**7/8/24 PROJECT NAME**

**Network File Sharing Server and Client**

Creating a high-level and low-level design for a network file-sharing application involves outlining the system architecture, components, and interactions at different levels of detail.

**High-Level Design (HLD)**

**AIM :**  **Create a network file-sharing application with server-client architecture, enabling file transfer, listing, uploading, authentication, and encryption.**

#### **1. Architecture Overview**

* **Client-Server Model: The system follows a client-server architecture where multiple clients can connect to a central server to upload, download, and list files.**

#### **2. System Components**

1. **Client Application:**
   * **Sends authentication credentials to the server.**
   * **Requests file listings.**
   * **Downloads files from the server.**
   * **Uploads files to the server.**
2. **Server Application:**
   * **Authenticates clients.**
   * **Provides file listings to authenticated clients.**
   * **Handles file download requests.**
   * **Handles file upload requests.**
3. **Network Communication:**
   * **Uses TCP/IP for reliable data transmission.**
   * **Secured using SSL/TLS for encrypted communication.**

#### **4. Data Flow Diagram**

1. **Client Authentication:**
   * **The client sends a password to the server for authentication.**
   * **The server verifies the password and responds with an acknowledgment.**
2. **File Listing:**
   * **The client requests a list of available files.**
   * **The server sends the file list to the client.**
3. **File Transfer:**
   * **Download: The client requests a file. The server sends the file data to the client.**
   * **Upload: The client sends a file. The server receives and stores the file data.**

### **Low-Level Design (LLD)**

#### **1. Detailed Component Design**

1. **Client Application:**
   * **Modules:**
     + **Network Module: Handles socket creation, connection, and communication.**
     + **Authentication Module: Manages sending and receiving authentication data.**
     + **File Transfer Module: Handles file upload and download operations.**
     + **User Interface Module: Provides command-line interface for user interaction.**
2. **Server Application:**
   * **Modules:**
     + **Network Module: Handles socket creation, binding, listening, and accepting connections.**
     + **Authentication Module: Verifies client credentials.**
     + **File Management Module: Manages file listings, uploads, and downloads.**
     + **Security Module: Manages SSL/TLS setup and encryption.**

