PBCST 304 PROJECT PHASE I PRESENTATION

LanChat

GROUP - 1

Group Members

05 MDL24CS012 Adwaitha P S
09 MDL24CS126 Alna Rebecca Siby
36 MDL24CS106 Jiya Mary Jacob
39 MDL24CS115 Joylin Alex
54 MDL24CS159 P Devananda
58 MDL24CS171 Riona Poly

Project Guide
Athira S Nair
Assistant Professor

Department of Computer Science and Engineering
Model Engineering College, Thrikkakara

Introduction/ Problem Statement

Problem Statement: Most messaging platforms rely on internet access, making them unusable in settings with limited or no connectivity. In such environments like remote areas, internet-restricted workplaces, emergency or disaster zones, users lack a reliable way to communicate digitally in real time over a local network.

There is a growing need for a reliable real-time communication solution that does not rely on internet access. LANChat addresses this need by providing a messaging platform that operates entirely within a Local Area Network (LAN). This enables users in the same network to exchange messages securely and instantly, ensuring continued communication in offline or restricted settings.

Objectives

- To enable Internet-free and seamless, real-time communication.
- To enable instant communication between users connected to the same LAN network.
- To create a lightweight and user-friendly chat interface suitable for small-scale environments.

OOP Concepts

- Modularity -> Backend is divided into clear layers — Controller, Service, and Repository
- Maintainability and Scalability -> The system is easy to update or extend
- Reusability -> Common service methods are reused across the application.

There are more OOP concepts such as

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Proposed System

1. **Overview**

A local network chat application with role-based access control (RBAC). Works offline within a LAN, enabling secure group communication.

2. **Components**

i) Client (Java Swing)

- Desktop application with a simple chat UI
- Connects to the server using the server's local IP
- Sends and receives messages via HTTP requests

ii) Server (Spring Boot + Embedded Tomcat)

- REST API backend built using Spring Boot
- Embedded Tomcat server handles HTTP communication
- Provides endpoints for login, chat, group management, etc.

Proposed System contd...

iii) Database (MySQL)

Stores:

- User credentials and roles (Admin, Moderator, Member)
- Chat groups and memberships
- Message history

3.**Key Features**

- **RBAC (Role-Based Access Control):** Controls access to chat groups based on user roles
- **LAN Communication (Offline):** No internet needed; works within a local network. Clients connect using the server's IP address
- **Tomcat as Web Server:** Embedded in Spring Boot, handles all incoming HTTP requests from clients
- **Multithreading for Real-Time Communication:** Each client connection is handled in a separate thread, ensuring responsiveness and supporting concurrent messaging.

OOP Concepts: Encapsulation and Inheritance

Concept	Usage	How It Is Used
Encapsulation	`User`, `ChatMessage`, `DatabaseHandler`	Private fields (e.g., username, password) accessed through getters/setters. Prevents direct access to sensitive data.
Inheritance	`User` → `AdminUser`; `Message` → `TextMessage`, `FileMessage`, `ImageMessage`	AdminUser inherits common user properties and adds admin actions. Different message types reuse common fields (sender, timestamp) from Message and add extra data (file path, image).

OOP Concepts: Polymorphism and Abstraction

Concept	Usage	How It Is Used
Polymorphism	`MessageHandler` Interface, Implementations: `GroupMessageHandler`, `PrivateMessageHandler`	The same method `sendMessage()` behaves differently: `GroupMessageHandler` broadcasts to all group members, while `PrivateMessageHandler` sends to one user. Client code calls `handler.sendMessage()` without knowing the implementation.
Abstraction	`MessageHandler` Interface	Hides message sending logic details. Provides a simple method `sendMessage()` for different chat types without exposing the underlying implementation.

Scope of LANChat

- Enable real-time text communication over a local area network using a client–server architecture.
- Support both broadcast and private messaging between users.
- Allows creation of multiple secure groups for different departments or teams.
- Handle multiple simultaneous clients through multithreading.
- Provide user identification and connection/disconnection notifications.
- Incorporate basic error handling for network disruptions.
- Offer scalability to operate over MANs and WANs.
- Implement encryption for all message transmissions to ensure confidentiality and protect against unauthorized access.
- Adaptable for use by emergency services, healthcare, corporate offices, education, defense, and disaster management.
- Potential for encryption in highly confidential environments.

