University of Dhaka
Department of Computer Science and Engineering
CSE 3111

SyncroX

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

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)**

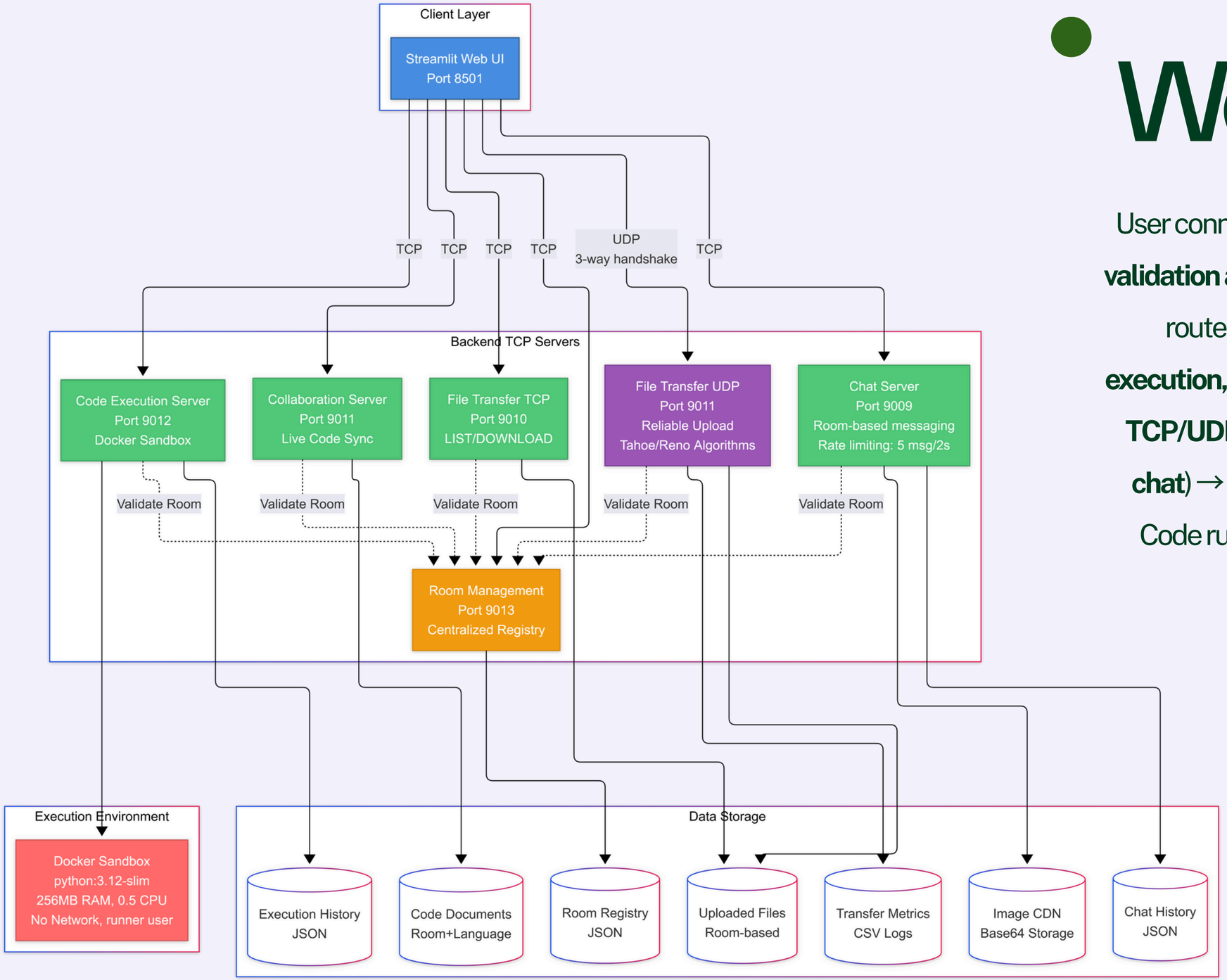
File Transfer

- **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(CWND, RWND)
- Buffered I/O for efficient data handling

