**CHAPTER FOUR**

**RESULTS AND DISCUSSION**

**4.1 Introduction**

Chapter four describes the implementation and analysis of the intranet based IM system whose architecture has been discussed in chapter three. The architecture described includes a number of components and each component has been implemented as a separate module. It is a mobile application on android operating system. It shows the implementation and other design issues, which are common to mobile applications, and the way they have been tackled in this IM application software. The system developed ensures that the user is able to send and receive chats using the intranet and guaranteeing the security of the messages as well as other user’s data to make sure they are secure from unauthorized access.

**4.2 Implementation Environment**

The implementation of this work was done using the Android Studio. Android Studio is an integrated development environment (IDE) for developing primarily with Java, but also with other languages. It is also an application platform framework for Java Mobile applications and others. The Android Studio IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The Android Studio Platform allows applications to be developed from a set of modular software components called modules. Applications based on the Android Studio Platform (including the Android Studio IDE itself) can be extended by third party developers. Android Studio IDE is an open-source integrated development environment. Android Studio IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactoring, and version control (supporting CVS, Subversion, Git, Mercurial and Clearcase).

All the functions of the IDE are provided by modules. Each module provides a well-defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. Android Studio contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow Android Studio to be extended. New features, such as support for other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the Android Studio IDE.

From July 2006 through 2007, Android Studio IDE was licensed under Sun's Common Development and Distribution License (CDDL), a license based on the Mozilla Public License (MPL). In October 2007, Sun announced that Android Studio would henceforth be offered under a dual license of the CDDL and the GPL version 2 licenses, with the GPL linking exception for GNU Class path Framework for simplifying the development of Java Swing desktop applications. The Android Studio IDE bundle for Java SE contains what is needed to start developing Android Studio plugins and Android Studio Platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again. The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are: User interface management (e.g. menus and toolbars), User settings management, Storage management (saving and loading any kind of data), Window management, Wizard framework (supports step-by-step dialogs), Android Studio Visual Library and Integrated development tools. Android Studio IDE is a free, open-source, cross-platform IDE with built-in-support for Java Programming Language.

**4.3 System Requirements**

The development of this work was carried out on a HP GS78 Laptop Computer with the following specifications;

* 500Gb Hard disk
* 4 Gigabytes RAM (Random Access Memory)
* 4 Gigahertz Processor Speed (Intel Corei3).
* 64-bit Operating System (Windows 7)

The deployment and testing was carried out on Samsung S6 mobile phone with the following specifications:

**4.4 Component Interaction of the Developed Intranet Based IM System**

This is an intranet peer to peer text chat system. The user existing user logs in with a username and password while new users register with a username and password. Successful login grants him access to the app messaging feature. A new user Chat list will be empty at first but the new user can search for users on the app by searching for their username and can then start a chat with the other user. An existing user will have the chat list populated with the most frequent chat and can chat with any one on the list. A user can also log out from a device and can log in on multiple device also. To send message, the user types the message and the message is been sent to the server which stores it and sends a sent receipt back to the user.

To receive a message, a background process starts when the user opens the app which checks for any undelivered message sent to the user by a contact from a server, the server returns any undelivered message and the background process saves the message and update the user interface of the new message, all these processes are shown in figures 4.1 to

