**CA326 Year 3 Project**

**Testing Documentation**

**Dylan Sloyan - 16511419**

**Samuel Wood - 16476066**

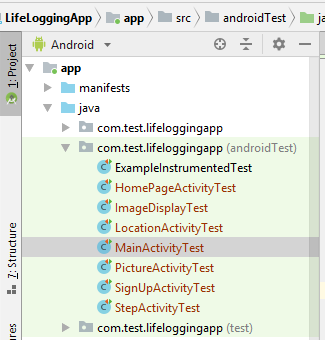
**Life Logging Application**

1. **Functional Testing**

1.1 **Activity Test Rules**

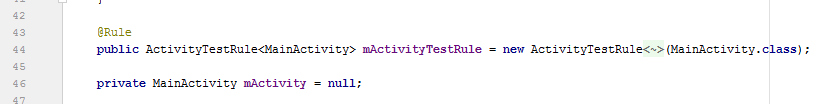
For each of the seven activities in our android app we implemented Activity Test Rules. In other words, rules that provide functional testing of a single activity. The testing library ‘JUnit 4’ and the ‘androidTest’ package were used to conduct these tests. To test every activity we had to create a new test class for each one. See the test classes in the below diagram.

*Fig 1.1 Activity test classes*



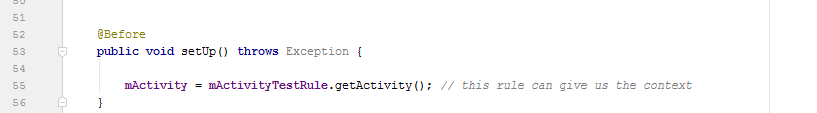
Each test class required an activity test rule and a ‘Contain activity instance’, this is displayed at line 46 of the below diagram. The instance is set to null as we haven’t yet acquired context for the activity.

*Fig 1.2 Activity test rule and contain activity instance for main activity.*



The ‘Before()’ method sets the preconditions required for our test. Once the activity is launched, this ‘setUp()’ method will provide us with context.

*Fig 1.3*



To test each activity a different attribute from each one was placed in its test code, for example a button or a textview. Now that the apps page is launched and context is acquired, the ‘testMALaunch()’ method will check if the Edit Text box for the users email is present in the Main Activity. The objective of the tear down method is to set the post conditions or ‘clean up’ process for the ‘testMALaunch()’ method. After the attributes presence has been tested, the ‘tearDown()’ method nullifies the contain activity instance as the apps activity is no longer launched.

*Fig 1.4 Launch test and ‘tearDown()’ method for Main Activity test rule.*

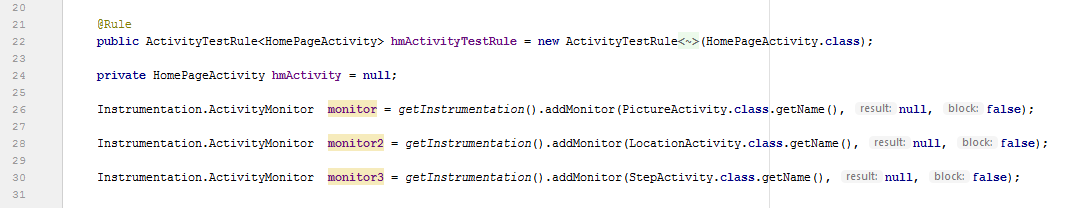


This test format was carried out to test each Activity for the Life Logging application. All of which were passed.

1.2 **Tests between Activities**

After implementing our Activity test rules we began to test certain scenarios regarding navigating through the app. The Home page activity was the most relevant in this regard as the Camera, Step counter and Location activity are all accessed from this activity. To test these scenarios, context of our activity has to be acquired. Once we have the activities context we can click the button for the next activity once its presence has been verified. This was done in the same way as the unit tests in Section 1.1. The click is executed with the help of the Espresso Framework. On the click the new activity should be launched. This is examined using Activity Monitors. To check that all activities can be properly opened from the home page, a monitor was created for the three activities in the Home Page activity test class.

*Fig 1.6 Monitors for each activity from the Home page.*



See Fig 1.7 for the ‘waitForMonitorWithTimeout’ method. That will wait for that monitors class to be activated until the timeout expires. In he timeout is measured in milliseconds.

The assert is then used to to check if the activity is ‘Not Null’ or not activated.

*Fig 1.7 The test method for the launch of Picture activity from the Home Page*

****

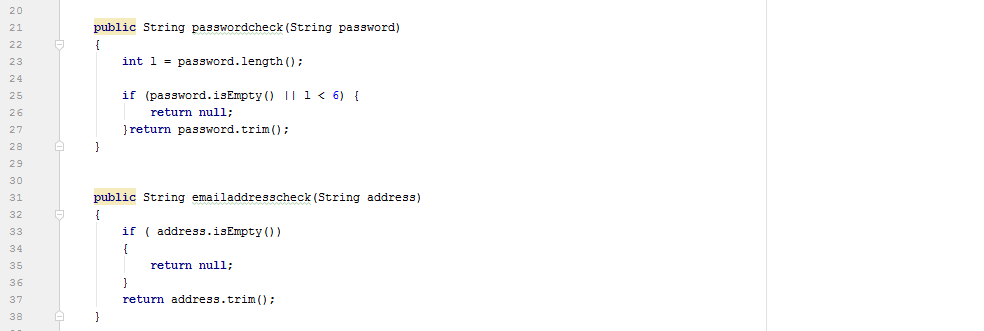
The following tests were carried out and passed in the same way as the above format to be sure the user could navigate through the app;

