PM4 - Code Analyzer

Group: Control+Alt+Defeat

Process Deliverable

The SE process that we chose to use to complete the group project, as outlined in our project proposal, PM2 and PM3 , is Agile. The Agile process has worked well for us so far so we do not want to change. Because the process we are using is Agile, for the process deliverable we will submit a summary from our end of sprint retrospective, and our prioritized tasks.

What went well: We broke down the upcoming tasks and the way we are going to prepare for our upcoming presentation.

What didn't go well: Throughout the sprint, communication was not the best. This has been our biggest issue throughout the semester which we are trying to overcome. There was a significant amount of days and weeks where there was no communication which lasted close to the deadline. This has not hindered our ability to produce our deliverables on time and good quality but as the semester progresses it will become more difficult to complete the work if we do not communicate.
What can be improved: Communication can be improved, specifically the communication around the plan for our project milestones. We have done this in our sprint so time will tell if we can follow our plan.

Prioritized tasks for the next sprint:

- Complete PM5
    - Final Presentation
    - Presentation Preparation
    - Final Report
    - Retrospective
- Communication throughout our team, specifically around uncertainty within project milestones and the upcoming presentation.

Black Box Test Plan

Test Case	Test Case Description	Input	Expected Output	Actual Results	Pass/Fail
1	Test if the program identifies similar coding styles	Input code: Function with clear variable naming conventions	Output: Similar coding style identified for team "A"	Empty	Pending
2	Identify code with poor error handling	Input code: Function without exception handling for edge cases	Output: Suggests improvement in error handling	Empty	Pending
3	Categorize highly commented code	Input: Code with inline and block comments explaining each step	Output: Team assignment includes note of detailed documentation practices	Empty	Pending
4	Handle empty input	Input: Blank file	Output: Error message: "No code provided"	Empty	Pending
5	Extremely large files	Input: Large code file	Output: Loading code output, estimated time: (estimatedTime)	Empty	Pending
6	Test with a supported language	Input: Language file	No errors reported	Empty	Pending
7	Check for identification of inefficient code patterns.	Input: Test code	"Nested loops may lead to performance issues."	Empty	Pending
8	Clear Error Messages	**:** Invalid Python code missing a	Error: "SyntaxError:	Empty	Pending

		closing parenthesis.	Missing closing parenthesis on line X."		
9	Test with code containing an SQL injection vulnerability.	Vulnerable code	"Potential SQL injection vulnerability."	Empty	Pending
10	Verify the tool's output in JSON format.	Python Code	Json File	Empty	Pending