Assignment: Real-Time WebSocket Chat Server with Topic Rooms

Language: Python 3.9+
Framework: FastAPI (WebSocket)
Deadline: 1 working day (8 hours maximum)

Objective

Develop a lightweight real-time chat server using FastAPI WebSockets that allows users to join topic-based chat rooms, send messages to users within the same topic, and automatically remove messages after a set time period.

Requirements

1. Basic Connection

- Clients connect to the server using WebSocket and provide:
- `{"username": "alice", "topic": "sports"}`
- If a username already exists in the same topic, append a numeric suffix to make it unique (e.g., `alice#2`).

2. Topic Rooms

- Each topic acts as a separate chat room.
- Users in the same topic receive each other's messages.
- If a topic becomes empty, it should be removed from the active topic list.

3. Messaging

- Messages are broadcast to all users in the same topic except the sender.
- The sender receives a delivery acknowledgment in JSON.
- Each message includes:
- `{`
- `  "username": "alice",`
- `  "message": "Hello World",`
- `  "timestamp": 1690000000`
- `}`
- Messages automatically expire and are deleted after 30 seconds (use `asyncio.create_task`).

4. Topic Listing

- When a user sends the command `/list`, the server responds only to that user with:
- `Active Topics:`
- `sports (2 users)`
- `movies (1 user)`

5. Session Cleanup

- When a user disconnects:
    - Remove them from the topic.
    - If the topic becomes empty, remove the topic entry.
    - Ensure other users in the topic are not affected.

6. Error Handling

- Handle invalid JSON payloads without crashing the server.
- Log disconnects and errors cleanly.

Deliverables

- `main.py` — WebSocket server using FastAPI.
- `client_example.py` — Simple Python client demonstrating:
    - Joining a topic
    - Sending and receiving messages
    - Using the `/list` command
- `README.md` — Brief instructions on how to run the server and client.

Testing Scenarios

- Two users join the same topic and exchange messages in real time.
- `/list` shows all active topics and user counts accurately.
- Messages expire and are not stored beyond 30 seconds.
- Topics disappear when all users leave.
- Invalid payloads are handled gracefully.

Constraints

- Python 3.9 or higher.
- FastAPI and Uvicorn must be used.
- No external databases or message brokers.
- No use of Socket.IO or other high-level WebSocket libraries.
- In-memory state only.

Deadline

This assignment is designed to be completed within **one working day** (approximately 6–8 hours).
Submit a working solution with clear instructions to run and test