System Validation document

Prepared by

Amala Rosi Mariadass

Student No:21816636

System Validation Document

The UWA smart parking

1 Purpose:

The purpose of this document is to structure and outline the different types of tests that were executed in the development and evaluation of UWA smart parking project.

This project is the user-friendly oriented project where the front end and back end system. The result of this various testing effort has been performed in hand in order to test the performance of the application. Validation of product which ensures meets the requirement of the client, which is executed by the user application.

2.Guiding Principles:

The main principle to approach for testing in UWA smart parking was the main focus, the system was built each developer was responsible for testing and the quality of the product which they designed. The development phase developed by the developer ensures all the test cases are covered, the expected outcome they have tested in depth and promises the quality of the product. The prime focus in testing the user experience of the application, the steps involved in using the system.

3.System Design Description:

UWA's student population is growing, putting pressure on the limited spaces available for parking. Long times spent searching for a parking space raises stress levels and also increases UWA's carbon footprint. The task in this project is to research, design, and prototype a system to provide accurate information on parking availability to students and staff driving to campus. Hence, we have designed a user-friendly mobile application where the student can park the car based on where the available space in the university. This is developed and implemented using UWA_smart Parking team finding the student with yellow permission and staff with red permission. Using Geo fencing technology, we designed the application where exactly the person has parked the and pop up the geo-coordinates. And also inform where the student has parked the car in which parking lot and remaining how many parking lots are available using heat map.

4.Quality and test objectives:

The following quality key attributes have given preference and tested on the basis for the different approach in specific to attain the expected test target. The different test methodology used here are:

- To verify the functionality baseline and the application features.
- Check the design of the application and solutions against user requirements early in the development process.
- To test and validate executable code against design during future stages of the development process.
- Compatibility testing of the software during running the application.
- To validate and expose the entire application to unexpected events, faults in database, networks or different unpredictable user behaviour.
- To determine the application or system which not meets design requirement, but also usable and meets the needs of its users.

Attributes	Description	Measure and Target	Priority
Correctness	Functions and events work as calculated	Achievement of all agreed features and No defects that prevent the real time working correctness of these feature	Must have
Integrity	To prevent unauthorised access, data security and protect data privacy.	Use cloud for accessing the data, the data is connected through web socket and access is given. The user information is saved in cloud and pass word are encrypted.	Must have
Maintainability	Whether is accessible to add the features, correct the defects or any	System is kept modular as possible by the developers.	Should have

	release changes to the system		
Availability	The planned application which is operational	100% of the system in deployment environment	Should have
Interoperability	Ease of which system can connect with other application	Server should run on operating system with necessary location	Must have
User Experience	Response and ease of use of the user interface of the system	Users do not require demonstration to operate the application. User satisfaction are positive using the system.	Must have

5 Test scope

The scope of testing for UWA smart parking project mainly focuses on the functionality of the application developed as part of the project, its integration with and ability to use different libraries. The quality and perfection are needed when using the mobile application, it should be designed to be user friendly and deliver to client.

In scope:

Ability facilities the parking solution for students/staff of UWA through UWA parking application.

- Database connected with cloud for users
- Overall User experience
- User account Management and associated security with the system.
- Backend testing with Junit.

Test Approaches

Test types

Type	Definition	Test Method	Execution/Timing

Unit testing	Testing which each unit and verifies the implementation of Application	Junit	After backend and front end was finished.
Interface testing	Testing the Mobile application through user interface	Appium	After UWA_parking application compleetd
Functional Testing	Testing the software written to verify the application	Manual testing by users	Whenever updating the application after release of the application
Documentation Validation	Correct format and in GitHub	Document	User and client review documentation

5.1Unit Testing:

Mobile testing application is done in android studio using Model view presenter architecture, the activity framework is used to validate each units and testing is done using various application and through various controls the view, control and flow of the whole main activity of the UWA_parking application.

The testing is done for various classes of the mobile application created for the UWA parking application using Junit testing

Mainactivitytest.java:

This class test verifies the login user and student id has passed correct input and includes all the possible test case. The testcase for each student number is verified and weather the password should not be empty. It passes each variable and testing verified done and verified. It passes all possible test case. The code is attached and framed below.

Test case	MainactivityTest for login page of UWA parking app
Description	Testing the main Login page of UWA mobile application
System Environment	Junit, Android Studio and emulator
Steps	1.Start the emulator using Android studio.

	2.Pass values student 3.Pass password. 4.If student number is string it passes error. 5.If wrong password error pop up error and doesn't move to next page 6.Tested by giving various input of student Number and Password.
Expected Results	1. Emulator should be started. 2. Tested the MainActivity. 3. Run the Mainactivitytest.java 4. No bugs found
Tested Feature	Pass
Tested by	Amala