Applied Networking Concepts

RTT Estimation

- **RTT = T_ACK - T_SEND** (per-chunk measurement)
- **SRTT = (1-0.125) x SRTT + 0.125 x RTT**
- **RTTVAR = (1-0.25) x RTTVAR + 0.25 x |SRTT-RTT|**
- **RTO = SRTT + 4 x RTTVAR** (retransmission timeout)

Congestion Control (Tahoe & Reno)

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

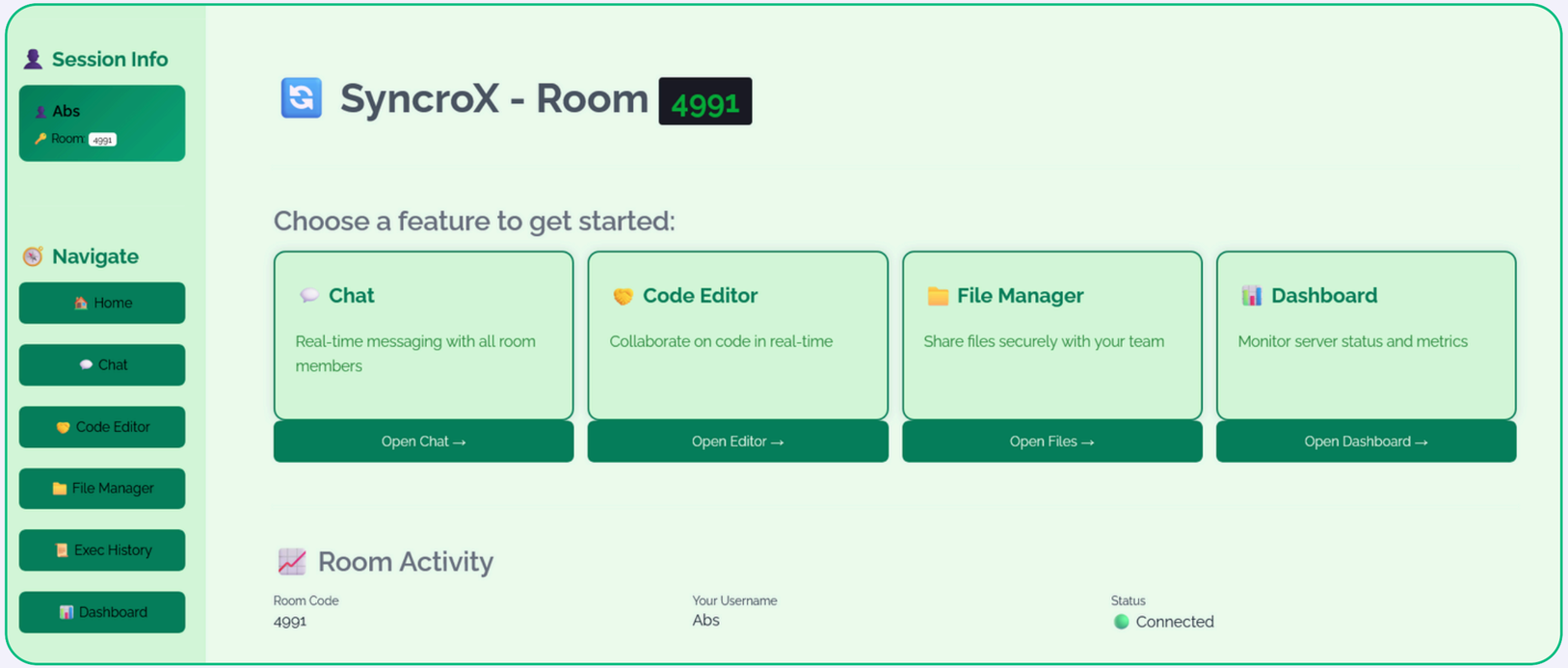


Figure: User Main Page

User Interface:

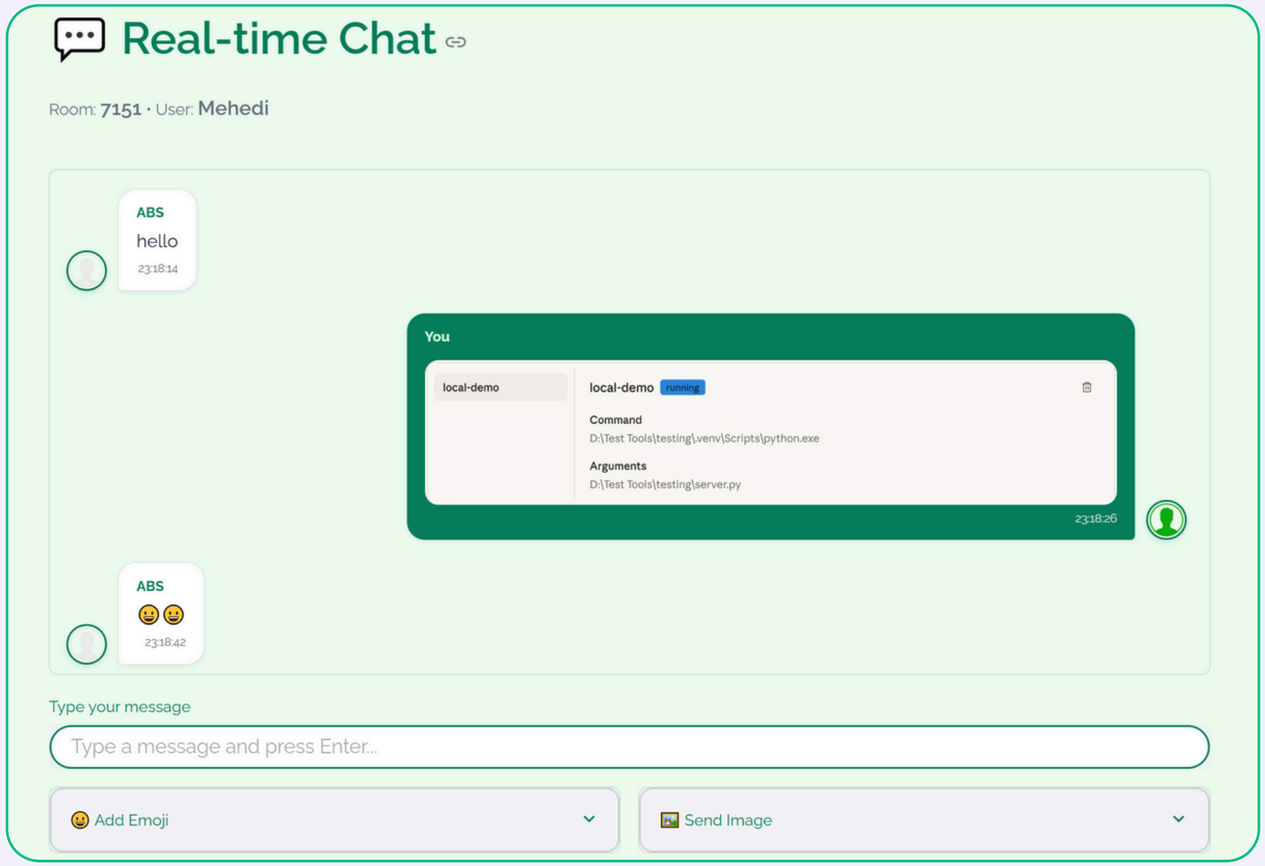


Figure: Chat Page

