**CS673 Software Engineering** 

**Team 2 - ProManager**

**Project Proposal and Planning**

| 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 | Pranjal Ekhande, Praveen Singh | 05-10-2024 | First Draft |
| 2.0 | Aman Jain, Dipayan Mazumder | 05-28-2024 | Second Draft |
| 3.0 | Dipayan Mazumder | 06-11-2024 | Third Draft |
| 4.0 | Pranjal Ekhande, Aman Jain, Mukul Jangid | 06-17-2024 | Final Draft |

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

2.0 [Related Work](#_heading=h.30j0zll)

3.0 [Proposed High level Requirements](#_heading=h.1fob9te)

4.0 [Management Plan](#_heading=h.3znysh7)

4.1 [Objectives and Priorities](#_heading=h.2et92p0)

4.2 [Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

4.3 [Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

5.0 [Configuration Management Plan](#_heading=h.1t3h5sf)

5.1 [Tools](#_heading=h.4d34og8)

5.2 [Deployment Plan if applicable](#_heading=h.2s8eyo1)

6.0 [Quality Assurance Plan](#_heading=h.17dp8vu)

6.1 [Metrics](#_heading=h.3rdcrjn)

6.2 [Code Review Process](#_heading=h.26in1rg)

6.3 [Testing](#_heading=h.lnxbz9)

6.4 [Defect Management](#_heading=h.35nkun2)

7.0 [References](#_heading=h.1ksv4uv)

8.0 [Glossary](#_heading=h.44sinio)

# Overview

We are proposing to contribute to a learning platform with **Project Management** Module that will be designed to facilitate project-based learning for students and professionals. The platform will provide an environment where users can **create** and **manage** **projects**, **access** **learning** resources, **collaborate** with team members, and **receive** **assessments** and **feedback** on their work.

We are planning to build a platform that will consist of several microservices, including User Management, Project Management. These services will work together to provide a seamless and comprehensive experience for users. The Project Management Service will be a crucial component of the platform, enabling users to create and manage projects, assign tasks, track progress, and collaborate with team members. This service will integrate with other services, such as User Management for authentication and authorization, Collaboration for real-time communication and document sharing, and Notification for sending project-related notifications.

# Related Work

Several project management tools and platforms exist in the market, such as Jira, Trello, Asana, and GitHub Projects. However, most of these tools are designed for professional software development teams and lack features specific to project-based learning, such as integration with learning resources, assessments, and feedback mechanisms.

Some learning management systems (LMS) like Blackboard and Canvas offer basic project management capabilities, but they are primarily focused on traditional course delivery and lack advanced collaboration and project tracking features. We are proposing a project-based learning platform that aims to bridge this gap by providing a comprehensive solution for project-based learning, combining project management functionalities with learning resources, collaboration tools, and assessment features.

# Proposed High level Requirements

# **Functional Requirements**

# **Essential Features**:

a. **Project Creation**

* As a user, I want to create a new project and edit the project, read the project, safely delete the project and delete the project so that I can manage tasks and resources.
* Estimated person-hours: 140-150

b. **Task Management**

* As a project member, I want to create, update, read and add tasks as complete, so that I can track my progress and responsibilities.
* Estimated person-hours: 120-130

c. **Project Access Control**

* As a project owner, I want to control who can view/edit and add new user to the project and its tasks, so that I can manage project access and privacy.
* Estimated person-hours: 90-100

d. **Comment Management**

* As a project member, I want to create, update, read and delete comments, so that I can my progress and responsibilities.
* Estimated person-hours: 50-60

# **Desirable Features** :

a. **Integration with Other Services**

* Integrate with the User Management Service for authentication and authorization.
* Integrate with the Collaboration Service for discussions, document sharing, etc.
* Integrate with the Notification Service to send project-related notifications.

b. **Project Templates**

* As a user, I want to create new projects from pre-defined templates for common project types, so that I can quickly set up projects with predefined structures and tasks.

c. **Project Reporting and Analytics**

* As a project owner/admin, I want to view reports and analytics on project progress, task completion, team performance, etc., so that I can monitor and evaluate project status.

# **Optional Features**:

1. Advanced reporting and visualization tools for project data.

2. Project portfolio management for managing multiple projects at an organizational level.

3. Integration with external project management tools or APIs.

# Management Plan

* 1. **Objectives and Priorities**

1. Deliver a fully functional and reliable Project Management Service that meets all essential requirements.

2. Maintain high code quality and test coverage for the Project Management Service.

3. Ensure proper integration and communication with other services, such as User Management, Collaboration, and Notification.

# **Risk Management**

The main risks identified for the Project Management Service include:

1. Complexity in handling project dependencies and concurrent updates from multiple users.

2. Challenges in integrating with other services, such as ensuring consistent data and communication protocols.

3. Potential performance issues or scalability concerns as the number of projects and users grows.

4. Security risks related to data breaches, unauthorized access, or vulnerabilities in the service.

**To manage these risks, we will:**

* Implement proper locking mechanisms and conflict resolution strategies for concurrent updates.
* Establish clear communication protocols and data contracts between services, and conduct thorough integration testing.
* Optimize the service for performance and scalability, and implement load testing and monitoring.
* Follow secure coding practices, conduct security testing, and regularly update dependencies and libraries.

