**Nutrition App Using Gemini Pro : Your Comprehensive Guide to Healthy Eating and Well-being"**

**Project Description**

Gemini Health App is an innovative mobile application designed to provide personalized dietary recommendations and nutritional advice using the advanced capabilities of the Gemini Pro model. The app leverages artificial intelligence to analyze user data, dietary preferences, and health goals, delivering tailored meal plans, nutritional insights, and wellness tips. The primary aim of Gemini Health App is to promote healthier eating habits and improve overall well-being through intelligent and data-driven recommendations.

**Scenario 1: Weight Loss Journey**

Sarah, a 28-year-old with a goal to lose 15 pounds, uses Gemini Health App to aid her in her weight loss journey. As a vegetarian with a moderate activity level, she inputs her dietary preferences and health goals into the app. Gemini Health App creates a calorie-controlled, nutrient-dense meal plan tailored to her vegetarian diet. Sarah logs her meals by taking photos or scanning barcodes, and the app provides feedback on her calorie intake and nutritional balance, suggesting necessary adjustments. By syncing her fitness tracker, the app integrates her physical activity data, offering comprehensive insights to help Sarah stay on track with her weight loss while maintaining proper nutrition.

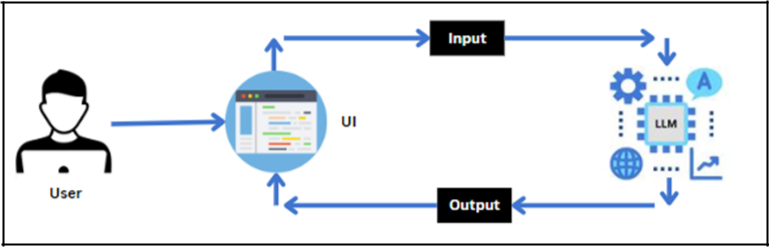
**Scenario 2: Managing Diabetes**

John, a 45-year-old with Type 2 Diabetes, relies on Gemini Health App to manage his condition through diet. He inputs his low-carb dietary preference and diabetes condition, and the app generates meal plans that focus on low carbohydrate and high fiber content to help control his blood sugar levels. John uses the app to log his meals, receiving immediate feedback on their suitability for his diabetes management. Detailed nutritional breakdowns highlight carbohydrate content and glycemic index, aiding John in making informed food choices. Additionally, the app provides educational resources about managing diabetes through diet, keeping John well-informed and empowered to handle his condition better.

**Scenario 3: Building the Muscle**

Emily, a 30-year-old strength training enthusiast, uses Gemini Health App to support her goal of gaining muscle mass. With a preference for high-protein meals and an intense workout regime, she inputs her dietary preferences and fitness goals into the app. Gemini Health App generates meal plans rich in protein and essential nutrients necessary for muscle growth. Emily benefits from a variety of high-protein recipes that cater to her needs, with each recipe including detailed instructions and nutritional information. By connecting her fitness tracker, the app accounts for her caloric expenditure and provide insights on balancing her protein intake with her workouts, optimizing her muscle-building efforts.

**Technical Architecture**



**Project Flow:**

* User interacts with the UI to enter the input.
* User input is collected from the UI and transmitted to the backend using the Google API key.
* The input is then forwarded to the Gemini Pro pre-trained model via an API call.
* The Gemini Pro pre-trained model processes the input and generates the output.
* The results are returned to the frontend for formatting and display.

**To accomplish this, we have to complete all the activities listed below:**

* **Requirements Specification**
  + Create a requirements.txt file to list the required libraries.
  + Install the required libraries
* **Initialization of Google API Key**
  + Generate Google API Key
  + Initialize Google API Key
* **Interfacing with Pre-trained Model**
  + Load the Gemini Pro pre-trained model
  + Implement a function to get gemini response
  + Implement a function to read PDF content
  + Write a prompt for gemini model
* **Model Deployment**
  + Integrate with Web Framework
  + Host the Application

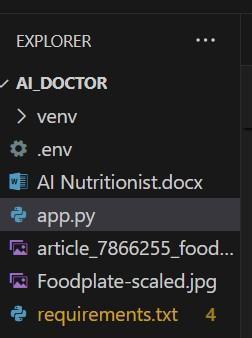
**Prior Knowledge:**

You must have the prior knowledge of the following topics to complete this project.

