# PURBANCHAL UNIVERSITY

**Biratnagar Nepal**

A Project report on

## "CHAT APPLICATION"

**In the partial fulfillment for the requirement of the 4th Semester Project-IV (subject code- BIT 256 CO) in the completion of Bachelor of Information Technology (BIT) degree at KIST college of Information Technology, under Purbanchal University.**


**Submitted By:-**

**1. Priya Kushawaha**

**2. Sudarshan Kapali**

**3. Suyog Bantawa Rai**

**Submitted To:-**

**Purbanchal University**

**Under The Guidance of**

**Mr. Roshan Shrestha**

**Lecturer, BIT**

**KIST COLLEGE OF INFORMATION AND TECHNOLOGY**

**KAMALPOKHARI, KATHMANDU NEPAL**

## KIST COLLEGE OF INFORMATION AND TECHNOLOGY

**KAMALPOKHARI, KATHMANDU NEPAL**


## CERTIFICATE

This is to certify that the project work entitled **"Chat Application"** is carried out by **PRIYA KUSHAWAHA (5431), SUDARSHAN KAPALI (5403)** and **SUYOG BANTAWA RAI (5401)** bonafide students of **KIST COLLEGE OF INFORMATION AND TECHNOLOGY** in partial fulfillment for the award of **BACHELOR IN INFORMATION AND TECHNOLOGY** of the **PURBANCHAL UNIVERSITY, BIRATNAGAR NEPAL**, during the year **2022-2023**. It is certified that all corrections indicated for internal assessment have been incorporated in the report submitted in the department library. The project report has been approved, as it satisfied the academic requirements in respect of the project work prescribed for the said degree.

The details of the students are as follows:-

| NAME | REGISTRATION NO. | SYMBOL NO. |
|---|---|---|
| Priya Kushawaha | 058-3-2-04730-2020 | 345498 |
| Sudarshan Kapali | 058-3-2-04746-2020 | 345512 |
| Suyog Bantawa Rai | 058-3-2-04748-2020 | 345514 |

Mr. Deepak Khadka
Program Coordinator, BIT

# KIST COLLEGE OF INFORMATION AND TECHNOLOGY KAMALPOKHARI, KATHMANDU

**Examiner's Certification**

The Project Report
On

**"Chat Application"**

**Developed by**

**Priya Kushawaha**
**Sudarshan Kapali**
**Suyog Bantawa Rai**

Is approved and is acceptable in qualified form.

| | |
|---|---|
| **Internal Examiner** | **External Examiner** |
| Name: | Name: |
| Designation: | Designation: |

# ACKNOWLEDGEMENT

It is with greatest satisfaction and euphoria that we are submitting our project report entitled **"Chat Application".** We have completed it as a part of the curriculum of **PURBANCHAL UNIVERSITY.**

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude to our **PROJECT MANAGER Mr.Roshan Shrestha** who guided us throughout the project. We would also like to thank our friends and family who continuously supported, motivated us and offered deep insight into the study.
We are indebted to all members of **KIST College,** for the valuable support and suggestions provided by them using their specific fields' knowledge. We are grateful for their cooperation during the period of our project..

Finally, we would also like to express our gratitude to **PURBANCHAL UNIVERSITY** who gave us the beautiful opportunity to explore on this wonderful project. This project helped us in doing a lot of research and we came to know about so many new things and we are really thankful.

We hope the university will accept our attempt as a successful project.

Thank you!

**PRIYA KUSHAWAHA (345498)**
**SUDHARSHAN KAPALI (345512)**
**SUYOG BANTAWA RAI (345514)**

# STUDENT'S DECLARATION

We hereby declare that the project report entitled **"Chat Application"** is based on our own work carried out during the course of our study under the supervision of **Mr. Roshan Shrestha** sir**.** We assert the statements made and conclusions drawn are an outcome of our research work.

Furthermore, we certify that this project submitted is our original work and has never been submitted in any institution for any other titles or awards.

| S.N. | Name | Registration No. | Symbol No. |
|---|---|---|---|
| 1. | Priya Kushawaha | 058-3-2-04730-2020 | 345498 |
| 2. | Sudharshan Kapali | 058-3-2-04746-2020 | 345512 |
| 3 | Suyog Bantawa Rai | 058-3-2-04748-2020 | 345514 |

# TO WHOM IT MAY CONCERN

This is to certify that Ms. Priya Kushawaha, Mr. Sudarshan Kapali and Suyog Bantawa Rai of **Bachelor in Information Technology** (**BIT**) have studied as per the curriculum of BIT 4th Semester and completed the project entitled **"Chat Application"**. This project is the original work of Ms. Priya Kushawaha, Mr. Sudarshan Kapali and Mr. Suyog Bantawa Rai and was carried out under the supervision of Mr. **Roshan Shrestha** as per the guidelines provided by Purbanchal University and certified as per the student's declaration that project **"Chat Application"** has not been presented anywhere as a part of any other academic work.

The detail of the student is as follows:

| | |
|---|---|
| Name of Students | : Priya Kushawaha |
| | : Sudarshan Kapali |
| | : Suyog Bantawa Rai |
| Semester | : 4th |
| Subject Code | : BIT 256CO |
| Project Title | : **Chat Application** |

**.....**
Mr. Roshan Shrestha
Project Supervisor
KIST College of Information Technology

# Table Of Contents

## ABSTRACT

This report presents an overview and analysis of a cutting-edge chat application designed to meet the communication needs of today's digital society. With the proliferation of smartphones and the increasing reliance on instant messaging, chat applications have become pivotal tools for connecting individuals, businesses, and communities worldwide.

The report commences by exploring the fundamental features that define this chat application, including user registration and authentication, one-on-one messaging capabilities. It underscores the significance of robust security measures, including end-to-end encryption, ensuring the privacy and data integrity of users.

Furthermore, the report examines the impact of chat applications on social dynamics, emphasizing their role in fostering virtual communities and enhancing collaboration through the inclusion of chatbots and automation features. Case studies showcase real-world applications across various sectors, from personal communication to business collaborations, underscoring the versatility and adaptability of this chat application.

As technology continues to evolve, the report highlights emerging trends and innovative features, such as voice and video communication, AI-driven enhancements, and cross-platform compatibility. These developments not only improve user experience but also shape the future of digital communication.

In conclusion, this report underscores the pivotal role of chat applications in the digital age, emphasizing their potential to revolutionize how people connect and collaborate. It highlights the need for continual innovation to meet evolving user expectations and addresses the challenges and opportunities inherent in the dynamic landscape of digital communication. This chat application represents a significant step toward fostering meaningful connections in an increasingly interconnected world.

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

We found that we were good at coding and had the ability to reason about the code very well and it didn't take long for us to realize that this is what we wanted to do for the rest of our life. Though when we began college we still wanted to go into programming, as the state of web applications became more interesting in the last few days, we got into that and that's what we are doing now! We really couldn't imagine doing anything else, and programming is definitely one of the top skills too.