Table1.Junit Testcase for Mainactivity.java

This is the test code written for checking the Mainactivitytest.java

```
package com.example.test.uwa_parking;

import android.app.Activity;
import android.app.Instrumentation;
import android.support.test.rule.ActivityTestRule;
import android.view.View;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;

import static android.support.test.InstrumentationRegistry.getInstrumentation;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static org.junit.Assert.*;

public class MainActivityTest {

    @Rule
    public ActivityTestRule<MainActivity> mMainActivityRule = new ActivityTestRule<>(MainActivity.class);


    private MainActivity mActivity = null;

    Instrumentation.ActivityMonitor monitor = getInstrumentation().addMonitor(register.class.getName(),null,false);


    @Before
    public void setUp() throws Exception {

        mActivity = mMainActivityRule.getActivity();

    }


    @Test
    public void testLaunchSecondActivityOnButtonClick(){
        assertNotNull(mActivity.findViewById(R.id.register));
```

```
        onView(withId(R.id.register)).perform(click());

        Activity register = getInstrumentation().waitForMonitorWithTimeout(monitor,5000);

        assertNotNull(register);

        register.finish();
    }


    @Test
    public void testLaunch(){
        View et_name = mActivity.findViewById(R.id.et_name);
        assertNotNull(et_name);

        View et_pass = mActivity.findViewById(R.id.et_pass);
        assertNotNull(et_pass);

        View cb = mActivity.findViewById(R.id.cb);
        assertNotNull(cb);

        View login = mActivity.findViewById(R.id.login);
        assertNotNull(login);

        View login_result = mActivity.findViewById(R.id.login_result);
        assertNotNull(login_result);

    }

    @After
    public void teardown() throws Exception {
        mActivity = null;
    }


}
```

Registertest.java: It checks all the registered featured are correct the registering page is tested each and every attribute of the mobile application. All the possible expected test case are tested and results are produced.

Test case	Junit test case2
Description	Testing the registerTest.java UWA mobile application
System Environment	Junit,Android Studio and emulator
Steps	1.Start the emulator using Android studio. 2.Pass values student Number 3.Pass password. 4.If wrong student is string it passes error. 5.If student and presses staff button pops error and doesn't pass the case.

	6.If wrong password entered wrong doesn't pass the test and fails
Expected Results	1. Emulator should be started. 2. Tested the all the attributes of Register page. 3. Run the registerTest.java 4. No bugs found
Tested Feature	Pass
Tested by	Amala

Table2.Junit Testcase for registertest.java

This the code to test register page of the UWA-parking mobile application

```
package com.example.test.uwa_parking;

import android.support.test.rule.ActivityTestRule;
import android.view.View;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;

import static org.junit.Assert.*;

public class registerTest {
    @Rule
    public ActivityTestRule<register> mRegisterRule = new ActivityTestRule<>(register.class);


    private register mActivity = null;

    @Before
    public void setUp() throws Exception {
        mActivity = mRegisterRule.getActivity();
    }


    @Test
    public void testLaunch(){
        View et_name = mActivity.findViewById(R.id.et_name);
        assertNotNull(et_name);

        View et_pass = mActivity.findViewById(R.id.et_pass);
        assertNotNull(et_pass);

        View et_repass = mActivity.findViewById(R.id.et_repass);
        assertNotNull(et_repass);

        View et_email = mActivity.findViewById(R.id.et_email);
        assertNotNull(et_email);


        View rg_role = mActivity.findViewById(R.id.rg_role);
        assertNotNull(rg_role);

        View rb_staff = mActivity.findViewById(R.id.rb_staff);
        assertNotNull(rb_staff);

        View rb_student = mActivity.findViewById(R.id.rb_student);
        assertNotNull(rb_student);

        View rg_permission = mActivity.findViewById(R.id.rg_permission);
        assertNotNull(rg_permission);

        View rb_red = mActivity.findViewById(R.id.rb_red);
        assertNotNull(rb_red);


        View rb_yellow = mActivity.findViewById(R.id.rb_yellow);
        assertNotNull(rb_yellow);

        View sign in = mActivity.findViewById(R.id.signin);
```

```
        assertNotNull(signin);

        View goback = mActivity.findViewById(R.id.goback);
        assertNotNull(goback);


        View register_result = mActivity.findViewById(R.id.register_result);
        assertNotNull(register_result);


    }

    @After
    public void tearDown() throws Exception {
        mActivity = null;

    }
}
```

3.Successtest.java: success entered the login to find the GPScoordinates and find parking lot screen and proceeds to map page.

Test case	Junit test case3
Description	Testing the sucessTest.java UWA mobile application
System Environment	Junit, Android Studio and emulator
Steps	1.Start the emulator using Android studio. 2.Eneterd the GPS coordinates. 3.In button and out button is tested 4.User finds the carpark lot. 5.All the map feature and different scenarios are checked
Expected Results	1. Emulator should be started. 2. Tested the all the attributes of Register page. 3. Run the successtest.java 4. No bugs found
Tested Feature	Pass
Tested by	Amala

