Change History:

Date:	Version Update:	Member:	Description:
2013/08/22	1.0	Michelle Peens	Document created.
2013/09/01	1.1	Michelle Peens	Document layout updated.
2013/09/10	1.2	Michelle Peens	Database component unit testing added.
2013/09/13	1.3	Michelle Peens	Database unit testing updated.
2013/09/15	1.4	Janine Venter	Project Scope updated
2013/09/15	1.5	Janine Venter	System Description updated
2013/09/15	1.6	Janine Venter	Support for Latex and Mimetex updated
2013/09/15	1.7	Janine Venter	Messaging updated
2013/09/15	1.8	Janine Venter	Main Activity Unit Test
2013/09/15	1.9	Janine Venter	Message Adapter Unit Test
2013/09/15	2.0	Janine Venter	Open issues added
2013/09/15	2.1	Janine Venter	Glossary added
2013/09/15	2.2	Janine Venter	Login Test added
2013/10/11	2.3	Michelle Peens	System Description: Support for LaTeX and MimeTeX Libraries Updated
2013/10/11	2.4	Michelle Peens	System Description: Messaging Updated
2013/10/11	2.5	Michelle Peens	System Description: Login Updated
2013/10/11	2.6	Michelle Peens	System Description: Updated
2013/10/12	2.7	Janine Venter	System Description Updated
2013/10/12	2.8	Janine Venter	System Description: Support for LaTeX and MimeTeX Libraries Updated
2013/10/12	2.9	Janine Venter	System Description: Messaging Updated
2013/10/12	3.0	Janine Venter	System Description: Login Updated

Table of Contents:

1. Introduction

1.1 Purpose

The purpose of this document is to explain the scope and results of testing that is performed during the incremental development of the software solution for the Latex Chat Application for our client Mr Will van Heerder. It will serve as a method for evaluating the progress and correctness of our solution against the set out requirements. Test cases will both include the testing of individual components as they are developed as well as the the testing of the components as one functional unit, after each iteration of our software development process.

1.2 Document Conventions

- Document Formatting: LaTeX

1.3 Project Scope

The aim of the project is to develop an open source android XMPP chat client which supports the embedded LaTeX base equations which are rendered as images. LaTeX based equations will be rendered on the handset to produce mathematical equations. Our system will also provide the ability to view, edit and correct equations before sending. The application will provide a similar functionality to yaxim. Exchange of images and mathematical expressions will be possible through our software solution. The TeXchat application will have the ability to show a preview of the entered text send the equation as LaTeX code and then render it on the receiving end on the client handset.

1.4 References

- Mr. Will van Heerden.

2. System Description

The goal of our software application is to provide a chat service that will allow users to exchange normal text messages and send mathematical equations displayed in a rendered image format. The application is intended to provide a service to users that require the ability and support for a chat client that allows them to communicate more efficiently and effortlessly in a scientific, and mathematical context. It will provide a more usable mobile version of a Latex chat application.

Support for LaTeX and MimeTeX Libraries
The application makes use of a Latex based library (MimeTeX) for the the rendering of equations as images on the mobile device. For this reason we have implemented the support for the MimeTex library, through the use of the Android NDK (Native Development Kit), which allows us to embed the native C/C++ code of the MimeTeX library, in the source code. The NDK acts as a bridge between the C/C++ code and Java code. The library is compiled through Eclipse using a special makefile that sets the correct compiler flags to indicate that it is in math mode and not text mode. The compiler flags represent different capabilities of the MimeTeX library, in this case we set it to math mode to support the mathematical LaTeX equations.

Messaging
parindent 5mm The purpose of this application is for messaging to be possible between multiple clients on the server. The application now supports this feature as well as allowing a user to send both plain text messages and Latex based equations, which are rendered on the client side and displayed as an inline images.
The rendering of the Latex based equations on the client side is provided through the use of the Mimetex library. The messages sent between the various clients on the server is stored statically through making use of a client side SQLite database, which also provides the functionality for the retrieval of messages.
Unsent messages, such as messages sent while the user was offline, is stored on the server and as soon as the client is online these messages are fetched and displayed on the user's handset. The server uses a queue to store these messages.

Login
The application allows a user to log in using an appropriate username and password combination. A user is authenticated by logging into the server. If a username and password combination is invalid the server will automatically hinder the user from logging in. Our user authentication component is implemented by the functionality provided by the server. The user is informed by the application that his/her attempt to log in was unsuccessful.