## 1.2 Programming Language

A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

## 1.3 JAVA Language

Java is a widely used, platform-independent, and object-oriented programming language. It's known for its robustness, security, and support for multithreading. Java applications run on a Java Virtual Machine (JVM), enabling cross-platform compatibility. It offers a rich standard library, automatic memory management, and a vibrant developer community. Java is popular in enterprise software and is used for web, mobile, and desktop application development.

## 1.4 Advantages of Java language

- Platform Independence
- Object-Oriented
- Robust and Secure
- Multithreading
- Rich Standard Library
- Garbage Collection
- Multithreading
- Security
- Cross-Platform Development
- Long-Term Support

## 1.5 Limitations of Java Language

- Performance Overhead
- Memory Consumption
- No Multiple Inheritance
- Complexity
- Lack of Operator Overloading

### 1.6 Project Description

The main aim of designing and developing this Internet Chat Application primarily based Engineering project is to provide secure and efficient methods for chatting with friends and family members using technology. Java language is used to develop this project. Users will have all options and features in the application like registration, login and logout.

### 1.7 Introduction to Chat Application

The Chat Application project has a central objective: to revolutionize modern communication. In today's digital age, effective and secure communication is paramount, be it in personal relationships or professional collaborations. The project seeks to fulfill this need by developing a versatile and user-friendly platform that empowers individuals and organizations to connect seamlessly and efficiently.

At its core, this project prioritizes user experience and security. It aims to offer a feature-rich communication experience, encompassing one-on-one messaging catering to diverse communication needs. Furthermore, it places a strong emphasis on privacy and data security, implementing robust encryption and authentication measures to ensure that user data remains protected. By combining these elements with cross-platform compatibility and a commitment to innovation, the Chat Application project strives to redefine the way people communicate in our increasingly interconnected world, setting new standards in instant messaging.

### 1.8 Advantages of Chat Application

- **Instant Communication:** This chat application enables instant messaging, allowing users to send and receive messages in real-time, making it ideal for quick conversations and responses.
- **Cost-Effective:** This chat application offers free messaging and calls over the internet, reducing the need for traditional SMS or international calling plans.
- **Group Chats:** Group chat features enable multiple users to participate in a single conversation, making it easy for teams, friends, or family members to communicate and collaborate together.
- **Searchable Conversations:** This chat application offers a search feature, allowing users to quickly find and reference past conversations, which is especially valuable for work-related discussions.
- **User-Friendly:** This chat application is typically designed with a user-friendly interface, making them accessible and easy to use for people of all ages.
- **Customization:** Users can customize their profiles, notification settings, and chat backgrounds, enhancing the user experience.

### 1.9 Disadvantages of Chat Application

- **Security Concerns:** This chat application can be susceptible to security breaches, hacking, and data leaks, especially if proper security measures like end-to-end encryption are not in place.
- **Privacy Issues:** Users may have concerns about their personal information and messages being accessed or shared without their consent, leading to privacy violations.
- **Spam and Unsolicited Messages:** This Chat application can be targets for spam, unwanted messages, and phishing attempts, which can disrupt communication and compromise security.
- **Device and Platform Dependency:** Users may be limited to specific devices or platforms, making it challenging to communicate with those using different apps or systems.
- **Data Retention and Storage:** Storing large volumes of chat data can consume device storage space and pose challenges for data management.

### 1.10 Problem Statement

- **Fragmented User Experience:** Many chat applications lack a unified and intuitive user interface, leading to fragmented user experience across devices and platforms. Users often struggle with inconsistent designs, making it difficult to seamlessly transition between mobile devices, desktop, computers, and web interfaces.

- **Privacy and Security Concerns:** Privacy breaches and security vulnerabilities have become increasingly prevalent, eroding user trust. Users require a chat application that prioritizes robust security measures, including end-to-end encryption and stringent authentication protocols, to safeguard their personal information and communication.

- **Limited Feature Sets:** Current chat applications, while functional, often lack a comprehensive suite of features to cater to diverse communication needs. Users seek a platform that not only offers text-based messaging but also supports multimedia sharing, voice and video calls, collaborative tools, and seamless integration with third-party services.

- **Ineffective Content Moderation:** Maintaining a safe and respectful online environment is a significant challenge. Existing chat applications struggle to effectively moderate content, leading to issues such as harassment, hate speech, and the dissemination of inappropriate content. A new chap application should incorporate advanced content moderation mechanisms to tackle these issues proactively.

- **Innovation and Adaptation:** As technology evolves, users expect their communication tools to keep pace. To remain relevant, a chat application should continually innovate, exploring emerging technologies like AI, automation, and augmented reality to enhance user experience and efficiency.

### 1.11 Objectives

The main objective of this project is to make it easier for users to chat with their family members and friends easily. Some of the objectives of this project are as follows:

- **Feature-rich Communication:** The project aims to develop a chat application that encompasses a wide range of features, from one-on-one messaging to group chats, multimedia sharing, voice and video calls, and beyond. It aspires to provide users with a comprehensive suite of tools to cater to their diverse communication needs.
- **User-Centric Design:** User experience is a paramount concern. The application will be designed with an intuitive interface, customization options, and accessibility features, ensuring that users of all backgrounds and abilities can engage effortlessly.
- **Privacy and Security:** In an era marked by data breaches and privacy concerns, the project places a strong emphasis on security. Implementing robust encryption measures and authentication protocols will safeguard user data and maintain their privacy.
- **Scalability and Cross-form Compatibility:** The project will be architected to accommodate future growth and technological advancements. Cross-platform compatibility will ensure that users can seamlessly transition between devices while preserving their communication history.
- **Innovation and Adaptability:** As the digital landscape evolves, the project will remain at the forefront of innovation. It will continually explore emerging technologies such as artificial intelligence and automation to enhance user experience and communication efficiency.

### 1.12 Scope

The Chat Application project encompasses the complete lifecycle of developing, testing, deploying, and maintaining a robust instant messaging platform. This includes the following scopes:

- **Research and Analysis:** Conducting market research and competitor analysis to identify user needs and gaps in existing solutions.
- **Development:** Building the application from the ground up, focusing on security, scalability, and user experience.
- **Testing:** Rigorous testing to ensure the application functions flawlessly across a variety of devices and scenarios.
- **Deployment:** Making the application available to users through app stores and web platforms.
- **Maintenance and Update:** Ongoing support, bug fixes, and feature enhancements to keep the application relevant and secure.

