AI Assistant Web Application

Built Using Python Flask and OpenAI API

Presented by: **MAYANK SINGH**

Batch : **AI PE OCT 25 (apo235d9q)**

Phone No/ Email : **8881083549 / mayanksinghsara80@gmail.com**

Date: **21-OCT-2025**

Project Overview: The AI Assistant

This project introduces a functional web-based AI assistant, designed to showcase seamless integration between modern web development practices and powerful large language models.

The application serves as a versatile tool, capable of interacting with the OpenAI API to perform several key tasks:

- Answering complex questions.
- Summarizing lengthy text passages.
- Generating creative or descriptive content.

Core Project Goals

We aimed to deliver a solution that is both powerful in capability and simple in execution.

Intuitive Interface

Create an easy-to-use, accessible web interface for direct interaction with advanced AI models.

Versatile Functionality

Implement distinct features—Q&A, summarization, and content creation—all accessible from a single platform.

Clean & Responsive UI

Ensure a smooth user experience across devices using modern HTML/CSS design principles.

System Architecture: Three-Tier Integration

The application leverages a clear separation of concerns, connecting the user interface to the AI engine via a robust Python backend.

Frontend

HTML and CSS provide the structure and style for the user interface.

Backend

Python Flask manages routes, API calls, and environment variables (dotenv).

AI Engine

The OpenAI **gpt-3.5-turbo** model handles all processing and generation requests.

Core Features and Capabilities

Answer a Question

Provides accurate and detailed responses based on the AI's extensive knowledge base.

Summarize Text

Efficiently condenses long articles or paragraphs into digestible, concise key points.

Generate Creative Content

Produces imaginative output, such as stories, poems, code snippets, or fresh ideas from simple prompts.

