**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 |

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**

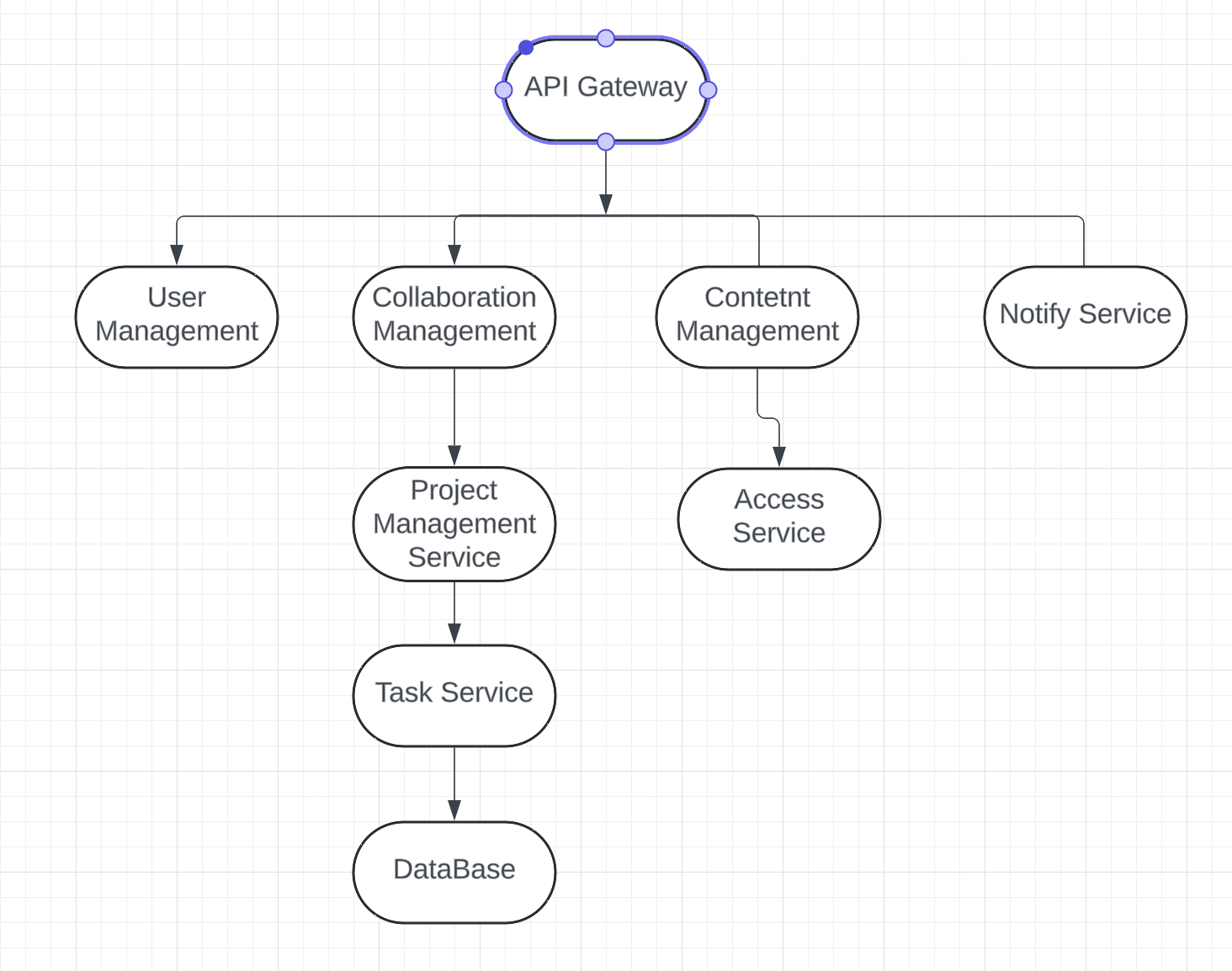
The Project Management Service is a crucial component of the project-based learning platform, designed to facilitate the creation, management, and tracking of projects for users. This service will provide a comprehensive set of features to support the entire project lifecycle, from project initiation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

The key goals of the Project Management Service are:

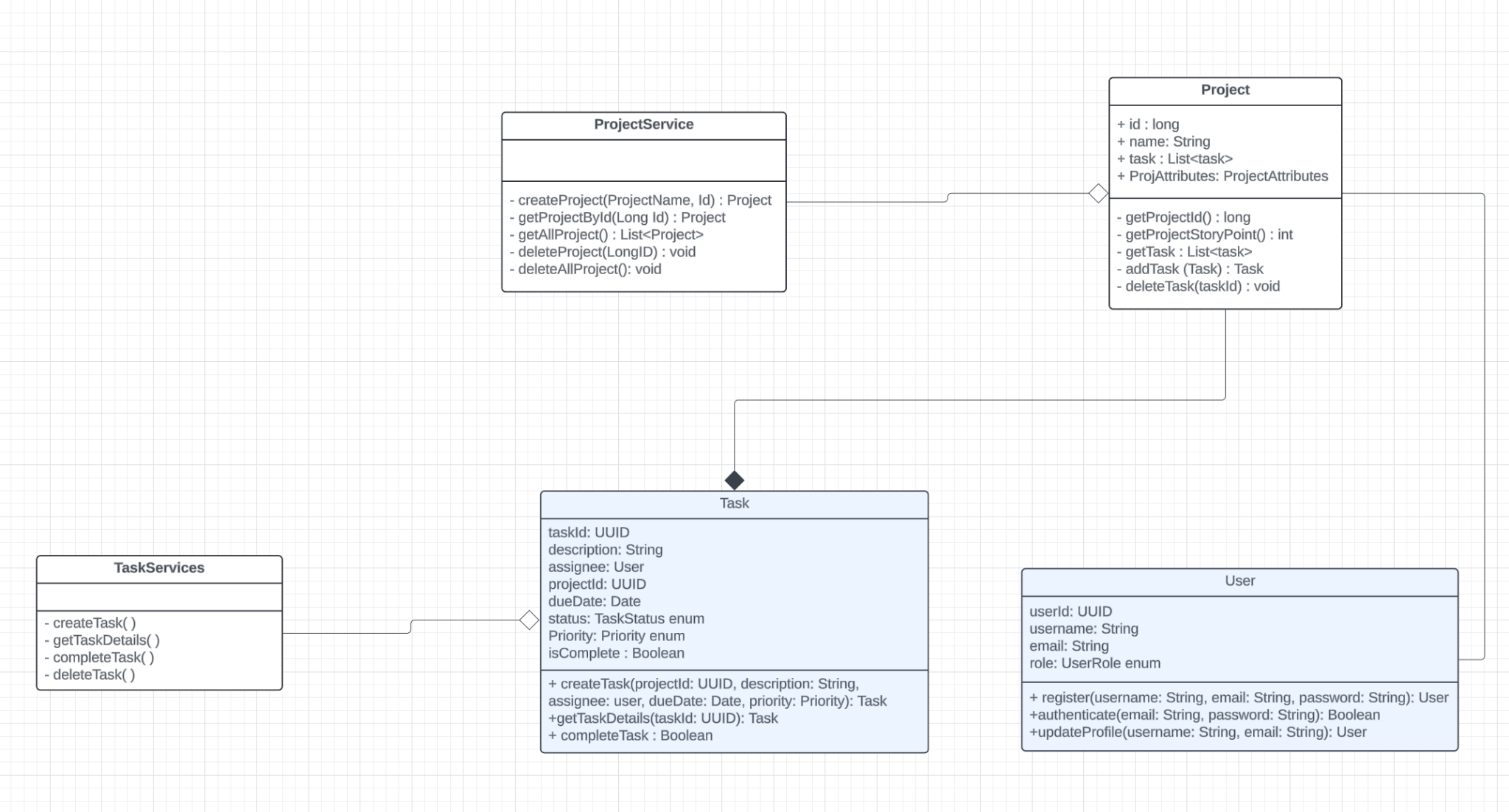
* Enable users to create and manage projects, including defining project details, setting milestones, and assigning tasks to team members.
* Provide a centralized platform for tracking project progress, monitoring task completion, and generating reports and analytics.
* Integrate seamlessly with other services in the platform, such as User Management, Collaboration, 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.