Limitations

- Works only within the same connected network in the base version.
- Performance may be affected with a very large number of connected users.
- Requires that all devices be powered and connected to the network for communication to work.
- Dependent on the network infrastructure; failure of the server or network disrupts communication.

Tools and Technologies Used in LANChat

Frontend (Client-Side)

- **Java Swing** – For building the desktop GUI (chat window, login page, etc.)
- **HTTP Client (e.g., HttpURLConnection)** – To send HTTP requests to the backend

Backend (Server-Side)

- **Spring Boot** – Main backend framework that handles APIs, business logic, and connections
- **Spring Web** – For creating RESTful endpoints
- **Spring Data JPA** – For mapping Java objects to database records and simplifying data access
- **Embedded Tomcat** – Lightweight web server built into Spring Boot

Database

- **MySQL** – Used to store user data, chat history, group details, and permissions
- Accessed using Spring Data JPA from the backend

Tools and Technologies contd...

Security and Access Control

- **Role-Based Access Control (RBAC)** – To restrict access to specific chat groups

Networking and Deployment

- **LAN (Offline)** – Communication is within a local network without internet
- **IP Address-based Client Connection** – Clients connect using the server's local IP

Development and Tools

- **VS Code** – For development and debugging
- **Maven** – For dependency and build management

Most Critical Libraries

1. `javax.swing` – UI
2. `java.net.HttpURLConnection` – Sending requests from Swing
3. `spring-boot-starter-web` – REST API backend
4. `spring-boot-starter-data-jpa` – Database interaction
5. `mysql-connector-java` – MySQL support

Expected Outcome – LANChat Application

Functional LAN-Based Messaging

- Real-time messaging over LAN/Wi-Fi
- Works offline; no internet needed
- Ideal for labs, hostels, offices, emergency zones

User Authentication

- Login with username and password
- Only registered users allowed
- Maintains privacy and identity

Group Chat Support

- Join predefined or custom groups
- Backend-managed groups
- Based on departments, teams, floors

Expected Outcome – LANChat Application

Real-Time Communication

- Instant message delivery
- Auto-refresh UI

Client-Server Architecture

- Frontend: Java Swing
- Backend: Spring Boot REST API
- Communication via HTTP

Data Persistence with MySQL

- Stores users, chats, groups
- Data retrieval on login
- No loss after server restart

Portability & Ease of Use

- Server on any LAN-connected device
- Plug-and-play setup

Project Timeline (4-5 Weeks)

1. **Week 1: Planning & Setup**
   - Finalize features & system design (architecture, DB schema).
   - Setup Java IDE, MySQL, LAN testing environment.
   - Practice basic Client-Server socket connection demo.
2. **Week 2: Core Backend Development**
   - Build TCP Server (multi-client with threads).
   - Build Client for message sending/receiving.
   - Implement message broadcasting.
   - Test communication via console (no GUI yet).
3. **Week 3: Authentication & Database Integration**
   - Design DB (Users, Groups, Chat History).
   - Implement Login/Registration with DB (JDBC).
   - Store & retrieve chat messages from DB.
   - Test authentication and secure access flow.

Project Timeline (4-5 Weeks) contd.

4. **Week 4: GUI & Group Messaging**
   - Design simple GUI (Login screen, chat window).
   - Link GUI with backend socket communication.
   - Implement group-based messaging and online user list.
   - Ensure full functionality via GUI.
5. **Week 5: Testing & Finalization**
   - Full system testing on LAN.
   - Handle disconnections, chat history load.
   - UI refinements, bug fixes.
   - Prepare report and demo presentation.

Conclusion

LANChat is a reliable, internet-free messaging platform built for Local Area Networks. Using Java Sockets, a Swing-based GUI, and secure user authentication, it offers seamless real-time chat even in offline environments.

Currently, the scope of the project is limited to communication within LANs. However, in the future, it can be expanded to support communication over Metropolitan Area Networks (MANs) or Wide Area Networks (WANs) by leveraging private IP addressing and secure networking protocols

This project offers a practical communication solution for institutions while providing hands-on experience in networking concepts like multithreading, socket programming, and database integration. Its modular, portable design makes LANChat highly scalable.

The completion of this project will serve as a practical demonstration of applying Object-Oriented Programming principles in solving real-world problems, while enhancing our technical proficiency in Java, networking, and database management.