Table2.Junit Testcase for succestest.java

Test suite for Success Page in tested the following code is shown below.

```
package com.example.test.uwa_parking;

import android.app.Activity;
import android.app.Instrumentation;
import android.content.Context;
import android.content.Intent;
import android.graphics.ColorSpace;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Parcel;
import android.os.Parcelable;
import android.support.test.InstrumentationRegistry;
import android.support.test.rule.ActivityTestRule;
import android.view.View;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;

import static android.support.test.InstrumentationRegistry.getInstrumentation;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static org.hamcrest.Matchers.instanceOf;
import static org.hamcrest.Matchers.is;
import static org.hamcrest.core.IsNull.notNullValue;
import static org.junit.Assert.*;

public class successTest {


    @Rule
    public ActivityTestRule<success> mSuccessRule = new ActivityTestRule<success>(success.class){
        @Override
        protected Intent getActivityIntent() {
            InstrumentationRegistry.getTargetContext();
            Intent intent = new Intent(Intent.ACTION_MAIN);
            intent.putExtra("username", "Hello");
            return intent;
        }
    };


    private success mActivity = null;

    Instrumentation.ActivityMonitor monitor = getInstrumentation().addMonitor(map.class.getName(),null,false);


    @Before
    public void setUp() throws Exception {
        mActivity = mSuccessRule.getActivity();

    }


    @Test
    public void testLaunchSecondActivityOnButtonClick(){
        assertNotNull(mActivity.findViewById(R.id.button_show));

        onView(withId(R.id.button_show)).perform(click());

        Activity success = getInstrumentation().waitForMonitorWithTimeout(monitor,5000);

        assertNotNull(success);

        success.finish();
    }


    @Test
    public void ensureIntentDataIsDisplayed() throws Exception {
        success activity = mSuccessRule.getActivity();

        String username = null;


        assertNull(username);


    }



    @Test
    public void testLaunch(){
        View ed1 = mActivity.findViewById(R.id.ed1);
        assertNotNull(ed1);

        View ed2 = mActivity.findViewById(R.id.ed2);
        assertNotNull(ed2);

        View park_result = mActivity.findViewById(R.id.park_result);
        assertNotNull(park_result);

        View button_in = mActivity.findViewById(R.id.button_in);
        assertNotNull(button_in);

        View button_out = mActivity.findViewById(R.id.button_out);
```

```
        assertNotNull(button_out);

        View button_gps = mActivity.findViewById(R.id.button_gps);
        assertNotNull(button_gps);
    }

    @After
    public void tearDown() throws Exception {
        mActivity = null;
    }
}
```

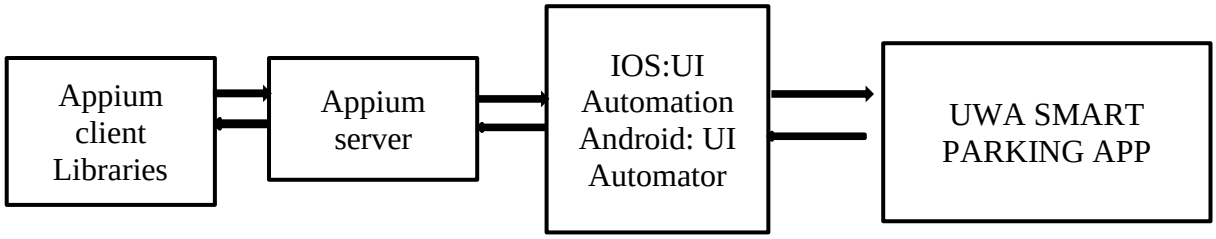
Unit Testing and MVC Presenter Testing:

Next step I tested the Login activity of the main screen using Android studio and MVC control presenter so the login activity itself fails at the login page. And doesn't move to the next screen.

Test case	Junit test case4
Description	Testing the sucessTest.java UWA mobile application
System Environment	Junit, Android Studio and emulator
Steps	1.Start the emulator using Android studio. 2.Eneterd the GPS coordinates. 3.Login activity passed giving correct input if pass empty or error in password does not proceed to login Page
Expected Results	1. Emulator should be started. 2. User id empty 3. Password empty 4. If user is not a string 5. Passing student and login details incorrect. 6. It fails to next register page.
Tested Feature	Pass
Tested by	Amala

The code for this in appendix below attached to test the initial stage of the mobile application screen for valid User using MVC and android test.

5.2Interface Testing: The interface testing is done by the Appium tool, its mainly a UI interaction testing where, the Appium framework which is used for mobile application we developed for the smart UWA parking. The Appium is a selenium framework which can support multiple different.



The following tools and libraries are required on the testing and development application:

- Java IDE c:\> java -version
- Java JDK C:\Program Files\Java\jdk1.8.0_92
  Android SDK
- Junit
- Emulator/Real Device
- Appium server
- Node.js C:\Users\yourUser\AppData\Roaming\npm

Environment Settings for functional testing are:

- platforms-tools folder which has path
  – C:\Users\youruser\AppData\Local\Android\sdk\platform-tools
- tools folder which has path – C:\Users\ your user \AppData\Local\Android\sdk\tools
- bin folder inside tools folder with path
  – C:\Users\youruser\AppData\Local\Android\sdk\tools\bin

The desired capabilities which are needed for setting up Appium webserver with the emulator are:

This is python code for setting the Appium server with the local host.

```
from appium import webdriver

import time

caps = {}

caps["platformName"] = "Android"

caps["platformVersion"] = "6.0"

caps["deviceName"] = "OPPO R9s"
```

```
caps["appActivity"] = ".MainActivity"

caps["appPackage"] = "com.example.test.myapplication"

caps["autoGrantPermissions"] = "true"

driver = webdriver.Remote("http://localhost:4723/wd/hub", caps)
```

The Server setting to connect automatic to server are:

```
{
  "platformName": "Android",
  "platformVersion": "6.0",
  "deviceName": "OPPO R9s",
  "appActivity": ".MainActivity",
  "appPackage": "com.example.test.myapplication",
  "autoGrantPermissions": "true"
}
```

Python code for tool setup and web driver capabilities.

These are the following steps taken in interface testing and those are followed bellow weather the mobile application fits the correct UI testing.