* Home Page activity to Picture activity
* Home Page activity to Location activity
* Home page activity to Step Activity
* Picture Activity to Image Display
* Main activity to Sign up
* Sign up to Main activity

1.3 **Testing our Authentication**

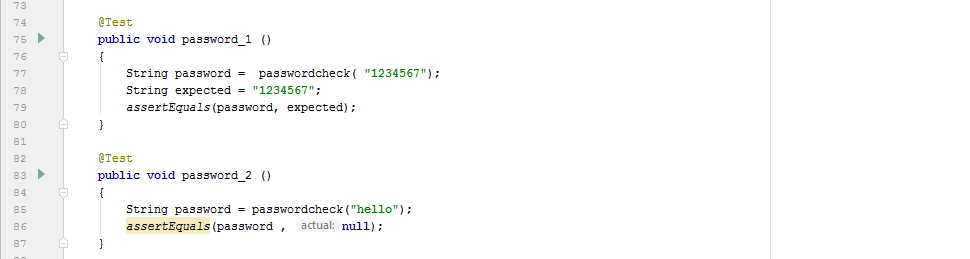
There are two pieces of authentication to gain access to our app, Email address and password. These entries are required at the user Log in page and the Sign up page. The app will prompt the user to re enter a valid email address should the initial email be invalid or if the text box is empty. In the Test classes for the two activities we implemented the methods in the below diagram as a means of testing validation, ie. if the entered password is less than 6 character long the password would fail or if there was not one entered the test would fail.

*Fig 1.8 password check and email check methods.*



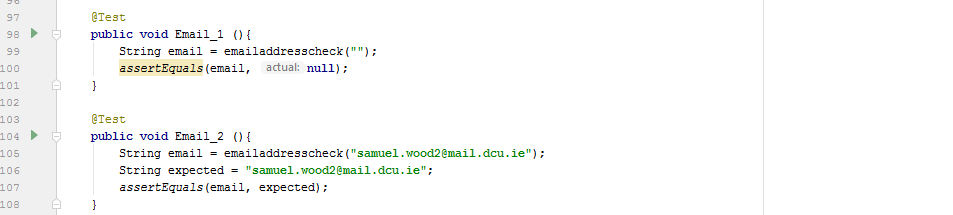
See Fig 1.9 and 1.10 for examples of the tests we ran. In ‘password\_1’ there is a password entered and it’s length is greater than 6 characters. Hence the expected string is the same as the password string and this password is valid. In ‘password\_2’ there is a password entered but its length is not greater than 6 characters. Therefore the password returns ‘null’ and the password is invalid.

*Fig 1.9 Password unit tests.*



In our ‘emailaddresscheck’ method, if the entered Email address edit text is empty, the function will return null. This is shown in the ‘Email\_1’method in Fig 1.10. This test resembles a scenario where the user would attempt to log in without an address. This scenario is invalid so the test returns null. Email\_2 resembles a real email address where the ‘email’ String is the same as the ‘expected’ String and the address is accepted.

*Fig 1.10 Email address unit tests.*



2. **User Testing**

2.1 **List of Tasks to be Completed by the User**

In the computing labs, 5 participants agreed to meet us to carry out our user testing. We selectively picked participants of different ages and mixed computing skills, in the effort to get the widest range of feedback available. Each participant was given a sheet with a list of tasks to be completed within the application.

The list of tasks ensured each participant used and understood all features of the application. During the process, participant observation was carried out, identifying and taking note of what features enhanced/tainted the overall user experience. Any relevant questions was also adhered to.

2.2 **Conducting Focus Group**

Once all participants had completed the list of tasks a focus group was then able to be carried out. The focus group consisted of 6 questions and lasted approximately for 10 minutes.

We were overall pleased with the focus group as participants answered each question honestly and to their full potential. For example, question 3 asked what could have been done better. The two main responses were if the location of captured images could be displayed on the map and if images could be loaded and displayed faster.

Some critical feedback we were already aware of and some were small issues we had overlooked. If we had more time available we would have been able to prioritise these issues and make the appropriate changes to improve the application.

2.3 **Conclusion**

Ultimately, we found user testing extremely beneficial for going further with development. At the end of the day it is the user that determines the success/failure of the application. It is safe to say being able to look at things from their perspective and taking note of their feedback was invaluable.

3. **Heuristic Analysis**

Our application is aimed at people of all ages and furthermore with specific disabilities, such as short-term memory loss or alzheimer's. To accommodate for this large audience the following techniques/approaches were strongly focused on throughout the user interface design process:

* Appropriate background colour allowing text to stand out.
* Use of icons and hints to be informative as possible.
* Buttons and card views are large and bright.
* Consistent page layout.
* Success/Error messages and validations where appropriate.
* Use of images to accompany text.
* Strategic and optimum use of space.

4. **Shneiderman’s Eight Golden Rules**

4.1 **Strive for consistency**

From the moment the user enters the app they will be able to familiarise themselves with the consistent layout throughout, providing a smooth and comfortable experience. The app keeps the user satisfied and on board with the effective and appropriate use of detail. Icons, hints and images can be easily identified within every page of the app making it next to impossible for the user to become confused/frustrated. Examples of consistency can be seen by the following screen captures: