Real-Time Chatting App (with Multimedia Support)

User Experience

You are a user who:

- Signs up and logs in using email and password
- Sees all other users (except yourself) in a sidebar after login
- Can see real-time online/offline status of others
- Can click on any user to open a private 1-on-1 chat
- Can send/receive **text, images, videos, audio, and files** in real time
- Can view past chat history and uploaded media

User Flow

1. **Registration / Login**
    - User enters name, email, password
    - Credentials are validated and stored securely
    - Redirected to main chat interface after login
2. **Main Chat Interface**
    - Left sidebar lists all users (excluding self) with online/offline indicator (dot/badge)
    - Right pane shows selected chat conversation
3. **Initiate Chat**
    - Clicking a user opens the chat window
    - Displays real-time messages + past history (scrollable)
    - Supports text input, emoji picker, and file upload options (image/video/audio/doc)
4. **Real-Time Interaction**
    - Messages and media appear instantly using Socket.IO
    - Online status updates in real time
    - Media uploads show preview or icon with download option

---

Core Features

Authentication

- User registration and login via email/password
- JWT-based authentication
- Password hashing with bcrypt
- Auth middleware to protect routes

User System

- MongoDB user schema: name, email, passwordHash, avatar (optional), isOnline (via socket)
- Online/offline tracking via Socket.IO presence
- Sidebar lists all users except self

Chat System

- Real-time messaging via **Socket.IO**
- **1-on-1 private chats** only (no groups)
- Store messages in MongoDB: senderId, receiverId, messageType, content (text/media), timestamp
- Chat messages include:
    - text
    - image (JPG, PNG, WebP)
- Socket connection is authorized via JWT token

File Upload System

- Users can upload files using drag-and-drop or file input
- Uploaded media/files stored in backend file system or cloud (e.g., Cloudinary / AWS S3)
- Media message type includes a file URL and metadata (name, size, type)

Evaluation Criteria

- Clean, scalable RESTful API design

- Proper data modeling and relationships (users ↔ messages)
- Secure token-based authentication & authorization
- Robust real-time messaging with Socket.IO
- File/media upload with validation and preview
- Minimal but clean UI using Tailwind CSS
- Frontend/backend separation and code organization
- Dockerization of both frontend and backend (optional bonus)

Tech Stack

- **Frontend**: React.js + Tailwind CSS
- **Backend**: Express.js + Node.js
- **Database**: MongoDB + Mongoose
- **Real-Time**: Socket.IO
- **Auth**: JWT + bcrypt
- **File Uploads**: multer (or cloud storage like Cloudinary)
- **Optional**: Docker (for containerization)