1.The interface testing code using Appium tool for the given below for login screen UWA_Parking:

```
el1 = driver.find_element_by_id("com.example.test.myapplication:id/et_studentno")

el1 = driver.find_element_by_id("android:id/action_bar")

el1.click()

el1.send_keys("21816636")

el2 = driver.find_element_by_id("com.example.test.uwa_parking:id/et_pass")

el2.send_keys("21816636")

el2.click()

el3 = driver.find_element_by_id("com.example.test. uwa_parking:id/register")

el3.click()
```

2. To Test registration screen of UWA_Parking:

```
el1 = driver.find_element_by_id("com.example.test.uwa_parking:id/et_studentno")

el1 = driver.find_element_by_id("android:id/action_bar")

el1.click()

el1.send_keys("21816636")

el2 = driver.find_element_by_id("com.example.test.myapplication:id/et_pass")

el2.send_keys("******")

el2.click()
```

```
el2 = driver.find_element_by_id("com.example.test.myapplication:id/et_repass")
el2.send_keys("******")
el2.click()
el3 = driver.find_element_by_id("com.example.test.myapplication:id/et_emailid")
el3.send_keys("21816636@student.uwa.edu.au")
el3.click()
el4 = driver.find_element_by_id("com.example.test.myapplication:id/et_student")
el4.click()
el5 = driver.find_element_by_id("com.example.test.myapplication:id/et_Yellow ")
el5.click()
el5 = driver.find_element_by_id("com.example.test.myapplication:id/et_signin ")
el5.click()
```

3. **To Test registration screen of UWA_Parking for Staff**

```
el1 = driver.find_element_by_id("com.example.test.uwa_parking:id/et_studentno")
el1 = driver.find_element_by_id("android:id/action_bar")
el1.click()
el1.send_keys("123456")
el2 = driver.find_element_by_id("com.example.test.myapplication:id/et_pass")
el2.send_keys("******")
el2.click()
el3 = driver.find_element_by_id("com.example.test.myapplication:id/et_repass")
el3.send_keys("******")
el3.click()
el4 = driver.find_element_by_id("com.example.test.myapplication:id/et_emailid")
el4.send_keys("123456@student.uwa.edu.au")
el4.click()
el5 = driver.find_element_by_id("com.example.test.myapplication:id/et_staff")
el5.click()
el6 = driver.find_element_by_id("com.example.test.myapplication:id/et_Red ")
el6.click()
```

```
el7 = driver.find_element_by_id("com.example.test.myapplication:id/et_signin ")
el7.click()
```

4. **To Test registration screen of GPS_Cordinates inside Geo fencing:**

```
el1 = driver.find_element_by_id("com.example.test.uwa_parking:id/et_latitude")
el1 = driver.find_element_by_id("android:id/action_bar")
el1.click()
el1.send_keys("******")
el2 = driver.find_element_by_id("com.example.test.myapplication:id/et_longitude")
el2.send_keys("******")
el2.click()
el3 = driver.find_element_by_id("com.example.test.myapplication:id/et_in")
el3.click()
el4 = driver.find_element_by_id("com.example.test.myapplication:id/et_showmap")
el4.click()
el5 = driver.find_element_by_id("com.example.test.myapplication:id/et_showgps")
el5.click()
```

5. **To Test registration screen of GPS_Cordinates Outside Geo fencing:**

```
el1 = driver.find_element_by_id("com.example.test.uwa_parking:id/et_latitude")
el1 = driver.find_element_by_id("android:id/action_bar")
el1.click()
el1.send_keys("******")
el2 = driver.find_element_by_id("com.example.test.myapplication:id/et_longitude")
el2.send_keys("******")
el2.click()
el3 = driver.find_element_by_id("com.example.test.myapplication:id/et_out")
el3.click()
el4 = driver.find_element_by_id("com.example.test.myapplication:id/et_showmap")
el4.click()
```

5.3 Functional Testing:

The functional testing of UWA_Parking normally consists in the areas of testing user interactions as well as testing the transactions with different manual test case procedures for testing. The various factors which are relevant in functional testing are:

1. Type of application based upon the business functionality usages
2. Target audience type (student, Staff, Admin)
3. Distribution channel which is used to spread the application.

The most fundamental test scenarios in the functional testing can be considered as:

1. To validate whether all the required mandatory fields are working as required.
2. To validate that the mandatory fields are displayed in the screen in a distinctive way than the non-mandatory fields.
3. To validate whether the application works as per as requirement whenever the application

Project Name: UWA Smart Parking	
Test Case	
Test Case ID: UWA01	Test Designed by: Amala
Test Priority (Low/Medium/High): Medium	Test Designed date: 28/09/2018
Module Name: UWA front end screen	Test Executed by: Amala
Test Title: Test login Functionality	Test Execution date: 28/09/2018
Description:	Verify login with user name and password
Pre-conditions: User has valid user name and password	
Dependencies: No depend ices	

Step	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**	**Notes**
1	Navigate to login page		User should be able to login	User is able to login	Pass	
2	Provide valid username	User =21816636	Credential can be entered	As expected	Pass	
3	Provide Valid password	Password =21816636	Credential can be entered	As expected	Pass	

4	Click on Login button		User logged	User logged successfully	Pass	

Post-conditions: User is validated with database and successfully login to the account. The account session details are logged in the database server.

