AI-Powered Web Assistant

This project presents an AI-Powered Web Assistant developed using Flask and a custom Python library published on PyPI. The system enables users to interact with an intelligent chatbot interface capable of generating contextual responses using Google Gemini AI. The project includes the creation of a custom Python package (ai_helpers_meena), API integration, user interface design, and real-time communication through a web-based chat system. The end result is a modular, scalable, and API-driven AI assistant that demonstrates practical knowledge of Python development, modern packaging tools, web technologies, and AI model integration.

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

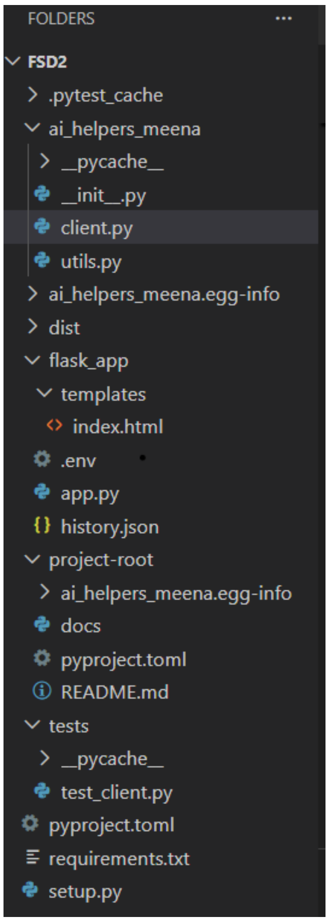
Frameworks & Libraries

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

Packaging Tools

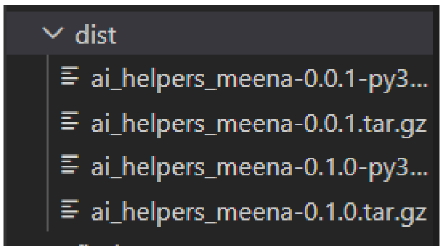
- setuptools
- wheel
- build
- twine

Creating the Python Library

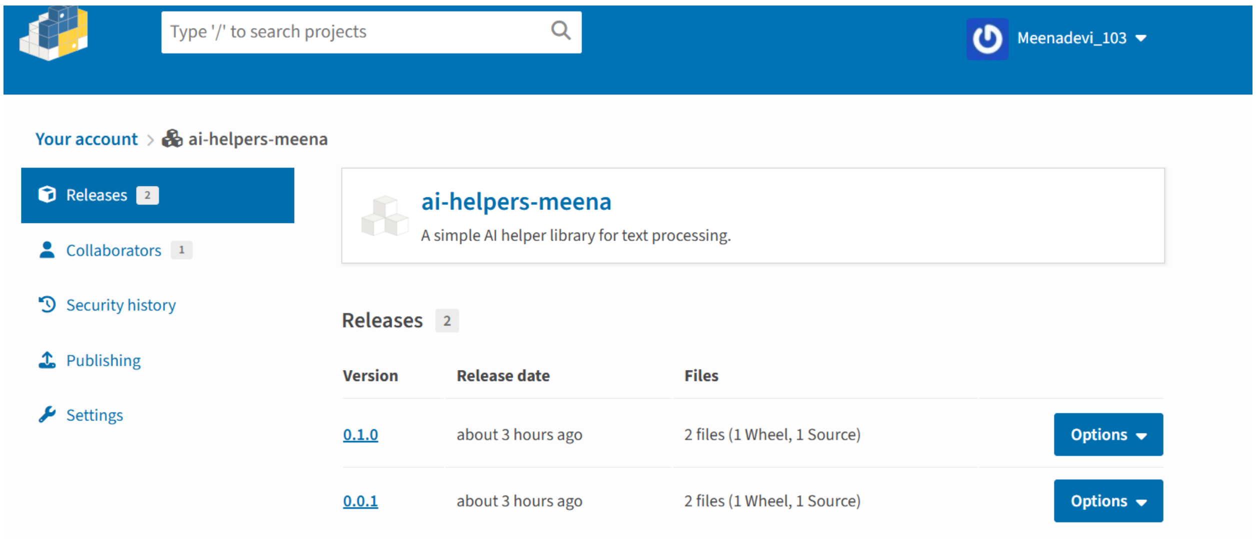


```
FOLDERS
v FSD2
  > .pytest_cache
  v ai_helpers_meena
    > __pycache__
      __init__.py
      client.py
      utils.py
  > ai_helpers_meena.egg-info
  > dist
  v flask_app
    v templates
        index.html
      .env
      app.py
      history.json
  v project-root
    > ai_helpers_meena.egg-info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

Packaging & Uploading to PyPI