* Generative AI Concepts
* NLP: <https://www.tutorialspoint.com/natural_language_processing/index.htm>
* Generative AI: <https://en.wikipedia.org/wiki/Generative_artificial_intelligence>
* About Gemini: <https://deepmind.google/technologies/gemini/#introduction>
* Gemini API: <https://ai.google.dev/gemini-api/docs/get-started/python>
* Gemini Demo: [https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/](https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb) [en/gemini-api/docs/get-started/python.ipynb](https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb)
* Streamlit: <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>

**Project Structure**

Create the Project folder which contains files as shown below:

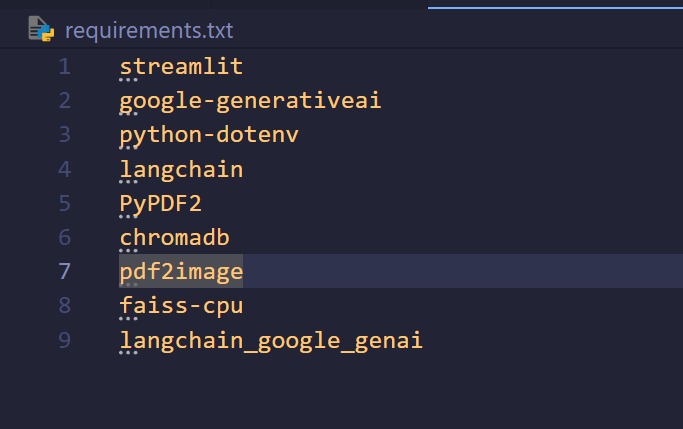


* images folder: It is established to store the images utilized in the user interface.
* .env file: It securely stores the Google API key.
* app.py: It serves as the primary application file housing both the model and Streamlit UI code.
* requirements.txt: It enumerates the libraries necessary for installation to ensure proper functioning.
* Additionally, ensure proper file organization and adhere to best practices for version control.

**Milestone 1: Requirements Specification**

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

**Activity 1: Create a requirements.txt file to list the required libraries.**



* streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
* streamlit\_extras: Additional utilities and enhancements for Streamlit applications.
* google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
* python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
* PyPDF2: It is a Python library for extracting text and manipulating PDF documents.
* Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.

**Activity 2: Install the required libraries**



* Open the terminal.
* Run the command: pip install -r requirements.txt
* This command installs all the libraries listed in the requirements.txt file.

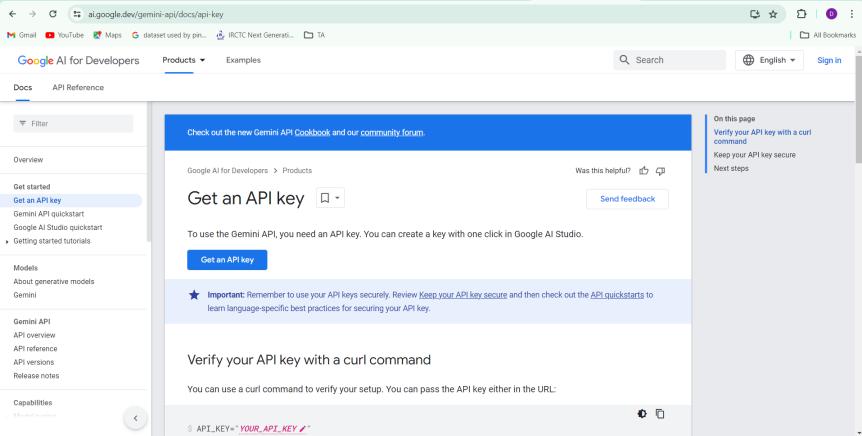
**Milestone 2: Initialization of Google API Key**

The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.