#### **2. Class Diagrams**

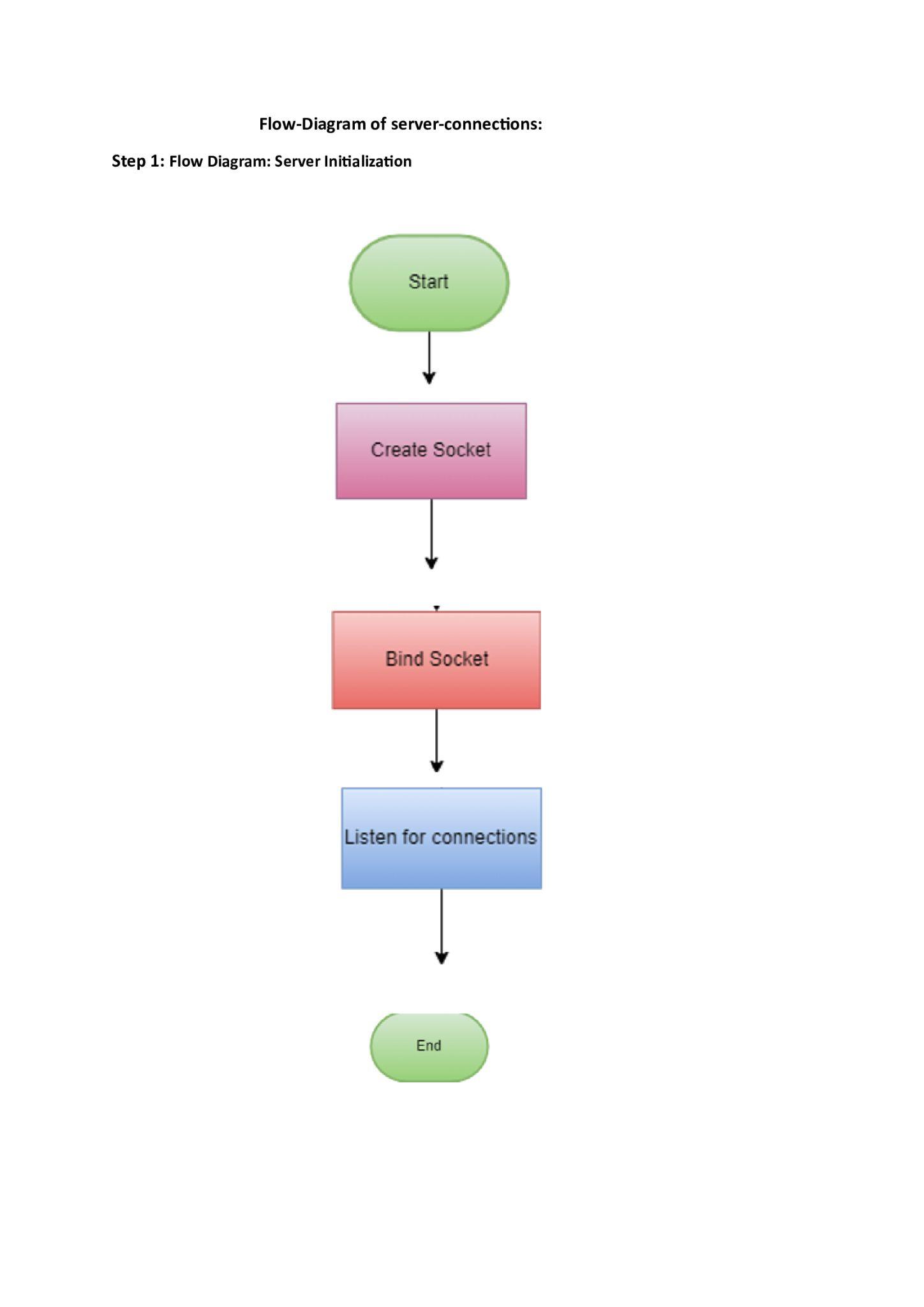
1. **Client Classes:**
   * **Client**
     + **Methods: connect(), authenticate(), requestFileList(), downloadFile(), uploadFile()**
   * **NetworkManager**
     + **Methods: createSocket(), connectToServer(), sendData(), receiveData()**
   * **FileManager**
     + **Methods: listFiles(), downloadFile(), uploadFile()**
   * **UserInterface**
     + **Methods: getCommand(), displayFileList(), displayMessage()**
2. **Server Classes:**
   * **Server**
     + **Methods: start(), acceptClient(), authenticateClient(), handleRequest()**
   * **NetworkManager**
     + **Methods: createSocket(), bindSocket(), listenForConnections(), acceptConnection(), sendData(), receiveData()**
   * **FileManager**
     + **Methods: listFiles(), sendFile(), receiveFile()**
   * **SecurityManager**
     + **Methods: initializeSSL(), createContext(), configureContext()**

#### **3. Sequence Diagrams**

1. **Client Authentication Sequence:**
   * **Client -> Server: connect()**
   * **Client -> Server: authenticate(password)**
   * **Server -> Client: authenticationResult()**
2. **File Listing Sequence:**
   * **Client -> Server: requestFileList()**
   * **Server -> Client: fileList()**
3. **File Download Sequence:**
   * **Client -> Server: downloadFile(fileName)**
   * **Server -> Client: sendFileData(fileData)**
4. **File Upload Sequence:**
   * **Client -> Server: uploadFile(fileName)**
   * **Client -> Server: sendFileData(fileData)**
   * **Server -> Client: uploadConfirmation()**

