Software Project Management Plan (SPMP)

Based on IEEE 1058 - Adapted for Student Projects

Project Title: Python code analyser

Members:

Zahra Bader Hayyan 2240006156

Hadeel abduallah alqhtani 2240003327

Noor Almuhsen 2240003505

Yomna Al moslem 2240003445

Alzahraa alabbad 2240006089

Eman Alnajem 2240002468

Wedad Mohammed AL-hussaini 2240002853

Advisor: Rahmah alzhrani

Version:1.0

Date: 12/10/2025

Table of Contents

1. Project Overview

1.1 Purpose, Scope, and Objectives

- Our project is mainly designed to provide a web-based tool that analyses Python source code and generates a structured report.. After working on several projects, we pointed out how crucial it is to evaluate the state and quality of the code. And this motivated us to develop a solution that simplifies code and supports developers in improving their work efficiently.

scope

The system will provide the following capabilities

a. Reading and checking Python code files.
b. Showing the number of lines, functions, and classes.
c. Handling simple code errors without crashing.
d. Giving a clear and easy-to-read report.

The following items are excluded from the project scope :

a. Detection of logical errors
b. Code execution
c. Practical performance optimization
d. Gui development
e. Integration and external services
f. Code generation

Project Objectives

a. To analyze Python source code and extract key information such as the number of lines, classes, functions, and imported libraries.
b. To generate a structured and detailed report that summarizes the code's structure and quality.
c. To detect syntax errors and provide feedback to help improve the correctness of the code.
d. To support easier collaboration and promote effective code reuse among development teams.

1.2 Assumptions, Constraints, and Risks

The following assumptions have been made during the planning and development of the project :

a. All team members have laptops or PCs

b. The project team will have access to all required development and communication tools (e.g., VS Code, GitHub, browser)
c. The user will upload a valid Python code file (py).
d. Both the web server and the hosting environment will support the latest versions of Python
e. The internet connection will be stable throughout the project development and testing duration

The following constraints have been identified and must be considered during the project lifecycle:

a. The project must be completed within 8 weeks.
b. The system will only support scripts written in Python.
c. The tool is web-based no offline version.
d. The group size is limited to the current members
e. Only open source tools are allowed

This section identifies the major risks associated with our project:

a. Missing the deadline
b. Delay in project completion due to underestimated time at development or testing
c. User misunderstood of how to upload the file or read reports
d. Fail to handle syntax errors due to limitations of syntax analysis tools

The lack of technical skills

1.3 Project Deliverables

Deliverable	Description	Expected Submission Date	Format
Project Proposal	Initial document describing the project idea, objectives, and feasibility.	18/09/2025	Word / PDF
***SPMP* (*Software Project Management Plan*)**	Defines how the project will be managed, including schedule, resources, and risks.	12/10/2025	Word / PDF
***SRS* (*Software Requirements Specification*)**	Lists all functional and non-functional requirements for the Python Code Analyzer.	6/11/2025	Word / PDF
***SDD* (*Software Design Document*)**	Describes the system architecture, data flow, and component design.	15/10/2025	Word / PDF
***STS* (*Software Test Specification*)**	Defines the test cases, expected results, and testing strategy.	4/12/2025	Word / PDF

Source Code	The complete Python code for the analyzer tool, including all modules and documentation.	18/12/2025	GitHub Repository / ZIP File
Final Report	A comprehensive report that summarizes all project stages and outcomes.	18/12/2025	Word / PDF
Presentation	A PowerPoint presentation summarizing project goals, implementation, and results.	18/12/2025	PowerPoint (PPTX)[i]

Table1: Project Deliverables

1.4 Schedule Summary

The project will be completed over approximately nine weeks, from late October to mid-December 2025. It consists of five major phases: Planning & Requirements, Design, Implementation, Testing, and Final Delivery. Each phase produces a key deliverable, ensuring consistent progress and timely completion.
A summary of the schedule is shown below, while the detailed Gantt chart is provided in Appendix B.

Phase	Duration	Major Deliverable	Deadline
Planning & Requirements Analysis	02-05 Nov 2025	Software Requirements Specification (SRS)	6 Nov 2025
Design Phase	07-19 Nov 2025	Software Design Document (SDD)	20 Nov 2025
Implementation Phase	21 – 30 Nov 2025	Working Prototype	30 Nov 2025
Testing Phase	02- 03 Dec 2025	System Test Specification (STS)	04 Dec 2025
Final Delivery & Presentation	14-17 Dec 2025	Final Report & Presentation	18 Dec 2025

Table2: summary of the schedule

Major Milestones:
- SRS approved by Week 11
- Design review completed by Week 13
- Prototype demonstration by Week 11
- Final delivery and presentation by Week 14

A detailed Gantt chart of these phases is provided in Appendix B.

1.5 References

1. IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans.
2. IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

3. Sommerville, I. Software Engineering, 10th Edition, Pearson Education, 2016.
4. Pressman, R. S. Software Engineering: A Practitioner's Approach, 8th Edition, McGraw-Hill, 2015.
5. CSC 305 Course Materials (Dr. Rahma Ahmed, Term 1, 2025/2026).

1.6 Definitions and Acronyms

This section defines the main terms and acronyms used throughout this document. Additional terms are provided in Appendix A (Glossary).

Term	Definition
SPMP	*Software Project Management Plan* – defines how the project is managed.
SRS	*Software Requirements Specification* – lists all functional and non-functional requirements.
SDD	*Software Design Document* – describes the system's architecture and design.
STS	*Software Test Specification* – defines test cases and expected results.
WBS	*Work Breakdown Structure* – a hierarchical breakdown of project tasks.
Gantt Chart	A visual schedule showing the timeline of project activities.
Milestone	A major project event used to measure progress (e.g., design completion).
QA	*Quality Assurance* – activities ensuring the software meets defined standards.
IDE	*Integrated Development Environment* (e.g., PyCharm, VS Code).
GitHub	A platform used for version control and collaboration.
Work Product	Any tangible item produced during software development (e.g., documents, code, reports).
Baseline	A formally reviewed and accepted work product that can only be changed through configuration management.
Analyzer	The main component of the system that reads and examines Python source code.

Report Generator	The module responsible for summarizing and displaying the analysis results in a structured format.
Syntax Error Handling	The process of detecting and managing invalid Python syntax without crashing the system.
Code Metrics	Quantitative measures extracted from the Python file, such as number of lines, functions, and classes.
Upload Module	The part of the system that allows users to upload .py files for analysis through the web interface.
Web Interface	The front-end of the system where users interact with the analyzer tool using a browser.

Table3: the main terms and acronyms used throughout this document

2. Project Organization

2.1 External Interfaces

The project team interacts with the following external entities:

• **Supervisor (Dr. Rahma Ahmed):** Provides feedback, guidance, and approval of project deliverables.

• **End Users (Students, kids, and beginner programmers):** Will use the Python Code Analyzer to understand and improve their code quality.

These external entities ensure that the project aligns with its academic goals and provides value to its intended users..

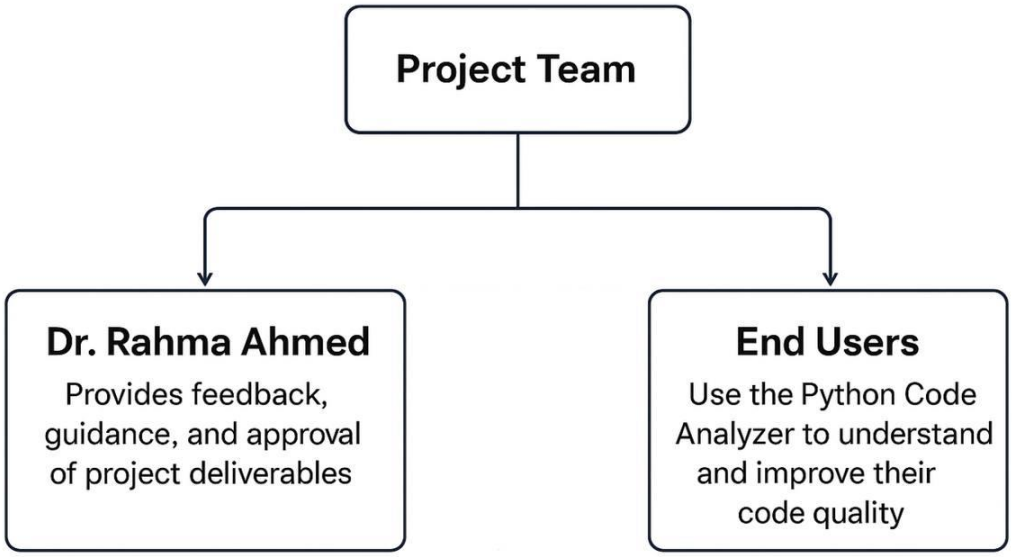

Figure 1: Project External Interfaces Diagram

2.2 Internal Structure

The project team consists of seven students with defined roles and responsibilities. The internal structure follows a collaborative model led by the project manager. Communication and collaboration are maintained through GitHub and WhatsApp to ensure smooth coordination and progress tracking.

Team Members and Roles:

Eman Alnajem – Project Manager / Backend Developer: Oversees the team's progress, manages GitHub repositories, communicates with the supervisor, and ensures all tasks are delivered on schedule.

Noor Almuhsen – Frontend Developer / Tester: Designs and implements the front-end interface and conducts functional and usability testing.

Yomna Almoslem – Frontend Developer: Develops UI components and collaborates on integration between front-end and back-end.

Wedad Alhussaini – Backend Developer: Implements core backend logic and assists in data processing and report generation.

Hadeel Alqahtani – Documentation Manager: Prepares technical documentation (SRS, SDS, final report) and manages version updates.

Zahra Hayyan – UI/UX Designer: Designs and refines the visual layout, ensuring accessibility and usability standards.

Alzahraa Alabbad – Configuration & Support Manager:
Handles version control, supports integration testing, and maintains configuration documentation.

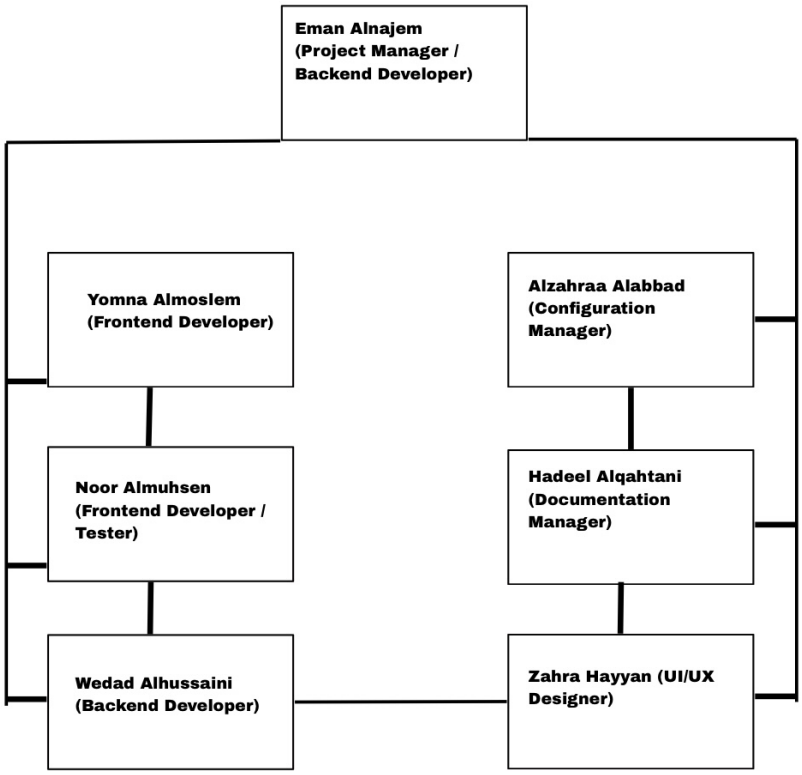

Figure 2: Internal team organizational structure

2.3 Roles and Responsibilities

Table 4: Work Breakdown Structure (WBS)

Team member	Role	Key Responsibilities
Eman Alnajem	Project Manager / Backend Dev	Team coordination, scheduling, communication with supervisor
Noor Almuhsen	Frontend Dev / Tester	Front-end design, Ul testing, quality verification
Yomna Almoslem	Frontend Dev / Tester	Interface design and integration
Wedad Alhussaini	Backend Developer	Implements backend functionality
Hadeel Alqahtani	Documentation Lead	Writes and manages all project documents
Zahra Bader Hayyan	UI/UX Designer	Visual design and usability improvements
Alzahraa alabbad	Config. Manager	Version control and support

3. Managerial Process Plans

3.1 Estimates & Staffing

The overall duration of the project is **approximately nine weeks**, starting from **early October to mid-December 2025**, ending with the final submission on **14 December 2025**.The team consists of seven members distributed across different responsibilities: project management, development, testing, documentation, design, and configuration management. The project follows a **plan-driven (Waterfall-based)** development approach, where each phase is completed sequentially before moving to the next, ensuring clear progress tracking and alignment with project goals.

Project Phase	Estimated Duration	Team Allocation	Description
Planning & Requirements Analysis phase	23 Oct – 6 Nov	2 members (Project Manager and Documentation Manager) (with input from all team members)	Define system goals and user needs and document functional and non-functional requirements.

Design phase	7 – 20 Nov	3 members (Designer and Developers)	Design system architecture and interface
Implementation Phase	21 – 30 Nov	3 members (Developers and Configuration Manager) (with collaboration from all roles)	Develop and integrate system components, connecting backend logic with the web interface.
Testing Phase	2 – 11 Dec	Tester/Quality Assurance	Verify system functionality and stability.
Final Delivery & Presentation	12-18 Dec	All Team member	Finalize the system, ensure stability, and prepare the final presentation and delivery.

Table5: Summary of project phases and responsibilities

3.2 Work Plan

The work plan outlines the **main activities and subtasks** of the project using a **Work Breakdown Structure (WBS)**.
Each activity includes the necessary resources, estimated duration, main deliverables, acceptance criteria, and its relationship with other tasks.
This structured breakdown helps in identifying dependencies, estimating workload, and ensuring effective project tracking.

Work Breakdown Structure (WBS)

Activity ID	Work Activity	Necessary Resources	Estimated Duration	Deliverables / Products	Acceptance Criteria	Predecessor / Successor
1.1	Requirement Gathering	Laptops, internet access, GitHub	2 days	Initial list of system goals and user needs	Requirements identified and approved by all members	— / Leads to 1.2
1.2	Requirement Analysis	Document tools, and team discussions	4 days	Documented functional and non-functional requirements	Requirements are clear, complete, and feasible	1.1 / Leads to 1.3
1.3	Requirement Validation	Team review session and approval by leader	3 days	Finalized requirements document	Requirements Confirmed by leader and team	1.2 / Leads to 2.1
2.1	Interface Design	Laptops, IDEs,	5 days	Create draft layout for	Design is clear and	1.3 / Leads to 2.2

		HTML/CSS tools		input and results display	beginner-friendly	
2.2	Architectural Design	IDEs, GitHub, Lucidchart	3 days	Architecture diagram showing backend, frontend	Architecture reviewed and approved by backend developers	2.1 / Leads to 2.3
2.3	Design Review & Refinement	Team meeting, GitHub	3 days	Finalized design document	Design aligns with requirements	2.2 / Leads to 3.1
3.1	Backend Development	Python, PyCharm, GitHub	4 days	Python modules	Code runs correctly and outputs expected results	2.3 / Leads to 3.2
3.2	Frontend Integration	HTML/CSS, VS Code, GitHub	3 days	web interface displaying results	Data displayed accurately through web interface	3.1 / Leads to 3.3
3.3	Internal Testing & Debugging	Development environment, test data	2 days	Stable and functional integrated version	All major errors fixed, basic functionality confirmed	3.2 / Leads to 4.1
4.1	Test Planning & Preparation	Python, test scripts	3 days	Defined test cases and prepared test data	Test plan covers all features and use cases	3.3 / Leads to 4.2
4.2	System Testing	Test environment, sample Python files	3 days	Executed test cases with documented results	All tests pass successfully; no major defects remain	4.1 / Leads to 4.3
4.3	Test Review & Fixes	Development tools, bug tracker	2 days	Updated stable version after bug fixing	Verified functionality after corrections	4.2 / Leads to 5.1
5.1	Final Review & Documentation	Word / PDF tools, GitHub repository	1 week	Completed documentation package and final build	Documents are complete, consistent, and error-free	4.3 / Leads to 5.2
5.2	Presentation & Delivery	Project slides, demo files	1 week	Final presentation of the Python Code Analyzer	Successful demo of main features	5.1 / End of Project

Table6: work breakdown structure

3.3 Project Tracking Plan

• We will keep track of our project to make sure we finish on time.

The team leader will check what we did every day and compare it with our plan.

At the end of each week, we will meet to talk about what is done and what we still need to finish.

If we want to change something in the plan, we will tell the team leader first before doing it.

We will write simple notes or reports about the progress to know where we are in the project.

We will check our progress by:

- How many tasks are done.
- How many problems or errors we fixed.
- If we are working on time with the schedule.

3.4 Risk Management Plan

This plan is to help us find and manage any problems that might happen in the project.

We will check the risks during our meetings and update them if something changes.

If a new risk appears, we will talk about it and find a way to fix it.

Risk	Likelihood	Impact	Mitigation
Some members don't have enough Python experience	Moderate	Tolerable	Help and teach each other, and review the code together
Delay in work because of weak organization	High	Serious	Make a clear plan and follow it every week
Data might be lost while testing	Low	Catastrophic	Save backup copies often
Library or tool problems	Low	Tolerable	Test the tools before we use them

Weak communication between team members	Low	Tolerable	Have weekly meetings and share updates
Too many errors in the code	Moderate	Serious	Check and test the code before submitting
Running out of time before finishing everything	Moderate	Serious	Focus on main parts first and leave extra for later

Table7: Risk Management Plan

3.5 Closeout Plan

At the end of the project, the team ensures that the Python Code Analyzer tool functions correctly and meets all project requirements.
Any minor issues are identified and fixed before closing the project.
After completing all tasks, the final version of the tool, along with the source code and documentation, is submitted to the course instructor.
Finally, the team holds a review meeting to discuss what went well, what challenges were faced, and how future projects can be improved.

4. Technical Process Plans

4.1 Process Model

The project will follow the plan-driven development model. All requirements, structure, and functions of the web-based Python analysis tool will be defined before implementation. The tool will generate an HTML report with information such as total lines, classes, and functions, handling syntax errors and each stage will follow the planned steps with testing after completion.

We chose the plan-driven model because it provides a clear roadmap, reduces risks, and ensures stable results. This structured approach fits the limited time of a semester project and helps deliver a complete, well-tested system.

4.2 Methods, Tools, and Techniques

The project will be implemented using Python 3.x as the main programming language. Several libraries will support the analysis process: ast and inspect for parsing code and extracting structural details, os and pathlib for handling files and directories, jinja2 for generating structured HTML reports, and pylint for linting and detecting unused imports or style issues. Additional libraries such as radon may be used to provide complexity and maintainability metrics.

Since the tool is web-based, a simple frontend interface will be created using HTML and CSS (generated through Python frameworks if needed) to allow users to upload their Python files, interact with the tool online, and view the generated analysis report in a browser.

The tool is also designed to handle simple syntax errors without crashing. Syntax error handling will be implemented using Python's built-in try–except structure to catch parsing errors, and the compile() function will be used to detect syntax issues before analysis begins.

For development, the team will use Visual Studio Code and PyCharm as IDEs. GitHub will serve as the version control system for collaboration and tracking changes. UML diagrams and design models will be prepared using Figma and Lucidchart. Testing will be performed with both unittest and pytest to ensure the accuracy and reliability of the tool.

4.3 Infrastructure

The project will be developed on personal laptops with stable internet access. The software environment includes Python 3.x, IDEs such as Visual Studio Code and PyCharm, and GitHub for version control.

Since the tool is web-based and generates HTML reports, it will be hosted on a lightweight web server and accessed through a standard web browser, which will be required to view the analysis results.

4.4 Product Acceptance

The product will be accepted if it:

- Successfully analyzes Python code and extracts metrics such as total lines, classes, functions, and imported libraries.
- Generates a structured HTML report that can be accessed through a web browser.
- Handles syntax errors gracefully by detecting them before analysis and reporting them clearly in the HTML report without interrupting execution.
- Meets all functional and non-functional requirements listed in the SRS.

The final approval will be given by the course instructor after demonstrating the tool's functionality and web interface.

5. Supporting Process Plans

5.1 Documentation Plan

All required documents (Proposal, SPMP, SRS, SDD, STS, and Final Report) will be prepared collaboratively, led by Hadeel Alqahtani, and reviewed by Eman Alnajem (Project Manager) before submission.

All documents will be stored and versioned in the GitHub repository.

5.2 Quality Assurance

The team will follow basic coding standards for Python (PEP8). Peer code reviews will be conducted by team members.

Unit tests using unittest and pytest will ensure software reliability and quality.

5.3 Configuration Management

GitHub will be used for version control. The project will maintain two main branches: main for stable versions and dev for ongoing work.

All commits will include clear messages and changes will be reviewed by the project manager before merging.

5.4 Problem Resolution

Bugs and issues will be reported through GitHub Issues.

The team will discuss and assign each issue to a responsible member, and fixes will be verified during weekly meetings. Unresolved conflicts will be escalated to the Project Manager for a decision.

6. Additional Plans (Optional)

• Training Plan:

Team members enhanced their skills in Python libraries (AST, Pylint, Jinja2) through online tutorials and peer learning.

• Security Plan:

The project does not handle sensitive data. All files are stored securely in GitHub with restricted access to team members only.

• Maintenance Plan:

After project submission, the team plans to fix any reported issues and may add new analysis features in future versions

7. Appendices

A. Glossary

Term	Definition
Analyzer	The main component of the system that reads and examines Python source code.
Baseline	A formally reviewed and accepted work product that can only be changed through configuration management.
Code Metrics	Quantitative measures extracted from the Python file, such as number of lines, functions, and classes.
Gantt Chart	A visual schedule showing the timeline of project activities.
GitHub	A platform used for version control and collaboration.
IDE	Integrated Development Environment (e.g., PyCharm, VS Code).
Milestone	A major project event used to measure progress (e.g., design completion).

QA	Quality Assurance - activities ensuring the software meets defined standards.
Report Generator	The module responsible for summarizing and displaying the analysis results in a structured format.
SDD	Software Design Document - describes the system's architecture and design.
SPMP	Software Project Management Plan - defines how the project is managed.
SRS	Software Requirements Specification - lists all functional and non-functional requirements.
STS	Software Test Specification - defines test cases and expected results.
Syntax Error Handling	The process of detecting and managing invalid Python syntax without crashing the system.
Upload Module	The part of the system that allows users to upload .py files for analysis through the web interface.
WBS	Work Breakdown Structure - a hierarchical breakdown of project work tasks.
Web Interface	The front-end of the system where users interact with the analyzer tool using a browser.
Work Product	Any tangible item produced during software development (e.g., documents, code, reports).

Table8: the Glossary

B. Gantt chart

Task ID	Task Name	Start Date	End Date	Duration (in Days)	Predecessor
1.1	Requirement Gathering	23-Oct-25	24-Oct-25	2	
1.2	Requirement Analysis	25-Oct-25	28-Oct-25	4	1.1
1.3	Requirement Validation	29-Oct-25	31-Oct-25	3	1.2
2.1	Interface Design	01-Nov-25	05-Nov-25	5	1.3
2.2	Architectural Design	06-Nov-25	08-Nov-25	3	2.1
2.3	Design Review & Refinement	09-Nov-25	11-Nov-25	3	2.2
3.1	Backend Development	12-Nov-25	15-Nov-25	4	2.3
3.2	Frontend Integration	16-Nov-25	18-Nov-25	3	3.1
3.3	Internal Testing & Debugging	19-Nov-25	20-Nov-25	2	3.2
4.1	Test Planning & Preparation	21-Nov-25	23-Nov-25	3	3.3
4.2	System Testing	24-Nov-25	26-Nov-25	3	4.1
4.3	Test Review & Fixes	27-Nov-25	28-Nov-25	2	4.2
5.1	Final Review & Documentatio	29-Nov-25	03-Dec-25	5	4.3
5.2	Presentation & Delivery	04-Dec-25	08-Dec-25	5	5.1

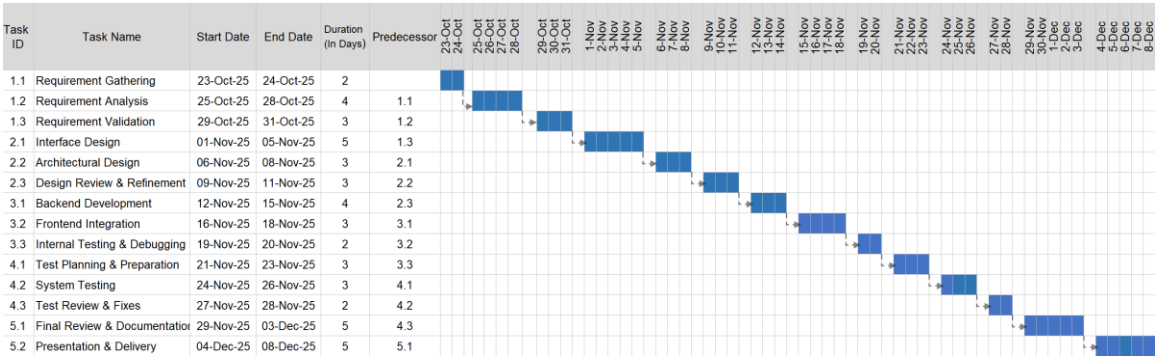

Table9: Gantt chart

C. WBS diagram

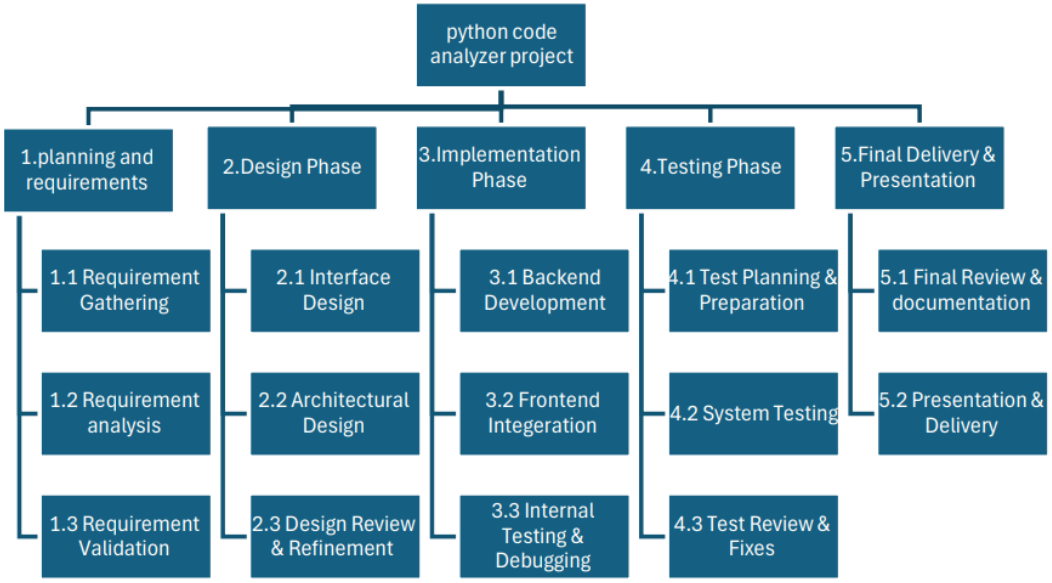

Figure 3: WBS diagram

Evaluation

To be completed by the instructor or supervisor.

3. Project SPMP (18 points / 3 marks) – Week 8

Section	Excellent (Full Points)	Good (75%)	Fair (50%)	Poor (25% or below)	Points
1. Project Overview (3 pts)	Clear purpose, well-defined scope, objectives measurable, risks realistic, deliverables & schedule complete.	Purpose/scope adequate, some objectives unclear, limited risks, minor gaps in deliverables/schedule.	Vague or incomplete scope, objectives weak, little risk identification.	Missing or irrelevant.	/3
2. Project Organization (2 pts)	Well-structured team org, clear roles/responsibilities, external interfaces defined.	Team roles mostly clear, some vagueness in structure or interfaces.	Roles incomplete or unclear, poor organization.	Missing or irrelevant.	/2

Section	Excellent (Full Points)	Good (75%)	Fair (50%)	Poor (25% or below)	Points
3. Managerial Process Plans (5 pts)	Estimates realistic, WBS & schedule clear, tracking plan defined, risks detailed with mitigation, closeout plan solid.	Most elements present but some weak (e.g., vague WBS or shallow risk analysis).	Several parts incomplete or poorly justified.	Very minimal or missing.	/5
4. Technical Process Plans (3 pts)	Process model justified, tools/methods listed, infrastructure described, acceptance criteria realistic.	Adequate but some parts generic or weak justification.	Vague process model, missing tools/infrastructure details.	Very poor or absent.	/3
5. Supporting Process Plans (3 pts)	Documentation plan complete, QA standards clear, version control explained, bug tracking method defined.	Most parts covered but not in detail.	Minimal coverage (e.g., only documentation).	Missing or irrelevant.	/3
6. Additional Plans (1 pt)	At least one plan (training, security, maintenance) included.	Mentioned but vague.	Minimal or irrelevant.	Not included.	/1
7. Overall Quality (1 pt)	Well-written, well-structured, free of major grammar/formatting issues, follows template fully.	Clear but some formatting/clarity issues.	Hard to follow, weak formatting.	Poorly presented or incomplete.	/1

Note: If a student is listed as a member at the beginning of the SPMP **but not mentioned within the tasks, roles, or responsibilities throughout the document**, that student **will not receive the same score as active contributors**.

- Marks for that student may be reduced to reflect lack of contribution.
- Instructors should verify **role allocation tables, WBS assignments, and responsibilities** for evidence of individual involvement