## CHAPTER 2

## MATERIALS AND METHOD

### 2.1 Header files

| | |
|---|---|
| import javax.swing.*; | The ‘import javax.swing.*;’ is used to include Swing GUI components and functionality in a Java program. |
| import java.awt.*; | The ‘import java.awt.*;’ is used to include the AWT components and functionality in a Java program for creating GUIs. |
| import java.awt.event.ActionEvent; | The ‘import java.awt.event.ActionEvent;’ is used in Java to represent an action event. |
| import java.awt.event.ActionListener; | The ‘import java.awt.event.ActionListener;’ is used to listen for and handle user-initiated events. |
| import java.awt.event.KeyEvent; | The ‘import java.awt.event.KeyEvent;’ is used in Java GUI programming to handle keyboard events. |
| import java.io.*; | The ‘import java.io.*;’ is used to access classes and functionality related to input and output operations in Java. |
| import java.net.Socket; | The ‘import java.net.Socket;’is used to establish a network socket connection in Java. |
| import java.net.ServerSocket; | The ‘import java.net.ServerSocket;’ is used to import the ServerSocket class from the Java Networking package. |
| import java.io.DataInputStream; | The ‘import java.io.DataInputStream;’ is used for reading binary data from an input stream. |
| import java.io.DataOutputStream; | The ‘import java.io.DataOutputStream;’ is used for writing binary data to an output stream. |
| import java.io.IOException; | The ‘import java.io.IOException;’ is used to handle input/output-related errors and exceptions. |
| import java.sql.*; | The ‘import java.sql.*;’ allows access to classes and interfaces from the Java SQL package. |
| import java.util.HashMap; | The ‘import java.util.HashMap;’ is used to import the HashMap class from the Java Util package. |
| import java.util.Map; | The ‘import java.util.Map;’ is used to import the Map interface from the Java Util package. |
| import | The ‘import java.awt.event.MouseAdapter;’ is used to |

| | |
|---|---|
| java.awt.event.MouseAdapter; | import the MouseAdapter class from the AWT package. |
| import java.awt.event.MouseEvent; | The ‘import java.awt.event.MouseEvent;’ is used to import the MouseEvent class from the AWT package. |
| import java.util.ArrayList; | The ‘import java.util.ArrayList;’ is used to import the ArrayList class from the Java Util package. |
| import java.util.List; | The ‘import java.util.List;’ is used to import the List interface from the Java Util package. |
| import java.sql.Connection; | The ‘import java.sql.Connection;’ is used to import the Connection interface from the Java SQL package. |
| import java.sql.SQLException; | The ‘import java.sql.SQLException;’ is used to import the SQLException class from the Java SQL package. |
| import java.sql.DriverManager; | The ‘import java.sql.DriverManager;’ is used to import the DriverManager class from the Java SQL package. |
| import java.sql.Statement; | The ‘import java.sql.Statement;’ is used to import the Statement interface from the Java SQL package. |

**Table 2.1 Header files**

### 2.2 Functions used

| | |
|---|---|
| render() | Sets up and renders the user interface for the login screen. |
| actionPerformed(ActionEvent e) | This is an event handler method that responds to actions triggered by buttons. |
| messageBox(String message, String title) | This method displays a message dialog box with the specified message and title using JOptionPane. |
| initializeApp(String LoginName) | Sets up the main chat application interface, including navigation bars, menus, panels, buttons, and event listeners. |
| addFriendFrame(String owner, String friendNameField, String friendID) | Creates a separate frame for adding a friend, including labels for the friend's name and ID and an "Add" button. |
| getConversationTableName(String clientName, String recipient) | Constructs a table name for storing chat messages between two users based on their names. |
| createMessageFrame(String recipient, String loginName) | Creates a chat message frame for communicating with a friend, including panels for displaying messages, a text field for input, and a "Send" button. |

| | |
|---|---|
| sendTextToServer(String msg) | Sends a message to the server using a DataOutputStream. |
| run() | This is the run method of the Runnable interface, which is executed when a new thread is started. |
| createNameTable(String name) | This method is used to create a new table in the MySQL database with the given name. |
| main(String[] args) | This is the main method of the Server class. |
| getFriends(String name) | Retrieves a list of friends for a given user by querying the database based on the user's name |
| addFriend(String tableName, String friendName, String SID) | Adds a friend to a user's friend list by inserting their information into the specified table in the database. |
| getMyFriends(String LoginName) | Retrieves the list of friends for a given user by querying the database using the user's name. |

**Table 2.2 Functions**

# CHAPTER 3

# SYSTEM DESIGN

## 3.1 Working Principle

The working principle of a chat application involves several key components and processes that enable real-time communication between users. Here's an overview of the fundamental principles:

- **Client-Server Architecture:** Chat applications typically follow a client-server architecture. The application's core components are divided into client-side and server-side elements.
- **User Authentication:** Users need to create accounts or log in securely to use the chat application. User authentication ensures that only authorized users can access the application.
- **Message Sending and Receiving:**
    - **Sending Messages:** When a user composes and sends a message, the client formats it and sends it to the server. The server receives the message and processes it.
    - **Receiving Messages:** Clients continuously listen for incoming messages from the server. When a new message arrives, the server pushes it to the recipient's client(s) in real-time.
- **Real-Time Communication:**
    - **WebSocket or Long Polling:** Chat applications often use technologies like WebSockets or long polling to establish persistent connections between clients and the server. This allows real-time message delivery without the need for constant polling by the client.
- **Message Storage:** Chat applications usually store message history to ensure that users can access past conversations. Messages are often stored in databases on the server, with appropriate indexing and organization.
- **User Interface:** The user interface of the chat application displays messages, user lists, and other features. It provides an intuitive and user-friendly experience for composing, sending, and receiving messages.