**Flow Diagram For Server Client Architecture**

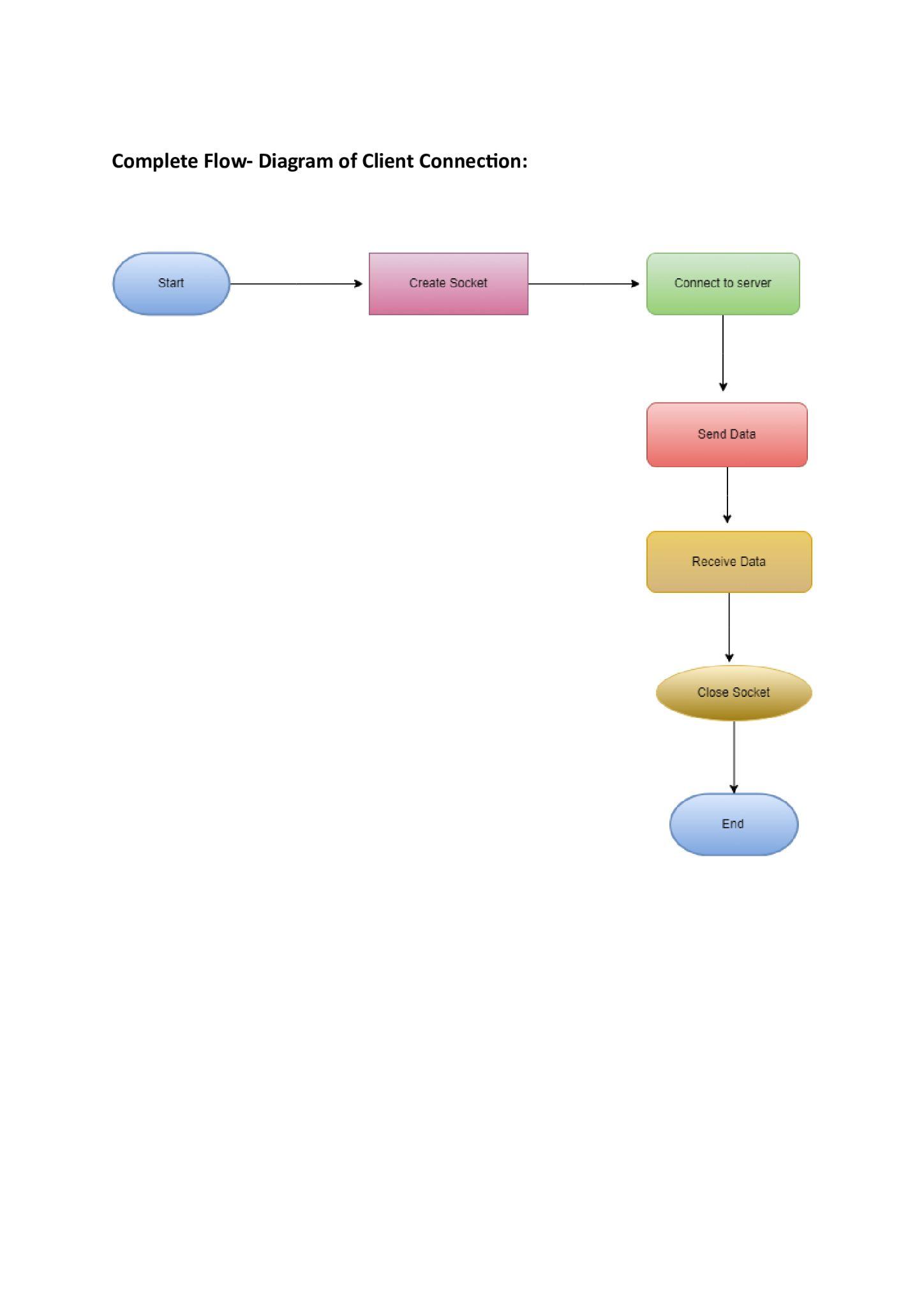
**The server application diagram is designed to represent the different modules and their interactions within the server application. Each module has a specific role in handling client requests, managing files, and ensuring secure communication.**

**CLIENT FLOW DIAGRAM**

**CLIENT- This is the main class that orchestrates the operations of the client application. It utilizes other modules to establish a connection to the server, authenticate, request file listings, and manage file transfers (upload/download).**

**The Client class uses the NetworkManager to send authentication credentials (e.g., password) to the server.**

**The server verifies the credentials and responds. The NetworkManager receives this response and passes it back to the Client.**

****