## Introduction

This chat application is designed to allow users to communicate via private or group conversations in real-time. It is built using a server-client architecture, where clients connect to a central server to send and receive messages. Java, WebSockets, and MySQL are used for communication and data persistence. The client user interface is built using Java Swing.

## System Architecture

The system is based on a multi-threaded server that handles connections from multiple clients. Each client communicates with the server using WebSockets, allowing for real-time message exchange. The server uses a MySQL database to store users, conversations, and messages. The system consists of three main components:

- **Client:** A Java Swing application where users can log in, join conversations, and send messages.
- **Server:** A multi-threaded server that manages client connections and processes messages.
- **Database:** A MySQL database for persisting user, conversation, and message data.

## Project Structure

The project is divided into the following packages:

- **Chat-application(client):** Contains the client-side code, including the user interface and networking logic.
- **Database:** Contain the sql to setup the sql database for our project.
- **Server:** Contains the server-side code, including the server's main logic.

## Client-Server Communication

The client connects to the server using a socket, and messages are sent over the network using WebSocket protocols. Each message is serialised and transmitted from the client to the server. The server then broadcasts the message to all other clients participating in the same conversation.

## Concurrency Handling

The server uses a fixed thread pool to handle multiple clients concurrently. Each client is managed by a separate thread, ensuring that messages can be processed in parallel.
For the database layer we handled it by using a 2 phase locking mechanism. That is locking the create, update, delete database operation and releasing it after the operation is done to prevent dirty read and dirty write.

## Features

- **Private and Group Messaging:** Users can participate in either private or group chats. A conversation object represents the chat and manages the list of participants.
- **Message History:** Messages are stored in a MySQL database with a timestamp, allowing users to retrieve message history even when they are offline.
- **User Status:** The system keeps track of each user's online/offline status and displays it in the chat UI.

## Conclusion

This chat application demonstrates the use of a multi-threaded server-client architecture, real-time message handling with WebSockets, and data persistence with MySQL. Future improvements could include more advanced features such as message encryption, file sharing, and rate limiting.