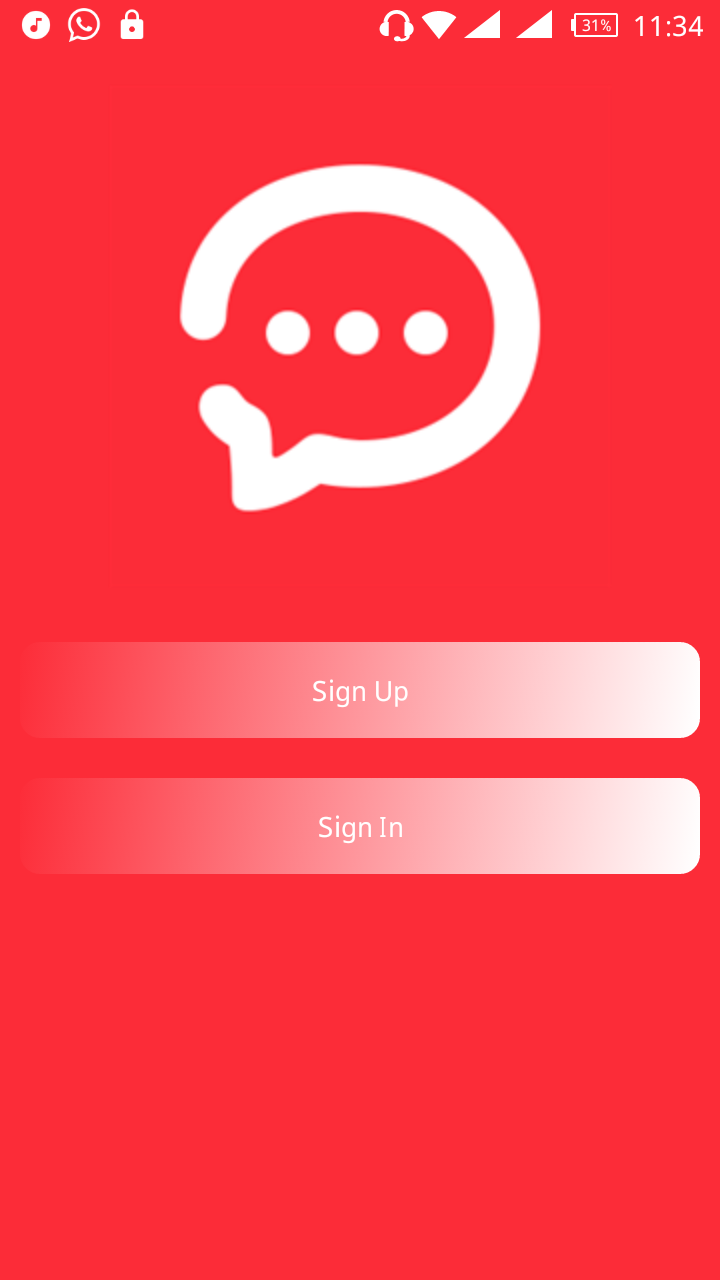


Figure 4.1 The signup page

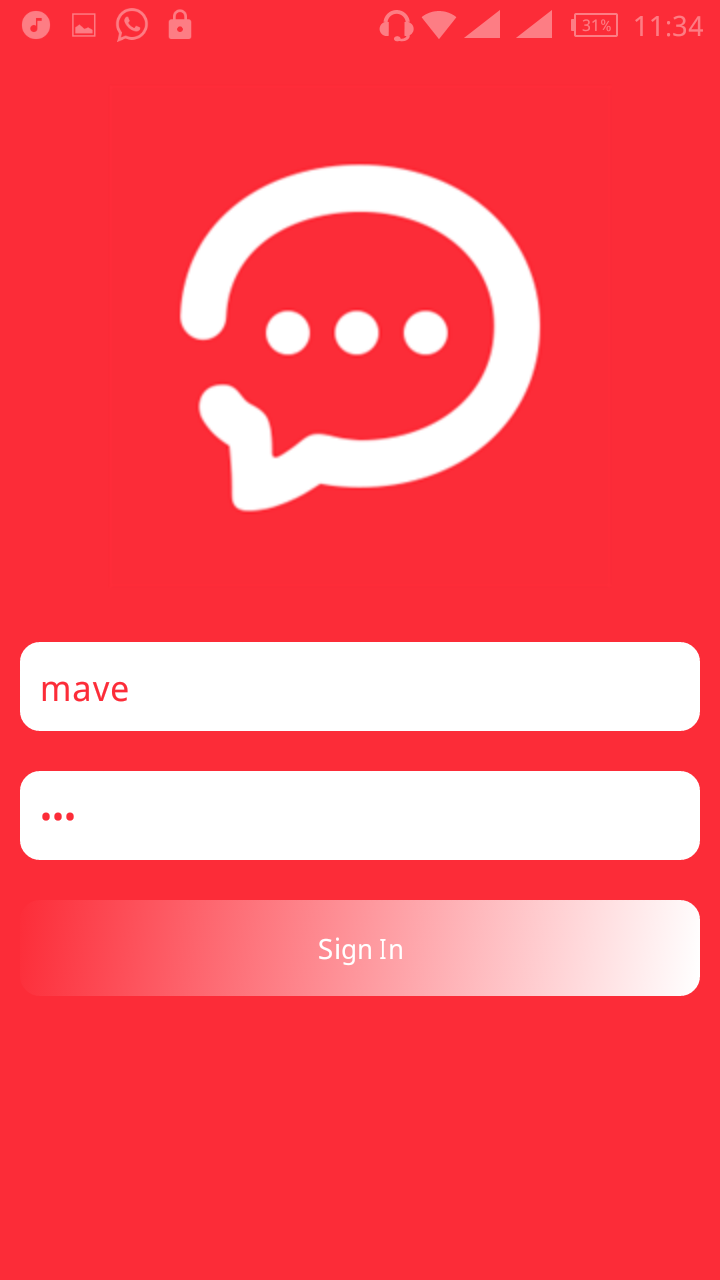


Figure 4.2 Existing user login

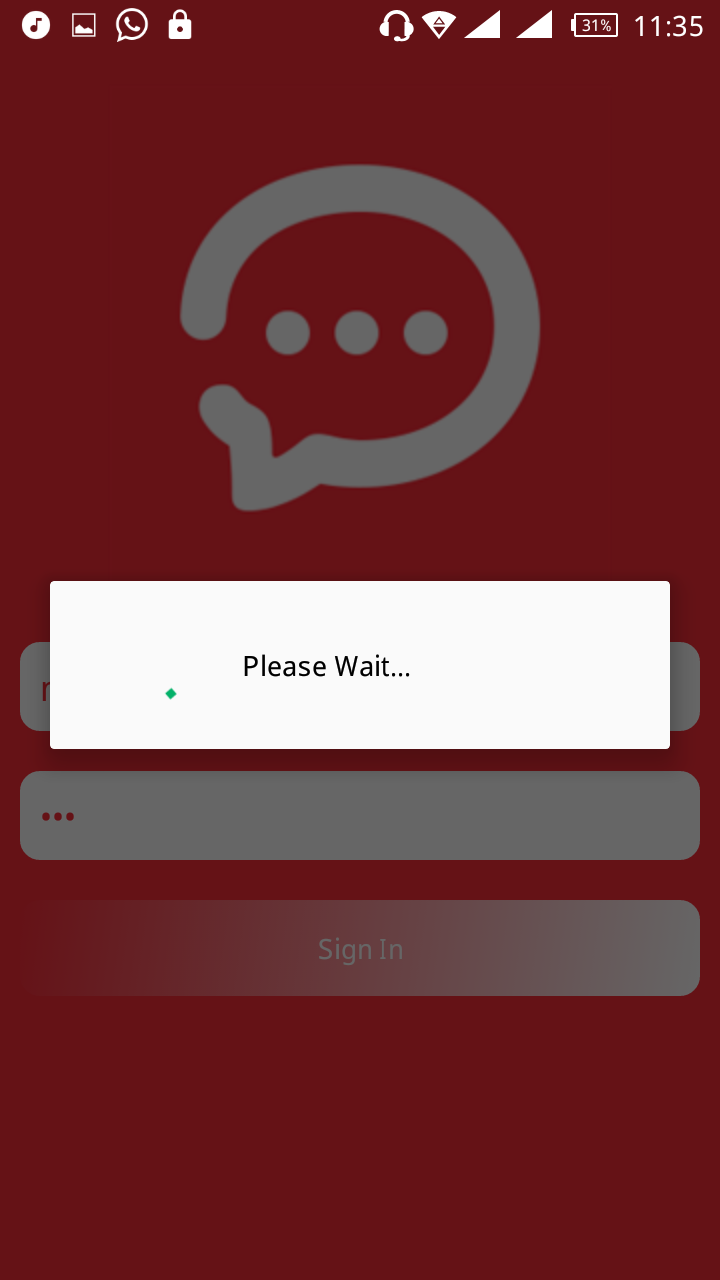


Figure 4.3 the Login process