## 3.2 Algorithm

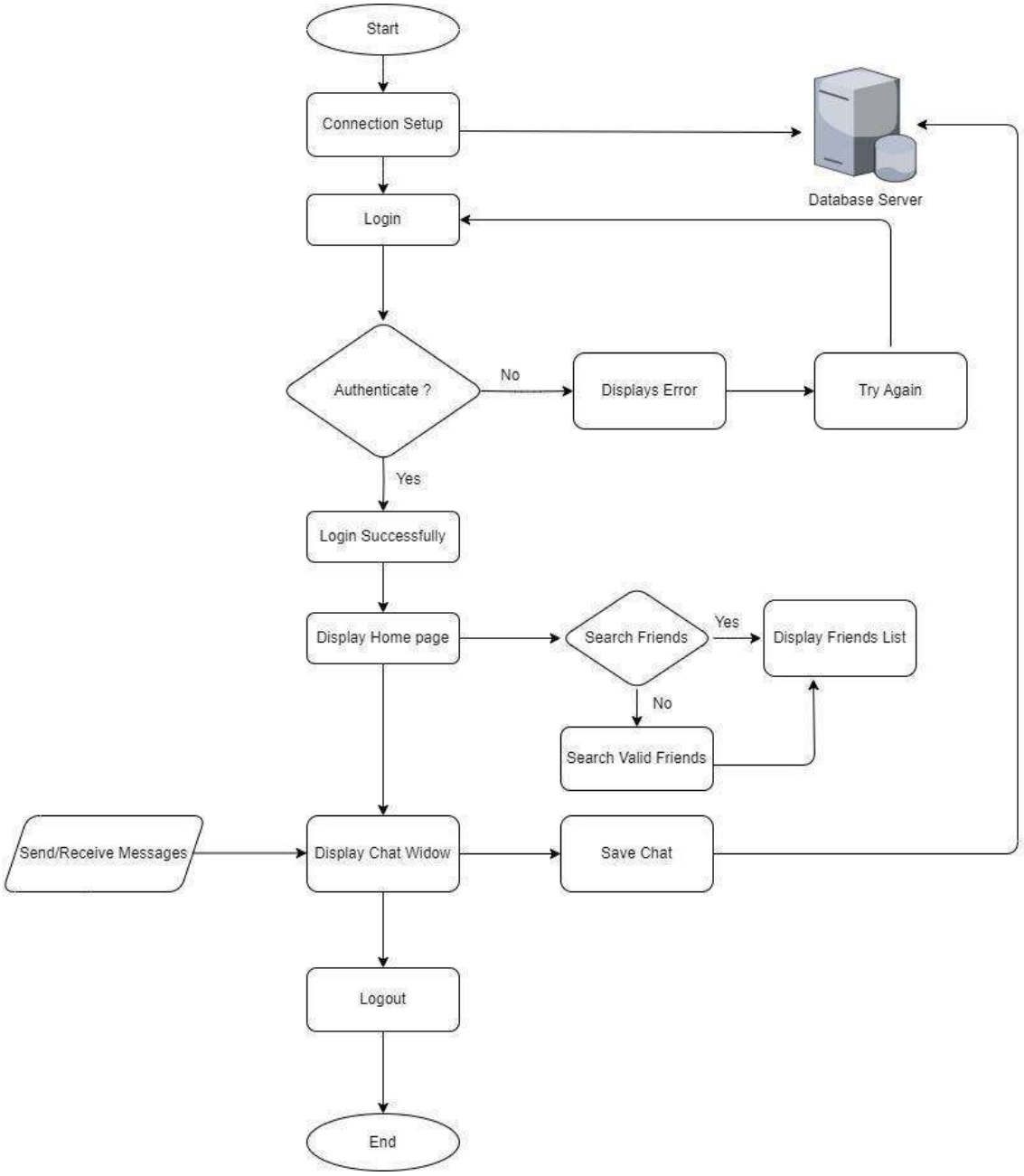
Step1: Start
Step2: Set up the Connection
Step3: Start the Code
- ➢ The home page will display

Step4: Login or Register
- ➢ For Registration
    - Enter Username, email and password
    - Submit
- ➢ For Login into the system
    - Enter username and password
    - Submit

Step5: Login Successful
Step6: Search the friends list
Step7: Add  friend and Refresh
Step8: Send/Receive Messages
Step9: Logout
Step10: End

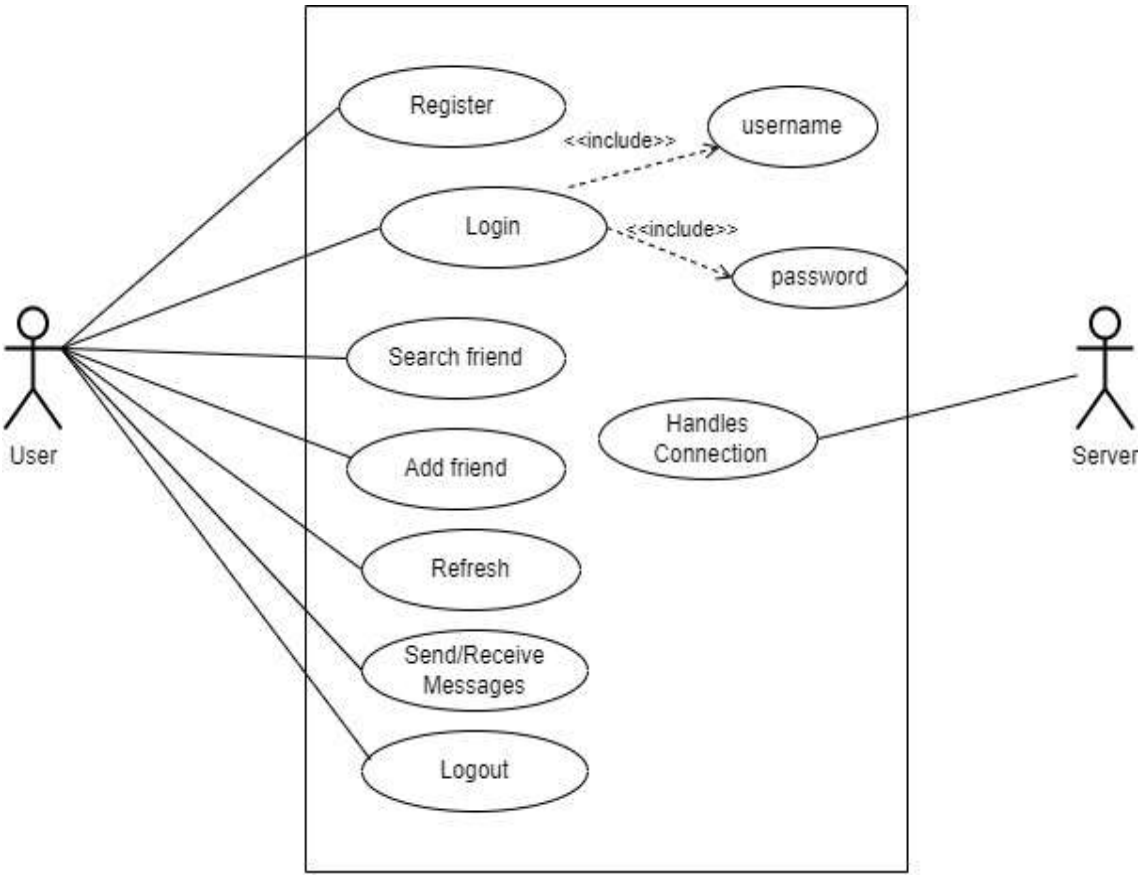
### 3.3 Flowchart

Flow chart is a diagram that represents workflow or process.