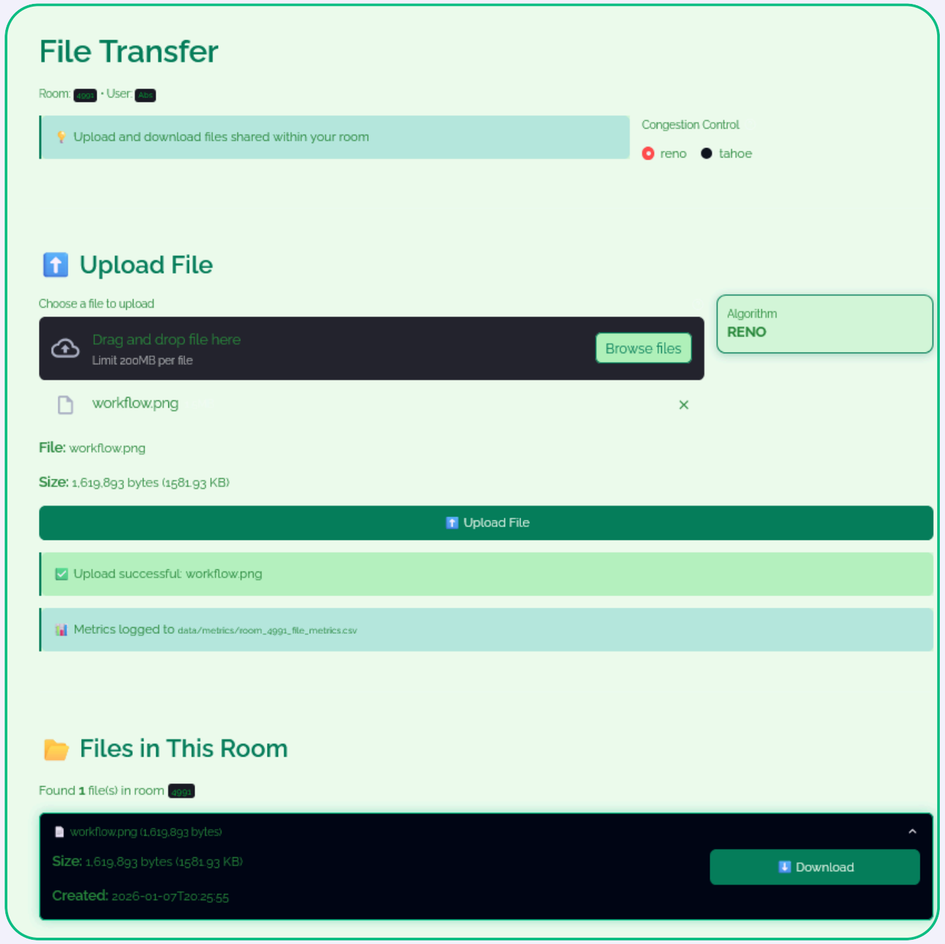


Figure: File Transfer Page

User Interface:

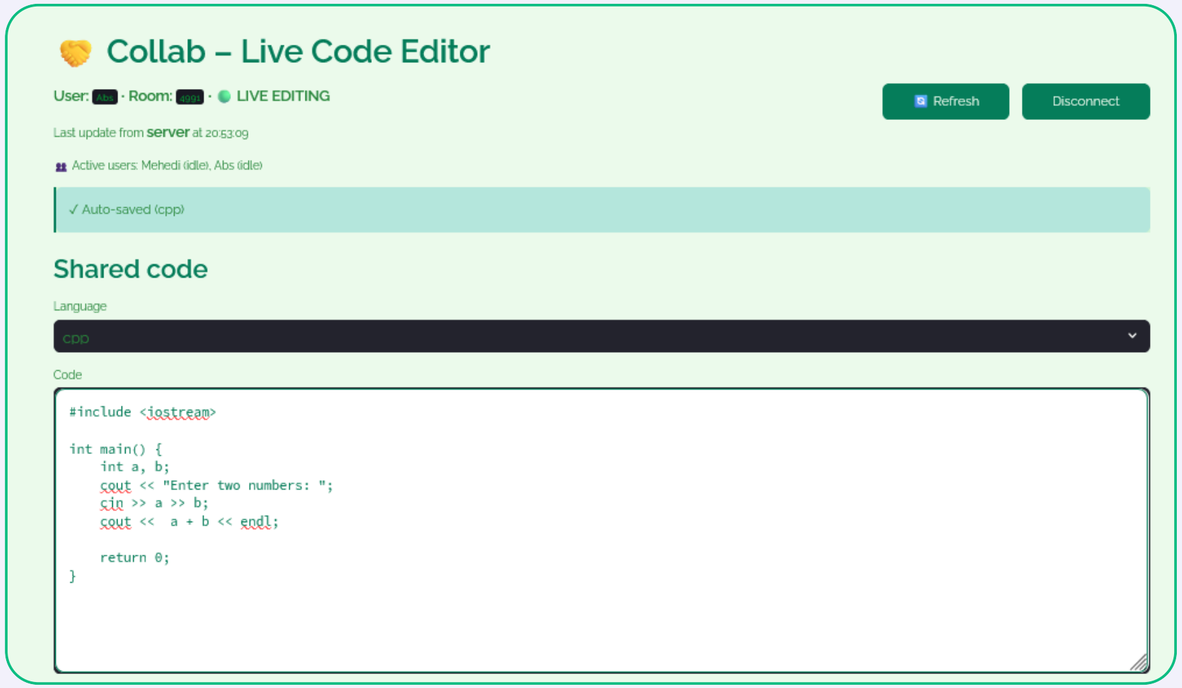


Figure: Live Code Editor Page

