Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

Project Title

(Try to choose a catchy title. Max 20 words).

StreamLine Chat - Seamless conversations through multithreaded TCP connectivity

Student/Team Information

Team Name:	
Team member 1 (Team Lead): Akshit – 220111979 akshitsharma02003@gmail.com	
Team member 2: Shobhit Bisht – 220111631 shobhitbisht2703@gmail.com	

Team member 3: Tanish Rawat – 220111278 tanishrawat10@gmail.com	
Team member 4: RajVardhan Singh Chauhan – 220111066 rajvardhansevenb@gmail.com	

PROJECT PROGRESS DESCRIPTION

Project Abstract

(Brief restatement of your project's main goal. Max 300 words).

StreamLine Chat is a high-performance, multithreaded TCP chat application written in C++ using the WinSock2 API. It supports real-time, bidirectional communication through concurrent send/receive threads, ensuring a seamless user experience. Designed with educational intent, the application avoids bloated frameworks and focuses on clear architecture, low resource use, and natural conversation flow. The console-based interface and clean separation of concerns make this a lightweight, extensible platform for further development into secure, GUI-based, or cross-platform tools.

Updated Project Approach and Architecture

(Describe your current approach, including system design, communication protocols, libraries used, etc. Max 300 words).

Updated Project Approach and Architecture

The project utilizes a multithreaded design on the client and server side:

Client: Uses CreateThread to spawn a send and a receive thread simultaneously for bidirectional communication.

Server: Accepts multiple clients using threads and maintains communication through a master fd_set list.

Technologies Used: C++, WinSock2, Windows threading API, and Windows OS as the base platform.

Architecture: Console-based UI, modular socket management, with future enhancements planned for encryption,

reconnection logic, and graphical interfaces.

Tasks Completed

(Describe the main tasks that have been assigned and already completed. Max 250 words).

Task Completed	Team Member
Client-side socket setup and messaging	Akshit , Shobhit
Server-side multiclient handler	Raj Vardhan Singh Chauhan , Tanish
Multithreaded communication handlers	Akshit , Shobhit
Input/output prompt management	Akshit

Challenges/Roadblocks

(Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words

Thread synchronization and console I/O overlap

Managing multiple client sockets efficiently on the server

Creating a user-friendly input experience with live incoming message updates These were solved using prompt reprinting, and careful thread management.

Tasks Pending

(Describe the main tasks that you still need to complete. Max 250 words).

Task Pending	Team Member (to complete the task)
Add reconnection logic	Akshit
Implement command system	Raj Vardhan
Add server-side logging	Akshit Shobhit
Explore GUI integration or encryption	All Members

Project Outcome/Deliverables

(Describe what are the key outcomes / deliverables of the project. Max 200 words).

Functional TCP chat application with client and server code

Bidirectional real-time communication with multithreading

Clean console-based interface

Educational demonstration of C++ sockets and thread usage

Foundation for future work including GUI and encryption

Progress Overview

(Summarize how much of the project is done, what's behind schedule, what's ahead of schedule. Max 200 words.)

We have completed the core chat engine: client/server socket handling, multithreading, and console interaction.

Approximately 75% of the project is complete.

Final week focus: command system, logging, minor bug fixes, and optional GUI/encryption prototyping.

Codebase Information

(Repository link, branch, and information about important commits.)

GitHub Repository: https://github.com/your-repo-linkBranch: mainThe codebase for StreamLine Chat is maintained on GitHub under the "main" branch. The repository includes all core modules of the project, including the multithreaded TCP socket setup for both client and server applications. Notable commits include the successful integration of send and receive handlers through Windows threads, server-side multicasting using fd_set, and input/output prompt synchronization. These milestones enabled real-time message flow without blocking the user interface, making the experience fluid and responsive. The client program initializes with user input and runs continuous threads for messaging, while the server handles multiple concurrent connections with consistent broadcast functionality. The modular design ensures that upcoming features—such as encryption, reconnection logic, and GUI extensions—can be integrated smoothly. All team members regularly commit updates, and comments are provided throughout the codebase to facilitate understanding and collaborative development.

Testing and Validation Status

(Provide information about any tests conducted)

Test Type	Status (Pass/Fail)	Notes
Send/Receive threads Server multicasting Client disconnect handling	Pass Pass Pass	Verified smooth operation under message load In development

Deliverables Progress

(Summarize the current status of all key project deliverables mentioned earlier. Indicate whether each deliverable completed, in progress, or pending.)

Client Chat App | Completed
Server App| Completed
Multithreading | Completed
Reconnection Logic | In Progress
Logging| Pending
Command System | In Progress
GUI/Encryption| In Progress