FastAPI Gemini AI Integration

Project Report

GitHub: https://github.com/peedaluk/fastapi-gemini-ai

1. Introduction

The FastAPI Gemini AI Integration project demonstrates how to build a secure, scalable REST API for generative AI using Google Gemini and Python's FastAPI framework. The system is designed to provide AI-powered text generation services via HTTP endpoints, enabling easy integration of large language models into modern applications.

2. Objectives

- Create a robust backend API for generative AI using FastAPI.
- Securely integrate Google Gemini AI for text generation.
- Provide a modular, extensible architecture for further AI enhancements.
- Document and demonstrate API usage with interactive documentation and example requests.

3. System Architecture

3.1 High-Level Overview

- API Layer: Built with FastAPI, exposes RESTful endpoints for client interaction.
- AI Integration: Communicates with Google Gemini via the official SDK and API key.
- Security: Uses environment variables for API key management to prevent credential leaks.
- Documentation: Auto-generates OpenAPI/Swagger UI at `/docs` for interactive exploration.

3.2 Component Diagram

Component	Description

FastAPI App	Handles HTTP requests and routes
Gemini Client	Sends prompts to Google Gemini and returns output
.env Configuration	Stores sensitive API keys securely
Uvicorn Server	Runs the ASGI application

4. Implementation Details

- API Endpoint `/chat`: Accepts a POST request with a text prompt and returns the Gemini AI-generated response.
- Environment Setup: Requires a `.env` file with `GOOGLE_API_KEY` for secure authentication.
- Error Handling: Returns appropriate error messages for missing keys or API failures.
- Extensibility: The modular codebase allows for easy addition of new endpoints or AI models.

5. Usage and Results

5.1 Setup

1. Clone the repository and install dependencies:
2. `git clone https://github.com/peedaluk/fastapi-gemini-ai.git`
3. `cd fastapi-gemini-ai`
4. `pip install -r requirements.txt`
5. Add your Google Gemini API key to a `.env` file:
6. `GOOGLE_API_KEY=your_google_gemini_api_key`
7. Start the server:
8. `uvicorn src.main:app --reload`

9. Access the interactive documentation at http://localhost:8000/docs.

5.2 Example Request

- Endpoint: `POST /chat`
- Request Body:
- `json`

```
{ "text": "Explain the concept of generative AI." }
```

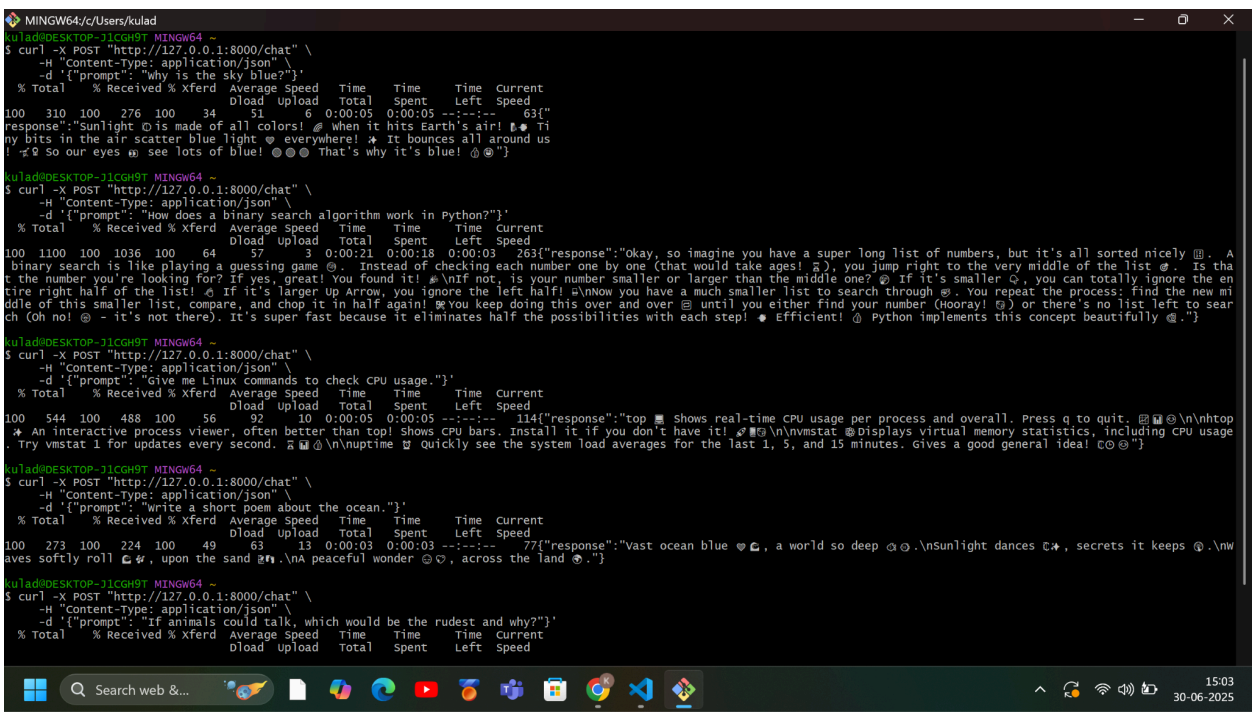
-
- Sample Response:
- `json`

```
{ "response": "Generative AI refers to..." }
```

-

5.3 Screenshots

Swagger UI Documentation



Sample Chat Request.

6. Evaluation

- Performance: FastAPI ensures low-latency responses and efficient handling of concurrent requests.
- Security: API keys are never hardcoded; environment variables are used throughout.
- Usability: The `/docs` endpoint allows for rapid testing and onboarding of new users or developers.
- Extensibility: The architecture supports the addition of new AI models, endpoints, or business logic with minimal changes.

7. Conclusion and Future Work

This project successfully demonstrates how to integrate state-of-the-art generative AI into a modern web API using FastAPI. The design prioritizes security, modularity, and ease of use. Future enhancements could include:

- Support for Gemini's multimodal (image/text) endpoints.
- User authentication and request rate limiting.
- Persistent conversation history and advanced context handling.
- Deployment to cloud platforms with CI/CD pipelines.

8. References

- Project GitHub Repository: https://github.com/peedaluk/fastapi-gemini-ai
- Google Gemini API Documentation: https://ai.google.dev/models/gemini
- FastAPI Documentation: https://fastapi.tiangolo.com/
- Implementing Gemini AI with Python: A Simple Guide: https://blog.stackademic.com/implementing-gemini-ai-with-python-a-simple-guide-71f8c148d24a

Prepared by:
KULADEEP DASARI