### 3.4 Functional Requirement

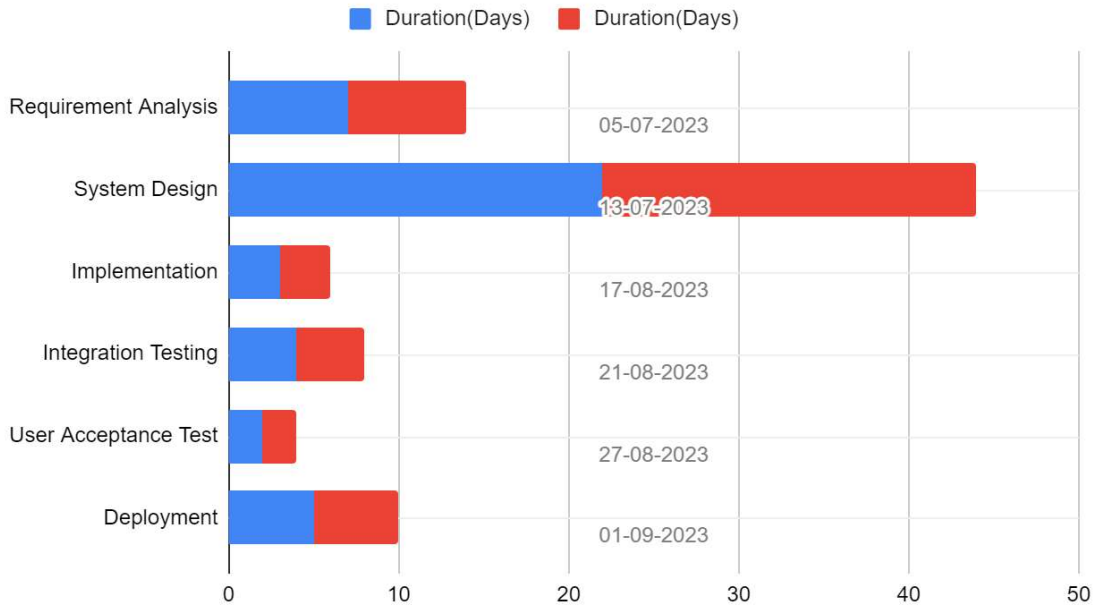
In software and system engineering, a functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between input and outputs.




**Use Case Diagram**

### 3.5 Gantt Chart

| Task | Start Date | End Date | Duration(Days) |
|---|---|---|---|
| **Requirement Analysis** | 05-07-2023 | 12-07-2023 | 7 |
| **System Design** | 13-07-2023 | 11-08-2023 | 22 |
| **Implementation** | 17-08-2023 | 21-08-2023 | 3 |
| **Integration Testing** | 21-08-2023 | 25-08-2023 | 4 |
| **User Acceptance Test** | 27-08-2023 | 29-08-2023 | 2 |
| **Deployment** | 01-09-2023 | 06-09-2023 | 5 |




**Gantt Chart**

# CHAPTER 4

# REQUIREMENT ANALYSIS & IMPLEMENTATION SYSTEM

## 4.1 Requirements

**Hardware Requirement**

- Memory (RAM): 6.00GB
- System Type: 64-bit OS, x-64 based processor
- Storage Capacity: 30 GB HDD
- CPU: 2.50 GHz

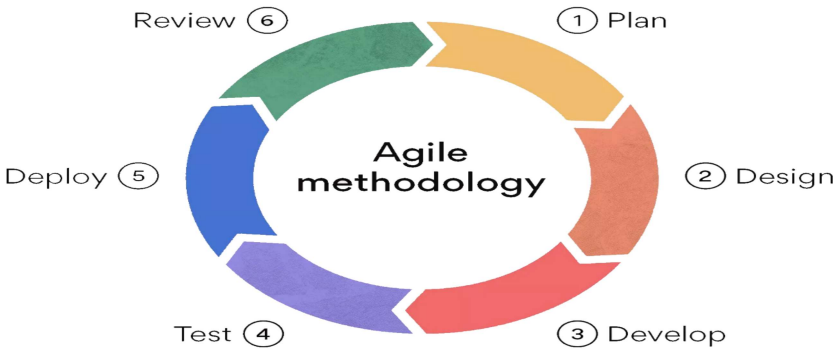
**Software Requirement**

- Operating System: Windows 7 or Higher
- Development Tools: Java 19 Intellij IDEA, or any IDE that supports Java

## 4.2 System Methodology

System Methodology is a methodology for systematically organizing the best ways to develop systems efficiently. It is a step-by-step process for developing any system. There are many system development methodologies. Some of them are Waterfall Model, Iterative Model, Develop Model, V-Model, Spiral Model, Lean and Agile Model, Prototype Model, etc.

In this project, we are going to use the Agile Development Methodology. Agile development methodology is an iterative and flexible approach to software development that emphasizes collaboration, adaptability, and customer satisfaction. It contrasts with traditional waterfall methods, which follow a linear, sequential approach. Agile is characterized by its principles and practices, which enable teams to respond to changing requirements and deliver high-quality software in shorter cycles.



I. **Plan:** In the planning phase, the Agile team collaborates with stakeholders to define project objectives and requirements. This involves creating a product backlog, which is a prioritized list of features, user stories, or tasks. The team discusses and estimates the effort required for each backlog item and decides what to include in the upcoming development cycle, known as a sprint or iteration.

II. **Design:** Design in Agile focuses on creating a blueprint for how the software will look and function. This includes detailing user interfaces, workflows, and architectural considerations. Agile teams often use techniques like user stories, wireframes, and mock-ups to visualize the design. Design activities may occur before or during development, depending on the Agile framework being used.

III. **Develop:** During the development phase, the team works on implementing the features and functionality defined in the design and selected for the sprint. Development is typically broken down into smaller tasks, often referred to as user stories or work items, to allow for incremental progress. Continuous integration and collaboration within the development team are essential to ensure that code is integrated smoothly.

IV. **Test:** Testing is an integral part of Agile and runs in parallel with development. Testers validate that each feature or user story meets acceptance criteria and is free of defects. Automated testing and test-driven development (TDD) practices may be employed to maintain code quality and identify issues early. The goal is to ensure that the software functions correctly and aligns with user expectations.

V. **Deploy:** Deployment involves releasing the developed features to a live or staging environment where end-users can access them. Agile promotes frequent and incremental deployments, allowing users to benefit from new features and improvements as soon as they are ready. Continuous integration and deployment (CI/CD) pipelines may be used to automate the deployment process, ensuring efficiency and reliability.

VI. **Review:** The review phase occurs after deployment and focuses on gathering feedback from users and stakeholders. Agile teams hold regular review meetings, such as sprint reviews or demos, where they showcase the completed work. Feedback is crucial for assessing whether the project is meeting its goals and satisfying user needs. Based on this feedback, the team may update the backlog, reprioritize items, and plan the next sprint or iteration. Continuous improvement is a core principle of Agile, and the review phase plays a central role in this iterative process.

In summary, Agile methodology emphasizes iterative development, collaboration, and responsiveness to change. These six steps represent a continuous cycle that allows Agile teams to adapt to evolving requirements, deliver incremental value, and maintain a high level of customer satisfaction throughout the software development process.

