Team Final Destination

18 May, 2016

Semaka Malapane -
Matthew Nel -
Pierre Ngameni -
Ruth O. Ojo - 12042804
Sfiso Shabangu -

Contents

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 requirements of the module.

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

1. Functional Requirements
   (a) The Android UI should allow for access to all system modules that require user interaction.
   (b) Each link and button should work
   (c) All input boxes should allowed for input
   (d) User input should be validate with respect to the required input
   (e) The Android module should interact with a REST API to retrieve all necessary content that should be displayed on the UI.
   (f) Usability

5 Software Risk Issues

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

8.3 Meetings

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

8.6 Special Requirements

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

1. Front-end process demo

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

2. 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.

3. 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.

4. 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.

5. Use of Dependency Injection

   As previously mentioned, the specification for the RSS requiress 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.

6. Use of Fragments

   The

7. 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

12 Remaining Test Tasks

13 Environmental Needs

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

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

18 Approvals

19 Glossary

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