the interface was very easy to use and navigate through.

The Negative response from the application was the lack of features. From a test case the User wanted more description of the profession e.g. Gardner could be split into several different section e.g. Horticulture. Also, the settings were a problem as the range of use has a maximum distance range of 100km. When creating the application its use was initially for local jobs but if the application could progress further there may be service in expanding the application.

# Testing throughout the application.

When I was implementing the Location within the android application I performed a number of physical tests. I tested out the location as I was developing the application and took note of what change of location was taking place.



Jobder retrieves its location from the last location available retrieved by the Location Manager. When I got the location working I started to record results.

Here is a table of the following location’s recorded.

|  |  |
| --- | --- |
| **Expected** | **Actual** |
| 1 53.364034, -6.258972 | 53.334505, -6.258745 |
| 2 53.361023, -6.25222 | 53.36204, -6.25485 |
| 3 53.327654, -6.258972 | 53.36204, -6.25485 |
| 4 53.365664, -6.258972 | 53.36427 -6.25894 |
| 5 53.364034, -6.258972 | 53.364343, -6.258456 |

From the results it is clear to see that the location received throughout the tests recorded in Glasnevin, Co Dublin are accurate to a certain point. I gathered the Expected Latitude and Longitude values from google Maps. <https://www.google.com/maps>. Before I validated my actual results, I knew that the location was not going to be fully accurate due the very precise location values from the location provider.

In Test Cases 2 and 3 I was not receiving the same actual location. This was because the location service had not updated and was currently receiving its last known location.

This could have been due to several factors including:

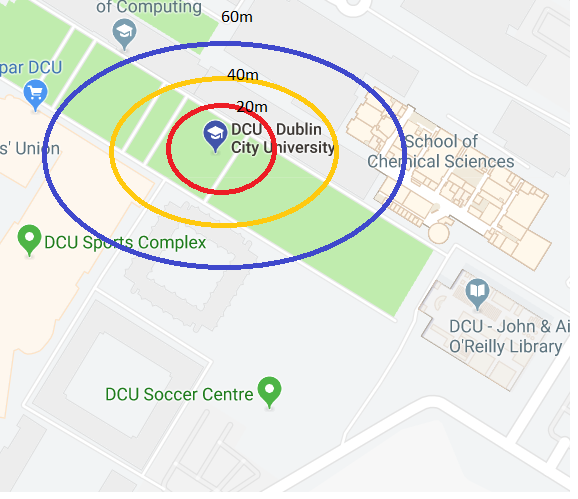
* The Location might not have been updated as it was in a similar location.
* Before I tested my third testcase I was inside a room testing the location and the GPS signal would find it difficult to penetrate through buildings.
* The location on the device may have been turned off.

The location is set to change every 10 meters in displacement for every minute. I creates a simple test in DCU that when I walked every 10 meters the location changes. In this simple experiment I used I tested from within the buildings to test the effect on the GPS signal through buildings. Here is the test results I expected.

**Here are the actual and expected results:**

|  |  |  |
| --- | --- | --- |
| **Meters** | **Latitude** | **Longitude** |
| **10** | **53.4567809** | **-6.344455** |
| **20** | **53.4567809** | **-6.344455** |
| **30** | **53.567777** | **-6236533** |
| **40** | **53.546555** | **-6.567216** |
| **50** | **53.556251** | **-6.451287** |
| **60** | **53.563211** | **-6.456743** |

**Area:**



**Analysis:**

The Results shown are changing location every 10 meters apart from the first two test cases as these were taking from inside the computing buildings, this was due to the strength of the GPS signal penetrating through the walls.

**References:**

[**https://en.wikipedia.org/wiki/JUnit**](https://en.wikipedia.org/wiki/JUnit)

[**https://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b**](https://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b)

[**https://developer.android.com/studio/test/monkey**](https://developer.android.com/studio/test/monkey)

[**https://geteasyqa.com/qa/best-test-case-templates-examples/**](https://geteasyqa.com/qa/best-test-case-templates-examples/)

[**http://softwaretestingfundamentals.com/black-box-testing/**](http://softwaretestingfundamentals.com/black-box-testing/)

[**https://en.wikipedia.org/wiki/White-box\_testing**](https://en.wikipedia.org/wiki/White-box_testing)

[**http://softwaretestingfundamentals.com/gray-box-testing/**](http://softwaretestingfundamentals.com/gray-box-testing/)