# CHAPTER 5

# FUTURE SCOPE AND CONCLUSION

## 5.1 Future Scope

I. **Artificial Intelligence and Chatbots:** Chat applications will increasingly incorporate AI-driven chatbots to provide automated responses, assist users with tasks, and enhance customer support. These chatbots will become more sophisticated, understanding natural language and context better.

II. **Voice and Video Integration:** The integration of high-quality voice and video calling within chat applications will continue to expand. Expect more reliable and seamless voice and video conversations features for enhanced visual experiences.

III. **Multi-Platform Accessibility**: Chat apps will become even more accessible across different devices and platforms. Users will expect a consistent experience whether they're on a smartphone, tablet, desktop, or smartwatch.

IV. **Enhanced Privacy and Security:** Given the growing concerns about data privacy and security, chat apps will place a stronger emphasis on end-to-end encryption, two-factor authentication, and user-controlled data sharing.

V. **Integration with E-commerce**: Chat applications will increasingly integrate with e-commerce platforms, enabling users to shop, make payments, and receive customer support directly within the app.

## 5.2 Conclusion

In conclusion, our chat application represents not just a tool for communication but a catalyst for connection and collaboration in the digital age. Through meticulous planning and development, we have crafted a solution that addresses the challenges of fragmented user experiences, security concerns, and limited feature sets.

With a user-centric approach, we've prioritized simplicity, customization, and privacy to ensure that our chat application seamlessly integrates into the lives of our users. Features like multimedia messaging, group chats, and robust content moderation empower users to communicate effectively and safely.

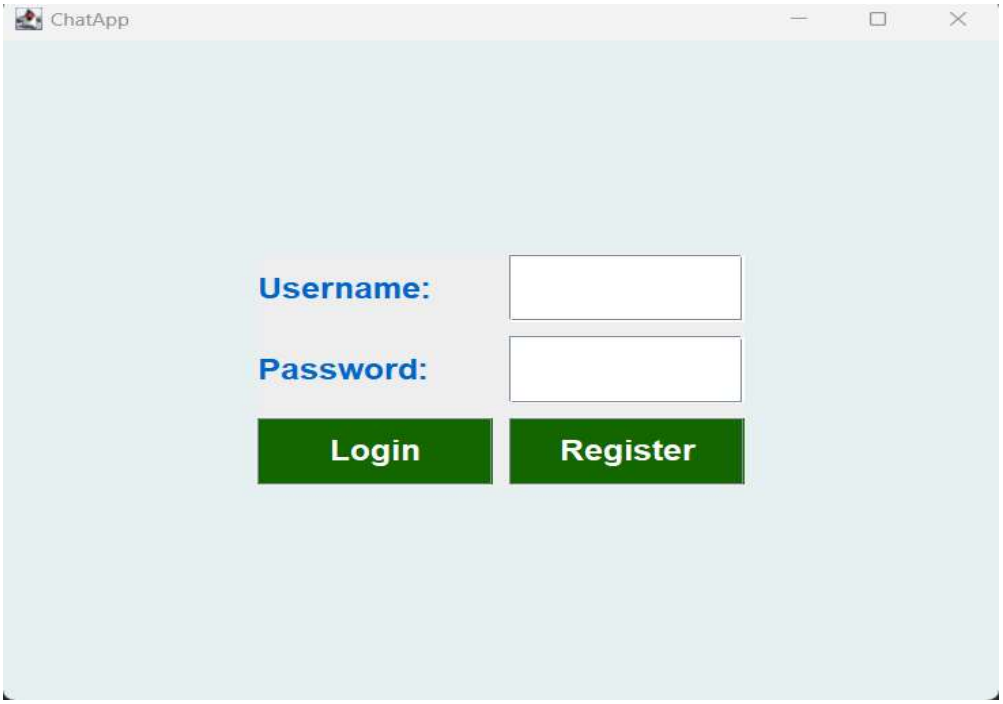
As we move forward, we remain committed to innovation, continually adapting to emerging technologies and user needs. Our vision is to provide a platform that not only bridges distances but also fosters meaningful connections. We invite you to join us on this journey, as together, we revolutionize the way people communicate and connect in the modern world. Thank you for your attention, and we look forward to your feedback and support.