Test case2: To validate with the cloud server database and logged in server.

Project Name: UWA Smart Parking	
Test Case	
Test Case ID: UWA02	Test Designed by: Amala
Test Priority (Low/Medium/High): Medium	Test Designed date: 28/09/2018
Module Name: UWA front end screen	Test Executed by: Amala
Test Title: Test login Functionality	Test Execution date: 28/09/2018
Description:	Verify login with user name and password
Pre-conditions: User has valid user name and password	
Dependencies: UWA_parking application	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page		User should be able to login	User is able to login	Pass	
2	Provide valid username	User = 1234	Credential can be entered	As expected	Pass	
3	Provide Valid password	Password =1234	Credential can be entered	As expected	Pass	
4	Click on Register button		User Registered	User registered successfully	Pass	
5	Provide valid Student Number	Student Number= 1234	User valid	User Student Number	Pass	
6	Provide Password	Password=1234	User password	User password	Pass	

Post-conditions: User is validated with database and successfully login to the account. The account session details are logged in the database server.

Test Case 3: User has Student no is different from Registration entered student name

Project Name: UWA Smart Parking	
Test Case	
Test Case ID: UWA03	Test Designed by: Amala
Test Priority (Low/Medium/High): High	Test Designed date: 28/09/2018
Module Name: UWA front end screen	Test Executed by: Amala
Test Title: Test login Functionality	Test Execution date: 28/09/2018
Description:	Verify login with user name and password
Pre-conditions: User has Student no is different from Registration entered student name	
Dependencies: UWA_Parking Application	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page		User should be able to login	User is able to login	Pass	
2	Provide valid username	User =2181 663678	Credential can be entered	As expected	Fail	
3	Click on Login button		User logged	User logged Failed	Fail	
Post-conditions: User entered wrong student number is more than 8 characters, hence login failed.						

Test case4: User entered wrong student number is more than 8 characters, hence login failed.

Project Name: UWA Smart Parking	
Test Case	
Test Case ID: UWA04	Test Designed by: Amala
Test Priority (Low/Medium/High): High	Test Designed date: 28/09/2018
Module Name: UWA front end screen	Test Executed by: Amala
Test Title: Test login Functionality	Test Execution date: 28/09/2018
Description:	Verify login with user name and password
Pre-conditions: User entered wrong student number is more than 8 characters, hence login failed.	
Dependencies: UWA_Parking Application	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes

1	Navigate to login page	User =21816636	User should be able to login	User is able to login	Pass	
2	Provide valid Password	User =********	Credential can be entered	As expected	Pass	
3	Navigate to Register button		User Registration entered	User Register	Pass	
4.	Provide student Number	User =2181663	User should enter correct email id	User is not able to login	Fail	Student number is different with login page and Registration Page
Post-conditions: User entered different student number and wrong student number in registration Page						

Test Case 5:To success login and finds the Parking lot in UWA geo fencing car park

Project Name: UWA Smart Parking	
Test Case	
Test Case ID: UWA05	Test Designed by: Amala
Test Priority (Low/Medium/High): High	Test Designed date: 28/09/2018
Module Name: UWA front end screen	Test Executed by: Amala
Test Title: Test login Functionality	Test Execution date: 28/09/2018
Description:	Verify login with user name and password
Pre-conditions: User in car park lot in UWA parking and display the GPS coordinates	
Dependencies: UWA_Parking Application	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User =21816636	User should be able to login	User is able to login	Pass	
2	Provide valid Password	User =********	Credential can be entered	As expected	Pass	
3	Navigate to Register button		User Registration entered	User Register	Pass	
4.	Provide student Number	User =21816636	User should enter correct email id	User is not able to login	Pass	
5.	Provide password	User=*******	Credential can be entered	As per expected	Pass	

6	Provide renter Password	User=*******	Credential can be entered	As per expected	Pass	
7	Please enter email address	User =21816636@student.uwa.edu.au	Credentials entered correctly	As per expected	Pass	
8	Select Student	Student click	Yellow tab	Yellow tab	Pass	
9	Navigate to Login Page	Sign -in	Sign -in	Sign -in	Pass	
10.	Navigate to GPS coordinates Page	Current Latitude	Latitude	Current Latitude	Pass	
11	Navigate to GPS coordinates Page	Current Longitude	Longitude	Current Longitude	Pass	
12	Navigate In option	Press in	IN	Press in	Pass	Successful
13	Navigate to google map	Current location	Car Park lot	Current GPS location	Pass	Successful
Post-conditions: User in car park lot in UWA parking.						

