Thanks! Based on the **diagram you've shared** and the **Skill-Sharing & Learning Platform project**, here's a breakdown of the project components along with **feature-wise responsibility division** and **sample diagrams/images suggestions** you can include in your documentation or presentation.

**📂 1. Breakdown of Project as per Diagram Structure**

**📝 Initial Doc (Main Document Components):**

* Description
* Functional Requirements
* Non-Functional Requirements
* Designs (System + Architecture Diagrams)

**🔍 2. Functional & Non-Functional Requirements Mapping**

**➤ Functional Requirements (Features)**

**Handled by All Members (Feature-wise split):**

* Skill Sharing Posts
* Learning Progress Updates
* Learning Plan Sharing
* Social Interactions (Likes/Comments)
* Follow/Unfollow Users
* User Profiles
* OAuth 2.0 Authentication
* Notifications

**➤ Non-Functional Requirements**

**Mainly handled by Member 4 (with contributions from all):**

* Performance (API response time, UI responsiveness)
* Security (OAuth, Role-based access)
* Scalability (Modular REST services)
* Maintainability (Clean code structure, documentation)
* Usability (UI/UX Design Standards)

**🎯 3. REST API vs Client Web App Responsibilities**

| **Area** | **REST API (Spring Boot)** | **Client Web App (React)** |
| --- | --- | --- |
| Posts | CRUD for Posts, Media Upload | Create Post UI, Display Feed |
| Learning Updates | CRUD APIs | Timeline Form, Display Updates |
| Learning Plans | CRUD APIs | Resource Planner UI |
| User Management | Follow/Unfollow APIs, OAuth | Profile UI, Followers View |
| Comments & Likes | Backend Logic + DB Models | Like Button UI, Comment Box |
| Notifications | Logic & Fetching APIs | Notification Bell UI |
| Auth | OAuth 2.0 Flow | Login Button, Redirects |

**🏗 4. Architecture Diagrams You Can Include**

**📌 A. System Architecture Diagram**

(Shows how all components interact at a high level)

**Include elements like:**

* Frontend (React Web App)
* Backend (Spring Boot REST API)
* Database (MySQL/PostgreSQL)
* OAuth Server (Google/Auth0)
* Notification Service
* File Storage (Local/Cloud for Media)

🖼 **Sample Illustration Example:**

Client Browser (React)

↓

Frontend Web App (React)

↓ ↘

REST API (Spring Boot) Auth Server (OAuth 2.0)

↓

Database (MySQL/PostgreSQL)

↓

Notifications / Media Service

**📌 B. Detailed Architecture for REST API**

* Show modules: PostController, UserController, AuthController, PlanController, etc.
* Show Service, Repository, Model layers.

🖼 Example Structure:

Controller Layer → Service Layer → Repository → DB

↑

Security Layer (OAuth AuthFilter)

**📌 C. Detailed Architecture for Client Web Application**

* Components like:
  + Login Page
  + Skill Post Feed
  + Learning Plan Page
  + Profile Page
  + Notification Icon
  + Comment Box
  + Sidebar Menu
* State Management (Redux/Context)
* API Integration (Axios)

🖼 Example:

App.js

├── Navbar

├── Sidebar

├── Routes

├── /home → SkillFeed.js

├── /plan → LearningPlan.js

├── /profile → ProfilePage.js

├── /login → OAuthLogin.js

**📌 5. Task Responsibility Mapping to Diagram**

| **Diagram Section** | **Assigned Member** | **Task** |
| --- | --- | --- |
| Functional Requirements (Posts, Progress, Plan, Comments) | Member 1, 2, 3 | Implement feature APIs + UI |
| Non-functional Requirements | Member 4 | Security, usability, scalability |
| System Architecture Diagram | Member 4 (lead) | Full system architecture sketch |
| Detailed REST API Architecture | Member 1 (lead) | REST endpoints, services, controllers |
| Detailed Client Web App Architecture | Member 3 (lead) | React components & routing flow |

**📸 Would You Like Me To:**

1. Generate these architecture diagrams as **image files**?
2. Provide a **ready-to-use PowerPoint/Google Slides Template** with diagrams filled in?
3. Create a **Markdown or Word doc format** with these structures?

Let me know and I’ll generate them right away!

Perfect! Here's a detailed breakdown of the **Functional and Non-Functional Requirements** separately for both the **Client Web Application (React)** and **REST API (Spring Boot)** based on your **Skill-Sharing & Learning Platform project** and the structure from your assignment document and diagram.

**✅ 1. Functional Requirements**

**📌 A. Client Web Application (React)**

These are the functionalities that users will interact with via the frontend interface.

| **Feature** | **Description** |
| --- | --- |
| User Authentication (OAuth 2.0) | Google or GitHub login button integration, redirect on successful login. |
| Skill Sharing Post Creation | Users can create and publish skill-sharing posts with text/images. |
| Skill Feed View | Dynamic feed displaying all posts from followed users or global feed. |
| Learning Progress Updates | Interface to update personal learning progress on specific skills/topics. |
| Learning Plans | UI to create, edit, and display personal or shared learning plans. |
| Commenting & Liking | Users can like posts or add comments through intuitive UI elements. |
| Social Interactions | Follow/Unfollow buttons, display follower/following counts. |
| Profile Management | View and edit profile, see own posts and learning updates. |
| Notification Interface | UI notification bell or dropdown to display alerts (likes, comments, new followers). |
| Responsive Design | Full mobile and desktop responsiveness using CSS frameworks. |

**📌 B. REST API (Spring Boot Backend)**

These are the core functionalities and logic that power your system.

| **Feature** | **Description** |
| --- | --- |
| OAuth 2.0 Authentication | API routes to handle OAuth token validation, user creation on first login. |
| User Management | REST endpoints for get/update profile, follow/unfollow actions. |
| Post Management | CRUD APIs for skill-sharing posts (create, read, update, delete). |
| Comment & Like APIs | Endpoints to like/unlike and post/delete comments on posts. |
| Learning Progress APIs | Create, update, fetch learning progress records for users. |
| Learning Plan APIs | Endpoints to manage learning plans: add goals, steps, deadlines. |
| Notification APIs | Generate & fetch notification data based on actions like comment/like/follow. |
| Role-Based Access Control | Optional admin privileges for platform moderation features. |
| File Handling | APIs for image/media upload (if applicable) and retrieval. |
| API Documentation | Swagger UI / OpenAPI spec to document and test endpoints. |

**⚙️ 2. Non-Functional Requirements**

**📌 A. Client Web Application (React)**

| **Requirement** | **Description** |
| --- | --- |
| Usability | Easy-to-navigate interface with clear icons, buttons, and layouts. |
| Performance | Minimal loading time via lazy loading and optimized bundle sizes. |
| Accessibility | WCAG standards: keyboard navigation, alt-texts, color contrasts. |
| Responsiveness | Adaptive UI across all device sizes using responsive CSS frameworks. |
| Maintainability | Modular React components, reusable hooks, clean code structure. |
| Scalability | Can easily add new pages/features with routing and modular design. |
| Error Handling | User-friendly error messages and fallback UIs. |
| Security | Secure storage of auth tokens (via HTTP-only cookies/local storage), prevent XSS/CSRF. |

**📌 B. REST API (Spring Boot Backend)**

| **Requirement** | **Description** |
| --- | --- |
| Performance | Optimized DB queries, caching, and asynchronous processing where necessary. |
| Security | OAuth 2.0-based access control, HTTPS usage, input validation, SQL injection prevention. |
| Scalability | Layered architecture (Controller → Service → Repository), enabling horizontal scaling. |
| Maintainability | Clean package structure (controllers, services, models), properly commented code. |
| Reliability | Exception handling mechanisms to avoid server crashes and provide graceful fallback. |
| API Rate Limiting | Optional - to control abuse/spam and prevent DDoS. |
| Logging & Monitoring | Log errors, requests, and service health (Spring Actuator, Logback, etc.). |
| API Documentation | Swagger/OpenAPI for easy API testing and team collaboration. |

**📑 Summary Table**

| **Area** | **Functional Requirements** | **Non-Functional Requirements** |
| --- | --- | --- |
| Client Web App | UI Features (Posts, Plans, Notifications, Auth) | Responsiveness, Accessibility, UI/UX, Error Handling |
| REST API | Endpoints (Posts, Auth, User, Progress, Plans) | Security, Logging, Rate Limiting, Scalability |

Would you like me to:

* Generate this into a **PDF/Doc/Markdown format**?
* Create a **task checklist for your team members** based on this?
* Generate **tables or images for presentation/assignment submission**?

Let me know, and I’ll prepare it immediately!