Testing GPT Agent

[GitHub Repository: GPT-Pilot]

Task:

Create a **web-based calculator application** in Python that performs addition and subtraction. Later, the user requests modifications to:

- Change the background color to yellow.
- Add a multiplication operation.

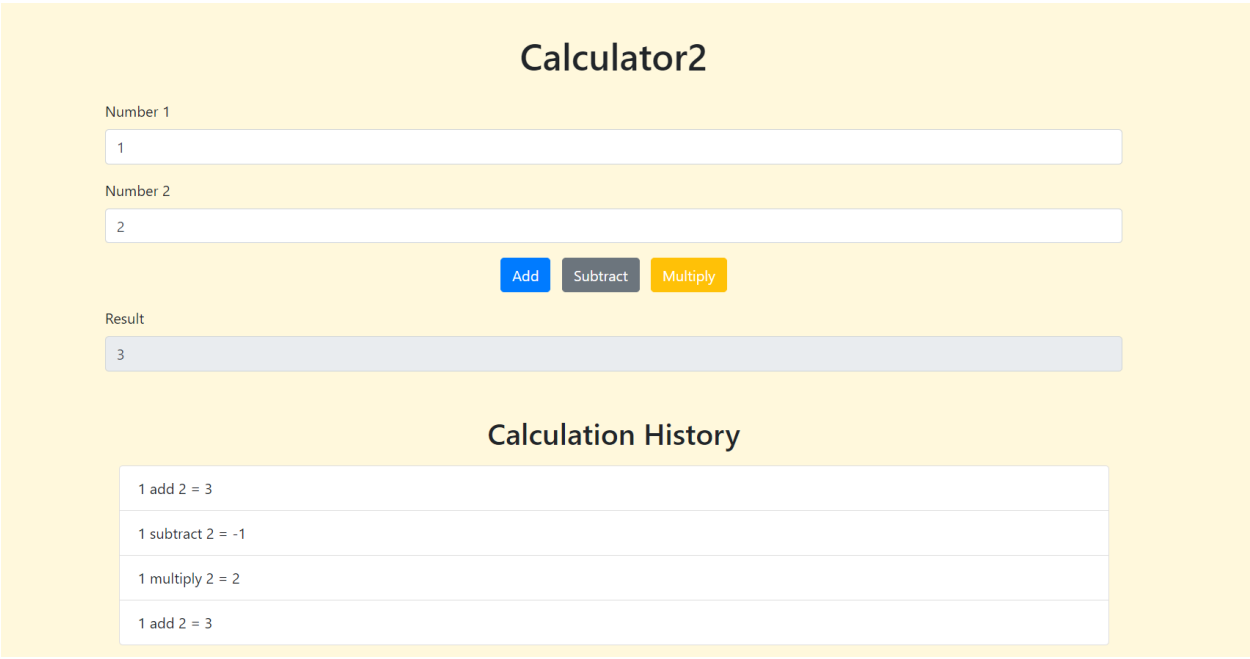
Advantages:

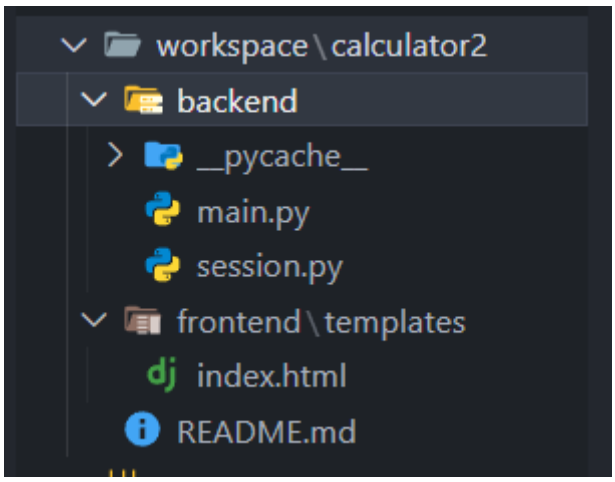
- **User feedback after every step**: The agent confirms with the user if everything is correct after each step of the development process.
- **Dependency management**: Automatically installs required Python libraries if they are missing.
- **Comprehensive repository search**: Scans the entire repository for context instead of limiting itself to active files.

Disadvantages:

- **No integration with existing projects:** The agent generates new code but cannot modify already existing codebases.
- **Sequential execution only:** It is not possible to execute individual steps on demand — the entire process must run from start to finish.
- **Limited output options:** Output is provided exclusively via the console. Integration into VS Code requires a paid plugin.
- **Model API connection:** You need to connect to model APIs, which can be expensive when using cloud services, or set up models locally, which may be inconvenient for some developers due to high computational resource requirements.

This setup highlights both the potential and limitations of the GPT-Pilot, making it a valuable yet situational tool for automated coding workflows.





README.md

workspace > calculator2 > README.md > # Calculator2 > ## Overview

```
# Calculator2

Calculator2 is a web application designed to perform basic arithmetic operations with a modern user interface. It leverages FastAPI for the backend and Bootstrap for a responsive frontend design. The application allows users to perform addition, subtraction, and multiplication operations, while maintaining a session-based history of calculations.

## Overview

Calculator2 is built using a combination of FastAPI for the backend and HTML/CSS with Bootstrap for the frontend. The backend handles arithmetic operations and session management, while the frontend provides a user-friendly interface for interaction. The application architecture is divided into two main components:

- **Backend**: Implemented in Python using FastAPI, it processes requests for arithmetic operations and maintains a session history of calculations.
- **Frontend**: Developed using HTML and styled with Bootstrap, it includes input fields for numbers, options to select operations, and displays results and calculation history.

### Project Structure

- `backend/`: Contains the FastAPI application and session management middleware.
  - `main.py`: The main FastAPI application file that defines routes for arithmetic operations.
  - `session.py`: Implements session management using cookies and an in-memory session store.
- `frontend/templates/`: Includes HTML templates for the frontend interface.
  - `index.html`: The main HTML page for the application, styled with Bootstrap.
```