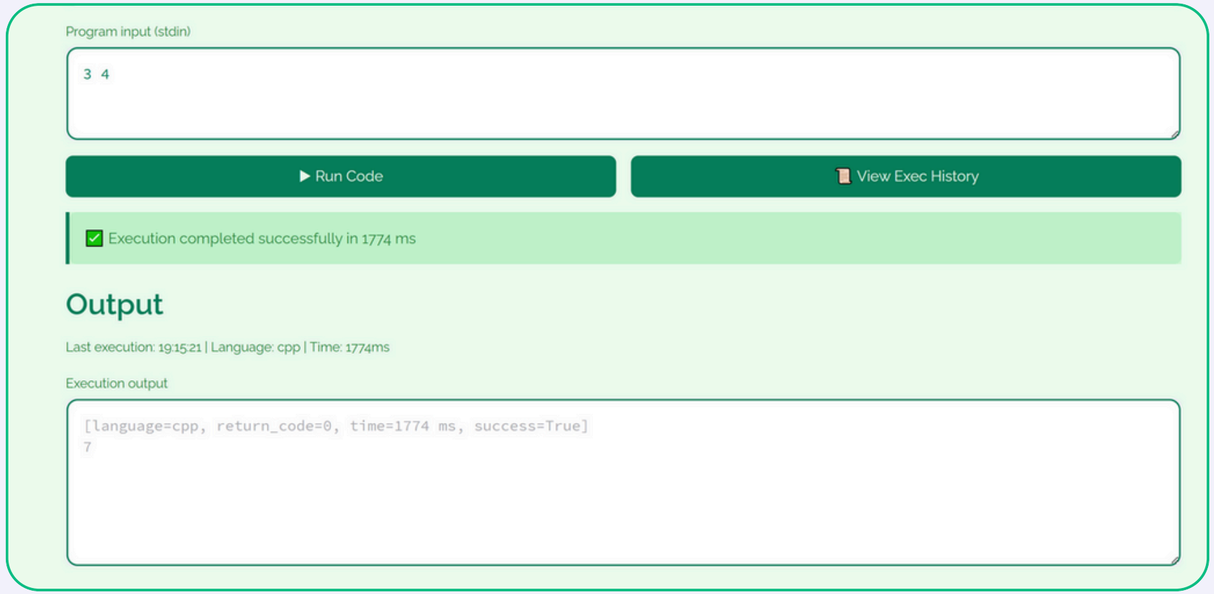
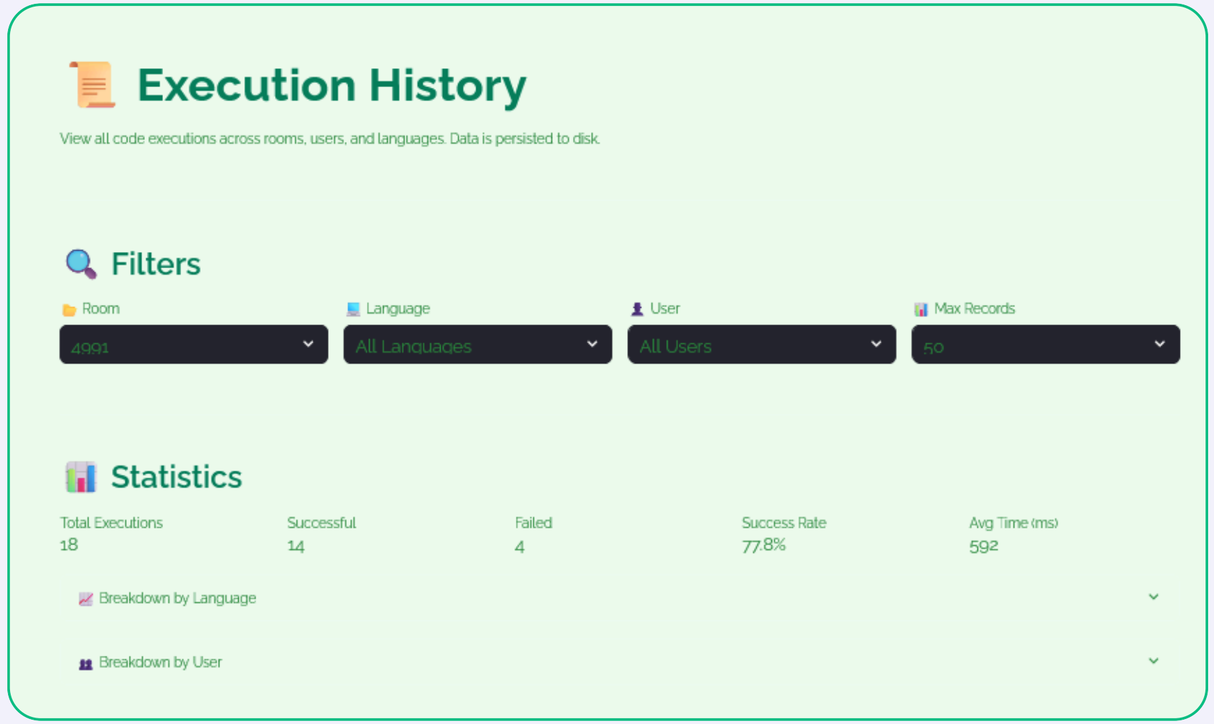