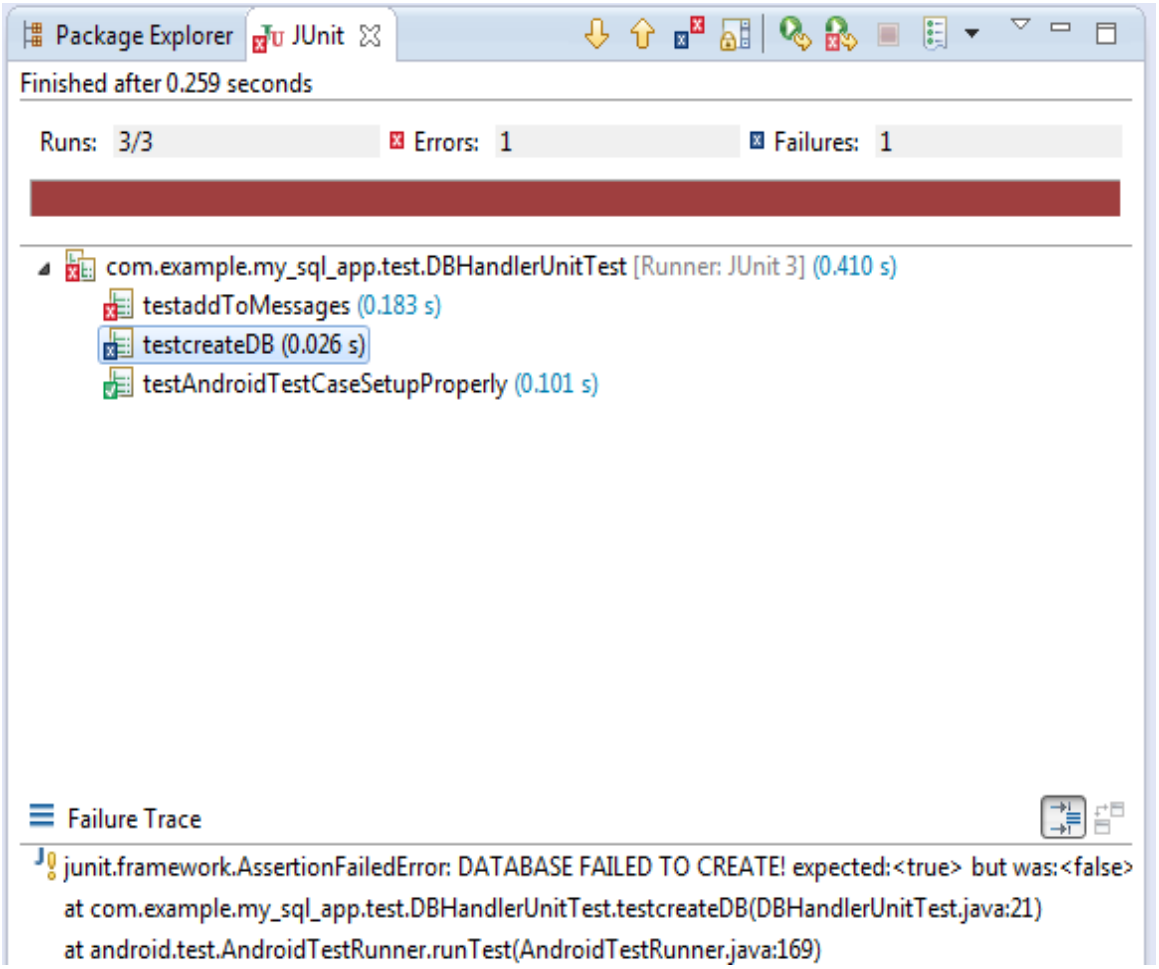
Register
The application allows someone to register to the TexChat service as a new user. Through the user registration form the user provides the necessary information and selects a username and password combination that is later used to authenticate the user on the server. After a user is successfully registered, he can immediately log in and start using the messaging functionality provided.

3. Project Testing

Unit Testing

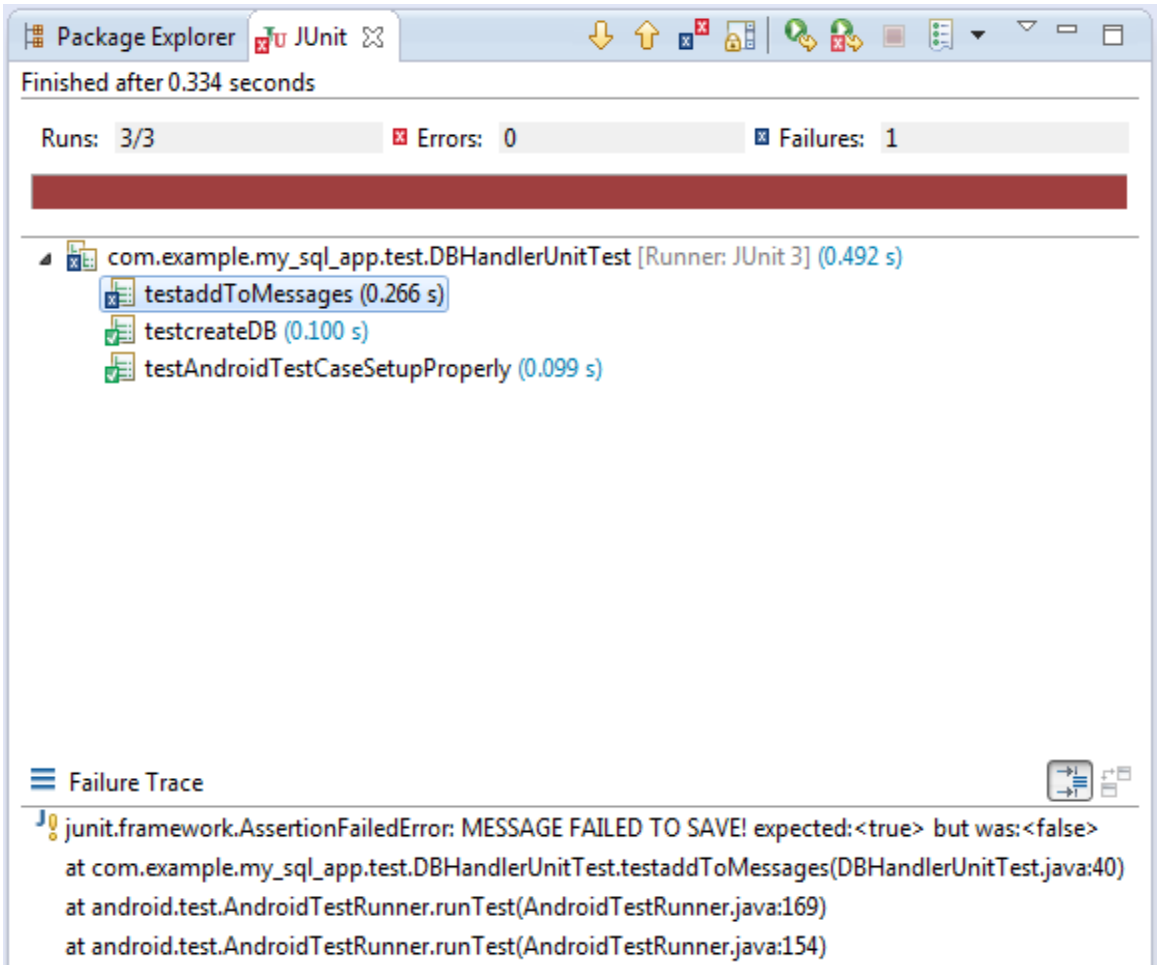
Database Handler Component Unit Test

The requirements for this component, the database handler, was to set up and successfully create a database and its required tables to persistently save the messages sent to and from clients and keep track of the logged in user. Therefore the unit test for this component includes three cases, namely to set up the database, successfully save information to the database, and to delete chat histories. Firstly to test if the database and its required tables are in fact successfully created, and not a null value after the create statement is called. The following images depict the results of this unit test on failure of this requirement, i.e the database is null.



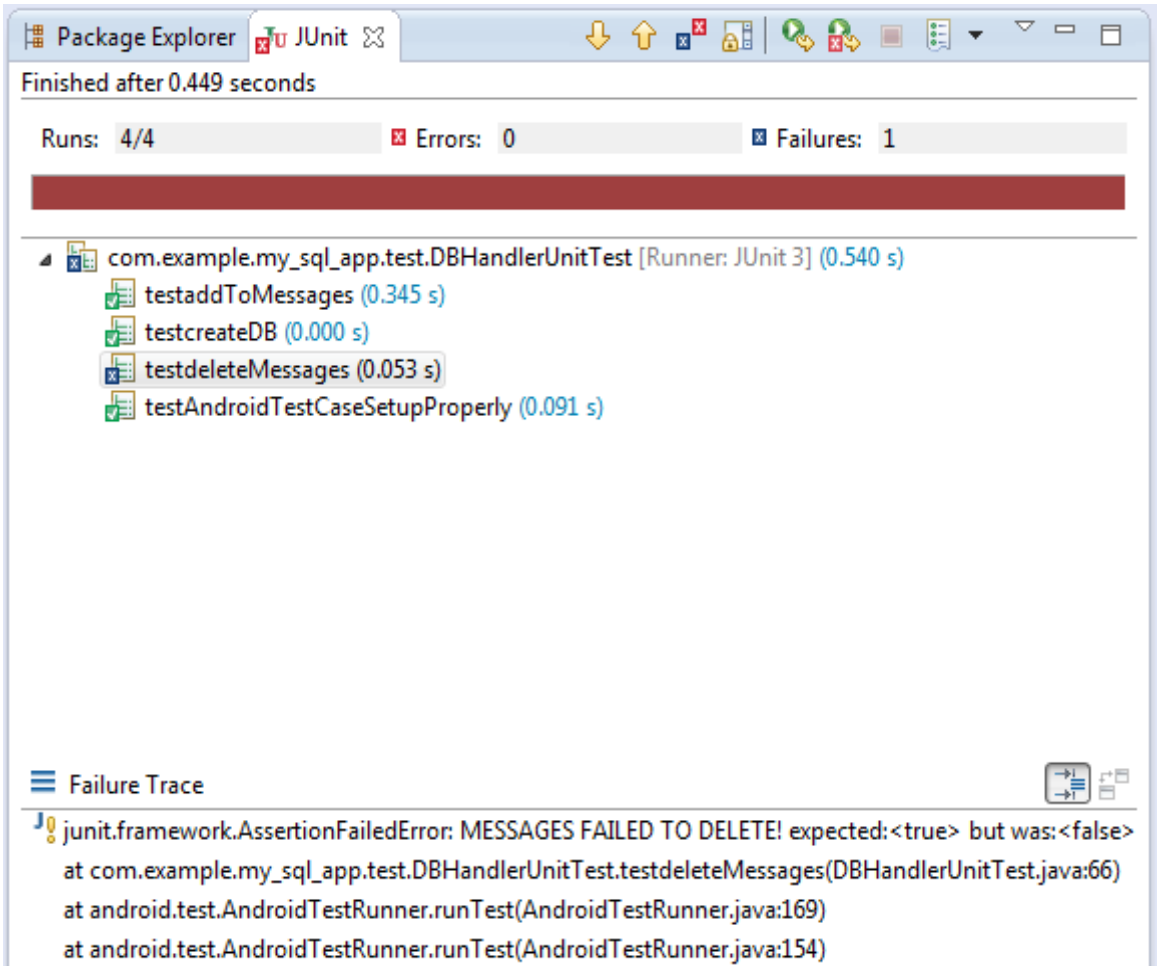
[Figure 1] Database Create Test

Secondly, the next case for this component's unit test is to ensure that the data, including the messages sent or received, is actually saved to the tables after a method like addToMessages or addRememberMe returns, i.e the data is not found in the database table after the method returned. The images below depict the results of this unit test case on failure of this requirement.



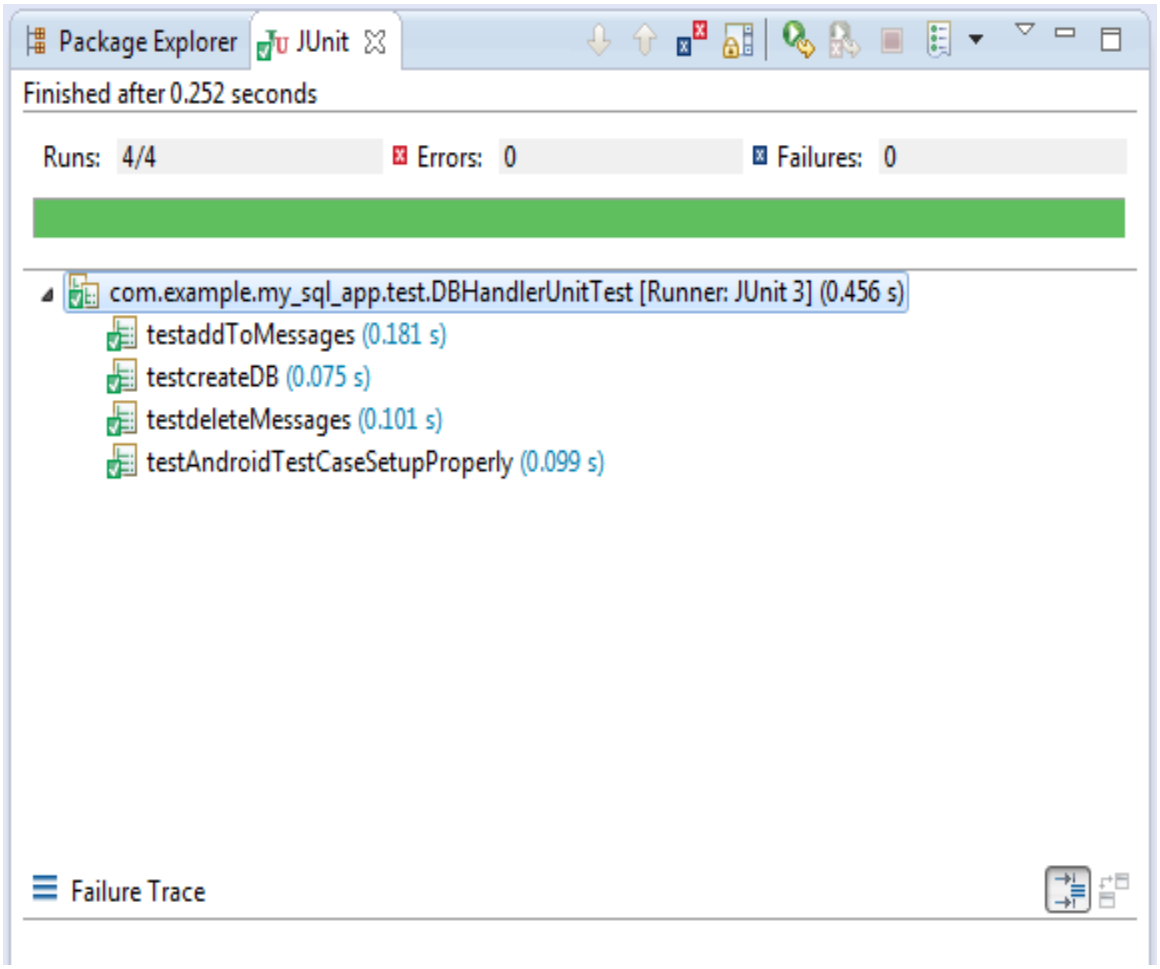
[Figure 2] Message Save Unit Test

The third test case is to ensure the when a user wants to delete chat history, all the relevant information is removed from the database. If all the data is successfully deleted, the test case will return true, otherwise it will return false and fail. The image for the failure of this component is depicted below.



