Project Title :

CHAT APPLICATION USING MULTITHREADING

1. Core Feature Implementation:

The chat system allows:

- Server-Client Architecture: One server, multiple clients communicating in real-time.
- Private & Broadcast Messaging: Users can send private or public messages.
- Concurrent Clients: Multiple threads handle simultaneous client connections.
- Simple GUI Interface: Built using Swing for intuitive user interaction.

2. Error Handling and Robustness

- **Try-Catch Blocks:** Applied to all socket, stream, and UI operations.

  Connection Timeout Handling: Graceful exit when client disconnects.
- **Input Validation**: Prevents blank or malformed messages.
- **Thread-Safety**: Shared resources synchronized where needed.

3. Integration of Components

The system follows a **modular design** with the integration of:

- Core Java (Socket, ServerSocket, Thread)
- Java Swing for GUI
- I/O Streams for message transmission
- MVC pattern for code separation

4. Event Handling and Processing

- Button clicks, message sends, and text input events handled via ActionListener.
- Client socket receives and dispatches messages using threads.
- Server spawns a new thread for each incoming client connection.

  Real-time message display with scrollable chat area.

5. Data Validation

- Message text trimmed and validated before sending.
- Username uniqueness ensured on client login.
- Empty message blocking to prevent unnecessary traffic.

6. Code Quality and Innovative Features

Clean, modular class structure.

Reusable utility functions (e.g., timestamp formatting).

Logging of server events (connections/disconnections).

Color-coded messages for better readability.

Multithreaded message dispatcher to reduce lag.

7. Project Documentation

The documentation includes:

- System Overview
- Use Case Diagrams
- Class Diagrams
- Technology Stack
- Installation Instructions
- Test Cases & Results
- Limitations and Future Enhancements

(I.)Entity Relationship (ER) Diagram

Not applicable in traditional sense (no database), but logical entities include:

User (Username, IP Address, Status)

Message (Sender, Recipient, Timestamp, Content)

(II.) Class Diagram

```
+----------------+     +----------------+      +----------------+
|   ChatServer   |<--> | ClientHandler  | <--> |  ChatClient    |
+----------------+     +----------------+      +----------------+
| - serverSocket |     | - socket       |      | - socket       |
| - clientList   |     | - input/output |      | - input/output |
| +startServer() |     | +run()         |      | +sendMessage() |
+----------------+     +----------------+      +----------------+
```

(III.) Java Code Structure

Packages:

- **server**: Contains ChatServer.java, ClientHandler.java
- **client**: Contains ChatClient.java, ChatGUI.java
- **utils:** Timestamp formatter, config loader

(V.) Code Snippets

Issuing a Book (Java):

```
public class ClientHandler extends Thread {
    private Socket socket;
    private BufferedReader input;
    private PrintWriter output;

    public void run() {
        try {
            String message;
            while ((message = input.readLine()) != null) {
                // broadcast or direct message
            }
        } catch (IOException e) {
            System.out.println("Client disconnected");
        } finally {
            try { socket.close(); } catch (IOException ignored) {}
        }
    }
}
}
```

(VI.) GUI Screenshots (Descriptions)

Login Screen: User enters unique username.

Chat Window: Text area for chat history, input field for new messages.

User List Panel: Displays online users for private messaging.

(VII.) Future Enhancements

File Sharing Support

Chat Room Creation

Emoji and Sticker Integration

Database Logging for Message History

Mobile App Integration

Secure Socket Layer (SSL) Encryption

Admin Tools for user kick/mute