Team Final Destination

18 May, 2016

Semaka Malapane - 13081129
Matthew Nel - 10126229
Pierre Ngameni - 15197876
Ruth O. Ojo - 12042804
Sfiso Shabangu - 12081622

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 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

Front-end processes demo Priority : high:

Usability

Test application access channel to other services, this involve testing is user is able to login and use other services (Login) Priority : high: This servers as a bridge between user and the service provided by the application, thus is important for the feature to work at all time. Test the application to respond to user interaction and response when input is imputed in the required field Priority High: This feature is the core for other features as input to the feature is further processed to other applications Test system persistence based on user input, user inputs publication information, filling up required fields (Add publication) Priority High: This feature inputs private data from the individual, data captured from this model needs to be secured and private to other research groups

Search query response based on view publication module, testing where the search query outputs the desired resource (View Publication) Priority High: Feature involve retrieval of sensitive data entered by the user and need to be available ever time as user request, user can request to edit the data in the module Editing of user profile, is the edit persisted and change in appropriate places in the application Priority Medium: Feature involves modification of user information with little impact to the overall system, thus not significant damage is done when the feature service cant be reached

Use of external mocks Priority : high: There were no use of external mocks

Gradle build with appropriate dependencies Priority : high: Dependencies in gradle build was partially implemented where a connection to a server was attempted but then it was not used anywhere in the code so was not successful.

Dependency Injection Priority : high: There was no use of dependency injection

Use of fragments Priority : high: There was no use of fragments so the advantage of a more modular system fell away. The code was implemented in such a way that a new activity was just created and extended from the main activity. The code was hard coded in making maintenance on the code harder.

Use of REST Priority : high: The use of REST was partially implemented and not done properly or was not completed. JSON object were created to store the user data but then it is stored locally in a file rather than using android features such as volley request or retro fit where you send the JSON objects to the back-end.

7 Features not to be Tested

The System is divided in to modules, which consist of service contract, in the Android module of the system, the module is required to mimic the entire modules that make up the system, and the service contract of each module in the system will not 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 of 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 file.

V 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.

VI Use of Fragments

The use of fragments allows for the containment of activities within the Main Android activity. As such, when a new interface for a system module is entered, no new activity is created, instead a reference to an activity is created. This reduces code complexity.

VII Use of REST

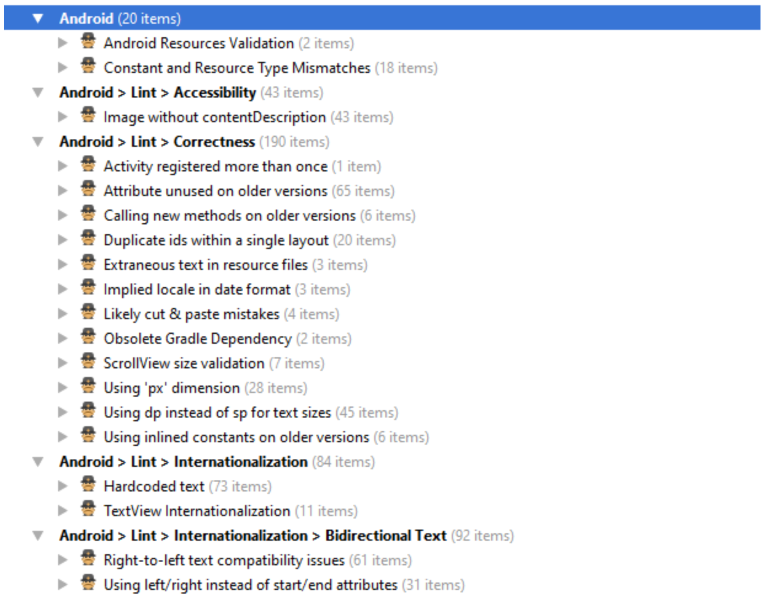
The Android module should interact with a REST API in order to receive the necessary objects to be displayed on the UI. REST http requests should be made depending on the required data and a back-end server should respond with the appropriate data, as long as no system error occurs.

9 Item Pass/Fail Criteria

Unit test of core modules making the application

Lint was used to carry out a few tests and the following results appeared.

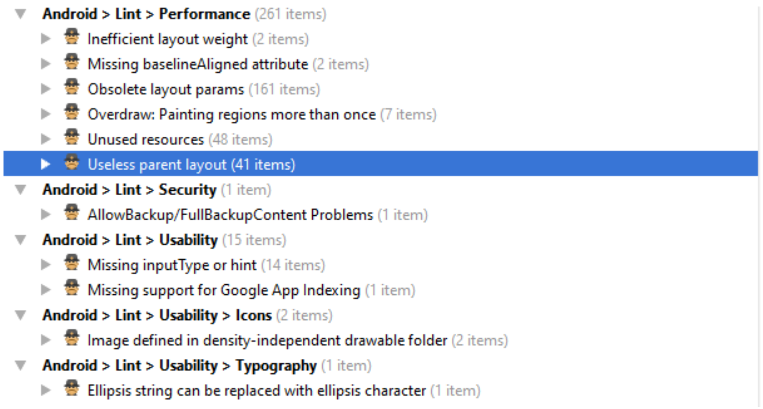
In the following image you can see the test results for the features such as accessibility, correctness internationalization and bidirectional text



```
▼ Android (20 items)
    ▶ Android Resources Validation (2 items)
    ▶ Constant and Resource Type Mismatches (18 items)
▼ Android > Lint > Accessibility (43 items)
    ▶ Image without contentDescription (43 items)
▼ Android > Lint > Correctness (190 items)
    ▶ Activity registered more than once (1 item)
    ▶ Attribute unused on older versions (65 items)
    ▶ Calling new methods on older versions (6 items)
    ▶ Duplicate ids within a single layout (20 items)
    ▶ Extraneous text in resource files (3 items)
    ▶ Implied locale in date format (3 items)
    ▶ Likely cut & paste mistakes (4 items)
    ▶ Obsolete Gradle Dependency (2 items)
    ▶ ScrollView size validation (7 items)
    ▶ Using 'px' dimension (28 items)
    ▶ Using dp instead of sp for text sizes (45 items)
    ▶ Using inlined constants on older versions (6 items)
▼ Android > Lint > Internationalization (84 items)
    ▶ Hardcoded text (73 items)
    ▶ TextView Internationalization (11 items)
▼ Android > Lint > Internationalization > Bidirectional Text (92 items)
    ▶ Right-to-left text compatibility issues (61 items)
    ▶ Using left/right instead of start/end attributes (31 items)
```

