120337H - Linganesan

Dopa
Test Plan

Version <1.0>

Revision History

Date	Version	Description	Author
23/8/2015	1.0	Test plan description	Linganesan

Table of Contents

Test Plan

1. Evaluation Mission and Test Motivation

Android applications run on a variety of devices and depend on various factors. Hence it is very important to debug, test and optimize our Android application. Having a reasonable test coverage for an Android application helps developer to enhance and maintain the Android application. Logging and debugging in android is done with the tool Android Device Bridge (ADB).

ADB is one of the useful tools that come with the Android SDK. The system level logs and user defined logs are shown when 'adb logcat' command is executed while the application is running in emulator or in the device. ADB tool also provides the shell access to the android system of the device connected to system via USB cable. This tool provides access to SQLite database and the tables created and can be used to display the entries in the table.

2. Target Test Items

- Locus list generation
- New Discipline creation
- Run list generation
- Match proper discipline with locus

3. Test Approach

Testing Techniques and Types

3.1.1 Unit Testing

In Unit testing each independent unit is tested separately, by isolating it from the remainder of the code to ensure parts of the code are working properly. Unit is the smallest testable part of the code, as in here the classes are treated as the base unit.

Technique Objective:	A unit test is used to verify a minimal unit of source code, such as a method or a class. The purpose of unit testing is to isolate the smallest testable parts and verify that they function correctly in isolation.
Technique:	• Initialize the environment before each test runs by setUp() • Test the pre-conditions for all other tests by testPreconditions() • The appropriate error or warning messages are displayed when invalid data is used • unit test perform against my application's UI by testText()
Oracles:	The ADT plugin then launches the test application and the application under test on a target emulator or device.
Required Tools:	JUnit provides several assertion-based testing, such as assertNull(), assertEquals(), assertTrue(), and assertSame(). If any of these returns false, then the test case is considered as failure.
Success Criteria:	Written in the same language (java) and integrated with existing tools and IDE. Must provide a fast feedback on the development cycle.
Special Considerations:	Requires a separate project for tests. Tests and code are not kept together.

3.1.2 User Interface Testing

User Interface (UI) testing verifies a user's interaction with the software. The goal of UI testing is to ensure that the UI provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Technique Objective:	UI testing ensures that the application returns the correct UI output in response to a sequence of user actions, such as entering keyboard input or pressing toolbars, menus, dialogs, images, and other UI controls.
Technique:	Use the ViewAssertions methods to check that the UI reflects the expected state or behavior, after these user interactions are performed.
Oracles:	The core API is small, predictable, and easy to learn.Espresso tests state expectations, interactions, and assertions clearly without the distraction, custom infrastructure, or messy implementation details getting in the way
Required Tools:	Espresso, Install the Android Testing Support Library. The Espresso API is located under the com.android.support.test.espresso package.
Success Criteria:	Easy to integrate with Gradle
Special Considerations:	Turn off animations on your test device. Leaving system animations turned on in the test device might cause unexpected results or may lead your test to fail. Specify the Android testing dependencies in gradle.build

3.1.3 Compatibility Testing

Variations in software versions, configurations, display resolutions, servers and Internet connect speeds can heavily impact the application behavior. Different specifications of devices can also make the applications to behave differently.

Technique Objective:	Make the application compatible to different versions of android.
Technique:	People use different android devices and hence a good application must be 100% reliable and give best visualization effects irrespective of the device specifications. The application does not require internet hence the speed of internet is not relevant or necessary scenario in this case. To check the device compatibility, the application is tested in the both Android tablet and smart phone. The application requires Android SDK 5.0or higher versions.
Oracles:	
Required Tools:	Compatibility Test Suite(CTS)
Success Criteria:	The technique supports testing: • Robustness tests test the durability of the system under stress. • Performance tests test the performance of the system against defined benchmarks, for example rendering frames per second.
Special Considerations:	• using Android platform from git repository • using unix-like host system • operating in in directory with android

4. Deliverables

Test Deliverables are documents that are given to the stakeholders when the software is being developed

4.1 Test Evaluation Summaries

I also requested my friends to test the application in their mobiles and got their feedback as part of user testing. Some of the feedback they gave was:

- Few persons complains about the navigation of the application
- Application was easy to understand

4.2 Reporting on Test Coverage

Reporting test execution results is very important part of testing, tester should make a complete test results report which includes the Test Pass/Fail status of the test cycle. Test summary report is a document which contains summary of test activities and final test results. After the testing cycle it is very important that developer communicate the test results and findings to the project stakeholders so that decisions can be made for the software release.

5. Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)
Prerequisite entry criteria is not met.	<Tester> will define the prerequisites that must be met before Load Testing can start. <User> will endeavour to meet prerequisites indicated by <Tester>.	• Meet outstanding prerequisites • Consider Load Test Failure
Test data proves to be inadequate.	<User> will ensure a full set of suitable and protected test data is available. <Tester> will indicate what is required and will verify the suitability of test data.	• Redefine test data • Review Test Plan and modify • components (that is, scripts) • Consider Load Test Failure
Database requires refresh.	<System Admin> will endeavour to ensure the Database is regularly refreshed as required by <Tester>.	• Restore data and restart • Clear Database