```
v 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
```



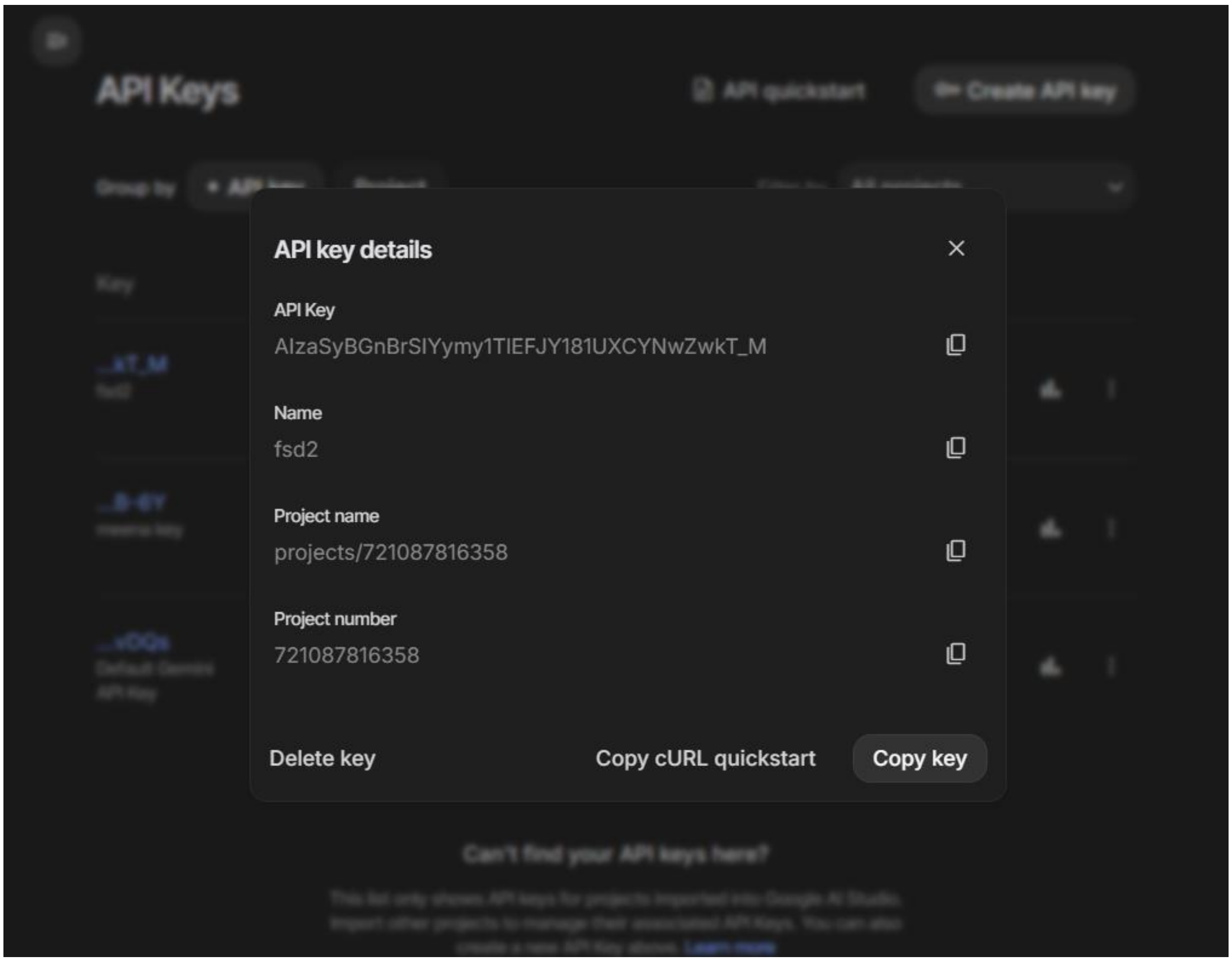
ai-helpers-meena

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

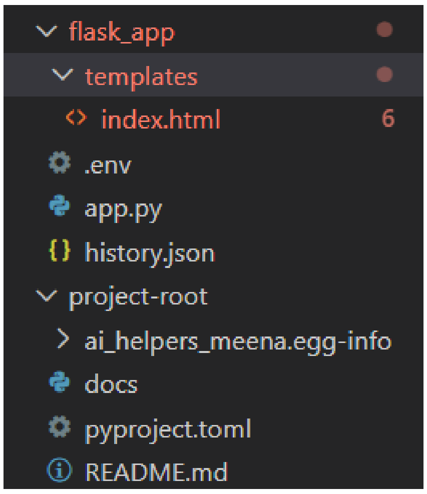
API Key
AIzaSyBGnBrSIYymy1TIEFJY181UXCYNwZwkT_M

Name
fsd2

Project name
projects/721087816358

Project number
721087816358

Flask Web Application Development



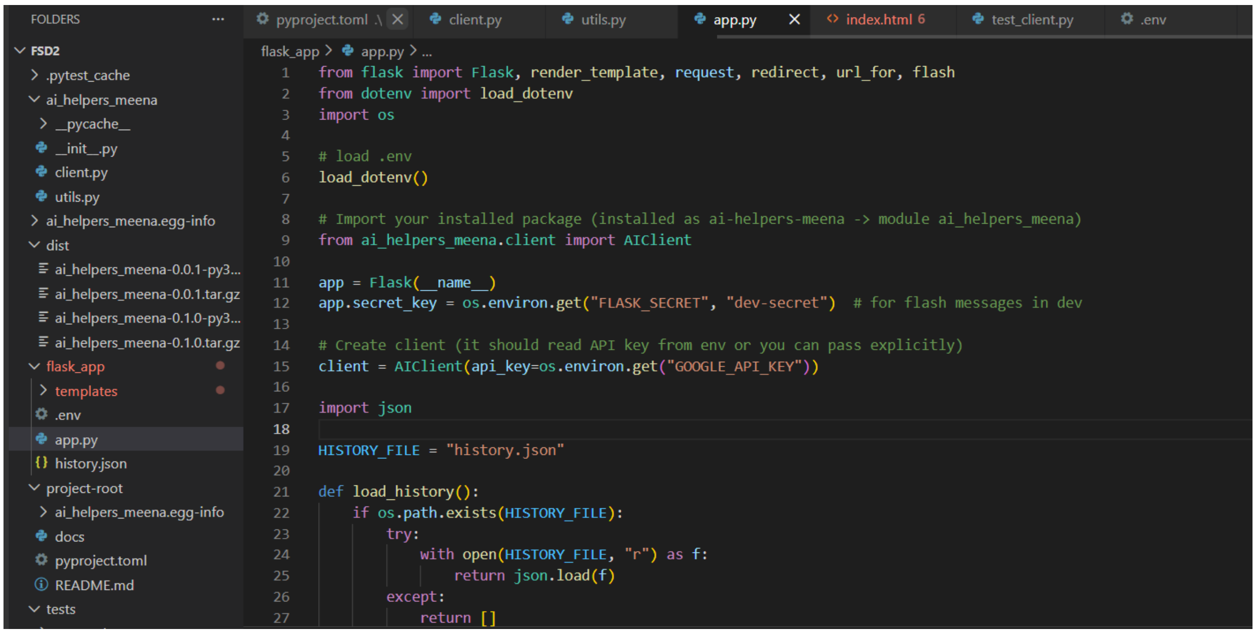
```
flask_app
  templates
    index.html        6
  .env
  app.py
  history.json
project-root
  ai_helpers_meena.egg-info
  docs
  pyproject.toml
  README.md
```

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

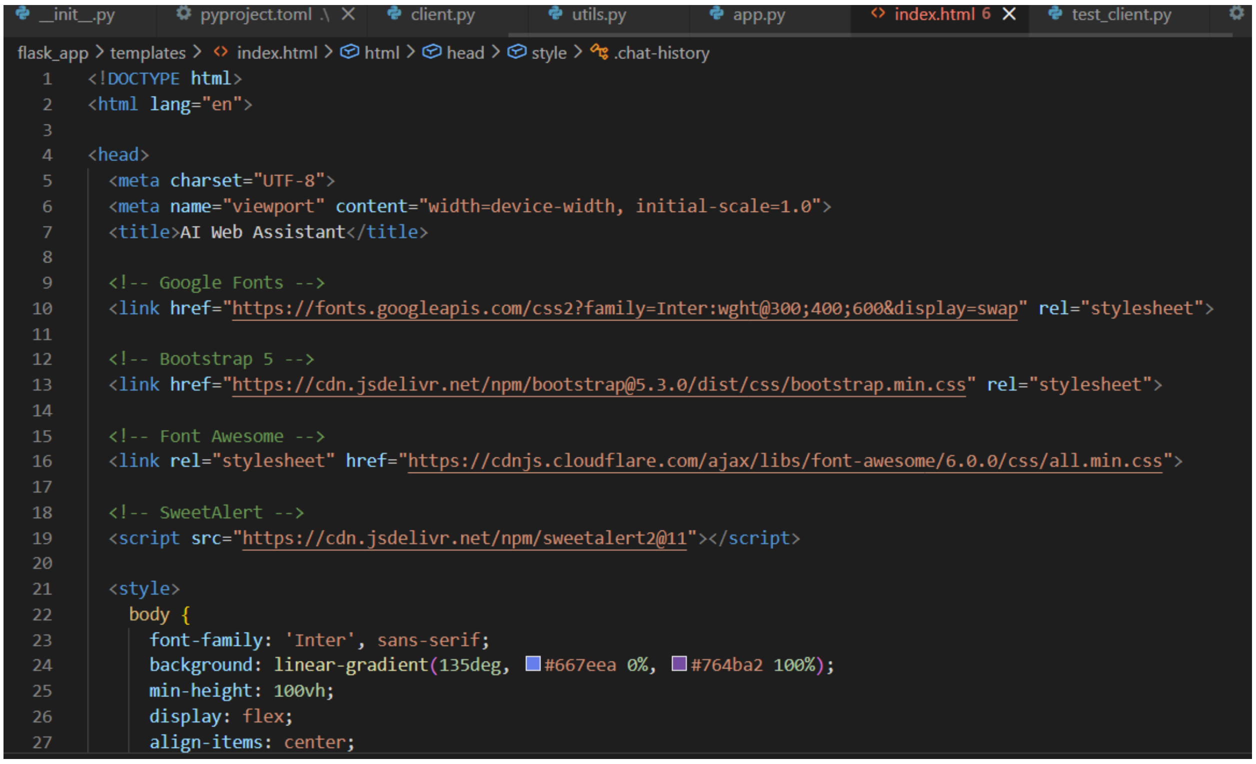
The UI includes:

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

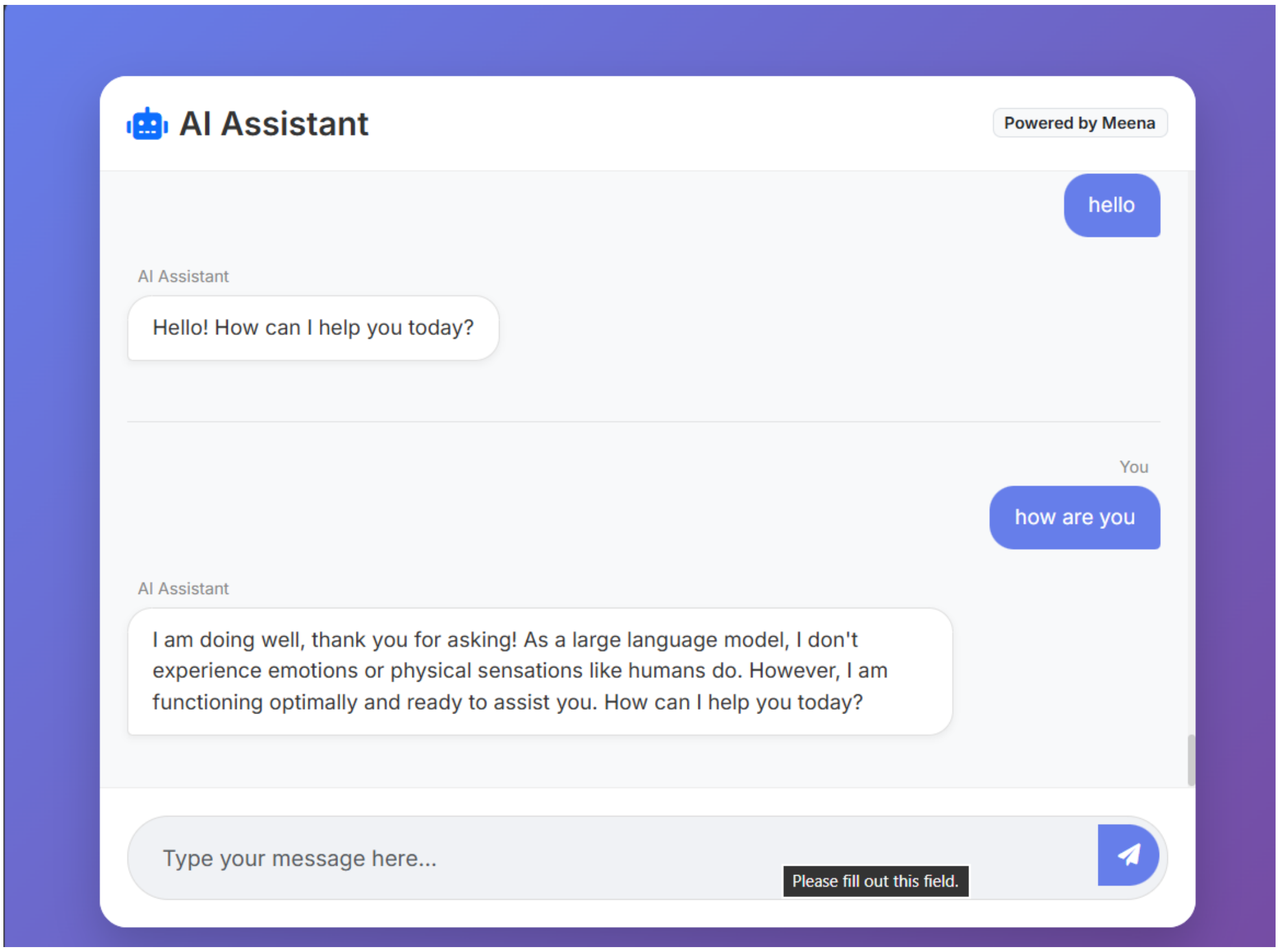
Key blocks include:

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

Index.html



```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AI Web Assistant</title>

  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap" rel="stylesheet">

  <!-- Bootstrap 5 -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

  <!-- Font Awesome -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

  <!-- SweetAlert -->
  <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

  <style>
    body {
      font-family: 'Inter', sans-serif;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      min-height: 100vh;
      display: flex;
      align-items: center;
```



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?

Challenges Faced:

- Model naming mismatch between Gemini v1 and v1beta caused API errors.
- Managing environment variables securely using .env.
- Ensuring the PyPI package installed correctly and updated versions took effect.
- Handling asynchronous chat-like UI layout with dynamic message rendering.
- Ensuring UI responsiveness across devices.

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