Project Title: GenAI Test Bot

Supervisor Name: John Peng
Email: kongyijipeng@gmail.com
Organization: Kongyiji
Description: We will build a bot that is integrated into Git that can write documentation and test suites for a submitted Git commit.

Goal: To create a bot integrated into a CI/CD (continuous integration and continuous deployment) workflow that can write documentation and test suites, which will be accurate > 90% of the time. Testing will be confined to testing Python and TypeScript code, using pytest.

Expected functionalities:

- The bot will generate code documentation and test suites automatically, based on each change made - and then committed - into the code repository. It will be integrated into the CI/CD pipeline and Git.
- The bot will be able to take user feedback and provide improved results based on that feedback and the imperfect first generation.
- (Optional) The bot will utilize user feedback to fine-tune a LLM that specializes in code generation based on pre-trained general-purpose OpenAI models.

Our Purpose & Stakeholders:
The GenAI Test Bot project aims to reduce the intensive manual labor of developers by building a bot that automates documentation and testing with up to 90% accuracy, which is time-saving and helps enhance the quality of code written. As a result, project managers will also enjoy improved code quality, faster development cycles, and streamlined project management. Furthermore, quality assurance teams will have access to comprehensive, automatically generated test suites, simplifying testing efforts. Lastly, end users will benefit from higher-quality software with fewer defects, ensuring a more reliable user experience.

Plan of Action:
As a starting point, a working set of human written test cases will be provided. We will not train an LLM model ourselves, hence no need for independent training data. Instead, we will use OpenAI APIs to access their pre-trained models. The AI will generate code documentation and test suites. We will also experiment with explicitly building code level semantics into generation, such as taking advantage of CFGs to predict execution paths. We will evaluate our application based on the code coverage and the correctness of test suites written by the bot, without human intervention. We aim for an accuracy of at least 90% in both metrics.

We will also look into potentially designing a system that takes user feedback and uses it to fine-tune a LLM, which will not only reach parity and beyond in terms of code generation performance, but also cost only a fraction in the long run, as opposed to being completely reliant on OpenAI GPT services. The performance of the bot can be improved iteratively in this way.

Tools/programming languages/product management systems used will be:

Python, TypeScript, Git, Github APIs, OpenAI APIs, AWS.