**BCS221 – Communication Networks - Spring-2025**

**Assignment-1 - [4 marks] – Maps to CLO-3**

**Design and Develop a Socket Programming based Networking Application**

**Due Date: 9 a.m., Monday, 28th April, 2025**

**Objective:**

The objective of this assignment is to enable students to develop skills in designing and implementing a networking application based on socket programming. The students can develop the application in a programming language of their choice including Python, Java and C/C++.

**Grading Rubric:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Assessment Criteria | Unsatisfactory (1-4) | Needs Improvement  (5-6) | Meets Expectations  (6-8) | Exceeds Expectations  (9-10) | Name:  Student ID: | Name:  Student ID: | Name:  Student ID: |
| Design a solution for a computing system | Design is unclear or missing, code is incomplete or not working, poorly written, and the student shows little or no understanding during Q&A. | Design is basic, code is partially working with limited structure, documentation is minimal, and student demonstrates partial understanding of their work. | Design and implementation are functional and correct, code is structured and well-documented, and the student explains the logic clearly during Q&A. | Design is well-thought-out, code is fully functional with clean structure and documentation, and the student demonstrates full command over the solution. |  |  |  |

|  |
| --- |
| **Deliverables:**   * Paste the complete and thoroughly commented source-code of both the client and the server applications. * Test your application by first running the client and server applications on the same machine, and then on different machines. * Screenshots of all the outputs |
| **Mention which of the two applications have you developed:**  s |
| **Paste your Code in this section:**  . |
| **Paste all the screenshots of your tests with description below:**  . |

**Description:**

Your task is to develop a socket-programming based networking application. You may develop **either a file transfer application or a multi-user chat application** the description of both of which is given below.

**Tutorials:** Read the following tutorials to learn how to extend the application we had seen in the lab on socket programming.

* [Socket programming in Java: A tutorial](https://www.infoworld.com/article/2853780/socket-programming-for-scalable-systems.html)
* [A Guide to Java Sockets](https://www.baeldung.com/a-guide-to-java-sockets)
* [Reading/Writing Files in Java](https://www.geeksforgeeks.org/file-handling-java-using-filewriter-filereader/)

**Option 1 – File Transfer Application**

**File Transfer Application:** Develop a client-server application where clients can upload/download files to/from the server. Clients can request specific files from the server, and the server can send the requested file to the client. You may enhance the code shared in the lab to add functionality to upload and download files. Note that you have to understand file I/O **streams** studied already in Intro. to Computing Science II.

**Server Side:**

1. **Set Up the Server Socket**: Create a Java class for the server. Set up a ServerSocket object to listen for incoming client connections on a specific port.
2. **Accept Client Connection**: Use the accept() method of the ServerSocket to accept incoming client connections. Once a client connects, accept the socket connection and create a separate thread to handle communication with that client.
3. **Receive Client Requests**: Within the thread handling the client connection, receive requests from the client. The client may request to upload a file or download a file.
4. **Handle Upload Request**: If the client requests to upload a file, the server should receive the file data from the client and save it to a specific directory on the server's file system. Ensure proper error handling and file management (e.g., checking for file existence, permissions, etc.).
5. **Handle Download Request**: If the client requests to download a file, the server should read the requested file from the file system and send it back to the client over the socket connection.
6. **Close Connections**: After handling the client's request, close the socket connection and any associated resources. Optionally, the server can continue listening for new client connections.

**Client Side:**

1. **Set Up the Client Socket**: Create a Java class for the client. Set up a Socket object to connect to the server's IP address and port.
2. **Send Requests to the Server**: Connect to the server and send requests to upload or download files. For uploading, send the file data to the server. For downloading, specify the file name to be downloaded.
3. **Handle Server Responses**: Receive responses from the server after sending requests. If downloading a file, receive the file data from the server and save it to a specified location on the client's file system.
4. **Close Connection**: After completing the file transfer operation, close the socket connection and release any associated resources.

**Additional Considerations:**

By following these steps, you can develop a simple and functional File Transfer Application using socket programming in Java.

**Option 2 – Multi-Client Chat Application with Client Names**

**Description:**

1. **Objective:** The objective of this assignment is to create a multi-client chat application where multiple clients can connect to a server and communicate with each other. Each client will have a unique name, and the server will facilitate communication among all connected clients.
2. **Functionality:**
   * **Client Connection:** Clients will connect to the server using TCP/IP sockets.
   * **Unique Names:** Each client will provide a unique name upon connection to the server. This name will be used to identify the client in the chat.
   * **Chat Communication:** Once connected, clients can send messages to the server, which will then broadcast the message to all connected clients. Messages will include the sender's name for identification.
   * **Multi-Client Support:** The server will handle multiple client connections simultaneously, allowing for a multi-user chat environment.
   * **Disconnect Handling:** Proper handling of client disconnections will be implemented to ensure smooth operation of the chat application.
3. **Implementation:**
   * **Server:** The server will be implemented in Java using the ServerSocket class to listen for incoming client connections. Each client connection will be handled in a separate thread to enable concurrent communication with multiple clients. The server will maintain a list of connected clients along with their names.
   * **Client:** The client application will also be implemented in Java using the Socket class to connect to the server. Upon connection, the client will provide a name, which will be sent to the server for identification. The client will continuously listen for messages from the server and display them to the user. The client will also allow the user to input messages to be sent to the server.