## Assignment Title: Peer-to-Peer File Sharing Application

**Objective:**

To create a simple peer-to-peer (P2P) file-sharing application using socket programming. This project will enhance understanding of networking concepts, socket communication, and multi-threading.

**Overview:**

Students will build a basic application that allows two users to connect over a network and share files directly. Each participant will take on a specific role (Client or Server) and will implement the necessary functionalities for file transfer.

**Roles:**

1. **Client**: Responsible for sending files to the server and requesting files.
2. **Server**: Responsible for receiving files from the client and serving files upon request.

**Requirements:**

1. **Socket Setup**:
    - Use TCP sockets for reliable communication.
    - Implement error handling for connection issues.
2. **File Transfer Protocol**:
    - The client should be able to:
        - Send a file to the server.
        - Request a file from the server.
    - The server should be able to:
        - Receive files from the client and save them locally.
        - Send requested files back to the client.
3. **User Interface**:
    - Create a simple command-line interface for both the client and server.
    - Provide options for:
        - Sending a file
        - Receiving a file
        - Viewing available files (server-side)
4. **Multi-threading (Server Side)**:
    - Implement multi-threading on the server to handle multiple clients (even though this assignment is primarily for two participants).
5. **File Integrity**:
    - Implement basic checks (e.g., file size comparison) to ensure the file has been transferred correctly.

**Deliverables:**

1. Source code for both the client and server.
2. A brief report (2-3 pages) detailing:
    - The design and architecture of the application.
    - Challenges faced and how they were resolved.
    - Instructions on how to run the application.

**Evaluation Criteria:**

1. **Functionality** (40%): Does the application meet all specified requirements?
2. **Code Quality** (30%): Is the code well-structured, documented, and easy to read?
3. **User Interface** (20%): Is the command-line interface user-friendly?
4. **Report** (10%): Clarity and completeness of the report.

**Suggested Technologies:**

- Python (using the `socket` and `threading` libraries)
- Java (using `java.net` and `java.io` packages)
- C# (using `System.Net.Sockets`)