**Risk Management Sheet Link:**

# **Timeline**

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | Iteration 0 | Environment Setup, Documentation, Initialisation | 8-9 |
| 2 | Project Creation  Task Management  Project Access Control (Essential) | Design, Implementation, Testing  Design, Implementation, Testing  Design, Implementation, Testing | 40-50  60-80  30-40 |
| 3 | Project Search and Filtering (Essential)  Project Reporting and Analytics (Desirable) | Design, Implementation, Testing  Design, Implementation, Testing | 20-30  60-80 |
| 4 | Integration with Other Services (Desirable)  Project Templates (Desirable)  Optional Features (if time permits) | Design, Implementation, Testing  Design, Implementation, Testing | TBD  TBD |

# Configuration Management Plan

# **Tools**

| Tech | Tool |
| --- | --- |
| Version Control | Git and GitHub |
| IDE | SpringBoot STS or IntelliJ. |
| CI/CD | Github |
| Dependency Management | Maven |

Code Commit Guideline and Git Branching Strategy

We have a basic Branching Strategy which might change as required but basic branching strategy would be as follows

* **Main** - This is the main production branch, containing the stable and deployable code.
* **Develop** - This is the integration branch where new features and bug fixes are merged before being promoted to the main branch.
* **Feature** -These are created from the develop branch and used for implementing new features or enhancements. Once complete, they are merged back into the develop branch via pull requests.
* **Bugfix** - These branches will be created from the Main branch and will be used to fix any bug on production.

Code Commit Guidelines:

* Developers will create feature branches from the develop branch and work on their assigned tasks within their respective branches.
* Commit messages should be clear, descriptive, and follow a consistent format.
* Before pushing changes to the remote repository, developers should ensure their code is properly tested and passes all existing tests.
* Pull requests should be created from feature branches to the develop branch for code review and integration.
* Code reviews should be conducted by at least one other team member before merging pull requests into the develop branch.

# **Deployment Plan if applicable:**

We will be deploying our application by using Heroku.

The Project Management Service will be deployed as a containerized microservice using Docker. The containers will be orchestrated and managed using Heroku, enabling scalability, load balancing, and high availability.

Database Deployement - Heroku

1. Sign up for Heroku - To deploy your database you need to [sign up](https://signup.heroku.com/) a free Heroku account. You need to fill in the data and just click on Create a free account.
2. Create a new app Once you have completed the registration process, you will see this page. Just click on Create new app. You also have to choose a region.
3. Configure your application - Go to the Resources menu. Just search for Postgres in the add-ons and select it.
4. Find the credentials - Now you have a working Postgres database. You need to find the credentials. Click on Heroku Postgres to find the data you need.
5. Add this credential in SpringBoot application

To deploy the application using Docker and Heroku, we will use following terminal command:

1. Login to Heroku - Log in to your Heroku account:

Command - login heroku

2. Build the Docker image - Build the Docker image for your application:

Command - `docker build -t registry.heroku.com/promanager-v1/web .`

3. Push the Docker image to Heroku - Push the built Docker image to the Heroku Container Registry:

Command - `docker push registry.heroku.com/promanager-v1/web`

4. Release the Docker container on Heroku - Release the Docker container on Heroku to deploy your application:

Command - `heroku container:release web -a APPLICATION\_NAME`

5. View the application logs - Monitor the deployment process and application behavior by viewing the logs:

heroku logs --tail -a promanager-v1

The deployment environments will include:

1. Development Environment: will be Used by developers for local testing and development.

2. Staging Environment: A pre-production environment for integration testing and staging deployments.

3. Production Environment: The live environment is accessible to end-users.

The deployment strategy will be a rolling update, where new versions of the service are gradually rolled out, minimizing downtime and ensuring zero-downtime deployments. Heroku inherits the functionalities that we require for this project

# Quality Assurance Plan

# **Metrics**

| Metric Name | Description |
| --- | --- |
| Code coverage  (JaCoCo) | Measure the percentage of code covered by unit tests to ensure adequate testing of the codebase. We are using JaCoCo which is a Java Code Coverage tool. The team has achieved more than 50+% code coverage. |
| Developer’s time and number of bugs | Evaluate the number of bugs that we found on each level of testing and how much effort it takes to fix those bugs by the developers. |
| Test Execution Time | Measure the time taken to execute the entire test suite, which can indicate the need for test optimization or parallelization. We are manually monitoring this time. |
| Cyclomatic Complexity | Cyclomatic Complexity is a software metric used to measure the complexity of a program by counting the number of linearly independent paths through a program's source code. |
| Code Duplication | Monitor the percentage of duplicate code across the codebase to identify opportunities for code reuse and maintainability improvements. |

# **Coding Standard**

| Language | Coding Standards |
| --- | --- |
| Java Duplication | Google Coding Standards |
| Python | Black |

# **Code Review Process**

For the Development branch the following processes will be followed

Development branch –> Dev Testing –> PR raise –> peer review –> Testing –> Dev (preprod) –> Prod –> Ticket close

Peer Review would be mainly done by Mukul and Pranjal.

# **Testing**

| Testing | Description |
| --- | --- |
| JUnit | Java Testing |
| Postman | API testing |

For manual testing we will cover exploratory testing, usability testing and regression testing.

For automation testing, we will include unit testing and integration testing.

# References

1. <https://docs.jetadmin.io/jet-bridge-deployment/database-heroku-deployment>

# Glossary

(Any acronym used in the document should be explained here)