Java Chat Application

INTRODUCTION :

This project presents a desktop-based Java Chat Application designed for real-time communication over a local network. It combines JavaFX for GUI, Java Socket API for networking, and MySQL for backend storage. The system supports encrypted messaging, automatic server discovery via JmDNS, and modular architecture for scalability. The goal was to build a secure, responsive, and teachable chat platform using full-stack Java technologies.

ABSTRACT :

The Java Chat Application enables real-time messaging between clients and a centralized server. It features a JavaFX-based interface, encrypted message transmission, and MySQL-backed data persistence. JmDNS simplifies server access across LAN, eliminating manual IP setup. The project demonstrates practical integration of frontend design, backend logic, and network automation in a cohesive Java ecosystem.

TOOLS USED :

▢ Client Side -

- JavaFX - for building the graphical user interface.
- JavaFX SceneBuilder - to visually design FXML-based layouts.
- JmDNS - for an easy search and connection to server.
- Maven - to build the entire project.

▢ Server Side -

- Java Socket API - for handling server - client communication.
- Java JDBC API - to interact with **MySQL** Database.
- JmDNS - to create the server on an mDNS name.
- Message Encryption - for secure communication between server - client.
- Maven - to build the entire project.

STEPS INVOLVED :

1. **Requirement Analysis & Planning -**

- Defined core features: user login, message encryption, real-time chat, and message history.
- Chose JavaFX for GUI and Java Socket API for networking.

2. **Designing the GUI -**

- Created FXML layouts using SceneBuilder.
- Implemented responsive scenes for login, registration, and chat windows.

3. **Setting Up the Server -**

- Built a multi-threaded server using Java’s Socket API.
- Integrated JDBC for database operations like user authentication and message logging.
- Added JmDNS for automatic server discovery across LAN.

4. **Client-Side Development -**

- Established socket connection to the server.
- Implemented message encryption before sending and decryption upon receiving.
- Designed event-driven UI interactions for sending and receiving messages.

5. **Database Integration -**

- Created MySQL tables for users and messages.
- Ensured secure and efficient data retrieval and storage using JDBC.

6. **Testing & Debugging -**

- Verified socket stability and message flow.
- Debugged UI transitions and thread synchronization issues.
- Validated encryption-decryption logic for message integrity.
- Tested JmDNS discovery across multiple devices.

7. **Packaging & Deployment -**

- Used Maven to build both client and server modules.

CONCLUSION :

This project successfully integrates GUI, networking, and database layers into a secure desktop chat application. Encryption ensures message privacy, while JmDNS streamlines server access. The experience deepened my understanding of Java’s ecosystem and strengthened my skills in modular design, debugging, and full-stack development—making it a valuable internship milestone.