



JOBDER APPLICATION

Testing Report



Name: Adrian Rabbitt

Student Number: 14500447

Title: Jobder

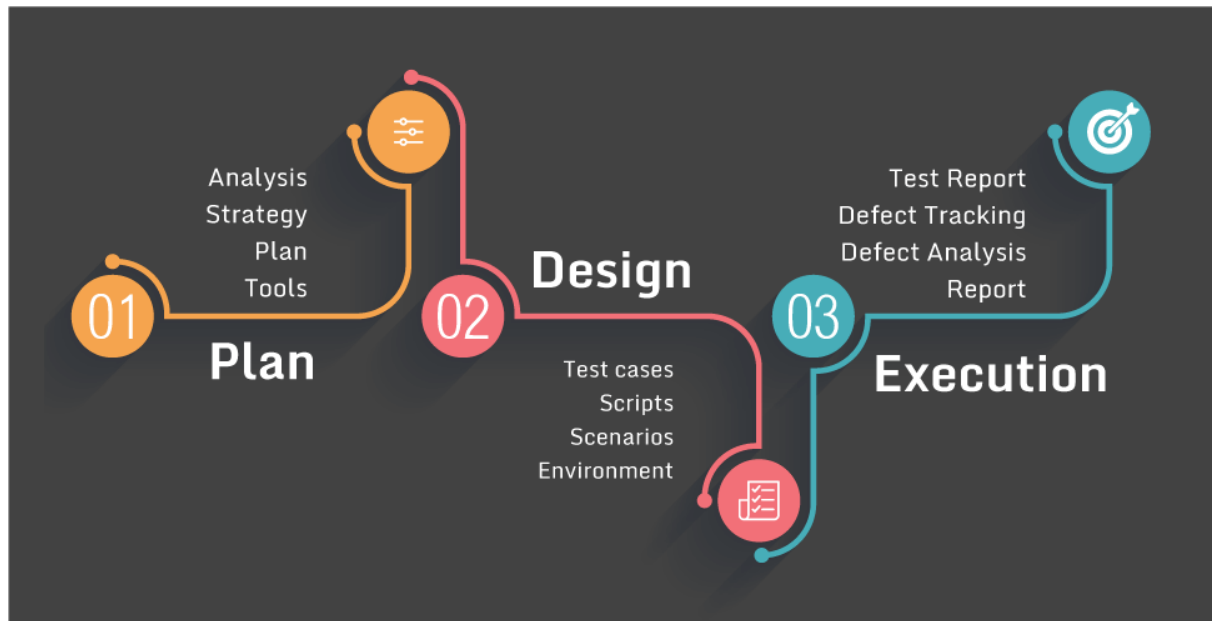
Finish Date: 20/05/2018

APRIL 22, 2018

Contents

1. Approach	2
2. Testing methods	3
2.1 The core of the testing framework is the "Test System"	3
3. Android and Android testing Frameworks.	5
3.1 Espresso	5
3.2 JUnit.	6
4. Testing of the Application.....	8
5. Functional UI Testing (Espresso).....	28
6. Performance Issues and Resolution.	49
7. Stress Testing in Android Using Monkey.....	51
8. Testing throughout the application.....	61
9. User Testing:.....	54

JOBDER APPLICATION TESTING



1. 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.

2. 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.

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

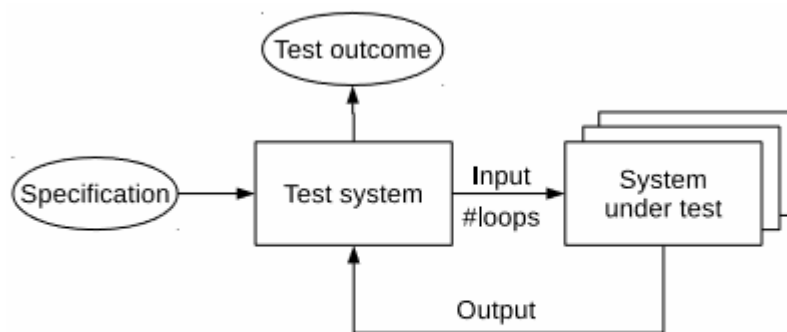


Figure 2.1: General overview of a testing process

White, grey and black box testing.

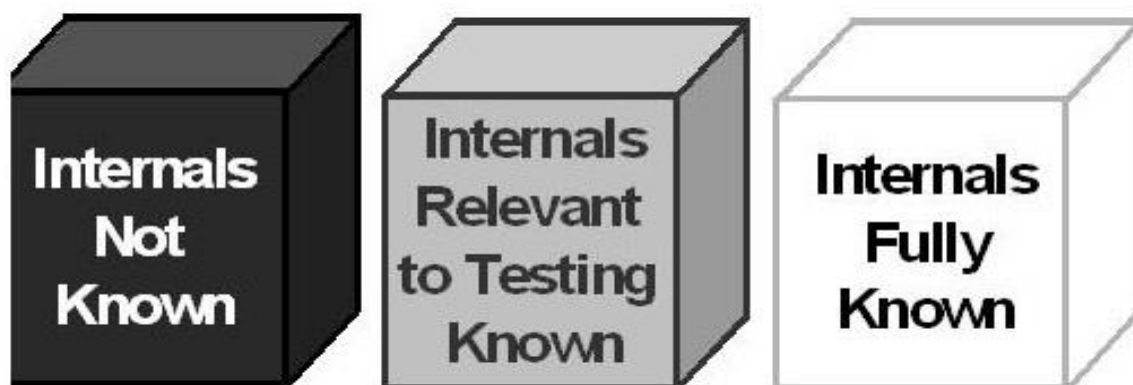
Software testing can be divided into three categories white, grey and black box testing. Black box testing is a method that mainly focuses on the functionality of the application without taking into its considerations of the applications internal functions. When implementing this test it can be applied 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.



3. 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 espresso 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.

3.1 Espresso

Espresso 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. Espresso 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 Espresso 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 Espresso 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.

3.2 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.

```
import junit.framework.*;

public class JavaTest extends TestCase {
    protected int value1, value2;

    // assigning the values
    protected void setUp(){
        value1 = 3;
        value2 = 3;
    }

    // test method to add two values
    public void testAdd(){
        double result = value1 + value2;
        assertTrue(result == 6);
    }
}
```

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.

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

//JUnit Suite Test
@RunWith(Suite.class)

@Suite.SuiteClasses({
    TestJUnit1.class ,TestJUnit2.class
})

public class JunitTestSuite {
}
```

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestJUnit1 {

    String message = "Robert";
    MessageUtil messageUtil = new MessageUtil(message);

    @Test
    public void testPrintMessage() {
        System.out.println("Inside testPrintMessage()");
        assertEquals(message, messageUtil.printMessage());
    }
}
```

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.

4. 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	Test case Description	Expected Result	Actual Result	Remarks
1				
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.

```
public void getdistance() throws Exception {  
  
    double a = 1.0;  
    double c = 1.0;  
    MapsActivity m =new MapsActivity();  
    // ;  
    assertEquals(a,m.getdistance( lat: 1.0, lng: 1.0), delta: .1);  
}
```

Visual Result



Test case Id	Test case Description	Expected Result	Actual Result	Remarks
2				
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

```

@Test
public void getFormattedDateFromTimestamp() throws Exception {

    String Expected = "Jan 18, 1970";
    long test = 1523737862;
    String Actual = ConversationAdapter.getFormattedDateFromTimestamp( timestampInMilliseconds: 1523737862);
    assertEquals(Expected, Actual);
    {
    }
}

```

Visual Result



Testing started at 01:17 ...

```

04/15 01:17:48: Launching ConversationAdapterT...
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProject
$ adb shell am force-stop com.example.adrian.firebase
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-an
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.ConversationAdapterTest
Client not ready yet..
Started running tests
Tests ran to completion.

```

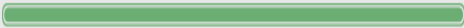
Test case Id	Test case Description	Expected Result	Actual Result	Remarks
3				
1	The value for Longitude will never be equal to null.	Longitude will be a whole number.	PASS	
2	The result will be approximately correct according to X10-3 correctness.	Correct Location for Longitude will be returned.	PASS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

JUnit.

```
public double getLongitude(){
    Junit location;
    double Expected = -6.25352; //Rounding off location as it is very difficult to
                                // get approximate value
    double Actual;

    if(location != null){
        double longitude = location.getLongitude();
    }
    Actual = getLongitude();
    assertEquals(Expected, Actual, .5);
}
```

Visual Result.



```
1 test passed - 0ms

Testing started at 01:43 ...

04/15 01:43:32: Launching getLongitude()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-de
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#getLongitude com.example.adrian.firebase.test/android.
```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
4				
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.

```

public double getLatitude(){
    Junit location;
    double Expected = 53.3729; //Rounding off location as it is very difficult to
                                // get approximate value
    double Actual;

    if(location != null){
        double longitude = location.getLatitude();
    }
    Actual = getLatitude();
    assertEquals(Expected, Actual, .5);
}

```

Visual Result.



1 test passed - 0ms

Testing started at 14:04 ...

04/15 14:04:37: Launching getLatitude()

No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-d

\$ adb shell am force-stop com.example.adrian.firebase

No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest

\$ adb shell am force-stop com.example.adrian.firebase.test

Running tests

\$ adb shell am instrument -w -x -s debug false -a class com.example.adrian.firebase.CDDMarkerTest\$TestLatitude com.example.adrian.firebase.test/android.

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
5				
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

```

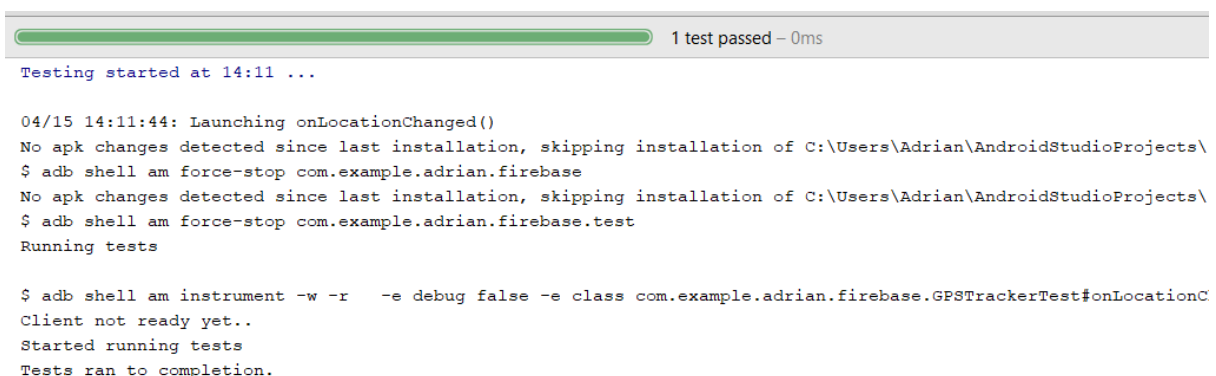
@Test
public boolean onLocationChanged() throws Exception {

    return true;

    boolean Expected = true;
    boolean Actual = true;
    assertEquals(Expected, Actual);
}

```

Visual Result



```

Testing started at 14:11 ...
04/15 14:11:44: Launching onLocationChanged()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests
$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#onLocationC
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
6				
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.

