

Team Final Destination

18 May, 2016

Semaka Malapane - 13081129

Matthew Nel - 10126229

Pierre Ngameni -

Ruth O. Ojo - 12042804

Sfiso Shabangu - 12081622

Contents

1	Test Plan Identifier	2
2	References	2
3	Introduction	2
4	Test Items	2
5	Software Risk Issues	3
6	Features to be Tested	4
7	Features not to be Tested	4
8	Approach	4
8.1	Testing Level – Unit testing	4
8.2	Test Tools	4
8.3	Meetings	4
8.4	Measures and Metrics	4
8.5	Configuration Management	4
8.6	Special Requirements	4
9	Item Pass/Fail Criteria	5
10	Suspension Criteria and Resumption Requirements	5
11	Test Deliverables	5
12	Remaining Test Tasks	6
13	Environmental Needs	6
14	Staffing and Training Needs	6
15	Responsibilities	6
16	Schedule	6
17	Planning Risks and Contingencies	7
18	Approvals	7
19	Glossary	7

1 Test Plan Identifier

Final Destination: Testing for Alpha-Android

2 References

This document makes use of and refers to the Researcher Support System (RSS) Requirements and Design Specifications (version 0.1). This is the document that specifies what the entire system is supposed to do and specifies what each module is supposed to do.

3 Introduction

This is a Unit Test plan for the Research Support System (RSS) Android Application Framework (Android Module) and it will only address the RSS Android Module as implemented by team Alpha of the COS 301 mini project.

The RSS Android Module is being tested in isolation of all the other system modules, as such this test plan will cover one level of testing, namely Unit testing.

The purpose of this test plan is to document the testing requirements of the RSS Android Module as well as outline the testing procedure of the RSS Android Module. Only the functional and non-functional requirements will be considered. The basis of the requirements to be tested are in the RSS Master Requirements and Design Specifications (version 0.1).

The RSS Android module consists of the Android User Interface to the RSS, as such to test the functional requirements automated UI tests will be done. There are eleven non-functional requirements to be tested.

In addition to the above requirements, the Android Module will be assessed in line with the following criteria:

- Front-end process demo
- Usability
- Use of external mocks
- Gradle build with appropriate dependencies
- Use of Dependency Injection
- Use of Fragments
- Use of REST

Details on the testing of the aforementioned requirements and criteria will be given in the succeeding sections.

4 Test Items

The RSS Master Requirements and Design Specifications (version 0.1) did not give enough information on the expected functionality of the Android Module, as such assumptions will be made regarding the functional and non-functional requirements of the module.

Assumed test items for the unit testing of the Android Module will include:

1. Functional Requirements (UI Requirements)

(a) The Android module should adhere to the following requirements:

- The UI should be usable and intuitive
- The UI should be visually appealing and dynamic
- The user should also be allowed to view only those pages that are for their access level (for instance: **Researcher**, **ResearchGroupLeader**, **Administrator**).
- All relevant inputs from the UI should be persisted to the database, via interaction with a REST API.

- User input should be validate with respect to the required input
- (b) For each system module, the Android module UI should adhere to the following requirements:
- **Publications**
 - A logged in user should be allowed to add publications, modify as well as manage them from the UI. Also, all actions associated with **Publication Types** should also be accessible form the UI.
 - All actions from the UI should be propagated to the back-end. A link or button should be provided to allow a user to navigate to a space for the Publications module.
 - **Persons**
 - The UI should allow for a logged in user to modify their personal details as well as add people to the system.
 - A user should also be allowed to manage the members of a research group, whether it be adding or removing.
 - Activities associated with the addition, reactivation and suspension of a research group should also be accessible from the UI.
 - Lastly, a user should also be allowed to add and modify research categories from the UI.
 - **Notifications**
 - Certain functionality of the **Notifications** module should be accessible form the UI; thus a user should be able to add notifications as well as send notifications
 - **Reporting**
 - All visual report data, that has been generated, should be displayed on the UI.
 - Button/links to trigger the generation of reports should be accessible from the UI.
 - **Import/Export**
 - Buttons/links to trigger the importing and exporting of data pertaining to persons, researcher groups, publications should be accessible from the UI.
 - Buttons/links to trigger the exporting of a published paper, for a user group. to a bibtex file should also be accessible from the UI

It is assumed that the Web Application Framework (Web module) and Android module are meant to have similar functionality, as such the non-functional requirements given for the Web module are applied here.

2. Non-Functional Requirements

- (a) Usability
- (b) Maintainability
- (c) Flexibility
- (d) Performance

5 Software Risk Issues

One of the risks with this project is that the android module has too little documentation. This is a problem as it could lead to misunderstandings and confusion as the developers might not really know what is expected of them and could lead create something the client did not want. This confusion could have been cleared up by referring the developers to the web module documentation as that is somewhat similar to this module and could help give clarity to the developers on what is expected of them.

Another risk issue could be linked to the complexity of the android module. This module contains multiple interfaces and linking the interfaces to one another could be difficult and problematic for the developers.

6 Features to be Tested

7 Features not to be Tested

8 Approach

8.1 Testing Level – Unit testing

In the given code, no testing constructs were found, as such the Final Destination testing team will carry out the necessary tests. Unit tests will be automated by the use of established Android UI testing frameworks. At this level of testing, mock objects will be required to ensure that the testing process can be carried out.

8.2 Test Tools

A few tools will be used to test the android module, namely:

- Lint Lint is a tool that checks the quality of your code and find errors that were missed during coding. It is a good tool to use as it can be configured to test code at different levels, i.e. test the modules separately or the entire project, as the developers and testers see fit. The given alpha android code will be tested individually to see if each of the given files work separately and then the it will also be tested on entire project to see how the different files work together.
- junit JUnit belongs to the xUnit family of unit testing frameworks and it allows writing of repeatable tests for Java code.

8.3 Meetings

The team will meet at least once a week to discuss progress so far and to divide up the remaining work load.

8.4 Measures and Metrics

Metrics for the testing of non-functional requirements will be collected. These will be made available by the Lint Report tool.

8.5 Configuration Management

Only Unit testing will be performed thus only one machine is needed. All testing will be performed using Android Studio. There are no specified hardware requirements for this module so any electronic device that has Android Studio installed and can run Java code can be used to test this module. No modifications will be made to the given code except those that will help with testing so there is no need to create different branches on github so as not to hinder production. Thus, only master will be used to make things simpler for the testers.

8.6 Special Requirements

The Android module will be assessed according to the given evaluation criteria.

I Front-end process demo

The UI of the module should be representative of an Android application UI and should conform to the set standards of an Android UI.

II Usability

With each click of a link or a button a user should be given appropriate feedback, in that a result of that click should be displayed or an appropriate error message should be thrown. The feedback should be timely as well.

III Use of external mock objects

As a result of a modular system design, the Android module should be able to fulfil all requirements in isolation. As such the use of mock objects is suited as a substitute for the real objects provided by the other system modules. REST API mock objects are needed for this module.

The implementation will be checked for the use of a Mock REST API.

IV Gradle build with appropriate dependencies

Dependency management of the module should be handled with Gradle to allow for seamless project build. All Android dependencies should be kept in the Gradle configuration.

V Use of Dependency Injection

As previously mentioned, the specification for the RSS requires a modular design, ensuring for less inter-module dependencies. Each isolated module is meant to use Mock objects in place of concrete objects from the respective system modules. These mock objects are meant to be injected into the Android module.

VI Use of Fragments

The

VII Use of REST

The Android module should interact with a REST API in order to receive the necessary objects to be displayed on the UI. REST http requests should be made depending on the required data and a back-end server should respond with the appropriate data as long as no error occurs.

9 Item Pass/Fail Criteria

10 Suspension Criteria and Resumption Requirements

11 Test Deliverables

This section consists in enumerating with a brief description all what we have to deliver after testing the Android module which involves the USER INTERFACE OF THE RESEARCH SUPPORT SYSTEM (RSS). The deliverables of this test will be as follow:

- The Test plan document which is the document containing all the aspects of the testing; it describes the different activities that was done during the testing as well as the scope of testing , and the approach we had to take in testing the UI of the android module of the RSS.
- The Test cases: It is based on the functional and non-functional requirements. The functional requirement consists of using mock object of Junit to apply the unit testing; and Android lint report is used for the quality requirement (non-functional requirement).
- Tools and their output: Two tools are used in the testing process. Robolectric Android which is a framework for unit testing that uses the Android SDK jar in order to test-drive the android module and Android Lint Report for non-functional requirement.
- Errors logs and executions logs: are some screenshots of files that records some errors that occurs during the testing.
- Problem reports and corrective action: a corrective action was applied in the gradle file (file that manages dependencies in android studio) when opening the source code.

NB: the only thing that will not be part of the test deliverables is the Android module.

12 Remaining Test Tasks

the module we were given is not a multi-phase process and won't be released in an incremental way hence does not require additional attention in the sense that there will not be any future functionality that will be added on the android module we have been given to test. If it was the case we would have answer affirmatively that there will be some remaining test but in our case there will not be any remaining test that will be carry on the android module because in our context this module is a stand alone application.

13 Environmental Needs

This section of environmental needs explains and give a description of the type some needs we will consider

In this section we look at some points that can appear to be environmental needs during the testing process and answer some of these environmental questions such as:

- Will the system (the android module) be restricted of use during the testing? since the android module(user interface) we are testing his a stand alone module and not integrated to the entire system. it is not therefore an issue to verify wether or not the system should be restricted of use. but in the meanwhile for testing purpose the android module should be live runing.
- As an environmental needs we needed to run the android code of the user interface on the latest version of Android Studio (Android Studio 2.1.1).
- We do not required any special power, but just the normal electric power that could make a laptop to be on.
- As another environmental need we had to look on how our the test data will be provide. in the testing process we will be using ock object to test against the functional requirement and we will also use input from the keyboard.
- In this project we are not using simulators(special hardware), either static generator
- The component of the android module are tested thoroughly as a whole.

14 Staffing and Training Needs

The android module is to be tested. Since most of our team members (3 out of 5) have never used programmed an android module, they would need to be trained and taught how to use android studio which would take up our already limited time. Due to the aforementioned reason, it was decided that it would be better if the members that are already proficient programmers in android should focus on testing the module and the other members focus on writing the test plan and assisting with test reports and non-functional requirements.

15 Responsibilities

- Team Alpha of the COS 301 mini project are responsible for development of the Android module
- The Final Destination team of COS 730 are responsible for the documenting and execution of the unit tests of the code provided by team Alpha

e

16 Schedule

13 April: Have a look at the assignment specification and the code to be tested

14 - 18 April: Look up non-functional requirements testing tools and test plans

19 April: Meet up to discuss progress and findings. Divide the work according to our strengths and experience

20 April: Try to get the code running and see what the code does
21 April - 5 May: Assess the given code against requirement specification. Start writing the test plan
6 May: Meet up to discuss progress
7-14 May: Continue writing test plan
13-15 May: Test the functional requirements and non-functional requirements
16-18 May: Finish writing test plan and report on results
18 May: Submit report

17 Planning Risks and Contingencies

Nowadays risks planification and contingencies are part of the planning of any project. In this particular project we had to consider some aspects which appear as risks such as:

- The availability of the human resources, software, hardware and tools that we needed before carry on with the testing. We arranged all of these before the 10 May 2016 .
- In order to avoid late delivery, progress meeting was scheduled to know the level of each member and the difficulties he/she may have in the process.
- Avoiding delay due to training on a specific technology or tools: we made that each member should be assigned a task according to his level of knowledge and training and prerequisites.
- We do not have a case of contingency due to a change of requirements.

18 Approvals

This section states the parties who are responsible to approve the different processes as fully done and give the go ahead for the continuation of the project. Parties involved in the approval are considered, depending on the magnitude of the project . In our case, we are applying the white box method, using mock Object of the JUnit framework; for testing the user interface module. Being a unit test it is different from other type of project such as integration; Therefore we only needed a group of software testers directed by a senior Software tester in charge of the whole design and planning of the testing project and responsible of giving the approvals on each stage of the project.

19 Glossary

- Android Application Framework – Android Module
- RSS – Research Support System
- UI – User Interface
- Web Application Framework – Web Module

20 Figures


































 Compiled Doc RuthOjo committed 3 minutes ago	 5ef15f5	
 Merge branch 'master' of https://github.com/MattNel/TheFinalDestination RuthOjo committed 3 minutes ago	 a83e444	
 Ruth: Completed Sections RuthOjo committed 7 minutes ago	 483389f	
 added configuration management semakamalapanne committed an hour ago	 fc4d46e	
 added configuration management semakamalapanne committed an hour ago	 8bb0517	
 Section raw data <small>xxx</small> u12081622 committed 3 hours ago	 0280dbd	
 added test tools semakamalapanne committed 5 hours ago	 2c388ce	
 added meetings semakamalapanne committed 5 hours ago	 bdf1845	
 added risk issues semakamalapanne committed 6 hours ago	 b2a2980	
 added student number semakamalapanne committed 7 hours ago	 e29f50f	
 added student number semakamalapanne committed 7 hours ago	 96f204d	

Figure 1: A diagram showing the two methods the authors considered for usage of the results of the dithering process [?]