Instant Messenger Plus : Chat and voice calling application

Dulanjaya Tennekoon
University of Moratuwa, Katubedda, Sri Lanka
dulanjayatennekoon@gmail.com

***Abstract*—The purpose of the Instant Messenger Plus Project is to provide an efficient way of communication through instant messaging by featuring with test messaging, file sharing and voice calling over the internet. The goal of the project is to develop a desktop application which follows the international standards in implementing the above features. The system development practices the Rational Unified Process (RUP) as the system development methodology. The Model-View-Controller (MVC) design pattern is used in the system design. The end product is a standalone desktop application which works according to the client server architecture. The Instant Messenger Plus application comes with an attractive interface for the user to engage in instant messaging, voice calling and file sharing with other users.**

Keywords—Client Server Architecture; Extensible Messaging and Presence Protocol; Instant Messaging; Institute of Electrical and Electronic Engineers; Rational Unified Process; Voice over Internet Protocol

I. INTRODUCTION

The Instant Messenger Plus (IMP) is an instant messaging application that allows users to send messages, make calls and send files to other Instant Message Plus users. The system is based on the client-server architecture which has a server to maintain the overall communication and an application for the client side. A user can send and receive messages securely through this system which implements the Extensible Messaging and Presence Protocol (XMPP) [1] in both the server and the client sides. The messages that can be sent through this system will be text messages, or images. Users can also make calls to other users through Voice over Internet Protocol (VoIP) methodology.

Instant messaging is a concept which is massively used today by billions of people around the world. When comparing the current GSM charges with the internet data rates, the low expenditure for internet data lead people to use instant messaging concepts widely in fulfilling modern communication needs. The instant messaging applications become popular in this context. The main motivation behind the development of the Instant Messenger Plus system is to provide a user with a comprehensive and a user friendly application which provides instant messaging services in an effective and a secured way.

Modern communication has become more prominent with the manifestation of the internet. Instant messaging, which is a use of the internet has become an efficient way of addressing the modern needs in communication. However, the security is a must in implementing the modern day communication to protect the users from threats. The IMP, as the main purpose, addresses the above communication need in an effectual way. This system offers three ways of communication; texting, voice-calling and file sharing and in implementing each of the above systems, the IEEE organization's standards and recommended protocols are used to provide the security of communication.

Instant Messenger Plus (IMP) is a desktop application which works according to the client server architecture and works with the Ignite Realtime Openfire server [7]. The application is developed according to the Rational Unified Process (RUP) software development methodology. The design of the application follows the Model View Controller Architecture. The end product is a client application that works with the Openfire server application that can be integrated for any organization by deploying the Openfire server application on a reserved server for the organization. The application can also be integrated for general users by hosting the Openfire server application in a commercial server.

II. LITERATURE REVIEW

Instant messaging (IM) and internet chat communication has a huge growth among the present population [1] and continues to display a strong growth in the market in recent years [2]. Instant messaging application become popular in this context having huge revenues and investors believe in continuing rapid growth in the market of instant messaging [3]. Therefore, developing an IM application by introducing new features has a huge value. The IM applications are becoming very important for internet companies accordingly [4].

The Instant Messenger Plus application has been developed considering the facts mentioned above. The application communicates using the Extensible Messaging and Presence Protocol. The XMPP is based on the Extensible Markup Language (XML) which provides the near real-time messaging and presence [5]. The Smack library which is an open source client library [6] is used for the implementation of the XMPP protocol in the Instant Messenger Plus application as the library provides more flexible features in implementing the communication functions. The IMP system has the client server application architecture. Ignite Realtime Openfire server which maintains the simplicity in

its deployment and which is a real time collaboration server license under the open source Apache License [7] is used as the communication management server while the IMP application works as the client side application.

The development of the application is done by following the Rational Unified Process which is an iterative software development process framework [8]. The Graphical User Interface of the application is very important as it can be the difference between the application acceptance and rejection [9]. JavaFX which is a set of graphics and media packages that comes with the Java language is used for the development of the rich client application [10] that operate reliably and consistently.

The "Facebook" has the instant messaging feature with the "Facebook messenger", and is a widely used instant messaging service. It provides the video calling feature as well [11]. However, the "Facebook" itself has some privacy issues which become more privacy concerns as for an instant messaging application [4]. The "WhatsApp" application is another messenger application which works with WhatsApp's own server [12]. However, the "WhatsApp" instant messaging system cannot be implemented on a private organization with their own server to enhance the security. The 'Spark' application is an Open Source, cross-platform IM client optimized for businesses and organizations. It features built-in support for group chat, telephony integration, and strong security [13]. But the user interface of the 'Spark' application is not much appealing, is very complex and does not offer a great end-user experience.