**Software Architecture**

The Project Management Service will be implemented as a microservice within the project-based learning platform's microservice architecture. It will communicate with other services using RESTful APIs and follow the principles of loose coupling and high cohesion. The service will be designed with a layered architecture, separating concerns and responsibilities into distinct layers, such as the Presentation Layer (API controllers), Business Logic Layer (service classes), and Data Access Layer (repositories). An API Gateway will act as a single entry point for client requests, routing them to the appropriate services and handling cross-cutting concerns like authentication, rate limiting, and caching. The following diagram illustrates the high-level software architecture of the project-based learning platform, highlighting the Project Management Service and its interactions with other services:

​​  


# Class Diagram



The Project Management Service will consist of the following main classes and their relationships:

**Project:** Represents a project and its associated data. Attributes include:

* name (String)
* description (String)
* start\_date (Date)
* end\_date (Date)
* status (String) - The class diagram doesn't show the data type but it likely represents the project's status (e.g.,"active", "completed", "on hold").
* tasks (List<Task>) - One-to-Many relationship with Task. A project can have many tasks.
* projectAttributes (ProjectAttributes) - This could be a custom class that holds additional project-specific data.

**Task:** Represents a task within a project and its associated data. Attributes include:

* task\_id (UUID) - Likely a unique identifier for the task.
* description (String)
* assignee (User) - A foreign key referencing the User class. Represents the user assigned to the task.
* project\_id (UUID) - A foreign key referencing the Project class.
* due\_date (Date)
* status (TaskStatus) - The class diagram uses an enum called TaskStatus. This likely represents the status of the task (e.g., "To Do", "In Progress", "Done").
* priority (Priority) - Another enum representing the priority of the task (e.g., "High", "Medium", "Low").

**User:** Represents a user of the platform and includes:

* user\_id (UUID) - Likely a unique identifier for the user.
* username (String)
* email (String)
* password (String) - Though it's not recommended to store passwords in plain text, this is a common way it's represented in class diagrams. In reality, passwords should be hashed before storing.
* role (UserRole) - An enum representing the user's role (e.g., "admin", "project\_manager", "team\_member").

# Database Design

# 

The Project Management Service will utilize a relational database to store project-related data. The following database schema illustrates the main tables and their relationships:

Projects: Stores details of each project, including name, description, dates, status, and owner.

Tasks: Stores details of tasks, including description, due date, status, assignee, and completion status, and links to the project they belong to.

Users: Stores user information such as username, email, hashed password, and role.

ProjectMembers: Acts as a join table to manage the assignment of users to projects and their roles within those projects.

This updated database design ensures that the data is organized according to the relationships and attributes specified in your class diagram, facilitating efficient storage and retrieval of project-related data.

# Security Design

The Project Management Service will implement the following security measures:

**Authentication and Authorization**: Integration with the User Management Service for user authentication and role-based access control (RBAC). JSON Web Tokens (JWT) will be used for stateless authentication and authorization.

**Data Encryption**: Sensitive data, such as user passwords and project data, will be encrypted using industry-standard encryption algorithms like AES-256.

**Input Validation**: All user input will be validated and sanitized to prevent injection attacks like SQL injection and Cross-Site Scripting (XSS).

**Secure Communication**: Communication between the Project Management Service and other services will be secured using HTTPS and encrypted using TLS/SSL.

**Audit Logging**: Audit logs will be maintained to track user actions, including project creation, updates, and deletions, for auditing and forensic purposes.

**Secure Coding Practices**: The service will be developed following secure coding best practices, such as the OWASP Top 10 security guidelines, to mitigate common web application vulnerabilities.

# Business Logic and/or Key Algorithms

The Project Management Service will implement the following key algorithms and workflows:

**Project Creation and Initialization**: This workflow will handle the creation of new projects, including validating user input, initializing project details, and setting up project members and their roles.

**Task Scheduling and Dependency Management**: An algorithm will be implemented to handle task scheduling, taking into account task dependencies, project milestones, and resource availability.

**Project Progress Tracking**: A progress tracking algorithm will calculate the overall project progress based on completed tasks, task dependencies, and project milestones. **Project Reporting and Analytics**: Algorithms will be implemented to generate various reports and analytics, such as project progress reports, task completion reports, and team performance reports.

# 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.

N/A

# References

Pending

# Glossary

Pending