# CHAPTER 6

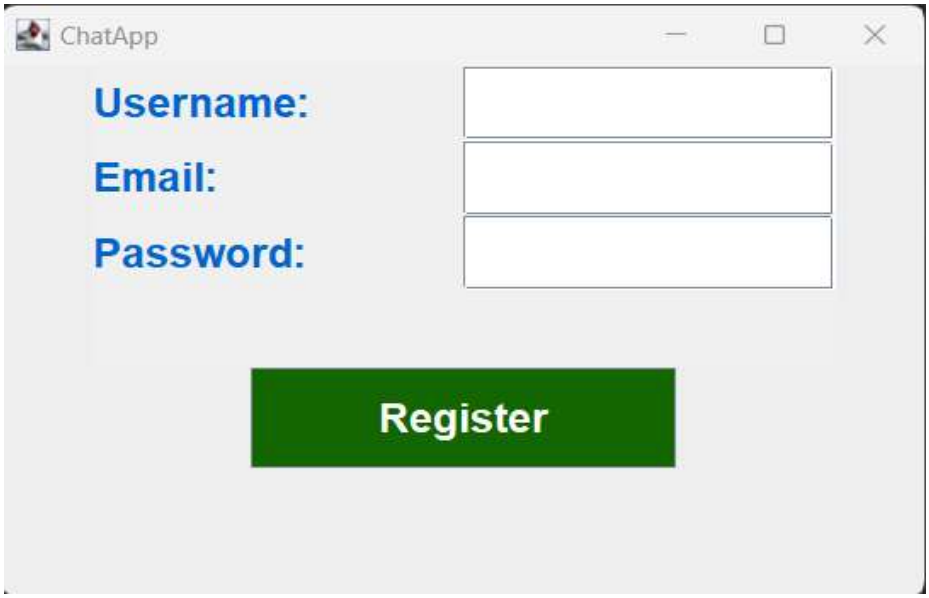
## Appendix

### 6.1 Screenshots

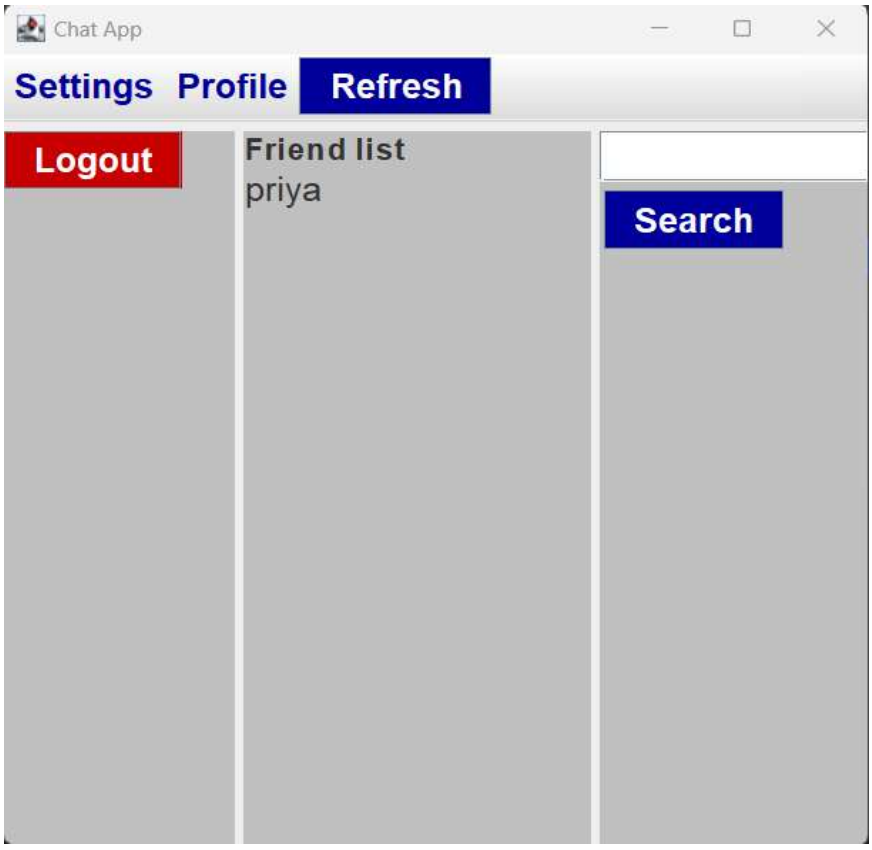
**Login:**

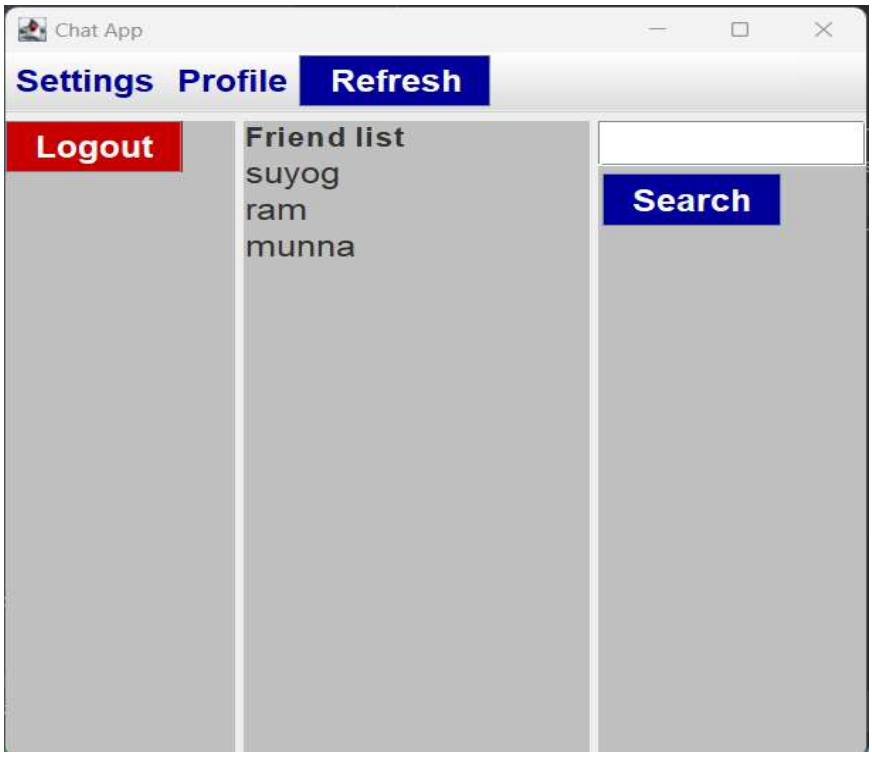



**Register:**

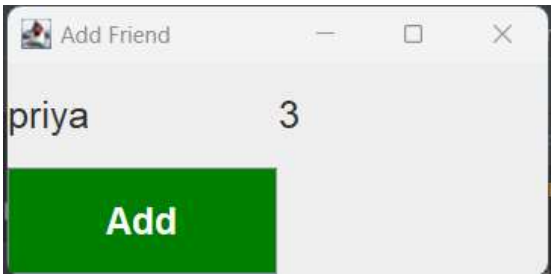


**Home Page:**

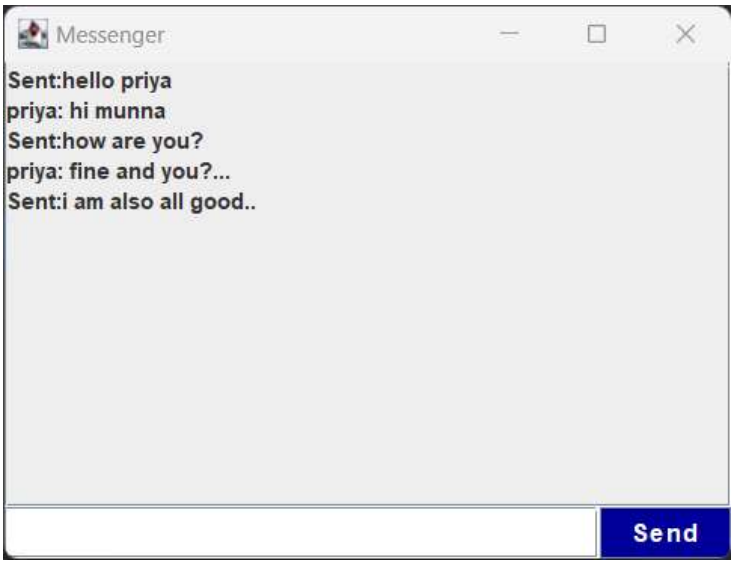

Chat App
Settings
Profile
Refresh
Logout
Friend list
priya
Search

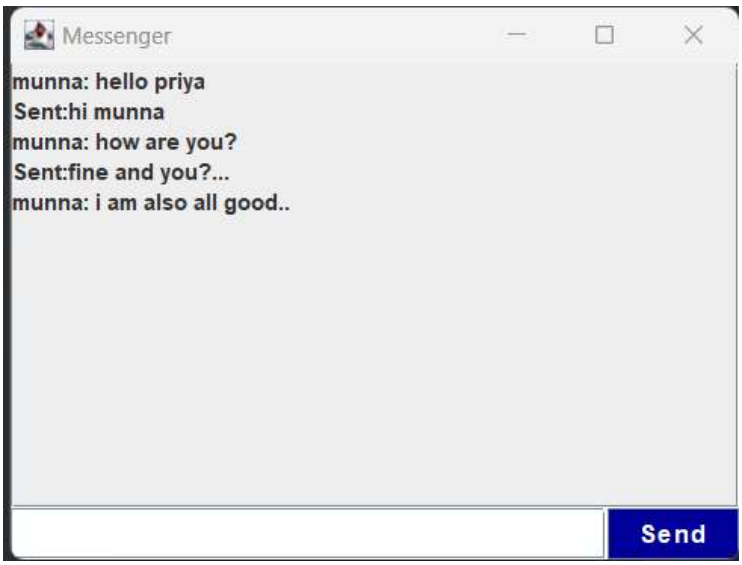

Chat App
Settings
Profile
Refresh
Logout
Friend list
suyog
ram
munna
Search

**Add friend:**




**Chat:**



### 6.2 Source Code

```
import javax.swing.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Server {
   UserInfo info = new UserInfo();
   private static List<String> clients = new ArrayList<>();
   private static Map<String, Socket> clientMap = new HashMap<>();

   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(12345); // Choose a port
         System.out.println("Server started. Waiting for clients...");

         Runnable loginServer = new LoginServer();
         Thread loginThread = new Thread(loginServer);
         loginThread.start();

         Runnable registerServer = new RegisterServer();
         Thread registerThread = new Thread(registerServer);
         registerThread.start();

         while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " + clientSocket.getInetAddress());
            DataInputStream inputStream = new
DataInputStream(clientSocket.getInputStream());
            String clientName = inputStream.readUTF();
            System.out.println(clientName);
            clients.add(clientName);
            clientMap.put(clientName, clientSocket);

            ClientHandler clientHandler = new ClientHandler(clientSocket, clientName);
            Thread clientThread = new Thread(clientHandler);
```

```
            clientThread.start();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public ResultSet getFriends(String name) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        String url = "jdbc:mysql://localhost/java_db";
        Connection conn = DriverManager.getConnection(url, info.userNameDB,
info.passwordDB);
        System.out.println("Connected to the database");
        Statement stm = conn.createStatement();
        ResultSet rs = stm.executeQuery("select * from Login where
UserName='"+name+"'");
        return rs;
    } catch (ClassNotFoundException | SQLException ex) {
        ex.printStackTrace();
        ResultSet rs=null;
        return rs;
    }
}
public void addFriend(String tableName,String friendName,String SID){
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        String url = "jdbc:mysql://localhost/java_db";
        Connection conn = DriverManager.getConnection(url, info.userNameDB,
info.passwordDB);
        System.out.println("Connected to the database");
        // Insert the message into the conversation table
        String insertQuery = "INSERT INTO " + tableName + " (ID, userName) VALUES (?,
?)";
        PreparedStatement preparedStatement = conn.prepareStatement(insertQuery);
        int id=Integer.parseInt(SID);
        preparedStatement.setInt(1, id);
        preparedStatement.setString(2, friendName);
        preparedStatement.executeUpdate();

        conn.close();
    } catch (ClassNotFoundException | SQLException ex) {
        ex.printStackTrace();
    }
```

```
    }
    public ResultSet getMyFriends(String LoginName){
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url = "jdbc:mysql://localhost/java_db";
            Connection conn = DriverManager.getConnection(url, info.userNameDB,
info.passwordDB);
            System.out.println("Connected to the database");
            Statement stm = conn.createStatement();
            ResultSet rs = stm.executeQuery("select * from "+LoginName);
            return rs;
        } catch (ClassNotFoundException | SQLException ex) {
            ex.printStackTrace();
            ResultSet rs=null;
            return rs;
        }
    }

    private static class ClientHandler implements Runnable {
        private Socket clientSocket;
        private String clientName;
        private DataInputStream inputStream;

        public ClientHandler(Socket clientSocket, String clientName) {
            this.clientSocket = clientSocket;
            this.clientName = clientName;
            try {
                inputStream = new DataInputStream(clientSocket.getInputStream());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void run() {
            try {
                while (true) {
                    String receivedText = inputStream.readUTF();
                    System.out.println("Received from " + clientName + ": " + receivedText);

                    // Broadcast the received text to all other connected clients
                    broadcastText(receivedText);
                }
            } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }

    private void broadcastText(String text) {
        String[] parts = text.split(":", 2); // Split the message into recipient and message text
        String recipientName = null;
        if (parts.length == 2) {
            recipientName = parts[0].trim();
            String messageText = parts[1].trim();

            Socket recipientSocket = clientMap.get(recipientName);
            if (recipientSocket != null) {
                try {
                    DataOutputStream recipientOutputStream = new
DataOutputStream(recipientSocket.getOutputStream());
                    recipientOutputStream.writeUTF(clientName + ": " + messageText);
                    recipientOutputStream.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        String msg=parts[1].trim();
        // Save the message in the conversation table or create one if it doesn't exist
        saveMessageToConversation(clientName, recipientName, msg);
    }

    private void saveMessageToConversation(String sender, String recipient, String
message) {
        UserInfo info = new UserInfo();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url = "jdbc:mysql://localhost/java_db";
            Connection conn = DriverManager.getConnection(url, info.userNameDB,
info.passwordDB);

            // Combine sender and recipient names to create a unique table name
            String tableName = sender.compareTo(recipient) < 0 ? sender + "_" + recipient :
recipient + "_" + sender;

            // Check if the conversation table exists, if not, create one
            if (!doesTableExist(conn, tableName)) {
```

```
                createConversationTable(conn, tableName);
            }

            // Insert the message into the conversation table
            String insertQuery = "INSERT INTO " + tableName + " (Sender, Message)
VALUES (?, ?)";
            PreparedStatement preparedStatement = conn.prepareStatement(insertQuery);
            preparedStatement.setString(1, sender);
            preparedStatement.setString(2, message);
            preparedStatement.executeUpdate();

            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            ex.printStackTrace();
        }
    }

    private boolean doesTableExist(Connection conn, String tableName) throws
SQLException {
        DatabaseMetaData metaData = conn.getMetaData();
        ResultSet tables = metaData.getTables(null, null, tableName, null);
        return tables.next();
    }

    private void createConversationTable(Connection conn, String tableName) throws
SQLException {
        Statement stmt = conn.createStatement();
        String createTableQuery = "CREATE TABLE " + tableName + " (ID INT
AUTO_INCREMENT PRIMARY KEY, Sender VARCHAR(255), Message TEXT)";
        stmt.executeUpdate(createTableQuery);
    }
  }
}
import javax.swing.*;

public class Dummy1 {

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
      Login obj = new Login();
      obj.render();
    });
  }
```

## References

https://draw.io/
https://lucid.app/
https://www.javatpoint.com/java-tutorial
https://www.w3schools.com/java/
https://www.freecodecamp.org/news/tag/java/