The Instant Messenger Plus application has considered the above issues and comes up with features to avoid those issues. The security of the applications is achieved through the use of XMPP protocol. The system can be implemented on a private organization only for the usage within the company. The same client application can also be used as a general instant messaging application available for the general public after deploying the Openfire application server in a commercial web server. The appealing user interfaces of the Instant Messenger Plus application provide a better usability and simplicity to the user. The application is developed considering the user experience in a greater extent.

III. SYSTEM MODELS

A. System Requirements

As functional requirements, initially the new user should be able create a new account in the server. The registered user needs to log in to the system with the user credentials. The user account of the user should be maintained with the details of the user and with a profile picture. As the main functional requirement, the user should be able to share instant messages with other users. The text messages which are sent by a user to another user should be delivered to the second user even though the second or the receiving user is not available online. The system should make notifications regarding the received messages and receiving calls. Furthermore, the user should be able to send picture messages to other users who are available online. The picture messages should be appeared on the chat view of the application. Moreover, the file sharing capability should be in the Instant Messenger Plus application.

Both pictures and files which are received for a particular user should be stored in a general folder in the file system of the operating system, so that by clicking the file name appeared on the chat view opens the file location highlighting the received file. Furthermore, the user should be able to make voice calls with the other users of the system. The Instant Messenger Plus application should use the computer mic and the speakers for the voice communication. Moreover, the user should be able to create group conversations and manage the group conversations. The users should be able to join the conversations which are already created. The system should have both open and closed conversations so that the closed conversations have a password to enter the group chat. The open conversations are free for everyone to join, and any user who is interested in a particular open group conversation can participate. Finally, the application should have the ability to manage contact lists with the features to add new contacts, to remove existing contacts, to accept friend requests and to reject friend requests. The messages received by the users should be stored in the application for every user who logs in to the server through a particular, same Instant Messenger Plus instance. The user should be able to disable the chat history storing feature and the notification feature of the received messages from the application as well.

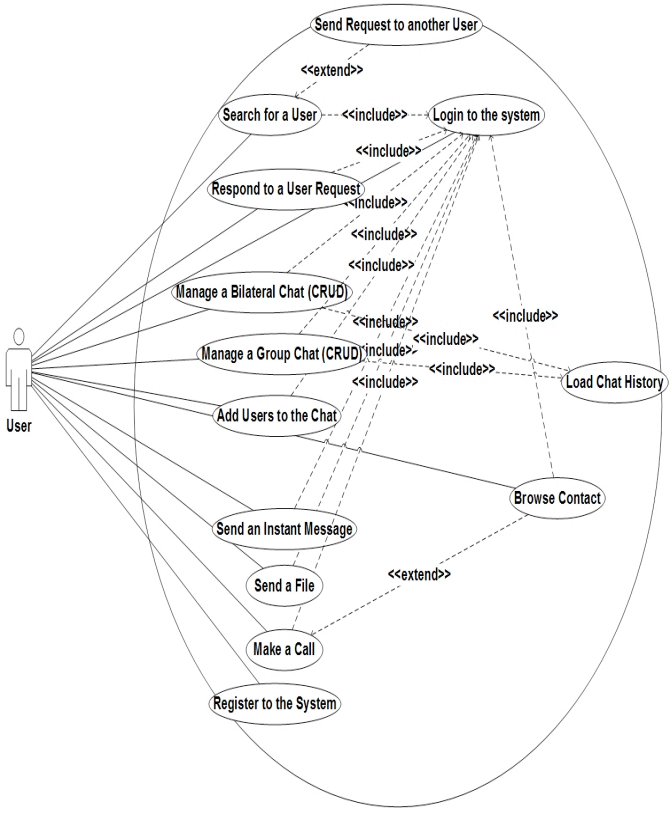

Fig. 1. Use Case Diagram

B. *System Design*

The logical view of the Instant Messenger Plus application describes the most important classes of the system and their organization. MVC design pattern has been applied in the UML class diagram to enrich the quality of the system architecture of the system and to make the application in such a way that the business logic is independent of the presentation logic. The system is based on the client server architecture, so that the application works with the Openfire server application. The connection management class builds the connection with the server to build the client server communication. The Smack API is used for the implementation of the XMPP protocol in the application.

