

**CS673 Software Engineering**

**Team 2 - DAMPP**

**Software Design Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Dipayan Mazumder | Security Lead | *DM* | 05-10-2024 |
| Pranjal Ekhande | Requirement Lead | *PE* | 05-10-2024 |
| Aman Jain | Configuration Lead | *AJ* | 05-10-2024 |
| Mukul Jangid | Design and Implementation lead | *MJ* | 05-10-2024 |
| Praveen Singh | Team Lead | *PS* | 05-10-2024 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1.0 | Dipayan Mazumder, Pranjal Ekhande | 05-10-2024 | First Draft |
| 2.0 | Aman Jain | 05-28-2024 | Second Draft |
| 3.0 | Aman Jain, Mukul Jangid | 06-10-2-24 | Third Draft |

1.0 [Introduction](#_heading=h.gjdgxs)

2.0 [Software Architecture](#_heading=h.30j0zll)

3.0 [Class Diagram](#_heading=h.1fob9te)

4.0 [UI Design (if applicable)](#_heading=h.3znysh7)

5.0 [Database Design (if applicable)](#_heading=h.2et92p0)

6.0 [Security Design](#_heading=h.tyjcwt)

7.0 [Business Logic and/or Key Algorithms](#_heading=h.3dy6vkm)

8.0 [Design Patterns](#_heading=h.1t3h5sf)

9.0 [Any Additional Topics you would like to include.](#_heading=h.4d34og8)

10.0 [References](#_heading=h.2s8eyo1)

11.0 [Glossary](#_heading=h.17dp8vu)

**Introduction**

The Project Management Service is a crucial component of the project-based learning platform, designed to facilitate the creation, management, and tracking of projects for users. This service will provide a comprehensive set of features to support the project lifecycle, from project initiation to completion.

The key goals of the Project Management Service are:

* Enable users to create and manage projects, including defining project details, setting milestones, and assigning tasks to team members.
* Provide a centralized platform for tracking project progress, monitoring task completion, and generating reports and analytics.
* Integrate seamlessly with other services in the platform, such as User Management, Its UI, and Notification services, to provide a cohesive and collaborative learning experience.
* Implement robust security measures, including authentication, authorization, and data protection, to ensure the confidentiality and integrity of project data.

**Architecture Description: Project Management Microservices**

This architecture diagram illustrates the structure of the ProManager microservice-based project management platform, designed to handle multiple clients' API requests efficiently. The platform is composed of several microservices, each responsible for distinct aspects of project management. Below is a detailed description of each component:

1. ProManager Microservice

The ProManager microservice acts as the core component, orchestrating the interaction between various microservices and handling incoming API requests from multiple clients, including web and mobile applications.

2. User Management Service

Functionality: Manages user-related operations such as registration, login, profile management, and user roles.

API Endpoint: User GET API

Database: Integrated with a relational database (PostgreSQL).

Technology Stack: Spring framework for the backend logic.

3. Project Management Service

Functionality: Handles the creation, editing, management, and deletion of projects. This service allows users to define project details, set milestones, and track project progress.

Database: Uses PostgreSQL for storing project-related data.

Technology Stack: Spring framework for handling business logic and data transactions.

4. Task Management Service

Functionality: Manages task-related operations within projects, including task creation, assignment, tracking, and completion. This service ensures that tasks are distributed and monitored effectively within a project.

Compute Engine: Utilizes a compute engine for handling intensive processing tasks.

Database: Stores task data in a PostgreSQL database.

Technology Stack: Spring framework for managing the backend processes.

5. Comment / Review Management Service

Functionality: Facilitates adding, editing, deleting, and retrieving comments or reviews related to projects and tasks. This service enhances collaboration and feedback mechanisms within the platform.

Orchestration Engine: Deployed and managed using Kubernetes for scalable and efficient microservice orchestration.

Database: Integrates with PostgreSQL to store comment and review data.

Technology Stack: Utilizes Spring framework for backend logic.

6. Multiple Clients

The platform supports multiple clients, including web browsers and mobile devices, which interact with the ProManager microservices via API requests. This ensures a responsive and unified user experience across different platforms.

