Module 2 — Real-time 1:1 messaging (WebSocket + persistence) (MVP)

Goal: working private chats with messages delivered in real-time and persisted.

What to do

1. **Backend** - messaging core
    - Models:
        - Chat: id, type(PRIVATE/GROUP), participants[]
        - Message: id, chatId, senderId, text, attachments[], timestamp, readBy[]
    - Add MongoDB collections for Chats and Messages.
    - Add REST endpoints:
        - POST /api/chats (create/get private chat)
        - GET /api/chats?userId=... (list chats)
        - GET /api/chats/{id}/messages?limit=50
    - WebSocket configuration: STOMP endpoint /ws and broker /topic.
    - Controllers for STOMP messages: on send -> save message to DB, publish to /topic/chats/{chatId}.
2. **Frontend - chat UI + WS**
    - Create ChatList and ChatWindow components.
    - Connect to WebSocket (stompjs/sockjs-client). Subscribe to /topic/chats/{chatId}.
    - Send messages via STOMP /app/chat.sendMessage.
    - Render message list, show sender, timestamp; scroll to bottom.

Deliverables

- Real-time chat between two browser windows (or two devices) showing messages instantly.
- Messages persisted and loadable on page refresh.

Acceptance criteria

- Message sent from User A appears instant for User B.
- Message saved in MongoDB and retrievable via REST.
- WebSocket reconnection basic handling implemented.

Minimal endpoint names (use these)

- POST /api/auth/register, POST /api/auth/login
- GET /api/chats — list
- POST /api/chats — create
- GET /api/chats/{chatId}/messages
- WS endpoint /ws, STOMP dest /app/chat.sendMessage, broadcast /topic/chats/{chatId}