Figure 1: git commits

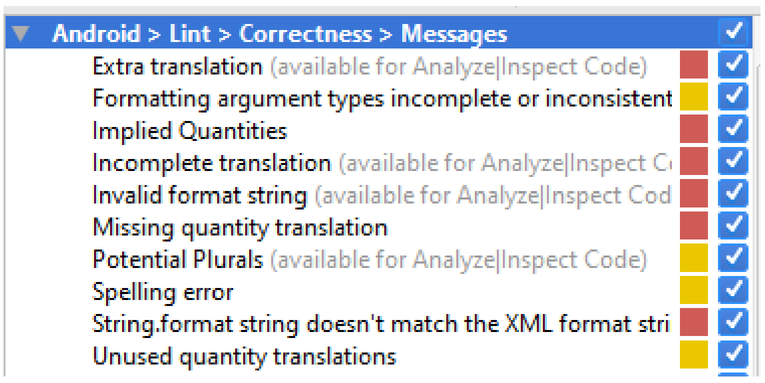
In this lint report the test for performance, security, usability, icons and typography.



```
▼ Android > Lint > Performance (261 items)
    ▶ Inefficient layout weight (2 items)
    ▶ Missing baselineAligned attribute (2 items)
    ▶ Obsolete layout params (161 items)
    ▶ Overdraw: Painting regions more than once (7 items)
    ▶ Unused resources (48 items)
    ▶ Useless parent layout (41 items)
▼ Android > Lint > Security (1 item)
    ▶ AllowBackup/FullBackupContent Problems (1 item)
▼ Android > Lint > Usability (15 items)
    ▶ Missing inputType or hint (14 items)
    ▶ Missing support for Google App Indexing (1 item)
▼ Android > Lint > Usability > Icons (2 items)
    ▶ Image defined in density-independent drawable folder (2 items)
▼ Android > Lint > Usability > Typography (1 item)
    ▶ Ellipsis string can be replaced with ellipsis character (1 item)
```

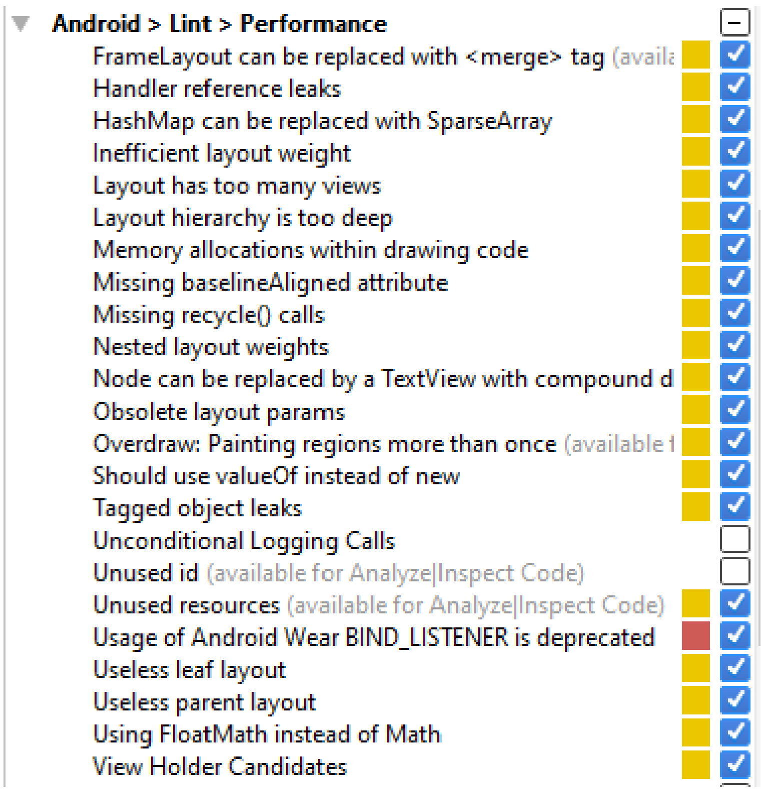
Figure 2: git commits

We set up lint with to check for the following criteria.



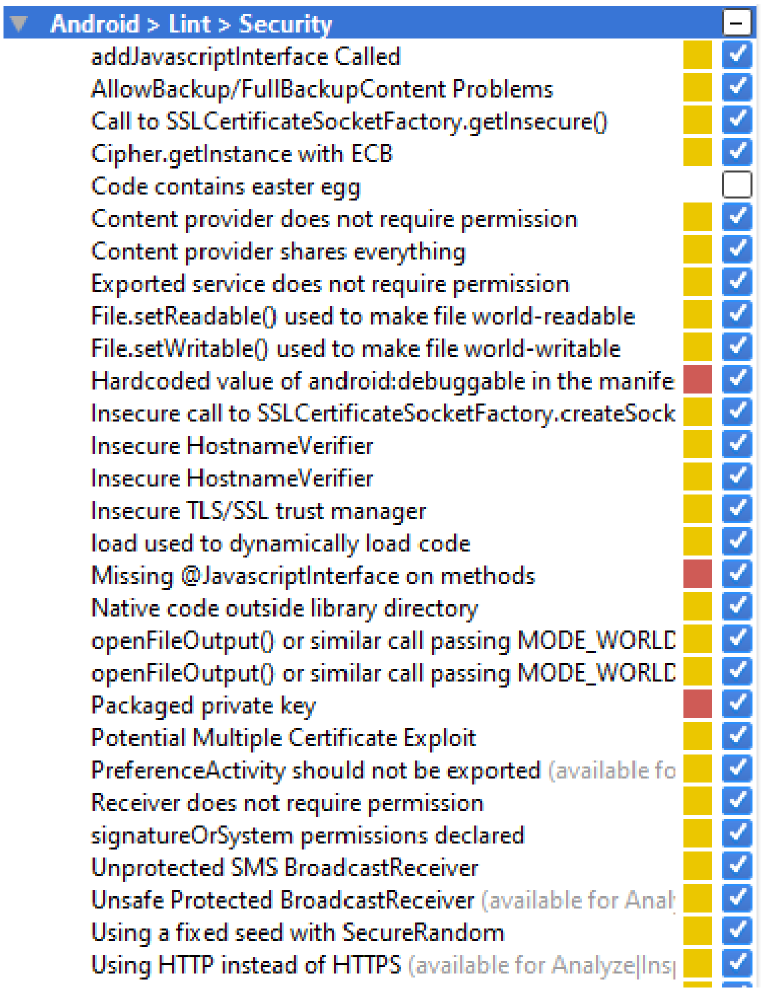
Android > Lint > Correctness > Messages		✓
Extra translation (available for Analyze|Inspect Code)	🟥	✓
Formatting argument types incomplete or inconsistent	🟨	✓
Implied Quantities	🟥	✓
Incomplete translation (available for Analyze|Inspect C	🟥	✓
Invalid format string (available for Analyze|Inspect Cod	🟥	✓
Missing quantity translation	🟥	✓
Potential Plurals (available for Analyze|Inspect Code)	🟨	✓
Spelling error	🟨	✓
String.format string doesn't match the XML format stri	🟥	✓
Unused quantity translations	🟨	✓

