# SWE 6633 SOFTWARE PROJECT PLANNING & MANAGEMENT
## Group 7 - Spring 2023

# Comprehensive Plan

# “PROJECT MANAGEMENT SYSTEM”

To Professor Dr. Hassan Pournaghshband (Ph.D.) Dept. Of Software Engineering College of Computing and SWE, Kennesaw State University

By Ryan O'Connor, Samuel Owoade, Cameron Page, Milly Namukasa and Harini Pammi

Version 2.2

Date: 3/26/2023

# Table of Contents

# Problem and Requirements

## A. Project Description and Introduction

This project aims to develop a web-based project management system that allows users to plan, track, and collaborate on projects. The system will support multiple users, teams, and projects, and will integrate with third-party tools such as calendars and document management systems. The project team will consist of five members, Ryan O'Connor, Samuel Owoade, Cameron Page, Milly Namukasa and Harini Pammi. The project duration will be within a 16-week semester.

## *B. Project Goals*

1. To develop a web-based project management system that allows users to plan, track, and collaborate on projects.
2. To support multiple users, teams, and projects within the system.
3. To integrate the system with third-party tools such as calendars and document management systems for enhanced functionality and usability.
4. To provide a user-friendly interface that simplifies the project management process and improves productivity.
5. To ensure the security and privacy of user data and project information through appropriate measures such as authentication and encryption.

## *C. User Stories*

1. As a project manager, I want to be able to create and assign tasks to team members so that we can track our progress and stay on schedule.
2. As a team member, I want to be able to see a calendar view of all the tasks assigned to me so that I can plan my workday effectively.
3. As a project manager, I want to be able to see a dashboard view of all my projects so that I can quickly assess their status and prioritize my tasks.
4. As a team member, I want to be able to upload and share files related to my tasks so that other team members can access them easily.
5. As a project manager, I want to be able to set deadlines and receive notifications when they are approaching so that I can ensure that the project stays on track.
6. As a team member, I want to be able to communicate with other team members via a messaging system so that we can discuss our tasks and collaborate effectively.
7. As a project manager, I want to be able to create milestones and track progress towards them so that I can see how far we have come in the project.
8. As a team member, I want to be able to see a list view of all the tasks assigned to me so that I can easily see what needs to be done next.
9. As a project manager, I want to be able to generate reports on the progress of my projects so that I can share them with stakeholders and make informed decisions.
10. As a team member, I want to be able to log my time spent on tasks so that I can accurately track my progress and ensure that I am meeting deadlines.

# Product Description

## A. Project Scope

The complete scope of the project can be broken down into several key components:

1. User management: The system will have a user management module that allows users to register, login, and manage their accounts. The system will also support different user roles such as admin, project manager, and team member.
2. Project management: The system will allow users to create projects, assign tasks, set deadlines, and track progress. Users will also be able to collaborate on projects by sharing documents and communicating with each other through the system.
3. Team management: The system will support team management features such as creating and managing teams, assigning team members to projects, and setting team permissions.
4. Calendar integration: The system will integrate with third-party calendar systems to allow users to schedule tasks and deadlines.
6. Reporting: The system will have reporting features that allow users to generate reports on project progress, team performance, and other key metrics.
7. Security: The system will have robust security features to protect user data and prevent unauthorized access.
8. User interface: The system will have a user-friendly interface that is easy to use and navigate.

## *B. Deliverables*

1. Wireframes: A set of visual representations of the user interface and layout of the project management system.
2. Diagrams: A set of diagrams that describe the project's architecture, data flows, and system components.
3. Development Plans: A detailed plan outlining the project's development roadmap, including milestones, timelines, and project scope.
4. Frontend Code Packages: The frontend codebase of the project management system that includes all the HTML, CSS, JavaScript, and other necessary files.
5. Backend Code Packages: The backend codebase of the project management system that includes all the server-side code, APIs, and database queries.
6. User Guide: A detailed guide that explains how to use the project management system, including its features, functionalities, and user interface.
7. Testing Reports: Reports detailing the results of various tests carried out on the system during the development process.
8. Deployment Guide: A guide that outlines how to deploy the project management system on various platforms

## ***C. Requirements***

Non-functional Requirements:

1. The application should be user-friendly and intuitive to use for users of all technical backgrounds.
2. The application should have a responsive design and be optimized for different screen sizes and devices.
3. The application should have a quick and seamless user experience, with minimal loading times and no noticeable delays or lag.
4. The application should be secure and maintain user data privacy, with proper encryption and protection of sensitive information.
5. The application should be scalable and able to handle a growing number of users and data.
6. The application should be accessible, meeting WCAG(W3.com) guidelines to ensure that users with disabilities can access and use the application.
7. The application should have a high level of availability, with minimal downtime and outages.
8. The application should be easy to maintain and update with new features and functionality.
9. The application should have clear and concise documentation for developers, testers, and users.
10. The application should be compatible with different web browsers and operating systems.

