Project Report: Real-Time Web Chat Server Using Flask and Socket.IO

1. Project Title

Real-Time Web Chat Server Using Flask and Socket.IO

2. Team Members

This project was developed and implemented by a two-member group as part of our Operating Systems Lab coursework:

Name: Jahidul Islam
ID: 0242220005101166

Name: Rabiya Bushra Roja
ID: 0242220005101370

3. Course Details

- Subject: Operating Systems Lab
- Instructor: Israt Jahan
- Department: CSE
- Institution: Daffodil International University
- Semester: Spring 2025

4. Introduction

This project presents the development of a real-time chat server using Flask (a Python web framework) and Socket.IO (a real-time communication protocol based on WebSockets). It enables multiple users to send and receive messages instantly via a browser-based interface.

The primary goal is to demonstrate how a concurrent server handles multiple client connections and implements real-time data transfer, which is a core topic in modern operating systems and distributed systems.

5. Motivation & Background

Real-time applications such as WhatsApp, Slack, Discord, and Google Chat use event-driven servers and socket-based communication to facilitate live interaction between users. These systems require concurrency, efficient resource management, and quick context switching — topics directly related to Operating Systems.

By implementing our own web-based chat server, we aimed to understand:
- How socket communication works
- How concurrent clients are handled on a server
- How real-time systems function at the OS and application level
- How the WebSocket protocol differs from HTTP

6. Objectives

- Build a basic concurrent server for communication
- Enable real-time message exchange between users
- Design a responsive web UI for chat
- Explore Flask-SocketIO as a WebSocket implementation
- Understand OS concepts such as event loops and asynchronous I/O

7. Technologies Used

Frontend:
- HTML5
- CSS3
- JavaScript
- Socket.IO JavaScript Client

Backend:
- Python 3.x
- Flask
- Flask-SocketIO

Tools:
- Git & GitHub
- Visual Studio Code
- firefox

9. Implementation Details

The backend was built using Flask, a micro-framework for Python. To handle real-time communication, we used Flask-SocketIO, which provides integration between Flask and the Socket.IO protocol.

Each time a user sends a message, the message is emitted to the server, which then broadcasts it to all connected sockets. All messages are displayed dynamically on the chat interface.

Key Functions:
- socketio.on('message') — Listens for incoming messages
- socketio.send(message, broadcast=True) — Sends messages to all connected users

10. How to Run the Application

Step 1: Install dependencies
pip install flask flask-socketio eventlet

Step 2: Run the server
python server.py

Step 3: Open browser and visit:
http://localhost:

Step 4: Open multiple tabs or devices to simulate multiple users chatting in real-time.

12. Testing Strategy

- Functional Testing: Verified message delivery and broadcasting
- Multi-client Testing: Opened multiple browser tabs for concurrency
- Network Testing: Verified messages over LAN
- Browser Compatibility: Tested in Chrome, Firefox

13. Key OS Concepts Learned

- Network sockets & server-client architecture
- Event-driven programming (similar to epoll/select)
- Handling concurrency and multiple user connections
- Inter-process and inter-thread communication via sockets

14. Future Enhancements

- Display timestamps next to messages
- Private messaging or chat rooms
- Upload and share files/images
- Save message history to a database
- Deploy on a cloud platform
- Add security measures

15. Challenges Faced

- Handling multiple clients initially with Tkinter
- Switching from raw sockets to a stable WebSocket-based architecture
- Ensuring cross-browser compatibility with Socket.IO
- Configuring eventlet for non-blocking performance

16. Conclusion

This project gave us practical experience with network programming, asynchronous server handling, and browser-based interfaces. It helped us understand the inner workings of WebSockets and their relation to OS-level socket management. Overall, it was an excellent learning experience that bridged the gap between theoretical concepts in Operating Systems and real-world applications.

17. References

- Flask documentation: https://flask.palletsprojects.com/
- Flask-SocketIO: https://flask-socketio.readthedocs.io/
- Socket.IO: https://socket.io/docs/
- Python official docs: https://docs.python.org/3/
- Eventlet: https://eventlet.net/