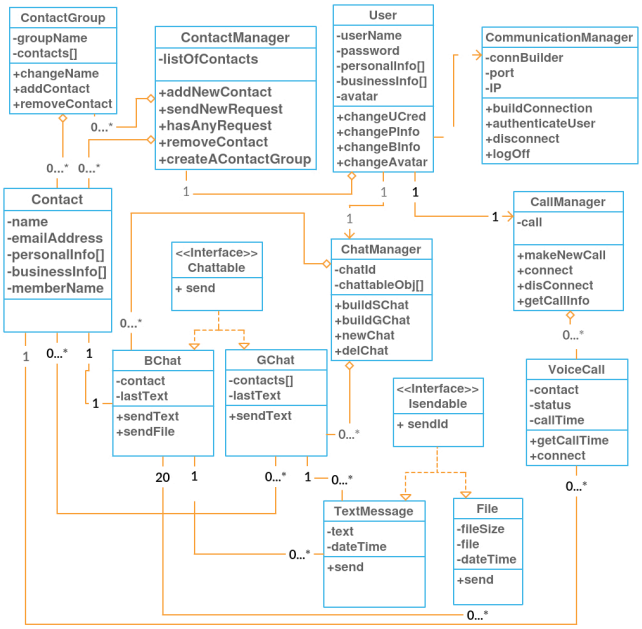

Fig 2. Domain Class Diagram (Business Logic)

The sequence diagram shows the interaction of making a group conversation. The diagram depicts the flow of the information among the objects instantiated with the blueprints of the class diagram.

The information which is passed in the system includes the communication in the client server system as well. The information that passes in the client server system is done through the implementation of the XMPP protocol in the application using the Smack library.

C. *Database Design*

The Instant Messenger Plus applications uses a database to manage the chat history of the user conversations. The single user conversations which are created by the user with another user will be stored in the database. The messages stored in the database include all the text messages, picture messages and the files. The database has only a link to a received or a sent file or a picture, which is stored in the file system of the computer.

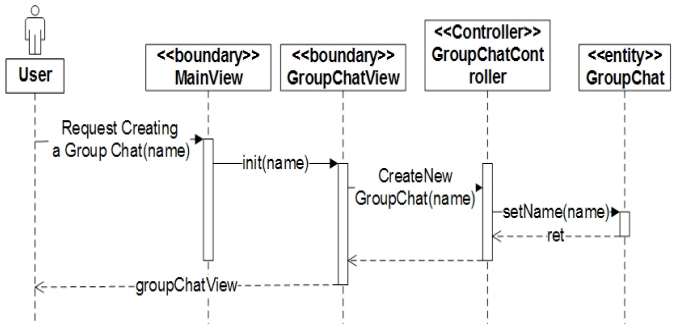

Fig. 3. System Sequence Diagram

A single instance of the Instant Messenger Plus application which is installed in a particular computer may be used by multiple users. Therefore the database of an instance of the IMP application keeps the records of every user who logs in to the system through that application instance. However, the application logic can control the enabling and disabling of the chat history as well as the features like clearing the chat history. The history of messages in a group conversation is not stored in the database as the chat history of a group conversation can be obtained from the server.

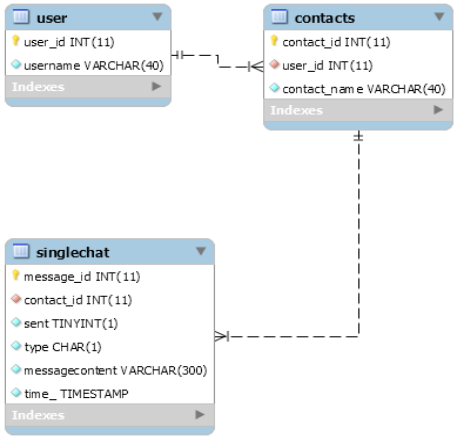

Fig. 4. Database Schema Diagram

IV. SYSTEM IMPLEMENTATION

A. *Implementation Procedure*

The Instant Messenger Plus is a standalone desktop application which is built according to the Client Server Application Framework. The Ignite Realtime Openfire Server application [7] is used as the server in the application architecture. The Openfire server is an off the shelf product developed by Ignite Realtime.

The Extensible Messaging and Presence Protocol (XMPP) is used to employ a secured model in the application and the Ignite Realtime Smack library is used in implementing the XMPP protocol in the client application. The XMPP protocol has a security built-in feature developed

by the Jabber cooperation as the protocol is being developed introducing security and online presence features to the existing HTTP protocol [5].