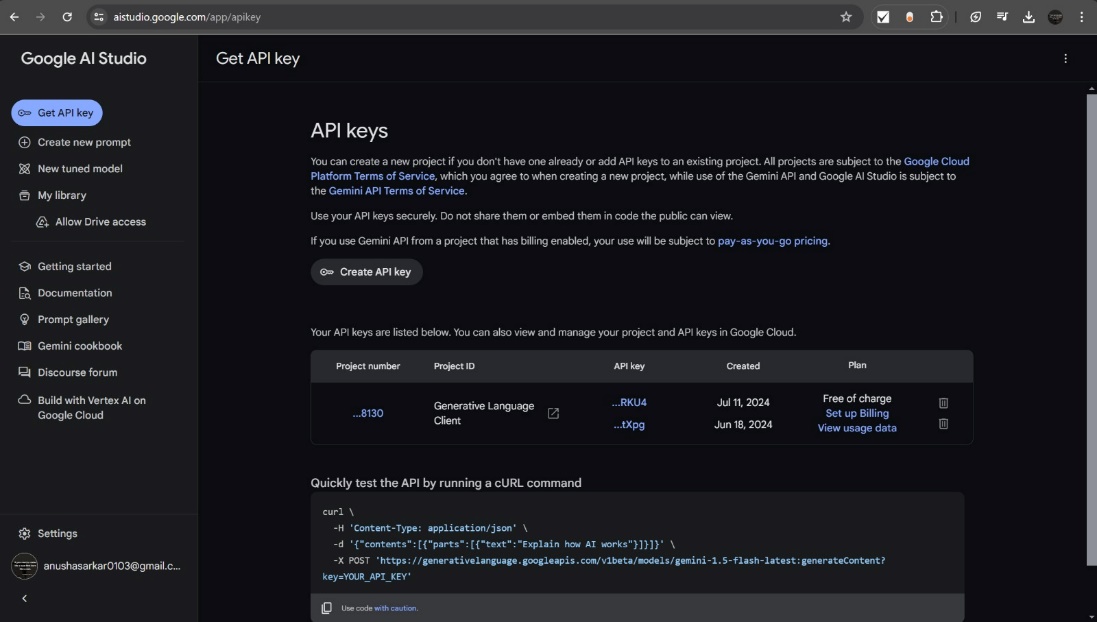
**Activity 1: Generate Google API Key**

Click the provided link to access the following webpage.

Link: <https://ai.google.dev/gemini-api/docs/api-key>

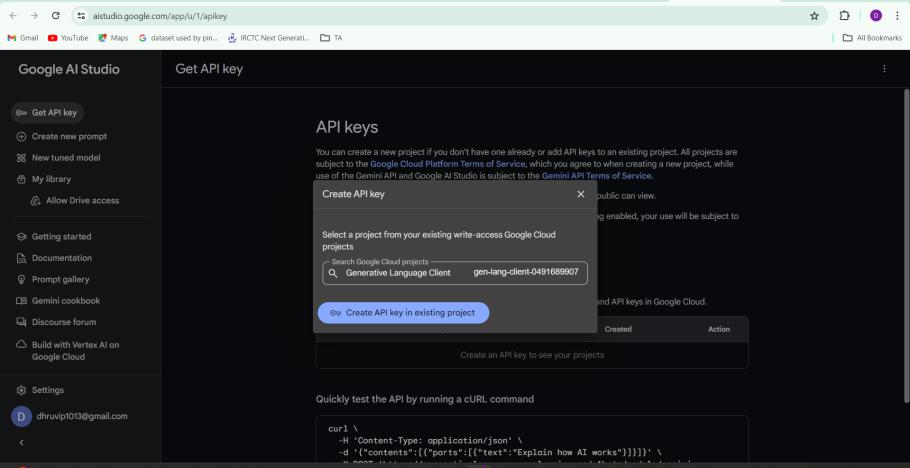


After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below.



Next, click on 'Create API Key' and choose the generative language client as the project.

Then, select 'Create API key in existing project'.



Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

**Activity 2: Initialize Google API Key**

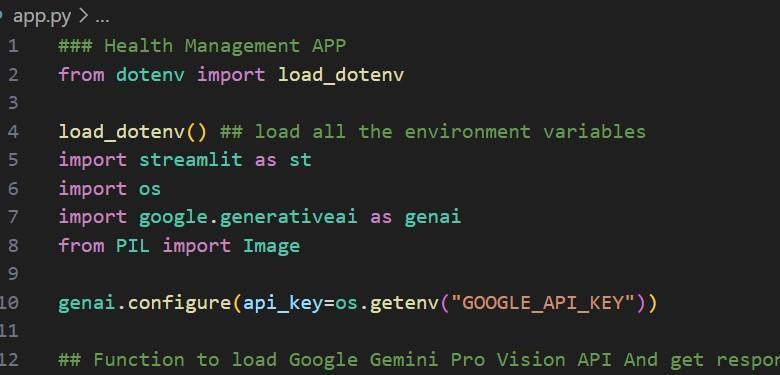


* Create a .env file and define a variable named GOOGLE\_API\_KEY.
* Assign the copied Google API key to this variable.
* Paste the API key obtained from the previous steps here.

**Milestone 3: Interfacing with Pre-trained Model**

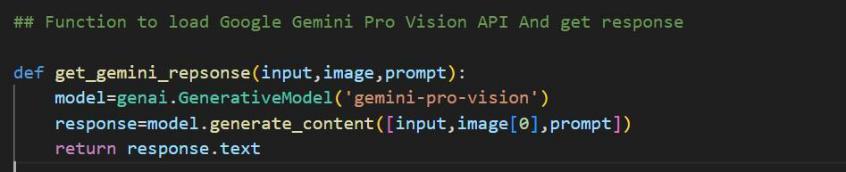
To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

**Activity 1: Load the Gemini Pro API**



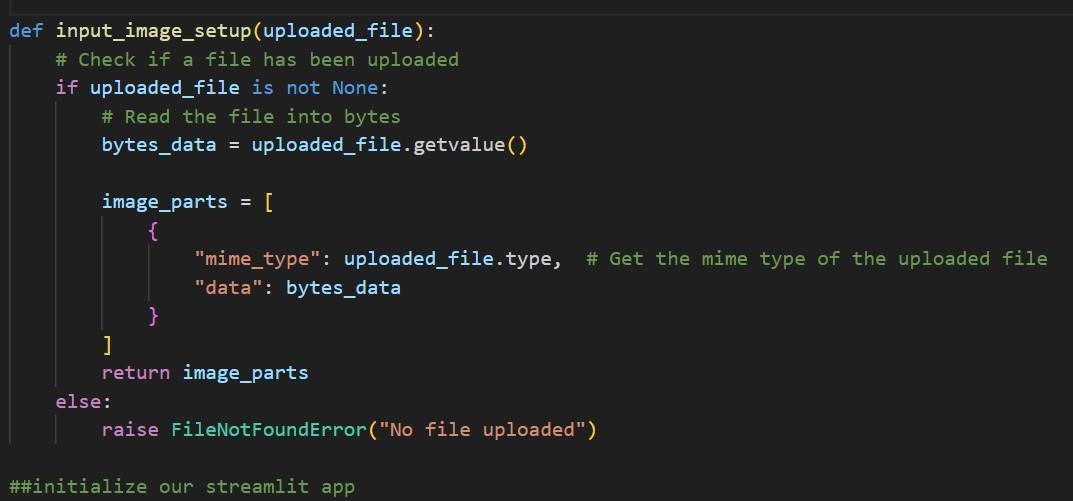
This code snippet is for initializing a health management application using Streamlit, an open-source app framework, and Google Generative AI services. The script starts by loading environment variables from a .env file using the load\_dotenv() function from the dotenv package. It then imports necessary libraries: streamlit for creating the web app interface, os for accessing environment variables, google.generativeai for utilizing Google's Generative AI capabilities, and PIL.Image for image processing. The genai.configure() function is called to set up the Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services.

**Activity 2: Implement a function to get gemini response**



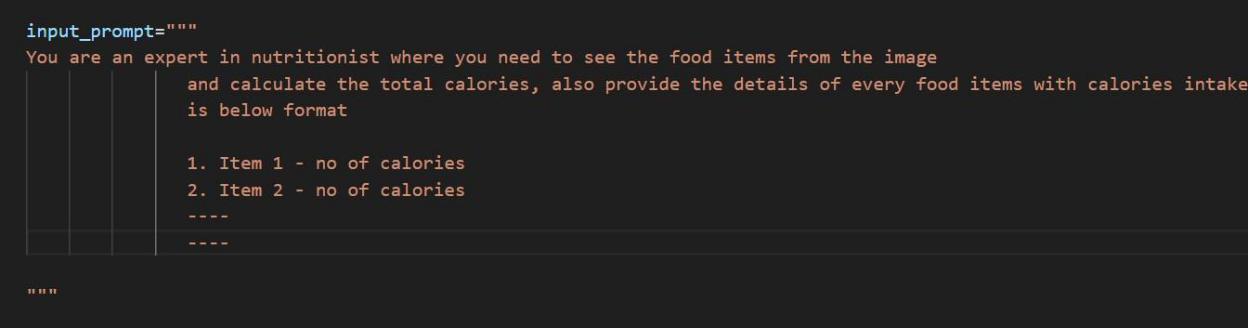
* The function get\_gemini\_response takes an input text as a parameter.
* It calls the generate\_content method of the model object to generate a response.
* The generated response is returned as text.

