SWE 4663 - Software Project Management

Term Project: Comprehensive Plan

Title Page

Project Title: Project Management System
Group Number: Group 6
Group Leader: Kahmin Keller
Group Members:

- Kahmin Keller - (A/B/C: Contribution Level)
- Kevin Shyavong - (Project Description and Cost Assessment)
- RJ Straiton - (Testing and Validation Plan & Technical Approach and Tools)
- Patrick Cox - (Scheduling Details & Gantt Chart, Task Delegation)
- Jared Louissaint - (A/B/C: Contribution Level)

---

1. Introduction

The purpose of this project is to develop a **Project Management System** that enables users to track various software development projects efficiently. The system will provide input for functional and non-functional requirements, allow monitoring of project efforts across different phases, and generate reports on expended hours.

This **Comprehensive Plan** expands upon the initial **Quick Plan** by adding detailed implementation strategies, risk mitigation plans, testing approaches, and clearly defined team roles.

---

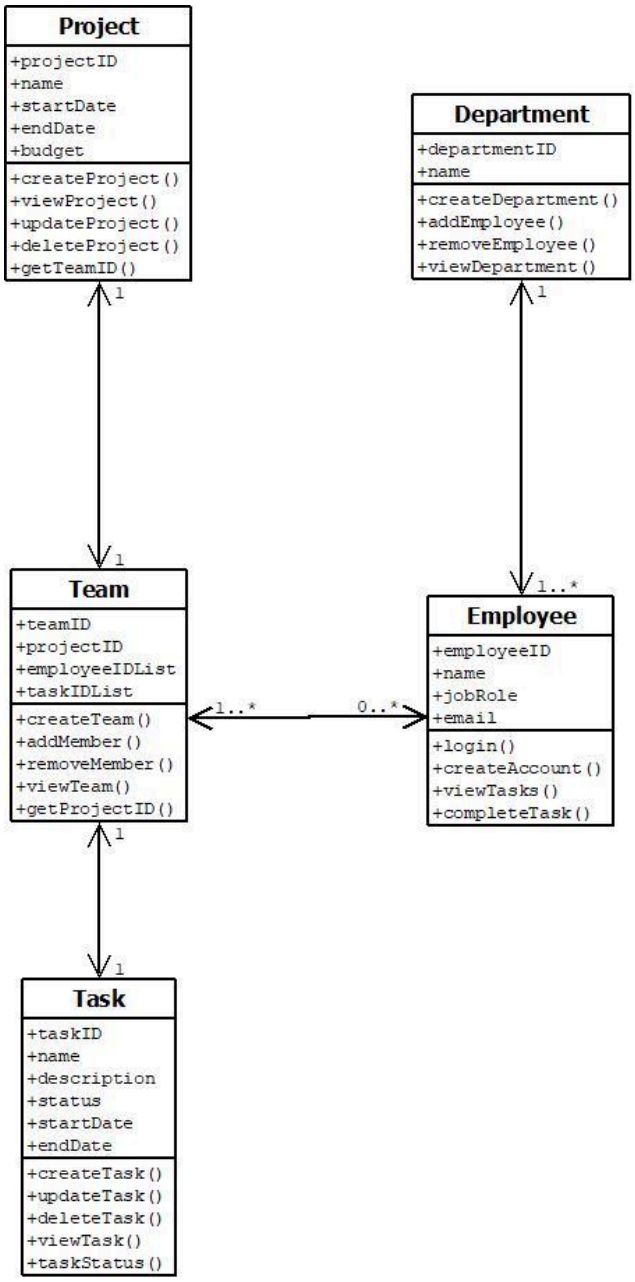
2. Project Description

Functionalities and Features

The Project Management System will include:

1. General Project Information Management

- Ability to create and manage multiple projects.
- Enter project descriptions, owners, and team members.
- Track and update project risks.
- Assign priority levels to projects and tasks.



-

2. Project Requirements Management

- Add and update functional and non-functional requirements.
- Categorize requirements under different project phases.
- Maintain a version history of requirement updates.
- Dynamic control of tasks and team composition.

3. Effort Monitoring and Tracking

- **Time Logging System**:

    - Users can record hours spent on different tasks like planning, coding, and testing.
    - Track hours daily or weekly.
    - Automatic alerts for excessive workload distribution.
- **Data Aggregation and Reporting**:
    - View total hours spent per project phase.
    - Export reports as CSV or PDF.
- **Task Breakdown for Better Tracking**:
    - Break down large tasks into smaller steps.
    - Track time for each step to understand progress.
    - Identify delays and adjust workload as needed.

4. User Roles and Access Control

- Define roles like **Project Manager, Team Member, and Administrator**.
- Limit access to certain features based on roles.
- Implement role-based security measures for sensitive project data.

5. Reporting and Visualization

- Dashboard with project progress overview.
- Charts and graphs to display workload and time spent.
- Real-time updates on project phases.

Project Goals

- Ensure **efficient project tracking**
- Improve **collaboration** among project teams
- Enhance **transparency** in project effort estimation
- Support **data-driven decision-making**

Major Deliverables

- **Fully functional software system** with core project management features.
- **User Documentation** including a manual for onboarding new users.
- **Technical Documentation** detailing software architecture, API endpoints, and database schemas.
- **Test Reports** to validate system reliability and performance.
- **Final Presentation** demonstrating the developed system and its capabilities.

---

3. Cost Assessment

Estimated Work Hours by Task

Task	Estimated Hours
Requirements Analysis	25 hrs
System Design	50 hrs
Development (Coding)	100 hrs
Testing & Validation	50 hrs
Project Management	25 hrs

Rationale

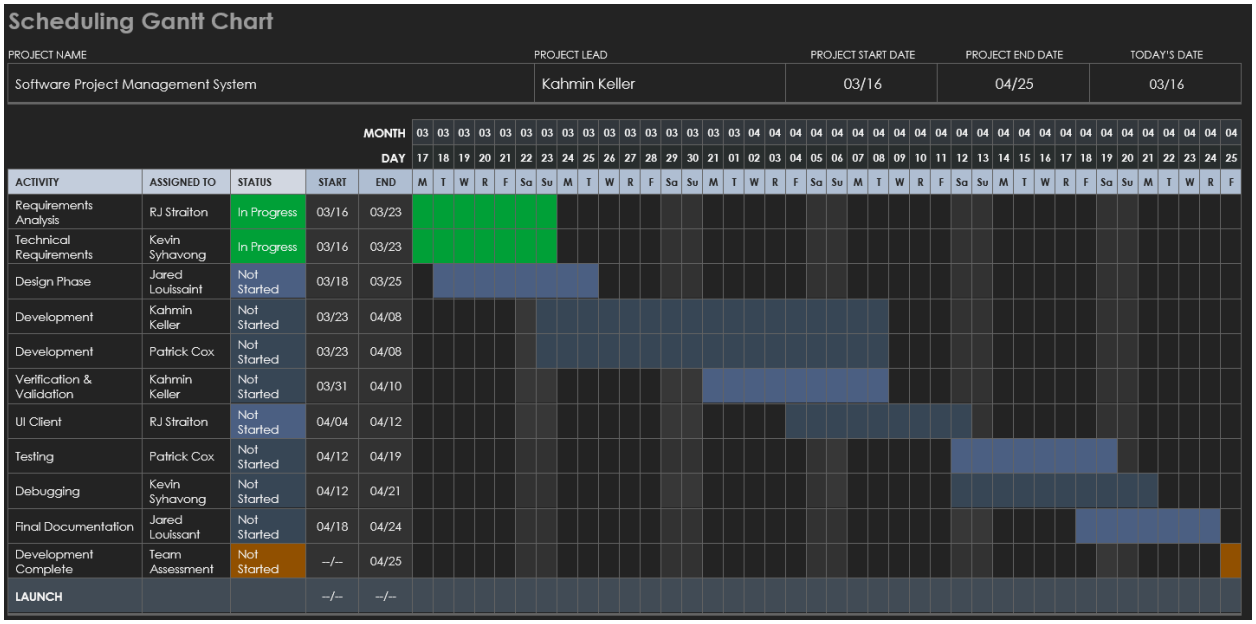
These estimates are based on standard industry practices and our collective team experience in previous coursework and project development.

---

4. Project Schedule

The project will follow the **Agile methodology**, using the following high-level timeline:

Phase	Duration	Description
Requirements Analysis	1 week	Define system requirements
Design Phase	2 weeks	Create system architecture and UI design
Development	4 weeks	Implement core functionalities
Testing & Debugging	2 weeks	Conduct system testing and bug fixes
Final Documentation	1 week	Prepare user manuals and reports



5. Risk Analysis and Mitigation

Top Risks and Mitigation Strategies

Risk	Impact	Mitigation Strategy
Scope Creep	High	Establish clear scope boundaries early. Regularly review with the team.
Team Coordination Issues	Medium	Set up structured communication channels (e.g., weekly stand-ups).
Time Constraints	High	Prioritize tasks, assign buffer time for unexpected issues.
Technical Challenges	High	Conduct feasibility studies before implementation.

---

6. Technical Approach and Tools

- **Programming Language:** C#,
- **Frameworks & Libraries:** .Net 8.0, NUnit, Windows Form App
- **Database:** PostgreSQL
- **Version Control:** GitHub

We will be developing this project using C# to create a desktop application that will enable users to track software development projects. We have decided on C# as we believe it would be best suited for developing an application. Many of us at this point in our academic years have likely gained some experience with C# as well, meaning it will take less time to learn. NUnit will be used for all unit tests for the application's logic.

---

7. Team Roles and Responsibilities

Team Member	Role	Responsibilities
Kahmin Keller	Project Manager	Oversees project execution, manages deadlines.
Kevin Syhavong	Lead Developer	Manages core system architecture and coding.
RJ Straiton	UI/UX Designer	Designs user interface and experience.
Patrick Cox	QA Engineer	Tests system functionality and performance.
Jared Louissaint	Documentation & Research	Prepares technical and user documentation, conducts feasibility studies.

---

8. Testing and Validation Plan

- **Code Review:** The project leader will review the code closely during development.
    - Thorough inspection ensuring all code entries align with a requirement
    - Guarantee that project architecture is maintained for modularity and low coupling.
    - The code review will subsequently lend towards the code testing.
- **Unit Testing:** Validate individual components and functions.
    - Unit tests will be implemented using NUnit.
    - Unit tests will follow these three parts:
        - 1. The name of the method being tested
        - 2. The conditions for the test
        - 3. The expected behavior of the method
        - Ex: Multiply_NumberbyZero_ReturnsZero()
- **Integration Testing:** Ensure modules work together as expected.
    - Integration unit tests will be implemented using NUnit.
    - Integration unit tests will follow the same naming conventions.
        - Ex: CalculatorUI_Title_IsCalculator()
- **User Testing:** Gather feedback from potential users and improve usability.
    - User testing will be conducted with the remote user testing method.
    - Users will be asked to complete a series of tasks in the program and then complete a short questionnaire about their experience.

---

9. Summary

This **Comprehensive Plan** provides a detailed roadmap for developing the **Project Management System**. It outlines the functional scope, cost estimates, risk mitigation strategies, technical tools, and implementation phases. The plan ensures a **structured, measurable, and goal-oriented approach** to project completion.

---

10. Bibliography/References

https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices

https://adamfard.com/blog/conduct-user-testing

This document serves as a foundation for executing and managing the project efficiently.