Development process is done in such a way that the development follows up with unit testing. Junit serves as the unit testing framework of the application development. The development of the application is done using the NetBeans IDE and using Java language version 1.8.0_91. Java default packages and APIs are used in the development. The graphical user interface (GUI) is designed with the use of the JavaFX technology and associated libraries. Thus the user interfaces are written in XML, the controller logic is written in the Java language and the GUI design enhancement is done using the Cascade Style Sheets (CSS).

The UX (User Experience) is a primary concern in the application architecture. The available wire framing tools are used to mock interfaces prior to build the final GUI design of the application. The GUI design is done using the JavaFX Scene Builder.

Furthermore, the application uses a Model View Controller architecture pattern in the application design. Java language has the new JavaFX design platform that consists of an in-built view controller design. The business logic is developed in parallel to the in-built design. Finally, as resources, the instant messenger plus project is done with the help of the documentation of the Smack API, XMPP protocol and the community support of the Ignite Realtime. The project is open collaborated as an open project in GitHub.

B. The Algorithm

Initially, the user needs to login to the user account with user credentials in order to send an instant message to another user,. Once the user credentials are entered in the login view, the application establishes the connection between the application itself and the server. The connection management is done from the connection manager class which is a model class. If the server is not available or if there is an internet connectivity problem, the connection manager class throws an illegal state exception which is handled in the logic controller class. Then the application executes the login procedure and if the user credentials are wrong, the application will handle that exception and prompts a notification message. If the user credentials are correct, then the login controller calls the main controller class to view the main view of the application.

```
Begin
Input user credentials
If the user credentials are correct then
        Login to the system
        Start message listener, call listener, file listener
        Start roster listener
Otherwise
        Display 'Bad Credentials'
End
```

Fig. 5. Login Action – Pseudocode

Once the contact is chosen from the contact list of the main view, the chat list view of the particular contact will appear. The contact list runs with a contact listener in order to implement this feature. Then the messages will be displayed in the chat list if the user has a previous chat history with that particular user. In this case, the system loads the chat history from the application database. If the data is not available then the application will create new contact id for that chat instance and begins to record the chat history.

```
Begin
Choose a contact
Load contract view
Display contact information
If database contain previous chat history then
        Display messages in the contact view
Otherwise
        Create a chat id in the database
End
```

Fig. 6. Create a chat – Pseudocode

The messages which are to be sent will be first sent to the contact. If the server is not available at that moment, then the application will display the notifications regarding the scenario. Once the message is successfully delivered, the message will be stored in the database with the information like the timestamp and the message type. Then the message will be painted on the chat list. The same procedures are followed in sending pictures and files too.

```
Begin
Enter a message
Send the message to the receiver
If the message sent successfully then
        Save the data in the database
        Display the message in the chat view
Otherwise
        Display 'Unable to send'
End
```

Fig. 7. Send a message – Pseudocode

If the message listener listens a receiving message, then the received message will be stored directly in the database with the sender's and receiver's identities and with the timestamp. Then the message will be displayed in the chat list view if the currently selected chat is in between the user and the sender of that received message. The same procedure is applied to the file and photo receiving scenarios as well.

V. MAIN INTERFACES

Initially, a user needs to enter the username and the password in this view to login to the system as shown in Figure 8.

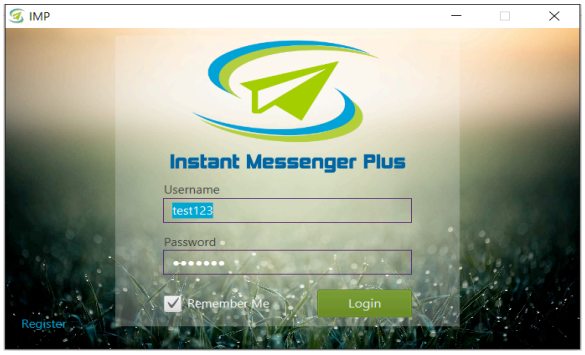

Fig. 8. Login View

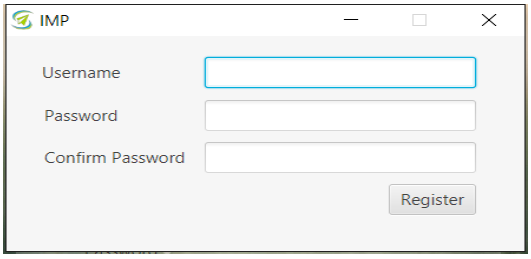

Fig. 9. Register View

User can register in the system with a username and a password as shown in Figure 9. Figure 10 shows the main view of the system including the contact list, conversation list, chat view and the account information. The current conversation is loaded to this user interface. The user can edit his or her user profile using the view shown in Figure 11 and the changes will be updated in the server once the user press the save button.

