Whisp: Real Time Java Chat with GUI

A MINI PROJECT REPORT

18CSC302J COMPUTER NETWORKS

Submitted by

Shaurya Singh Srinet (RA2111032010006)

SCHOOL OF COMPUTING

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

(WITH SPECIALIZATION IN INTERNET OF THINGS)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(KATTANKULATHUR CAMPUS)

CHENNAI 603203

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this mini project report "**Whisp: Real Time Java Chat with GUI**" is the bonafide work of "**Shaurya Singh Srinet (RA2111032010006)"** who carried out the project work for **18CSC302J – Computer Networks** under my supervision at **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2023 – 2024.

SIGNATURE

Faculty In Charge
Dr. R. PRABHU
Assistant Professor
Department of Networking and Communications
SRM Institute of Science and Technology

SIGNATURE

HEAD OF THE DEPARTMENT
Dr. Annapurani Panaiyappan. K
Professor and Head,
Department of Networking and Communications
SRM Institute of Science and Technology

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The development of ChatWave, a realtime chat application using web sockets, was motivated by several key factors:

- **RealTime Communication:** In an increasingly connected world, the need for realtime communication tools has grown. ChatWave aims to provide a simple and efficient platform for users to engage in immediate conversations.
- **Learning and Research:** As part of our studies in the Fundamentals of Networks subject at the University of Granada, we recognized the educational value of building a chat application. It allowed us to gain practical experience with socket programming and network fundamentals.
- **OpenSource Collaboration:** We wanted to create an opensource project that could be improved by the community and serve as a learning resource for other developers interested in socketbased applications.

1.2 SCOPE

The scope of the ChatWave project encompasses the following aspects:

- **RealTime Chat:** ChatWave provides a realtime chat platform where users can send and receive messages instantly.
- **Graphical User Interface (GUI):** The application includes an intuitive GUI, making it accessible and userfriendly.
- **Socket Programming:** The project leverages socket programming to enable communication between the server and clients.
- **Customizable Usernames:** Users have the option to set custom usernames, adding a personal touch to their interactions.
- **Server Configuration:** The server component allows for the definition of the IP address and port to which clients can connect.

1.3 OBJECTIVES

Primary Objectives:

- **RealTime Communication:** To develop a chat application that enables users to engage in realtime conversations.
- **UserFriendly Interface:** To provide a userfriendly GUI that simplifies the chat experience.
- **Socket Programming Proficiency:** To gain expertise in socket programming and network communication.
- **Customizable Usernames:** To allow users to set unique usernames for their chat sessions.
- **Open Source Collaboration:** To create an opensource project that fosters collaboration and learning within the developer community.

Secondary Objectives:

- **Scalability:** Explore opportunities to enhance the scalability of the chat application to accommodate more users.
- **Security:** Implement additional security features to protect user data and privacy.
- **CrossPlatform Compatibility:** Investigate the potential for running ChatWave on various operating systems.
- **Advanced Features:** Consider the addition of advanced features such as file sharing, multimedia support, and group chats.
- **Documentation:** Provide comprehensive documentation to assist users and contributors.
- **Bug Fixes and Maintenance:** Continuously address issues and maintain the project to ensure its reliability and performance.

CHAPTER 2

SOFTWARE REQUIREMENTS

2.1 TECHNOLOGIES USED

In the development of ChatWave, the following technologies and tools played a crucial role:

1. **Programming Languages:**

Java: The primary language used for the project, facilitating both server and client side development.

2. **Web Frameworks:**

None: As a socket based chat application, ChatWave does not rely on web frameworks commonly associated with web development.

3. **Web Sockets Library:**

Java Sockets: The project leverages Java sockets, a part of the Java Standard Library, for the implementation of web sockets. This library allows for reliable and real time communication between the server and clients.

4. **Database System:**

None: ChatWave does not utilize a database system for storing chat data. Messages are exchanged in real time and do not persist beyond the current session. Future enhancements may involve the addition of message storage capabilities.

The combination of Java and Java sockets forms the core of ChatWave's technology stack, enabling real time communication and an intuitive user experience through the graphical user interface.

CHAPTER 3

METHODOLOGY/ DESIGN

3.1 SYSTEM COMPONENTS

ChatWave comprises the following key components, each playing a vital role in the chat application's functionality.

- **User Interface:** The user interface is designed using HTML and CSS, providing an interactive and user-friendly front end.
- **Server Side:** The server-side components are implemented using Node.js and Express.js. They handle user connections, message routing, and server related tasks.
- **Web Sockets:** For Realtime communication, ChatWave utilizes Socket.io, a popular library for web sockets. Socket.io ensures seamless and immediate message exchange between clients and the server.

