AI-Powered Web Assistant

AI-Powered Web Assistant is a Flask-based web application designed to provide intelligent responses to user queries using an AI text-generation library created by the developer. As part of this project, a custom Python package named ai_helpers_meena was developed, documented, packaged, and uploaded to PyPI. This package contains reusable AI helper functions that process user prompts and generate meaningful responses.

The Flask application imports and uses this custom library to handle all AI-related operations. Users can enter a question or message in the web interface, and the system processes the input through the library functions and displays an AI-generated response. The interface is styled using Bootstrap for a clean, user-friendly experience.

This project demonstrates the complete workflow of:

- Building a Python library
- Packaging it using modern tools (build and twine)
- Publishing it to the Python Package Index (PyPI)
- Installing it using pip into an application
- Integrating the library within a Flask web environment

By completing this project, the developer gains practical experience in Python packaging, API integration, Flask development, template rendering, and web application deployment. It also showcases how custom AI utilities can be reused across multiple projects by converting them into installable packages.

Objective:

- Build a custom Python library (ai_helpers_meena)
- Upload it to PyPI
- Install the library using pip
- Use it inside a Flask application
- Integrate a real AI model (Google Gemini – gemini-2.0-flash)
- Create a responsive UI for chat interaction
- Maintain chat history

SOFTWARE & TECHNOLOGIES USED

Languages

- Python
- HTML
- CSS
- Jinja2
- JavaScript

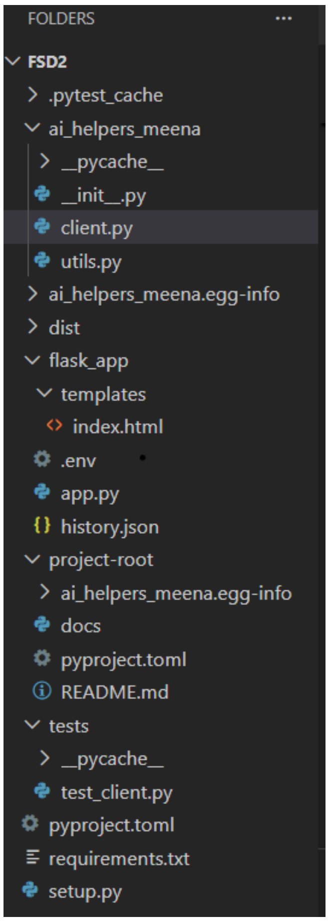
Frameworks & Libraries

- Flask
- google-generativeai
- python-dotenv
- SweetAlert
- Bootstrap
- FontAwesome

Packaging Tools

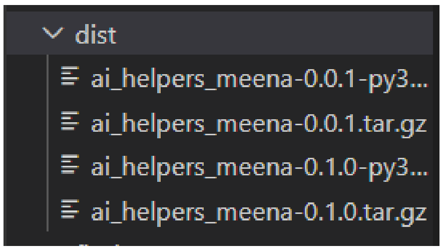
- setuptools
- wheel
- build
- twine

Creating the Python Library

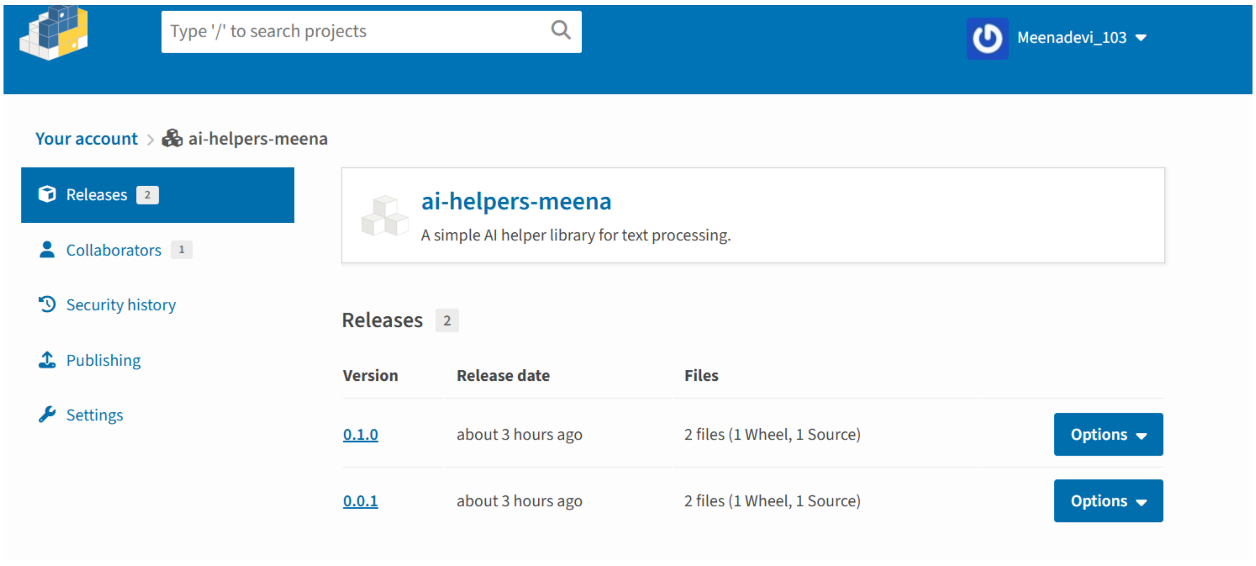


```
FOLDERS
FSD2
  .pytest_cache
  ai_helpers_meena
    __pycache__
    __init__.py
    client.py
    utils.py
  ai_helpers_meena.egg-info
  dist
  flask_app
    templates
      index.html
    .env
    app.py
    history.json
  project-root
    ai_helpers_meena.egg-info
    docs
    pyproject.toml
    README.md
  tests
    __pycache__
    test_client.py
  pyproject.toml
  requirements.txt
  setup.py
```

Packaging & Uploading to PyPI



```
dist
  ai_helpers_meena-0.0.1-py3...
  ai_helpers_meena-0.0.1.tar.gz
  ai_helpers_meena-0.1.0-py3...
  ai_helpers_meena-0.1.0.tar.gz
```



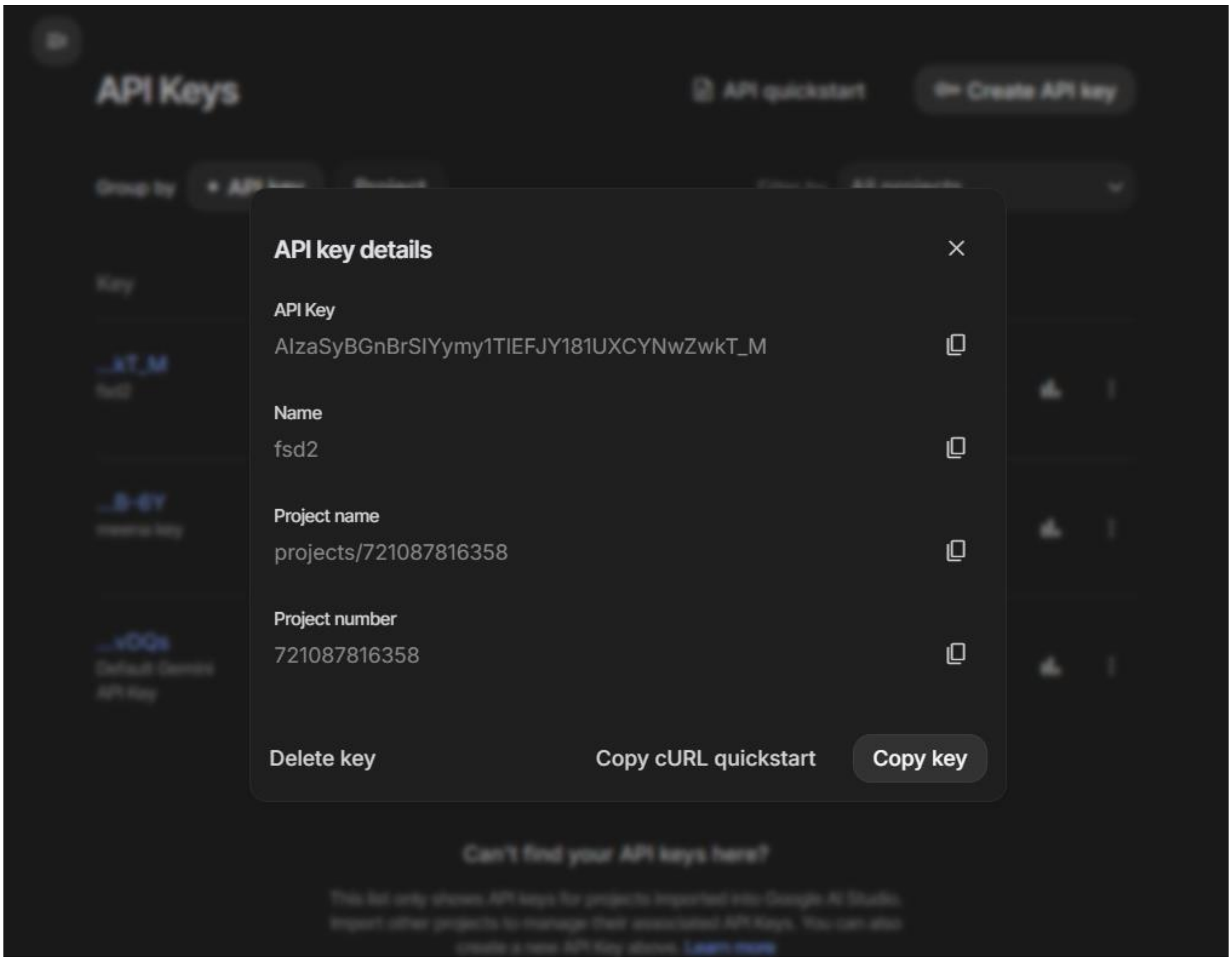
ai-helpers-meena

A simple AI helper library for text processing.

Releases 2

Version	Release date	Files	
0.1.0	about 3 hours ago	2 files (1 Wheel, 1 Source)	Options
0.0.1	about 3 hours ago	2 files (1 Wheel, 1 Source)	Options

API key details



API key details

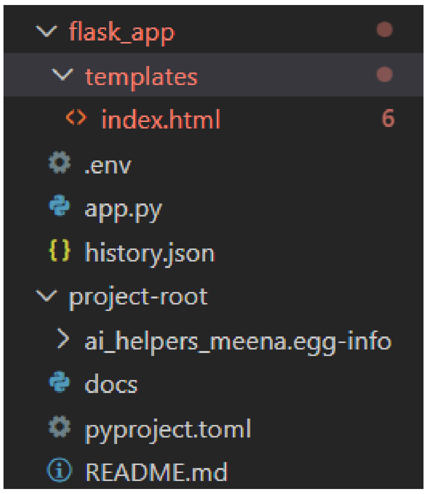
API Key
AIzaSyBGnBrSIYymy1TIEFJY181UXCYNwZwkT_M

Name
fsd2

Project name
projects/721087816358

Project number
721087816358

Flask Web Application Development



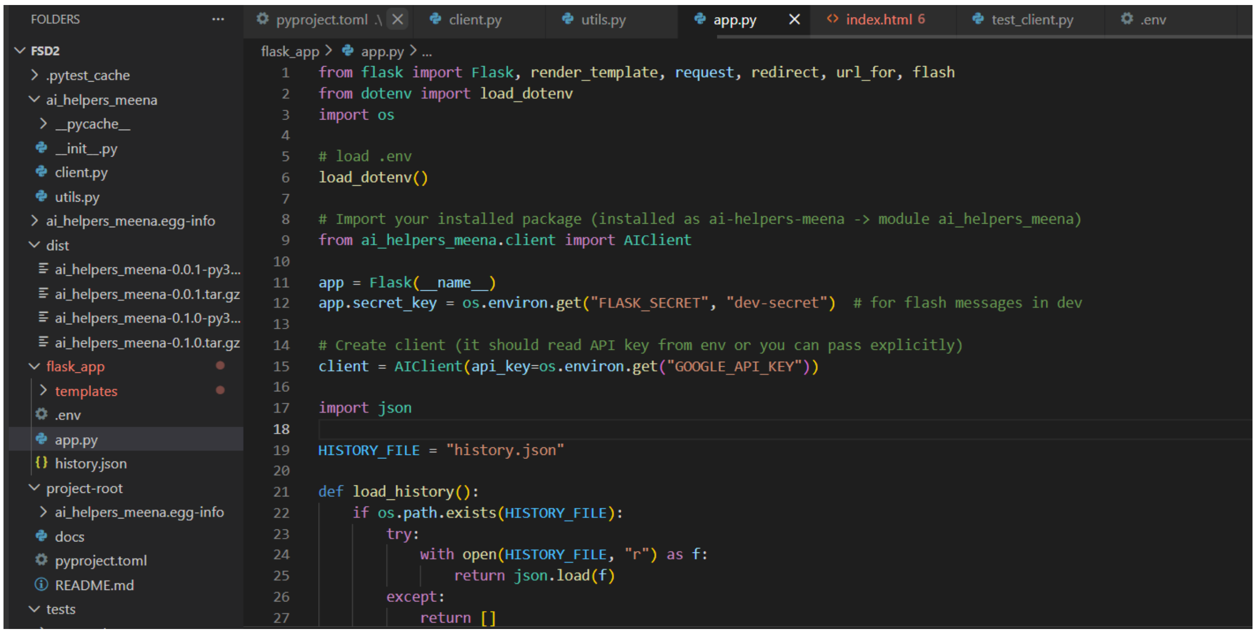
```
flask_app
  templates
    index.html        6
  .env
  app.py
  history.json
project-root
  > ai_helpers_meena.egg-info
  docs
  pyproject.toml
  README.md
```

app.py – Backend Logic

Handles:

- Loading API key
- Creating AIClient

- Processing user queries
- Storing chat history
- Rendering UI



```python
from flask import Flask, render_template, request, redirect, url_for, flash
from dotenv import load_dotenv
import os

# load .env
load_dotenv()

# Import your installed package (installed as ai-helpers-meena -> module ai_helpers_meena)
from ai_helpers_meena.client import AIClient

app = Flask(__name__)
app.secret_key = os.environ.get("FLASK_SECRET", "dev-secret")  # for flash messages in dev

# Create client (it should read API key from env or you can pass explicitly)
client = AIClient(api_key=os.environ.get("GOOGLE_API_KEY"))

import json

HISTORY_FILE = "history.json"

def load_history():
    if os.path.exists(HISTORY_FILE):
        try:
            with open(HISTORY_FILE, "r") as f:
                return json.load(f)
        except:
            return []
```

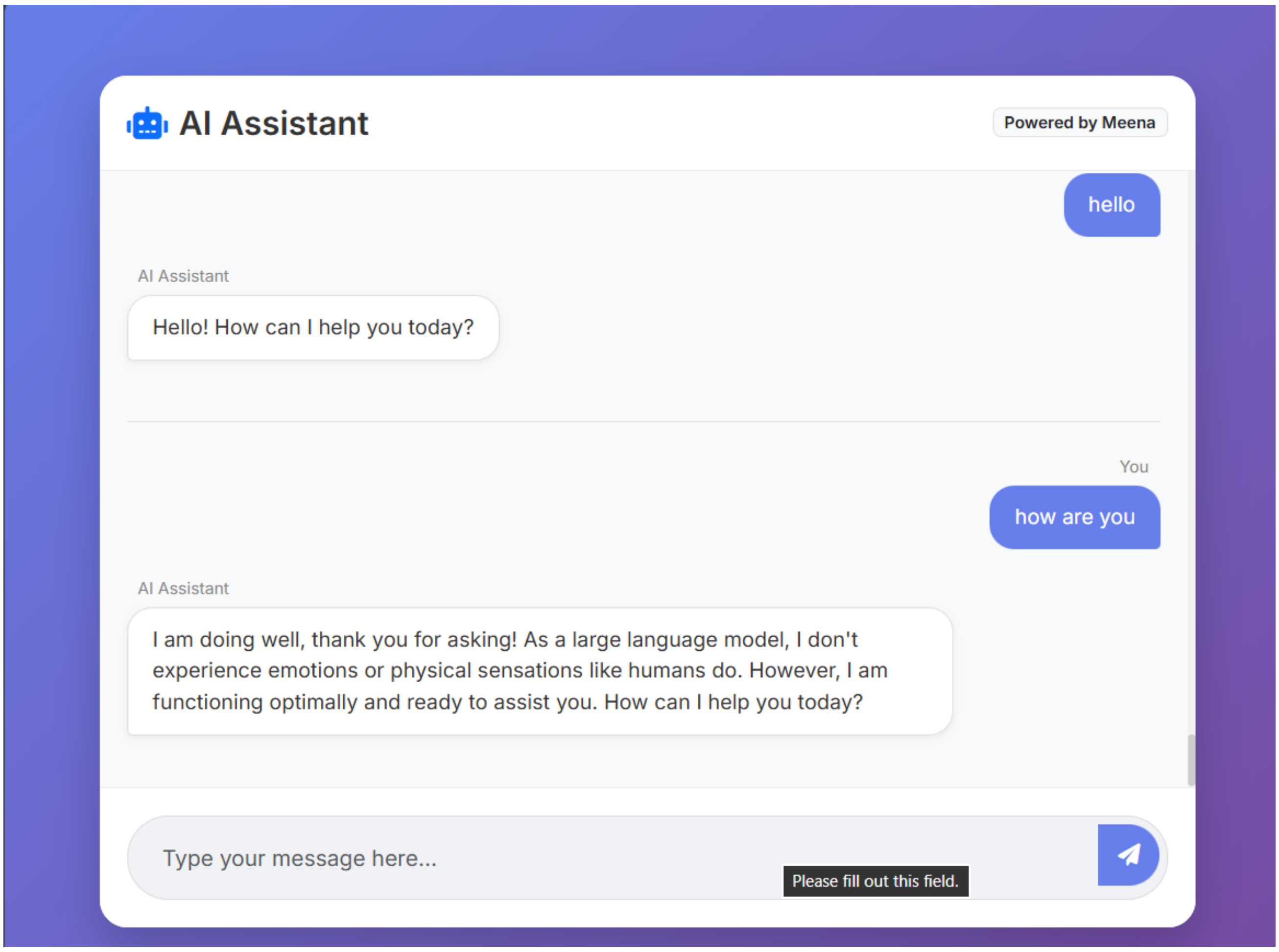
FRONTEND DESIGN — index.html

The UI includes:

- Chat bubbles
- Scrollable chat history
- SweetAlert notifications
- Elegant gradients and shadows
- Responsive layout

Key blocks include:

- User message
- AI response
- First-time empty state



hello

AI Assistant

Hello! How can I help you today?

You

how are you

AI Assistant

I am doing well, thank you for asking! As a large language model, I don't experience emotions or physical sensations like humans do. However, I am functioning optimally and ready to assist you. How can I help you today?

WORKFLOW OF THE APPLICATION

1. User enters a question.
2. Flask receives it and passes it to AIClient.
3. AIClient checks:
    - Valid API key → Uses gemini-2.0-flash model
    - No key → Uses mock responses
4. Response is formatted and returned.
5. UI displays messages in chat format.
6. History is updated.

CONCLUSION

This project demonstrates the complete pipeline of building a modular AI assistant system. The assignment helped in understanding:

- Creation and packaging of reusable Python libraries
- Publishing to PyPI
- Integrating Flask with external modules
- Using Google Gemini AI for text generation
- Building responsive, interactive frontends
- Managing environment variables and secrets securely

The final result is a polished, fully functional AI Web Assistant suitable for academic and real-world applications.

FUTURE ENHANCEMENTS

- Add voice input/output
- Add streaming responses
- Add user authentication
- Save chat history to a database