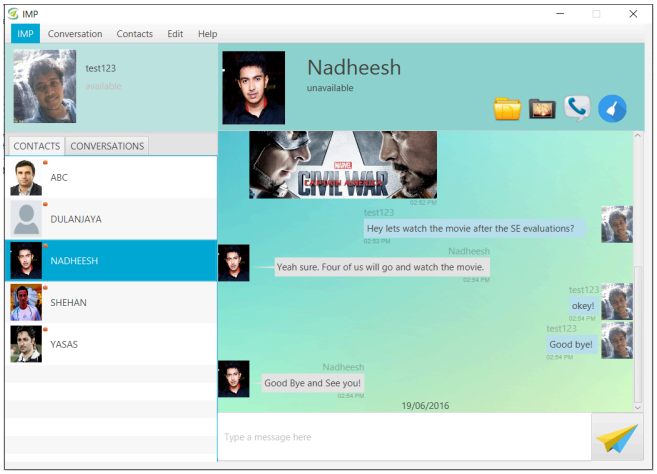

Fig. 10. Main View

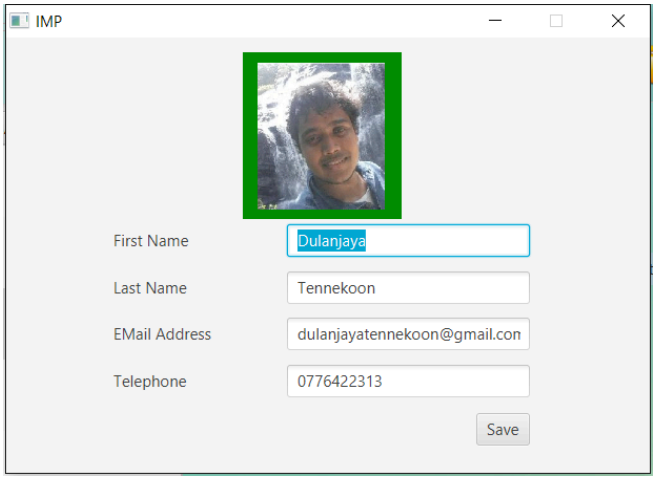

Fig. 11. Account View

VI. System Testing and Analysis

A. Testing Approach

The system tests are done in several approaches. Unit testing is done in order to test the functionalities of the system and each and every model class, and controller class has been tested writing unit test cases. Junit testing framework is used for the unit testing in the system. Moreover the same framework is used for the database testing as well. The database CRUD operations (Create, Read, Update, and Delete) are tested for the verification of the correct functionality of the database handler and database access classes. This is done in such a way that the user interface is completely isolated from the database.

Furthermore, the user interfaces have been tested for the correct functionality and performance. User Interface testing is carried out for JavaFX user interface of the application using the JemmyFX browser. JemmyFX browser was ran on the same IDE and UI tests were recorded. Then for each UI test case, the unit tests were written.

The security testing is done using several approaches. Wireshark software is used to analyze the communication datagrams (packet data send to the server by the application via the XMPP protocol) and verifies the system security that can be violated by malicious hackers.

Configuration tests are done for the Microsoft Windows Operating systems, by testing the application for the correct functionality (i.e. establishing the connection between the server and the client) without any security violation policies from Windows's firewalls, antivirus application's firewall and from router's firewall.

Moreover, the performance testing is done with the Microsoft Windows Resource Monitor tool for the limited usage of memory. The application is tested further for any memory leaks in the heap memory of the Java Virtual Machine (JVM) by analyzing the heap memory of the JVM by the 'VisualVM' software. Java test cases are written in order to test the communication performance to be verified in line with the system requirement specifications.

Unit testing is done for the business logic (model) classes and the controller classes. The unit tests are written and verified the correct behavior of each unit. A system bug is once identified from unit testing, that misbehave system by returning some malformed conversation jabber identities (JID) for closed conversations.

B. Aspects related to performance, security and failures

Instant Messenger Plus application is highly concerns with the performance of the communication. The application is tested for the message latency of transfer and the application succeeds the tests to keep the latency minimum to be in line with the system requirements. Furthermore, the application is tested for the memory usage. The tests are done with the Microsoft Windows Resource Monitor.

