Project Documentation – Test Mate

1. Overview

Test Mate is a lightweight proof-of-concept web application that automatically generates unit tests for Python functions.
The goal is to provide developers with a simple interface where they can paste their code, generate tests using an AI model, and then copy or download the results.

This project was designed as an MVP: it is not meant to be a production system, but a tangible prototype to showcase how AI-powered test generation can be integrated into a developer workflow.

2. Features

- **No sign-up required**: The app can be accessed online directly via a browser without creating a profile.
- **AI-powered test generation**: Python code is sent to a backend endpoint which communicates with an LLM to generate corresponding unit tests.
- **Editable results**: Users can view, edit, copy, or download the generated tests before using them.
- **User-friendly interface**: Built with React + Vite, styled simply for quick usage.
- **Deployment-ready**: The entire system runs inside Docker containers for portability and easy setup.

3. Architecture

- **Frontend**: React + TypeScript, providing the user interface.
- **Backend**: FastAPI service that exposes a `/generate-test` endpoint. This endpoint handles the AI request and returns test code.
- **Nginx**: Serves the frontend build and proxies requests to the backend.
- **Deployment**: Hosted on an AWS EC2 instance, exposed under a single URL.

3.1 Backend – Technical Details

The backend is built with **FastAPI** and connects to Hugging Face Inference API for LLM-based test generation.

Key configuration:

```
    CHAT_URL = "https://router.huggingface.co/v1/chat/completions"

    MODEL_ID = "meta-llama/Meta-Llama-3-8B-Instruct:novita"
```

4. Deployment

The application is currently deployed on **AWS EC2**:
👉 http://13.48.104.125/

Both the frontend and backend are reachable under this address, using Nginx as a reverse proxy.

For local development and testing:

```
docker compose build
docker compose up
```

This will start all required services, and the app will be available at `http://localhost:80`.

5. How to Use

1. Open the app in your browser.
2. Paste your Python function into the **Source Code** editor.
3. Click **Generate Tests**.
4. The generated pytest code will appear in the **Generated Tests** panel.
5. Copy or download the results to integrate them into your project.

Note: If you want to try the system without writing new code, an additional file named **`samples.py`** is included under the docs folder. This file contains small example functions that can be used to quickly test the generator. You can paste any of these examples into the editor to see how the test generation works.

6. Limitations

- The AI model may sometimes produce incomplete or incorrect tests.
- Only Python functions are supported in this MVP.

- Error handling is basic and meant only for prototyping purposes.

7. Next Steps / Improvements

- Support for multiple programming languages.
- Smarter error handling and user feedback.
- Integration with GitHub repositories for direct test generation.
- Option to run tests directly in the browser.