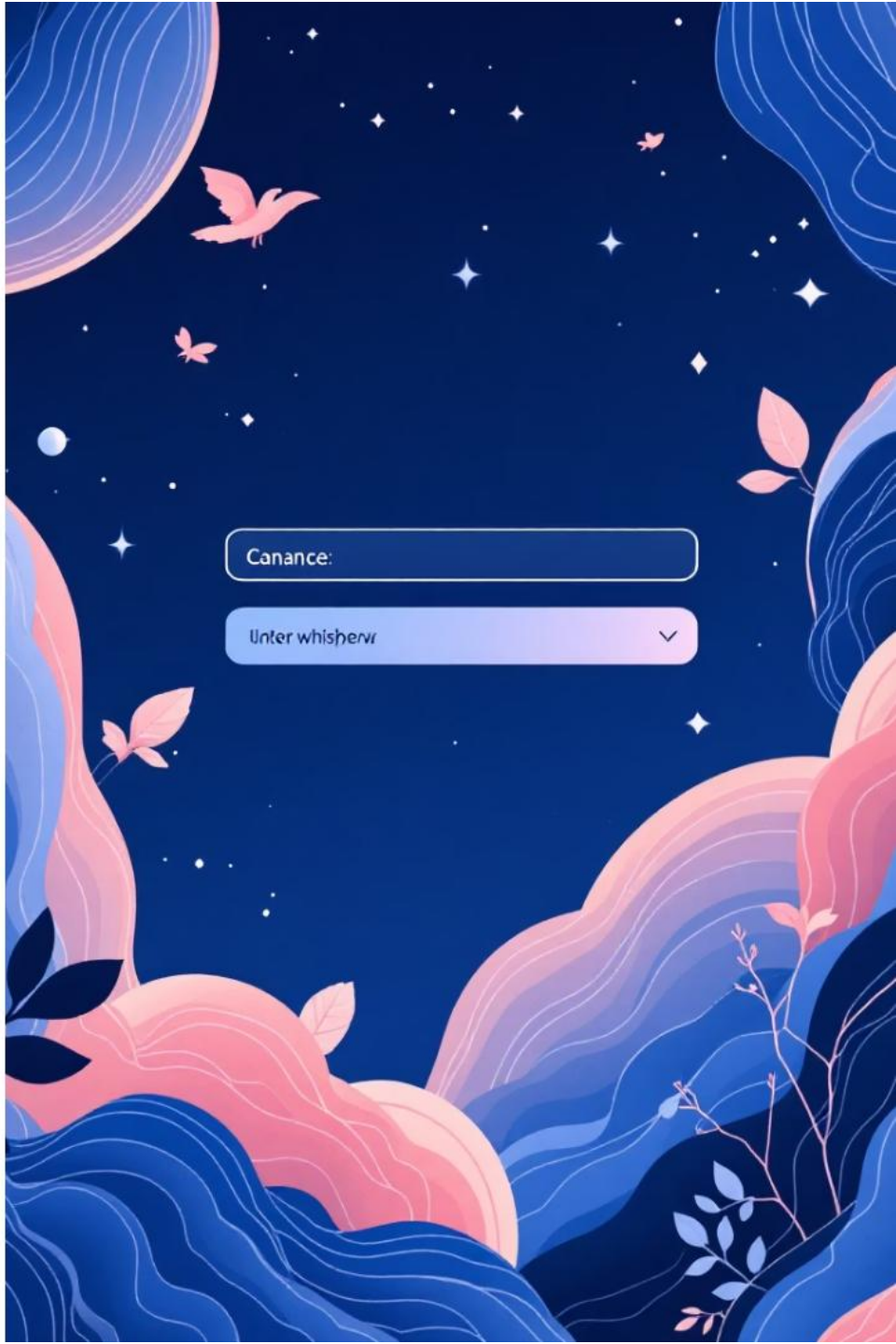
User Feedback System

Allows users to mark responses as helpful or unhelpful, providing valuable data for potential model refinement.

User Experience: Interface Design

The interface is designed for minimal cognitive load, prioritizing ease of navigation and clear output display.

- **Function Selector:** A dropdown menu to instantly toggle between Question, Summarize, or Creative modes.
- **Input Area:** A dedicated text box for the user to submit their query or content.
- **Response Display:** A dedicated, clear section where the AI-generated text is rendered.
- **Feedback:** Simple Yes/No buttons immediately follow the response for quality control.



Step-by-Step Usage Guide

01

Access the Application

Open the locally running application in your web browser: http://127.0.0.1:5000/

02

Select Function

Use the dropdown to choose the required mode: Question, Summarize, or Creative.

03

Enter Prompt

Type or paste your content or question into the input text area.

04

Generate Response

Click the **“Get Response”** button to send the prompt to the OpenAI API.

05

Review and Provide Feedback

Examine the generated output and optionally submit feedback on its quality.

Technical Implementation: Getting Started

Setting up the AI Assistant requires installing a few key packages and securely managing the API key.

1. Install Dependencies

```
pip install flask openai python-dotenv
```

These packages enable the web server, external API connectivity, and secure environment handling.

2. Secure API Key

Create a `.env` file in the root directory to store your secret key:

```
OPENAI_API_KEY=sk-your-openai-key
```

This prevents exposing sensitive credentials in the main code repository.

3. Run the App

```
python ai_assistant.py
```

The Flask server will start, making the application accessible on the local network.

4. Access Locally

View the functional application in your browser:

> http://127.0.0.1:5000/

Demonstration: Summarization Example

1

User Input (Mode: Summarize)

"Summarize the importance of AI in education."

