CSC 305: Software Engineering

Term 1: 2025/2026

Course Coordinator: Dr. Rahma Ahmed

Project Proposal Form

Section A: Software Details:

1. Python code analyzer

---

2. Team Information

Name	**Student ID**	**Role in Project**
Zahra Bader Hayyan	2240006156	Tester
Hadeel abduallah alqhtani	2240003327	Reports writer
Noor Almuhsen	2240003505	Frontend developer
Yomna Al moslem	2240003445	Frontend developer
Alzahraa alabbad	2240006089	Backend developer
Eman Alnajem	2240002468	Backend developer, leader
Wedad Mohammed AL-hussaini	2240002853	Backend Developer

---

3. Project Overview

The proposed project aims to develop a Python code analysis tool that automatically examines Python source files and generates a structured report. This tool is meant to help developers, students, and beginner programmers by providing clear insights into their code. It highlights potential issues and suggests improvements to make the code easier to read and maintain.

The generated report starts with an overview that includes metrics like total lines of code, number of classes, functions, and imported libraries. Next, the tool offers detailed analyses of each class and function, covering their structure, parameters, and size. There is also a notes section that identifies potential issues such as code duplication, missing comments, or unused libraries, along with straightforward suggestions to improve code quality.

By automating code review tasks, this project promotes good programming practices, improves software quality, and provides educational value for learners. It acts as a helpful tool for personal development and future improvements in code analysis and software engineering.

---

4. Objectives

• To help programmers quickly understand the overall structure of their Python code without reading every line.

• To make it easier to maintain and improve code by identifying notes such as unused imports or missing comments.

• To support learners and beginners by giving them clear feedback that improves their coding practices.

• To provide a simple reporting tool that saves time and effort when reviewing or documenting code.

• To reduce errors caused by Python's sensitivity to indentation and small details by highlighting potential issues early.

---

5. Scope of the Project

The project will include:

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

The project will not include:

- **Support for other programming languages.**
- **Finding security or performance problems in the code.**
- **Running python programs.**

---

6. Key Features

Our tool will be able to provide the user with the following key features :

- The ability to read and analyze a python script
- Generates a structured report that including overview, details and suggestions.
- Extraction of classes, functions and imported libraries.
- recommendations/notes to improve the efficiency and readability of the script.
- Error handling (e.g., syntax errors) without interrupting the analysis.
- Quality improvement
- Support for collaboration and reusability by offering clear and organized reports.

---

7. Tools and Technologies

- Python 3.x
- ast and inspect libraries for parsing and code structure analysis
- os and pathlib for managing files
- radon for complexity and maintainability metrics
- pylint / flake8 for linting and style checks
- jinja2 for generating structured HTML reports

- GitHub for version control and collaboration
- Visual Studio Code and PyCharm for development
- Figma / Lucidchart for UML diagrams and design
- unittest / pytest for testing

Techniques:

- Static code analysis :analyzing source code without executing it
- Abstract Syntax Tree (AST) parsing :to extract classes, functions, and imports
- Metrics calculation (lines of code, comments, classes, functions, nesting levels)
- Complexity analysis (e.g., cyclomatic complexity to detect functions that are difficult to maintain)
- Error handling for syntax errors and incomplete code
- Report generation in a structured and user-friendly HTML format
- Testing and validation on Python projects of varying sizes

---

8. Timeline / Milestones

Phase	Description	Target Date
Planning	Prepare and submit the project management plan.	12 Oct
Status report 1	Submit the first status report	19 Oct
Requirements	Define and document project requirements	2 Nov
Status report 2	Submit the 2nd Status report	16 Nov
Development	Implement the tool's functionality, write code for all required features plan driven	25 Nov
Testing	Develop the project test plan, write test cases, and define testing approach.	30 Nov

رح تحاسب على التفاصيل

Phase	**Description**	**Target Date**
Final Delivery	Conduct final testing, polish documentation, and submit the complete project.	14 Dec

---

9. Potential Challenges

Understanding and analyzing code structure – Accurately interpreting different programming constructs and patterns can be complex.

Handling diverse code inputs – The tool should work correctly with programs of varying sizes, styles, and complexity.

Ensuring accurate results – Avoiding false positives or missed issues in the code analysis requires thorough validation.

Integrating multiple components – Ensuring that all parts of the tool work seamlessly together may require careful coordination.

Error handling – Properly managing unexpected inputs or edge cases to prevent crashes or incorrect analysis.

Optimizing performance – Ensuring the tool runs efficiently, even with larger or more complex code.

Creating a user-friendly interface – Presenting analysis results clearly and intuitively for users.

---

10. Expected Outcomes

The final product will deliver a Python-based code analysis tool that automatically examines Python source files and produces a structured HTML report. The report will include an overview of the project (such as total lines of code, number of classes, functions, and imported libraries), a detailed breakdown of each class and function, and notes with recommendations for improving the code.

The impact of the project is to provide students and beginner programmers with a simple tool that supports learning, saves time during code review, and supports good programming practices. By highlighting potential issues such as unused imports, missing comments, or duplicated code. The tool helps improve software quality and makes code easier to understand, document, and maintain

---

Section B. Evaluation

To be completed by the instructor or supervisor.

2. Project Proposal (12 points / 2 marks) – Week 4

Criteria	Excellent (4)	Good (3)	Fair (2)	Poor (0-1)	Awarded Score
Title & Motivation	Strong, concise title; well-justified problem.	Adequate but generic.	Weak or unclear link.	Missing.	/4
Team Details	Full list with clear roles and leader.	Some roles unclear.	Incomplete info.	Missing.	/4
Completeness	All template sections filled with quality.	Most sections complete.	Several missing/weak.	Template ignored.	/4