Figure 3: git commits



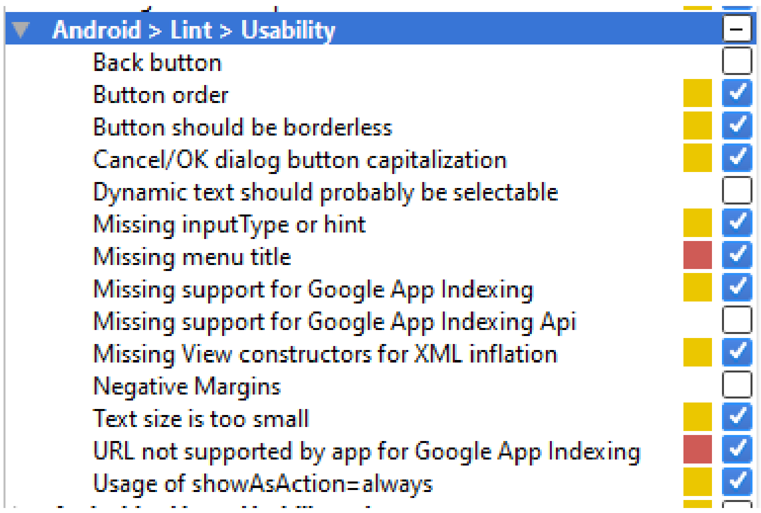
Android > Lint > Performance		−
FrameLayout can be replaced with <merge> tag (availa	🟨	✓
Handler reference leaks	🟨	✓
HashMap can be replaced with SparseArray	🟨	✓
Inefficient layout weight	🟨	✓
Layout has too many views	🟨	✓
Layout hierarchy is too deep	🟨	✓
Memory allocations within drawing code	🟨	✓
Missing baselineAligned attribute	🟨	✓
Missing recycle() calls	🟨	✓
Nested layout weights	🟨	✓
Node can be replaced by a TextView with compound d	🟨	✓
Obsolete layout params	🟨	✓
Overdraw: Painting regions more than once (available f	🟨	✓
Should use valueOf instead of new	🟨	✓
Tagged object leaks	🟨	✓
Unconditional Logging Calls		☐
Unused id (available for Analyze|Inspect Code)		☐
Unused resources (available for Analyze|Inspect Code)	🟨	✓
Usage of Android Wear BIND_LISTENER is deprecated	🟥	✓
Useless leaf layout	🟨	✓
Useless parent layout	🟨	✓
Using FloatMath instead of Math	🟨	✓
View Holder Candidates	🟨	✓

Figure 4: git commits



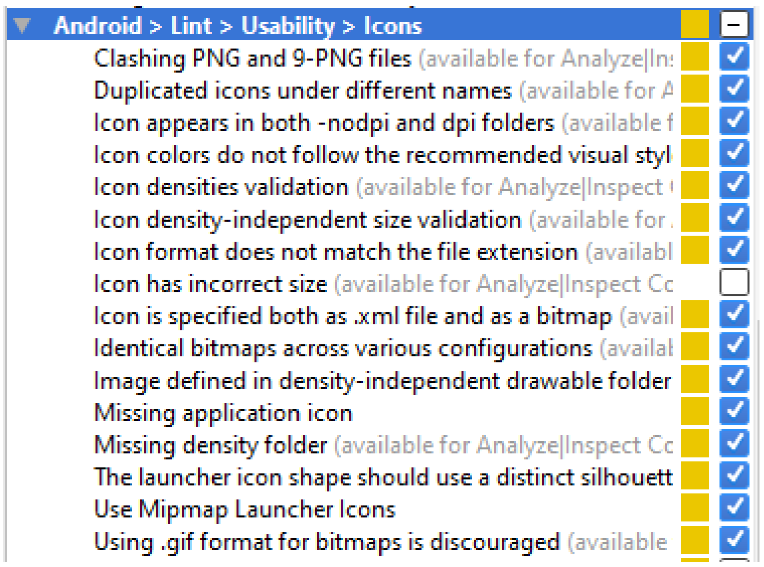
Android > Lint > Security		☐
addJavascriptInterface Called	🟨	✓
AllowBackup/FullBackupContent Problems	🟨	✓
Call to SSLCertificateSocketFactory.getInsecure()	🟨	✓
Cipher.getInstance with ECB	🟨	✓
Code contains easter egg		☐
Content provider does not require permission	🟨	✓
Content provider shares everything	🟨	✓
Exported service does not require permission	🟨	✓
File.setReadable() used to make file world-readable	🟨	✓
File.setWritable() used to make file world-writable	🟨	✓
Hardcoded value of android:debuggable in the manife	🟥	✓
Insecure call to SSLCertificateSocketFactory.createSock	🟨	✓
Insecure HostnameVerifier	🟨	✓
Insecure HostnameVerifier	🟨	✓
Insecure TLS/SSL trust manager	🟨	✓
load used to dynamically load code	🟨	✓
Missing @JavascriptInterface on methods	🟥	✓
Native code outside library directory	🟨	✓
openFileOutput() or similar call passing MODE_WORLD	🟨	✓
openFileOutput() or similar call passing MODE_WORLD	🟨	✓
Packaged private key	🟥	✓
Potential Multiple Certificate Exploit	🟨	✓
PreferenceActivity should not be exported (available fo	🟨	✓
Receiver does not require permission	🟨	✓
signatureOrSystem permissions declared	🟨	✓
Unprotected SMS BroadcastReceiver	🟨	✓
Unsafe Protected BroadcastReceiver (available for Anal	🟨	✓
Using a fixed seed with SecureRandom	🟨	✓
Using HTTP instead of HTTPS (available for Analyze|Ins	🟨	✓

Figure 5: git commits



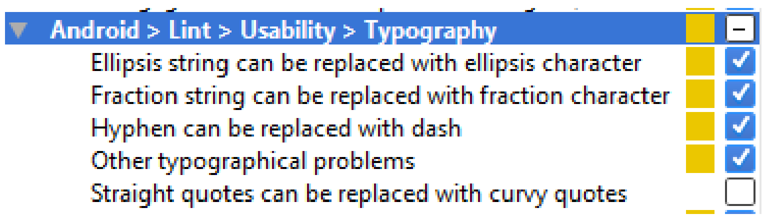
Android > Lint > Usability		☐
Back button		☐
Button order	🟨	✓
Button should be borderless	🟨	✓
Cancel/OK dialog button capitalization	🟨	✓
Dynamic text should probably be selectable		☐
Missing inputType or hint	🟨	✓
Missing menu title	🟥	✓
Missing support for Google App Indexing	🟨	✓
Missing support for Google App Indexing Api		☐
Missing View constructors for XML inflation	🟨	✓
Negative Margins		☐
Text size is too small	🟨	✓
URL not supported by app for Google App Indexing	🟥	✓
Usage of showAsAction=always	🟨	✓

Figure 6: git commits



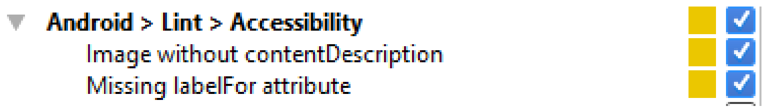
Android > Lint > Usability > Icons
- ✓ Clashing PNG and 9-PNG files (available for Analyze|In:
- ✓ Duplicated icons under different names (available for A
- ✓ Icon appears in both -nodpi and dpi folders (available f
- ✓ Icon colors do not follow the recommended visual styl
- ✓ Icon densities validation (available for Analyze|Inspect (
- ✓ Icon density-independent size validation (available for .
- ✓ Icon format does not match the file extension (availabl
- ☐ Icon has incorrect size (available for Analyze|Inspect Cc
- ✓ Icon is specified both as .xml file and as a bitmap (avail
- ✓ Identical bitmaps across various configurations (availal
- ✓ Image defined in density-independent drawable folder
- ✓ Missing application icon
- ✓ Missing density folder (available for Analyze|Inspect Cc
- ✓ The launcher icon shape should use a distinct silhouett
- ✓ Use Mipmap Launcher Icons
- ✓ Using .gif format for bitmaps is discouraged (available

