**Title:** Comprehensive Plan

**Group #**: 1

**Group Leader:** Alex Husseini

*Can confirm that each member did their share along with their contributions*

**Group Members:**

1. Alex Husseini

*Contribution: Resources, Assumptions, Summary*

2. Cole Brumbelow

*Contribution: Assumptions, Summary*

3. DJ Berzas

*Contribution: Schedule, Costs*

4. Aiman Nawaz

*Contribution: Problem and Requirements, Product/Project Description, Product/Project*

*Attributes, Risks*

5. Ethan Wise

*Contribution: Process and Methods*

**Table of Contents**

[Introduction 4](#_Toc196412599)

[Problem and Requirements 4](#_Toc196412600)

[Product/Project Description 4](#_Toc196412601)

[Project Scope 4](#_Toc196412602)

[Functional Requirements 5](#_Toc196412603)

[Nonfunctional Requirements 5](#_Toc196412604)

[Product/Project Attributes 5](#_Toc196412605)

[System Design and Architecture 6](#_Toc196412606)

[Database Schema 6](#_Toc196412607)

[UI Design 7](#_Toc196412608)

[Implementation Summary 7](#_Toc196412609)

[5.1 General Information Module 7](#_Toc196412610)

[5.2 Requirements Module 7](#_Toc196412611)

[5.3 Effort Tracking Module 8](#_Toc196412612)

[5.4 Risk Management Module 8](#_Toc196412613)

[5.5 Export Module 8](#_Toc196412614)

[Schedule 9](#_Toc196412615)

[Costs 12](#_Toc196412616)

[Resources 15](#_Toc196412617)

[Summary 17](#_Toc196412618)

[References 18](#_Toc196412619)

# Introduction

The software development lifecycle is often hindered by poor coordination, lack of oversight, and disorganized requirements tracking. This project addresses such challenges by developing a centralized Project Management System (PMS) tailored for software development teams. The PMS enables real-time monitoring of project metrics, including effort distribution across development phases, requirement tracking, and risk management.

# Problem and Requirements

The PMS resolves issues around requirement disorganization, unclear visibility into project status, and inefficient communication. It allows teams to track each phase of the SDLC (Software Development Life Cycle), from planning through testing. Each project in the system includes team members, requirements, risks, and tracked effort across analysis, design, coding, testing, and management.

# Product/Project Description

## Project Scope

The goal of this project is to develop a Project Management System (PMS) designed to facilitate the tracking and management of multiple software projects. The system will serve as a centralized platform to input and manage project-related data, including team members, requirements, effort allocation, risk status, and progress reporting. The tool aims to improve productivity, resource visibility, risk mitigation, and overall project effectiveness.

## Functional Requirements

1. The system shall allow users to create, edit, and delete project records, including project name, owner, team members, and description.
2. The system shall enable users to input and update functional and non-functional requirements for each project.
3. The system shall provide a feature to log effort spent on project tasks, categorized by phase (e.g., requirements analysis, design, coding, testing, project management).
4. The system shall generate real-time reports displaying total hours logged per requirement and project phase.
5. The system shall maintain a risk register where users can add, update, and monitor risks, including their severity, likelihood, and mitigation strategies.
6. The system shall support user authentication, ensuring that only authorized users can access and modify project data.
7. The system shall allow users to export project reports in PDF and CSV formats.
8. The system shall send automated notifications to project managers when risk levels exceed predefined thresholds.

## Nonfunctional Requirements

1. The system shall provide a response time of no more than 2 seconds for any user request under normal load conditions.
2. The system shall support at least 50 concurrent users without performance degradation.
3. The system shall be accessible via a web-based interface, ensuring compatibility across different operating systems and devices.
4. The system shall store all project data in a secure database, ensuring data integrity and preventing unauthorized modifications.
5. The system shall comply with industry security standards, including data encryption for sensitive information.
6. The system shall maintain 99% uptime, ensuring minimal downtime for users.
7. The system shall provide an intuitive and user-friendly interface, requiring no more than one hour of training for new users.
8. The system shall be designed for scalability, allowing future expansion to support larger projects and additional features without requiring major architectural changes.

# Product/Project Attributes

The Project Management System (PMS) and its associated deliverables will be evaluated based on specific attributes that align with the project’s goals, ensuring measurable outcomes and facilitating quality assurance. These attributes define the effectiveness, usability, and reliability of both the software product and its supporting documentation.

# System Design and Architecture

The PMS is structured as a multi-tabbed Tkinter interface. Each tab encapsulates a core functionality module, ensuring modularity and separation of concerns. Backend logic is organized into dedicated classes and functions that interface with the PostgreSQL database using psycopg2.

## Database Schema

1. users: Stores user credentials and roles (id, username, password\_hash, role, created\_at)
2. projects: Holds project metadata (id, project\_name, owner, project\_description, project\_scope, target\_users, technology\_stack, platform)
3. team\_members: Links members to projects with roles and skill levels (id, project\_id, name, role, responsibilities, skill\_level)
4. requirements: Contains functional/non-functional requirements per project (id, project\_id, requirement\_name, description, requirement\_type, status, created\_at, updated\_at)
5. effort\_tracking: Logs work effort by requirement and date (id, project\_id, requirement\_id, date, requirements\_analysis, designing, coding, testing, project\_management)
6. risks: Maintains the risk register with statuses (id, project\_id, risk\_name, risk\_description, risk\_status)

All foreign keys reference their respective primary keys, enforcing relational integrity. Timestamp columns track creation and updates. Data types ensure strong typing (e.g., NUMERIC for effort, TEXT for descriptions).

## UI Design

The graphical user interface was built entirely using the Tkinter library and organized using the ttk.Notebook widget to create a tabbed layout. Each tab corresponds to a distinct functional module such as Projects, Requirements, Risks, Effort Tracking, and Exports. Within each tab, layout is handled by grid managers and frames to group elements logically. Dropdowns (ttk.Combobox) are used for project selection and risk severity filtering. Tables are implemented with ttk.Treeview widgets to present data such as team members, requirements, and logged efforts in a structured, scrollable format. Data entry and editing use pop-up forms built using Toplevel windows, which encapsulate input fields and ensure modal behavior. The design emphasizes usability and consistent styling using ttk elements, with dynamic population of widgets based on backend data.

# Implementation Summary

## 5.1 General Information Module

This module manages the creation and storage of core project data. Upon selecting “New Project,” the user is prompted via a pop-up form (Tkinter Toplevel window) to enter the project name, owner, description, scope, target users, technology stack, and platform. The system validates these entries and then inserts them into the projects table via parameterized SQL queries to avoid SQL injection. The main window dynamically updates to reflect the new project, and other modules automatically load its associated data.

## 5.2 Requirements Module

Requirements are managed per project, and each is stored as either functional or non-functional. Users select a project, then use a form to add or edit a requirement’s name, description, type, and status. A timestamp is generated using CURRENT\_TIMESTAMP. These entries are stored in the requirements table with a foreign key linking to the appropriate project. The interface categorizes entries using Treeview columns, supporting inline edits through double-click bindings that populate editable fields in modal windows.

## 5.3 Effort Tracking Module

This is one of the most dynamic parts of the system. Users select a requirement and a date, then input hours spent on each of the five phases: requirements analysis, designing, coding, testing, and project management. The form prevents negative numbers and duplicate entries for the same date-requirement pair. Data is inserted using parameterized SQL into the effort\_tracking table. Users can review past entries in a Treeview list, and a backend SQL aggregation (SUM with GROUP BY) is used to calculate and display total hours by requirement and phase. This feature provides real-time visibility into team workload and progress distribution.

## 5.4 Risk Management Module

Risks are logged with name, description, and status (low, medium, high). The user can create, update, or delete risks linked to a specific project. Internally, each risk is stored in the risks table, and modifications are reflected in the GUI using Treeview row selections. Update and delete operations are performed by selecting a risk and launching the form in edit mode. Changes are saved using an UPDATE SQL query. Risk levels are visualized via color-coded Treeview rows to highlight high-risk entries for the project manager.

## 5.5 Export Module

The export functionality enables data to be written into CSV files for archival or reporting purposes. Each type of export—Projects, Requirements, Risks, or Effort Tracking—is handled by a separate button. Upon triggering an export, the system queries the database using optimized JOINs to combine necessary details (e.g., project name with each requirement). The resulting data is written to CSV using Python’s csv.writer, and the user selects the file destination via a Tkinter file dialog. Error handling ensures the user is notified of any export failures, such as empty result sets or write permissions issues.

All modules are encapsulated in Python classes to promote reusability and maintainability. Each database operation is abstracted behind a layer of helper functions, making the system adaptable to schema changes or backend upgrades. The GUI design follows a unified layout approach for consistency, making the learning curve minimal for new users.

### 

* Peer-to-peer learning sessions will be implemented to address challenges to make sure of high-quality coding practices.
* Workshops on agile development and project management methodologies will be organized throughout to make sure collaboration is smooth and no problems arise.

**Additional Costs and Logistics**

* Server costs: $20 a month potentially for cloud hosting
* Development and testing equipment: team members will be using personal computers, making additional hardware unnecessary.
* Training costs: Some workshops and courses might have some costs, but free resources will be a main priority.
* Collaboration expenses: Google Workspace and Slack will be used to allow communication between team members.

# Summary

The Project Management System (PMS) is a comprehensive solution designed to streamline project management tasks and ensure efficient project tracking. Built using Python with a Tkinter graphical user interface (GUI), the system is easy to use and intuitive for all project team members. The PostgreSQL backend ensures secure, scalable, and efficient data management, making it suitable for both small and large teams. The system centralizes all critical project data, providing project managers with a unified platform to oversee and manage their teams and tasks effectively.

Core features of the system include secure user authentication, allowing only authorized personnel to access project information. With different user roles assigned to varying levels of access, the system ensures that sensitive data is protected while providing the appropriate stakeholders with necessary permissions. The system also includes modules for managing project information, team structures, and tracking risks. The project information module serves as the central repository for project goals, timelines, and documentation, while the team management module helps assign roles, track involvement, and monitor progress. Risk management ensures that potential risks are identified, documented, and mitigated throughout the project lifecycle.

Additional functionalities include a requirements management module that tracks project requirements to ensure all goals are met, and an effort tracking module that logs the time and resources spent on each task. The export feature allows for the generation of reports and project data exports in formats like CSV or PDF, making it easy to share progress with clients and stakeholders. By bringing these elements together, the Project Management System offers a comprehensive, easy-to-use solution that supports project success, enhances team collaboration, and ensures transparency throughout the project lifecycle.

### 

# References

*Tsui, Frank F. Managing Systems and IT Projects. Sudbury, Mass., Jones & Bartlett Learning, 2011.*

*Python Software Foundation. Python Documentation. Python.org,* [*https://docs.python.org*](https://docs.python.org)*.*

*PostgreSQL Global Development Group. PostgreSQL Documentation. PostgreSQL.org,* [*https://www.postgresql.org/docs/*](https://www.postgresql.org/docs/)*.*

*TkDocs. Tkinter Reference. TkDocs.com,* [*https://tkdocs.com*](https://tkdocs.com)*.*