Functional Requirements:

11. The application should allow users to view their scheduled tasks and events in a calendar format.
12. The application should allow users to add new tasks and events to their calendar.
13. The application should allow users to delete and edit existing tasks and events from their calendar.
14. The application should allow users to search for tasks and events based on different criteria.
15. The application should allow users to receive reminders for upcoming tasks and events.
16. The application should allow users to share their calendar with other users.
17. The application should allow users to view their calendar in different formats, such as week, month, and year.
18. The application should allow users to customize their calendar with different colors and themes.
19. The application should allow users to import and export their calendar data to other platforms.
20. The application should allow users to log in and authenticate using Google sign-in.
21. The application should use JSON Web Tokens (JWT) for authentication and authorization.
22. The application should use the Material UI library to design its user interface.

23. The application should use Big Calendar API to display calendar data.
24. The application should use Node.js and SQLite for its backend development.
25. The application should allow users to filter their calendar data based on different categories, such as work, personal, or school.
26. The application should allow users to view a summary of their upcoming tasks and events on the homepage.
27. The application should allow users to receive notifications and alerts for important events or changes to their calendar.
28. The application should allow users to invite others to events and send out email notifications.
29. The application should have a feedback or help section where users can report issues or provide suggestions for improvement.
30. The application should have a contact page with information on how to get in touch with customer support or technical assistance.

# Schedule

## A. Schedule

**PROJECT MANAGEMENT SYSTEM**

**Course** **SWE 6633 SOFTWARE PROJECT PLANNING & MANAGEMENT**

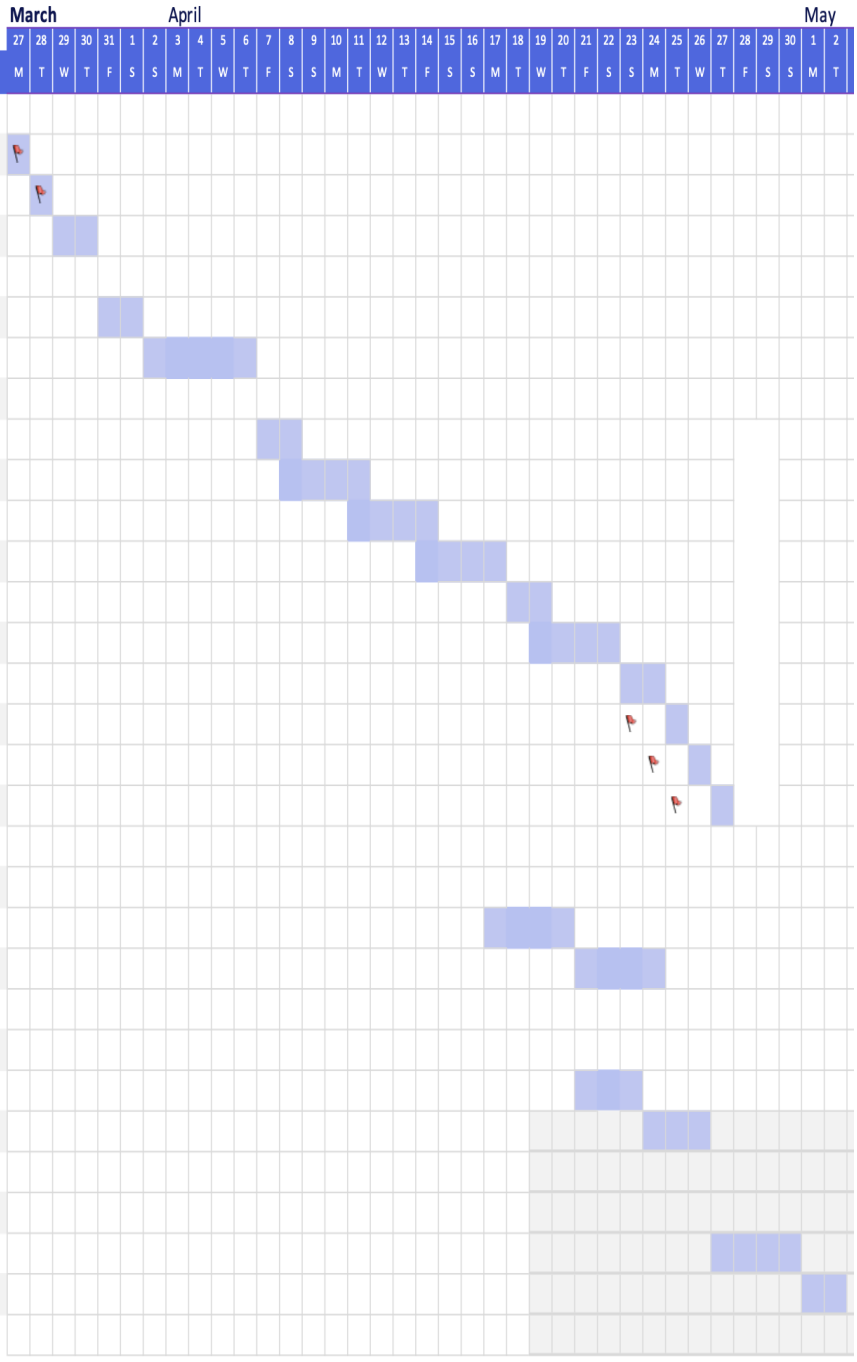
**Group 7 - Spring 2023**

Project start date: 3/27/23

Milestone marker: 1

Scrolling increment: 0

| Milestone description | Assigned to | Progress | Start | Days |
|---|---|---|---|---|
| **Planning** | | | | 4 |
| Update Requirement List | | 25% | 3/27/23 | 1 |
| Design Logical ERD | | | 3/28/23 | 1 |
| Update Resource List | | 100% | 3/29/23 | 2 |
| **UI Design** | | | | 7 |
| Design a component tree that shows the structure of the user | | 60% | 3/31/23 | 2 |
| Design wireframes for each page/component of the UI | | 50% | 4/2/23 | 5 |
| **Front End Development** | | | | 21 |
| Set up the React project with required dependencies | | | 4/7/23 | 2 |
| Use Material UI to build the components in the app | | | 4/9/23 | 3 |
| Integrate Big Calendar API to implement scheduling feature3 | | | 4/12/23 | 3 |
| Implement functionality to display scheduled tasks/events | | | 4/15/23 | 3 |
| Develop form to add new tasks/events | | | 4/18/23 | 2 |
| Implement functionality to delete/update tasks/events | | | 4/20/23 | 3 |
| Develop search functionality | | | 4/23/23 | 2 |
| Test the UI components and features | | | 4/25/23 | 1 |
| Optimize UI for different screen sizes/devices | | | 4/26/23 | 1 |
| Document code and update project documentation | | | 4/27/23 | 1 |
| **Back End Development** | | | | 8 |
| Develop back-end using Node.js and SQLite | | | 4/17/23 | 4 |
| Implement API endpoints and handle database interactions | | | 4/21/23 | 4 |
| **Authentication** | | | | |
| Implement OAuth for Google sign-in | | | 4/21/23 | 3 |
| Use JWT for authentication and authorization | | | 4/24/23 | 3 |
| Testing | | | | |
| Write unitand integration tests | | | 4/27/23 | 4 |
| Perform manual testing | | | 5/1/23 | 2 |



## B. **Summary**

The timeline for the entire project, including all of its phases and tasks, is shown in the Gantt chart above. Planning and development are the two key aspects of the project.

A logical ERD is designed during the planning phase, and the component tree and wireframes are used to create the UI design. The planning phase also involves revising the requirements and resource lists. The duration of this phase is from March 27, 2023, to April 12, 2023.

Testing, front-end development, and back-end development are all part of the development phase. Setting up the React project, creating UI components, connecting APIs, adding functionality for scheduled activities and events, creating a form for adding new tasks and events, and adding search capability are all included in front-end development. April 13, 2023, through April 30, 2023, are the dates for this phase.

Implementing API endpoints, managing database interactions, and constructing the back end using Node.js and SQLite are all included in back-end development. OAuth for Google sign-in and JWT for authentication and authorization will be introduced during this phase, which also covers authentication. The duration of this phase is from April 13, 2023, through April 26, 2023.

Writing unit and integration tests is a last aspect of testing. April 27, 2023 marks the beginning of this period, which concludes on May 1, 2023.

The timeline for each task and step is presented in detail in this chart, enabling efficient project management and tracking of progress.

# Cost Assessment

## A. Assessment

For the cost estimate, the COCOMO model was used. The organic mode was chosen to calculate the total effort needed for project completion due to this project's relatively nominal complexity and its overall flexibility. The basic form to calculate COCOMO is the following: Effort (E) = [3.2 x (size)^1.05 x PROD(f's)], where PROD(f's) is the summation of the cost drivers for the project. Therefore, the calculated effort for this project is from the following function:

Effort = [3.2x(10)^1.05x (1)]
Effort = [3.2x(11.22) x (1)]
Effort = [35.90x1]
Effort = 35.9

| Cost Estimate Information | Value |
|---|---|
| COCOMO Model | Organic |
| Size | 10 |
| PROD(f's) | 1 |
| Effort | 35.9 person-months |

## B. Summary

The model value is Organic due to the project's nominal complexity and overall flexibility. The size of the project was 10, and the PROD(f's) cost driver was 1, resulting in an effort estimate of 35.9 person-months. It has been determined that the cost drivers are of nominal influence on the product's cost estimate. This determination was due to many factors, specifically the overall team size being five with an above average programming ability and time to delivery being over two months. Therefore, the value of PROD(f's) is 1. Because the product is being classified as average, the size variable within the COCOMO function has been set to 10, signifying 10KLOC or 10 thousand lines of code. A percentage of 8 has been added to the cost estimation, which has been calculated by taking the sum of effort 35.9 * 0.08 and adding the resulting value of 2.87 to the sum above 35.9. This results in a cost estimate of 38.77 person months for project completion.

# Resources

**A. People**

a. Ryan O'Connor – Developer
b. Samuel Owoade – Developer
c. Harini Pammi – Developer
d. Cameron Page – Developer
e. Milly Namukasa – Developer

B. **Skills**:

There are several different types of skills that are necessary for this project as there are several different phases. The developers must be able to create and implement design plans. Familiarity with a project management tool like Jira is a bonus. The developers must be familiar with basics of the software development life cycle and preferably agile development principles. Developers should at least be familiar with JavaScript and SQL when working in the development phase of the project.

**C. Development Tools**

- VS Code
- GitHub
- Heroku
- OAuth
- Lighthouse
- Jest

**D. Design Tools:**

- Figma
- Lucidchart
- Material UI

**E. Programming Languages and Frameworks:**

- React
- JavaScript
- SQLite
- HTML
- CSS

**F. Productivity Tools:**

- Microsoft Office (Word, Excel, PowerPoint)
- Jira
- MS Teams

# Process and Methods

## A. Planning

**1. Update the requirements list**

- Conduct stakeholder interviews to gather requirements
- Review existing documentation to ensure accuracy and completeness
- Prioritize and categorize requirements based on importance and feasibility
- Create user stories and use cases to ensure that all requirements are captured

**2. Design a logical ERD**

- Create user stories and use cases to ensure that all requirements are captured
- Identify all entities and their relationships
- Map out the attributes for each entity

## B. Frontend Development

**1. Set up the React project with required dependencies:**

- Install Node.js and create a new React project using create-react-app.
- Install any necessary dependencies, such as react-router-dom for routing.

**2. Use Material UI to build the components in the app:**

- Convert wireframes into Material UI components
- Install Material UI and its dependencies.
- Use Material UI components and styling to create a consistent look and feel throughout the app.

**3. Integrate Big Calendar API to implement the scheduling and calendar features:**

- Read the documentation for the Big Calendar API and understand how to use it.
- Implement the API into the React components that need it.

**4. Implement the functionality to display scheduled tasks and events:**

- Create a data model for tasks and events.
- Use the API to fetch scheduled tasks and events.
- Display the scheduled tasks and events on the calendar.

**5. Develop a form to add new tasks and events:**

- Create a form component to collect information about new tasks and events.
- Use the API to send the new task/event data to the backend.

**6. Implement the functionality to delete and update existing tasks and events:**

- Add a delete button to the task/event components.
- Use the API to delete the task/event from the backend.
- Add an edit button to the task/event components.
- Create a form to edit the task/event data and use the API to update the backend.

**7. Develop a search functionality to search for tasks and events:**

- Create a search bar component.

- Use the API to filter tasks/events based on search criteria.

8. **Test the UI components and features to ensure that they are working as expected:**

- Test each UI component individually.
- Test each UI feature individually.
- Conduct end-to-end testing of the entire app.

9. **Optimize the UI for different screen sizes and devices:**

- Use responsive design techniques to ensure that the app looks good on all screen sizes.
- Test the app on different devices to ensure that it works as expected.

10. **Document the code and update project documentation as needed**:

- Write comments in the code to explain what each section does.
- Create documentation that explains how to use the app.
- Update the documentation as changes are made to the app.

## C. Backend Development Plan

1. **Develop the back-end using Node.js and SQLite.**

- Set up a new Node.js project with required dependencies.
- Configure the project for SQLite database connectivity.
- Write the server-side code using Node.js and SQLite to handle HTTP requests and responses.
- Implement middleware for handling common functionality like logging, error handling, etc.

2. **Implement the API endpoints and handle database interactions.**

- Design and document the API endpoints based on the front-end requirements.
- Write the code to implement each API endpoint using the appropriate HTTP method and response format.
- Write database queries and code to handle data persistence.
- Implement security measures to prevent unauthorized access to API endpoints.
- Write unit tests to ensure that the API endpoints are working as expected.

3. **Implement OAuth for Google sign-in:**

- Research the OAuth 2.0 protocol and Google Sign-In API.
- Register the application with Google and obtain the client ID and secret key.
- Configure the backend to handle the OAuth 2.0 flow and exchange authorization code for access token.
- Store the user profile data and access token in the database.
- Generate a JWT token and return it to the frontend.

**4. Use JWT for authentication and authorization:**

- Implement middleware to decode and verify the JWT token.
- Implement authorization checks for each API endpoint based on the user role and permissions.
- Handle invalid or expired JWT tokens and return appropriate error responses.
- Implement refresh token mechanism to extend the validity of JWT token.
- Encrypt sensitive data in JWT payload.

## D. Testing and Quality Assurance

**Create a testing plan:**

**• Identify the different types of testing to be conducted, such as unit testing, integration testing, system testing, and user acceptance testing.**

**• Develop test cases and test scenarios for each type of testing.**

**• Allocate resources and schedule time for testing activities.**

**Perform unit testing:**

**• Test individual functions and components in isolation to ensure they function correctly.**

**• Use testing frameworks like Jest for frontend and Mocha/Chai for backend to automate unit testing.**

**• Identify and fix any bugs or issues found during unit testing.**

**Perform integration testing:**

**• Test the interaction between different components and modules in the system.**

**• Ensure that the frontend and backend components work together seamlessly.**

**• Identify and fix any bugs or issues found during integration testing.**

**Perform system testing:**

**• Test the entire application as a whole, simulating real-world usage scenarios.**

**• Ensure that the application meets all functional and non-functional requirements.**

**• Identify and fix any bugs or issues found during system testing.**

**Perform user acceptance testing:**

**• Involve end-users in testing the application to validate its usability, functionality, and real-world effectiveness.**

**• Collect feedback from users and make any necessary changes to the application.**
**• Obtain final sign-off from stakeholders to confirm that the application meets their expectations.**
**Perform security testing:**
**• Test the application for vulnerabilities and potential security risks.**
**• Use tools like OWASP ZAP for vulnerability scanning.**
**• Implement any necessary security measures to address identified risks and vulnerabilities.**
**Perform performance testing:**
**• Test the application's performance under various loads and conditions.**
**• Use tools like JMeter or LoadRunner to simulate load on the application.**
**• Optimize the application to improve its performance and scalability.**
**E. Deployment and Maintenance**

**Create a deployment plan:**
**• Identify the target environment for deployment, such as cloud or on-premises servers.**
**• Develop a step-by-step guide for deploying the application to the target environment.**
**• Allocate resources and schedule time for deployment activities.**
**Deploy the application:**
**• Follow the deployment plan to deploy the application to the target environment.**
**• Test the application in the target environment to ensure it functions correctly.**
**• Resolve any issues that arise during deployment.**
**Monitor and maintain the application:**
**• Monitor the application's performance, security, and availability after deployment.**
**• Use monitoring tools like New Relic or Datadog to track application metrics.**
**• Perform regular maintenance tasks, such as updates, backups, and security patches.**
**• Address any issues or bugs that arise after deployment in a timely manner.**
**Provide user support and training:**
**• Offer support and assistance to users as they begin using the application.**

**• Develop training materials and resources to help users effectively use the application.**
**• Collect user feedback and use it to improve the application over time.**
**Continuously improve the application:**
**• Regularly evaluate the application's performance and user satisfaction.**
**• Implement updates and improvements based on user feedback and emerging best practices.**
**• Maintain open communication with stakeholders to ensure the application continues to meet their needs and expectations.**

# Risks

## A. Risk Analysis

**1. Complexity:** This comes in the form of various dependencies, constraints, and variations being considered. The system entails all the requirements to be accurately captured and the results to be accurately produced. As the dependencies, constraints and variations rise, so will the increase in the complexity, and the challenges of accurately capturing them.

| **RISK ANALYSIS REPORT OF COMPLEXITY** | | |
|---|---|---|
| **RISK ID:** 1 | PROBABILITY: High | IMPACT: High |
| **Description:** The challenge of capturing various dependencies, constraints, and variations accurately, leading to increased system complexity and difficulty in producing accurate results. | | |
| **Refinement/Context:** Accuracy of capturing dependencies, constraints, and variations.<br>1. Difficulty in identifying all necessary dependencies and constraints.<br>2. Complexity of accurately capturing variations in requirements.<br>3. Challenges in properly documenting dependencies and constraints. | | |
| **Mitigation/Monitoring:**<br>1. Conduct research and testing to ensure the chosen technology stack is suitable for the project.<br>2. Set up regular communication channels with stakeholders to obtain timely feedback.<br>3. Use Agile methodologies to accommodate changes in project requirements. | | |
| **Management/Contingency Plan:** Engage in continuous testing and feedback with stakeholders to ensure accuracy and adjust as needed. | | |
| **Current Status:** Ongoing | | |

**2. Technical Difficulties:** The technology used in the program's development may be too outdated or use platforms that lack adequate developer support. The platforms used may lack the necessary security updates to curb against the attempts of hacking and other security breaches. Technical issues that may be encountered during the development process may cause delays in the project schedule, increase the costs of development, and lead to poor performance by the system.

| RISK ANALYSIS REPORT OF TECHNICAL DIFFICULTIES | | |
|---|---|---|
| **RISK ID:** 2 | PROBABILITY: Medium | IMPACT: High |
| **Description:** Possibility of encountering technical difficulties during development due to outdated technology or lack of adequate support or security updates, leading to delays, increased costs, and poor system performance. | | |
| **Refinement/Context:** Outdated technology, lack of developer support, and inadequate security updates.<br>1. Using technology stacks with inadequate support and limited resources.<br>2. Failure to perform regular security updates.<br>3. Technical issues encountered during the development process. | | |
| **Mitigation/Monitoring:** | | |

| |
|---|
| 1. Conduct regular security updates and use reliable technology stacks.<br>2. Have technical support on standby to address any issues that may arise. |
| **Management/Contingency Plan:** Have contingency plans in place to address any technical difficulties that may arise during the development process. |
| **Current Status:** Ongoing |

**3. User Adoption Risk:** introducing the system to end users may come with challenges such as the system being difficult to use, the failure to meet the user needs, and lack of adequate training for the end users.

| **RISK ANALYSIS REPORT OF USER ADOPTION RISK** | | |
|---|---|---|
| **RISK ID:** 3 | PROBABILITY: Medium | IMPACT: Medium |
| **Description:** Challenge of introducing the system to end-users who may find it difficult to use or may have specific needs that the system fails to meet. Lack of adequate training may impact system adoption. | | |
| **Refinement/Context:** Difficulty of use, failure to meet user needs, lack of adequate training.<br>1. End-users may find the system difficult to use.<br>2. The system may not meet the specific needs of end-users.<br>3. Inadequate training for end-users. | | |
| **Mitigation/Monitoring:**<br>1. Conduct usability testing to ensure ease of use and provide adequate training for end users.<br>2. Engage end users in the development process to incorporate their feedback. | | |
| **Management/Contingency Plan:** Provide ongoing support and training for end users. | | |
| **Current Status:** Ongoing | | |

**4. Changes in Requirements:** as the program is being developed, changes in the requirements will occur. Failure to implement and recognize these changes can be costly financially and in time and scheduling.

| **RISK ANALYSIS REPORT OF CHANGES IN REQUIREMENTS** | | |
|---|---|---|
| **RISK ID:** 4 | PROBABILITY: High | IMPACT: Medium |
| **Description:** Likelihood of changes in project requirements during development, which may have financial implications and impact project timeline if not recognized and implemented. | | |
| **Refinement/Context:** Changes in project requirements during the development process.<br>1. Failure to recognize and implement changes in requirements.<br>2. Changes in project scope leading to increased complexity.<br>3. Changes in project requirements causing delays and additional costs. | | |
| **Mitigation/Monitoring:**<br>1. Use Agile methodologies to accommodate changes in project requirements.<br>2. Regularly communicate with stakeholders to obtain timely feedback. | | |
| **Management/Contingency Plan:** Develop contingency plans to address any potential delays or financial impacts due to changes in requirements. | | |
| **Current Status:** Ongoing | | |

## B. Assumptions

1.Communication and Collaboration: It is assumed that the effective communication channel that will be used is Teams and collaboration tools are established to ensure smooth and efficient collaboration among team members.
2.Technical Knowledge and Skills: It is expected that every team member will have the requisite technical expertise to contribute to the project.
3.Project Scope and Requirements: It is believed that all team members have a thorough understanding of the project's goals and specifications.
4.Realistic Schedule and Budget: It is assumed that the project schedule and budget are realistic and feasible, considering the available resources and the complexity of the project.5.Risk Management: It is considered that potential risks and challenges are identified and addressed proactively, and that any issues or conflicts that arise are addressed and resolved collaboratively.

# Product Attributes

## A. Overview of Attributes

**1. Design**:

- Usability: How easy it is for users to understand the layout and functionality of the interface.
- Functionality: How well the wireframes represent the intended features and workflows of the project management system.
- Accessibility: The extent to which the wireframes consider accessibility needs for users with disabilities.

**2. Diagrams:**

- Usability: How easy it is for users to understand the system architecture and data flows.
- Functionality: How well the diagrams represent the intended components and functionality of the project management system.
- Integration: The extent to which the diagrams consider integration with third-party tools and services.

**3. Development Plans:**

- Functionality: The scope and detail of the development plan, including milestones, timelines, and project scope.
- Supportability: The availability of resources to help developers troubleshoot issues and learn how to implement the system effectively.

**4. Frontend Code Packages:**

- Usability: How easy it is for users to navigate and use the user interface of the project management system.
- Functionality: The range and effectiveness of the features provided by the frontend code packages.
- Performance: The speed and responsiveness of the frontend code packages in handling various tasks and requests from multiple users.
- Accessibility: The extent to which the frontend code packages consider accessibility needs for users with disabilities.

**5. Backend Code Packages:**

- Functionality: The range and effectiveness of the server-side features provided by the backend code packages, such as APIs and database queries.
- Performance: The speed and responsiveness of the backend code packages in handling various tasks and requests from multiple users.
- Security: The level of protection provided by the backend code packages for sensitive data and user information.

- Reliability: The ability of the backend code packages to function consistently and without errors over time.
- Scalability: The ability of the backend code packages to handle increased usage and data over time without compromising performance or functionality.
- Supportability: The availability of resources to help developers troubleshoot issues and learn how to implement the backend code packages effectively.

**6. User Guide:**

- Usability: How easy it is for users to understand and use the user guide to learn how to use the project management system.
- Functionality: The range and effectiveness of the features explained in the user guide.
- Accessibility: The extent to which the user guide considers accessibility needs for users with disabilities.
- Supportability: The availability of resources to help users troubleshoot issues and learn how to use the system effectively.

**7. Testing Reports:**

- Functionality: The scope and detail of the testing reports, including the types of tests carried out and their results.
- Reliability: The accuracy and consistency of the testing reports in identifying issues and verifying functionality.

8**. Deployment Guide:**

- Functionality: The scope and detail of the deployment guide, including the supported platforms and configurations.
- Supportability: The availability of resources to help users troubleshoot issues and learn how to deploy the system effectively.