7. Supporting Technologies

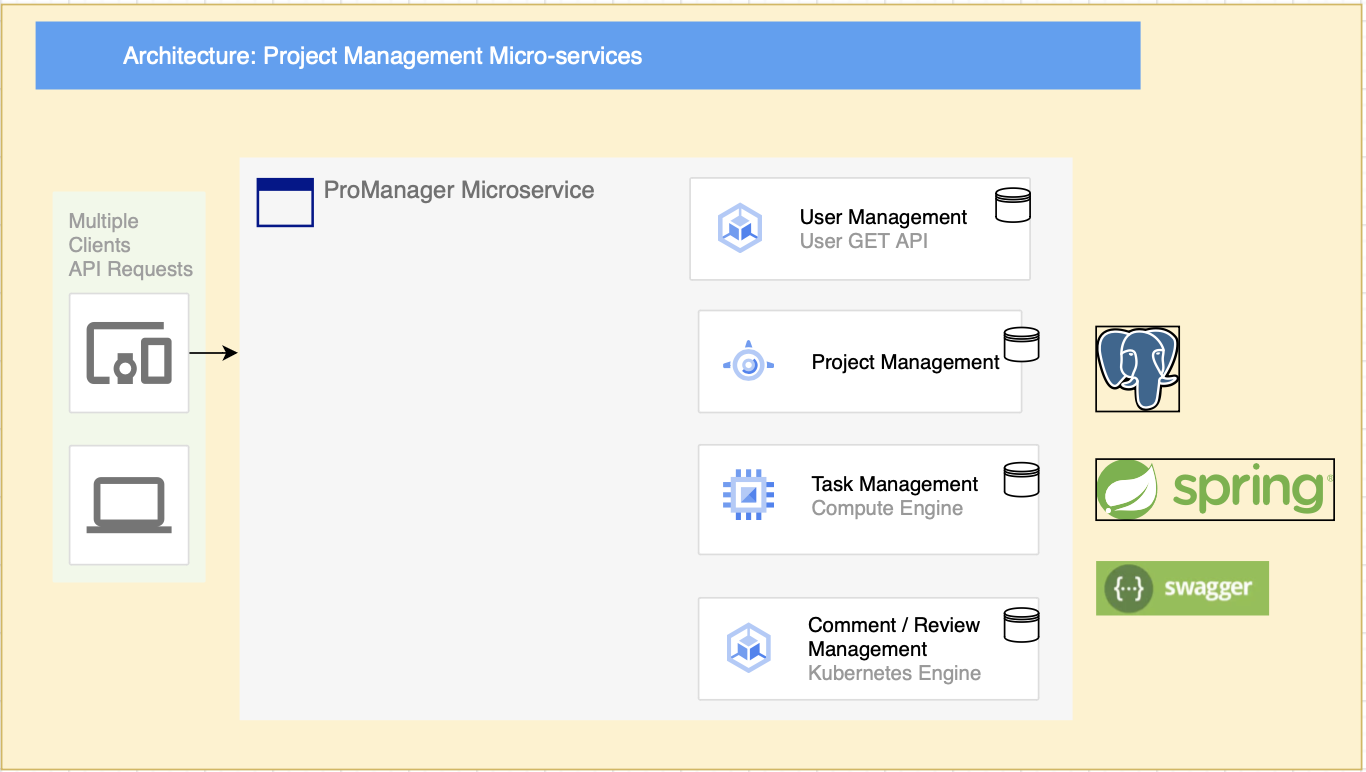
PostgreSQL: The primary relational database used across all services for data storage and management.

Spring Framework: Used extensively for backend logic and RESTful API development, ensuring modularity and ease of maintenance.

