Group A

Chat application

1. Environment & Dependencies (Mandatory task)
    - **Virtual Environment**: Set up using `python3 -m venv venv` to ensure isolated project dependencies.
    - **Essential Packages**: Install FastAPI, Uvicorn, SQLAlchemy (or SQLModel), psycopg2 (for PostgreSQL), passlib (for password hashing), and python-jose (for JWT handling).
2. JWT Authentication & Role-Based Access Control (RBAC) (Mandatory Task)
    - **User Model**: Design a SQLAlchemy model with a `role` field, categorizing users as `admin` or `user`.
    - **Authentication Endpoints**:
        - **Signup**: Accepts user credentials, hashes the password, and stores the user with an assigned role.
        - **Login**: Verifies credentials, generates a JWT token signed with HS256, embedding the user's role within the token.
    - **RBAC Dependency**: Implement a reusable dependency to restrict access to routes based on the user's role.
3. Protected WebSocket Chat (Mandatory task)
    - **WebSocket Endpoint**: Establish a `/ws/{room_id}` route in FastAPI to handle WebSocket connections.
    - **JWT Authentication**: Secure the WebSocket connection by verifying the JWT token, typically passed as a query parameter or header.
    - **Chat Functionality**:
        - On connection, fetch and send recent messages from PostgreSQL using cursor-based pagination.
        - Broadcast incoming messages to all connected clients in the same room and store them in the `messages` table.
        -

Group B

(Choose any 1 task from Task 1 and Task 2)

1. PostgreSQL Persistence
    - **Database Integration**: Utilize SQLAlchemy (or SQLModel) for ORM-based interaction with PostgreSQL.
    - **Models**:
        - **User**: Stores user information, including credentials and role.
        - **Room**: Represents chat rooms, potentially with metadata like room name or description.
        - **Message**: Captures individual messages, linking them to users and rooms, and storing timestamps.
    - **Relationships**: Define appropriate relationships between models (e.g., one-to-many between Room and Message).
2. Admin Dashboard & Analytics(Advance)
    - **Admin Interface**: Integrate **SQLAdmin** with FastAPI to provide a web-based interface for managing models.

- **Analytics Endpoints**:
    - **Messages per Room**: Endpoint to retrieve the count of messages sent in each room.
    - **User Activity**: Endpoint to track user participation, such as messages sent or rooms joined.
    - **Date Filters**: Implement optional date filters to analyze activity over specific periods.
- **Admin Security**: Protect dashboard routes using the RBAC dependency to ensure only authorized users can access them.
- **Optional Feature**: Implement CSV export functionality for analytics data.
-
- **Group C**
-

1. Sentiment Analysis (Only if you are interested)
    - Dataset: IMDb Reviews, Twitter Sentiment dataset.
    - Task:
        - Clean and preprocess text.
        - Vectorize using TF-IDF or Word Embeddings.
        - Train a classifier to predict positive/negative sentiment.
        - Test the model on unseen reviews.

Deliverables:

- A Jupyter notebook with:
    - Clean, commented code.
    - Markdown explanations of each step.
    - Visualizations of data & results.
- Saved model file (`.pkl` or `.h5`).
- Optional: short README.md explaining the project.