**Al-Alamein inernational university**

**Network course**

****

**Advanced Socket Programming in Java - Multi-Threaded Secure Chat Application**

**1. Introduction**

**Description of how a Multi-Threaded Secure Chat Application is created using Java Socket Programming. The project showcases basic networking concepts such as TCP socket communication, multi-threading, JSON-based messaging, and file transfer functionality. The application allows various clients to connect to a central server, exchange real-time messages, and transfer files securely.**

**2. Project Overview**

**Objectives**

**Develop a multi-threaded chat server that can handle multiple client connections simultaneously.**

**Support real-time messaging with public, private, and system message capability.**

**Enable secure file transfer between clients.**

**Offer error handling and network interruption tolerance.**

**Provide well-documented code for future maintenance.**

**3. System Architecture**

**Server Components**

**ChatServer Class**

**Listens on a specified port (default: 8080).**

**Uses ServerSocket to manage client connections.**

**Manages multiple clients using ExecutorService thread pool.**

**Manages message broadcasting and file transfers.**

**ClientHandler (Inner Class)**

**Each client connection runs in a separate thread.**

**Manages incoming messages (text or files).**

**Stores client state (nickname, active transfers).**

**Client Components**

**ChatClient Class**

**Connects to server using Socket.**

**Manages user input (messages & file transfers).**

**Runs a background thread to receive server messages.**

**FileTransfer Helper Class**

**Manages file chunking and reassembly.**

**Manages transfer progress.**

**4. Key Features Implemented**

**1. Multi-Threaded Server**

**Uses ExecutorService to handle multiple clients in an efficient way.**

**Prevents blocking I/O operations by allocating each client to a separate thread.**

**2. JSON-Based Messaging Protocol**

**Messages follow a systematic structure:**

**json**

**{**

**"type": "public\_message",**

**"sender": "Alice",**

**"content": "Hello everyone!",**

**"timestamp": 1620000000**

**}**

**Supports different types of messages:**

**public\_message (broadcast message to all)**

**private\_message (direct message)**

**file\_transfer (file data chunks)**

**system\_message (server messages)**

**3. File Transfer System**

**Files are split into chunks (4KB each).**

**Each chunk is Base64-encoded for safe transfer.**

**Progress tracking ("progress": 75%).**

**Resumable transfers (in case of connection loss).**

**4. Challenges & Solutions**

**Port already in use Automatic fallback to next available port (8081, 8082, etc.)**

**JSON parsing errors Added try-catch blocks for malformed messages**

**8. Conclusion**

**This project effectively demonstrates advanced Java socket programming for scalability, reliability, and security. The chat program includes real-time messaging, file transfer, and multi-client management features, and it offers a good foundation for further expansion.**

**Team members**

**Yusuf Yasser 22100809**

**Mahmoud Hossam 22100828**