5.4 Documentation Validation:

To make sure all the requirement documents are in place for during the entire project and for future reference.

a)System Requirements Validation

Documentation Name	System Requirements Document and prototype
Author	UWA_Smart Parking team
Design Description	Functional and Non-functional Prototype design
Completion Date	4th June 2017
Content type	• Client, and UWA team members • Use cases

		• Functional and Non-functional requirement
Validation Criteria		
Document Review	Criteria	Documentation provides the required contents to be delivered. Documentation is formatted for second year process.
	Reviewer	UWA parking team
Requirements Validation	Criteria	Requirements as per client
	Methods of validation	• Team Peer review • Survey
	Verified by	DR.Rachel Cordell

Table5. System requirements validation

b) User Manual validation

Documentation Name		User Manual
Author		UWA_Smart Parking team
Design Description		In detail Mobile application installation in mobile
Completion Date		31st October 2018
Content type		• Application requirement • Front end and backend • Different functionalities for Staff and student of UWA • Reports of bug
Validation Criteria		
Document Review	Criteria	UWA_smart parking peer review
	Reviewers	UWA parking team
Application setup procedure	Criteria	Requirements as per client
	Methods of validation	• UWA team with team members mobile. • With other team members

	Verified by	UWA_parking team members

C) Maintenance Manual validation

Documentation Name		Maintenance manual
Author		UWA_Smart Parking team
Design Description		In detail Mobile application front end and backend process and procedures.
Completion Date		31st October 2018
Content type		• Backend setup • Front end set • Cloud server connection • Directory tree • Interface development
Validation Criteria		
Document Review	Criteria	UWA_smart parking peer review
	Reviewers	UWA parking team
Application setup procedure	Criteria	Requirements as per client
	Methods of validation	UWA team Verifying every setup Peer internal review
	Verified by	UWA_parking team members

Appendix:

Pictures of the mobile application testing using Android, Junit and Appium Tool.

