NetClassroom

Building a Networked Classroom System

CSE 3111: Computer Networking Project

Batch: 29 3rd Year 1st Semester 2024

Date

January 8, 2026

Submitted by

Aditto Raihan (Roll-09)
Jubair Ahammad Akter (Roll-59)

Department of Computer Science and Engineering
University of Dhaka

Contents

1 Overview

In this project, we design and implement **NetClassroom**, a TCP-based networked classroom system that allows teachers and students to interact over a custom client–server architecture.

NetClassroom supports text-based communication, assignment management, and basic classroom coordination over a reliable TCP connection. The main focus of this project is to apply concepts from the computer networking course (CSE3111) in a practical system.

1.1 Background

Traditional learning management systems often hide low-level networking details behind web frameworks and cloud services. In contrast, NetClassroom is built directly on top of TCP sockets, so that connection management, message framing, and file transfer are explicitly controlled in our code.

1.2 Scope

The current scope of NetClassroom includes:

- A central TCP server that manages multiple concurrent clients.
- Role-based clients (teacher / student).
- Public chat, private messaging, and assignment handling.
- Basic file transfer support for assignment submission.

2 Motivation

The key motivations behind developing NetClassroom are:

- To build a **custom classroom system** using low-level networking primitives instead of ready-made web platforms.
- To practically **apply networking concepts** such as sockets, reliable data transfer, flow control, and congestion control.
- To gain hands-on experience with **multi-client TCP communication** and basic protocol design.
- To create a simple but functional system that can be extended in future to a full-featured learning platform.

3 Problem Statement

Modern online classrooms require:

1. Real-time text communication between teacher and students.
2. A simple way to create, distribute, and collect assignments.
3. Support for both individual and group activities.

However, most existing tools are heavy-weight and depend on complex web stacks. Our problem can be stated as:

How can we design and implement a lightweight, TCP-based classroom system that supports messaging, assignment management, and basic session handling while explicitly demonstrating key networking concepts?

4 Design Goals and Objectives

The main objectives of NetClassroom are:

- **TCP-based platform:** Design a classroom system that uses `java.net.Socket` and `ServerSocket` for all communication.
- **Role-based access:** Support separate functionality for teachers and students.
- **Real-time chat:** Enable classroom-wide messages and private messages.
- **Assignment management:** Allow teachers to create assignments (solo and group) and receive submissions.
- **Scalable design:** Use multi-threading to handle multiple concurrent clients.
- **Extensibility:** Keep the design modular so that future features (encryption, database, better UI) can be integrated.

5 Project Features

5.1 Core Functionalities

- User login with role selection (Teacher / Student).
- Public classroom chat visible to all connected users.
- Private messaging between:
  - Student $\leftrightarrow$ Student

  - Teacher ↔ Student

- Assignment management:
  - Creation of solo and group assignments.
  - Group creation and management for group assignments.
  - File-based assignment submission.
- Submission tracking for teachers (who submitted, when, and file details).

5.2 Interface

- **Terminal Client:** Initial version based on command-line input and output.
- **JavaFX GUI Client:** Enhanced client with graphical interface for better usability.

6 Block Diagram / Work Flow Diagram

6.1 High-Level Workflow

At a high level, NetClassroom follows this workflow:

1. Client connects to the server using a TCP socket.
2. User logs in and selects a role (teacher or student).
3. Server creates and maintains a session for the connected client.
4. Client sends requests (chat, private message, assignment operation) using a custom, HTTP-like text protocol.
5. Server parses the request, performs the required operation, and sends a structured response.
6. Messages and files are delivered reliably via TCP.
7. Server either broadcasts the message to all users or routes it to specific target(s).

6.2 System Block Diagram

Below is an example TikZ block diagram. You can modify/edit this as needed.

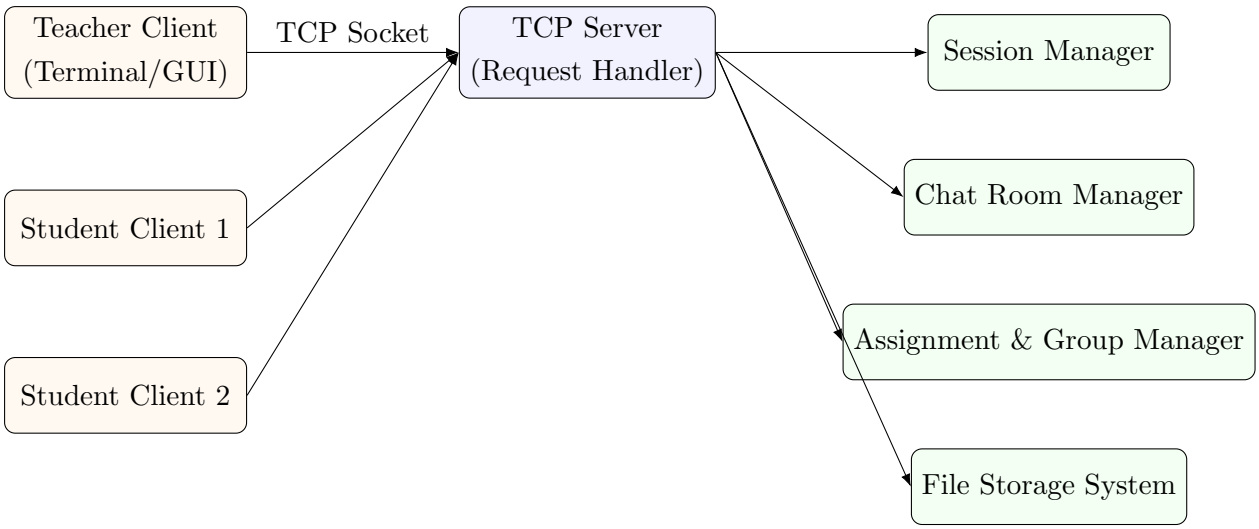

Figure 1: High-level block diagram of NetClassroom system.

7 Tools & Technologies

7.1 Programming Language

- Java (for both server and client implementation).

7.2 Networking

- `java.net.ServerSocket` for server-side TCP listening.
- `java.net.Socket` for client-side connections.

7.3 User Interface

- JavaFX framework for graphical client.
- Standard console I/O for terminal client.

7.4 Development Environment

- IDEs: IntelliJ IDEA / VS Code.
- Build and run: `javac`, `java`, Gradle/Maven (if used).

7.5 Others

- Custom text-based protocol inspired by HTTP-style request/response.
- File I/O for reading/writing assignment submission files.

8 Applied Networking Concepts

In this section, we explain how networking concepts from the course are applied in NetClassroom.

8.1 Socket Programming

- Persistent TCP sockets are used to maintain an end-to-end connection between each client and the server.
- The server listens on a fixed port and accepts incoming connections using a blocking `accept()` call.

8.2 Client–Server Architecture

- The server is the central coordination point and is responsible for:
  - authenticating users,
  - routing messages (broadcast or private),
  - managing assignments and groups.
- Clients act as endpoints that send requests and render responses.

8.3 Reliable Data Transfer (RDT)

- TCP guarantees ordered, reliable delivery of bytes.
- All chat messages, control commands, and file chunks for assignments are sent over TCP streams.
- Our application protocol adds simple delimiters/headers to separate logical messages inside the TCP byte stream.

8.4 Flow Control

- TCP's built-in flow control ensures that the sender does not overwhelm the receiver's buffer.
- During large file transfers, the OS-level TCP stack automatically adjusts the sending rate based on receiver's window size.

8.5 Congestion Control

- TCP congestion control adapts to network conditions using algorithms like slow start and congestion avoidance (implementation in OS).
- When multiple students upload files simultaneously, congestion control mechanisms help avoid network collapse.

8.6 Session Management

- The server maintains a session object for each connected user.
- The session stores:
  - username and role (teacher/student),
  - current room/classroom state,
  - socket I/O streams.

8.7 Multiplexing and Multi-threading

- The server uses a separate thread (or runnable) per client connection.
- This allows multiple clients to send/receive messages concurrently.

9 Implementation Details

This section describes how the system is implemented, including the main modules and algorithms.

9.1 Overall Architecture

NetClassroom follows a modular architecture:

- **Server Core:** accepts connections, spawns client handlers.
- **Client Handler:** per-connection thread that reads, parses, and processes client requests.
- **Managers:**
  - Session Manager
  - Chat Room Manager
  - Assignment & Group Manager
  - File Storage Handler
- **Clients:** terminal-based and JavaFX-based front-ends.

9.2 Custom Protocol Design

Requests and responses follow a simple text-based, HTTP-like format. For example:

Sample Request (Chat Message):

```
ACTION: PUBLIC_CHAT
SENDER: student01
CONTENT-LENGTH: 23

Assalamu Alaikum Sir!
```

Sample Response:

```
STATUS: OK
MESSAGE: Broadcast delivered
```

This structure makes it easier to parse commands and add new actions.

9.3 Server-side Algorithm (High Level)

Algorithm 1: Server Main Loop

1. Create a `ServerSocket` on the chosen port.
2. While `true`:
   (a) Call `accept()` to wait for a new client.
   (b) Wrap the returned `Socket` with input/output streams.
   (c) Create a new `ClientHandler` thread and start it.

Algorithm 2: ClientHandler Run Method

1. Read a request from the input stream.
2. Parse headers and body to identify the action.
3. Switch on the action type:
   - `LOGIN` → authenticate & create session.
   - `PUBLIC_CHAT` → broadcast to all sessions.
   - `PRIVATE_CHAT` → deliver to specific session.
   - `CREATE_ASSIGNMENT` → update assignment list.
   - `SUBMIT_ASSIGNMENT` → receive file and store.
4. Construct a response and send it back to the client.
5. Repeat until the client disconnects.

9.4 Client-side Implementation

- Terminal client uses a main loop that:
  1. Reads user commands from console.
  2. Formats them into protocol requests.
  3. Sends them to the server via the TCP socket.
  4. Reads and displays server responses.
- GUI client uses JavaFX controllers to:
  - Bind text fields and buttons.
  - Update the chat area and assignment views when new messages arrive.

10 Result Analysis

In this section, we present major input/output screenshots and briefly discuss the behavior of the system.

10.1 Login and Role Selection

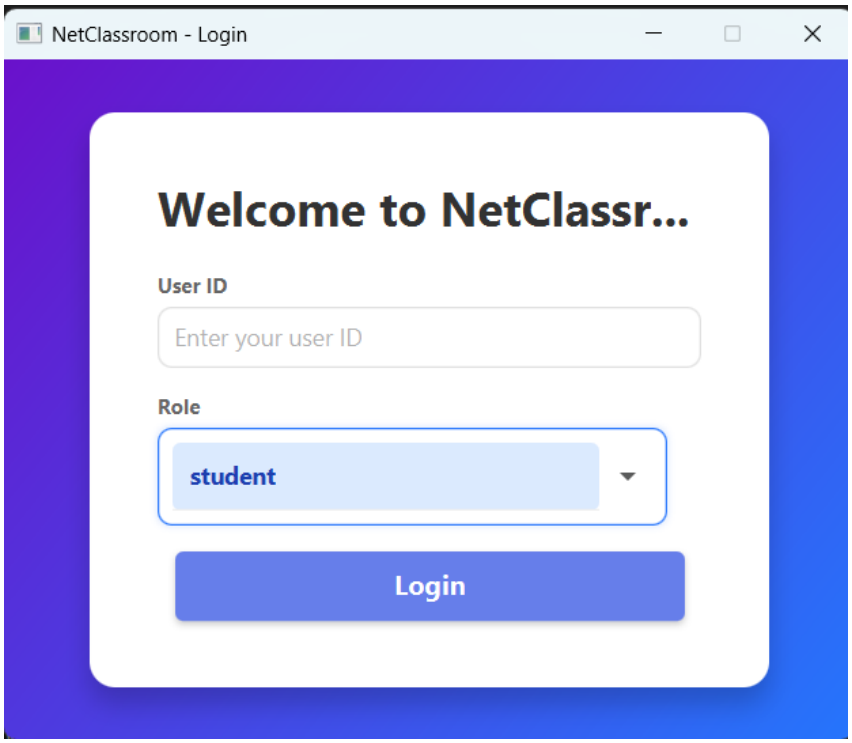

Figure 2: Login screen for NetClassroom (teacher/student role selection).

10.2 Public Chat Interface

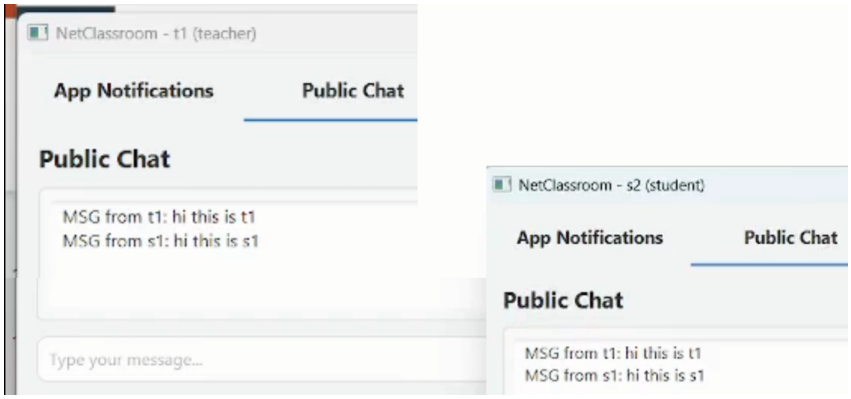

Figure 3: Public classroom chat view showing messages from multiple users.

10.3 Assignment Creation and Submission

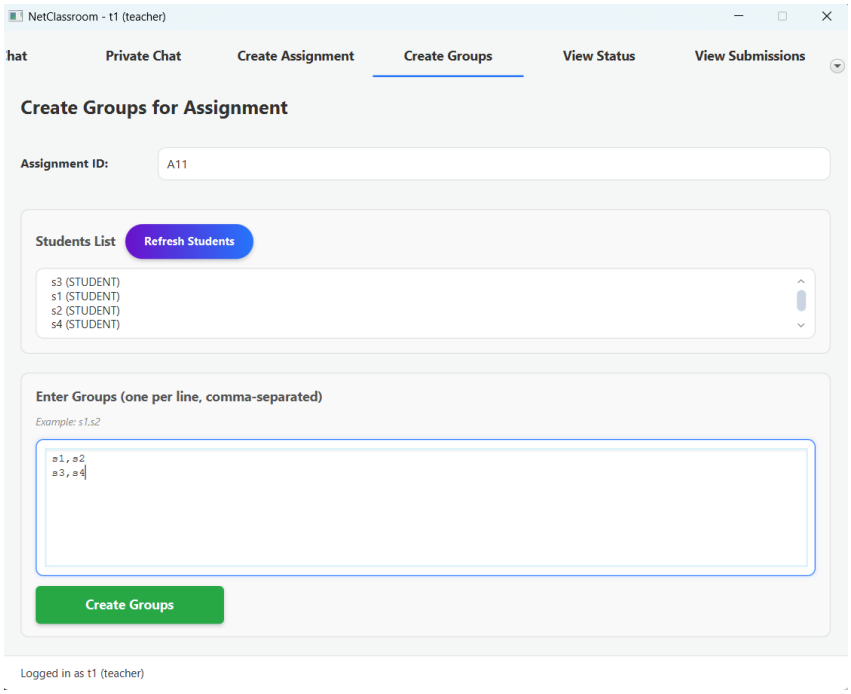

Figure 4: Teacher creating an assignment and student submitting a file.

10.4 Observation and Discussion

Here you can briefly write your observations, for example:

- The system successfully handled N concurrent clients without crashing.
- Chat messages were delivered reliably and in order.
- File submissions were correctly saved and listed for the teacher.

11 Summary of the Project

NetClassroom demonstrates how a practical classroom management system can be built on top of basic TCP socket programming.

We implemented:

- A multi-threaded TCP server.
- Role-based teacher and student clients.
- Public and private chat functionalities.
- Assignment creation, group formation, and file-based submission.

Throughout the project, we directly applied several networking concepts such as reliable data transfer, client–server design, flow control, congestion control, and session management.

12 Limitations and Future Plan

12.1 Current Limitations

- No encryption: communication is currently in plain text.
- No dedicated database: data is stored in memory and simple files.
- Limited user interface design, especially for the terminal client.
- Scalability is not evaluated for very large numbers of users.

12.2 Future Enhancements

- Integrate TLS/SSL to provide secure, encrypted communication.
- Add a relational or NoSQL database for persistent storage of users, messages, and submissions.
- Improve the JavaFX GUI with better layouts, notifications, and error handling.
- Add attendance tracking and analytics for teachers.
- Implement reconnection logic and better fault tolerance.

Source Code Repository

The complete source code of the NetClassroom Networking Project is available publicly on GitHub:

- **GitHub Repository:**
  https://github.com/Jubair-Adib/NetClassroom$_N etworking_P roject$

  To download the project:
  [basicstyle=,language=bash] `git clone https://github.com/Jubair-Adib/NetClassroom`$_N etworkir$
  On Windows (with JavaFX 17 installed at 'C:-sdk-17.0.17'), the project can be compiled and run using:

Compile (Server + Clients) [basicstyle=,language=bash] `javac -d out --module-path "C:-sdk-17.0.17" `$^--add-modulesjavafx.controls-sourcepathsrc^s rc/client/*.javasrc/protocol/*.javasrc/server/*.javasrc/utils/*.java$

Run TCP Server [basicstyle=,language=bash] `java -cp out server.TCPServer`

Run JavaFX Classroom Client [basicstyle=,language=bash] `java --module-path "C:-sdk-17.0.17" `$^- - add - modulesjavafx.controls - cpoutclient.ClassroomApp$

All Java source files, configuration, and sample uploads used in this report are contained in the above repository.