## CNT3004 – Socket-Based Networked File Sharing Cloud Server

**Project Objective**

The primary objective of this collaborative project is to design, implement, and evaluate a distributed file sharing system in the cloud employing a server-client architecture. The system should facilitate efficient and secure file transfer between multiple clients connected to a central server.

**Project Requirements**

* **Server-Client Architecture:** Implement a robust server-client architecture using Python where clients initiate connections to a central server for file requests and transfers. *The server and client processes must run on separate computers. A single computer cannot be used to host both the server and clients.*
* **File Transfer Protocols:** Utilize appropriate network protocols (e.g., TCP, UDP) for reliable and efficient file transmission.
* **Multithreading:** Employ multithreading on the server side to handle concurrent client requests and improve system performance.
* **File Types and Sizes:** Support a variety of file types (e.g., text, audio, video) with minimum specified size ranges:
  + Text files: 25 MB
  + Audio files: 1 GB
  + Video files: 2 GB
* **File Operations:** Implement the following Client file operations:
  + **Connect [server IP Port]:** Initiates connection from client to server with the specified files.
    - **Authentication:** Implement a secure authentication mechanism (e.g., username/password) to restrict access to the file sharing system. Do not transmit password in clear text. Encryption is recommended.
  + **Upload [server IP]:** Clients can upload files to the server. Server will prompt client if file already exists and requires user input to be overwritten.
  + **Download [filename]:** Clients can download files from the server. Server will respond with error message if file does not exist.
  + **Delete:** Clients can delete files from the server. Server will respond with error message if file is currently being processed or does not exist.
  + **Dir:** Clients can view a list of files and subdirectories in the server’s file storage path.
  + **Subfolder [{create | delete} path/directory]:** Clients can create or subfolders in the server’s file storage path
* **Performance Evaluation:** Collect and analyze performance metrics such as:
  + Upload and download data rates (MB/s) of upload and download
  + File transfer times
  + System response times
* **Error Handling:** Implement appropriate error handling mechanisms to gracefully handle exceptions and provide informative error messages to clients.

**Project Design**

* **Server-Side Application Design:**
  + Create a multithreaded server to handle multiple client connections simultaneously.
  + Implement data structures (e.g., file system, client connection pool) to efficiently manage files and client information.
  + Utilize logical file naming conventions that identifies the file type (ie text vs video) while avoiding duplicate file names. (ie TS001 for Text-Server file)
  + Implement file transfer logic using appropriate network protocols.
  + Handle authentication and authorization requests.
  + FTP is not permitted.
* **Client-Side Application Design:**
  + Implement a user interface for clients to interact with the file sharing system.
  + Implement network communication logic to connect to the server and send/receive file requests.
  + Handle file uploads, downloads, and directory operations.
  + Provide feedback to the user regarding the status of file transfers and operations.
* **Network Analysis Application Design:**
  + Implement a module that collects statistics from both server and client applications
  + Module should start with the server application.
  + Network statistics including upload/download data rates, file transfer times and system response times will be stored (ie dictionary or dataframe) to be reviewed offline. This data will be use for the report.

**Project Evaluation**

* **Performance Analysis:** Conduct experiments to measure the system's performance under various load conditions. Analyze the collected data to identify bottlenecks and areas for optimization.
* **Security Assessment:** Evaluate the system's security measures to ensure that it is protected against unauthorized access and data breaches.
* **Operation:** All testing should be performed from the client-side application issuing specific commands to produce the expected file operations.

**Project Submission**

* **Report:** Submit a comprehensive project report in doc or pdf, that includes:
  + Cover page that includes name of project, date, course section and each member’s name
  + Introduction and project objectives (1 page)
  + System architecture and design (1 -2 pages)
  + Implementation details (3 pages)
  + Experimental results and analysis presented in captioned diagrams, graphs and screenshots. Each should be supported with 1 – 2 paragraphs each. (3 -4 pages)
  + Problems faced (1 -2 paragraphs)
  + What was learned (1 – 2 paragraphs from each team member)
  + Individual contribution table (displays activity type and percentage of effort for each team member)
  + Conclusions and future work (3 paragraphs)
* **Code:** Submit well-commented and organized source code for the server, client and analysis components. This should be at minimum 3 python modules.
* **Video Presentation:** Prepare and narrate a concise presentation summarizing the project's key aspects and demonstrating the system's functionality specifically:
  + **Discussion of each component (server, analysis, client). Narration should be equally distributed to each team member.**
  + **Significant code snippets (not entire codebase). This should include:**
    - **Authorization method including obfuscation or encryption**
    - **Analysis method including how data is stored**
    - **Server-side logic**
    - **Client command input logic**
    - **Socket setup on client or server**
  + **Client operation showing connect, Upload, Download, Dir, and Subfolder actions.**
  + **Note, faces are not necessary, only screen recordings and audible narration is required.**
    - **Overall video submitted in .mp4 container with a time of 8-12 minutes. Use the Sim Lab for video assistance.**

### Reference

* <https://realpython.com/python-sockets/>
* <https://www.youtube.com/watch?v=TMbHg43as_E&list=PLhTjy8cBISErYuLZUvVOYsR1giva2payF&index=9>
* <https://www.thelearningpoint.net/computer-science/networking-an-introduction-to-client-server-programming--in-python>
* <https://www.networxsecurity.de/glossary-d1/n-d1/network-socket/>
* <https://www.bogotobogo.com/cplusplus/sockets_server_client.php>
* <https://cs.lmu.edu/~ray/notes/javanetexamples/>
* <https://www.educba.com/socket-programming-in-c-plus-plus/>
* <https://www.bogotobogo.com/cplusplus/sockets_server_client.php>
* <https://www.geeksforgeeks.org/how-to-encrypt-and-decrypt-strings-in-python/>