Figure 7: git commits



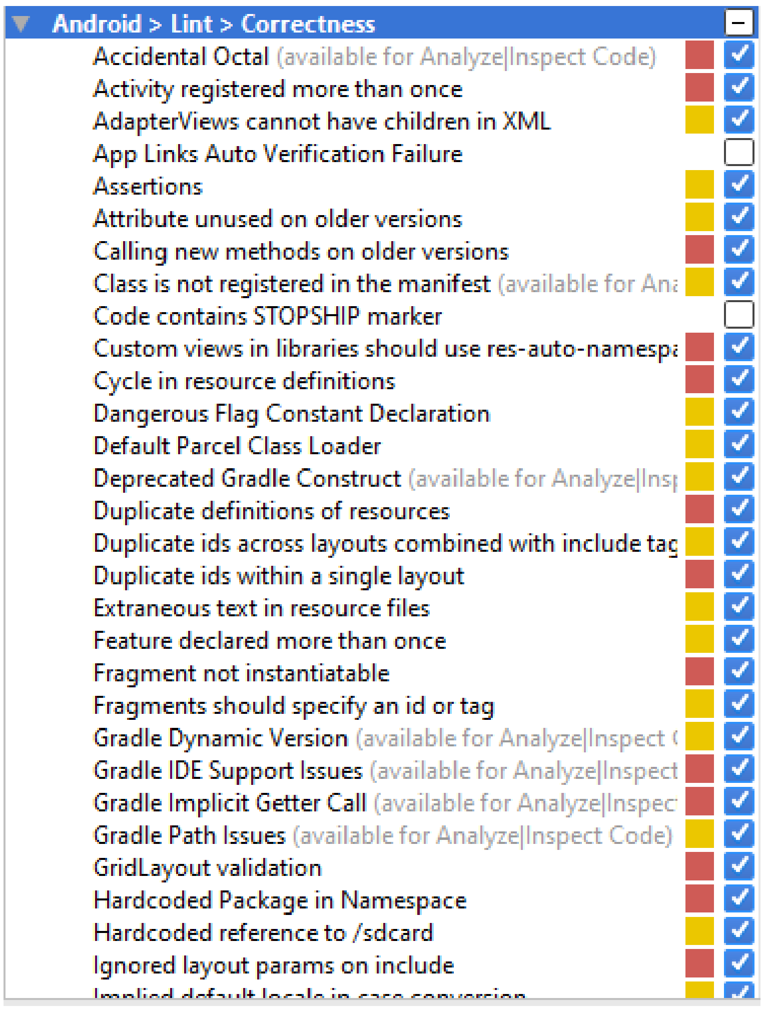
Android > Lint > Usability > Typography
- ✓ Ellipsis string can be replaced with ellipsis character
- ✓ Fraction string can be replaced with fraction character
- ✓ Hyphen can be replaced with dash
- ✓ Other typographical problems
- ☐ Straight quotes can be replaced with curvy quotes

Figure 8: git commits



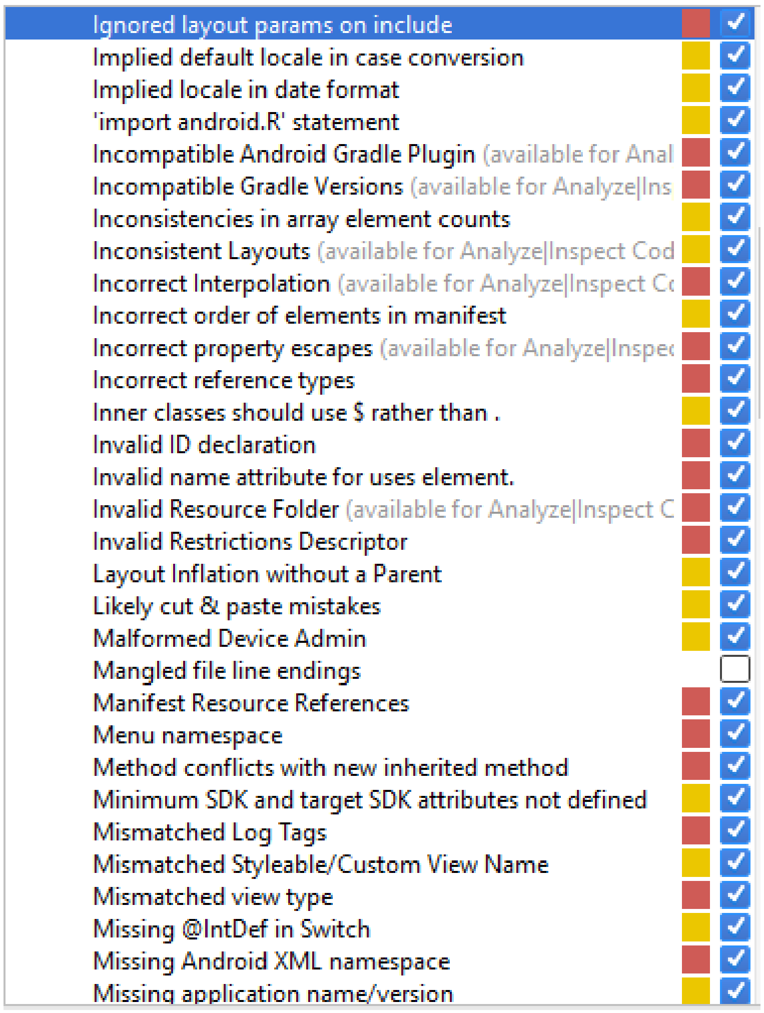
✓ Android > Lint > Accessibility
- ✓ Image without contentDescription
- ✓ Missing labelFor attribute