```

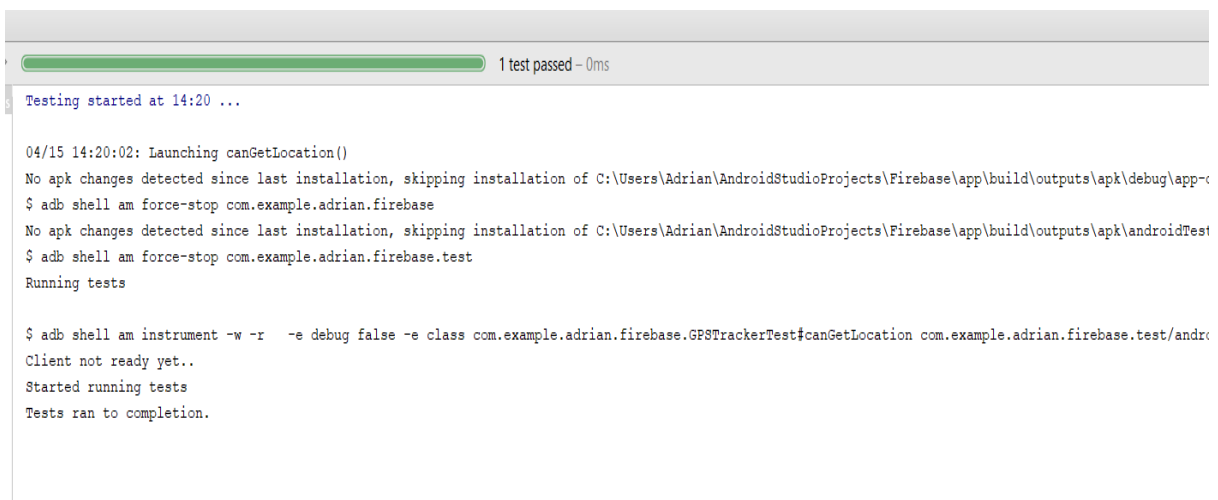
public boolean canGetLocation() throws Exception {

    if(Location != null)
    {
        return true;
    }

    boolean Expected = true;
    boolean Actual = GPSTracker.canGetLocation();
    assertEquals(Expected, Actual);
}

```

Visual Result.



The screenshot shows the Android Studio interface with the test runner at the bottom. A green progress bar at the top indicates '1 test passed - 0ms'. Below it, the text 'Testing started at 14:20 ...' is visible. The main log area shows the following output:

```

04/15 14:20:02: Launching canGetLocation()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -x -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#canGetLocation com.example.adrian.firebase.test/android
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
7				
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.

```

public void stopUsingGPS() throws Exception {

    boolean Expected = false;
    boolean Actual = GPSTracker.stopUsingGPS();

    assertEquals(Expected,Actual);

}

```

Visual Result.

The screenshot shows the Android Studio test runner interface. At the top, a green progress bar indicates that 1 test passed in 0ms. Below the progress bar, the text 'Testing started at 14:25 ...' is visible. The main area displays the test execution log, which includes the following text:

```

04/15 14:25:28: Launching stopUsingGPS()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-de
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#stopUsingGPS com.example.adrian.firebase.test/android.
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
8				
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.

```

public void onProviderEnabled() throws Exception {
    boolean Expected = false;
    boolean Actual = GPSTracker.onProviderEnabled();

    assertEquals(Expected, Actual);
}

```

Visual Result.

The screenshot shows the Android Studio test runner interface. At the top, a green progress bar indicates '1 test passed - 0ms'. Below this, the text 'Testing started at 14:51 ...' is visible. The main area displays the test execution log, which includes the following text:

```

04/15 14:51:51: Launching onProviderEnabled()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTes
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#onProviderEnabled com.example.adrian.firebase.test/a
Client not ready yet..
Started running tests
Tests ran to completion.

```


Test case Id	Test case Description	Expected Result	Actual Result	Remarks
9				
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.

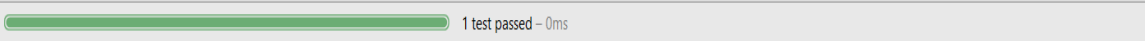
```

public void onStatusChanged() throws Exception {
    boolean Expected = false;
    boolean Actual = GPSTracker.onStatusChanged();

    assertEquals(Expected,Actual);
}

```

Visual Result.



```

04/15 14:54:09: Launching onStatusChanged()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-d
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.GPSTrackerTest#onStatusChanged com.example.adrian.firebase.test/andr
Client not ready yet..
Started running tests
Tests ran to completion.

```

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

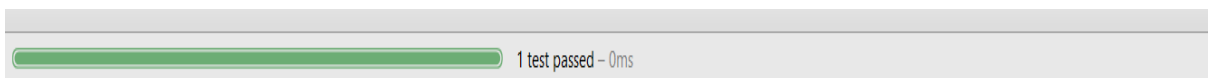
JUnit.

```
public void onMarkerClick() throws Exception {

    boolean Expected = false;
    boolean Actual = GPSTracker.onMarkerClicked();

    assertEquals(Expected,Actual);
}
```

Visual Result.



```
04/15 14:57:41: Launching onMarkerClick()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\deb
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\and:
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.MapsActivityTest#onMarkerClick com.example.adrian.firebase.t
Client not ready yet..
Started running tests
Tests ran to completion.
```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
11				
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.

```

public void onMessageReceived() throws Exception {

    boolean Expected = false;
    boolean Actual = GPSTracker.onMessageRecieved();

    assertEquals(Expected,Actual);
}

```

Visual Result.

```

04/15 15:02:51: Launching onMessageReceived()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-de
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.MyFirebaseMessagingServiceTest#onMessageReceived com.example.adrian.f
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
12				
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.

```

~ ~ ~ ~ ~
public void onBackPressed() throws Exception {


    boolean Expected = false;
    boolean Actual = GPSTracker.onBackPressed();

    assertEquals(Expected,Actual);

}

```

Visual Result.


1 test passed - 0ms

```

04/15 15:11:19: Launching onBackPressed()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debu
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\andr
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.EmployeeTest#onBackPressed com.example.adrian.firebase.test/a
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
13				
1	View all the sentence suggestion when the onGetSentenceSuggestions() function is called.	onGetSentenceSuggestions() returns true with the input as "H" with the expected output of "He,Hello,Him"	Fail	The output returned false as it did not consist of the String "He,Hello,Him"

JUnit.

```

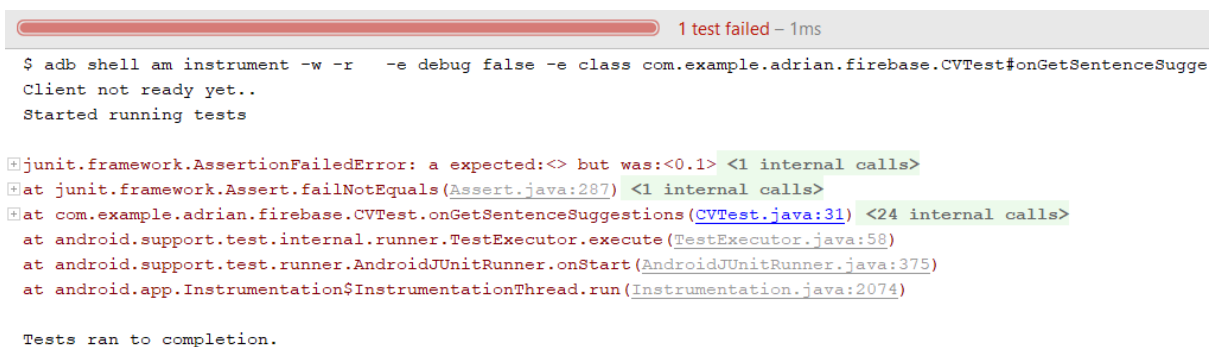
public void onGetSentenceSuggestions() throws Exception {

    //input = H
    String Expected = "He,Hello,Him";
    String Actual = GPSTracker.onGetSentenceSuggestions();

    assertEquals(Expected,Actual);
}

```

Visual Result.



```

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.CVTest#onGetSentenceSugge
Client not ready yet..
Started running tests

* junit.framework.AssertionFailedError: a expected:<> but was:<0.1> <1 internal calls>
* at junit.framework.Assert.failNotEquals(Assert.java:287) <1 internal calls>
* at com.example.adrian.firebase.CVTest.onGetSentenceSuggestions(CVTest.java:31) <24 internal calls>
  at android.support.test.internal.runner.TestExecutor.execute(TestExecutor.java:58)
  at android.support.test.runner.AndroidJUnitRunner.onStart(AndroidJUnitRunner.java:375)
  at android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:2074)

Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
14				
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.


```
public void onGetSuggestions() throws Exception {

    //input = H and array return set to 1.
    String Expected = "He";
    String Actual = GPSTracker.onGetSuggestions();

    assertEquals(Expected,Actual);

}
```

Visual Result.



```
04/15 15:25:43: Launching onGetSuggestions()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\android
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.CVTest#onGetSuggestions com.example.adrian.firebase.test/android
Client not ready yet..
Started running tests
Tests ran to completion.
```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
15				
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	

