GenieTest - Functional Test Generation Tool using the Power of AI

- Introduction
- Architecture Diagram
- Data Generation and Cleaning
- MCP Host and Server
- Components Supported
- Pre-requisites
- Prompt Practices
- How to Use?
- Future Enhancements
- Make Test Generation Better and More Reliable with GenieTest!

Introduction

Helps QE engineers by fast tracking the process of writing test automation scripts!

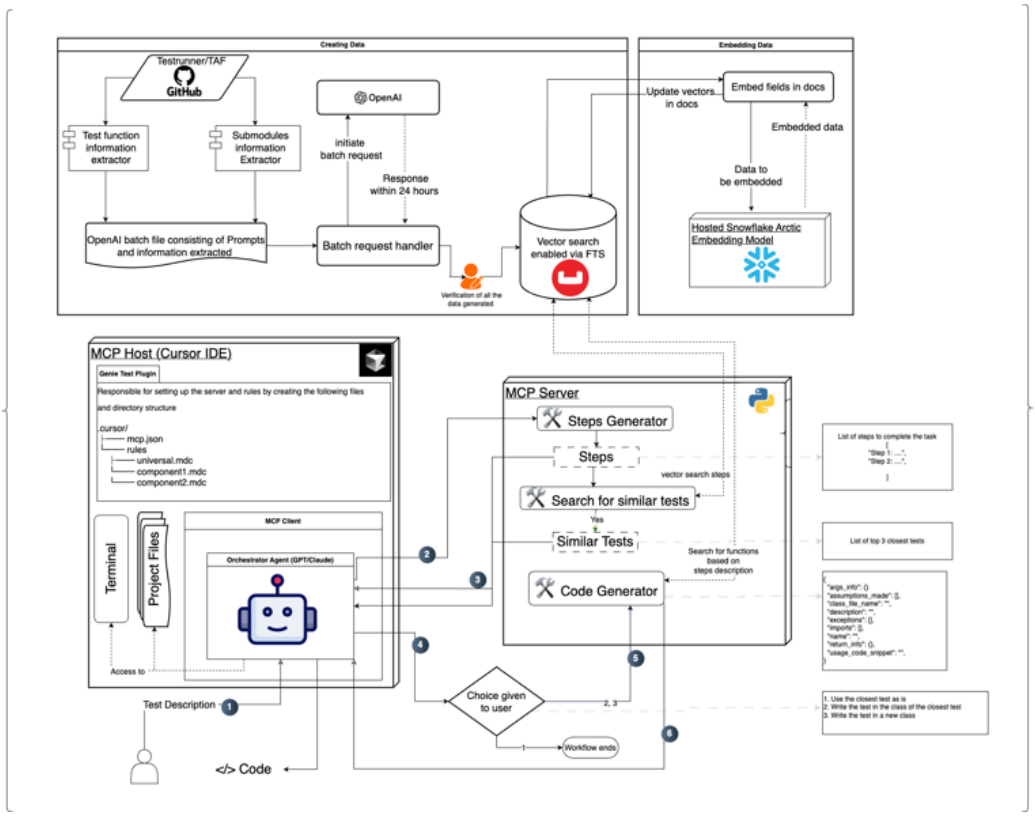
Why is Cursor not enough?

- Does not have project specific context
- Can’t control what is and is not prompted to the user
- Not very reliable, especially with large projects

We overcome this problem of Cursor by the following:

- MCP Server
- Cursor Rules (Prompt Engineering)
- Extension for Cursor
- Creating a custom dataset for RAG!

Architecture Diagram



Data Generation and Cleaning

- Chunked data from Testrunner and TAF repositories to create the following databases:
  - **Test Functions Database:** Test functions are stored here along with their description, steps and functional dependencies.
  - **Helper Functions Database:** Helper functions from the Submodules are stored here along with their usage patterns.
- Utilised **OpenAI integration** and **Batch API** for initial data chunking and extraction.
- **Manual quality assurance** of the generated data was performed to ensure consistency, maintain uniform formatting and eliminate noises.
- Embeddings were generated using a **hosted Snowflake Arctic model**.

MCP Host and Server

MCP or **Model Context Protocol** is used to fetch context from external databases to your LLM. The MCP Host is the Cursor IDE. It consists of:

- The **MCP Client** (Cursor Agent)
- **Cursor Rules** - Added on top of the context to control how the tools are used and to maintain required testing standards (both generic and component specific)

- Access to the files of the test repository
- The Terminal

The MCP Server has the tools that are used for test generation. The following are the tools of GenieTest:

- **Steps Generator:** Breaks down user input into structured steps via an LLM call
- **Display Steps:** Displays the steps to the user for review
- **Show Closest Test:** Embeds the steps and vector searches into the Test functions database to find the closest test to exist
- **Generate Code:** Uses the **closest test/creates a new test in same class/creates a new test in new class** according to the user's preference. Vector searches each step into the Helper functions database to fetch helper functions that can be used for code generation. It gives this as a context to Cursor to write the new test.
- **Review and Edit Tool:** Given the code and instructions to edit an existing test, this tool fetches closest tests along with helper functions that can be used. This new context is used by Cursor to edit the code.

Components Supported

GenieTest's tools can be tested on the following components:

- Eventing
- XDCR
- GSI

We are working on extending support to more components in the coming days!

Pre-requisites

- VS Code version 1.90.0 or higher
- Active workspace with a Test Repository (Testrunner or TAF)
- Running MCP Server instance

Prompt Practices

Better your prompts are, better the generated code is. The following are a few prompt practices we advise you to follow:

- A well written test description with clarity on the **test environment, functionalities, resources and constraints** if any.

- Include keywords and phrases to make sure that the MCP Client (Cursor) fetches the appropriate tools from the MCP Server. **“Can you generate steps for this test?”**, **“Show the Closest Test”**, **“Generate Code for this Test”** are sample phrases that can help the MCP Client fetch appropriate tools.

How to Use?

- Install the Cursor Extension - to auto-configure the MCP Server and Cursor Rules
- Go to **Cursor > Cursor Settings > MCP** and make sure that the ***genie-test*** **MCP** is enabled
- Make sure you are on the ‘***Agent’*** mode on Cursor as this best supports the MCP Tools
- Go ahead and give prompts to describe the tests that you want our tool to generate.

View our Demo Videos to get better clarity on prompt practices and how GenieTest’s tools are used.

Future Enhancements

- Extend to other components of Testrunner and TAF
- Optimisations to the Review and Edit Tool
- Better conf support
- Updating the databases
- Jira Integration

Make Test Generation Better and More Reliable with GenieTest!

Do try our tool out and let us know your thoughts!