Figure 9: git commits



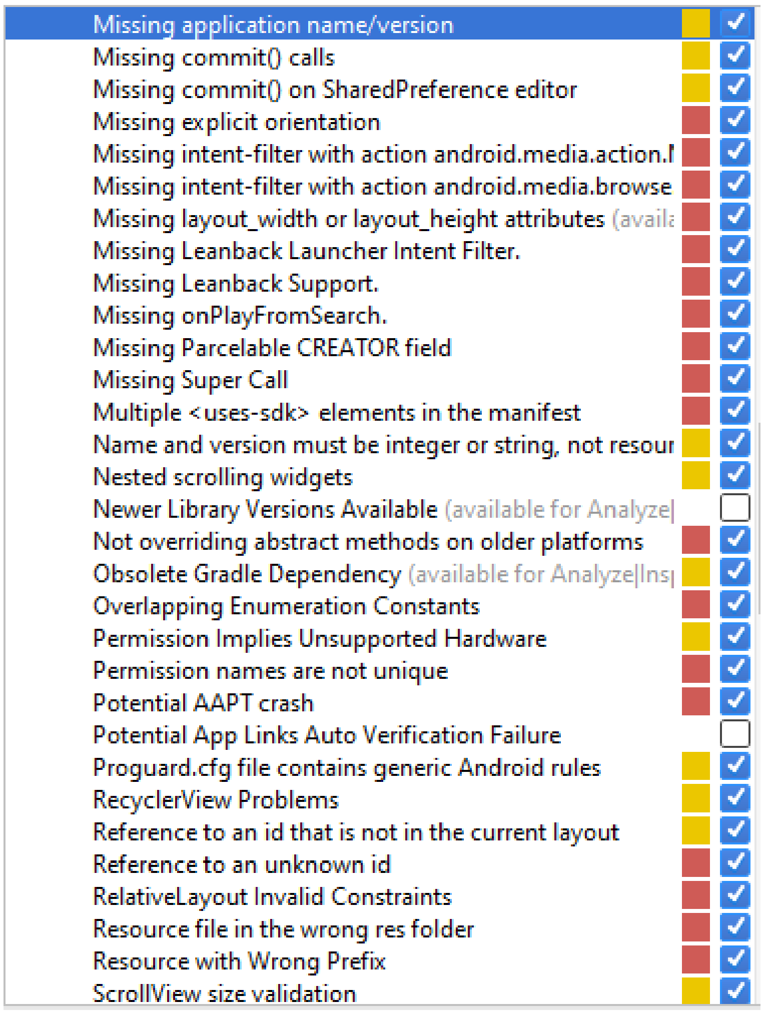
Android > Lint > Correctness		☐
Accidental Octal (available for Analyze|Inspect Code)	🟥	✓
Activity registered more than once	🟥	✓
AdapterViews cannot have children in XML	🟨	✓
App Links Auto Verification Failure		☐
Assertions	🟨	✓
Attribute unused on older versions	🟨	✓
Calling new methods on older versions	🟥	✓
Class is not registered in the manifest (available for Ana	🟨	✓
Code contains STOPSHIP marker		☐
Custom views in libraries should use res-auto-namespa	🟥	✓
Cycle in resource definitions	🟥	✓
Dangerous Flag Constant Declaration	🟨	✓
Default Parcel Class Loader	🟨	✓
Deprecated Gradle Construct (available for Analyze|Insp	🟨	✓
Duplicate definitions of resources	🟥	✓
Duplicate ids across layouts combined with include tag	🟨	✓
Duplicate ids within a single layout	🟥	✓
Extraneous text in resource files	🟨	✓
Feature declared more than once	🟨	✓
Fragment not instantiatable	🟥	✓
Fragments should specify an id or tag	🟨	✓
Gradle Dynamic Version (available for Analyze|Inspect (	🟨	✓
Gradle IDE Support Issues (available for Analyze|Inspect	🟥	✓
Gradle Implicit Getter Call (available for Analyze|Inspec	🟥	✓
Gradle Path Issues (available for Analyze|Inspect Code)	🟨	✓
GridLayout validation	🟥	✓
Hardcoded Package in Namespace	🟥	✓
Hardcoded reference to /sdcard	🟨	✓
Ignored layout params on include	🟥	✓
Implied default locale in case conversion	🟨	✓

Figure 10: git commits



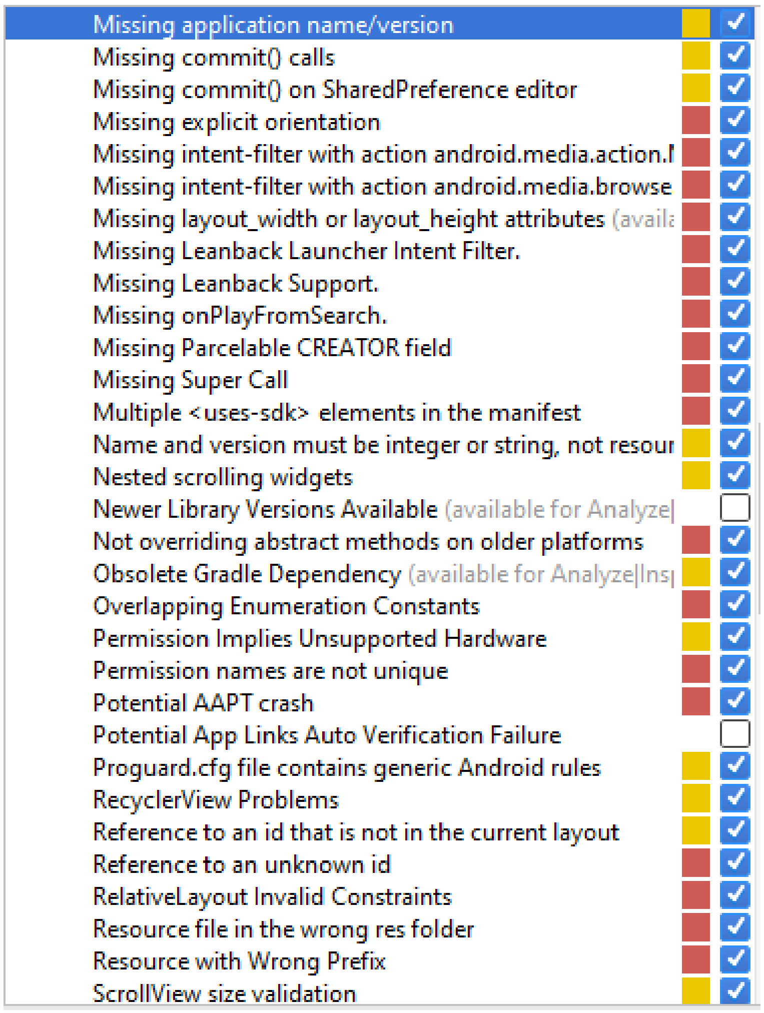
Ignored layout params on include	🟥	✓
Implied default locale in case conversion	🟨	✓
Implied locale in date format	🟨	✓
'import android.R' statement	🟨	✓
Incompatible Android Gradle Plugin (available for Anal	🟥	✓
Incompatible Gradle Versions (available for Analyze|Ins	🟥	✓
Inconsistencies in array element counts	🟨	✓
Inconsistent Layouts (available for Analyze|Inspect Cod	🟨	✓
Incorrect Interpolation (available for Analyze|Inspect Co	🟥	✓
Incorrect order of elements in manifest	🟨	✓
Incorrect property escapes (available for Analyze|Inspec	🟥	✓
Incorrect reference types	🟥	✓
Inner classes should use $ rather than .	🟨	✓
Invalid ID declaration	🟥	✓
Invalid name attribute for uses element.	🟥	✓
Invalid Resource Folder (available for Analyze|Inspect C	🟥	✓
Invalid Restrictions Descriptor	🟥	✓
Layout Inflation without a Parent	🟨	✓
Likely cut & paste mistakes	🟨	✓
Malformed Device Admin	🟨	✓
Mangled file line endings		☐
Manifest Resource References	🟥	✓
Menu namespace	🟥	✓
Method conflicts with new inherited method	🟥	✓
Minimum SDK and target SDK attributes not defined	🟨	✓
Mismatched Log Tags	🟥	✓
Mismatched Styleable/Custom View Name	🟨	✓
Mismatched view type	🟥	✓
Missing @IntDef in Switch	🟨	✓
Missing Android XML namespace	🟥	✓
Missing application name/version	🟨	✓