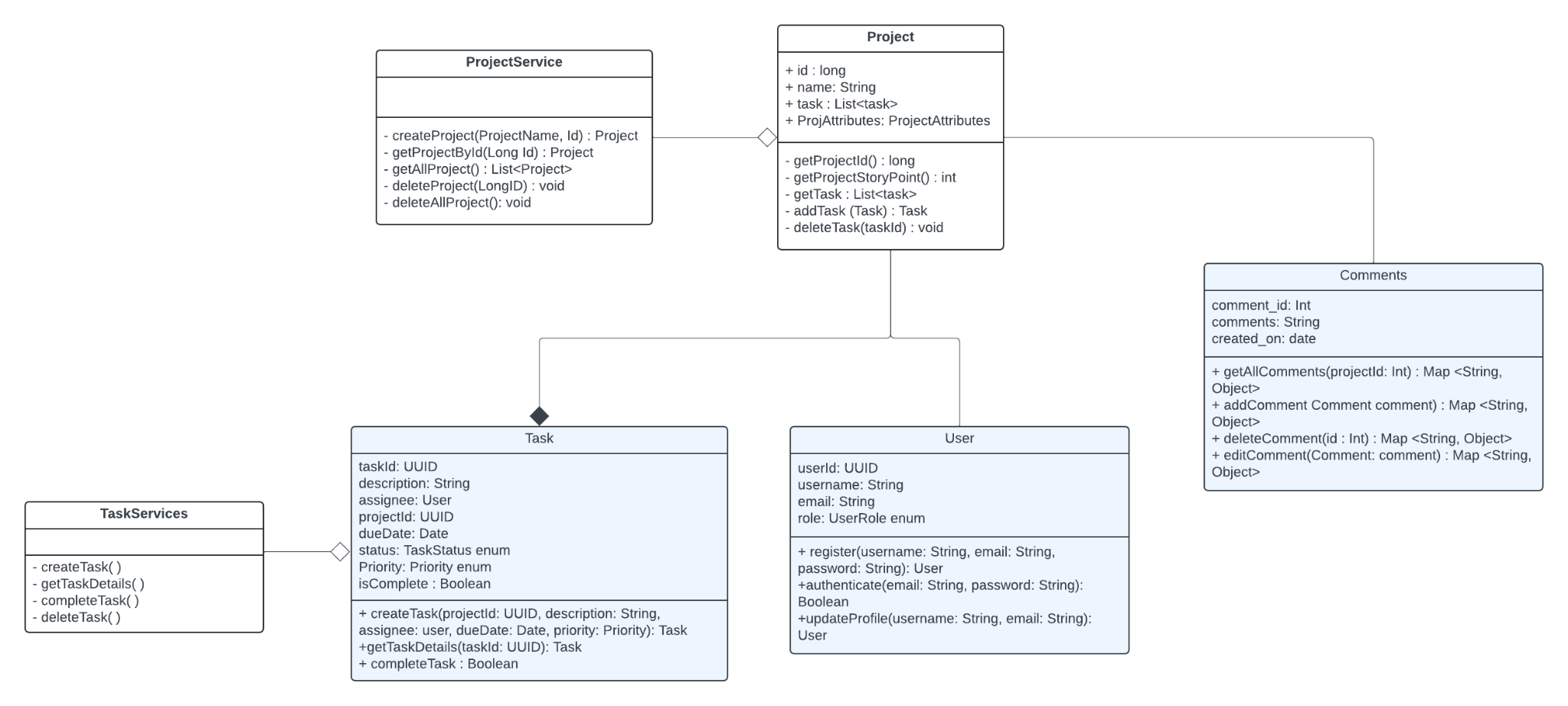
Swagger: Integrated for API documentation and testing, enabling developers to understand and interact with the API endpoints efficiently.

This architecture ensures that each microservice can scale independently, handle specific tasks efficiently, and work together cohesively to provide a robust project management solution.

​​



# Class Diagram



This class diagram represents the architecture of a project management system. The This class diagram represents the architecture of a project management system, showcasing the interactions and relationships between key classes such as Project, Task, User, and Comments. The Project class is designed to manage project-specific details, including project ID, name, associated tasks, and various attributes. It includes operations to create, retrieve, and delete projects, demonstrating a clear separation of responsibilities and promoting modular design. The Task class represents individual tasks within a project, encapsulating details such as task ID, description, due date, status, priority, and the assigned user. This class is equipped with operations for task creation, detail retrieval, and completion, ensuring a comprehensive approach to task management within projects.

