Skill-Sharing & Learning Platform - Initial Document

1. Functional Requirements

REST API Functional Requirements

- **User Authentication**: OAuth 2.0 login with social media accounts.
- **Skill Sharing Posts**:
    - Upload up to 3 photos or short videos (max: 30 sec) per post.
    - Add descriptions to posts.
- **Learning Progress Updates**:
    - Users can post updates on their learning journey.
    - Predefined templates to guide input.
- **Learning Plans**:
    - Users create structured learning plans with topics, resources, and timelines.
    - Plans can be updated as users progress.
- **Interactivity**:
    - Like and comment on posts.
    - Edit/delete own comments.
    - Post owners can delete comments on their posts.
- **User Profiles & Social Features**:
    - Profile page displaying user activity.
    - Follow other users to see their posts.
    - Publicly visible profiles.
- **Notifications**:
    - Receive notifications for likes and comments.

React Client Functional Requirements

- **User Interface**:
    - Login page with OAuth 2.0 authentication.
    - Home feed displaying skill-sharing posts.
    - User profile page showing posts and learning progress.
- **Post Creation and Interaction**:
    - Upload media, add descriptions, and share learning updates.
    - Like and comment features.
    - Learning plan creation and editing.
- **Navigation & Notifications**:
    - Menu for navigating between sections.
    - Real-time notifications.

2. Non-Functional Requirements

- **Security**:
  - OAuth 2.0 authentication.
  - Data encryption and secure API endpoints.
- **Performance**:
  - Optimized database queries.
  - Caching for frequently accessed data.
- **Scalability**:
  - RESTful API design allowing easy expansion.
  - Modular front-end components for maintainability.
- **Usability**:
  - Responsive UI for mobile and desktop.
  - User-friendly and intuitive interface.
- **Maintainability**:
  - Clean and well-documented codebase.
  - GitHub version control and CI/CD integration.

3. System Architecture

High-Level Architecture

- **Frontend**: React.js with state management.
- **Backend**: Spring Boot REST API with a PostgreSQL database.
- **Authentication**: OAuth 2.0 integration.
- **Storage**: Cloud storage for media uploads.
- **Deployment**: Hosted on a cloud platform with CI/CD.

REST API Architecture

- **Controllers**: Handle HTTP requests and route them to services.
- **Services**: Business logic layer processing requests.
- **Repositories**: Database interactions using JPA.
- **Security**: Middleware for authentication and authorization.

Frontend Architecture

- **Components**: Reusable UI components (Post, Profile, Notifications).
- **State Management**: Context API or Redux.
- **Routing**: React Router for navigation.

4. Work Breakdown Structure (WBS)

Task	Member Responsible	Deadline
Backend setup (Spring Boot, DB)	Member 1	March 24
OAuth 2.0 authentication	Member 2	March 27
API development (CRUD operations)	Member 3	April 10
Frontend setup (React, Routing)	Member 4	March 24
UI Design & Implementation	Member 4	April 5
API-Frontend Integration	Member 1 & 3	April 15
Testing & Debugging	All Members	April 20
Documentation & Finalization	All Members	April 30