Figure 11: git commits



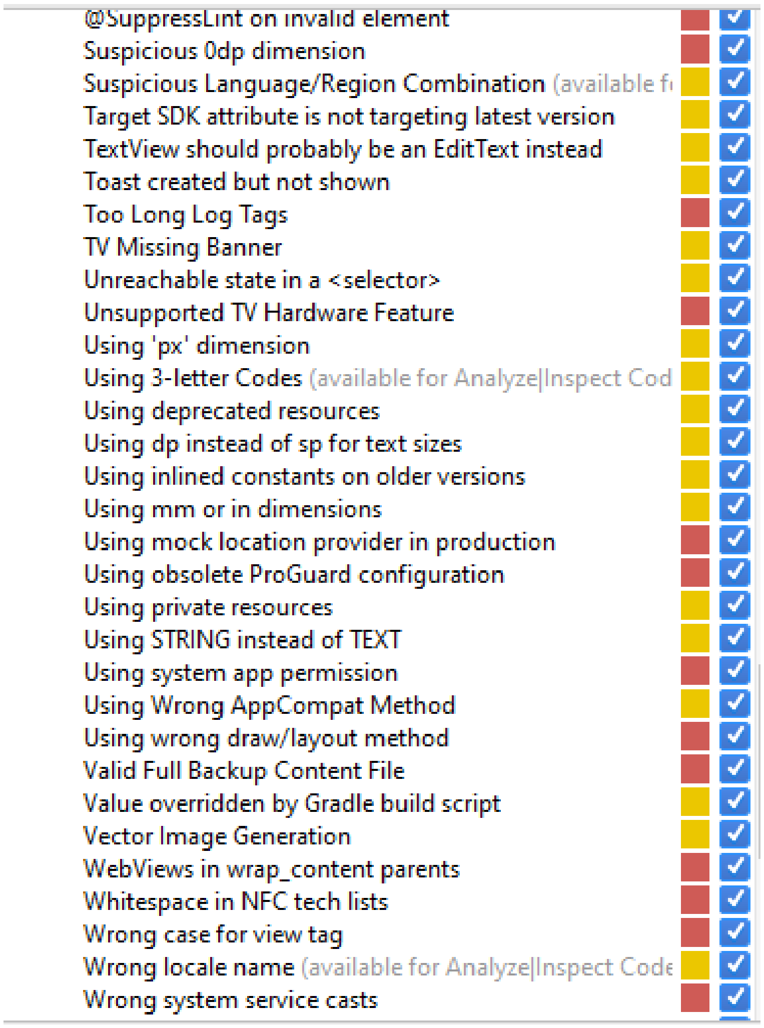
Missing application name/version	🟨	✓
Missing commit() calls	🟨	✓
Missing commit() on SharedPreference editor	🟨	✓
Missing explicit orientation	🟥	✓
Missing intent-filter with action android.media.action.	🟥	✓
Missing intent-filter with action android.media.browse	🟥	✓
Missing layout_width or layout_height attributes (avail	🟥	✓
Missing Leanback Launcher Intent Filter.	🟥	✓
Missing Leanback Support.	🟥	✓
Missing onPlayFromSearch.	🟥	✓
Missing Parcelable CREATOR field	🟥	✓
Missing Super Call	🟥	✓
Multiple <uses-sdk> elements in the manifest	🟥	✓
Name and version must be integer or string, not resour	🟨	✓
Nested scrolling widgets	🟨	✓
Newer Library Versions Available (available for Analyze|		☐
Not overriding abstract methods on older platforms	🟥	✓
Obsolete Gradle Dependency (available for Analyze|Ins	🟨	✓
Overlapping Enumeration Constants	🟥	✓
Permission Implies Unsupported Hardware	🟨	✓
Permission names are not unique	🟥	✓
Potential AAPT crash	🟥	✓
Potential App Links Auto Verification Failure		☐
Proguard.cfg file contains generic Android rules	🟨	✓
RecyclerView Problems	🟨	✓
Reference to an id that is not in the current layout	🟨	✓
Reference to an unknown id	🟥	✓
RelativeLayout Invalid Constraints	🟥	✓
Resource file in the wrong res folder	🟥	✓
Resource with Wrong Prefix	🟥	✓
ScrollView size validation	🟨	✓

Figure 12: git commits



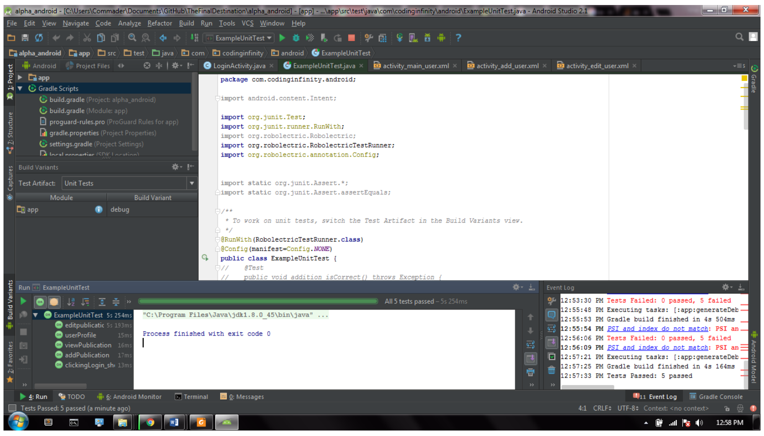
Missing application name/version	🟨	✓
Missing commit() calls	🟨	✓
Missing commit() on SharedPreference editor	🟨	✓
Missing explicit orientation	🟥	✓
Missing intent-filter with action android.media.action.I	🟥	✓
Missing intent-filter with action android.media.browse	🟥	✓
Missing layout_width or layout_height attributes (availa	🟥	✓
Missing Leanback Launcher Intent Filter.	🟥	✓
Missing Leanback Support.	🟥	✓
Missing onPlayFromSearch.	🟥	✓
Missing Parcelable CREATOR field	🟥	✓
Missing Super Call	🟥	✓
Multiple <uses-sdk> elements in the manifest	🟥	✓
Name and version must be integer or string, not resour	🟨	✓
Nested scrolling widgets	🟨	✓
Newer Library Versions Available (available for Analyze|		☐
Not overriding abstract methods on older platforms	🟥	✓
Obsolete Gradle Dependency (available for Analyze|Ins	🟨	✓
Overlapping Enumeration Constants	🟥	✓
Permission Implies Unsupported Hardware	🟨	✓
Permission names are not unique	🟥	✓
Potential AAPT crash	🟥	✓
Potential App Links Auto Verification Failure		☐
Proguard.cfg file contains generic Android rules	🟨	✓
RecyclerView Problems	🟨	✓
Reference to an id that is not in the current layout	🟨	✓
Reference to an unknown id	🟥	✓
RelativeLayout Invalid Constraints	🟥	✓
Resource file in the wrong res folder	🟥	✓
Resource with Wrong Prefix	🟥	✓
ScrollView size validation	🟨	✓

Figure 13: git commits



@SuppressLint on invalid element	🟥	✓
Suspicious 0dp dimension	🟥	✓
Suspicious Language/Region Combination (available f	🟨	✓
Target SDK attribute is not targeting latest version	🟨	✓
TextView should probably be an EditText instead	🟨	✓
Toast created but not shown	🟨	✓
Too Long Log Tags	🟥	✓
TV Missing Banner	🟨	✓
Unreachable state in a <selector>	🟨	✓
Unsupported TV Hardware Feature	🟥	✓
Using 'px' dimension	🟨	✓
Using 3-letter Codes (available for Analyze|Inspect Cod	🟨	✓
Using deprecated resources	🟨	✓
Using dp instead of sp for text sizes	🟨	✓
Using inlined constants on older versions	🟨	✓
Using mm or in dimensions	🟨	✓
Using mock location provider in production	🟥	✓
Using obsolete ProGuard configuration	🟥	✓
Using private resources	🟨	✓
Using STRING instead of TEXT	🟨	✓
Using system app permission	🟥	✓
Using Wrong AppCompat Method	🟨	✓
Using wrong draw/layout method	🟥	✓
Valid Full Backup Content File	🟥	✓
Value overridden by Gradle build script	🟨	✓
Vector Image Generation	🟨	✓
WebViews in wrap_content parents	🟥	✓
Whitespace in NFC tech lists	🟥	✓
Wrong case for view tag	🟥	✓
Wrong locale name (available for Analyze|Inspect Code	🟨	✓
Wrong system service casts	🟥	✓

Figure 14: git commits



```java
package com.codinginfinity.android;

import android.content.Intent;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;


import static org.junit.Assert.*;
import static org.junit.Assert.assertEquals;

/**
 * To work on unit tests, switch the Test Artifact in the Build Variants view.
 */
@RunWith(RobolectricTestRunner.class)
@Config(manifest=Config.NONE)
public class ExampleUnitTest {
//    @Test
//    public void addition_isCorrect() throws Exception {
```

```
All 5 tests passed – 5s 254ms
ExampleUnitTest 5s 254ms
  editpublicatic 5s 193ms
  userProfile 15ms
  viewPublication 16ms
  addPublication 17ms
  clickingLogin_sh 13ms

"C:\Program Files\Java\jdk1.8.0_45\bin\java" ...

Process finished with exit code 0
```

```
12:53:30 PM Tests Failed: 0 passed, 5 failed
12:55:48 PM Executing tasks: [:app:generateDeb
12:55:53 PM Gradle build finished in 4s 504ms
12:55:54 PM PSI and index do not match: PSI an
12:56:06 PM Tests Failed: 0 passed, 5 failed
12:56:09 PM PSI and index do not match: PSI an
12:57:21 PM Executing tasks: [:app:generateDeb
12:57:25 PM Gradle build finished in 4s 164ms
12:57:33 PM Tests Passed: 5 passed
```

Tests Passed: 5 passed (a minute ago)

Figure 15: git commits

Login: Test passed Add Publication: Test passed View Publication: Test passed Edit profile Test passed

10 Suspension Criteria and Resumption Requirements

Application testing reach a finite point, through lint report, the project included many errors and application running on the device kept on crushing because of these errors identified.

Implementation of the application is poorly implemented with no respect to android fragments usage, these resulted in many switches between different views in the application that puts a strain on the deployed device resources and replication, this resulted in some test not being able to be ran as switching between views lack reference path to which view to go to next, with the use of android fragments this could have being avoided.

Gradle build with appropriate dependencies. A attempted connection to be made to a server was created but attempts to test should be stopped as its not used in the code.

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 graddle file (file that manages dependencies in android studio) when opening the source code.

Note: 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 functionnality 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

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 unit tests of the code, as 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 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 planification 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

20.1 Git Commits

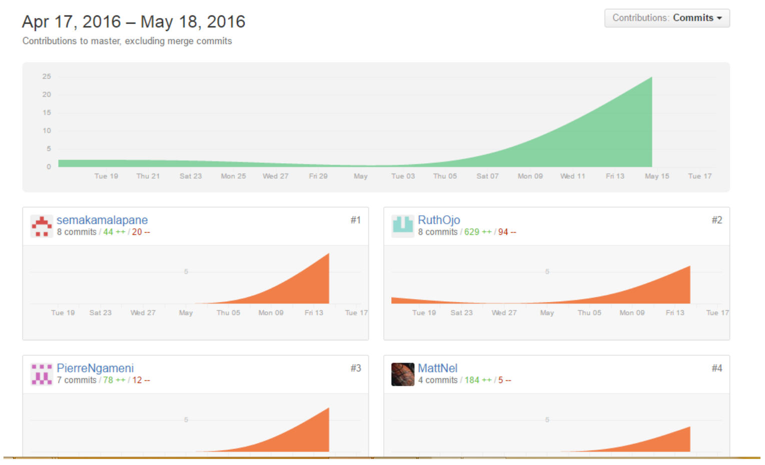
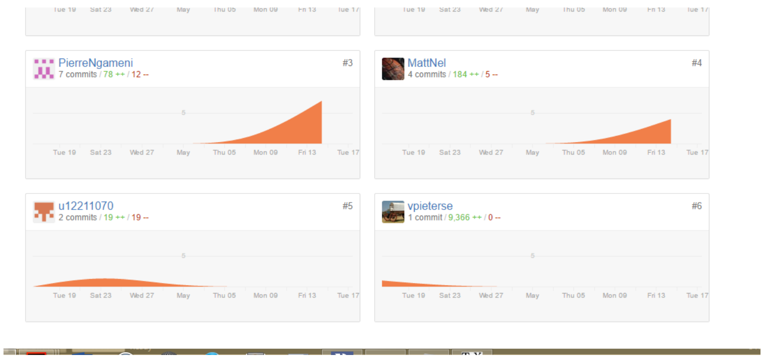

Apr 17, 2016 – May 18, 2016
Contributions to master, excluding merge commits
Contributions: Commits
25
20
15
10
5
0
Tue 19
Thu 21
Sat 23
Mon 25
Wed 27
Fri 29
May
Tue 03
Thu 05
Sat 07
Mon 09
Wed 11
Fri 13
May 15
Tue 17
semakamalapane
8 commits / 44 ++ / 20 --
#1
RuthOjo
8 commits / 629 ++ / 94 --
#2
PierreNgameni
7 commits / 78 ++ / 12 --
#3
MattNel
4 commits / 184 ++ / 5 --
#4

Figure 16: git commits



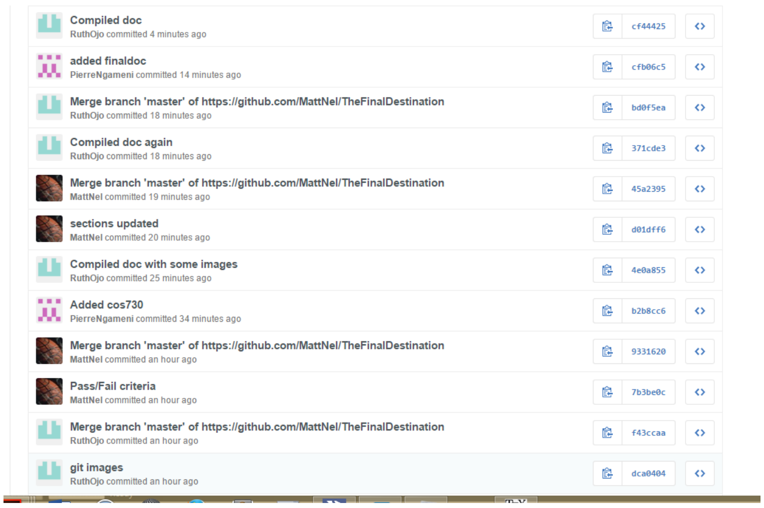
PierreNgameni #3
7 commits / 78 ++ / 12 --

MattNel #4
4 commits / 184 ++ / 5 --

u12211070 #5
2 commits / 19 ++ / 19 --

vpieterse #6
1 commit / 9,366 ++ / 0 --

Figure 17:



Compiled doc
RuthOjo committed 4 minutes ago
cf44425

added finaldoc
PierreNgameni committed 14 minutes ago
cfb06c5

Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
RuthOjo committed 18 minutes ago
bd0f5ea

Compiled doc again
RuthOjo committed 18 minutes ago
371cde3

Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
MattNel committed 19 minutes ago
45a2395

sections updated
MattNel committed 20 minutes ago
d01dff6

Compiled doc with some images
RuthOjo committed 25 minutes ago
4e0a855

Added cos730
PierreNgameni committed 34 minutes ago
b2b8cc6

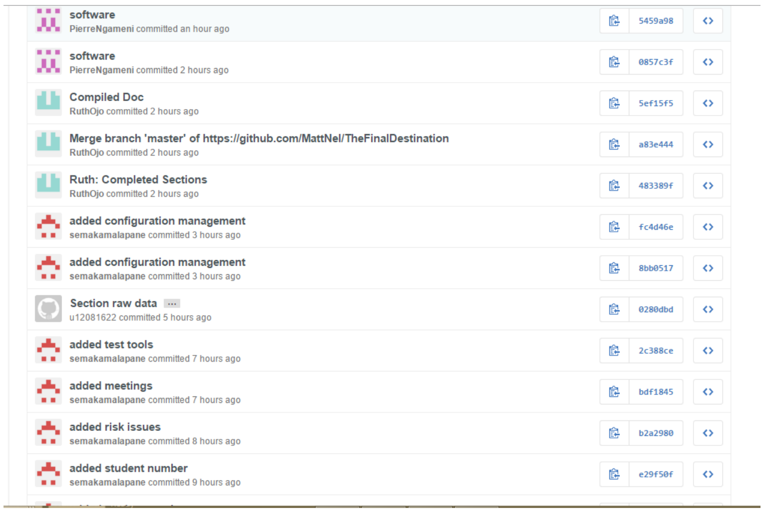
Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
MattNel committed an hour ago
9331620

Pass/Fail criteria
MattNel committed an hour ago
7b3be0c

Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
RuthOjo committed an hour ago
f43ccaa

git images
RuthOjo committed an hour ago
dca0404

Figure 18:



software
PierreNgameni committed an hour ago
5459a98

software
PierreNgameni committed 2 hours ago
0857c3f

Compiled Doc
RuthOjo committed 2 hours ago
5ef15f5

Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
RuthOjo committed 2 hours ago
a83e444

Ruth: Completed Sections
RuthOjo committed 2 hours ago
483389f

added configuration management
semakamalapane committed 3 hours ago
fc4d46e

added configuration management
semakamalapane committed 3 hours ago
8bb0517

Section raw data …
u12081622 committed 5 hours ago
0280dbd

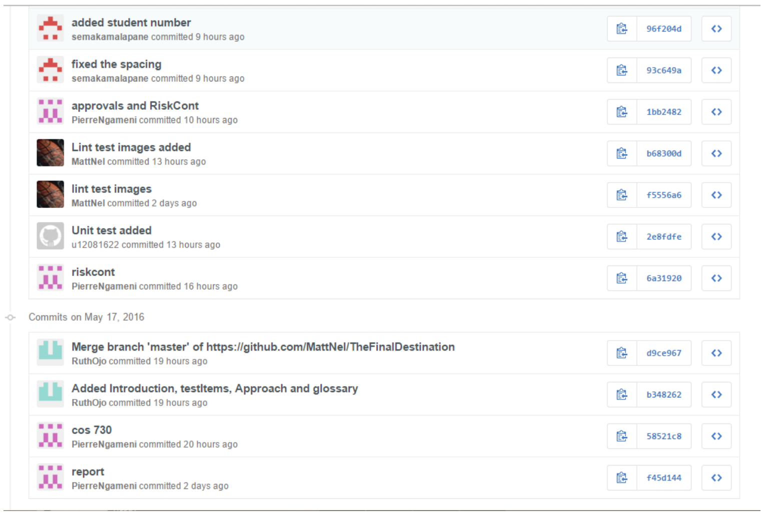
added test tools
semakamalapane committed 7 hours ago
2c388ce

added meetings
semakamalapane committed 7 hours ago
bdf1845

added risk issues
semakamalapane committed 8 hours ago
b2a2980

added student number
semakamalapane committed 9 hours ago
e29f50f

Figure 19:



added student number
semakamalapane committed 9 hours ago 96f204d

fixed the spacing
semakamalapane committed 9 hours ago 93c649a

approvals and RiskCont
PierreNgameni committed 10 hours ago 1bb2482

Lint test images added
MattNel committed 13 hours ago b68300d

lint test images
MattNel committed 2 days ago f5556a6

Unit test added
u12081622 committed 13 hours ago 2e8fdfe

riskcont
PierreNgameni committed 16 hours ago 6a31920

Commits on May 17, 2016

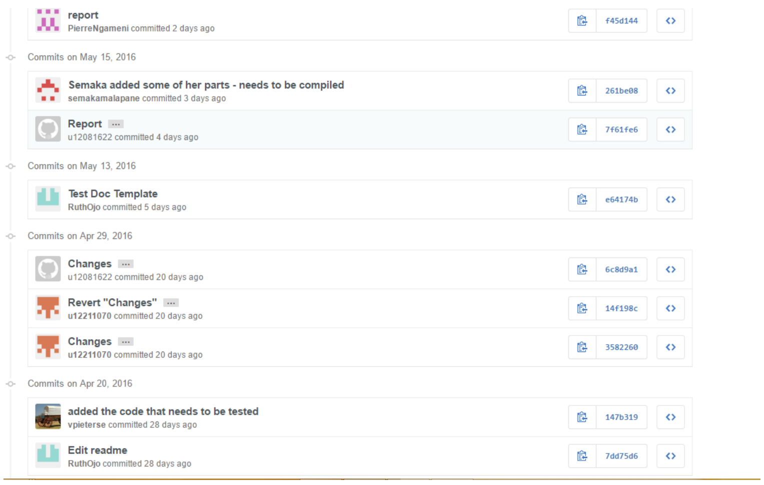
Merge branch 'master' of https://github.com/MattNel/TheFinalDestination
RuthOjo committed 19 hours ago d9ce967

Added Introduction, testItems, Approach and glossary
RuthOjo committed 19 hours ago b348262

cos 730
PierreNgameni committed 20 hours ago 58521c8

report
PierreNgameni committed 2 days ago f45d144

Figure 20:



report
PierreNgameni committed 2 days ago f45d144

Commits on May 15, 2016

Semaka added some of her parts - needs to be compiled
semakamalapane committed 3 days ago 261be08

Report …
u12081622 committed 4 days ago 7f61fe6

Commits on May 13, 2016

Test Doc Template
RuthOjo committed 5 days ago e64174b

Commits on Apr 29, 2016

Changes …
u12081622 committed 20 days ago 6c8d9a1

Revert "Changes" …
u12211070 committed 20 days ago 14f198c

Changes …
u12211070 committed 20 days ago 3582260

Commits on Apr 20, 2016

added the code that needs to be tested
vpieterse committed 28 days ago 147b319

Edit readme
RuthOjo committed 28 days ago 7dd75d6

Figure 21: