**Reliable chat service**

**Software Design Document**

Submitted by:

Sanyam Seth (12BCE0303)

Gurleen Kaur brar (12BCE0538)

Submitted to: Prof. Lydia Jane

Date: 21-February-2015

**TABLE OF CONTENTS**

[1. INTRODUCTION 1](#_Toc414739806)

[1.1 Purpose 1](#_Toc414739807)

[1.2 Scope 1](#_Toc414739808)

[1.3 Overview 2](#_Toc414739809)

[1.4 Reference Material 2](#_Toc414739810)

[2. SYSTEM OVERVIEW 2](#_Toc414739811)

[3. SYSTEM ARCHITECTURE 3](#_Toc414739812)

[3.1 Design model 3](#_Toc414739813)

[3.2 Sequence Diagrams 4](#_Toc414739814)

[3.3 Explanation of the diagrams 6](#_Toc414739815)

[4. Design issues 7](#_Toc414739816)

[The various design issues faced during the development of this system software are: 7](#_Toc414739817)

[I. Inclusion of the image authentication during the user login 7](#_Toc414739818)

[II. Image recognition module 7](#_Toc414739820)

**III. Attaching of files along with the chat box 7**

[5. Design details 8](#_Toc414739822)

# 

# INTRODUCTION

## Purpose

The purpose of the Detailed Design Document is to define the detailed design for all components of the **Reliable chat service** which are specified in the [SRS]. The low-level components are designed, coded and tested

## Scope

The system software uses an encryption algorithm to encrypt the chat between the two users. The system software uses a three password authentication system for login module. This is aimed at protecting the users from unauthorized access of their accounts. The encryption technique aims to provide a secure channel between the users. The software can be implemented on any server or a localhost by providing a connection between two or more systems.

## Overview

This DDD is organized as follows. Chapter 2 of this document is a short introduction to the general context of the application to be made and to the background of this project. Chapter 3 describes the overall architecture and all the components in a consistent way. Chapter 4 consists of a dictionary for related keywords. Chapter 5 consists of Various components description for the project. Chapter 6 includes the User Interface for the project

## Reference Material

*This section is optional.*

**Face recognition**: <http://en.wikipedia.org/wiki/Facial_recognition_system>

**Encryption** : http://en.wikipedia.org/wiki/Encryption

**Opencv**: http://en.wikipedia.org/wiki/OpenCV

**MySQL:** http://en.wikipedia.org/wiki/MySQL

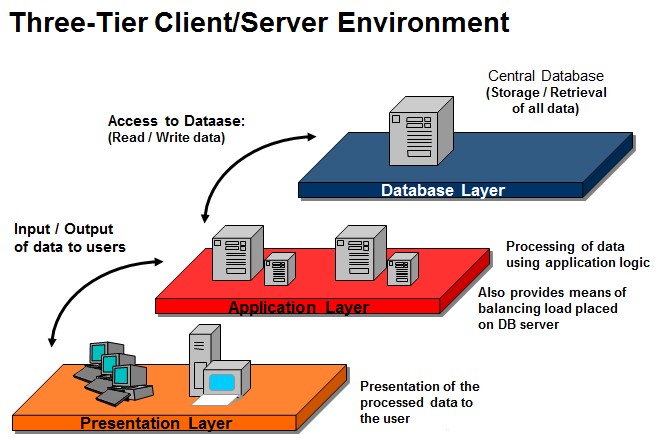
# SYSTEM OVERVIEW

The project is divided into two modules frontend and backend. Frontend consists of all the assets to improve user experience. Back-end consists of various modules to provide secure routes to send data back and forth. Both frontend and backend are further divided into sub modules. The various modules include the login module, sending friend request, searching for friends, sending messages, receiving messages, change passwords, and logout modules. Each has been implemented independently and then connected in the back-end with each other.

# SYSTEM ARCHITECTURE

## 3.1 Design model

This includes the higher level overview of the modules used in the application. The system uses the 3 tier which includes client, server, database server and database. The detailed diagram is given below as:



The top tier of the system is referred to as client tier, is the application’s user interface, which gathers input and displays output. Users interact directly with the application through the user interface, which is typically a web browser. It translates tasks and results to something user can understand. The user chats with the other users directly through this tier.

The middle tier implements business logic, controller logic and presentation logic to

control interactions between the application’s clients and its data. The middle tier acts as an intermediary between data in the information tier and the application’s clients. The middle-tier controller logic processes client requests (such as requests to view a product catalog) and retrieves data from the database. The middle-tier presentation logic then processes data from the information tier and presents the content to the client. Web applications typically present data to clients as HTML documents. Business logic in the middle tier enforces business rules and ensures that data is reliable before the application updates a database or presents data to users.

The bottom tier (also called the data tier or the information tier) maintains the application’s data. This tier typically stores data in a relational database management system (RDBMS). This basically contains the database server along with the database in the backend. The database server is specialized for retrieving the data being queried by the users.

## Sequence Diagrams

**Login:**

****

**Initiate Chat Window:**



**Send Message:**



## Explanation of the diagrams

1. Chat Client - The chat client is the focal point of the client subsystem. After successful connection to the database, a new GUI LoginWindow is created. The login window attempts to login the specified user with the specified password.
2. Login Window - generates a GUI log in window. A single button labeled “Log In” is created. Upon opening the user is prompted to enter both a username and password. When this has been completed and the button is pressed, the username and password to the server for authentication.
3. Chat Server – The chat server is the main server through which chat is executed. Initially the server is activated using a successful connection to the database. Then it also authenticates a user that is attempting to log on with a specified username and password.
4. Send Message- The message is sent by using the chat window. The window is initially opened and the message is sent. The chat is updated simultaneously in both the windows.
5. Receive message- The message is received using the same window by clicking on it. The window is clicked and the message is received by the recipient.

# Design issues

# The various design issues faced during the development of this system software are:

# Inclusion of the image authentication during the user login

# The inclusion of the image during the login of a particular user is the biggest issue afar as deign is concerned of the software. The inclusion of the image is not an easy job to inculcate. The acquisition of the image followed by the matching of the image with the current image in the database requires a huge database and advance knowledge of python image library. Instead of that the idea the one which can be implemented is that we can use three passcodes in a sequential fashion. This will help reduce the database size and increase the software speed.

# Image recognition module

# The image recognition technique is very complex technique and it requires a lot of programming using OpenCV (a library of python) which makes the software slow as the program requires a lot of processing. The database used to store the images becomes very big and complex. It will also lead to the problem of multiple backends. The alternative to this as discussed earlier is the three level passcodes.

1. **Attaching of files along with the chat box**

The files attached along with chat may be of any size which may slow down the software as the file attached with the chat will be stored on the local server i.e the database. This will make the database very complex and also increase its size to large extent. The solution to this problem is that restrictions can be imposed on the size of the files to be uploaded on the server like word documents, powerpoint presentations etc.

# Design details

# The simple class level design of the system to be implemented is as follows:

# *C:\Users\HP\Desktop\diagram600 (1).png*

The explanation of the above sequence diagrams given in the section 3.2 is as follows:

1. The following classes are located within the **Client Subsystem.**
2. Chat Client - The chat client is the focal point of the client subsystem. This class defines the following methods: The **main()** method within this class is executed at start up in order to create a socket connection to the server. The **login()** attempts to login the specified user with the specified password. The **completeLogin()** method is used to request the server send the chat client the complete list of all his/her buddies. The **sendMessage()** methodis used to send a message to the specified user. The **receiveMessage()** method decouples incoming message packets and sends them to the ChatWindow message display. The **userListUpdateMethod()** updates the logged-on status of a single user. And the **showError()** method generates a message box displaying the type of error that has occurred.
3. ChatWindow - The constructor **ChatWindow()** generates a two window box. The top window holds the conversation between clients and cannot be accessed by user. The bottom window is where the client enters his/her information. When typing is completed the user presses the “send” button. This executes the **actionPerformed()** method creates a message, which is displayed in the conversation window using the **updateConversation()** method and then packaged up and sent to the remote client. The **run()** method is defined here.
4. LoginWindow - The constructor **LoginWindow()** generates a GUI log in window. A single button labeled “Log In” is created. Upon opening the user is prompted to enter both a username and password. When this has been completed, and the button is pressed, the **actionPerformed()** method is executed, which sends the username and password to the server for authentication.
5. ContactListWindow - The constructor **ContactListWindow()** generates a GUI to manage and display contacts. Two lists are created in the window. The top displays all the contacts available to choose from. The bottom displays all the users buddies. By double clicking on the name in the bottom list the user can initiate a chat with a contact. The window also listens for messages from the server, which tells the window a remote user has initiated a chat. The **updateList()** method updates the contact list window by changing the logged on state of the specified user to the specified state. The **distributeMessage()** method displays the message from the specified user into the correct window.
6. The following classes are located within the **Server Subsystem**.
7. ChatServer – The chat server is the main class in the server subsystem. This class defines the following methods: The **main()** method is used to initiate the server. The **authenticate()** method authenticates a user that is attempting to log on with a specified username and password.
8. User – The constructor **User()** initializes the attributes of a new User object, which include the username and password, loggedOn to indicate the users current status, an ArrayList of buddies for the user, and an ArrayList of names the user is a buddy of. There are several accessors and mutators defined in this class to access the above attributes.
9. The following classes are located within the **Communication Subsystem**.
10. Command – This interface class defines the following abstract method: The **execute()** method within this class directs the executing command object to one of the various classes within the communication subsystem. The accepting class properly defines execute() for the specified objects.
11. MessageCommand – The constructor **MessageCommand()** creates a new object, which consists of a message object. This class properly executes the message commands by using its **execute()** method.
12. Message – The constructor **Message()** creates a new object, which consists of a destination, source and message. There are several methods defined which can access each attribute.
13. LoginCommand – The class constructor **LoginCommand()** creates a new object, which consists of the username and password. This class properly executes LoginCommand commands by using its **execute()** method.