Project Description

The **Study Resource Sharing Platform** is designed as a collaborative space for people to share, discover, and manage study resources in a secure, efficient, and user-friendly environment. The project's primary goal is to facilitate the exchange of Literary materials, such as notes, books, and comics, by providing tools for uploading, organizing, and finding high-quality resources. The intended audience are people to love to read and share their thoughts and interests with other people, that benefit from a centralized repository of shared study resources. By enabling functionalities like comments, ratings, and bookmarks, the platform encourages interaction and feedback, helping users evaluate the value of each resource. The project incorporates file analytics to track views and downloads, allowing contributors to gauge the reach of their resources. Additionally, it provides a robust moderation system for administrators to ensure quality control and uphold standards.

It is designed to support user engagement through features like comments, activity feeds, and notifications, which foster a community atmosphere among users. With its intuitive dashboard, easy-to-navigate UI, and customized user management system, the platform aims to serve as an invaluable Literary resource for its audience, ultimately contributing to an improved Reading experience.

Task	**Start Week**	**Duration (weeks)**
User Authentication	Nov -Week 1	2 weeks
User Management	Nov -Week 2	1 week
Improved Search	Nov -Week 2	2 weeks
Enhanced File Preview	Nov -Week 3	1 week
Comments Feature	Nov -Week 3	1 week
Rating System	Nov -Week 4	1 week
Bookmarking Feature	Nov -Week 4	1 week
Home Page with Dashboard	Dec -Week 1	1 week
Session Management & Security	Dec -Week 1	1 week
Finalizing project features	Dec -Week 2	1 week

My project follows the **MVC architecture** with the **Service Layer pattern** to handle business logic.

1. MVC Architecture

- **Model**: The Resource entity class represents the data structure for resources, defining fields like title, description, tags, uploadDate, and s3Url. The ResourceRepository, which extends JpaRepository. Together, these form the model layer, representing both the data and access to that data in the database.

- **View**: The views (templates like upload.html, uploads.html, and preview.html stored in static templates directory) render the UI for uploading, viewing, and previewing resources. They interact with Thymeleaf to dynamically display data injected by the controller via the Model object.

- **Controller**: The ResourceController and PreviewController handle requests related to resources and previews, coordinating between the view and service layers. These controllers define the endpoints and manage HTTP requests, using the service layer to perform logic on data from the model.

2. Service Layer Pattern

- **Services** (ResourceService, PreviewService): These classes act as the **Service Layer**, containing the business logic of the application. Instead of placing this logic directly in controllers, it is abstracted into services, making controllers lean and focused on handling HTTP requests. For example:

    - ResourceService manages operations like uploading resources to S3 and storing metadata in the database, and retrieving resources.
    - PreviewService handles generating presigned URLs for S3, ensuring secure and temporary access to files.

3. Integration of MVC and Service Layer

In the project, the MVC architecture works seamlessly with the Service Layer pattern:

- Controllers (ResourceController, PreviewController) serve as entry points for HTTP requests, delegating business logic to the appropriate service (ResourceService or PreviewService), which in turn interacts with the model (Resource entity and ResourceRepository).
- This division keeps controllers focused on request handling and view selection, while services handle data processing and interactions with external services (like AWS S3).