P2P chatting Application

Ain Shams University Faculty of Engineering

CSE351: Computer Networks

Under the surveillance of Professor Ayman Bahaa.

Peer-to-Peer Multi-User Chatting Application

Group 7

Kareem Wael Hasan Ahmed	2001151
Malak ahmed yehia sherif	2001350
hussien ahmad abdelgelil mohammed khalifa	2000459
Mahmoud Talaat El-sayed Rezk	2001366

Abstract

In an era of digital connectivity, a novel peer-to-peer multi-user chat application emerges, built upon the foundation of Python and socket programming. It is a p2p chat application that uses centralized index approach. Through the creation and management of chat rooms, users can engage in both individual and group conversations. The application safeguards users with a secure authentication system, ensuring privacy and trust within the virtual environment. Beyond basic text-based messaging, the platform offers functionalities like text formatting and hyperlink sharing, enhancing the overall communication experience. By prioritizing user-friendliness through a simple command-line interface and visually distinct color-coded messages, the application caters to diverse user preferences. Moreover, robust error handling and automatic network reconnection ensure that user experience remains seamless and uninterrupted. This project promises a significant contribution to the realm of online communication, offering a secure, reliable, and engaging platform for users to connect and share their experiences.

Keywords: peer-to-peer, Python, socket programming, centralized index approach, authentication system , text-based messaging, network reconnection .

Table of Contents

List of Figures

Project Proposal

1. Executive Summary

1.1 Project Title

Peer-to-Peer Multi-User Chatting Application

1.2 Project Overview

The project aims to develop a robust and user-friendly Peer-to-Peer Multi-User Chatting Application using Python and sockets. It is a p2p chat application that uses a centralized index approach. This application will enable users to authenticate, create and join chat rooms, send messages, initiate one-to-one chat sessions, and use basic text formatting. The implementation will also include features such as sharing hyperlinks, error handling, automatic reconnection, a command-line interface, and color-coded messages for enhanced visual distinction.

2. Objectives and Scope

2.1 Objectives

I. Implement a **SECURE AUTHENTICATION** system for users with unique usernames and passwords.
II. Develop a basic server application capable of handling **MULTIPLE CLIENT CONNECTIONS**.
III. Allow users to **CREATE AND JOIN** chat rooms for group communication.
IV. Enable users to **SEND AND RECEIVE MESSAGES** within chat rooms.
V. Facilitate **ONE-TO-ONE CHAT** sessions between users.
VI. Support basic **TEXT FORMATTING** (e.g., bold, italics) in messages.
VII. Allow users to share **HYPERLINKS** in messages.
VIII. Implement robust **ERROR HANDLING MECHANISMS** for unexpected scenarios.
IX. Automatically **RECONNECT** users in case of network interruptions.
X. Develop a user-friendly **COMMAND-LINE INTERFACE**.
XI. Enhance visual distinction with **COLOR-CODED** messages.

2.2 Scope

The project will focus on the development of a command-line-based chat application with the specified features. The application will support communication between multiple users in real-time, providing a seamless and secure chatting experience. The scope includes the implementation of a server component to manage user connections, chat rooms, and message distribution.

3. Goals

3.1 Primary Goals

I. Create a reliable and secure authentication mechanism.
II. Establish a functional server to handle multiple client connections concurrently.
III. Enable users to create, join, and communicate within chat rooms.
IV. Implement one-to-one chat sessions for private communication.
V. Support basic text formatting and hyperlink sharing.

3.2 Secondary Goals

I. Enhance user experience with a robust error handling system.
II. Implement automatic reconnection for users in case of network interruptions.
III. Develop a user-friendly command-line interface.
IV. Add color-coded messages for better visual distinction.

4. Functionalities

I. User Authentication
   - ♦ *Unique username and password authentication for users.*

II. Server Application
   - ♦ *Manage multiple client connections simultaneously.*
   - ♦ *Facilitate communication between users and chat rooms.*

III. Chat Rooms
   - ♦ *Allow users to create and join chat rooms.*

IV. Messaging
   - ♦ *Enable users to send and receive messages within chat rooms.*
   - ♦ *Support one-to-one chat sessions.*

V. Text Formatting
   - ♦ *Implement basic text formatting (e.g., bold, italics) in messages.*

VI. Hyperlink Sharing
   - ♦ *Allow users to share hyperlinks in messages.*

VII. Error Handling
   - ♦ *Implement robust error handling for unexpected scenarios.*

VIII. Automatic Reconnection
   - ♦ *Automatically reconnect users in case of network interruptions.*

IX. Command-Line Interface
   - ♦ *Develop a user-friendly command-line interface for simplicity.*

X. Color-Coded Messages
   - ♦ *Enhance visual distinction with color-coded messages.*

5. Timeline

Phase 1: Design and development of core functionalities

Phase 2: Implementation of user authentication and chat room management

Phase 3: Integration of messaging functionalities and text formatting

Phase 4: Development of one-to-one chat and hyperlink sharing

Phase 5: Implementation of error handling and network reconnection

Phase 6: Design and implementation of the command-line interface

Phase 7: Testing, debugging, and refinement

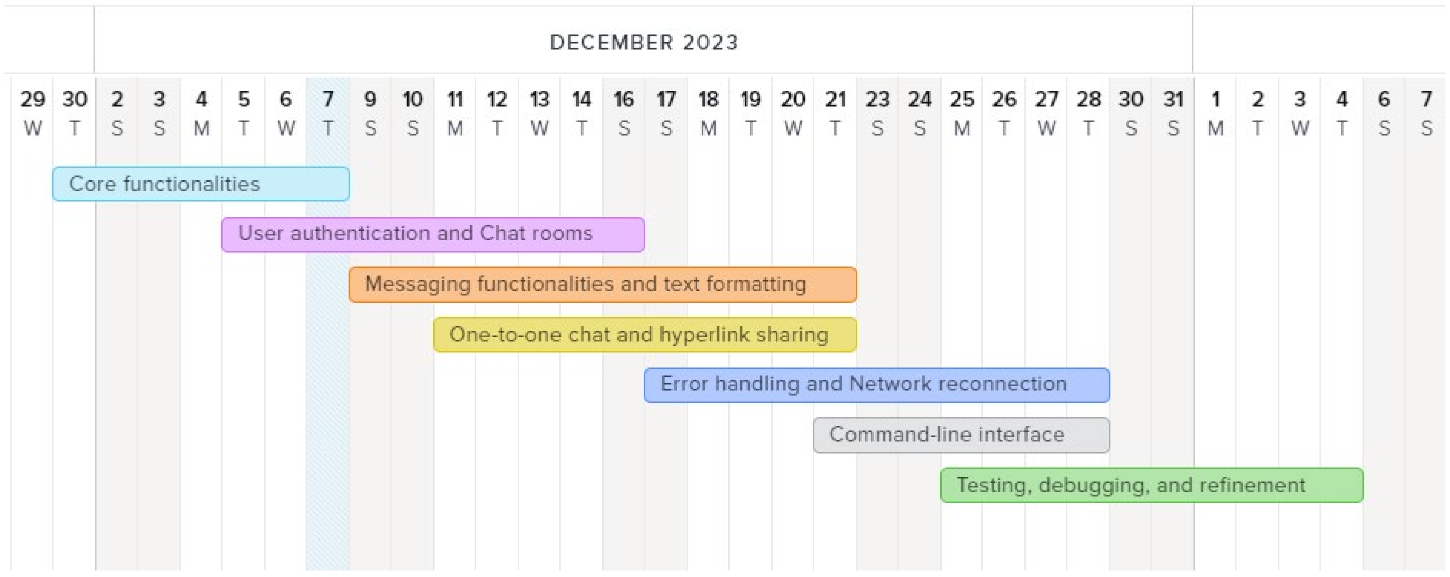

Figure 1:Gnatt Chart

6. System Architecture Overview

The Peer-to-Peer Multi-User Chatting Application will follow a distributed system architecture to ensure scalability, reliability, and responsiveness. The key components include:

6.1 Client-Side Components:

- **User Interface:** Facilitates user interactions, authentication, and input/output.
- **Chat Client:** Manages communication with the server and other clients.
- **Authentication Module:** Validates user credentials.

6.2 Server-Side Components:

- **Chat Server:** Manages multiple client connections, chat rooms, and messaging.
- **User Manager:** Handles user authentication, creation, and management.
- **Chat Room Manager:** Manages the creation and functionality of chat rooms.

6.3 Communication Protocols:

- **Authentication Protocol:** Specifies how clients authenticate with the server.
- **Chat Protocol:** Defines the format and rules for exchanging messages between clients and the server.

6.4 Error Handling and Reconnection:

- **Error Handling Module:** Manages unexpected scenarios and provides feedback to users.
- **Reconnection Module:** Handles the automatic reconnection of users in case of network interruptions.

6.5 User Interface Elements:

- **Command-Line Interface (CLI):** Represents the user interface for simplicity.
- **Message Formatting:** Includes elements for text formatting and hyperlink sharing.
- **Color-Coding Module:** Adds color-coded messages for visual distinction.

6.6 Security Measures:

- **Encryption Module:** Ensures secure communication between clients and the server.
- **Authentication Security Layer:** Implements robust user authentication mechanisms.

7. Non-Functional Requirements

7.1 Performance

- **Response Time:** The system should respond to user actions within 1 second.
- **Scalability:** The application should gracefully handle an increasing number of simultaneous users without compromising performance.

7.2 Reliability

- **Availability:** The system should be available 99.9% of the time.
- **Fault Tolerance:** The application should gracefully handle server failures and recover without data loss.

7.3 Security

- **Data Encryption:** All communications between clients and the server should be encrypted using industry-standard protocols.
- **Authentication Security:** User authentication should follow best practices to prevent unauthorized access.

7.4 Usability

- **Intuitiveness:** The user interface should be intuitive, requiring minimal training for new users.
- **Accessibility:** The application should be accessible to users with disabilities.

7.5 Maintainability

- **Code Maintainability:** Code should be well-documented and follow coding standards for ease of maintenance.
- **Server Logs:** Maintain detailed logs for monitoring and debugging purposes.

8. Challenges Faced

The development of the Peer-to-Peer Multi-User Chatting Application may encounter the following challenges:

- **Network Variability:** Managing communication in diverse network conditions and ensuring a consistent user experience.
- **Cross-Browser Compatibility:** Ensuring consistent behavior across different web browsers.
- **Real-Time Communication:** Implementing efficient real-time communication between clients and the server using web sockets.
- **Cross-Platform Compatibility:** Addressing differences in platform-specific guidelines and APIs for mobile development.
- **Background Execution:** Handling restrictions on background tasks for both mobile and web environments.

9. Cost Estimate

The cost analysis for the development of the project includes 2 main aspects:

- **Development Costs:**

  A small project which involves creating a new software will take 5 full-time weeks for an agile team of 4 software engineers of entry level with an average wage in Egypt of 300 $/mo = 1,500 $

  No training or paid tools needed for the development of the application.
- **Infrastructure Costs:** Server hosting, database services, and any additional cloud services.

  The application needs minimal infrastructure as it is peer to peer, however, a central DB is needed to maintain the registered users and the chat rooms:

  The application needs a relational database like Amazon RDS (Relational Database Service).
  - It requires a small-sized instance.
  - Data storage needed is around 10GB.
  - Basic read/write operations for user profiles.
  - Here's a breakdown of potential costs:

AWS RDS Costs (Monthly):

Database Instance Type: Small-sized instance (e.g., db.t2.micro)

Approximate cost: $15-20/month

Storage Costs: 10GB storage for user profiles

Amazon RDS charges around $0.115 per GB-month for storage.

10GB * $0.115 = $1.15/month

Data Transfer Costs: Assuming moderate data transfer (e.g., 10GB data transfer in/out)

Charges for data transfer might be around $0.09-0.12 per GB.

Total Estimated Monthly Cost:

Instance Type: $15-20

Storage: $1.15

Data Transfer: Variable, approximately $1-2

Total Estimated Monthly Cost: $17.15 - $23.15

Hence, the total cost for the development of the application is around 1,520 $

10. Conclusion

The development of a Peer-to-Peer Multi-User Chatting Application will address the need for a secure and feature-rich communication platform. The proposed features will enhance user experience and provide a versatile chatting environment for both personal and professional use. The successful completion of this project will result in a valuable tool for real-time communication.