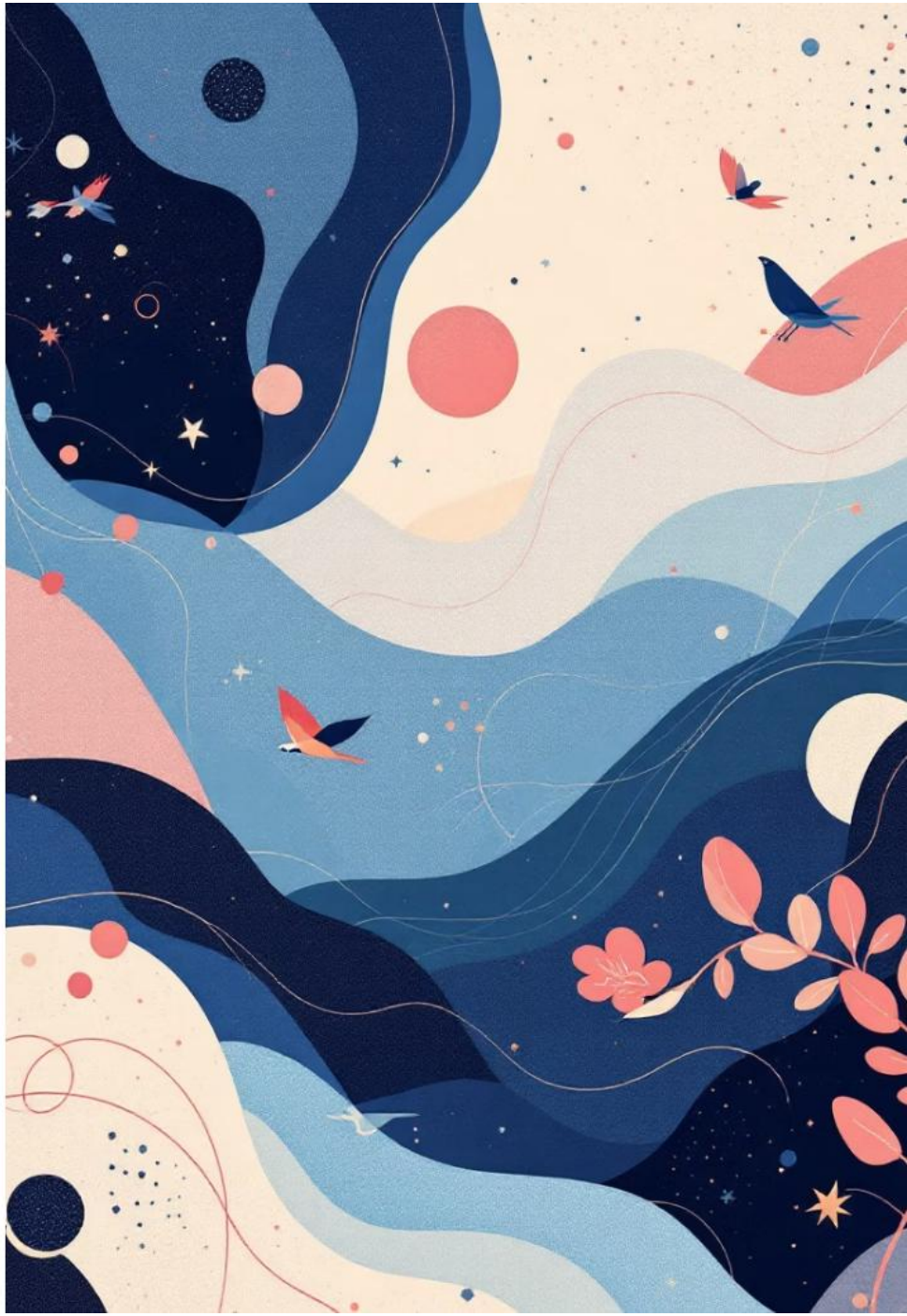
The AI receives this prompt and applies the pre-defined system message for summarization.

2

AI Generated Output

"AI enhances personalized learning, automates grading, and provides new teaching insights for better student outcomes."

The GPT model efficiently distills the core concepts into a concise, 15-word summary.



Conclusion and Future Scope

Core Success

Successfully integrated the power of OpenAI's LLM with a simple, secure Python Flask web application.

Key Functions

Proven capability across three vital tasks: Q&A, text summarization, and creative generation.

Next Steps

Future development could include full conversational chat, user authentication, and deployment to a live server (e.g., Heroku or AWS).

Resources for Further Study

- OpenAI Documentation: https://platform.openai.com/docs
- Flask Documentation: https://flask.palletsprojects.com