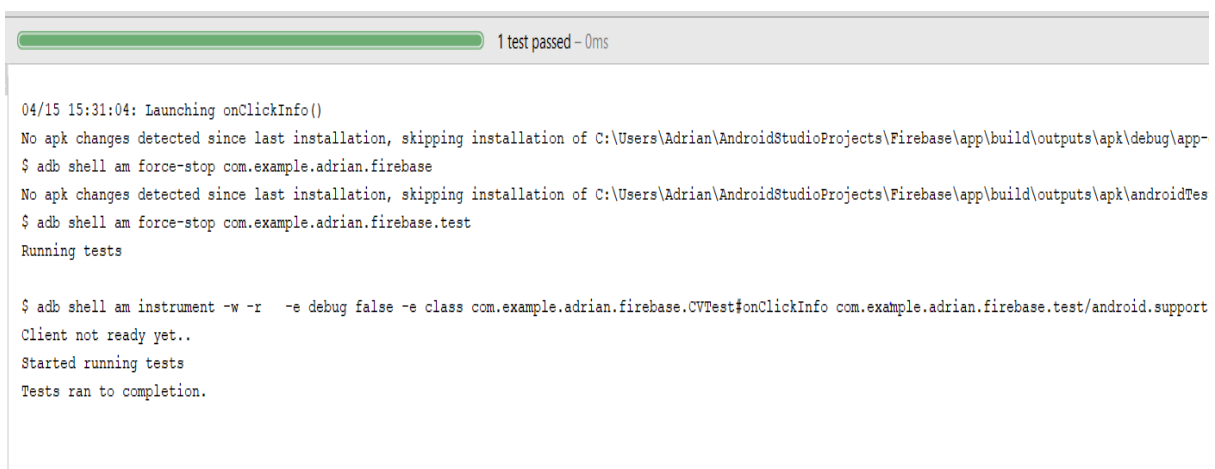
JUnit.

```
public void onClickInfo() throws Exception {

    boolean Expected = true;
    boolean Actual = GPSTracker.onClickInfo();
    assertEquals(Expected, Actual);

}
```

Visual Result.



The screenshot shows the Android Studio interface with a green progress bar at the top indicating '1 test passed - 0ms'. Below the progress bar, the test execution log is visible, showing the following output:

```
04/15 15:31:04: Launching onClickInfo()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTes
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.CVTest#onClickInfo com.example.adrian.firebase.test/android.support
Client not ready yet..
Started running tests
Tests ran to completion.
```

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

JUnit.

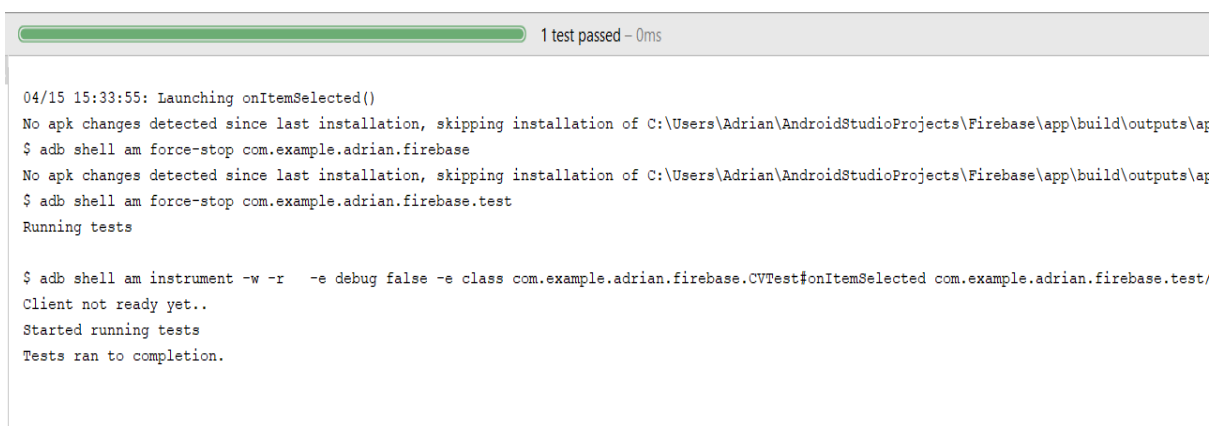
```

public void onItemSelected() throws Exception {

    boolean Expected = true;
    boolean Actual = GPSTracker.onClickListener();
    assertEquals(Expected, Actual);
}

```

Visual Result.



The screenshot shows the Android Studio test runner interface. At the top, a green progress bar indicates that 1 test passed in 0ms. Below the progress bar, the test execution log is visible, showing the following output:

```

04/15 15:33:55: Launching onItemSelected()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.CVTest#onItemSelected com.example.adrian.firebase.test/
Client not ready yet..
Started running tests
Tests ran to completion.

```


Test case Id	Test case Description	Expected Result	Actual Result	Remarks
17				
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.

```

public void onSignUp() throws Exception {

    String Email = "adrian@gmail.com";
    String Password = "12345678";
    boolean Expected = true;
    boolean Actual = GPSTracker.onSignUp();
    assertEquals(Expected, Actual, " ");
}

```

Visual Result.



```

$ adb shell am force-stop com.example.adrian.firebase
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data/local/tmp/c
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.LoginEmployeeTest#onSignUp com.example.adrian.firebase.t
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
18				
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.

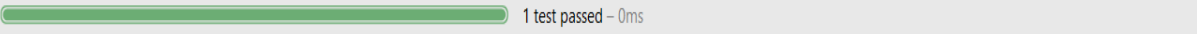
```

public void onLogin() throws Exception {

    String Email = "adrian@gmail.com";
    String Password = "12345678";
    boolean Expected = true;
    boolean Actual = GPSTracker.onLogin(Email,Password);
    assertEquals(Expected, Actual, " ");
}

```

Visual Result.



```

$ adb shell am force-stop com.example.adrian.firebase
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data/local/tmp/c
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.LoginEmployeeTest#onSignUp com.example.adrian.firebase.t
Client not ready yet..
Started running tests
Tests ran to completion.

```

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

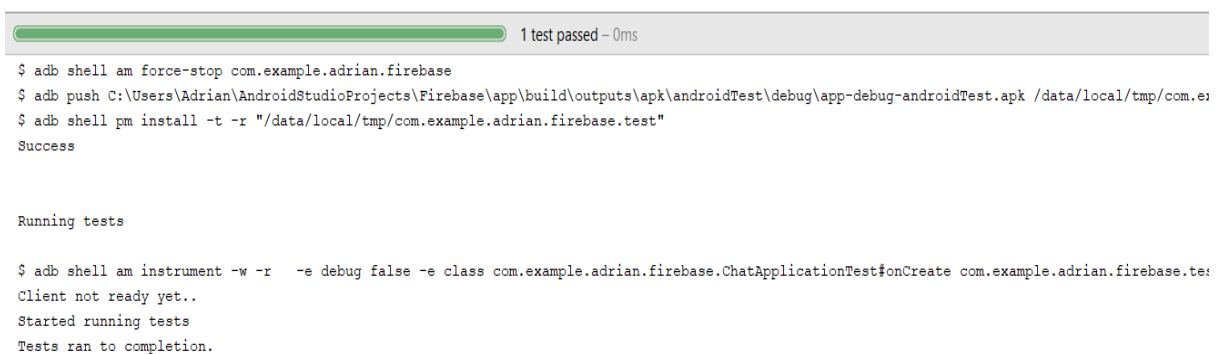
JUnit.

```

public void onCreate() throws Exception {
    boolean Expected = true;
    boolean Actual = GPSTracker.onCreate();
    assertEquals(Expected, Actual);
}

```

Visual Result.



```

$ adb shell am force-stop com.example.adrian.firebase
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data/local/tmp/com.e
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.ChatApplicationTest#onCreate com.example.adrian.firebase.te
Client not ready yet..
Started running tests
Tests ran to completion.

```

Test case Id	Test case Description	Expected Result	Actual Result	Remarks
21				
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.

```

public void getFilter() throws Exception {

    boolean Expected = true;
    boolean Actual = GPSTracker.getFilter();
    assertEquals(Expected, Actual);

}

```

Visual Result.



```

04/15 15:51:47: Launching getFilter()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.ContactsAdapterTest#getFilter com.exan
Client not ready yet..
Started running tests
Tests ran to completion.

```

5. Functional UI Testing (Espresso).

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

1) Testing The execution of splash screen.

```
import android.support.test.filters.LargeTest;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;

import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

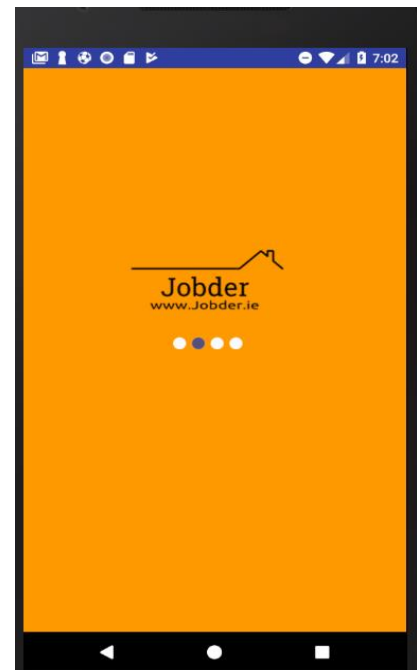
@RunWith(AndroidJUnit4.class)
@LargeTest
public class MainActivityTest {

    private String mStringToBetyped;

    @Rule
    public ActivityTestRule<MainActivity> mActivityRule = new ActivityTestRule<>() {
        MainActivity.class;
    };

    @Before
    public void initValidString() {
        // Specify a valid string.
        mStringToBetyped = "The Thread finished excuting on Splash Screen";
    }

    @Test
    public void changeText_sameActivity() {
        // Type text and then press the button.
    }
}
```



Test Result:

```
1 test passed - 4s 120ms

Testing started at 19:58 ...