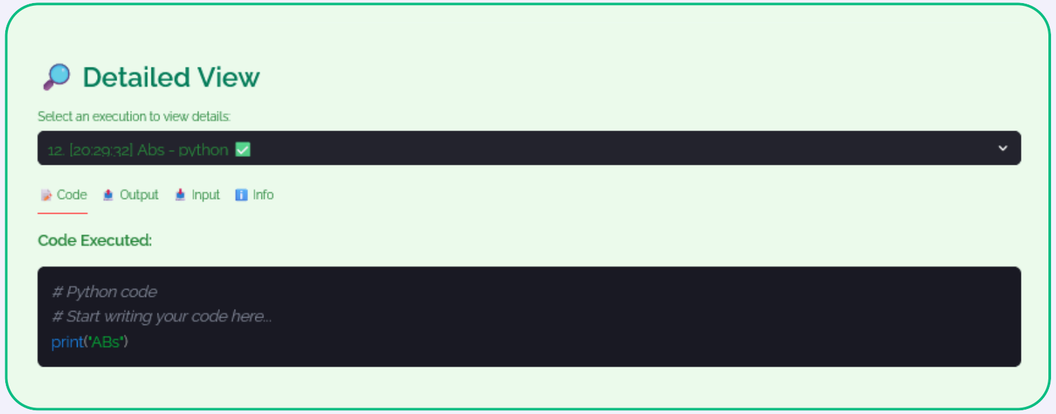
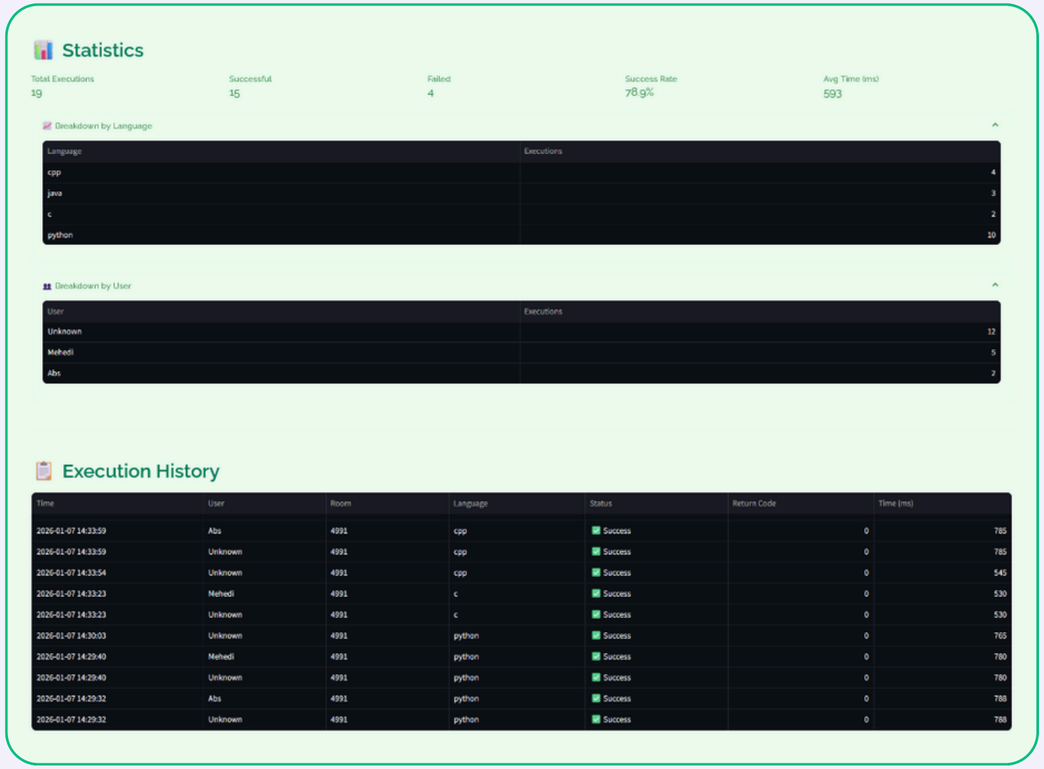


