Technical Assignment: Secure Group Messaging (Backend only)

Objective

Develop the backend portion of a secure group messaging system that supports both private (closed) and open groups along with user authentication, messaging, and basic security (message encryption).

The solution should be built using Node.js.

You are strongly encouraged to use AI-based tools (e.g., ChatGPT, Claude, Copilot, Cursor, Windsurf, etc.) and other productivity plugins to accelerate development.

Core Requirements

1. User Authentication

- **Endpoints:**
  - **Registration:** Accept email and password.
  - **Login:** Issue a token or session.
- **Data Storage:**
  - Use a lightweight data store (e.g., MongoDB Atlas free tier or Azure SQL free tier) to store user credentials.
- **Validation & Security:**
  - Ensure proper email validation and secure password handling (e.g., hashing).

2. Group Management

Group Types & Join/Leave Rules:

- **Group Creation:**
  - **Parameters:** When creating a group, the owner must specify:
    - Group name
    - Group type: **Private (closed)** or **Open (public)**
    - Maximum number of members (minimum 2 to unlimited)
  - **Business Rules:**
    - A **private group** is restricted for joining without owner approval, while any member may leave at any time.
    - An **open group** allows users to freely join and leave at any time.
- **Join Group Behavior:**
  - **Open Group:**

        - Users can join immediately without any approval.

    - **Private Group:**
        - Users must submit a join request.
        - The join request appears in a dedicated section (to be handled by the frontend) accessible by the group owner.
        - The owner may then approve or decline the request.

    - **Banishment Rule:**
        - If a user is banished from any group (open or private), they must submit a new join request and receive explicit approval from the owner to rejoin.

    - **Leave & Cooldown Policy (Private Groups):**
        - If a user leaves a private group, they must wait **48 hours** before they can request to join that group again.

- **Leave Group Behavior:**
    - **Open Group:**
        - Users can join or leave without restrictions.
    - **Private Group:**
        - Users can leave at any time, but rejoining must follow the cooldown rule as mentioned above, unless they are rejoining after being banished (which always requires a join request and approval).
    - **Group Owner Constraints:**
        - The group owner must transfer the owner role to another member before leaving the group.
        - A group can only be **deleted** by the owner if they are the sole member (i.e., no other member exists).
- **Additional Administrative Features:**
    - **Banishment:**
        - The owner can banish a member from the group.
        - Once banished, the affected user must submit a new join request (regardless of the group type) to be allowed back, and the owner must approve this request.

3. Messaging

- **Functionality:**
    - Enable users to send and retrieve messages within a group.
- **Message Encryption:**
    - Use AES-128 to encrypt messages before storage, and decrypt them when retrieved.
- **Data Storage:**
    - Save messages along with metadata (sender ID, timestamp, group ID).

4. Real-Time Messaging (Simulated)

- **Implementation:**
    - Simulate real-time behavior (e.g., by returning timestamped acknowledgments or using a basic queuing mechanism).
    - Actual WebSocket integration is optional. If not implemented, provide documentation on how such functionality could be extended.

5. API Documentation & Error Handling

- **Documentation:**
    - Provide a comprehensive README with:
        - Setup and run instructions.
        - Detailed API documentation (endpoints, methods, request and response formats).
        - Explanation of any known issues or incomplete features.
- **Error Handling:**
    - Ensure robust error handling and logging throughout the backend application.

6. Security Considerations

- **Encryption:**
    - Use AES-128 for encrypting/decrypting message payloads.
- **Sensitive Data Handling:**
    - Secure handling of passwords, tokens, and other sensitive data is required.

7. Use of AI-Based Tools

- **Productivity Enhancement:**
    - You have full permission to use AI-based coding tools, editors, and plugins.

Deliverables

1. **Source Code Repository:**
    - Develop your solution within a shared Git repository.
    - Use clear commit messages that document your progress and thought process.

2. **README Documentation:**
    - Include detailed setup instructions.
    - Document all API endpoints (URLs, methods, request/response formats).
    - Explain any known issues or limitations.
    - Describe how AI-based tools were used to enhance productivity.

3. **Code Comments & Logging:**

- Ensure the code is well-commented to explain major logic blocks and any non-obvious decisions.
- Implement logging (including debug logs where applicable) to aid in troubleshooting and evaluation.

4. **Additional Notes:**
   - If you implement extra features (e.g., real-time WebSockets), document the enhancements and explain additional challenges addressed.