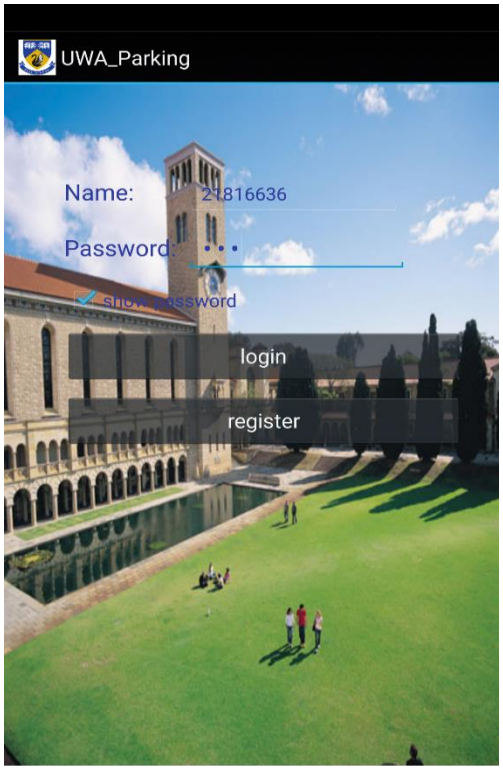


Figure1.Front Screen Mainactivity.Test

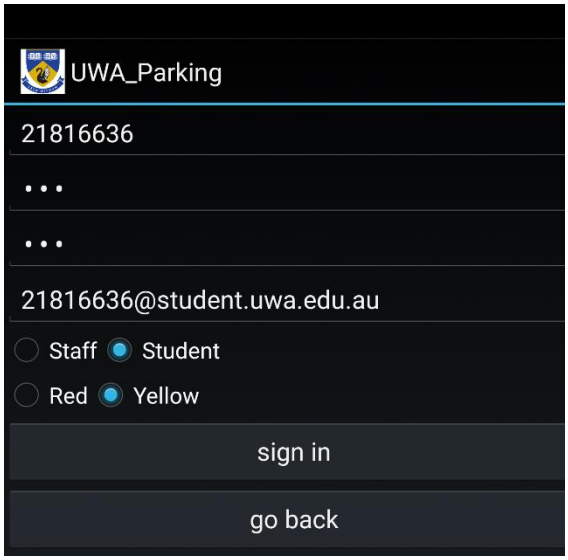


Figure2. Register Pagetest.Java testing

Pictures User Interface testing of mobile application

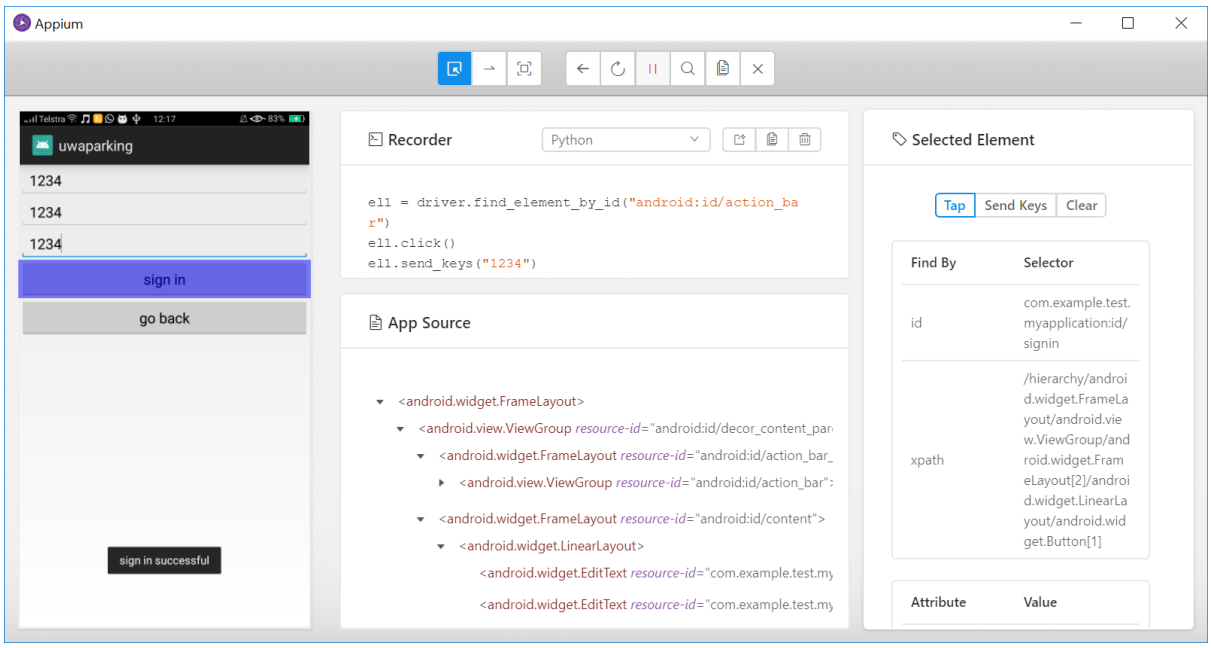


Figure3. Appium Inspector window for testing the login button and different testing sceanrios

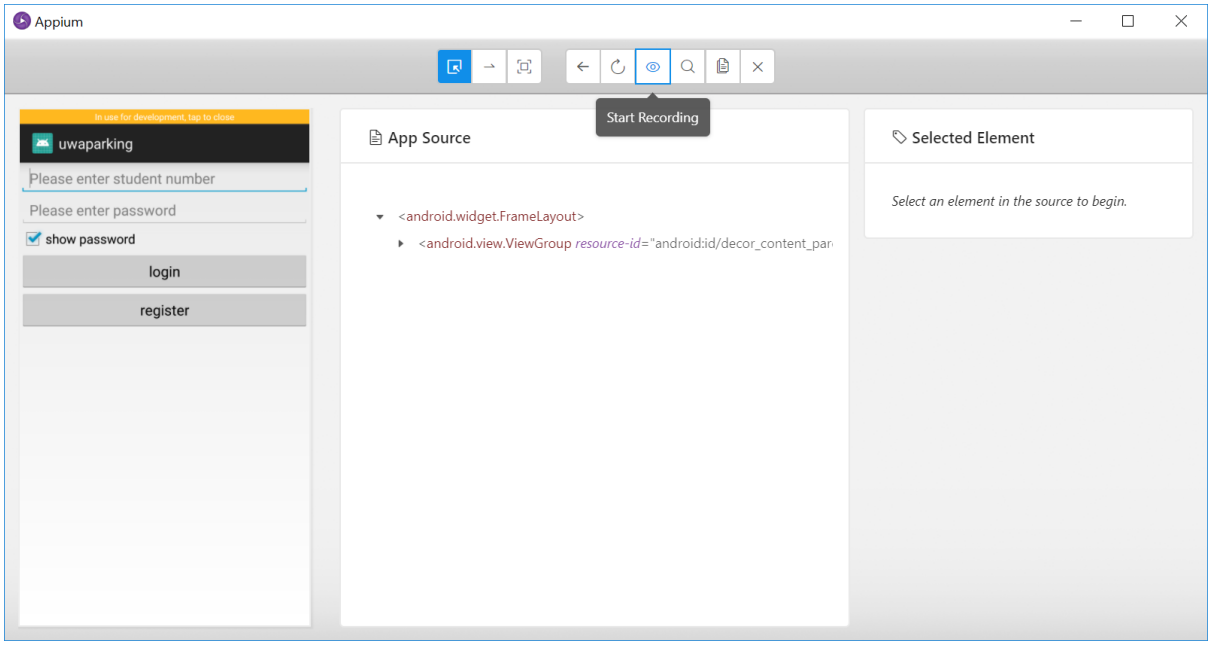


Figure4. How the Appium window records and pass each key to the attribute window.

