

---

University of Dhaka  
Department of Computer Science and Engineering  
CSE 3111



A Unified Real-Time Collaboration & Communication Platform

Submitted By

H.M. Mehedi Hasan (13)  
MD. Abu Bakar Siddique (47)

---

# Problem Domain & Motivations



## Disconnected Tools

Teams use **multiple apps for messaging, file exchange, and code sharing**



## Code Comparison

Students **struggle to compare code and troubleshoot together**



## Delays

Switching between tools causes **delays in collaboration**

---

# OBJECTIVES<sup>+</sup>

## Implement custom TCP protocols for different services

Chat, Editor, File Transfer, Code Execution, and Room Management servers on dedicated ports

## Apply TCP congestion control algorithms with RTT estimation

Tahoe & Reno slow start, congestion avoidance, and Jacobson/Karels timeout calculation

## Ensure secure sandboxed code execution

Docker-based isolation with resource limits, network restrictions, and language-specific timeouts

## Enable real-time collaboration features

Live code synchronization, instant message broadcasting, and persistent chat history

# Key Features

## Real-Time Chat

- Room-based chat with unique **4-digit codes**
- **Rate limit:** max 5 messages every 2 seconds
- Supports **text, emojis & base64 images**

## Collab Editor

- Live code sync for multiple users
- Supports **Python, C, C++, Java**
- Auto-save every **2 seconds**
- Last-write-wins conflict handling

## Code Execution Engine

- Isolated Docker execution per request
- Limits: no network, 256 MB RAM, 0.5 CPU
- Full **stdin/stdout/stderr capture**
- Supports **compilation (C, Java)**

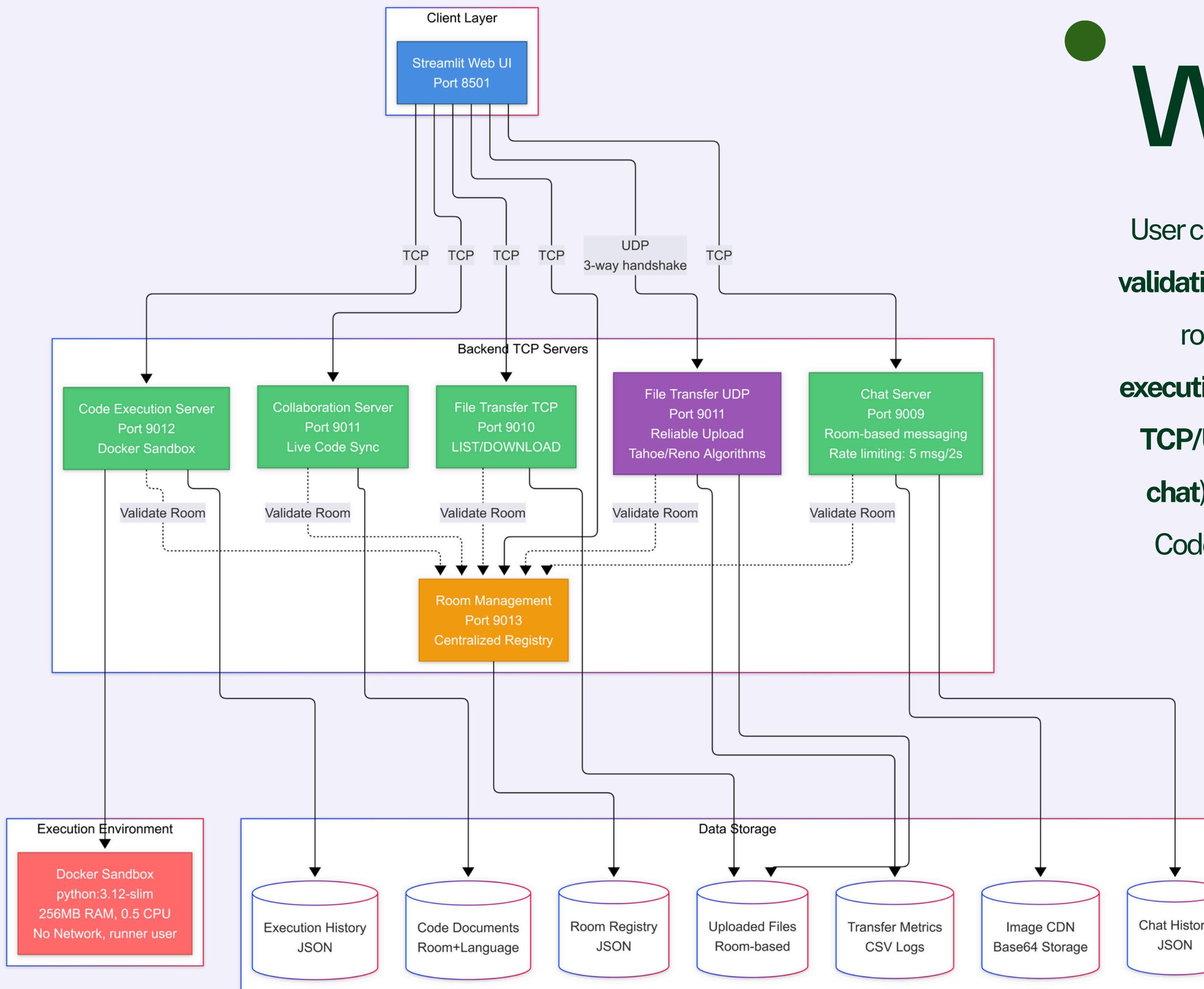
## File Transfer

- TCP Tahoe & Reno selection
- **4 KB chunks with ACKs**
- Per-chunk real-time RTT

## System Dashboard

- Real-time server status monitoring
- RTT & CWND visual charts
- Historical network performance data

# Workflow



User connects via **Streamlit UI** → **Room validation** at central registry → Requests routed to specialized services (**code execution, collaboration, file transfer** via **TCP/UDP with Tahoe-Reno algorithm, chat**) → Output persisted in storage → Code runs securely in **isolated Docker sandbox**.

# Tools & Technologies

## Languages

- Python 3.10+ (Core)
- C/C++ (Execution)
- Java (Execution)

## Frontend

- Streamlit 1.36.0
- Pandas
- Matplotlib

## Backend

- Socket Programming  
(Pure Python TCP)
- Threading (Concurrent connections)

## Infrastructure

- Docker 7.0.0
- python:3.10-slim image
- GCC/G++ Compiler
- OpenJDK

# Applied Networking Concepts

## Custom TCP/UDP Socket Connections

- **Chat (9009):** HELLO, CREATE\_ROOM, JOIN\_ROOM, MSG, IMG\_SEND, BYE
- **File TCP (9010):** LIST, DOWNLOAD with binary streaming
- **File UDP (9011):** SYN/SYN-ACK/ACK handshake, DATA/ACK, FIN/FIN-ACK
- **Collab (9011):** JOIN, SET, GET, USERS for document sync
- **Room Mgmt (9013):** CREATE, JOIN, LEAVE, EXISTS, LIST

## Flow Control & Rate Limiting

- Chat: **Max 5 messages per 2 seconds per user**
- File: **4KB chunk size with sliding window**
- In-flight packets limited to  $\min(\text{CWND}, \text{RWND})$
- Buffered I/O for efficient data handling

# Applied Networking Concepts

## RTT Estimation

- $\text{RTT} = \text{T\_ACK} - \text{T\_SEND}$  (per-chunk measurement)
- $\text{SRTT} = (1-0.125) \times \text{SRTT} + 0.125 \times \text{RTT}$
- $\text{RTTVAR} = (1-0.25) \times \text{RTTVAR} + 0.25 \times |\text{SRTT}-\text{RTT}|$
- $\text{RTO} = \text{SRTT} + 4 \times \text{RTTVAR}$  (retransmission timeout)

## Congestion Control (Tahoe & Reno)

- **Slow Start:** CWND grows exponentially ( $\text{cwnd} += 1$  per ACK)
- **Congestion Avoidance:** Linear growth ( $\text{cwnd} += 1/\text{cwnd}$  per ACK)
- **Tahoe:** On 3 dup ACKs  $\rightarrow \text{ssthresh} = \text{cwnd}/2$ ,  $\text{cwnd} = 1$
- **Reno Fast Recovery:**  $\text{cwnd} = \text{ssthresh} + 3$ , then  $\text{cwnd} = \text{ssthresh}$

# Applied Networking Concepts

## Reliable Data Transfer (UDP)

- **3-way handshake:** SYN->SYN-ACK->ACK
- **Cumulative ACKs:** Server ACKs highest in-order seq
- **Sliding window with sequence number tracking**
- **4-way termination:** FIN->FIN-ACK for graceful close

## Threading & Concurrency

- **Thread-per-client:** Daemon thread per connection
- Lock-based synchronization for shared state
- Separate I/O streams prevent race conditions
- Background polling for real-time updates

# User Interface

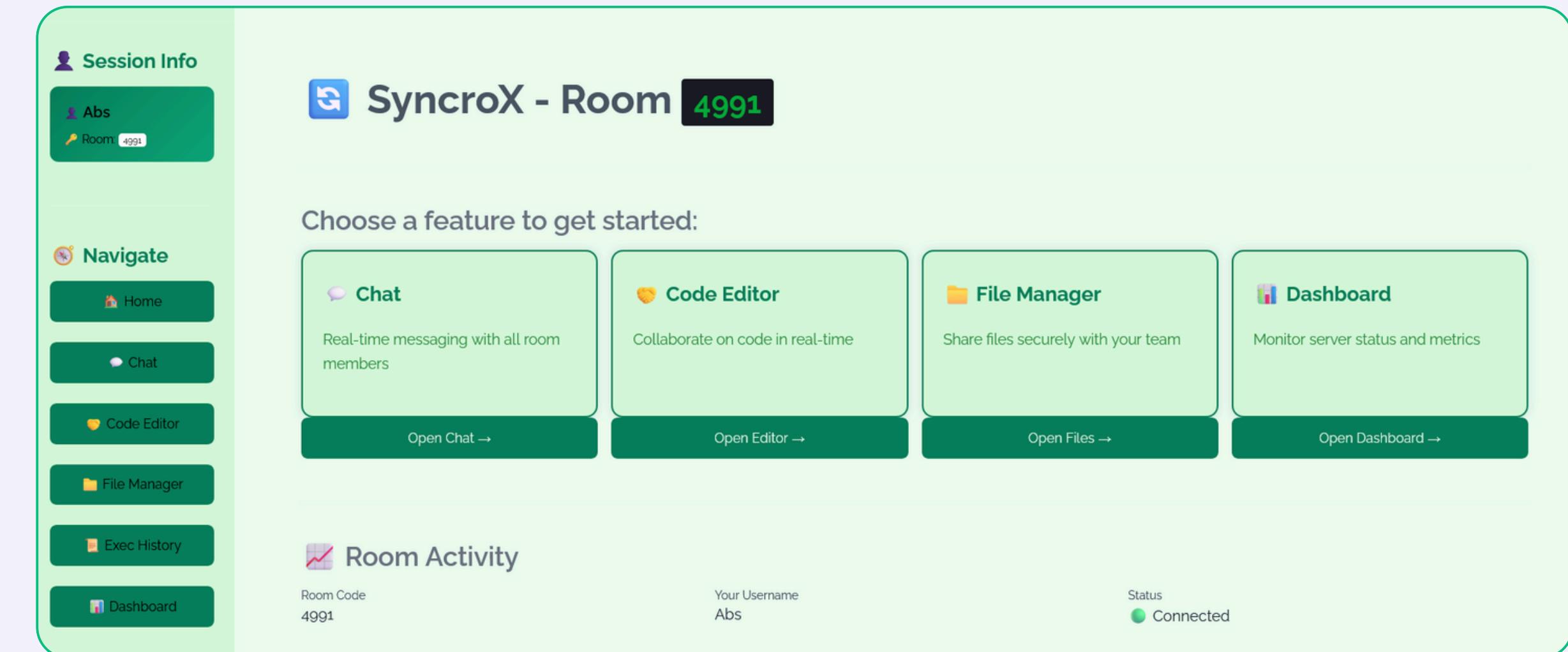
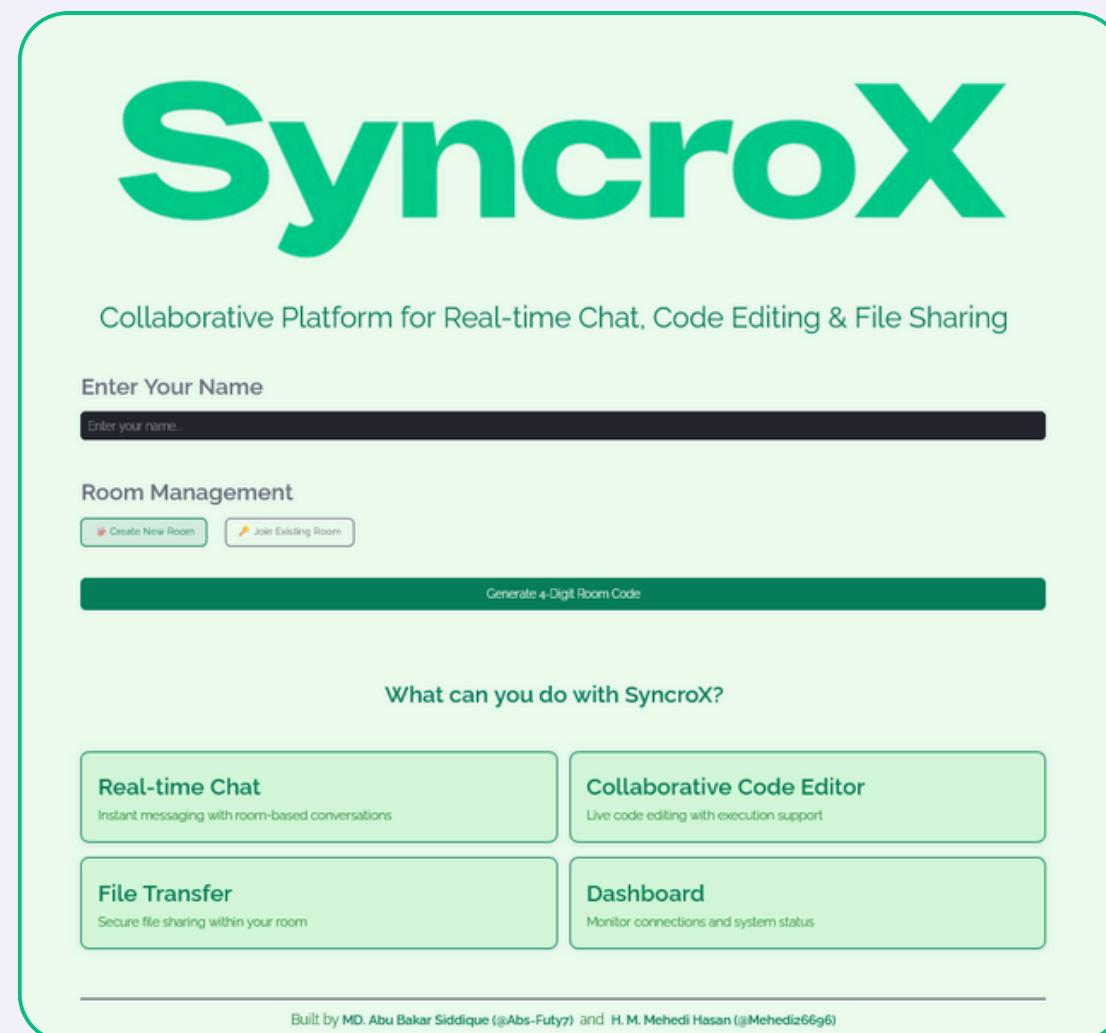


Figure: Landing Page

Figure: User Main Page

# User Interface

## Real-time Chat

Room: 7151 · User: Mehedi

The screenshot shows a real-time chat interface. At the top, there's a header with a speech bubble icon and the text "Real-time Chat". Below it, the room number "7151" and user name "Mehedi" are displayed. The main area contains a message from "ABS" saying "hello" at 23:18:14. Below that is a "local-demo" card with the status "running". The card displays the command "D:\Test Tools\testing\venv\Scripts\python.exe" and arguments "D:\Test Tools\testing\server.py". A timestamp "23:18:26" is shown next to the card. At the bottom, there's another message from "ABS" with two smiley faces at 23:18:42. A text input field with placeholder "Type your message" and a "Send" button are at the very bottom.

Figure: Chat Page

## File Transfer

Room: 7151 · User: Mehedi

The screenshot shows a file transfer interface. At the top, there's a header with the text "File Transfer". Below it, the room number "7151" and user name "Mehedi" are displayed. The main area has a teal header bar with the text "Upload and download files shared within your room". On the right, there's a "Congestion Control" section with "reno" and "tahoe" options. The central part shows an "Upload File" section with a "Drag and drop file here" area containing "workflow.png" (size 1,619,893 bytes). A "Browse files" button is available. A green progress bar indicates the file was uploaded successfully. Below this, a message says "Metrics logged to data/metrics/room\_4991\_file\_metrics.csv". At the bottom, there's a "Files in This Room" section showing "1 file(s) in room 4991". The file listed is "workflow.png" (size 1,619,893 bytes), created on 2026-01-07T20:25:55, with a "Download" button.

Figure: File Transfer Page

# User Interface:

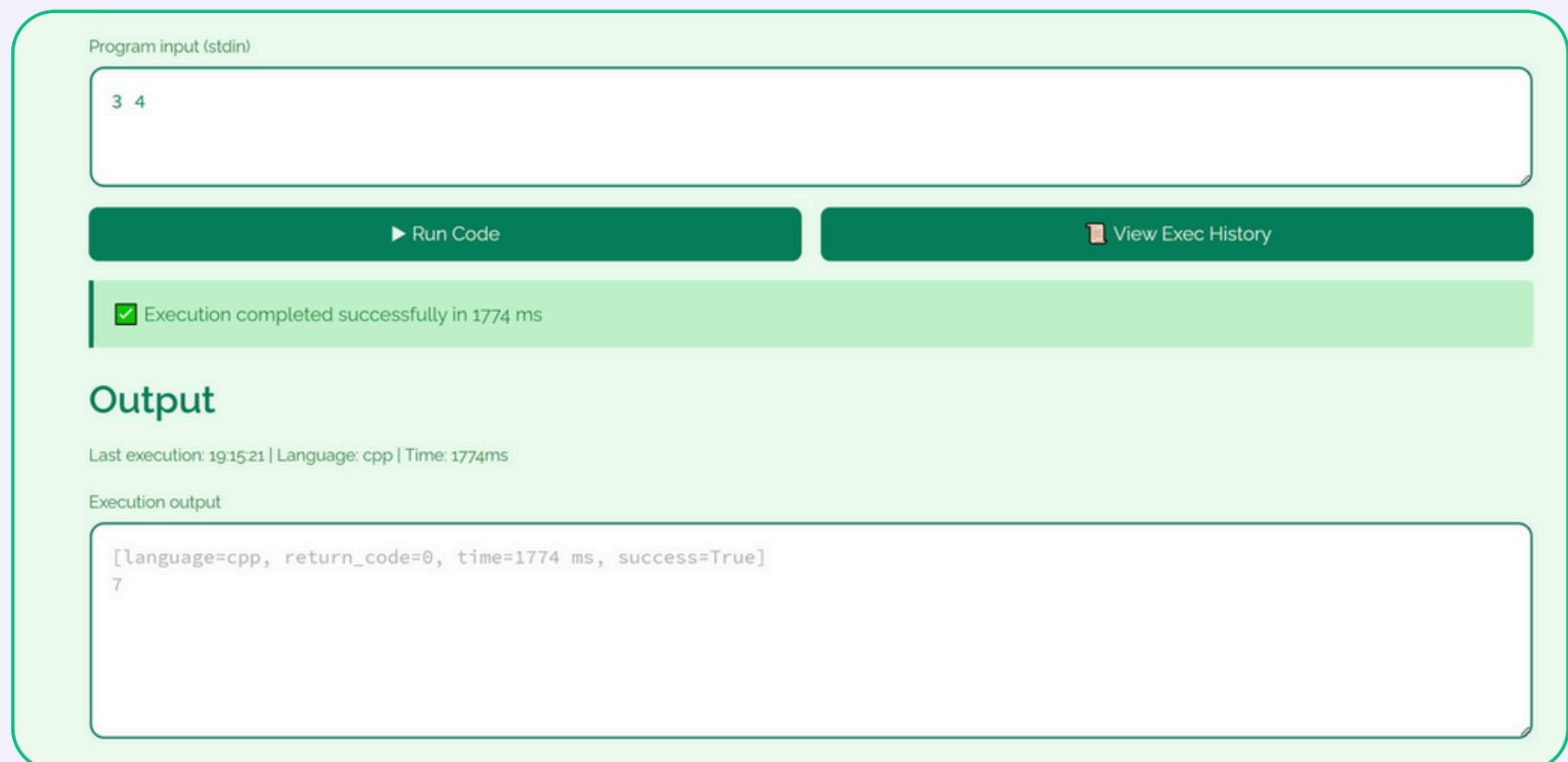
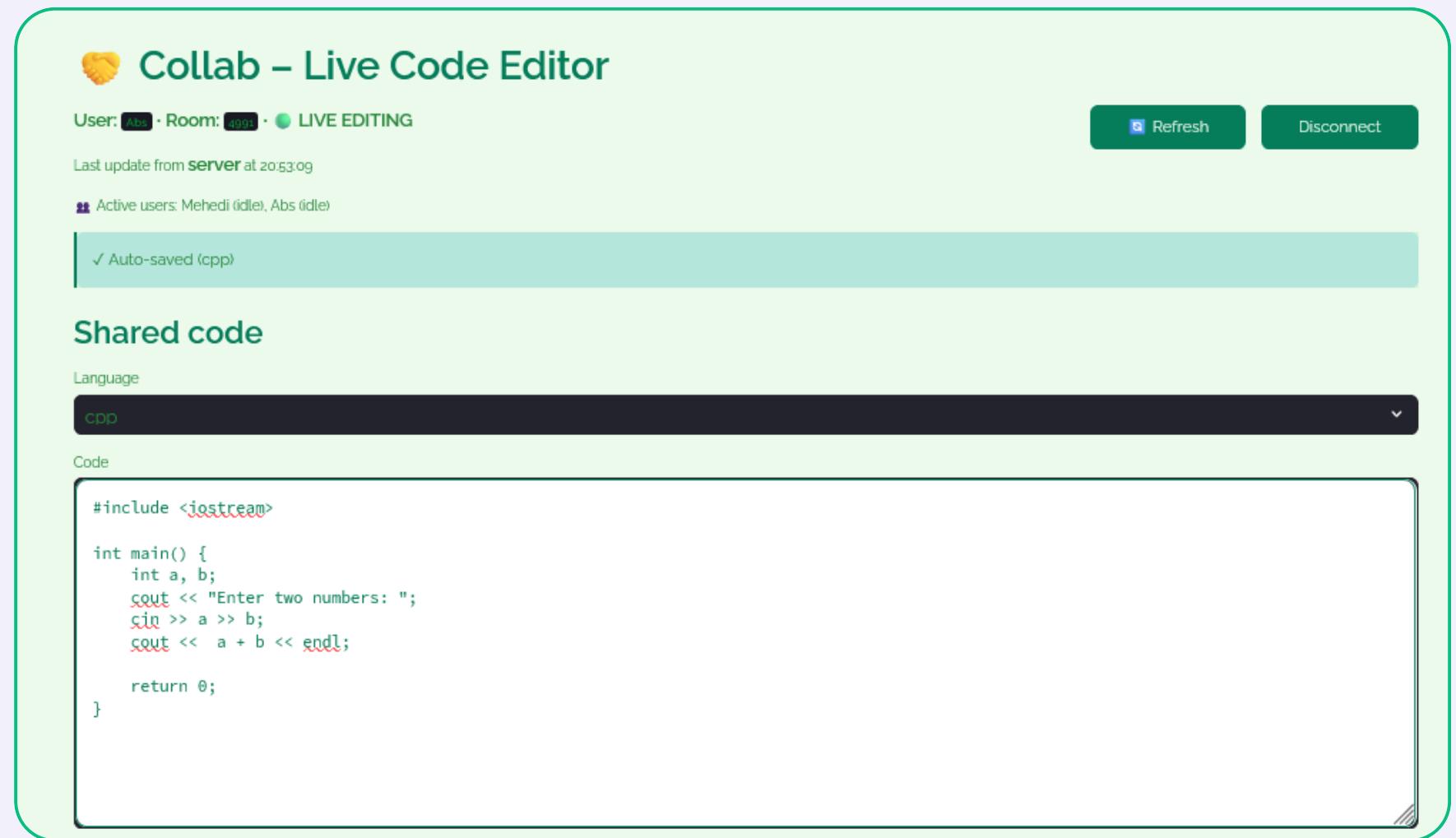


Figure: Live Code Editor Output

Figure: Live Code Editor Page

# User Interface:

## Execution History

View all code executions across rooms, users, and languages. Data is persisted to disk.

### Filters

Room: 4991 | Language: All Languages | User: All Users | Max Records: 50

### Statistics

Total Executions	Successful	Failed	Success Rate	Avg Time (ms)
18	14	4	77.8%	592

[Breakdown by Language](#) [Breakdown by User](#)

Figures: Code Execution History Page

### Detailed View

Select an execution to view details:

12 [20:29:32] Abs - python

Code Output Input Info

#### Code Executed:

```
# Python code
# Start writing your code here...
print("ABs")
```

### Statistics

Total Executions	Successful	Failed	Success Rate	Avg Time (ms)
19	15	4	78.9%	593

[Breakdown by Language](#)

Language	Executions
cpp	4
java	3
c	2
python	10

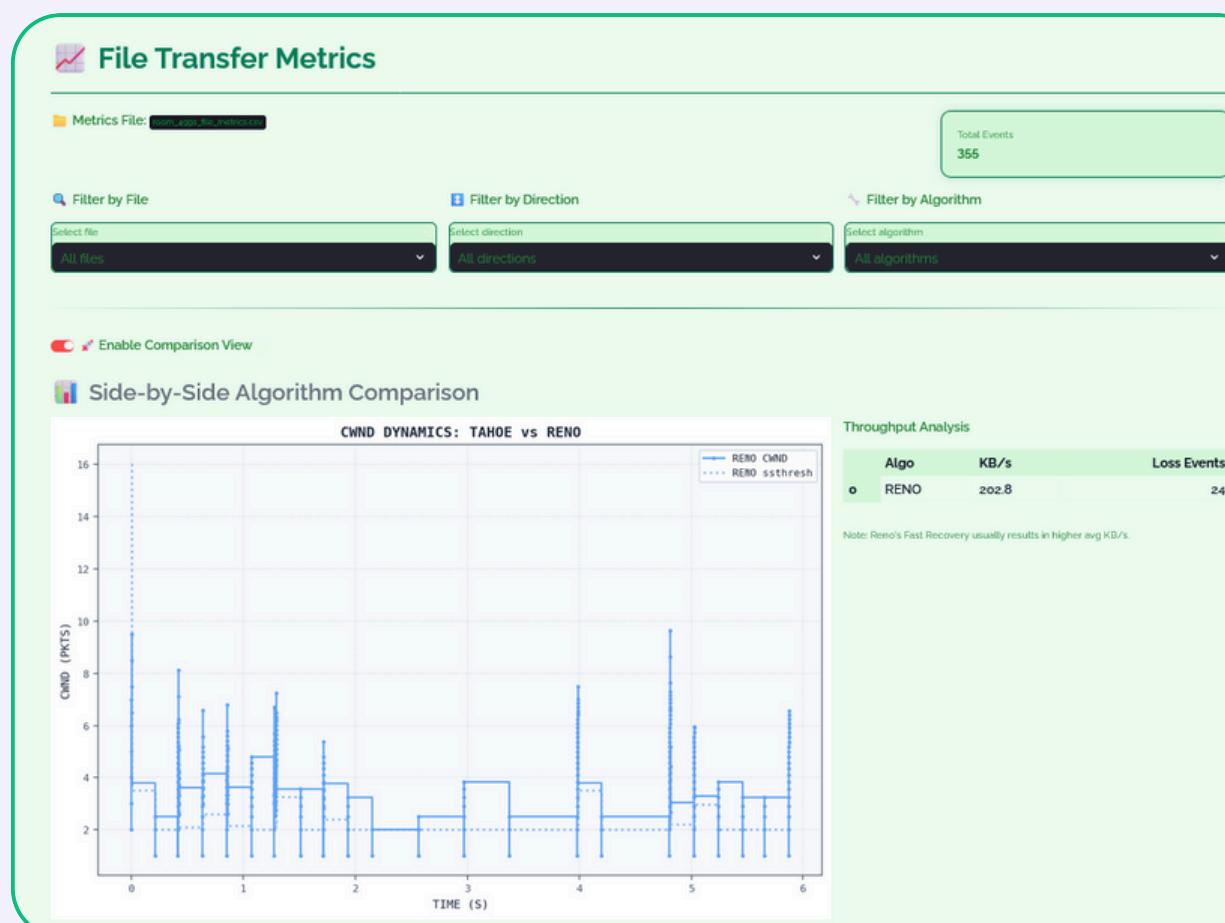
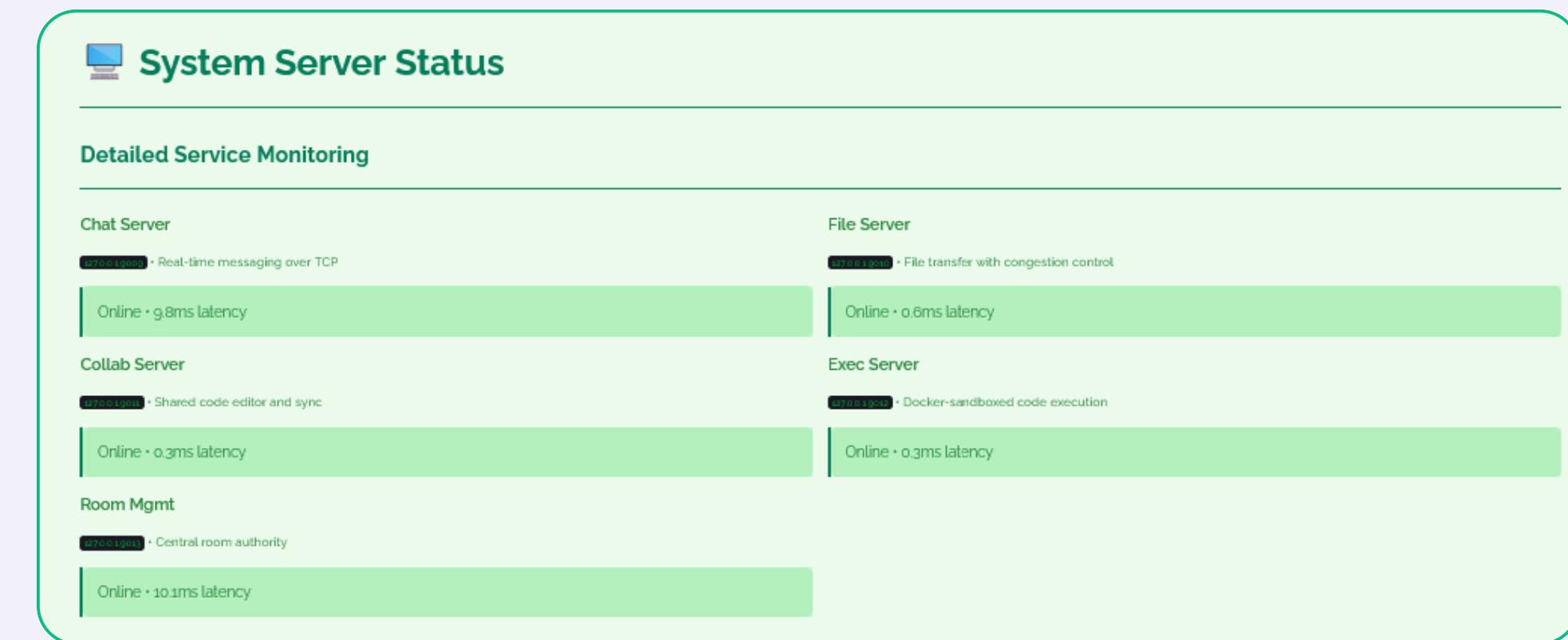
[Breakdown by User](#)

User	Executions
Unknown	12
Mehedi	5
Abs	2

### Execution History

Time	User	Room	Language	Status	Return Code	Time (ms)
2026-01-07 14:33:59	Abs	4991	cpp	Success	0	785
2026-01-07 14:33:59	Unknown	4991	cpp	Success	0	785
2026-01-07 14:33:54	Unknown	4991	cpp	Success	0	545
2026-01-07 14:33:23	Mehedi	4991	c	Success	0	530
2026-01-07 14:33:23	Unknown	4991	c	Success	0	530
2026-01-07 14:30:03	Unknown	4991	python	Success	0	765
2026-01-07 14:29:40	Mehedi	4991	python	Success	0	780
2026-01-07 14:29:40	Unknown	4991	python	Success	0	780
2026-01-07 14:29:32	Abs	4991	python	Success	0	788
2026-01-07 14:29:32	Unknown	4991	python	Success	0	788

# User Interface



Figures: Analytical Dashboard Page

# Limitations

- **Scalability:**  
Single instance, ~50 concurrent users max
- **Persistence:**  
No database-data lost on restart
- **Authentication:**  
Basic usernames, no password protection
- **Collaboration:**  
Last-write-wins only, no cursor tracking

# Future Plans

- **PostgreSQL** database integration
- **JWT authentication** with password-protected rooms
- Cursor position sharing
- **Screen sharing & collaborative whiteboard**

---

Thank you<sup>•</sup>