*Network File Sharing Server and Client:*

High and Low level designs:

**High-Level Design (HLD)**

**The high-level design outlines the architecture and major components of the system, providing a broad overview of how the system works.**

1. **System Overview:**
   * **The project aims to create a client-server application for file sharing over a network.**
   * **The server will host and manage files, while clients can upload, download, and list files from the server.**
2. **Architecture:**
   * **Client-Server Architecture: A central server interacts with multiple clients.**
   * **Protocols: The system will use TCP/IP for reliable data transmission.**
   * **Security: Authentication and encryption will be implemented to secure data transfer.**
3. **Components:**
   * **Server Component:**
     + **File Storage: Stores the shared files.**
     + **Request Handler: Manages client requests (upload, download, list).**
     + **Authentication Module: Handles user authentication.**
   * **Client Component:**
     + **User Interface: Allows users to interact with the system.**
     + **File Operations: Manages file uploads, downloads, and listings.**
     + **Network Communication: Manages communication with the server.**
4. **Use Cases:**
   * **File Upload: Clients can upload files to the server.**
   * **File Download: Clients can download files from the server.**
   * **File Listing: Clients can view a list of available files on the server.**
5. **Non-Functional Requirements:**
   * **Performance: The system should handle multiple concurrent connections.**
   * **Scalability: The architecture should support future expansions.**
   * **Security: Data should be encrypted during transmission.**

**Low-Level Design (LLD)**

**The low-level design provides detailed technical information about the components and their interactions.**

1. **Server Component Details:**
   * **File Storage:**
     + **Directory Structure: Organizes files in a structured directory.**
     + **File Metadata: Stores information like file name, size, and permissions.**
   * **Request Handler:**
     + **Request Parsing: Decodes incoming requests (e.g., using JSON or XML).**
     + **Request Processing:**
       - **Upload: Saves uploaded files to the server's file system.**
       - **Download: Sends requested files to the client.**
       - **List: Retrieves and sends the list of available files.**
   * **Authentication Module:**
     + **User Database: Stores user credentials securely.**
     + **Session Management: Manages user sessions and tokens.**
2. **Client Component Details:**
   * **User Interface:**
     + **Command Line Interface (CLI): Accepts commands for uploading, downloading, and listing files.**
     + **Graphical User Interface (GUI) (optional): Provides a more user-friendly interface.**
   * **File Operations:**
     + **File Selection: Allows users to choose files for upload/download.**
     + **Progress Tracking: Displays the progress of file operations.**
   * **Network Communication:**
     + **Socket Programming: Utilizes sockets for network communication.**
     + **Protocol Implementation: Implements the protocol for request-response communication.**
3. **Security Considerations:**
   * **Encryption:**
     + **Data Encryption: Encrypts data during transmission using SSL/TLS.**
   * **Authentication:**
     + **Password Hashing: Stores passwords securely using hashing algorithms.**
     + **Token-Based Authentication: Uses tokens to manage authenticated sessions.**
4. **Error Handling:**
   * **Client-Side:**
     + **Connection Issues: Handles cases where the server is unreachable.**
     + **Invalid Commands: Provides feedback for invalid user inputs.**
   * **Server-Side:**
     + **File Not Found: Returns appropriate error messages when files are missing.**
     + **Unauthorized Access: Denies access to unauthorized users.**
5. **Data Flow:**
   * **File Upload:**
     + **Client selects a file -> Sends upload request with file data -> Server stores the file and responds.**
   * **File Download:**
     + **Client requests a file -> Server retrieves the file -> Sends the file to the client.**
   * **File Listing:**
     + **Client requests a file list -> Server retrieves and sends the list.**

**Flowcharts**

**File Upload Flowchart**

1. Start
2. Client selects a file to upload.
3. Client sends an upload request with the file data to the server.
4. Server authenticates the client.
5. Server stores the file in the file system.
6. Server sends an acknowledgment to the client.
7. End

**File Download Flowchart**

1. Start
2. Client requests a list of files from the server.
3. Server sends the list of available files.
4. Client selects a file to download.
5. Client sends a download request to the server.
6. Server authenticates the client.
7. Server retrieves the file and sends it to the client.
8. Client receives the file.
9. End

**File Listing Flowchart**

1. Start
2. Client sends a list request to the server.
3. Server authenticates the client.
4. Server retrieves the list of files.
5. Server sends the list to the client.
6. End