[Figure 3] Deleting Message Unit Test

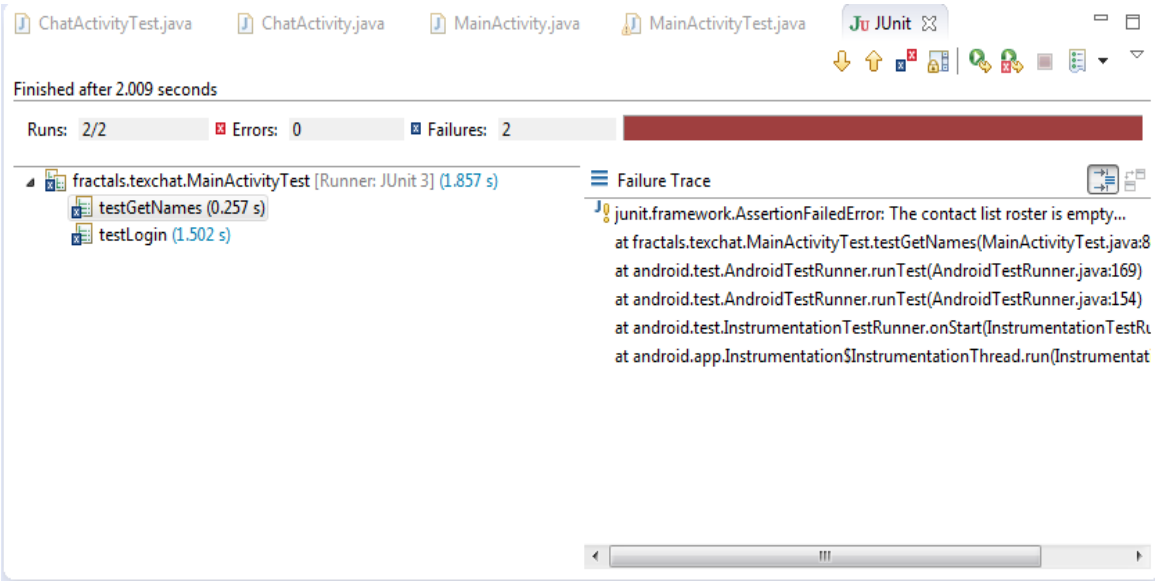
When these three cases do not fail, and the database and tables are successfully created and the methods addToMessages and addRememberMe successfully saves data, and deleting a chat with a user, removes all the relevant information the following results can be expected from the unit test for this component, i.e when successful.



[Figure 4] All Tests Successful

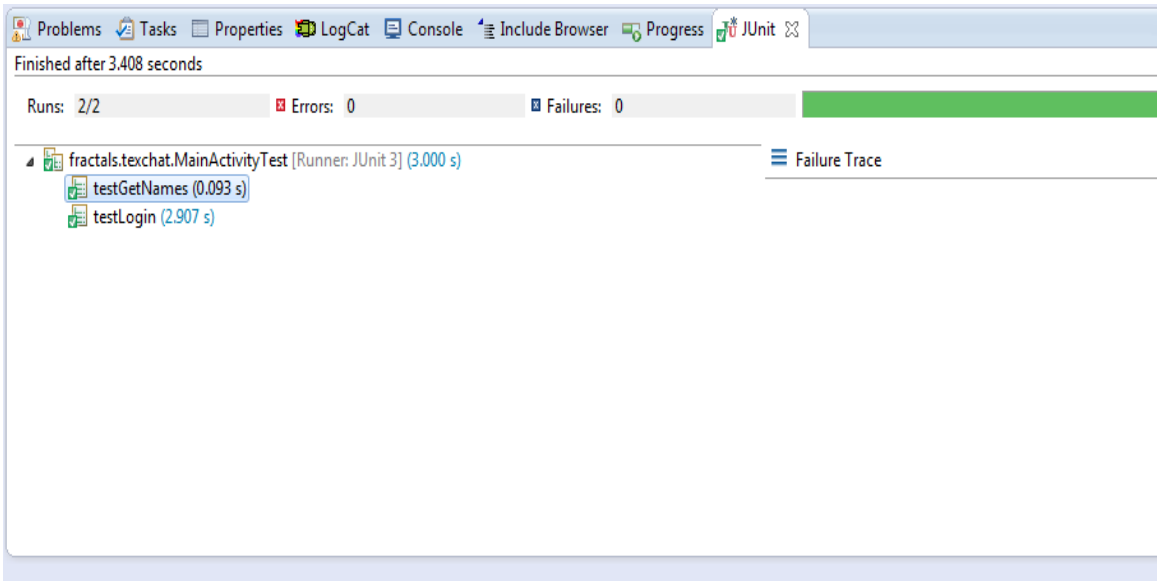
Main Activity Unit Test

The Test for Main Activity tests the successful login onto the server.
If the Null test returns null then it means that the test was unsuccessful. Both of those tests are displayed below.
The diagram below depicts the assertNull test which fails when the object returns null.



[Figure 5] Logging in to server, Unsuccessful, getting names unsuccessful

The diagram below depicts the assertNotNull test which succeeds when the object returns not null.



[Figure 6] Logging in to server, Succcessful, get names successful

Integration Testing
Testing complete functionality within the system

Non Functional Testing
Performance testing, load testing

4. Open Issues

- Changing the code to adapt to the unit testing.

5. Glossary

- Agile - Development methodology
- NDK - Native Development Kit
- JUnit - Testing Framework