Figure: Live Code Editor Output

User Interface:

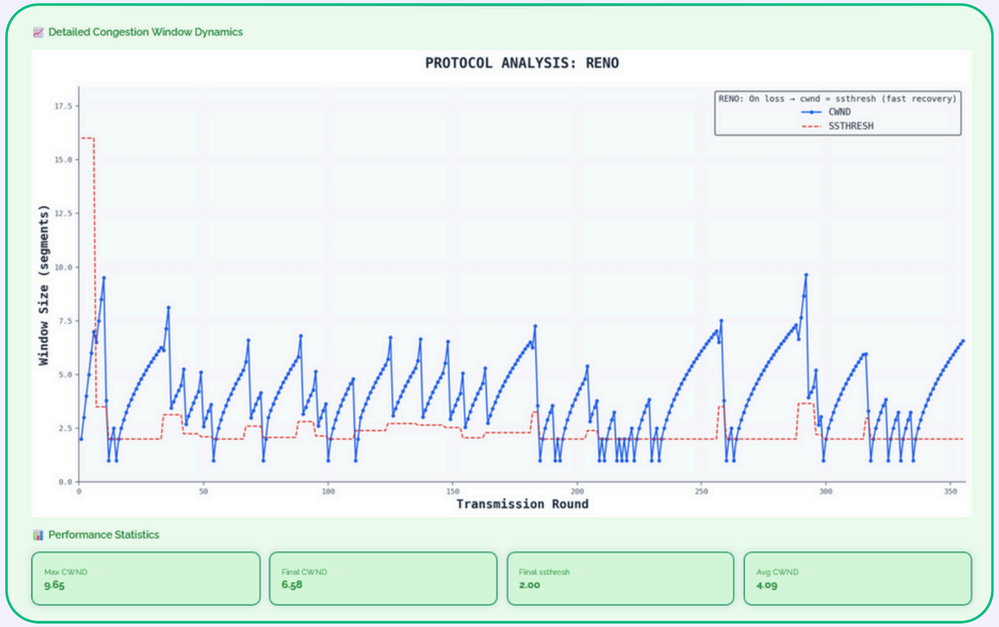


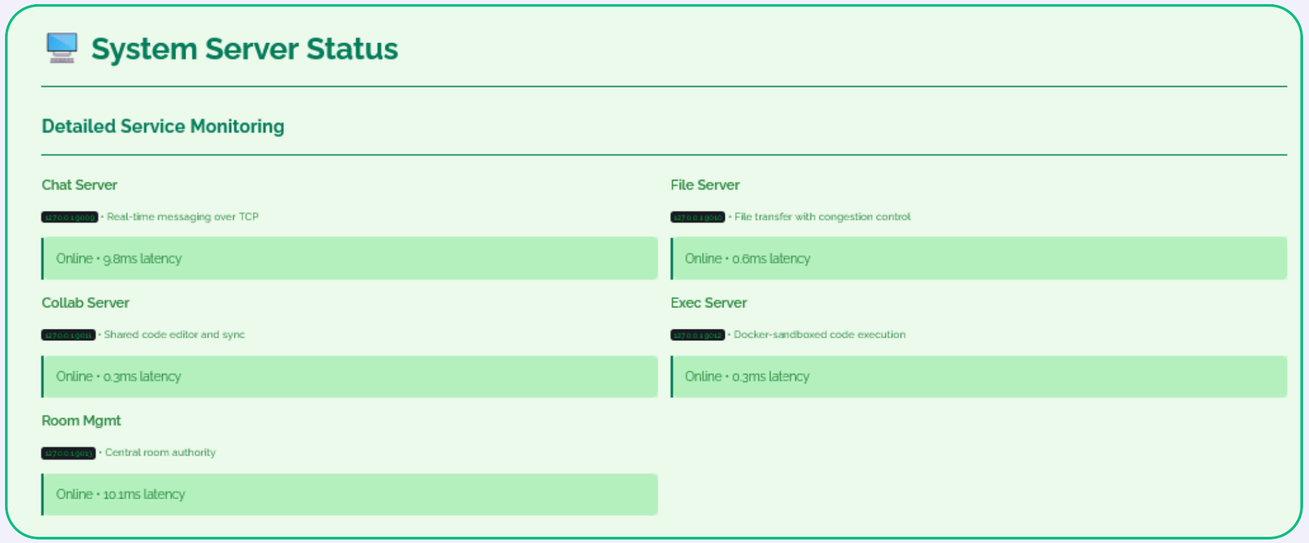


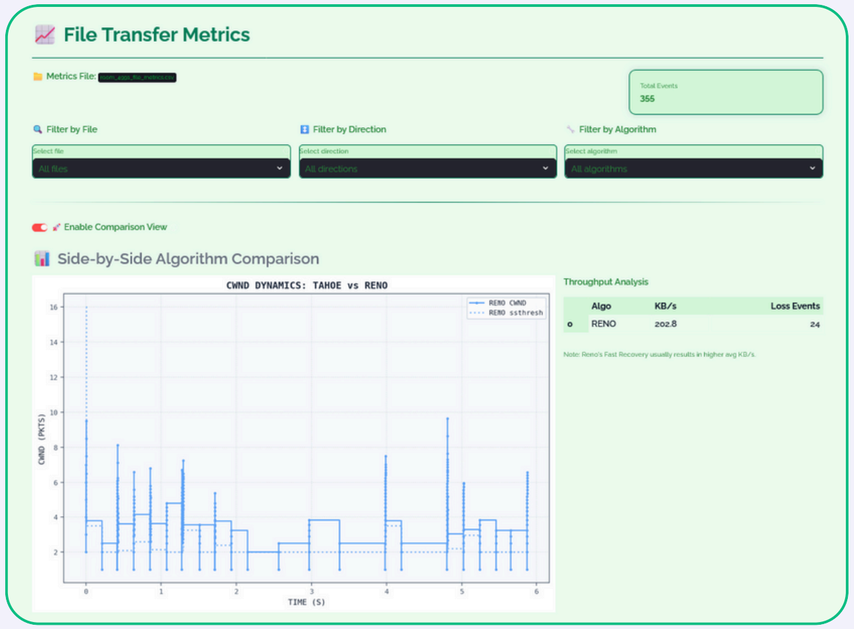


Figures: Code Execution History Page

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