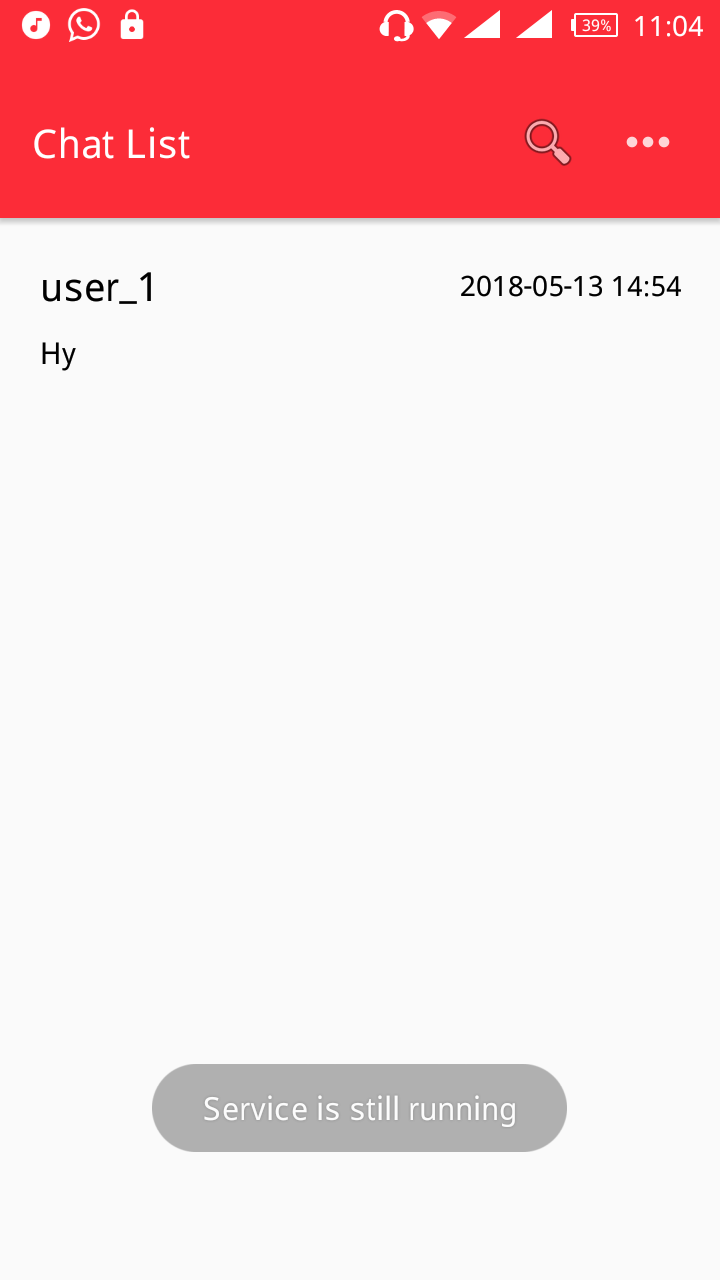
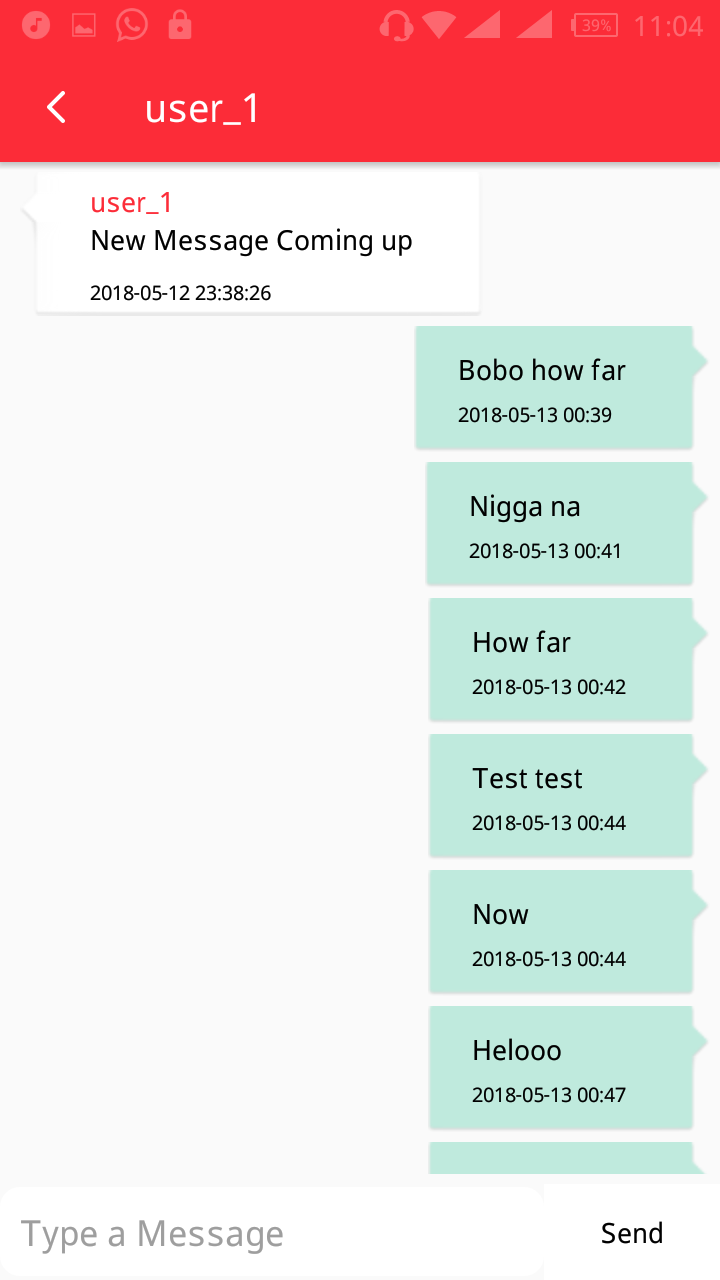


