Group Members: Aimable, Javon, Evan, Femi, Tony

\*NOTE\*: Evan did not contribute to this, could not get ahold of him.

**Phase 2 and 3 Socket Program Design Specification: Network Architecture Design**

**Server Components**

**MainServer - Primary server component responsible for initializing and managing the chat server infrastructure.**

* Main Responsibilities
  + Server socket initialization and port binding
  + Client connection acceptance
  + Thread pool management
  + Resource allocation and deallocation
  + System shutdown coordination
* Core server class that initializes the server socket and manages client connections
* Listens for incoming connections on a specified port
* Maintains a list of connected clients (ClientHandler instances)
* Implements thread pooling for handling multiple clients efficiently

Key Methods for Implementation

* void initialize() // Setup server resources
* void acceptConnections() // Handle incoming connections
* void broadcastMessage() // Send message to all clients
* void removeClient() // Handle client disconnection
* void shutdown() // Graceful server shutdown

**ClientHandler - Manages individual client connections and their communication with the server.**

* Manages individual client connections
* Runs in its own thread to handle client communication
* Maintains client metadata (username, IP address, connection status)
* Responsible for message broadcasting and private messaging

Key Methods for Implementation

* void processMessages() // Handle incoming messages
* void sendMessage() // Send message to client
* void handleDisconnect() // Manage client disconnection
* void validateMessage() // Verify message integrity

**MessageProcessor - Handles message formatting, validation, and routing between clients.**

* Main Responsibilities  
  + Message parsing and formatting
  + Message validation
  + Message routing
  + Error handling
  + Message queuing
* Handles message formatting and processing
* Implements message types (broadcast, private, system notifications)
* Manages message queuing and delivery confirmation

**Client Components**

**ChatClient- Manages the client's connection to the server and handles user interactions.**

* Main Component Responsibilities
  + Server connection management
  + Message sending/receiving
  + User interface updates
  + Error handling
  + Connection state management
* Manages connection to server
* Maintains local state (connection status, message history)
* Implements reconnection logic on connection loss
* Key Methods for Implementation
  + Key Responsibilities
  + Server connection management
  + Message sending/receiving
  + User interface updates
  + Error handling
  + Connection state management

**MessageListener- Dedicated component for handling incoming messages from the server.**

* Main Key Responsibilities
  + Message reception
  + Connection monitoring
  + UI update triggering
  + Error detection
* Runs in separate thread
* Listens for incoming messages from server
* Updates GUI with received messages
* Handles server disconnection events

**Phase 3: Setup and Prototyping**

**Code in Attached Zipfile**

**ServerGUI - Femi**

**A screenshot of a computer

Description automatically generated**

**Client GUI Development – Javon**

**Testing and Analysis**

**Tools:**

* Wireshark
* Filter Expressions
* Screenshots

**Analysis Steps:**

* 1. **Develop communication scenarios for testing**:
     1. Create test cases where multiple clients connect to the server, exchange messages, and the server relays messages. Include private messaging and broadcasting to all clients except the sender.
  2. **Capture packets from both client and server machines**:
     1. Run Wireshark on both the server and client machines to monitor the packet flow.
     2. Capture all traffic related to the chat, including **TCP** packets as this protocol ensures reliable message delivery.
  3. **Filter packets by IP address**:
     1. Use **Wireshark filters** to isolate the traffic between the clients and server by filtering IP addresses of each client and server machine.
     2. Focus on filtering for relevant **TCP ports** (e.g., tcp.port == <server port>) used by your chat application.
  4. **Take screenshots of Wireshark displays**:
     1. Take screenshots of significant communication points, such as the establishment of TCP connections (handshake), message transmissions, and server relays.
     2. Include evidence of successful message broadcasting, private messages, and authentication attempts (if applicable).

**Data Collection:**

* **Packet Capture Data (PCAP)**:
  + Collect packet data from **Wireshark** at key points of communication: when a client connects, during message relay, and for private messages.
  + Save the **PCAP** files for future analysis and documentation.
* **Analyzing Specific Features**:
  + **Connection Setup (Handshake)**: Verify the establishment of a successful **TCP handshake** between clients and server (SYN, SYN-ACK, ACK sequence).
  + **Message Relay**: Ensure that when a message is sent, it is properly forwarded to all other clients except the sender. Look for **TCP segments** that carry message data and verify the destination IP addresses.
  + **Client List Display**: Check if the server properly updates and broadcasts the list of connected clients to all active clients.
  + **Errors or Resets**: Watch for any **RST (Reset)** packets, **timeouts**, or **ICMP error messages**, which could indicate communication issues between the client and server.
  + **Data Integrity**: Ensure the message payloads are complete and correctly received without fragmentation or loss.
  + **Authentication**: Analyze packets to verify if username/password exchanges are handled correctly, ensuring that credentials are securely transmitted.
* **Filtering**:
  + Use **Wireshark filters** to narrow down relevant traffic:
    - **IP address filter** to isolate communication between specific clients and the server.
    - **TCP filter** to focus only on the port your application is using.
    - **String search** for message content within the payload to verify that the correct message is being relayed to the correct client.

**Detailed Team Member Responsibilities**

**Aimable M. (Team Leader)**

**Primary Focus: Core Server Development & Project Coordination**

1. **Server Core Development**
   * Implement MainServer class including:
     + Server socket initialization and configuration
     + Thread pool management system
     + Client connection acceptance logic
     + Resource allocation/deallocation mechanisms
2. **Project Coordination**
   * Setup and maintain project structure:
   * Create development milestones and deadlines
   * Ensure component integration
   * Monitor progress and adjust assignments
3. **GitHub Repository Management**
   * Setup repository structure
   * Maintain documentation

**Evan**

**Primary Focus: Network Protocol Implementation**

1. **TCP Socket Implementation**
   * Develop MessageProcessor class
   * Implement message routing system
   * Create connection pooling mechanism
2. **Protocol Development**
   * Define message formats
   * Implement message validation
   * Create error handling protocols
3. **API Documentation**
   * Document all network-related methods
   * Create sequence diagrams for message flow
   * Provide integration guidelines

**Javon**

**Primary Focus: Client GUI Development**

1. **Client Interface Implementation**
   * Develop ClientGUI components
   * Implement event handlers
   * Create input validation system
2. **GUI Features**
   * Message display formatting
   * User list management
   * Connection status indicators
   * Error message displays
3. **Client-Side Integration**
   * Connect GUI to network layer
   * Implement message sending/receiving
   * Create client-side error handling

**Femi**

**Primary Focus: Server GUI Development**

1. **Server Interface Implementation**
   * Develop ServerGUI components
   * Create monitoring displays
   * Implement admin controls
2. **Documentation**
   * Create user manual with screenshots
   * Document server configuration
   * Provide setup instructions
   * Create troubleshooting guide
3. **Integration Testing**
   * Test GUI-Server integration
   * Verify admin controls
   * Validate monitoring systems

**Tony**

**Primary Focus: Testing & Analysis**

1. **Testing Framework**
   * Create test scenarios
   * Implement automated tests
   * Create performance tests
2. **Wireshark Analysis**
   * Setup capture configurations
   * Define test scenarios
   * Document packet flows
3. **Error Handling**
   * Implement validation
   * Create error logging system
   * Implement recovery procedures

Phase 2 and 3 Socket Program Design Specification

Group Members: Aimable, Javon, Evan, Femi, Tony