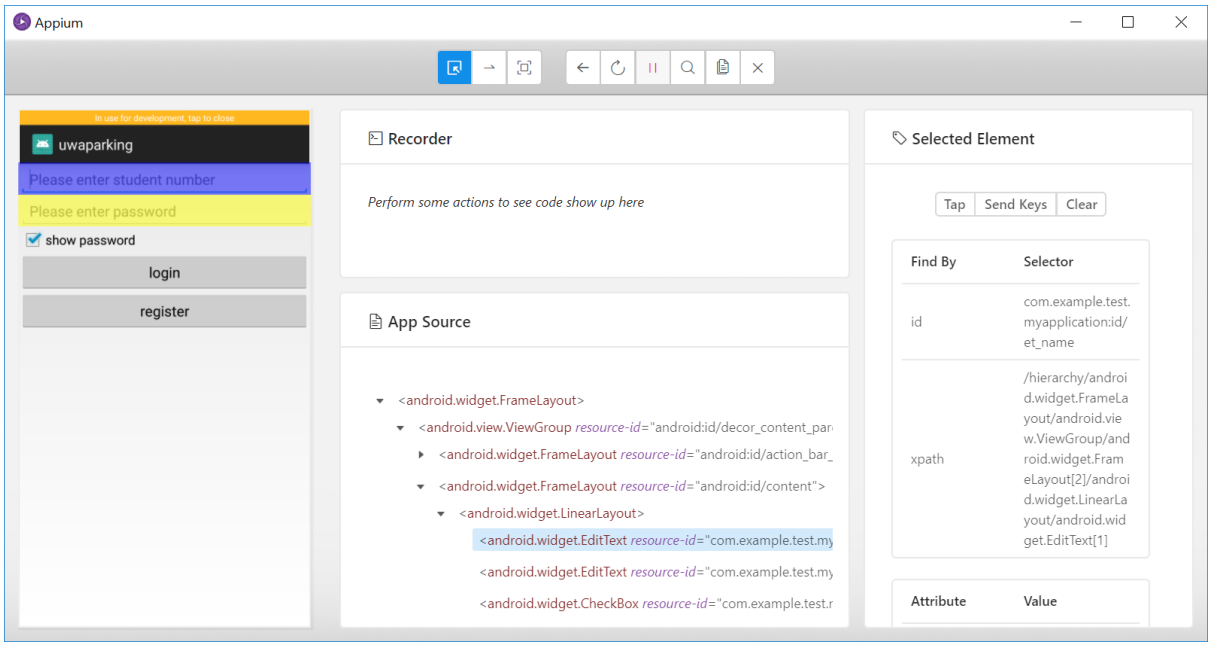


Figure5. Testing screenshots of each bar and button using send Keys

Code for MVC testing and Android for LoginPresentertest:

```
Package com.example.test.uwa_parking.login;

import org.juint.Before;
```

```
import static org.junit.Assert.*;
@Runwith(MOCKitoJUnitRunner.class)
public class MainactivityTest{
   @Mock
                private LoginView view;
                @Mock
                Private LoginService service;
                @Mock
                private LoginPresenter presenter;
   @Before
                Public void setUp() throws Exception{
                 presenter = new Mainactivity(view, service);

   }
                //if User enter empty student number
                @Test
                public void shouldShowErrorMessageWhenUsernameIsEmpty() throws Exception{
                  when(view.getUsername()).thenReturn("");
                  presenter.onLoginClicked();
                  verify(view).showUsernameError(R.string.username_error);

                }
                //if the studentnumber and password not entered and empty
                @Test
                public void shouldShowErrorMessageWhenUsernameIsEmpty() throws Exception{
                  when(view.getUsername()).thenReturn("");
                  when(view.getPassword()).thenReturn("");
                  presenter.onLoginClicked();
                  verify(view).showUsernameError(R.string.username_error);
                  verify(view).showPasswordError(R.string.password_error);

                }
                //if the student no is present and not entered password
                @Test
                public void shouldShowErrorMessageWhenPasswordIsEmpty() throws Exception{
                  when(view.getUsername()).thenReturn("12345678");
                  when(view.getPassword()).thenReturn("");
                  presenter.onLoginClicked();
                  verify(view).showPasswordError(R.string.password_error);

                }
                //if the student no is string this test case and password not enetered
                @Test
                public void shouldShowErrorMessageWhenPasswordIsEmpty() throws Exception{
                  when(view.getUsername()).thenReturn("abcd"));
```

```
        when(view.getPassword()).thenReturn(""));
        presenter.onLoginClicked();
        verify(view).showUsernameError(R.string.username_error);
        verify(view).showPasswordError(R.string.password_error);
    }
    //if the studentno is string this test case and password is entered wrong
    @Test
    public void shouldShowErrorMessageWhenPasswordIsEmpty() throws Exception{
        when(view.getUsername()).thenReturn("abcd"));
        when(view.getPassword()).thenReturn(""));
        presenter.onLoginClicked();
        verify(view).showUsernameError(R.string.username_error);
        verify(view).showPasswordError(R.string.password_error);

    }
    // both studentid and password correct enters registration
    @Test
    public void shouldStartMainActivityWhenUsernameAndPasswordAreCorrect()throws Exception{
        when(view.getUsername()).thenReturn("12345678"));
        when(view.getPassword()).thenReturn("aaaa"));
        when(service.login("12345678", "aaaa")).thenReturn(true);
        presenter.onLoginClicked();
        verify(view).startMainactivity();


    }
    //if studentid and password wrong fails login
    @Test
    public void shouldShowLoginErrorWhenUsernameAndPasswordAreInvalid() throws Exception{
        when(view.getUsername()).thenReturn("12345678"));
        when(view.getPassword()).thenReturn("aaaa"));
        when(service.login("12345678", "aaaa")).thenReturn(false);
        presenter.onLoginClicked();

        verify(view).showLoginError(R.string.login_failed);


    }
}
package com.example.test.uwa_parking.login;

public interface LoginiView{
   String getUsername();
   Void showUsernameError(int resId);
   String getPassword();
```

```
  void showPasswordError(int resId);
  void startMainActivity();
  Void showLoginError(int resId);
}
package com.example.test.uwa_parking.login;

public class LoginPresenter{
                private LoginView view;
                private LoginService service;

                public LoginPresnter(LoginView view, LoginService service){
                        this.view = view;
      this.service = service;
   }
 public void onLoginClicked(){
   String username = view.getUsername();
   if(username.isEmpty()){
    view.showUsernameError(R.string.username_error);
    return;
   }
   String password = view.getPassword();
   if (password.isEmpty()){
    view.showPasswordError(R.string.password_error);
                 return;
    }
                boolean loginSucceeded = service.login(username, password);
                if (loginSucceeded){
                  view.startMainActivity();
                  return;
 }
 view.showLoginError(R.string.login_failed);
}
}
```