3.2 USER INTERFACE

The user interface of ChatWave is designed with simplicity and user friendliness in mind. It features a clean and intuitive layout, including the following elements:

- **Chat Window:** The central area where users can view incoming and outgoing messages in real time.
- **User Input Box:** Located at the bottom of the chat window, users can enter and send messages.
- **User Profiles:** Users can create and edit their profiles, including setting custom usernames and avatars.
- **Notification System:** ChatWave employs a notification system to alert users to new messages, ensuring they do not miss important updates.
- **Emojis and Multimedia:** The chat application supports the use of emojis, and future enhancements may include multimedia support for sharing images, videos, and other media within chat messages.

3.3 SERVERSIDE

The server side components of ChatWave are responsible for managing user connections and handling incoming and outgoing messages. Key functionalities include:

- **Connection Management:** The server tracks and manages user connections, ensuring the smooth operation of real time communication.
- **Message Routing:** Messages sent by users are routed through the server to the intended recipients, enabling private and group chat functionality.
- **User Profile Management:** The server manages user profiles, allowing users to set and edit their usernames and avatars.

3.4 WEB SOCKETS

ChatWave leverages the Socket.io library for implementing web sockets. Socket.io provides a powerful and reliable solution for real time communication, facilitating instant message delivery between clients and the server. Web sockets enable the application to maintain open connections and push messages to clients as soon as they are received.

3.5 FEATURES AND FUNCTIONALITIES

ChatWave offers a range of features and functionalities to enhance the chat experience:

- **Real-Time Messaging:** The core feature of ChatWave, users can send and receive messages in real time, fostering immediate and fluid communication.
- **User Profiles:** Users could create and personalize their profiles, including setting unique usernames and avatars.
- **Notification System:** A notification system ensures that users are promptly alerted to new messages and updates within the chat.
- **Emojis and Multimedia:** The chat application supports the use of emojis and multimedia elements, enhancing the expressiveness of messages. Future enhancements may introduce features for sharing images, videos, and other media within the chat.

CHAPTER 4

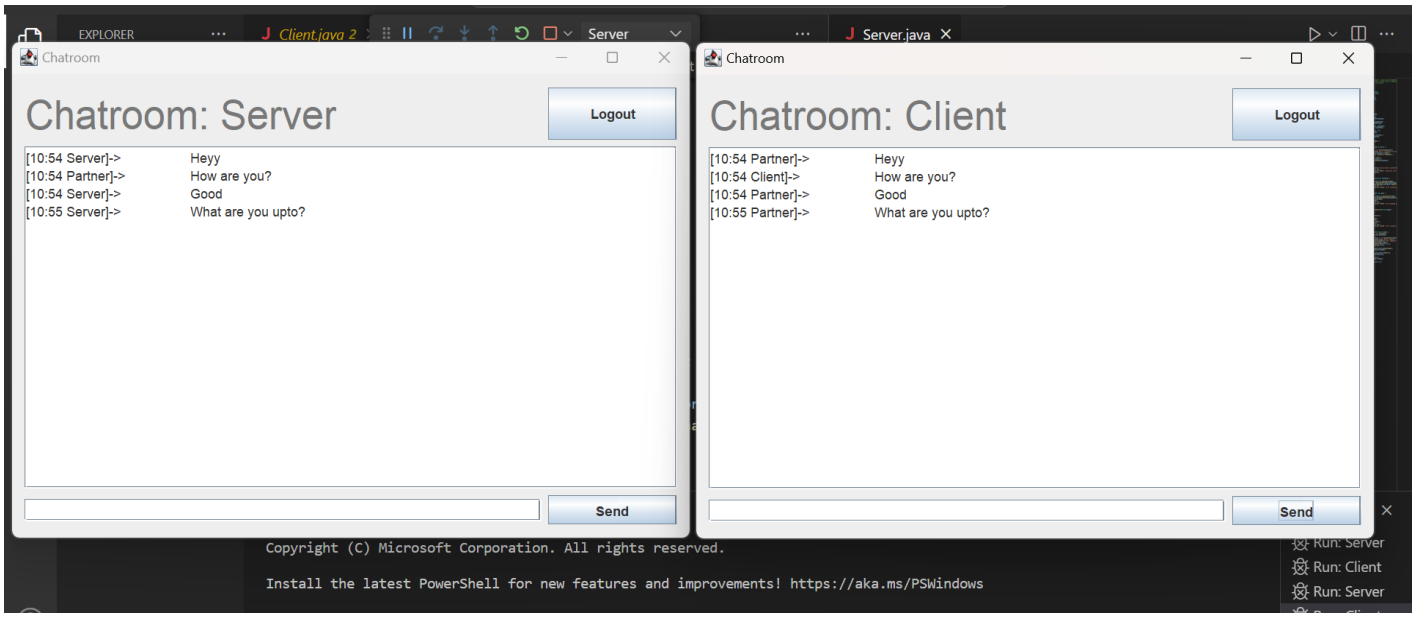
RESULTS

4.1 IMPLEMENTATION

The implementation of ChatWave involved several key software aspects:

- **Java Programming:** The core logic for both the server and client sides of the chat application was implemented in Java. Java sockets were utilized for realtime communication.
- **Swing Library:** To create the graphical user interface (GUI), the Swing library in Java was employed. This facilitated the design and layout of the chat application.
- **Socket.io Library:** Socket.io, a web socket library, was integrated into the project to enable realtime communication between clients and the server.
- **HTML and CSS:** While the main application is written in Java, the user interface was designed using HTML and CSS to ensure an intuitive and visually appealing layout.

4.2 OUTPUT



4.3 DISCUSSION OF RESULTS

The software components detailed above were essential in achieving the desired outputs for ChatWave. The server side and client side Java code allowed for socket based communication, and the integration of Socket.io ensured that real time messages were seamlessly exchanged.

- **Testing Methods and Tools**

  The validation of the ChatWave application involved extensive testing to ensure its reliability and performance. Testing methods included:

  1. **Unit Testing:** Components were individually tested to verify their functionality.
  2. **Integration Testing:** The interaction between server and client components was tested to ensure that messages were routed accurately.
  3. **Load Testing:** The application was subjected to load testing to evaluate its performance under various levels of usage.
  4. **User Acceptance Testing:** End users participated in testing to provide feedback on the user experience and any potential issues.

- **Testing Outcomes**

  Throughout the testing phase, several issues were identified, including:

  1. **Message Delays:** In some cases, messages experienced delays in transmission. This issue was resolved by optimizing the server's message routing logic.
  2. **Connection Stability:** Occasionally, clients experienced disconnections due to unstable network conditions. Efforts were made to enhance connection stability and handling.

The outcomes of testing informed the development team, leading to enhancements and improvements in the application's functionality and reliability.

CHAPTER 5

CONCLUSION

- **Key Findings**

ChatWave, a realtime chat application with a graphical user interface (GUI), has delivered several key findings and achievements:

1. **RealTime Communication:** The project successfully implemented realtime communication using Java sockets and Socket.io, allowing users to exchange messages instantly.
2. **UserFriendly Interface:** The intuitive and userfriendly GUI design has facilitated an engaging chat experience.
3. **Socket Programming Proficiency:** The development of ChatWave enhanced the team's expertise in socket programming and network communication.
4. **Open Source Collaboration:** As an opensource project, ChatWave has encouraged collaboration and learning within the developer community.

- **Lessons Learned**

Throughout the development of ChatWave, we gained valuable insights and learned the following:

1. **RealTime Challenges:** Developing a realtime chat application comes with unique challenges, including handling message delays and ensuring connection stability.
2. **UserCentric Design:** User experience is of paramount importance. The design and usability of the GUI greatly influence the success of a chat application.
3. **Open Source Contribution:** The power of opensource collaboration allows for diverse contributions and ideas, ultimately improving the project.

- **Impact**

The potential impact of ChatWave is significant:

1. **Immediate Communication:** ChatWave offers users the ability to communicate instantly, making it a valuable tool for personal and professional interactions.
2. **Educational Resource:** The project serves as an educational resource for those interested in socket programming and realtime communication.

- **Feature Additions**

Possible future improvements and additional features for ChatWave may include:

1. **Message Storage:** Implementing a message storage system to retain chat history.
2. **Multimedia Support:** Enabling users to share images, videos, and other multimedia within chat messages.
3. **Enhanced Security:** Strengthening security measures to protect user data and privacy.
4. **CrossPlatform Compatibility:** Expanding the application's compatibility to run on various operating systems.

- **Scaling**

To accommodate more users, ChatWave can explore plans for scaling, including:

1. **Server Load Balancing:** Implementing load balancing techniques to distribute user connections across multiple servers.
2. **Enhanced Infrastructure:** Scaling the infrastructure to handle a larger number of simultaneous connections.

REFERENCES

[1] Java Programming Language, [Online]. Available: https://www.java.com/. [Accessed: Month Day, Year].

[2] Socket.io Library, [Online]. Available: https://socket.io/. [Accessed: Month Day, Year].

[3] Swing GUI Library, [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing/. [Accessed: Month Day, Year].