**Activity 3: Implement a function to read the Image and set the image format for Gemini Pro model Input**



The function input\_image\_setup processes an uploaded image file for a health management application. It first checks if a file has been uploaded. If a file is present, it reads the file's content into bytes and creates a dictionary containing the file's MIME type and its byte data. This dictionary is then stored in a list named image\_parts, which is returned by the function. If no file is uploaded, the function raises a FileNotFoundError, indicating that an image file is required but not provided. This setup ensures that the uploaded image is correctly formatted and ready for further processing or analysis in the application.

**Activity 4: Write a prompt for gemini model**



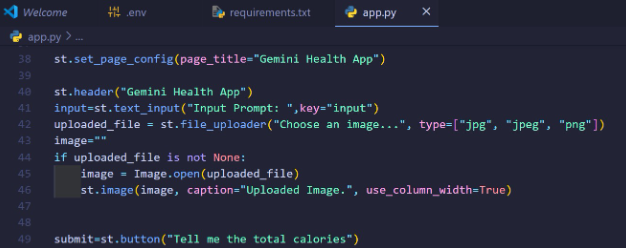
The variable input\_prompt is a multi-line string designed as a prompt for a Gemini Health App model. It instructs the model to analyze an image of food items, identify each food item, and calculate the total calories. Additionally, the model is to provide a detailed breakdown of each food item with its respective calorie count. The expected output format is a numbered list where each item is listed alongside its calorie content, ensuring clarity and structured information for the user. This prompt is likely used in conjunction with an AI service that can process images and generate nutritional information based on the visual data provided.

**Milestone 4: Model Deployment**

We deploy our model using the Streamlit framework, a powerful tool for building and sharing data applications quickly and easily. With Streamlit, we can create interactive web applications that allow users to interact with our models in real-time, providing an intuitive and seamless experience.

**Activity 1: Integrate with Web Framework**

Gemini Health App:



* If “Tell me the total calories button click”:



●

This code initializes a Streamlit application titled "Gemini Health App" by setting the page title and creating the app's header. It includes a text input field for users to enter a custom prompt and a file uploader for users to upload an image in JPG, JPEG, or PNG format. If an image is uploaded, it is opened using the PIL library and displayed within the app with a caption. A button labeled "Tell me the total calories" is also provided, which users can click to trigger the application's functionality for analyzing the uploaded image to calculate and display the total calorie content of the food items depicted.

**Activity 2: Host the Application**

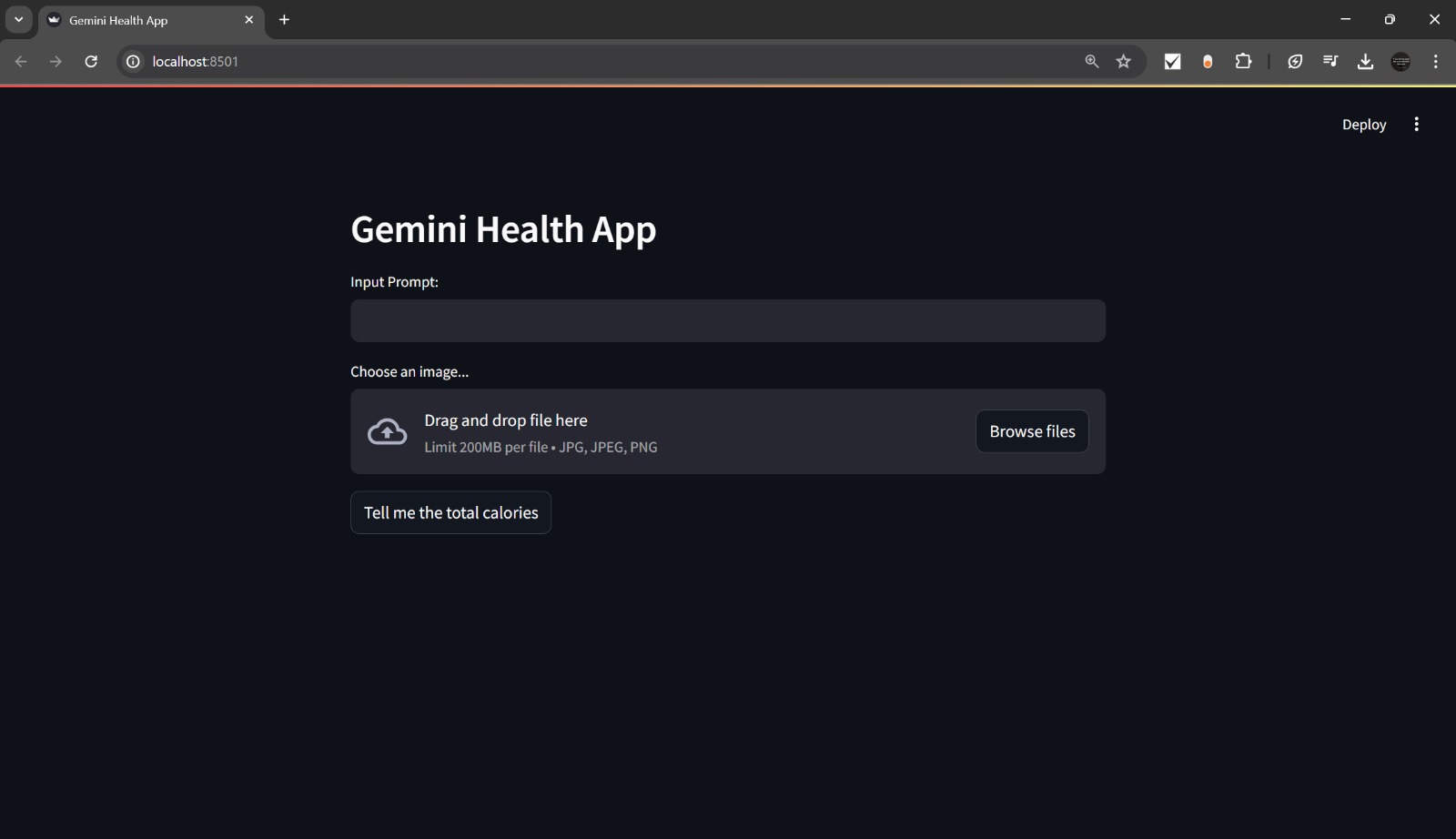
Launching the Application:

* To host the application, go to the terminal, type - streamlit run app.py
* Here app.py refers to a python script.

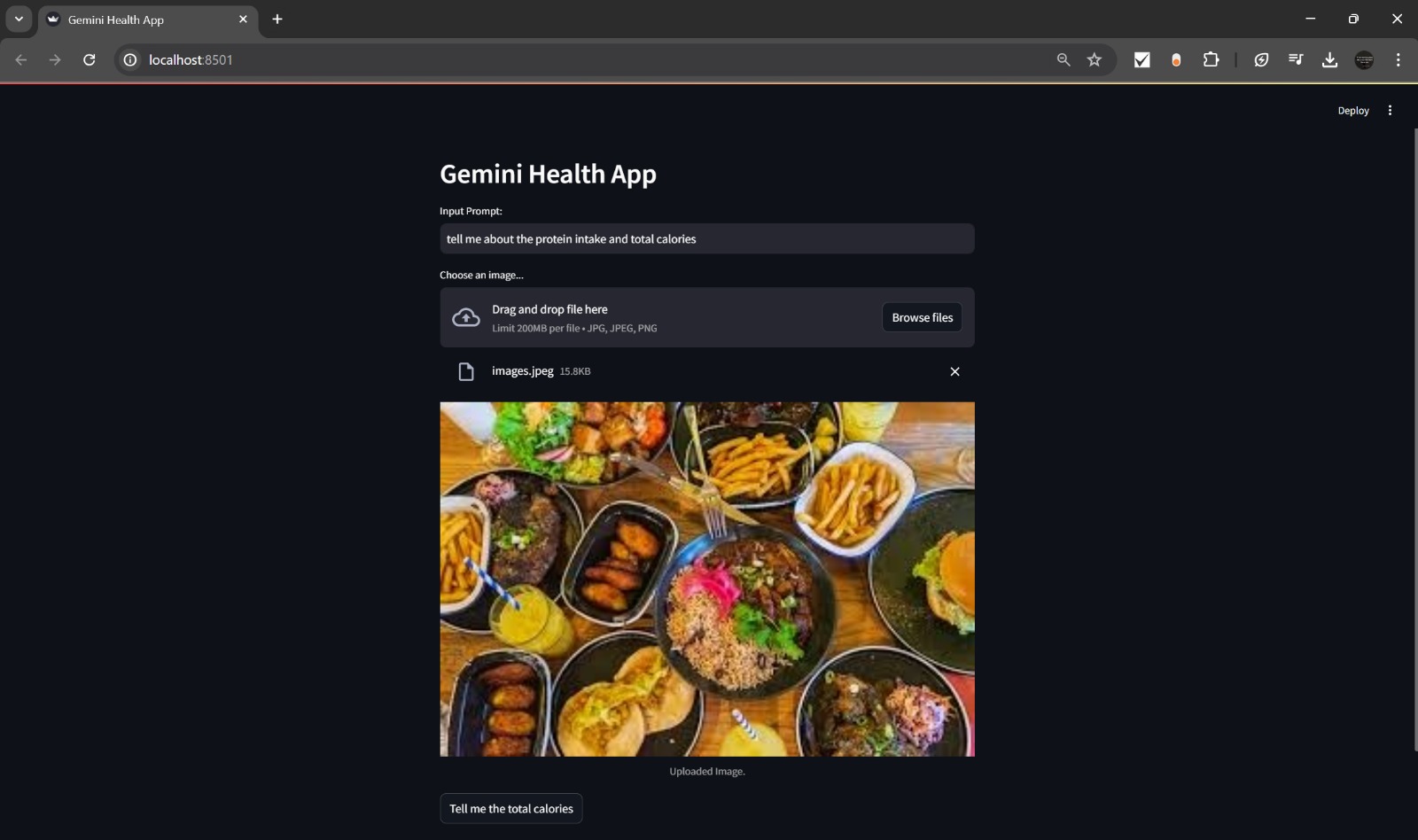




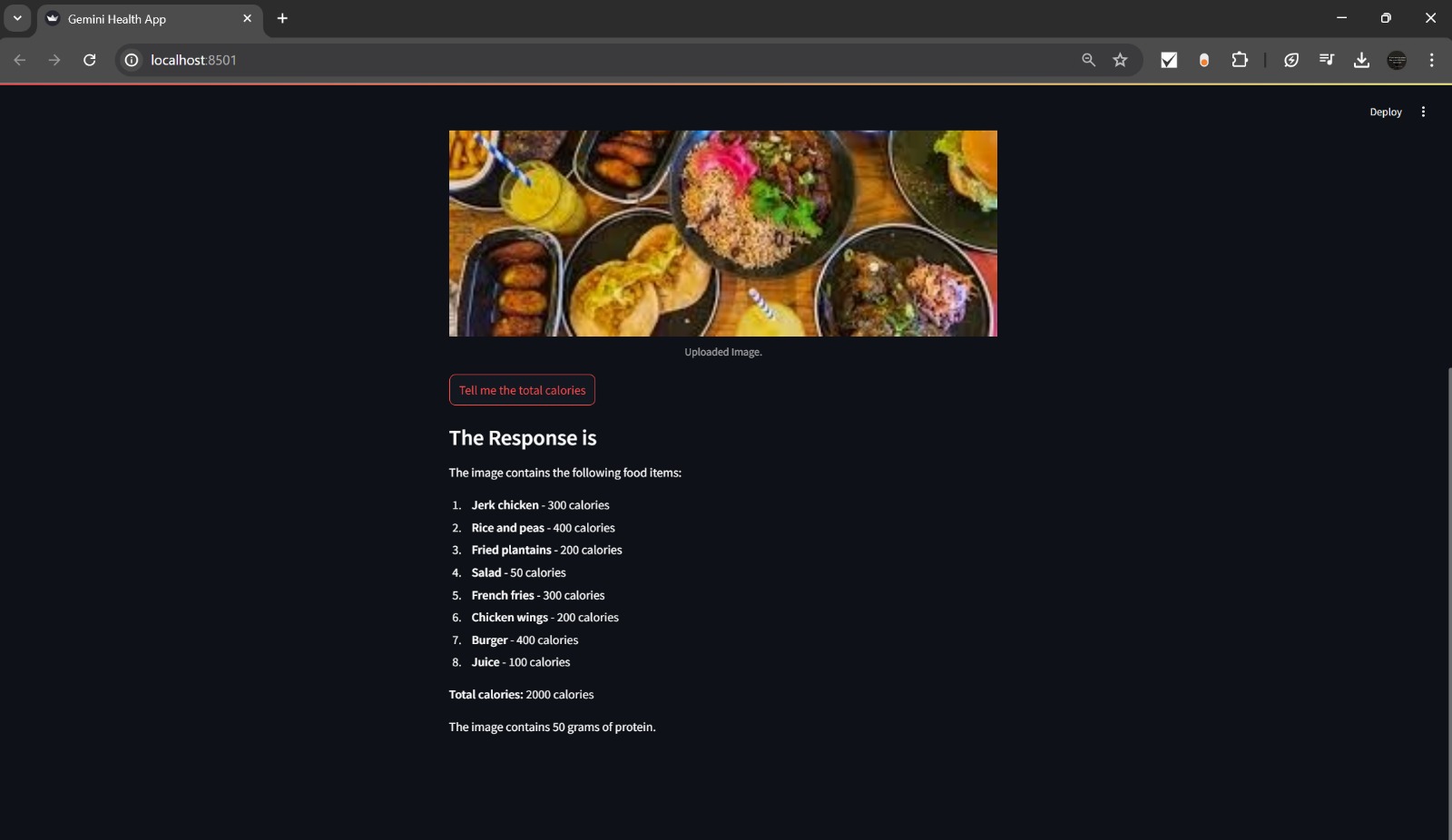
Run the command to get the below results



