VIT®

BHOPAL

Name – Snehardhya Karmakar

Registration No. : 25BEC10063

Faculty Name – Dr. G Vishnuvarthanan

Class Slot – C21+F11+F12

Project Report: Instagram Caption Generator Web App

1. Project Overview

The Instagram Caption Generator is a Flask-based web application that automatically generates short, catchy captions for Instagram posts based on a user-provided description and selected mood. The app uses the OpenAI API to produce natural and engaging captions using AI language models (GPT-based models like gpt-3.5-turbo).

If an API key is not available, the app features a mock caption generator for offline testing, ensuring usability even without an internet connection or API access.

2. Objectives

- To design a user-friendly web application capable of generating creative Instagram captions using AI.
- To integrate Flask for building the backend web framework.
- To utilize OpenAI's Chat Completion API for generating intelligent captions.
- To implement a mock fallback system for offline usage or when API keys are missing.
- To handle exceptions gracefully (e.g., rate limit or quota errors).

3. Tools and Technologies Used

Technology / Library	Purpose
Python 3	Core programming language
Flask	Web application framework
OpenAI Python SDK	To interact with GPT-based AI models
dotenv (python-dotenv)	To securely load API keys from .env file
HTML (Jinja2 Template)	For rendering the web interface
Gunicorn / Flask built-in server	To run the application
Environment Variables (.env)	For secure configuration of the API key

4. System Architecture

The system follows a simple client-server model:

1. User Interface (Frontend)
   - The user inputs a post description and selects a mood.
   - Submits the form via POST request to the Flask backend.

2. Flask Backend
   - Reads form data from the request.
   - Constructs a prompt to guide the AI model.
   - Sends this prompt to OpenAI's Chat Completion API (if API key available).

3. OpenAI API / Mock Fallback
   - If API is available: the GPT model generates a caption.
   - If API is missing: the system uses generate_mock_caption() to simulate output.

4. Response Rendering
   - The generated (or mock) caption is sent back and displayed on the HTML page.

5. Key Code Components

Includes environment setup, AI client initialization, mock caption generator, route handling, API call, and exception handling.

6. Features

✅AI-powered caption generation using GPT models
✅Mock fallback for offline testing
✅Custom error messages for API rate limits or connectivity issues
✅Simple and clean Flask-based UI
✅Secure environment variable handling

7. Example Usage

Input:
- Description: "Enjoying coffee on a rainy morning ☕"
- Mood: "Romantic"

AI Output:
"Rain, coffee, and cozy vibes ☕ #PerfectMorning"

Mock Output (no API key):
"Romantic vibe: Enjoying coffee on a rainy morning ☕. #mockcaption"

8. Error Handling & Fallback

The application includes robust exception handling to detect and respond to API quota or rate limit issues, provide friendly fallback messages, and log errors without breaking user experience.

9. Security & Deployment

- API keys are stored securely using .env files.
- Flask debug mode is only enabled during local development.
- The app can be deployed using Gunicorn or Docker on platforms like Render, Heroku, AWS EC2, or Vercel.

10. Future Improvements

- Add user authentication to store caption history.
- Support multiple languages for captions.
- Integrate with Instagram API for direct posting.
- Add advanced mood-based templates.
- Provide caption length and hashtag control.

11. Conclusion

This project demonstrates a successful integration of Flask web development and OpenAI's generative AI capabilities to produce a practical, creative, and user-friendly application. It showcases strong backend design, exception handling, and clean code organization — suitable for real-world deployment and further expansion.

12. References

- Flask Documentation: https://flask.palletsprojects.com/
- OpenAI Python SDK: https://platform.openai.com/docs/api-reference
- Python-dotenv: https://pypi.org/project/python-dotenv/
- Jinja2 Templates: https://jinja.palletsprojects.com/