Project Proposal

Project Name  
The name of this project is **Mutest**.

Team with Contact information  
The project idea and execution is going to be developed by only one person.   
Name: Vinnicius Gabriel Rocha Santiago de Castro  
Email: [cas21003@gmail.com](mailto:cas21003@gmail.com)   
Number: (208) 607-5606.  
  
Project Purpose  
The project purpose is to design and implement a tool that introduces controlled mutations into Python source code and reports a mutation to evaluate the effectiveness of the test suite.

Background   
I’m a software engineering student with a quality assurance certificate and I have been working on my Python abilities. Unfortunately, I’m not used to writing test cases and unit testing (because the certification focused more on the documentation and requirements rather than testing, so even though I should be more knowledgeable in the testing side but I’m not so I will need to research and try an error during the development of the project.

Description  
Mutest is a software tool that applies mutation testing techniques to evaluate the effectiveness of existing automated test suites. The system works by introducing small modifications, called mutations, into the source code. These mutations can include changes such as flipping Boolean values, altering arithmetic operators, or modifying conditional statements. Once mutations are applied, the tool executes the project’s test suite against the modified code.

If the test suite fails to detect the introduced changes, the mutations are recorded as surviving mutants. These surviving mutants represent weaknesses in the test suite and are stored for further analysis. Mutest compiles the results into a structured report that highlights the types of mutations applied, the total number of surviving mutants, and the parts of the code where they occurred.

The tool is designed to function as an automated process. It will be implemented with a mutation engine capable of handling common code transformations, a test execution module that runs the existing tests against the mutated program, and a reporting component that generates outputs in both human-readable and machine-readable formats. An additional integration component will allow the tool to run as part of a continuous integration environment, ensuring that mutation testing can be triggered automatically during the software development workflow.

Significance  
This project demonstrates **advanced QA automation** and **pipeline engineering**, which are highly valued in industry.

On my resume, I would highlight:

* “Developed a mutation testing tool for CI/CD pipelines (Mutest), improving test suite robustness by detecting undetected code mutations.”
* Skills in **CI/CD, automated testing, software quality assurance, pipeline integration, DevOps tooling**.

New Computer Science Concepts  
To complete Mutest, I will need to learn:

* Mutation testing algorithms and AST (Abstract Syntax Tree) manipulation.
* CI/CD pipeline automation (e.g., GitHub Actions custom runners).
* Possibly Docker containerization for isolated test runs.
* Reporting/visualization (generating readable reports for developers).

Interestingness  
I’m interested in QA and CI/CD, and this project combines both.

* It solves a **real-world problem** developers face: weak but “passing” test suites.
  + Helps dev teams detect weak test coverage earlier.
  + Reduces risk of undetected bugs.
  + Improves confidence in automated testing pipelines.
* It’s **resume-worthy** and aligns with my career goals in QA/DevOps.
* It challenges me to learn new technical concepts (AST, pipeline integration).

Tasks and Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Task** | **Hours** | **Goal** |
| 1 - 2 | Research mutation testing and AST basics | 18 | Research foundation and parsing prototype |
| 3 – 4 | Build basic mutant generator (3 operator types only) | 18 | Core mutation engine prototype |
| 5 - 6 | Implement test execution and kill/survive tracking | 20 | First end-to-end test run |
| 7 | Mutation score calculation | 10 | Mutation scoring integrated |
| 8 - 9 | Basic CLI and simple reporting | 18 | Usable CLI with reporting |
| 10 - 11 | Minimal CI/CD integration | 15 | Pipeline-ready prototype |
| 12 | Documentation | 10 | Complete documentation |
| 13 | Testing the tool | 10 | Final tests |
| 14 | Final polish and presentation prep | 10 | Final demo and presentation |

Total hours: 129

I would like to acknowledge the assistance of OpenAI’s ChatGPT, which helped me in brainstorming and organizing the project schedule.

Resources  
Languages: Python.   
Tools: GitHub Actions, Docker, VS Code IDE  
Research: AI, online docs.   
Estimated Cost: $0

Dependencies

* Python 3.x runtime.
* Testing framework: PyTest (initial).
* GitHub Actions for CI/CD.
* Local dev: Windows + WSL (Linux).
* Deployment: GitHub repo.
* No special hardware/software purchases are required.

Risk  
This project main risk is the **learning curve** (knowledge gaps), however, this is not the only risk spotted, risk include:

* **Time risk**: Feature creep (too many features beyond 129 hours).
* **Integration risk**: Ensuring smooth pipeline automation.
* **Scope risk**: Keeping language support simple (start with Python only).