04/19 19:58:04: Launching MainActivityTest
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-debug.apk /data/local/tmp/com.example.adri
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase"
Success

$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data/lo
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success
```

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.

2) Options Screen.

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class OptionsTesting {

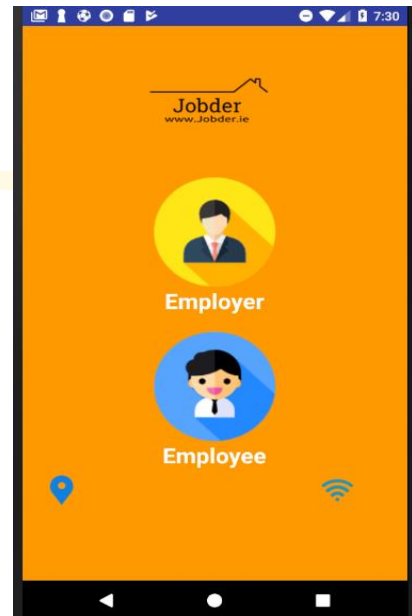
    private String mStringToBetyped;

    @Rule
    public ActivityTestRule<Options> mActivityRule = new ActivityTestRule<>(Options.class);

    @Before
    public void initValidString() {
        // Specify a valid string.
        mStringToBetyped = "Testing the Employer and Employee buttons.";
    }

    @Test
    public void ChooseEmployee() {
        onView(withId(R.id.Employee)).perform(click());
        onView(withId(R.id.Employee)).check(matches(not(isEnabled())));
    }

    @Test
    public void ChooseEmployer() {
        onView(withId(R.id.Employer)).perform(click());
        onView(withId(R.id.Employer)).check(matches(not(isEnabled())));
    }
}
```



Test Result

```

All 2 tests passed - 1s 458ms

Testing started at 20:35 ...

04/19 20:35:59: Launching OptionsTesting
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app
$ adb shell am force-stop com.example.adrian.firebase
$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.OptionsTesting com.example.adrian.fire
Client not ready yet..
Started running tests
```

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.

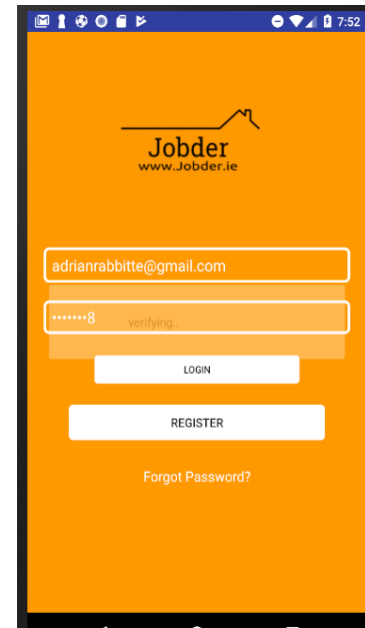
```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class LoginTest {

    @Rule
    public ActivityTestRule<LoginEmployee> mActivityRule = new ActivityTestRule<>(LoginEmployee.class);

    @Test
    public void LoginEmployee() {
        try {
            // Type email and password
            Espresso.onView(ViewMatchers.withId(R.id.email))
                .perform(ViewActions.typeText(stringOfBelongsTo: "adrianrabbite@gmail.com"), ViewActions.closeSoftKeyboard());
            Espresso.onView(ViewMatchers.withId(R.id.password))
                .perform(ViewActions.typeText(stringOfBelongsTo: "12345678"), ViewActions.closeSoftKeyboard());

            // Click login button
            Espresso.onView(ViewMatchers.withId(R.id.appCompatButtonLogin)).perform(ViewActions.click());
        } catch (Exception e) {
            //view not displayed logio
        }
    }
}
```

3)Login Employee Page.



Test Result

```
1 test passed - 12s 877ms

Testing started at 20:52 ...

04/19 20:52:07: Launching LoginEmployee()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\c
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\c
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.LoginTest#LoginEmployee com.example.adrian.fire
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion.

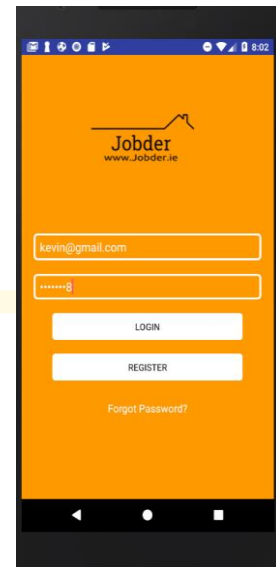
When the user logs in with the correct credentials the user is logged in successfully. This function is returned from the Firebase AUTH database.

4)Login Employer Screen

```
@Rule
public ActivityTestRule<LoginEmployer> mActivityRule = new ActivityTestRule<>(LoginEmployer.class);

@Test
public void LoginEmployee() {
    try {
        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.email))
            .perform(ViewActions.typeText(stringToBeTyped: "kevin@gmail.com"), ViewActions.closeSoftKeyboard());
        Espresso.onView(ViewMatchers.withId(R.id.password))
            .perform(ViewActions.typeText(stringToBeTyped: "12345678"), ViewActions.closeSoftKeyboard());

        // Click login button
        Espresso.onView(ViewMatchers.withId(R.id.appCompatButtonLogin)).perform(ViewActions.click());
    } catch (Exception e) {
        //view not displayed logic
    }
}
```



Test Result

```
1 test passed - 8s 40ms

Testing started at 21:02 ...

04/19 21:02:30: Launching LoginEmployer()
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-debug.apk
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\app-androidTest.apk
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -x -e debug false -e class com.example.adrian.firebase.LoginEmployerTest#LoginEmployer com.example.adrian.firebase.test/androidx.test.runner.AndroidJUnit4
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion

When the user logs in 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 it failed as adrianrabbitte@gmail.com was not a registered Employer.

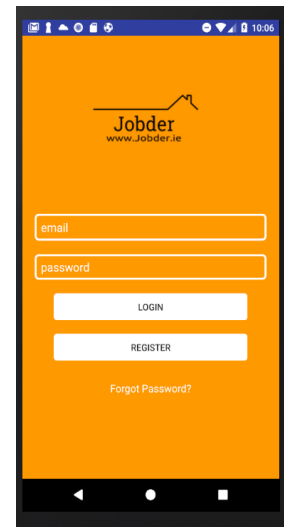
5) Signup Button.

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SignupTest {

    @Rule
    public ActivityTestRule<LoginEmployer> mActivityRule = new ActivityTestRule<>(LoginEmployer.class);

    @Test
    public void LoginEmployer() {
        try {
            // Type email and password

            // Click login button
            Espresso.onView(ViewMatchers.withId(R.id.ButtonRegister)).perform(ViewActions.click());
        } catch (Exception e) {
            //view not displayed logic
        }
    }
}
```



Test Result.

```
>> 1 test passed - 3s 722ms

Testing started at 23:06 ...

04/19 23:06:17: Launching SignupTest
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\app-debug.apk
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\app-androidTest.apk
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -x -e debug false -e class com.example.adrian.firebase.SignupTest com.example.adrian.firebase.test/android.support.test.runner.AndroidJUnit4
Client not ready yet..
Started running tests
Tests ran to completion.
```

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

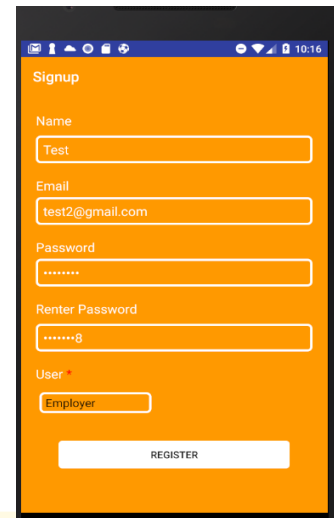
```
public ActivityTestRule<Signup> mActivityRule = new ActivityTestRule<>(Signup.class);

@Test
public void Signup() {
    try {
        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.Name))
            .perform(ViewActions.typeText( stringToBeTyped: "Test"), ViewActions.closeSoftKeyboard());
        Espresso.onView(ViewMatchers.withId(R.id.Email))
            .perform(ViewActions.typeText( stringToBeTyped: "test2@gmail.com"), ViewActions.closeSoftKeyboard());

        Espresso.onView(ViewMatchers.withId(R.id.Password))
            .perform(ViewActions.typeText( stringToBeTyped: "12345678"), ViewActions.closeSoftKeyboard());

        Espresso.onView(ViewMatchers.withId(R.id.re))
            .perform(ViewActions.typeText( stringToBeTyped: "12345678"), ViewActions.closeSoftKeyboard());

        // Click login button
        Espresso.onView(ViewMatchers.withId(R.id.signup)).perform(ViewActions.click());
    } catch ( Exception e ) {
        //view not displayed logic
    }
}
```



Signup

Name
Test

Email
test2@gmail.com

Password
12345678

Reenter Password
12345678

User
Employer

REGISTER

Test Result

```
1 test passed - 11s 980ms

Testing started at 23:16 ...

04/19 23:16:25: Launching RegesterTest
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\output
$ adb shell am force-stop com.example.adrian.firebase
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.RegesterTest com.example.adrian.firebase.test/andr
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion.

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

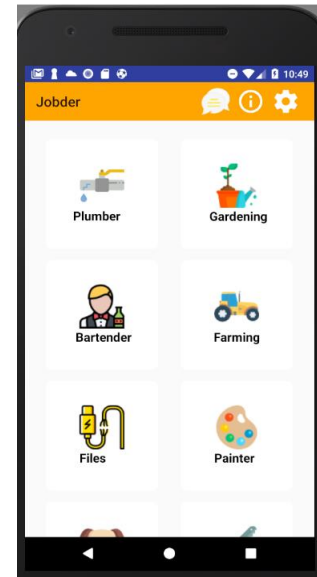
7) Employer Home Page.

```
public class EmployerHomePage {

    @Rule
    public ActivityTestRule<Employer> mActivityRule = new ActivityTestRule<>(Employer.class);

    @Test
    public void Employer() {
        try {
            // Type email and password

            Espresso.onView(ViewMatchers.withId(R.id.settings)).perform(ViewActions.click());
            Espresso.onView(ViewMatchers.withId(R.id.info)).perform(ViewActions.click());
            Espresso.onView(ViewMatchers.withId(R.id.messages)).perform(ViewActions.click());
            // Click login button
            // Espresso.onView(ViewMatchers.withId(R.id.signup)).perform(ViewActions.click());
        } catch (Exception e) {
            //view not displayed logic
        }
    }
}
```



Test Result:

```
> 1 test passed - 2s 662ms

$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data
$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.EmployerHomePage#Employer com.example.adrian.firebase.EmployerHomePage
Client not ready yet..
Started running tests
Tests ran to completion.
```

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.

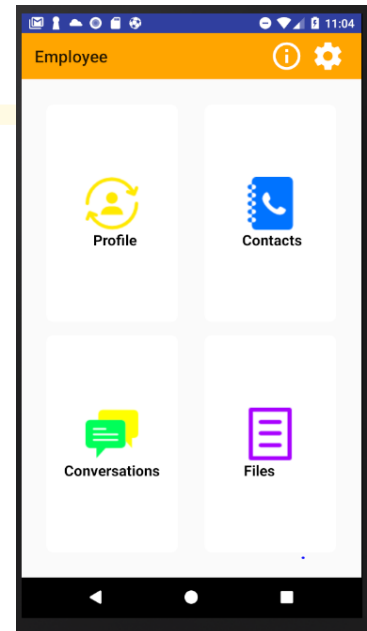
```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class EmployeeProfileTest {

    @Rule
    public ActivityTestRule<Employee> mActivityTestRule = new ActivityTestRule<>(Employee.class);

    @Test
    public void Employer() {
        try {
            // Type email and password

            Espresso.onView(ViewMatchers.withId(R.id.settings)).perform(ViewActions.click());
            Espresso.onView(ViewMatchers.withId(R.id.info)).perform(ViewActions.click());

            // Click login button
            // Espresso.onView(ViewMatchers.withId(R.id.signup)).perform(ViewActions.click());
        } catch (Exception e) {
            //view not displayed logic
        }
    }
}
```



Test Result.

```
1 test passed - 1s 926ms

Testing started at 00:02 ...

04/20 00:02:22: Launching EmployeeProfileTest
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\debug\ap
$ adb shell am force-stop com.example.adrian.firebase
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidT
$ adb shell am force-stop com.example.adrian.firebase.test
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.EmployeeProfileTest com.example.adrian.firebase.test/android.supp
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion.

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

9) CV Application Test.

```
@Test
public void Register() {
    try {
        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.firstName))
            .perform(ViewActions.typeText( stringOfTypeset "Test"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.secondName))
            .perform(ViewActions.typeText( stringOfTypeset "TestSecondName"), ViewActions.closeSoftKeyboard());

        Espresso.onView(ViewMatchers.withId(R.id.email))
            .perform(ViewActions.typeText( stringOfTypeset "test2@gmail.com"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.mobile))
            .perform(ViewActions.typeText( stringOfTypeset "0877777777"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.phone))
            .perform(ViewActions.typeText( stringOfTypeset "0877777777"), ViewActions.closeSoftKeyboard());

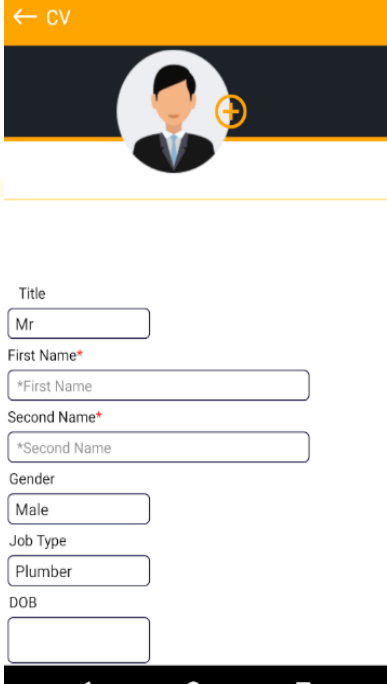
        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.Address1))
            .perform(ViewActions.typeText( stringOfTypeset "Kilsarn Loughduff"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.Address2))
            .perform(ViewActions.typeText( stringOfTypeset "Co Cavan"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.Info))
            .perform(ViewActions.typeText( stringOfTypeset "This is for specific Information"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.pTitle))
            .perform(ViewActions.typeText( stringOfTypeset "JobTest"), ViewActions.closeSoftKeyboard());

        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.skills))
            .perform(ViewActions.typeText( stringOfTypeset "creative"), ViewActions.closeSoftKeyboard());
    }
}
```



Test Result.

1 test passed – 6s 811ms

Testing started at 00:18 ...

04/20 00:18:42: Launching CVApplicationTest
No apk changes detected since last installation, skipping installation of C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\o
\$ adb shell am force-stop com.example.adrian.firebase
\$ adb push C:\Users\Adrian\AndroidStudioProjects\Firebase\app\build\outputs\apk\androidTest\debug\app-debug-androidTest.apk /data/lo
\$ adb shell pm install -t -r "/data/local/tmp/com.example.adrian.firebase.test"
Success

Running tests

\$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.CVApplicationTest com.example.adrian.firebase.
Client not ready yet..
Started running tests
Tests ran to completion.

Conclusion.

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

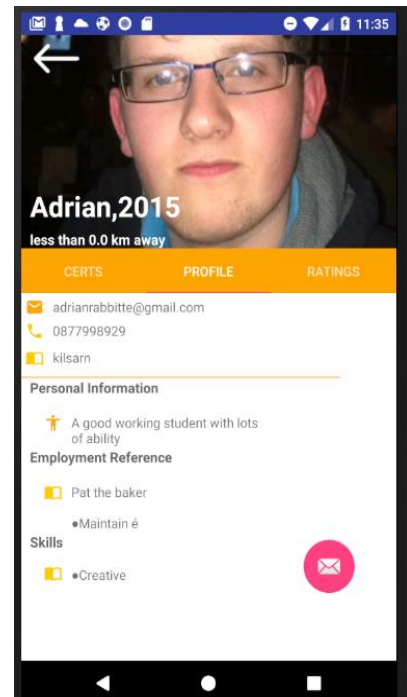
10)Profile

```
RunWith(AndroidJUnit4.class)
@LargeTest
public class ProfileTest {

    @Rule
    public ActivityTestRule<Main2Activity> mActivityRule = new ActivityTestRule<>()

    @Test
    public void SwipeTabsRight() {
        try {
            // Type email and password
            onView(withId(R.id.viewpager)).perform(swipeLeft());
            onView(withId(R.id.viewpager)).perform(swipeRight());

            // Click login button
            onView(withId(R.id.signup)).perform(ViewActions.click());
        } catch (Exception e) {
            //view not displayed logic
        }
    }
}
```



Test Result.

```
>> All 2 tests failed - 382ms

15 Connected to process 11092 on device Nexus_5X_API_26 [emulator-5554]
15 Capturing and displaying logcat messages from application. This behavior can be disabled in the "Logcat output" section of the "Deb
15 W/zygote: Skipping duplicate class check due to unrecognized classloader
15 W/DynamiteModule: Local module descriptor class for com.google.firebase.auth not found.
15 W/DynamiteModule: Local module descriptor class for com.google.firebase.auth not found.
I/BiChannelGoogleApi: [FirebaseAuth: ] No Fallback module; NOT setting up for lazy initialization
W/DynamiteModule: Local module descriptor class for com.google.firebase.auth not found.
I/FirebaseAuth: [FirebaseAuth:] Loading module via FirebaseOptions.
I/FirebaseAuth: [FirebaseAuth:] Preparing to create service connection to gms implementation
D/FirebaseApp: com.google.firebase.crash.FirebaseCrash is not linked. Skipping initialization.
V/FA: Cancelling job. JobID: 1063484833
V/FA: Registered activity lifecycle callback
I/FirebaseInitProvider: FirebaseApp initialization successful
V/FA: Collection enabled
V/FA: App package, google app id: com.example.adrian.firebase, 1:395552461617:android:a9758754e1181997
I/FA: App measurement is starting up, version: 11910
I/FA: To enable debug logging run: adb shell setprop log.tag.FA VERBOSE
```

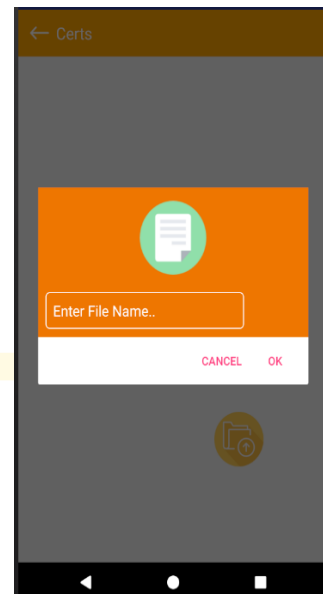
Conclusion.

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.

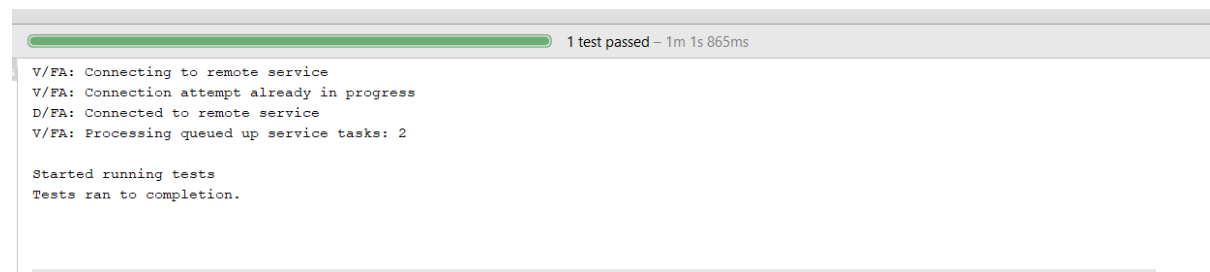
11) Uploading Images from Camera Roll.

```
public ActivityTestRule<Certificates> mActivityRule = new ActivityTestRule<>(Certificates.class);

@Test
public void Upload() {
    try {
        // Click login button
        Espresso.onView(ViewMatchers.withId(R.id.upload)).perform(ViewActions.click());
        Espresso.onView(ViewMatchers.withId(R.id.dialog))
            .perform(ViewActions.typeText(stringToBeTyped: "SafePass"), ViewActions.closeSoftKeyboard());
        Espresso.onView(ViewMatchers.withId(R.id.book_now)).perform(ViewActions.click());
    } catch (Exception e) {
        //view not displayed logic
    }
}
```



Test Result:



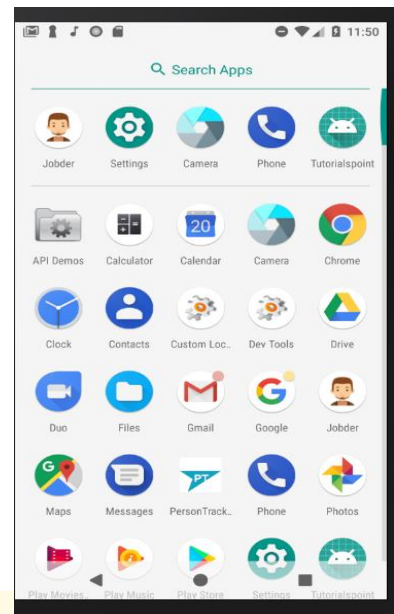
Conclusion.

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.

```
@Rule
public ActivityTestRule<Certificates> mActivityRule = new ActivityTestRule<>(Certificates.class);

@Test
public void back() {
    try {
        Espresso.onView(ViewMatchers.withId(R.id.back)).perform(ViewActions.click());
    } catch (Exception e) {
        //view not displayed logic
    }
}
```



Test Result.

1 test passed - 1s 270ms

Running tests

```
$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.UploadTest#back com.example.adrian.firebase.test/android.support.test
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion.

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

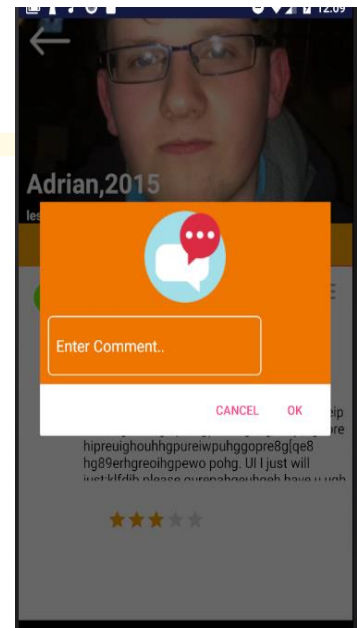
13) Upload Rating for Employee.

```
@Rule
public ActivityTestRule<Rating> mActivityRule = new ActivityTestRule<>(Rating.class);

@Test
public void Rating() {
    try {

        onView(allOf(withId(R.id.rate), isDescendantOfA(firstChildOf(withId(R.id.viewpager)))))
            .perform(click());

        // Click login button
        onView(withId(R.id.signup)).perform(click());
    } catch (Exception e) {
        //view not displayed logic
    }
}
```



Test Result:

```
Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.UploadTest$back com.example.adrian.firebase.test/android.support.test
Client not ready yet..
Started running tests
Tests ran to completion.
```

Conclusion:

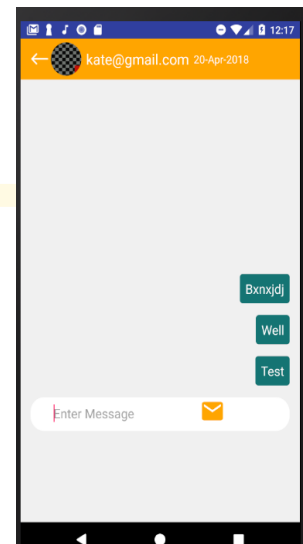
When the Employer creates a review a rating pop up dialog appears where the user can enter their details.

14) Users Sending Messages.

```
@Rule
public ActivityTestRule<ChatApplication> mActivityRule = new ActivityTestRule<>(ChatApplication.class);

@Test
public void MessageUser() {
    try {
        // Type email and password
        Espresso.onView(ViewMatchers.withId(R.id.edittext))
            .perform(ViewActions.typeText(stringToBeTyped: "Test"), ViewActions.closeSoftKeyboard());

        Espresso.onView(ViewMatchers.withId(R.id.img)).perform(ViewActions.click());
    } catch (Exception e) {
    }
}
```



Test Result

```
1 test passed - 1s 270ms

Running tests

$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.UploadTest#back com.example.adrian.firebase.test/android.support.test
Client not ready yet..
Started running tests
Tests ran to completion.
```

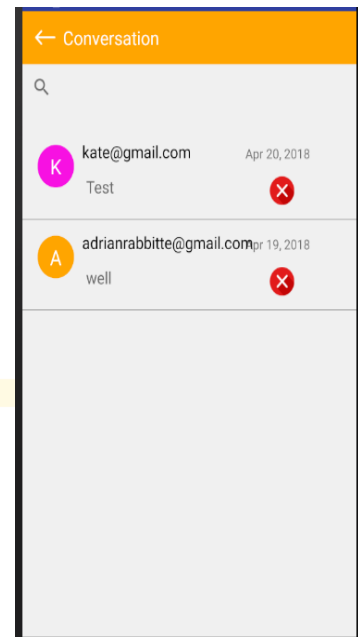
Conclusion:

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
public void Conversations() {
    try {
        // Type email and password

        // Click login button
        Espresso.onView(ViewMatchers.withId(R.id.messages)).perform(ViewActions.click());
    } catch (Exception e) {
        //view not displayed logic
    }
}
```



Test Result

```
Running tests
$ adb shell am instrument -w -r -e debug false -e class com.example.adrian.firebase.UploadTest#back com.example.adrian.firebase.test/android.support.test
Client not ready yet..
Started running tests
Tests ran to completion.
```

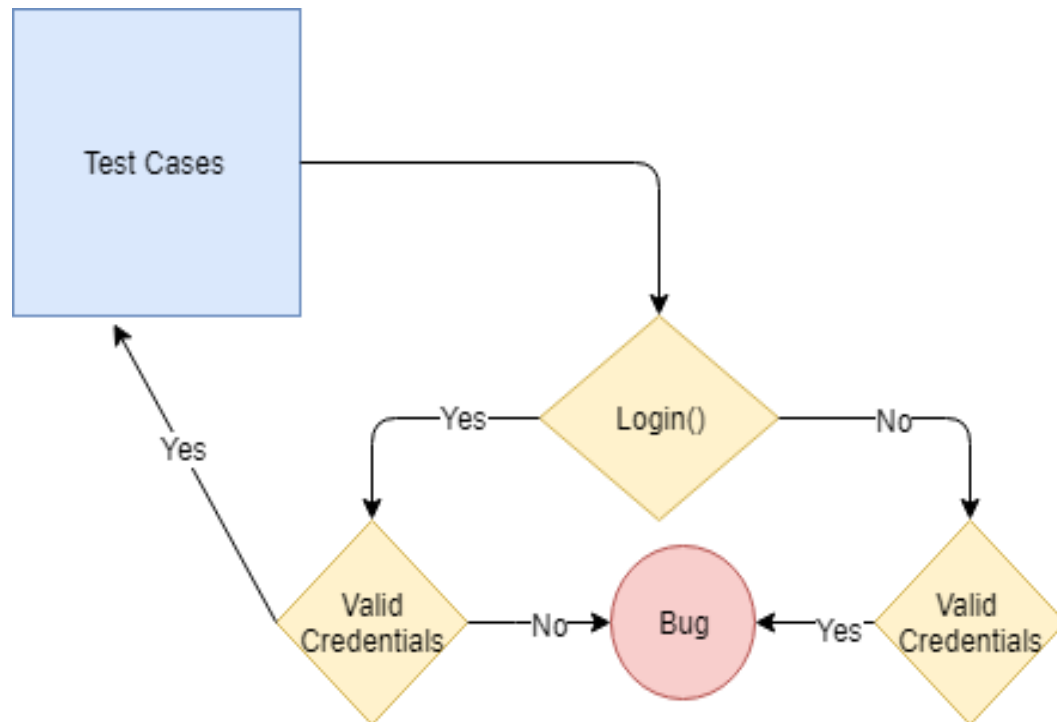
Conclusion

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

Different Test Cases Applied with Espresso.

Login

When testing my login in the android application using Espresso 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:

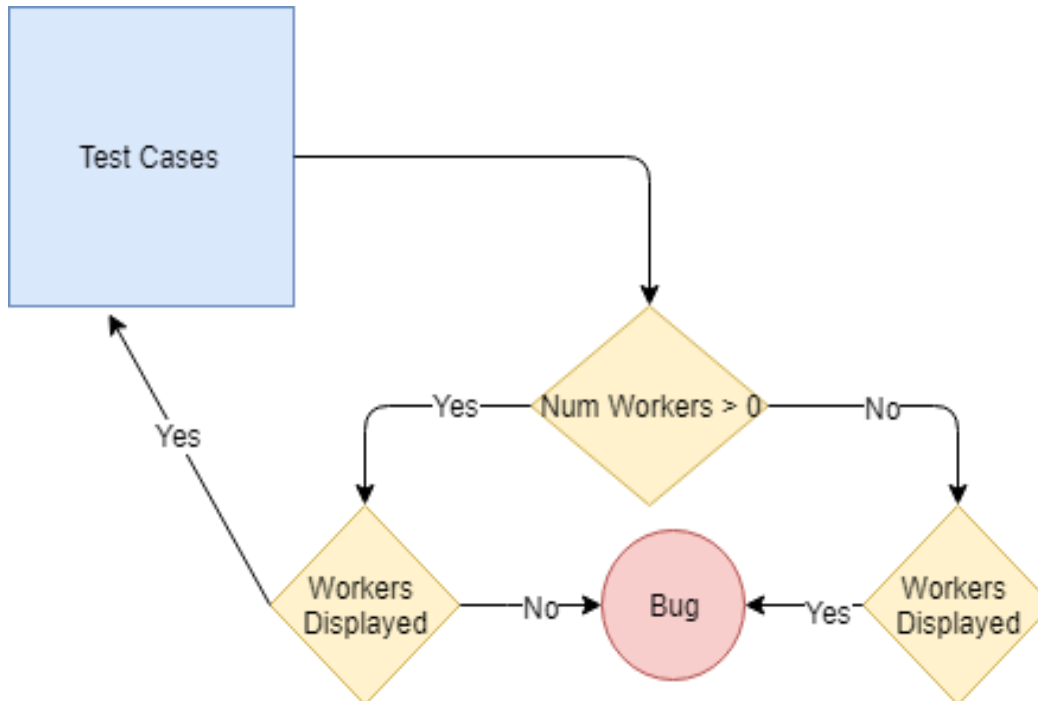
1. Valid Email and Valid Password
2. Invalid Name and Valid Password
3. Valid Name and Invalid Password
4. Invalid Name and Invalid Password

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:

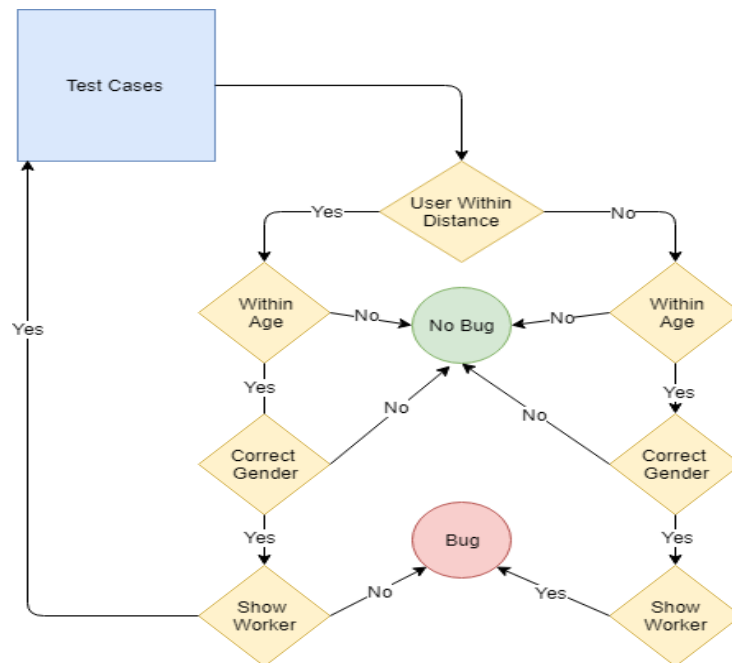
- 1) Plumbers = 2
- 2) Gardening = 3
- 3) Farming = 1
- 4) Painter = 0

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:

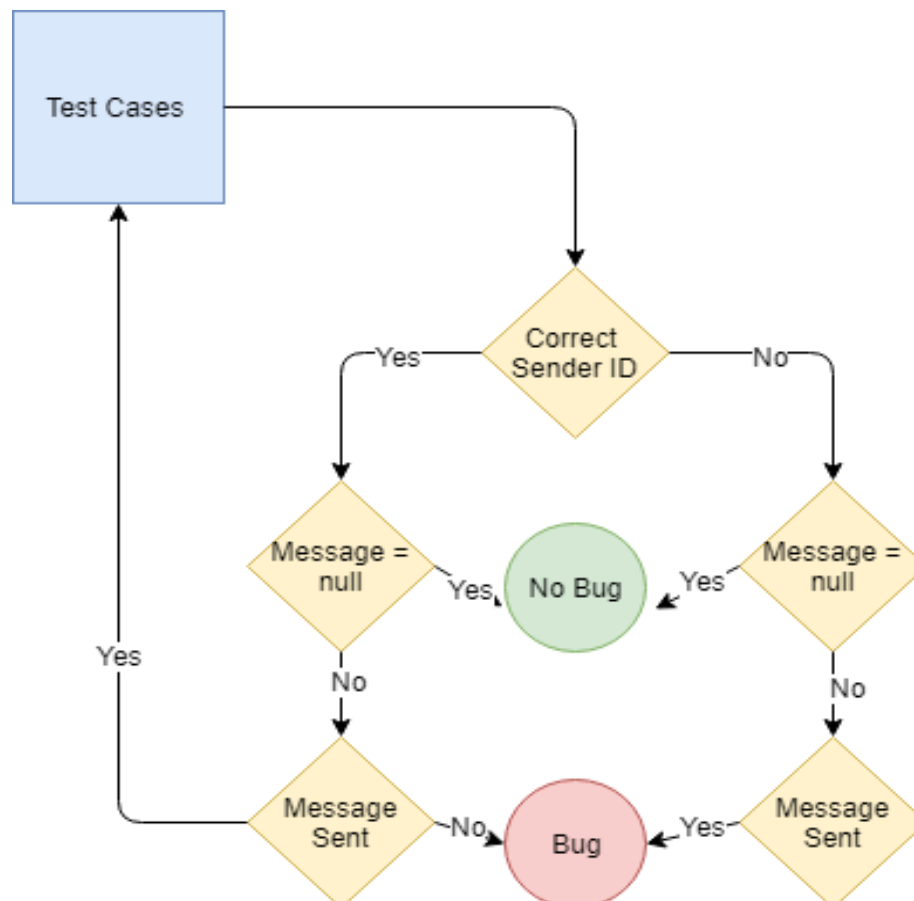
1. Male,30,20km
2. Female,20,15km
3. Male,50,10km
4. Female, 30,100km

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:

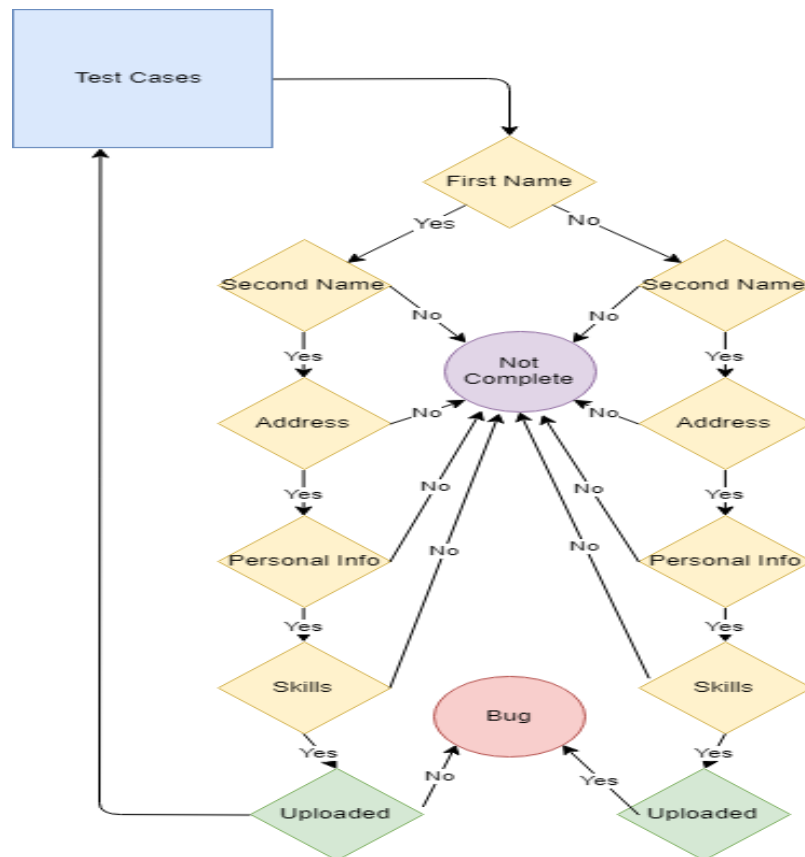
1. Valid Sender ID, Message
2. Valid Sender ID, Null
3. Invalid Sender ID, Null
4. Invalid Sender ID, Messag

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:

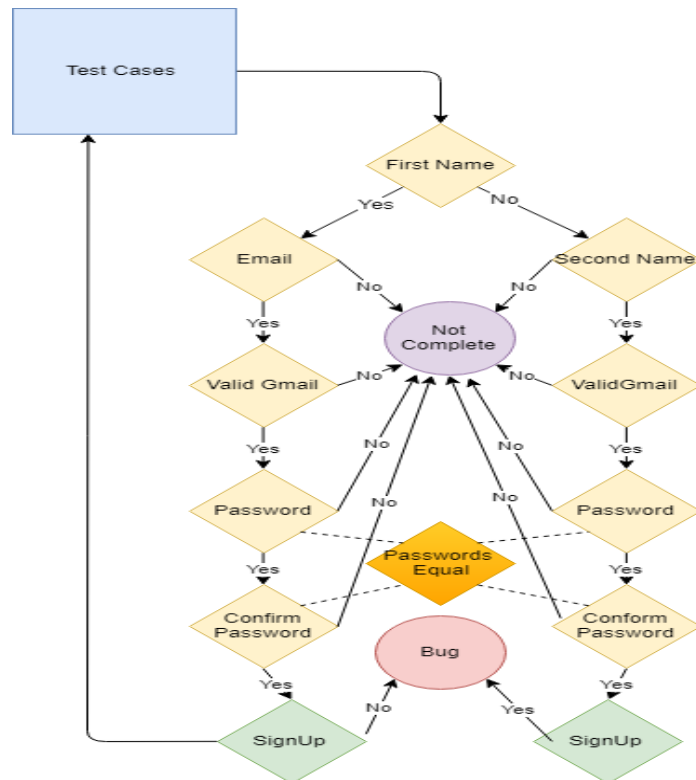
1. First Name, Second Name, Address, Personal Info, Skills
2. Null, Second Name, Address, Personal Info, Skills
3. First Name, Null, Address, Personal Info, Skills
4. First Name, Second Name, Null, Personal Info, Skills
5. First Name, Second Name, Address, Null, Skills
6. First Name, Second Name, Address, 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:

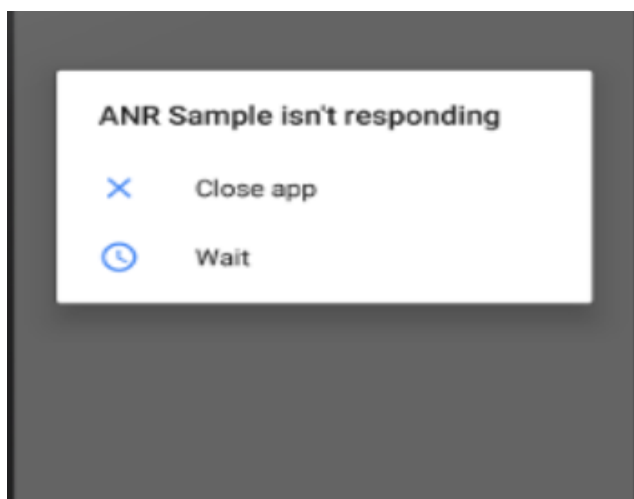
All the test cases were passed as expected as I had already performed user testing during the development process.

6. 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.



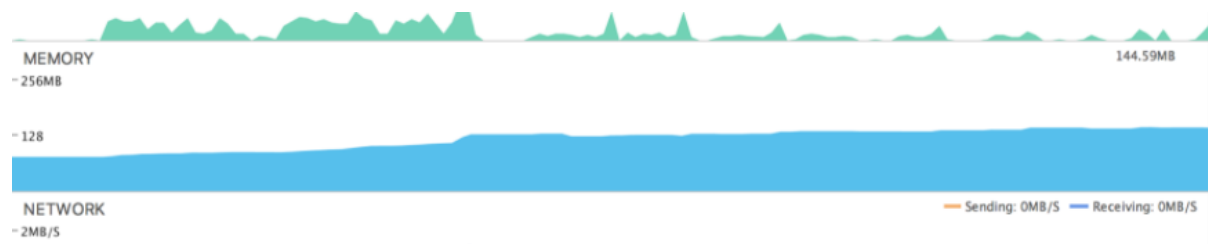
Resolution.

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 `getApplicationContext()`.

e.g.

```
Glide.with( activity: Main2Activity.this).load(localFile).asBitmap().centerCrop().into(new BitmapImage  
    @Override  
    protected void setResource(Bitmap resource) {  
  
        image.setImageBitmap(resource);  
    }  
}
```



```
Glide.with(getApplicationContext()).load(localFile).asBitmap().centerCrop().into(new Bitmap  
    @Override  
    protected void setResource(Bitmap resource) {
```

7. 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. Monkey's 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
```

(summarised)

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

```
v 500
bash arg: -p
bash arg: com.example.adrian.firebase
bash arg: -v
bash arg: 500
args: [-p, com.example.adrian.firebase, -v, 500]
arg: "-p"
arg: "com.example.adrian.firebase"
arg: "-v"
arg: "500"
data="com.example.adrian.firebase"
Monkey: seed=1524483457022 count=500
AllowPackage: com.example.adrian.firebase
IncludeCategory: android.intent.category.LAUNCHER
IncludeCategory: android.intent.category.MONKEY
/ Event percentages:
/ 0: 15.0%
/ 1: 10.0%
/ 2: 2.0%
/ 3: 15.0%
/ 4: -0.0%
/ 5: -0.0%
/ 6: 25.0%
/ 7: 15.0%
/ 8: 2.0%
/ 9: 2.0%
/ 10: 1.0%
/ 11: 13.0%
```

2) Results

```
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LA
=0x10200000;component=com.example.adrian.firebase/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.c
cmp=com.example.adrian.firebase/.MainActivity } in package com.example.adrian.firebase
:Sending Touch (ACTION_DOWN): 0:(441.0,910.0)
:Sending Touch (ACTION_UP): 0:(441.58774,916.52997)
// Rejecting start of Intent { act=android.intent.action.MAIN cat=[android.intent.
=com.google.android.apps.nexuslauncher/.NexusLauncherActivity } in package com.google.
launcher
:Sending Touch (ACTION_DOWN): 0:(109.0,547.0)
:Sending Touch (ACTION_UP): 0:(93.475624,530.9959)
:Sending Trackball (ACTION_MOVE): 0:(1.0,0.0)
// Injection Failed
:Sending Touch (ACTION_DOWN): 0:(682.0,643.0)
:Sending Touch (ACTION_UP): 0:(682.4294,662.56055)
:Sending Touch (ACTION_DOWN): 0:(1019.0,950.0)
:Sending Touch (ACTION_UP): 0:(1015.32153,899.1159)
:Sending Touch (ACTION_DOWN): 0:(850.0,1748.0)
:Sending Touch (ACTION_UP): 0:(864.2556,1756.0968)
:Sending Touch (ACTION_DOWN): 0:(17.0,500.0)
:Sending Touch (ACTION_UP): 0:(23.292957,501.0399)
:Sending Trackball (ACTION_MOVE): 0:(3.0,3.0)
:Sending Trackball (ACTION_MOVE): 0:(2.0,0.0)
// Rejecting start of Intent { act=android.intent.action.MAIN cat=[android.intent.
cmp=com.google.android.calendar/com.android.calendar.AllInOneActivity } in package com
calendar
:Sending Trackball (ACTION_MOVE): 0:(4.0,-3.0)
//[calendar_time:2018-04-20 15:57:54.542 system_uptime:13505336]
// Sending event #100
:Sending Touch (ACTION_DOWN): 0:(82.0,1670.0)
:Sending Touch (ACTION_UP): 0:(44.97901,1631.5028)
:Sending Touch (ACTION_DOWN): 0:(8.0,1349.0)
:Sending Touch (ACTION_UP): 0:(3.0620532,1347.6108)
:Sending Touch (ACTION_DOWN): 0:(297.0,518.0)
:Sending Touch (ACTION_UP): 0:(296.9883,514.41895)
:Sending Touch (ACTION_DOWN): 0:(572.0,65.0)
:Sending Touch (ACTION_UP): 0:(593.0461,134.4244)
:Sending Touch (ACTION_DOWN): 0:(488.0,1161.0)
:Sending Touch (ACTION_UP): 0:(482.24704,1159.2058)
:Sending Trackball (ACTION_MOVE): 0:(-4.0,0.0)
:Sending Trackball (ACTION_MOVE): 0:(3.0,0.0)
:Sending Trackball (ACTION_MOVE): 0:(-1.0,-5.0)
:Sending Trackball (ACTION_MOVE): 0:(-1.0,1.0)
:Sending Trackball (ACTION_MOVE): 0:(-4.0,0.0)
//[calendar_time:2018-04-20 15:57:55.842 system_uptime:13506636]
// Sending event #200
:Sending Touch (ACTION_DOWN): 0:(517.0,704.0)
:Sending Touch (ACTION_UP): 0:(556.1829,812.5558)
```

```
// Sending event #400
:Sending Trackball (ACTION_MOVE): 0:(-4.0,3.0)
:Sending Trackball (ACTION_UP): 0:(0.0,0.0)
:Sending Touch (ACTION_DOWN): 0:(507.0,678.0)
:Sending Touch (ACTION_UP): 0:(429.4049,741.5254)
:Sending Touch (ACTION_DOWN): 0:(296.0,579.0)
:Sending Touch (ACTION_UP): 0:(208.45094,621.8547)
:Sending Touch (ACTION_DOWN): 0:(478.0,601.0)
:Sending Touch (ACTION_UP): 0:(474.52704,599.7329)
:Sending Trackball (ACTION_MOVE): 0:(4.0,-1.0)
:Sending Touch (ACTION_DOWN): 0:(924.0,65.0)
:Sending Touch (ACTION_UP): 0:(901.5055,61.27103)
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHE
=0x10200000;component=com.example.adrian.firebase/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.catego
cmp=com.example.adrian.firebase/.MainActivity } in package com.example.adrian.firebase
:Sending Touch (ACTION_DOWN): 0:(240.0,1533.0)
:Sending Touch (ACTION_UP): 0:(228.23294,1570.9656)
:Sending Trackball (ACTION_MOVE): 0:(-2.0,-5.0)
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHE
=0x10200000;component=com.example.adrian.firebase/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.catego
cmp=com.example.adrian.firebase/.MainActivity } in package com.example.adrian.firebase
// Rejecting start of Intent { act=android.intent.action.VIEW dat=notification.log: cmp
android.apps.maps/com.google.android.apps.gmm.notification.log.NotificationLoggingActivity }
com.google.android.apps.maps
:Sending Touch (ACTION_DOWN): 0:(17.0,1514.0)
:Sending Touch (ACTION_UP): 0:(38.61078,1507.3988)
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHE
=0x10200000;component=com.example.adrian.firebase/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.catego
cmp=com.example.adrian.firebase/.MainActivity } in package com.example.adrian.firebase
:Sending Touch (ACTION_DOWN): 0:(317.0,654.0)
// Rejecting start of Intent { act=android.settings.APPLICATION_DEVELOPMENT_SETTINGS cm
.settings/.development.DevelopmentSettingsDisabledActivity } in package com.android.setting
:Sending Touch (ACTION_UP): 0:(248.69824,694.5095)
:Sending Touch (ACTION_DOWN): 0:(133.0,160.0)
Events injected: 500
:Sending rotation degree=0, persist=false
:Dropped: keys=1 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=6382ms (0ms mobile, 0ms wifi, 6382ms not connected)
// Monkey finished
```

Analysis

After I performed the Stress test using monkey it was successful. The results show the different actions that the pseudocode code performed 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.

```
:Sending rotation degree=0, persist=false
:Dropped: keys=1 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=10816ms (0ms mobile, 0ms wifi, 10816ms not connected)
// Monkey finished
```

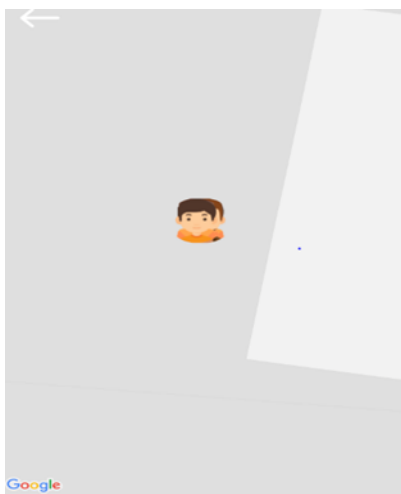
The Stress test stops when the application data access is removed. This is because a lot of the applications sources are dependent on a third party (The Server).

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.

```
Bitmap smallMarker = Bitmap.createScaledBitmap(b, width, height, false);
Bitmap smallMarker1 = getCircleBitmap(smallMarker);
mMap.addMarker(new MarkerOptions().position(location).title(studentList.get(i).getfName()).
```



8. 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. "xxxxxxx@gmail.com" where a different address e.g. xxxxxxx@hotmail.ie would not be applicable. As the application stands it is too 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 sections 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.

9. 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.

```
location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

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:

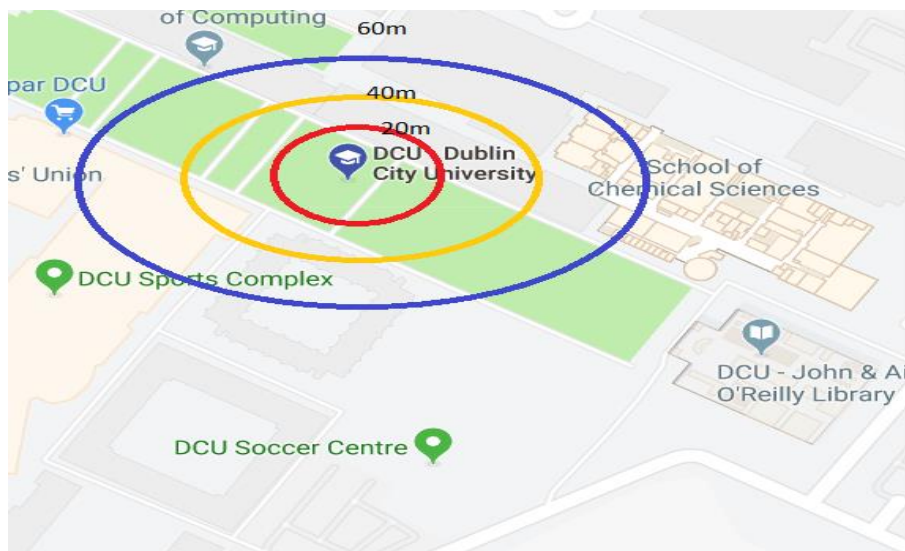
- The Location might not have been updated as it was in a similar location.
- Before I tested my third testcase I was inside a room testing the location and the GPS signal would find it difficult to penetrate through buildings.
- The location on the device may have been turned off.

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	-6.236533
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://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b>

<https://developer.android.com/studio/test/monkey>

<https://geteasyqa.com/qa/best-test-case-templates-examples/>

<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/>

