Chat Application – Project Report

Abstract

The project focuses on the design and implementation of a real-time chat application using Java, socket programming, and JavaFX. It provides a lightweight and secure communication platform where multiple users can connect to a server and exchange text messages. The application includes a modern dark-themed interface, differentiates between user messages, and integrates basic encryption for secure transmission. This project demonstrates practical knowledge of networking, multithreading, and graphical user interface (GUI) development in Java.

Introduction

Real-time communication applications are essential in today's world, powering both social interactions and professional collaboration. Popular platforms such as WhatsApp, Telegram, and Slack show the demand for fast and secure communication.

The aim of this project was to build a miniature version of such systems—a simple, yet functional chat application. The project allowed us to explore fundamental concepts in Java networking (sockets), client-server architecture, encryption, and GUI design. By integrating these technologies, the application provides an environment where multiple users can exchange messages in a secure and user-friendly manner.

Tools Used

- **Java (JDK 17/21):** For developing the core client-server architecture.
- **JavaFX:** To design an interactive and modern user interface.
- **Maven:** For dependency management and structured project building.
- **Socket Programming:** To establish and manage communication between server and clients.
- **AES Encryption Utility:** To enhance security by encrypting and decrypting chat messages.
- **IDE (VS Code):** For coding, debugging, and project management.

Steps Involved in Building the Project

1. Server Implementation

- Built a Chat Server class using Server Socket to listen for incoming connections.
- Added multithreading (ClientHandler) to allow multiple users to chat simultaneously without blocking.
- Handled broadcasting of messages to all connected clients.

2. Client Implementation

- Developed a Chat Client class that connects to the server through a socket.
- Implemented message sending and receiving in real-time.
- Added differentiation of messages so that users see their own messages as "You" and others with their assigned username.

3. Common Utilities

- Created a Message class for structured message representation.
- Implemented CryptoUtil class to perform AES encryption/decryption, ensuring that messages exchanged are secure.

4. User Interface (UI)

- Designed with JavaFX to provide a graphical interface instead of console-based communication.
- Features include:
    - A dark violet-black theme (Telegram-like).
    - White text for readability.
    - A list view to display messages in sequence.
    - A text input field and a send button for interaction.
- Enhanced UI styling through an external CSS file.

5. Testing and Execution

- First, the server application was started in one terminal.
- Next, multiple clients were launched in separate terminals or windows.
- Verified message flow between multiple users.
- Ensured correct formatting: user sees “You” for their messages, while others see the sender’s username.
- Checked encryption and decryption for message security.

Conclusion

This project successfully delivers a real-time chat application with a simple design and secure communication. It demonstrates key concepts of Java networking, multithreading, socket programming, and GUI development with JavaFX.

Key learnings include:

- Building a client-server model in Java.
- Handling multiple users through threads.
- Designing interactive GUIs with themes.
- Applying basic encryption to secure communication.

The project can be extended with advanced features like file sharing, user login/authentication, emojis, and deployment over the cloud. Overall, it provided valuable experience in bridging theory with practical implementation of modern communication systems.