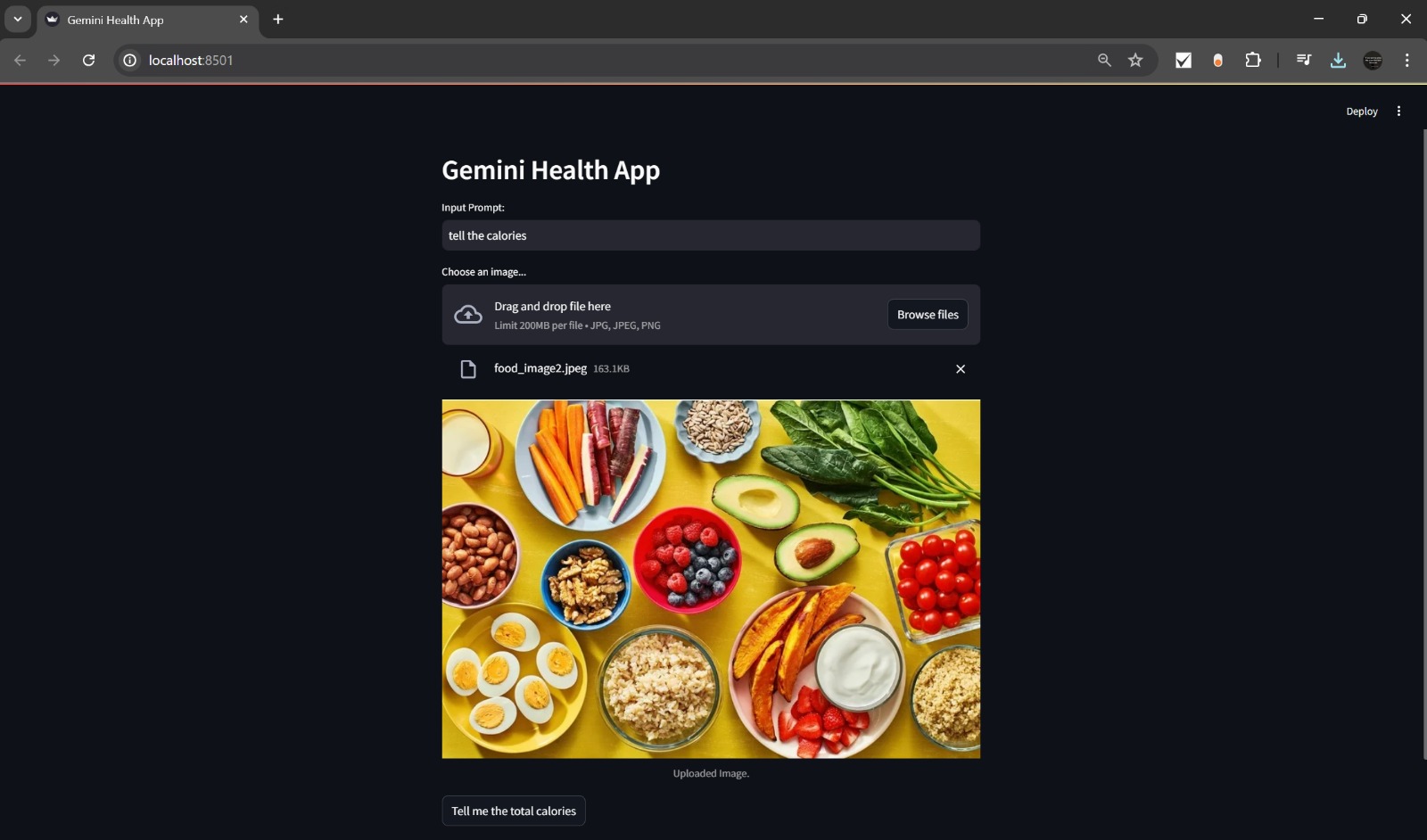
Input 1:



Output 1:



Input 2:



Output 2:

