TEST PRECEDURES

Report for TUmap Application

Adama Coulibaly
Willy Lulciuc
David Mason
Sean Morgan
Ben walker

December 10, 2012

Contents

1. System Overview

Project Abstract

TUmap is a highly customized campus specific Android application that aims to ease navigation around campus for new students and guests. It will be Android based and levy Google Maps API for routing and an open source mapping toolkit for our custom map. The interface will include a simple drop down menu of buildings on campus. Once a building is selected, a user's location will be automatically polled and a route will be traced to their destination. As the user moves, the phone will update the GPS location and the route if necessary.

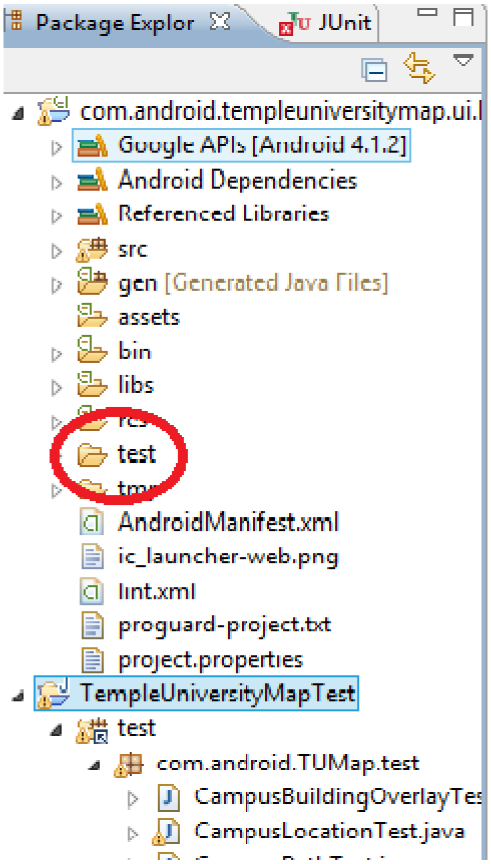
2. Document Overview

This Test Procedures document is to describe the steps involved in TUmap testing. There are several possible test methods, and the most appropriate one depends whether the test can be performed in a flexible and appropriate way. Some Test procedures are more time consuming and others can be inconclusive. This document covers the testing method using Robolectric. Robolectric is unit test framework for Android application. Using Robolectric tests run on Java Virtual Machine which is must faster than running tests on an Android emulator or device. Instead of deploying the application and tests to the Android emulator (which can be very slow), Robolectric runs your tests directly in your computer's JVM, reducing typical test turnaround times from minutes to seconds.   This document describe the following test method:

- Configuration of TempleUniversityTest Project
- Unit tests results
- Integration tests results
- Acceptance test results

3. Configuration of Test Project

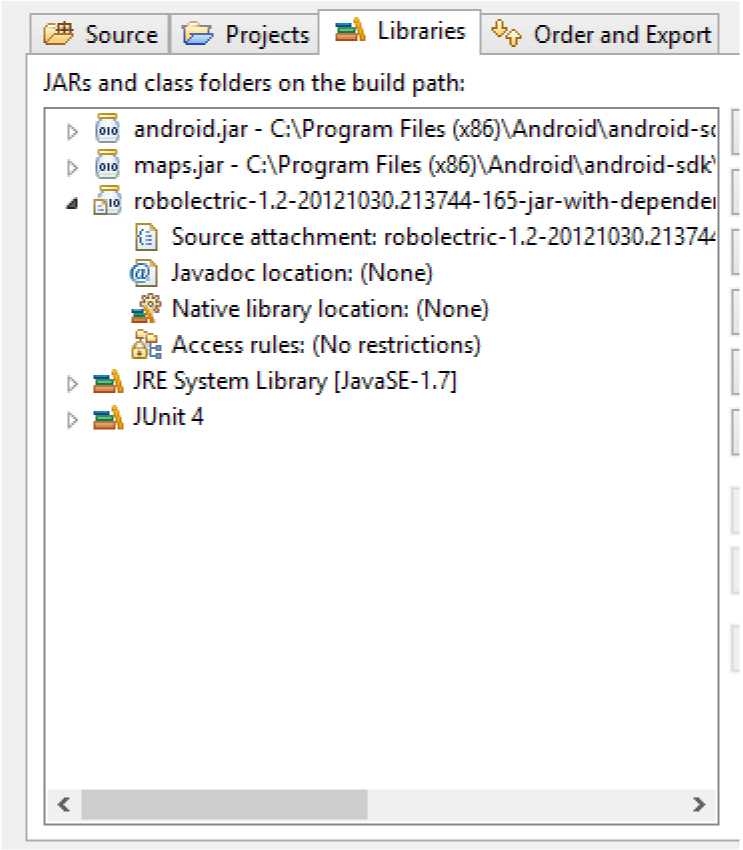
3.a Create a test source folder under TempleUniversityMap project



```
com.android.templeuniversitymap.ui.l
    Google APIs [Android 4.1.2]
    Android Dependencies
    Referenced Libraries
    src
    gen [Generated Java Files]
    assets
    bin
    libs
    res
    test
    tmp
    AndroidManifest.xml
    ic_launcher-web.png
    lint.xml
    proguard-project.txt
    project.properties
TempleUniversityMapTest
    test
        com.android.TUMap.test
            CampusBuildingOverlayTes
            CampusLocationTest.java
```

3.b Create and configure Java project TempleUnivermapTest

- ○ Add JUnit library : Robolectric does not support JUnit3 so we choose JUnit4
- ○ Add Robolectric jar: using Robolectric-with-dependencies Jar. we don't need to setup maven
- ○ Add Android Jars
- ○ Add maps Jar



JARs and class folders on the build path:

```
▷ android.jar - C:\Program Files (x86)\Android\android-s
▷ maps.jar - C:\Program Files (x86)\Android\android-sdk\
◢ robolectric-1.2-20121030.213744-165-jar-with-depende
     Source attachment: robolectric-1.2-20121030.21374
     Javadoc location: (None)
     Native library location: (None)
     Access rules: (No restrictions)
▷ JRE System Library [JavaSE-1.7]
▷ JUnit 4
```

4. Unit Testing

4.a Test Case 1: CampusLocationTest

Procedure:

- **Create test Run Configuration: TUmapTestConfig**
- **Seclect CampusLocationTest**
- **import com.xtremelabs.robolectric.RobolectricTestRunner;**
- **import org.junit.runner.RunWith;**
- **import static org.hamcrest.CoreMatchers.equalTo;**
- **import static org.junit.Assert.assertThat;**
- **add @RunWith(RobolectricTestRunner.class**

Function Tested:

- test_getBuildingName()
- test_getAbbreviation()
- test_getAddress()

- test_getCity()
- test_getState()
- test_getZipCode()
- test_getGeoLocation

Code

```
import static org.hamcrest.CoreMatchers.equalTo;
import static org.junit.Assert.*;
import static org.junit.Assert.assertThat;

import com.android.templeuniversitymap.util.CampusLocation;
import com.xtremelabs.robolectric.RobolectricTestRunner;





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

@RunWith(RobolectricTestRunner.class)
public class CampusLocationTest {
       CampusLocation CampLoc = new CampusLocation("Anderson Hall",
                                         "ANDRSN",
                                         "1114 W Berks Street",
                                         "Philadelphia",
                                         "PA",
                                         "19122",
                                         "39.980820",
                                         "-75.152680");

       @Test
      public void TestGetBuildingName() {
             String buildingName = CampLoc.getBuildingName();
             assertEquals("Expeted Value", buildingName, "buildingName" );
      }


      public void getAbbreviation() {
             String abr = CampLoc.getAbbreviation();
             assertEquals("Expeted Value", abr, "ANDRN" );
      }


       @Test
      public void getAddress() {
            String add = CampLoc.getAddress();
             assertEquals("Expeted Value", add, "1114 W Berks Street" );
```

```
    }

     @Test
    public void getCity() {
        String city = CampLoc.getCity();
         assertEquals("Expeted Value", city, "philadelphia" );
    }

     @Test
    public void getState() {
        String state = CampLoc.getState();
         assertEquals("Expeted Value", state, "PA" );
    }


    public void getZipCode() {
        String zip = CampLoc.getZipCode();
         assertEquals("Expeted Value", zip, "19122" );
    }


     @Test
    public void getLatitude() {
        String getla = CampLoc.getLatitude();
         assertEquals("Expeted Value", getla, "39.980820" )
    }

     @Test
    public void getLongitude() {
        String getLon = CampLoc.getLongitude();
         assertEquals("Expeted Value", getLon, "-75.152680" );
    }


}
```

4.bTest Case 2: BuildingPath

Procedures

- **Create test Run Configuration: TUmapTestConfig**
- **Seclect BuildingPathTest**
- **Select each method to be tested**
- **import com.xtremelabs.robolectric.RobolectricTestRunner;**
- **import org.junit.runner.RunWith;**
- **import org.hamcrest.CoreMatchers.equalTo;**
- **import org.junit.Assert.assertThat;**
- **add @RunWith(RobolectricTestRunner.class)**

Function Tested:

- **testShowPathFrom()**
- **testsetEndLongitude()**
- **testsetEndLongitude()**
- **testSetEndLatitude()**

Code

```
package com.android.TUMap.test;

import static org.junit.Assert.assertEquals;

import com.android.templeuniversitymap.util.CampusPath;
import com.xtremelabs.robolectric.RobolectricTestRunner;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.osmdroid.util.GeoPoint;

@RunWith(RobolectricTestRunner.class)
public class CampusPathTest {
      final CampusPath campusP = null;

      @SuppressWarnings("null")
      @Test
      public void testShowPathFrom(double  startLatitude, double  startLongitude) throws
Exception {

            final double startlatitude =39.980770;
        final double startlongitude = -75.156883;
        final double endLatitude =39.977922;
        final double endLongitude = -75.156366;
        final GeoPoint end = new  GeoPoint(endLatitude, endLongitude);
        final GeoPoint start = new  GeoPoint(startlatitude, startlongitude);
        campusP.calculateDirections(startlatitude, startlongitude, endLatitude,
endLongitude);
        assertEquals("Direction Path", start, end );
      }


      @SuppressWarnings("unused")
      @Test
      private void testCalculateDirections() throws Exception {
            // Not Implemented
      }
```

```
	@Test
	public void testsetEndLongitude(double in){
		 final double endLongitude =39.977922;
			campusP.setEndLongitude(endLongitude);
			assertEquals(" End Latitude", in, endLongitude);

	}


	@Test
	public void testSetEndLatitude(double in){
		 final double endLatitude =39.977922;
		campusP.setEndLatitude(endLatitude);
		assertEquals("toFromStri altitude", in, endLatitude);

	}


}
```

5. Integration Testing

5.a CampusOverlayTest and MapActivityTest

Procedure:

- **Create test Run Configuration: TUmapTestConfig**
- **Seclect CampusOverlayTest and MapActicityTest**

Method tested and Expected output

Class	**Method**	**input**	**Expected output**
CampusMapActivityTest	onCreate(Bundle savedInstanceState)	Previously saved state of the application	User Interface
	onCreateOptionsMenu(Menu menu)	Menu object	User Interface component

	onOptionsItemSelected(MenuItem item)	The selected Menu item	Loads Activity Which corresponds To selected item
	changeClearMapItemState(boolean state)	Boolean corresponding To whether ClearMap is enabled	Clear Map item state changes
	onDestroy()	null	DB connection is closed
	onClose()	null	SearchMenu closes
	onQueryTextSubmit(String query)	String of text To be searched For in database	Elements in Database which Match query
	test_draw()	Set Canvas and MapView	Marker is loaded from resources and draw on the map
	test_FoodOverlay()	Food marker	Display food icon
	test_GetAllOverlay()	null	All overlay (food and building)

	test_setMarker(final Bitmap marker)	Overlay icon	Marker in resource Should be display

6. Acceptance Testing

6.a showDirectionsTo()

Description

Verify that route is displayed from one static point (start) to another static point (End)

Precondition

User must know End location

Input

End geolocation coordinates (end Latitude, and end Longitude)

Output

Display route from current location to End location

6.b onOptionsItemSelected()

Description

Provide the option to select destination location from Menu

Precondition

know the name of building name

Input

Enter initial of building name

Output

Display option menu to select specific building starting with Initial

6.c FoodOverlay()

Description

Display food overlay marker on the map

Precondition

Know existing restaurant and food store location on campus

Input

Enter the geolocation coordinates of restaurant

Output: Display marker on specific location

6.d BuildingOverlay()

Description

Display marker for each building

Precondition

Know different building name and location

Input

Enter the geolocation coordinates of building

Expected Output

Display building marker on the map

6.e onItemLongPress()

Description

Show information of overlay when click

Precondition

Click on icon

Input: overlay item

Output

Display overlay title, description and name.

7 Responsibility

7.1 Developers

Developers test their code and provided functionality during the development process so basic bugs are avoided. It is also necessary to thoroughly read (and understand) all specification and documentation available to make sure that the implementation is compatible with the specification.

Name of Developers	Roles
Adama Coulibaly	Support Manager
Willy lulciuc	Development Manager
David Mason	Planning Manager
Sean Morgan	Team Leader
Ben Walker	Quality/Process Manager