Perfect! Now let's create the **Architecture Diagram with all API endpoints**. While I can’t draw the image directly in text, I’ll describe the full structure and endpoints clearly. You can use tools like **draw.io**, **Lucidchart**, or **Whimsical** to convert this into a diagram.

**📄 5. Architecture Diagram with Endpoints**

**📘 1. System Overview (Modular Architecture)**

[Client Browser]

↓

[Frontend - React.js]

↓ ⇄ REST APIs

[Backend - Spring Boot]

↓

[Database - MySQL/PostgreSQL]

↓

[External Services - Payment (Razorpay), Email (SendGrid), Chat (Firebase)]

**🧩 2. Modules & Internal Communication**

**🔹 Module 1: Authentication & User Management**

* **POST** /api/auth/register – Register with role (freelancer/client)
* **POST** /api/auth/login – Login and get JWT
* **GET** /api/user/profile – Get own profile (based on role)
* **PUT** /api/user/profile – Update profile

**🔹 Module 2: Project Management (Client)**

* **POST** /api/projects – Post a new project
* **GET** /api/projects – View all projects
* **GET** /api/projects/{id} – Get project by ID
* **PUT** /api/projects/{id} – Edit a project
* **DELETE** /api/projects/{id} – Delete a project
* **GET** /api/projects/my – View own posted projects

**🔹 Module 3: Proposal Management (Freelancer)**

* **POST** /api/proposals/{projectId} – Submit a proposal
* **GET** /api/proposals/my – View proposals submitted
* **GET** /api/proposals/project/{projectId} – Client views all proposals for their project
* **PUT** /api/proposals/{proposalId}/accept – Client hires freelancer
* **PUT** /api/proposals/{proposalId}/reject – Client rejects proposal

**🔹 Module 4: Messaging (Chat)**

* WebSocket: /ws/chat or use Firebase Realtime DB
* **GET** /api/messages/{userId} – Get chat history with a user
* **POST** /api/messages/send – Save chat message if not using Firebase

**🔹 Module 5: Payment & Wallet**

* **POST** /api/payments/deposit – Client deposits payment
* **POST** /api/payments/withdraw – Freelancer requests withdrawal
* **GET** /api/payments/history – View payment history
* **POST** /api/payments/release – Client approves payment to freelancer

**🔹 Module 6: Review & Rating**

* **POST** /api/reviews/{projectId} – Submit a review
* **GET** /api/reviews/{userId} – Get reviews for a user

**🔹 Module 7: Admin Panel**

* **GET** /api/admin/users – View all users
* **PUT** /api/admin/user/{id}/block – Block/unblock user
* **GET** /api/admin/projects – View all projects
* **DELETE** /api/admin/project/{id} – Delete abusive/spam project
* **GET** /api/admin/disputes – List reported issues

**🔐 3. Authentication & Authorization**

* JWT token stored in HTTP-only cookies or Authorization header.
* Middleware intercepts all protected routes.
* Role-based access for endpoints:
  + /api/projects (Client)
  + /api/proposals (Freelancer)
  + /api/admin/\* (Admin)

**🔄 4. External Services Integration**

| **Service** | **Usage** | **Endpoint Used** |
| --- | --- | --- |
| **Razorpay** | Payment processing | /api/payments/\* |
| **SendGrid** | Emails for registration/hiring | Triggered via service layer |
| **Firebase** | Realtime chat | /chat/messages (RTDB) |

**📌 Suggestions for Diagram Drawing (drag & drop flow):**

**Client Side:**

* Login/Register
* Project Board
* Proposal Dashboard
* Wallet
* Chat

**Arrows to → Spring Boot Backend**

**Backend connects to:**

* ✅ MySQL (DB)
* ✅ Razorpay (Payment API)
* ✅ Firebase (Chat)
* ✅ SendGrid (Mail API)

Next up:  
➡️ **Low-Level Design (LLD)** – detailing the structure of classes, services, and request/response DTOs.

Reply **continue** when you're ready.