The application security is an important aspect. The application uses the XMPP protocol in order to implement the communication system between the server and the client application. By using the Wireshark tool, packets are analyzed and the result is as same as the XMPP library elaborates. The received messages are stored securely in the application database and are needed to be protected in the root access. However, the receiving files and photos are not safeguarded as it is mentioned not to do in the system requirement document.

Failures occurred before testing the application when the server is being disconnected while the application is turned on and fixed that bug. If the transferring messages are corrupted, then it is natural to occur failures. However, the application is developed in such a way that it guarantees the delivery, as it confirms the message delivery report via the server.

VII. CONCLUSION

As the conclusion, Instant Messenger Plus has the ability to communicate among users efficiently. The main objective of IMP was to provide an efficient way of instant messaging which has verified to be the output of the proposed system. The system can effectively communicate using the text and picture messages, file sharing and voice calling. The management of group conversations and single chats gives the application a great usability. The user interfaces are very fluid and much appealing. The application is developed in such a way considering the user experience (UX) as a huge concern for a chat application and the output of the project is the successful Instant Messenger Plus application. All the functional, nonfunctional requirements have been covered in the final product.

The communication disciplines are adhered to the standard protocols and therefore, the data and information are safeguarded well. The application is developed in the Object Oriented paradigm and has used the best coding practices as well. The design patterns like MVC gives a much understanding to the source codes and it enhance the future works as well. The software engineering disciplines that are applied to the system lead the system for the future development.

The application is developed with the support of open collaboration and the best solutions were obtained in implementing the features and functionalities. Application is committed to the GitHub as an open project. Furthermore, the application can be developed further to enhance the video conferencing feature as a function to the application, and the user interface can also be improved further, to enrich the user experience.

REFERENCES

[1] Raymond B. Jennings III, Erich M. Nahum, David P. Olshefski, Debanjan Saha, Zon-Yin Shae, Chris Waters, "www.cs.columbia.edu," 2006. [Online]. Available: www.cs.columbia.edu/~nahum/papers/ieee-network-instant-messaging.pdf. [Accessed 19 06 2016].

[2] A. T. M. R. FIRM, "Instant Messaging Market, 2015-2019," THE RADICATI GROUP, INC., PALO ALTO, CA, USA, 2015.

[3] "AppMess," [Online]. Available: http://appmess.com/news/27538-investors-believe-in-continuing-rapid-growth-of-messaging-market/. [Accessed 19 06 2016].

[4] P. Sikka, "Market Realist," MArket Realist, 26 09 2014. [Online]. Available: http://marketrealist.com/2014/09/different-social-networks-making-use-users-data/. [Accessed 19 06 2016].

[5] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," Jabber Software Foundation, 08 2004. [Online]. Available: https://xmpp.org/rfcs/rfc3920.html. [Accessed 19 06 2016].

[6] "Ignite Realtime: Smack API," Ignite Realtime, [Online]. Available: http://www.igniterealtime.org/projects/smack/. [Accessed 19 06 2016].

[7] D. Herzmann, "Ignite Realtime: Openfire Server," Ignite Realtime, 21 03 2016. [Online]. Available: http://www.igniterealtime.org/projects/openfire/. [Accessed 19 06 2016].

[8] "Rational Unified Process - Wikipedia, the free encyclopedia," Wikipedia, 31 05 2016. [Online]. Available: https://en.wikipedia.org/wiki/Rational_Unified_Process. [Accessed 19 06 2016].

[9] M. G. Miranda, "THE IMPORTANCE OF GRAPHIC USERS INTERFACE, ANALYSIS OF GRAPHICAL USER INTERFACE DESIGN IN THE CONTEXT OF HUMAN-COMPUTER INTERACTION," in *3rd International Conference on Education and New Learning Technologies*, Barcelona, Spain, 2011.

[10] M. Pawlan, "What Is JavaFX?," ORACLE, 04 2013. [Online]. Available: http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm. [Accessed 19 06 2016].

[11] Wikipedia, "Facebook Messenger - Wikipedia, the encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/Facebook_Messenger. [Accessed 19 06 2016].

[12] "WhatsApp Legal Info," WhatsApp, 07 07 2012. [Online]. Available: https://www.whatsapp.com/legal/. [Accessed 19 06 2016].

[13] "Ignite Realtime: Spark IM Client," Ignite Realtime, [Online]. Available: https://www.igniterealtime.org/projects/spark/. [Accessed 15 10 2016].