TEAM - A

MultiThread-Based Chat System with Socket Communication

2022200056 AN, CHAE WON (TEAM LEADER)
2020320126 LEE, WON JUN
2021270131 LEE, SEO JUN
2022320119 AISYAH HUMAIRA BINTI ANUAR

Problem Statement

Inefficient Handling of Multiple Connections

Existing CLI-based chat servers often struggle to manage numerous simultaneous client connections efficiently.

Lack of Real-Time Reliability

Many current solutions face issues with message delays and occasional message loss.

Scalability Constraints

As the number of users increases, existing chat systems fail to scale effectively.

Limited Concurrency Management

Inadequate concurrency control mechanisms in existing systems lead to resource contention.

Limitations of Existing Solutions

Concurrency Handling	Many chat servers use single-threaded architectures, Causing slow performance when many users are active.
Real-Time Communication	Inefficient messaging systems lead to slow message delivery.
Error Management	Poor handling of network errors and disconnections causes unstable chat environments.
Inter-Process Communication	Existing solutions often neglect robust IPC mechanisms, limiting the system's reliability and performance.

Our Approach and Methodology

Socket Programming

- TCP connections
- Real-time communication

Multithreaded Architecture

- Thread-based server design
- Dedicated thread for each client connection

Robust Error Handling

- Comprehensive exception handling
- Network error management
- Graceful disconnection handling

Modular Design

- Well-structured codebase for maintainability
- Scalability

Results

Efficient Concurrency Management

The multithreaded design allows the server to manage many users at once without slowing down.

Improved Resource Utilization

Threads share memory within a process, reducing the overhead compared to creating separate processes for each user.

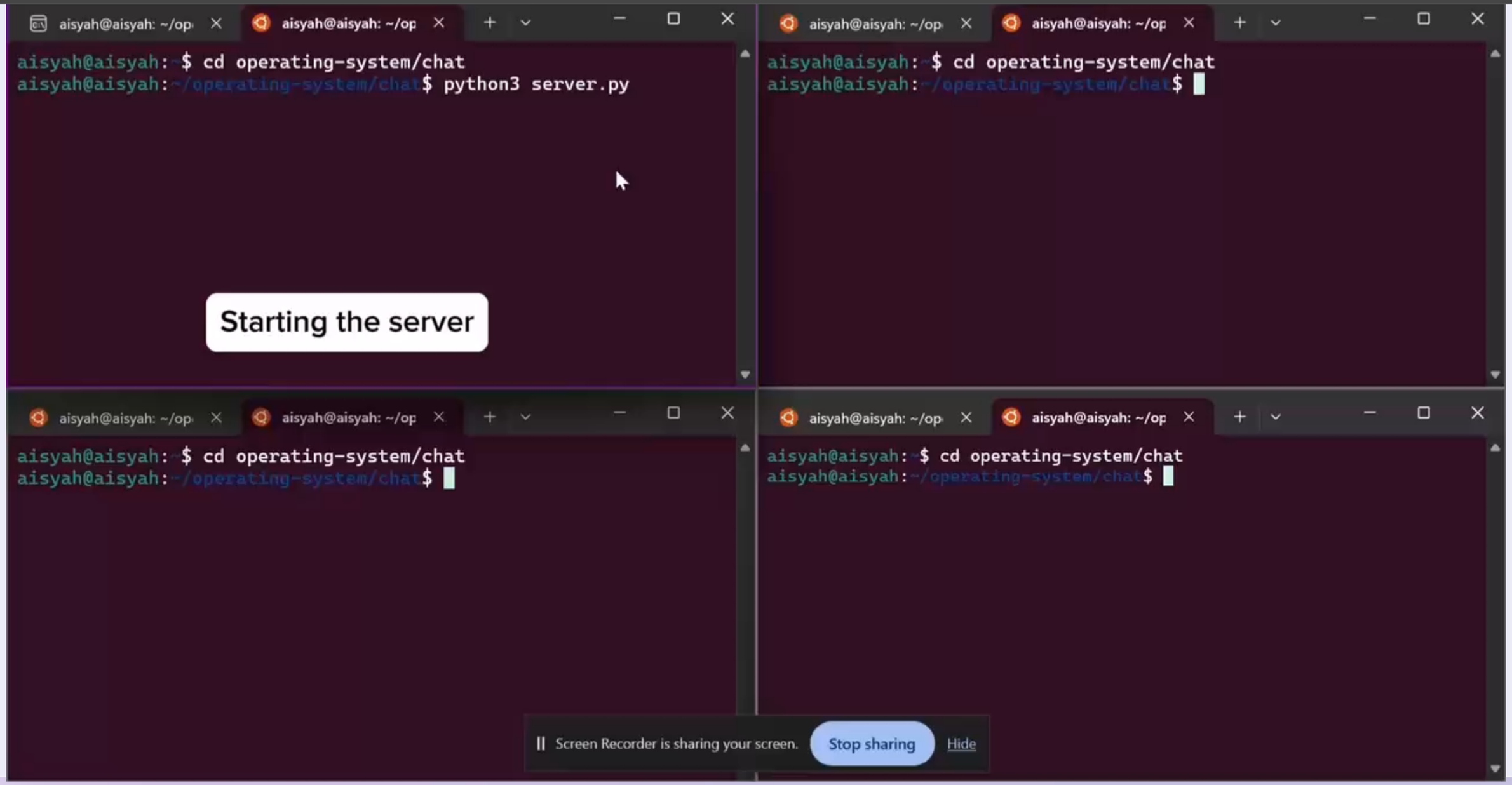
Scalability

The system efficiently supports an increasing number of users by creating and managing threads dynamically, demonstrating OS-level thread scheduling.

Enhanced Stability

Robust error handling in the multithreaded design ensures the server remains stable, even when individual threads encounter issues like disconnections.

Demo Video


aisyah@aisyah: ~/op
aisyah@aisyah: ~/op
aisyah@aisyah: ~$ cd operating-system/chat
aisyah@aisyah:~/operating-system/chat$ python3 server.py
Starting the server
aisyah@aisyah: ~/op
aisyah@aisyah: ~/op
aisyah@aisyah: ~$ cd operating-system/chat
aisyah@aisyah:~/operating-system/chat$
aisyah@aisyah: ~/op
aisyah@aisyah: ~/op
aisyah@aisyah: ~$ cd operating-system/chat
aisyah@aisyah:~/operating-system/chat$
aisyah@aisyah: ~/op
aisyah@aisyah: ~/op
aisyah@aisyah: ~$ cd operating-system/chat
aisyah@aisyah:~/operating-system/chat$
Screen Recorder is sharing your screen.
Stop sharing
Hide