The User class is fundamental for managing user information and authentication processes. It includes attributes such as user ID, username, email, and role, along with operations for registering, authenticating, and updating user profiles. This setup allows for robust user management and secure authentication mechanisms, which are critical for maintaining the integrity of the project management system. Additionally, the Comments class is designed to manage user comments on tasks and projects, with attributes like comment ID, text, and creation date, and operations for adding, deleting, and editing comments. This enables effective communication and feedback within the project environment.

ProjectService and TaskServices are service classes that encapsulate the business logic for project and task management. ProjectService handles the creation, retrieval, and deletion of projects, while TaskServices manages the creation, detail retrieval, completion, and deletion of tasks. These services ensure that the core functionalities are implemented efficiently and can be easily maintained or extended. The diagram illustrates the composition relationship between Project and Task, indicating that tasks are integral to projects, and the association between Task and User, showing that tasks are assigned to users. This structured approach facilitates clear and maintainable code, enhancing the overall system's scalability and reliabilit

# **Database Design Description**

# This database design supports the ProManage project management platform by defining the structure and relationships between different entities involved in project and task management. The design consists of five main tables: project\_ci, task\_ci, comments, APP\_USER, and PROJECT\_USER.

# **1. project\_ci Table**

# **Purpose:** Stores information about projects.

# **Columns:**

# project\_id (PK): Primary key, unique identifier for each project.

# projectname: Name of the project.

# description: Detailed description of the project.

# owner\_id (FK): Foreign key referencing APP\_USER.User\_ID, indicates the owner of the project.

# created\_on: Timestamp for when the project was created.

# updated\_on: Timestamp for when the project was last updated.

# status: Current status of the project (e.g., active, completed).

# type: Type of the project.

# active: Boolean indicating if the project is active or not.

# **2. task\_ci Table**

# **Purpose**: Stores information about tasks within projects.

# **Columns**:

# project\_id (FK): Foreign key referencing project\_ci.project\_id, indicates the project this task belongs to.

# task\_id (PK): Primary key, unique identifier for each task.

# task\_name: Name of the task.

# description: Detailed description of the task.

# status: Current status of the task (e.g., to-do, in-progress, done).

# priority: Priority level of the task (e.g., high, medium, low).

# assigned\_user\_id (FK): Foreign key referencing APP\_USER.User\_ID, indicates the user assigned to this task.

# due\_date: Due date for the task completion.

# created\_on: Timestamp for when the task was created.

# updated: Timestamp for when the task was last updated.

# **3. comments Table**

# **Purpose**: Stores comments related to tasks.

# **Columns**:

# comment\_id (PK): Primary key, unique identifier for each comment.

# user\_id (FK): Foreign key referencing APP\_USER.User\_ID, indicates the user who made the comment.

# comments: The text content of the comment.

# created\_at: Timestamp for when the comment was created.

# 4. APP\_USER Table

# **Purpose**: Stores information about users in the system.

# **Columns**:

# User\_ID (PK): Primary key, unique identifier for each user.

# NAME: Name of the user.

# EMAIL: Email address of the user, must be unique.

# PASSWORD: Password for the user account (should be stored securely using hashing).

# 5. PROJECT\_USER Table

# Purpose: Manages the relationship between users and projects, including roles and permissions.

# Columns:

# PROJECT\_USER\_ID (PK): Primary key, unique identifier for each project-user relationship.

# PROJECT\_ID (FK): Foreign key referencing project\_ci.project\_id, indicates the project.

# USER\_ID (FK): Foreign key referencing APP\_USER.User\_ID, indicates the user.

# ROLE: Role of the user within the project (e.g., owner, contributor, viewer).

# created\_at: Timestamp for when the relationship was created.

# updated\_at: Timestamp for when the relationship was last updated.

# Relationships

# One-to-Many Relationship:

# A project (project\_ci) can have multiple tasks (task\_ci), as indicated by the project\_id foreign key in the task\_ci table.

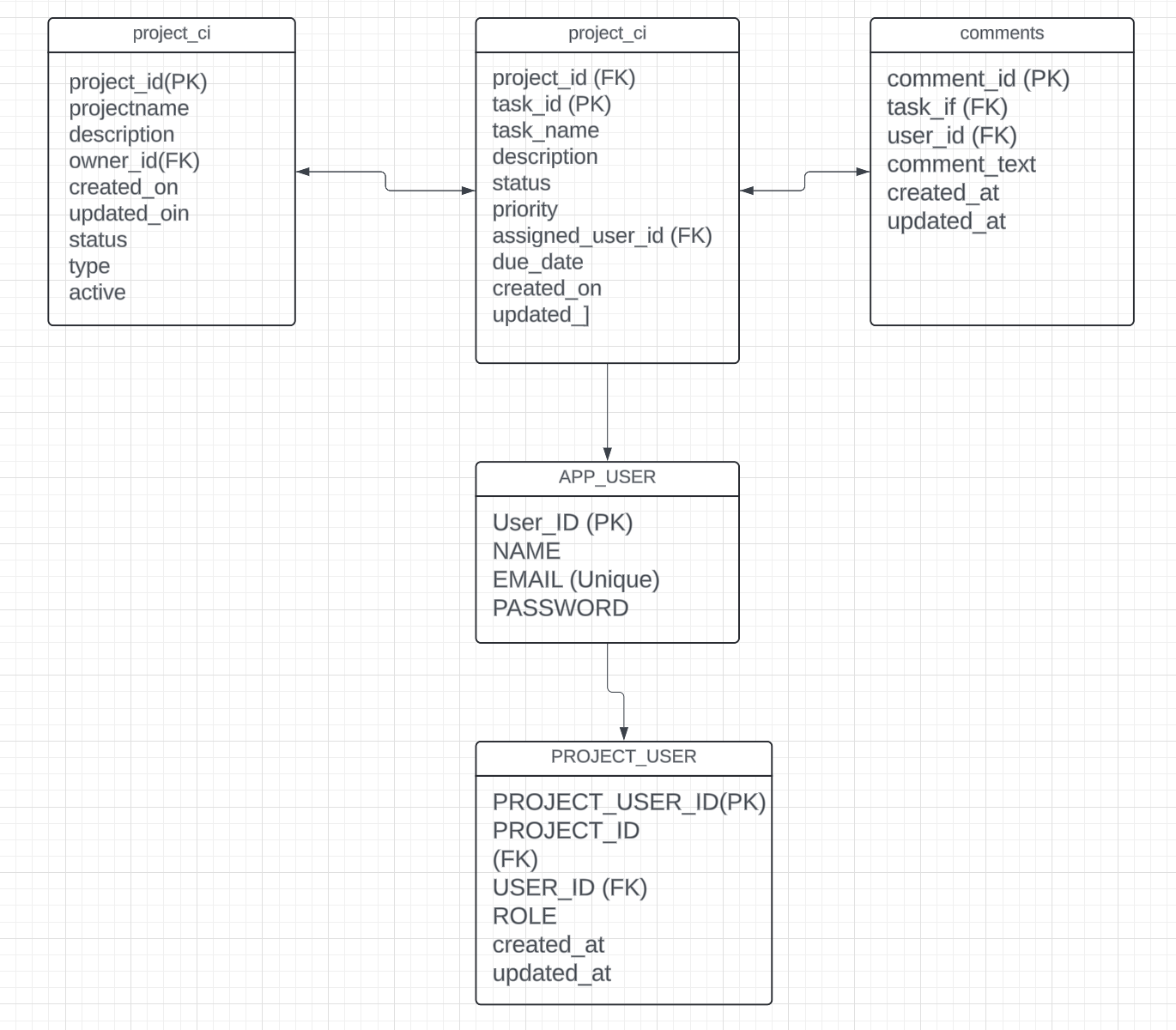
# A task (task\_ci) can have multiple comments (comments), as indicated by the task\_id foreign key in the comments table.

# A user (APP\_USER) can be associated with multiple tasks (task\_ci), as indicated by the assigned\_user\_id foreign key in the task\_ci table.

# A project (project\_ci) can have multiple users with different roles (PROJECT\_USER), as indicated by the project\_id foreign key in the PROJECT\_USER table.

# This database design ensures a well-structured and relational approach to managing projects, tasks, comments, and user roles. It allows for scalability and maintainability, making it easier to extend functionalities in the future.

# 



# 

# Security Design

The Project Management Platform employs a robust security design to safeguard user data and ensure system integrity. By integrating secure authentication and authorization mechanisms, encrypting sensitive data, validating inputs, ensuring secure communication, maintaining audit logs, and adhering to secure coding practices, the platform provides a secure environment for managing projects and collaborating effectively. These measures collectively ensure the confidentiality, integrity, and availability of data, thereby maintaining a high level of trust and security for all users.

# Design Patterns

The Project Management Service leverages the following design patterns to enhance maintainability, scalability, and flexibility:

1. **Repository Pattern**:
   * **Description**: Abstracts the data access layer and provides a centralized way to interact with the database.
   * **Usage**: Employed in ProjectService and TaskService to handle CRUD operations for projects and tasks without exposing the underlying database details.
2. **Observer Pattern**:
   * **Description**: Implements a subscription mechanism to notify interested parties when project or task updates occur.
   * **Usage**: Can be used to notify users about updates, integrating with a notification service.
3. **Strategy Pattern**:
   * **Description**: Defines a family of algorithms or strategies, encapsulates each one, and makes them interchangeable.
   * **Usage**: Used to implement different task scheduling and reporting strategies, enabling easy switching between different algorithms for task management and project reporting.
4. **Decorator Pattern**:
   * **Description**: Adds additional responsibilities to an object dynamically, providing a flexible alternative to subclassing.
   * **Usage**: Employed to extend the functionality of project or task objects by adding additional behaviors such as logging, validation, or formatting.
5. **Facade Pattern**:
   * **Description**: Provides a simplified interface to a complex subsystem, making it easier to use.
   * **Usage**: Used in ProjectService and TaskService to offer a simplified interface for complex operations like project reporting or task scheduling, hiding underlying complexities

# Integration Considerations

# **Repository Pattern**: Applied in ProjectService and TaskService.

**Observer Pattern**: Potentially integrated to notify changes.

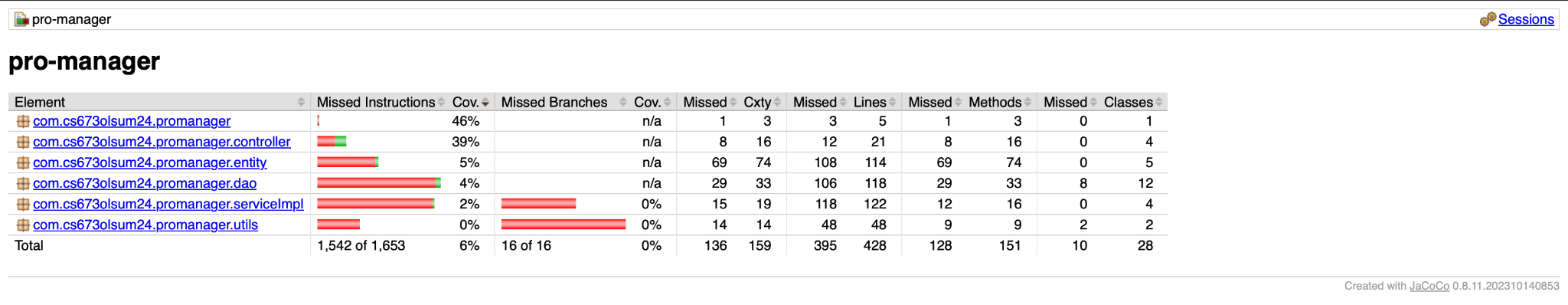
**Strategy Pattern**: Used for scheduling and reporting strategies.

**Decorator Pattern**: Extends Task and Project functionalities.

**Facade Pattern**: Simplified interface through ProjectService and TaskService.

# Any Additional Topics you would like to include.

**Code Coverage**



# References

# Glossary

UML - Unified Modelling Language,

API: Application Programming Interface

HTTP: Hypertext Transfer Protocol

UI: User Interface