Figure 4.4 User chat list

Figure 4.5 Chat Interface

**4.5 Application Testing**

**4.5.1 Testing Plan**

Testing is one of the most important part of a software development life cycle. There are different ways of approaching an application to test. The items listed below will focus on identifying the scope of testing this IM application.

* Items / functionality to be tested
* Testing Strategies and approach to be followed
* Identifying the roles and responsibilities
* Test Deliverables
* Define the Entry and Exit Criteria for each phase of testing
* Test Environment-Hardware / Software
* List Risks and contingencies / mitigation plans to overcome

The scope includes development and execution of the test suit consisting of the test scenarios for Unit Testing, Integration Testing, and System Testing features. They are as follows:

* Unit Testing needs done at the development stage
* Test cases are based on Use cases and Requirement specifications discussed in chapter three
* Test execution carried out mostly through manual approach

**4.5.2 Test Approach**

The different levels of testing that are carried out during the Test Execution are shown below in Table 4.1

Table 4.1 levels of testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Types** | **TestLevels**  **(Risk Level)** | **Strategy / Methods** | **Techniques for identification of test cases** |
| Unit Testing | Low | Black Box Testing | Basic function testing. |
| UI / Usability Testing | White | White Box Testing | Basic ease-of-use and function testing. |
| Functionality Testing | High | Black Box Testing | Test scenario shall be identified and based on that test cases shall be developed. |
| Usability Testing | High | Black Box Testing | Specify the functional areas to regression needed. |
| Regression Testing | Low | Black Box Testing | Specify the functional areas for which regression is needed. |
| Data Integrity | High | Database Testing | This will validated by using a set of database scenarios with set of database scenario with set of available or prepared data. |

**4.5.3 Entry Criteria**

The following conditions are met for test plan entry criteria:

* All functionality described in requirement specification has been implemented
* All unit and Integration Test cases should have been successfully executed
* Test environment includes test database with the master data operational

The android app on the other hand is first run on the Android emulator, included in the

Android SDK in harmony with the IDEA intelliJ IDE. However an emulator is good for checking functionality but for tests on usability and real-time tests, the hardware devices (android mobile phones) are used.

**4.5.4 Setting up the Test Environment (Installing the App)**

The native app when compiled has the file format "apk". The file must be transferred to the mobile device first and installed. This is done via USB. But since android does not allow to installation of non-market apps by default, the option has to be set to allow it for that. Once this is done, dummy trial accounts are created for testing.

**4.5.5 Testing Methods Overview**

**4.5.5.1 Functional Testing**

Here the basic functionality of the app is tested first one at a time individually then after integrating it with other modules continuously and performing functionality tests. It includes:

* Functionality of each module will be covered based on requirements specifications
* Check for valid and invalid data
* Tests carried on local host intranet
* Checks to ensure interdependence of modules

**4.5.5.2 Data and Database Testing**

Here the data stored on the database is checked for consistency and validity. It includes:

* Data checks will be done on all types of database in scope
* Data checks like insert, update, retrieval will be done for all possible transactions
* Check whether database logging of data is proper or not
* Data validity check

**4.5.5.3 UI / Usability Testing**

This section deals with testing the basic "feel" and usability of the app from the end-users’ point of view. It includes:

* Tests include look and feel aspects
* Running the app in landscape as well as portrait
* Ease-of-use tests

**4.5.5.4 Performance Testing**

This section deals with testing the performance of the app during regular use. It includes:

* Check response time for messages to be sent and delivered and other time sensitive requirements to ensure that it is consistent with the requirements
* Memory usage and CPU usage tests for the mobile phones and PC for local host server.

**4.5.5.5 Failure and Recovery Testing**

This section deals with testing the app’s functionality on failure of any service or the app itself and their recovery mechanisms. It includes:

* Check for abnormal shutdowns for the application, system failures / network malfunctions.
* Check for data loss due to data corruption, system failures, and database failures.

**4.5.6 Test Results Overview**

The tests were executed in process and the results were documented. Most tests were completed successfully and a few defects were encountered. The most troublesome defect was that the App crashes when the Wifi network is not put ON because it cannot reach to the local server connection. This is a problem and defect with the API and little can be done to fix it. In addition to this possibly, since the servers are hosted on the local host platform, the performance is up to par with the internet connection. This is a major result.

Although, few defects were encountered and fixed. The code was continuously updated and kept track of using GIThub account repository. This distributed revision control and source code management (SCM) gives a clear overview of the changes was made and a proper version tracking enabled to track changes and affect bug fixes.

**4.6 System Maintenance**

This type of maintenance proves the lifespan and efficiency of the program. Having a mobile android based secure reminder system on one’s phone has its responsibilities; proper and regular maintenance would make it run smoothly always. The system developed will stand the test of time if properly maintained.

**4.7 Users’ Guide**

This reminder system has been developed to a level that will enable users to navigate around easy and quick. The system is menu driven whereby, for any particular operation carried out, a menu must be specified and there are few number of options for the user to select depending on what the user wants to do. Therefore, the application has been made simple enough for use, and no extra document needed as users’ guide aside the details offered in this chapter four of the project write-up.