Comprehensive Plan – Deliverable 2

Group 3

Shams Hasan – [A]

Ameer Salameh – [A]

Ashlynn Sutton – [A]

Harry Nguyen – [A]

Julissa Urbiola – [A]

Table of Contents

Introduction

This document outlines the development of a **Project Management System** designed to help teams track and manage software projects more effectively. The system will allow users to input and monitor key project details, requirements, and the time spent on various tasks like requirements analysis, design, coding, testing, and project management. By providing a clear overview of project progress and effort allocation, this tool aims to improve efficiency and transparency in project workflows.

This report is organized into sections: a **Table of Contents**, **an Introduction, a** detailed **Body** with subsections (using tables, graphs, and charts where applicable), a **Summary**, and a **Bibliography/References**. Additionally, the document includes the **Group Title**, **Group Number**, **Group Leader's Name**, and **Group Members' Names** with their respective contributions, clearly indicating whether each member (A) did their share, (B) did less than their share, or (C) did nothing.

Our team is organized with specific roles, including a **Project Manager**, **Risk Analyzers**, **Developers**, **Designers**, and **Testers**, to ensure smooth collaboration and efficient progress. We will deliver a prototype focusing on the core features above. The following sections explain our plan and progress.

Objectives

Group Three's mission is to develop the Project Management System, an efficient tool used to help software development teams plan and organize their projects. Objectives of this system consist of the following: create a user-friendly interface in which users can seamlessly add general project information, enable requirements and goals management, implement logging hours for effort tracking, and offer visual representations for easy understanding.

To ensure that these objectives are met within the time constraints of project deadlines, Group Three will utilize planning documents (both the quick and comprehensive plans) and a Trello board to track the development progress of the Project Management System. Tasks will be assigned to all group members on the Trello board so that everyone is contributing, and this will also aid in mapping out the deliverables as well as the project status and progress.

To make certain that the Project Management System is developed correctly, a GitHub repository will be used to share code. This will involve pull requests so that code is reviewed by everyone before merging it into the main branch and will therefore help reduce errors or bugs. In addition, all group members will communicate each week via GroupMe and/or the Teams channel to facilitate collaboration.

Major External Deliverables

Our team will produce the following external deliverables:

1. **Quick Plan Document** – A high-level overview of project objectives, requirements, and estimated effort.
2. **Comprehensive Plan Document** – A detailed report covering requirements, methodology, cost estimation, and risks.
3. **Project Management System (PMS) Source Code** – Fully documented, hosted in a GitHub repository.
4. **Functional Project Management System** – A working system with project tracking, requirement management, and effort logging.
5. **Testing & Validation Report** – A document detailing test cases, results, and issue resolutions.

Schedule

Task ID	Work Breakdown Structure	Planned Start	Planned Finish	Workload Planned	Workload Finished	Progress
1	- Create GitHub repo for organization and tracking - Brainstorm what features to implement and record under deliverable section	2/7/25	2/23/25	-Set up our GitHub repository README.md and edit roles under TEAM_ROLES.md file -Complete assigned sections for the first deliverable.	-30 min team meeting -GitHub Repo set up and updated with roles and appropriate documents	100%

	- Work on sections for the 1st deliverable				- Completed all assigned sections for the first deliverable.	
2	- Add in missing sections and content from last deliverable - Add in sections required for Deliverable 2 - Create project through IntelliJ	3/9/25	3/16/25	Work on: -TOC, Schedule, Risk/Assumptions page(s) -Functional + non-functional requirements -Deliverables page + content for D1 & D2 -Rationales on: 1) process + schedule (Agile methodology) 2) the cost estimate -Project Description of Deliverables and Project Attributes and Measurements -Process and Methods -Detailed cost overview (resources, equipment, etc.)	Completed adding in missing sections from deliverable one. Completed expanding each section of plan for deliverable two. Completed new sections for deliverable two.	100%
3	- Begin back-end development - Create database - Determine where to deploy project	3/17/25	3/23/25	Work on: - Create database schema - Set up database - Create tables - Set up connection to database		
4	- Continue back-end development - Begin front-end development	3/24/25	4/6/25	Work on: - Coding back-end - Begin testing - Creating front-end prototype - Begin coding front-end pages		

5	- Finalize coding - Finish testing - Deploy system	4/7/25	4/21/25	Work on: - Finish coding front-end screens - Finish APIs and other back-end logic - Finish testing system functionality		

Process & Schedule Rationale

Software Development Methodology

We are utilizing the **Agile methodology** due to its flexibility, iterative development approach, and ability to incorporate feedback at each phase. Agile allows us to **prioritize core functionalities first** while refining additional features based on team input and testing feedback.

Timeline Justification

The schedule was determined based on:

1. **Task complexity** – More time was allocated to development and testing, as these phases are the most resource-intensive.
2. **Parallel Tasking** – Some tasks, such as documentation and project coordination, run concurrently to maximize efficiency.
3. **Buffer Time** – Additional time was included to accommodate potential delays and ensure project completion within the deadline.

Ensuring Feasibility

- **Workload is evenly distributed** among team members.
- **Frequent progress reviews** ensure we stay on track.
- **Agile sprints** allow continuous improvements without delaying deliverables.

Project Description

- The system includes three main features:

1. **General Project Information:** Users can add a high-level project description, the project manager's name, team members (which can be updated as needed), and a list of risks with their status.
2. **Project Requirements:** Users can input and manage functional and non-functional requirements to ensure the project aligns with its goals.
3. **Effort Tracking:** Users can log hours spent on different tasks daily or weekly and view the total hours expended for each requirement and phase. This helps teams identify bottlenecks and allocate resources effectively.

Functional and Non-Functional Requirements

Functional Requirements:

1. **Project Details Management:** The system shall allow users to create, update, and view project details, including project name, description, project manager, and team members.
2. **Requirements Management:** The system shall enable users to add, edit, and delete functional and non-functional requirements, with the ability to categorize them (e.g., priority, status).
3. **Effort Tracking:** The system shall allow users to log hours spent on tasks (e.g., requirements analysis, design, coding, testing) and display the total hours expended per task or phase.
4. **Risk Management:** The system shall allow users to add, update, and track project risks, including risk description, impact, likelihood, and mitigation status.
5. **Progress Reporting:** The system shall provide a simple dashboard or summary view to display key project metrics, such as total hours logged, risks, and requirements status.

Non-Functional Requirements:

1. **Usability:** The system shall have a simple, intuitive user interface that requires minimal training to use.
2. **Data Management:** The system shall store project data in a structured format (e.g., JSON file, simple database) that can be easily backed up and restored.
3. **Accessibility:** The system shall be accessible via a web-based interface or a local application, depending on the chosen implementation.
4. **Performance:** The system shall handle up to 10 concurrent users during demonstrations without significant performance degradation.

5. **Development Constraints:** a) The system shall be developed using a programming language and tools already familiar to the team. b) The system shall not require advanced frameworks or libraries unless explicitly approved by the instructor.

Tech Stack

Our **Project Management System (PMS)** will be developed using the following technologies:

Frontend

- **HTML, CSS, JavaScript** – For building an intuitive and responsive user interface.
- **Bootstrap/Tailwind CSS** – To enhance styling and maintain a modern design.
- **React.js or Vue.js** (Optional) – For creating a dynamic, component-based frontend if time allows.

Backend

- **Java (Spring Boot)** – Handles business logic, user authentication, and API development.
- **IntelliJ IDEA** – The main development environment for backend services.
- **RESTful APIs** – Used for communication between frontend and backend.

Database & Data Storage

- **PostgreSQL/MySQL** – A relational database to store project details, users, and task logs.
- **Firebase or MongoDB** (Optional) – Could be used for real-time data updates or NoSQL needs.
- **Docker** (Optional) – To containerize and streamline deployment.

Version Control & Collaboration

- **Git & GitHub** – Source control for managing and tracking code changes.
- **GitHub Actions** – To automate CI/CD workflows and testing.

Project Management & Testing

- **Trello or Jira** – Task management and sprint tracking.
- **Postman** – API testing to ensure smooth backend functionality.
- **JUnit & Selenium** – Unit and end-to-end testing tools for quality assurance.

Deployment & Hosting

- **Heroku or AWS EC2** – Hosting for the web application.
- **Nginx/Apache** – Web server to manage request handling efficiently.

Rough Cost Estimate

- Total Estimate Effort: ~200 person-hours
    - Requirements Gathering – 15 hours
    - System Design – 25 hours
    - Development (Frontend and Backend) – 90 hours
    - Testing and debugging – 35 hours
    - Project Management (Team Meetings and Team Coordination) – 20 hours
    - Deployment – 15 hours

Complexity Level

This project is classified as **moderate complexity** because it requires full-stack development, user authentication, data storage, and visualization features. It does not involve highly complex algorithms but does require structured project management, testing, and coordination.

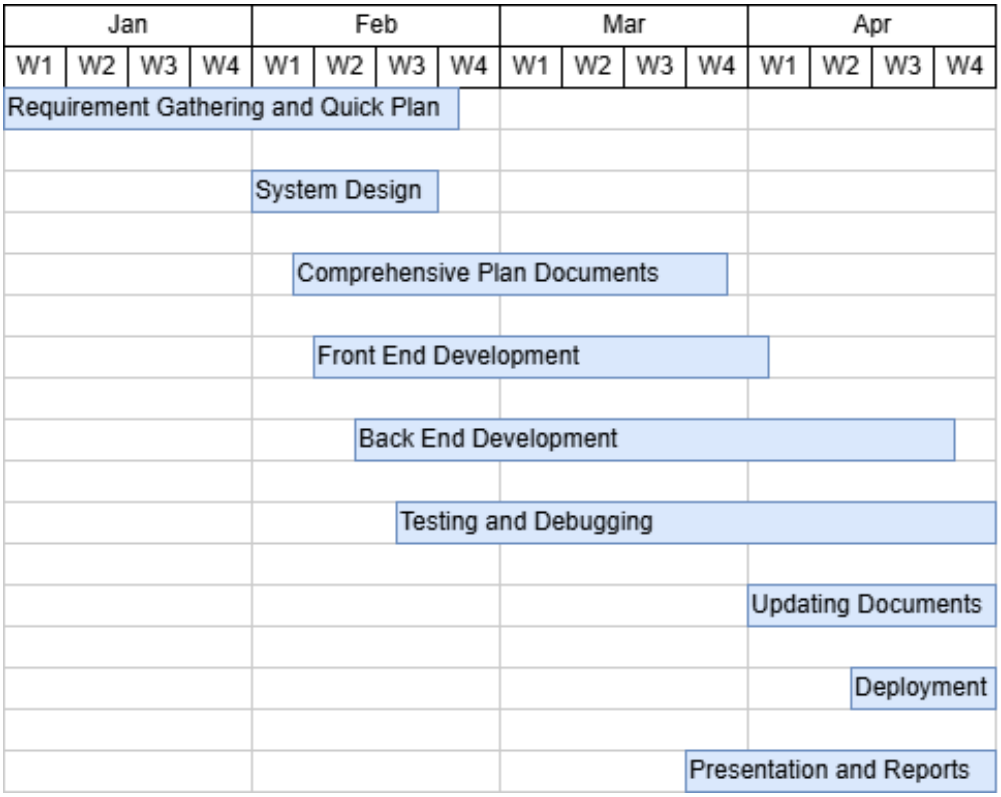
Scope & Depth

The project will deliver a **functional Project Management System (PMS)** with core features such as project tracking, requirement management, and effort logging. While scalability and advanced integrations (e.g., third-party APIs) are not the focus, the system will be **fully functional** within its defined scope.

Person-Hour Justification

The 200-hour estimate is based on prior experience with similar development projects. Each phase has been assigned time based on expected workload, factoring in buffer time for unforeseen issues like debugging, feature refinements, and additional testing.

High-Level Estimate



Core Features

The Project Management System (PMS) will be developed in phases to allow for the prioritization of features. Each phase will be rolled out based on the importance of the system's core functionality. This means that the highest priority features will form the foundation of the PMS.

Our primary features will focus on the implementation of project management capabilities. This will include an input system for project details, allowing for the storage of the project description, project owner information, and user management data. This phase will also include the core features of inputting and tracking basic project requirements.

Once the primary features are set up, the secondary features will focus on the tracking capabilities. This will include a system that allows time to be tracked and generates reports based on the work completed under a project. This phase will also include risk management features allowing users to record and monitor project risks throughout the life cycle.

The final phase will focus on improving the user experience by implementing a user interface with a dashboard to view the total hours spent on the project. The dashboard will display breakdowns of the time spent and work completed in requirements analysis, design, coding, testing, and project management activities.

Throughout the development of each phase, testing will be done to ensure high-quality software is produced along with strong system stability and performance.

Risks and Assumptions

I. Risks

- Scope Creep
  - Risk Level: Medium
  - Description: New feature that expands the project initial scope was added.
  - Mitigation Strategy: Clearly define requirements and follow a strict change management process.
- Team Availability Issues
  - Risk Level: High
  - Description: Team members may not be available due to other responsibilities and projects
  - Mitigation Strategy: Assign backup roles, track progress, and ensure proper workload distribution
- Inaccurate Effort Estimation
  - Risk Level: High
  - Description: The time needed for development, testing, and debugging may be underestimated.
  - Mitigation Strategy: Use buffer time in the schedule and monitor progress regularly

II. Assumptions

- All team members will complete their assigned tasks on time and communicate effectively.
- Project scope will remain stable throughout the project. The core features outlined in the plan will not drastically change mid-project.
- The estimated timeline is realistic, and the project can be accomplished by the end of this course.

Change Management Process

The Change Management Process below outlines how our team will deal with any modifications to the project scope, requirements, or deliverables. Having a centralized process will ensure that all changes are evaluated and documented before implementation so that they do not interfere with our progress or the quality of the project.

1. Change Request Identification
    - Changes may come up based on the following:
        - Instructor feedback
        - Team member suggestions
        - Technical constraints found during implementation

All changes will be documented along with the source to maintain visibility and traceability throughout the project.

2. Change Request Documentation
    - Changes will be formally documented with the following:
        - Change request ID
        - Description
        - Name of the requestor
        - Impact of change
        - Reason for change
        - Present State and Target State
3. Impact Assessment

    Before the change is approved, the team will go through an impact analysis to ensure that the change will be beneficial and to determine the effects on the project. The analysis will look at how the change will impact the timeline, how many additional person-hours it might take, if and how it will affect the other features, and lastly if any risks are involved. This analysis will help all team members understand the change before we move to approving.

4. Approval Process
    - Change request will be reviewed during the team meetings
    - Impact analysis will be reviewed
    - Team will vote
        - Minor changes require majority vote
        - Major changes will require a unanimous vote
5. Implementation
    - Change will be assigned a team member and a priority status
    - Schedule will be updated if needed

- Documentation will be updated

6. Verification and Validation
    - Change will be tested to ensure proper functionality
    - Members will review and update documents as necessary

Summary

The Project Management System aims to streamline the tracking and management of software projects by allowing users to monitor project requirements, team members, risks, and effort distribution efficiently. Our Quick Plan outlines the key functionalities, estimated person-hours, high-level schedule, and risk assessment to ensure smooth project execution.

The project will be developed over an estimated 200 person-hours, covering requirements gathering, system design, front-end and back-end development, testing, deployment, and project management. A Gantt Chart has been created to provide a high-level timeline.

This Quick Plan serves as the foundation for our Comprehensive Plan, where we will go into more detail about our project's technical details, resource planning, and risk management strategies. This system is designed to increase project transparency and improve efficiency by providing a structured approach to tracking development progress, risk assessment, and team contributions.

The core functionalities of the system include:

• Project Details Management: Allowing users to input and modify key project information, team members, and project risks.

• Requirement Tracking: Ensuring functional and non-functional requirements are documented and updated as needed.

• Effort Logging & Visualization: Providing time tracking capabilities to measure team contributions across various project phases.

• Risk Monitoring: Allowing project managers to log potential risks, assign statuses, and track resolution efforts.

Bibliography/References

SWE 4663 Course Materials and Project Requirements Document

Team Member Contributions

Week 1 (2/2/25-2/9/25):

- Ashlynn (2/6/25):

  Created blank Word document for the quick plan and shared it with all members.

- Shams (2/7/25):

  Created doc; formatted title page, table of contents, schedule, and team member contribution pages (30 mins)

- Ashlynn (2/7/25):

  Created Teams channel and invited all members.

- Ashlynn, Harry, and Julissa (2/7/25):

  Attended a Teams meeting for 30 minutes to discuss the project plan, requirements, and deliverables.

Week 2 (2/9/25-2/16/25):

- Shams (2/16/25):

  Updated Intro and Schedule section of doc, created + formatted team roles file under the GitHub repository (40 mins)

- Ameer Salameh:

Logging contributions to GitHub, Trello, and our word shared documentation.

Week 3 (2/17/25-2/24/25):

- Ashlynn (2/19/25):

  Wrote the objectives section of the quick plan and picked role in GitHub.

- Ameer (2/20/25):

  Wrote the summary section with some risks edits.

- Shams, Ashlynn, Harry, Julissa (2/20/25):

  Attended meeting to discuss team roles and quick plan section assignments (30 mins)

- Harry (2/20/25):

  Wrote risks and assumption section. Edit, and reformat quick-plan document

- Julissa (2/20/25):

  Wrote the core features section.

- Ashlynn (2/22/25):

  Added contributions section to quick plan and added tasks to Trello board.

- Ameer (2/23/25):

  Edits to line spacing as well as grammar and fonts.

Week 4 (3/9/25-3/16/25):

- Shams (3/9/25):

Created comprehensive plan doc; added in schedule, functional/non-functional requirements, and risk/assumptions pages

- Ashlynn (3/11/25):

Made some changes to the objectives section. Fixed font style and size for some sections to make document consistent.

- Ashlynn (3/13/25):

Finished objectives section. Added major external deliverables section. Added tech stack section.

- Ameer Salameh (3/13/25):

Added and updated the Process & Schedule Rationale section. Expanded the Cost Estimation Rationale section. Refined and updated the Tech Stack section with additional details.
Revised the Major External Deliverables section for clarity.

- Ashlynn (3/15/25):

Added table of contents.

- Julissa (3/16/25):

Added the change management process

- Ashlynn (3/16/25)

Added in three tasks to schedule chart.