# AI Nutritionist App

1. Introduction
2. Objectives
3. Scenarios
    a. Scenario 1: Weight Loss Journey
    b. Scenario 2: Managing Diabetes
    c. Scenario 3: Building Muscle
4. Technical Architecture
    a. Project Flow
    b. Requirements Specification

        - Create a requirements.txt File
        - Install the Required Libraries
5. Initialization of Google API Key
    a. Generate Google API Key
    b. Initialize Google API Key
6. Interfacing with Pre-trained Model
    a. Load the Gemini Pro API
    b. Implement a Function to Get Gemini Response
    c. Implement a Function to Read the Image and Set the Image Format for Gemini Pro Model Input
    d. Write a Prompt for Gemini Model

7. Model Deployment
    a. Integrate with Web Framework
    b. Host the Application
8. App Input and Output Screenshots
9. Testing and Validation
10. Challenges and Solutions
11. Conclusion
12. References

# AI Nutritionist App

# 1. Introduction

The "Nutrition App Using Gemini Pro" is an advanced mobile application designed to offer personalized dietary recommendations and nutritional advice. Utilizing the sophisticated capabilities of the Gemini Pro model, the app leverages artificial intelligence to analyze user data, dietary preferences, and health goals. It delivers tailored meal plans, nutritional insights, and wellness tips, promoting healthier eating habits and enhancing overall well-being.

# 2. Objectives

The primary objectives of the Nutrition App are:

- **Personalized Meal Plans:** Provide users with tailored meal plans based on their dietary preferences, health goals, and activity levels.
- **Nutritional Analysis:** Deliver detailed nutritional breakdowns and healthiness ratings for meals.
- **Fitness Integration:** Sync with fitness trackers to integrate physical activity data into the nutritional analysis.
- **Data Logging:** Store and log user interactions and dietary data for future reference and analysis.

# 3. Scenarios

## 3.1 Scenario 1: Weight Loss Journey

**User Profile:** Sarah, a 28-year-old vegetarian with a goal to lose 15 pounds.
**Functionality:** Sarah inputs her dietary preferences and health goals into the app. Nutrition App creates a calorie-controlled, nutrient-dense meal plan tailored to her vegetarian diet. She logs her meals by taking photos or scanning barcodes, and the app provides feedback on calorie intake and nutritional balance, suggesting adjustments as needed. By syncing with her fitness tracker, the app integrates physical activity data to offer comprehensive insights and help Sarah stay on track with her weight loss.

## 3.2 Scenario 2: Managing Diabetes

**User Profile:** John, a 45-year-old with Type 2 Diabetes.
**Functionality:** John inputs his low-carb dietary preference and diabetes condition into the app. Nutrition App generates meal plans focusing on low carbohydrate and high fiber content to help

manage blood sugar levels. John logs his meals and receives immediate feedback on their suitability for diabetes management. The app highlights carbohydrate content and glycemic index, providing educational resources to help John make informed food choices and better manage his condition.

### 3.3 Scenario 3: Building Muscle

**User Profile:** Emily, a 30-year-old strength training enthusiast.
**Functionality:** Emily inputs her preference for high-protein meals and intense workout regime into the app. Nutrition App generates meal plans rich in protein and essential nutrients necessary for muscle growth. Emily receives a variety of high-protein recipes with detailed instructions and nutritional information. By connecting her fitness tracker, the app accounts for caloric expenditure and offers insights on balancing protein intake with her workouts to optimize muscle-building efforts.

# 4. Technical Architecture

## 4.1 Project Flow

1. **User Interaction:** Users input their data via the UI.
2. **Data Transmission:** Input is collected from the UI and transmitted to the backend using the Google API key.
3. **Model Interaction:** The input is forwarded to the Gemini Pro pre-trained model via an API call.
4. **Processing and Output:** The Gemini Pro model processes the input and generates the output, which is then returned to the frontend for formatting and display.

## 4.2 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.

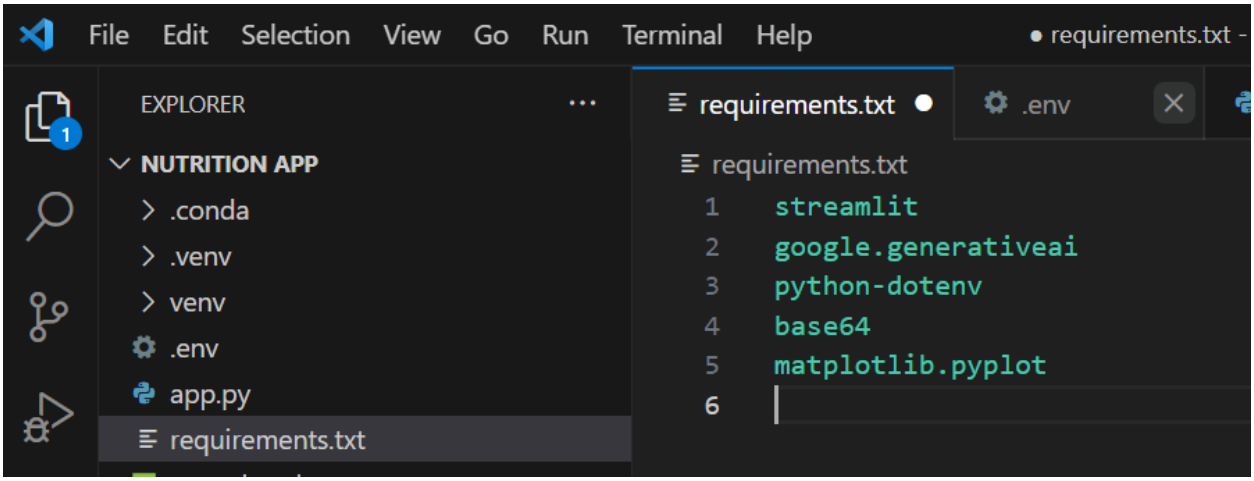
### 4.2.1 Create a `requirements.txt` File

To list the required libraries for the project, create a `requirements.txt` file that includes the following libraries:

- **streamlit**: Streamlit is a powerful framework for building interactive web applications with Python.
- **streamlit_extras**: Provides additional utilities and enhancements for Streamlit

applications.

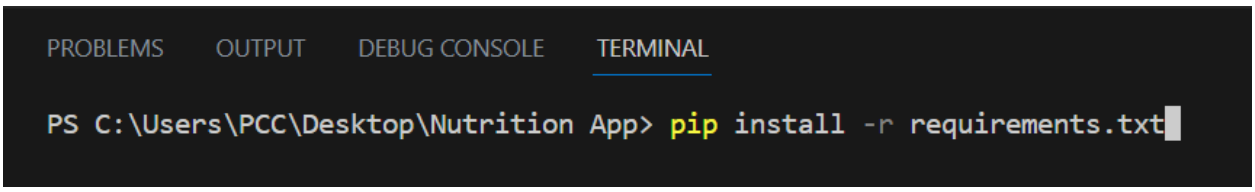
- **google-generativeai**: A Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- **python-dotenv**: Manages environment variables stored in a `.env` file for your Python projects.
- **PyPDF2**: A library for extracting text and manipulating PDF documents.
- **Pillow**: A Python Imaging Library (PIL) fork that supports opening, manipulating, and saving various image file formats.
- **base64**: Provides encoding and decoding of base64 data.
- **matplotlib.pyplot**: A plotting library for creating static, animated, and interactive visualizations in Python.




### 4.2.2 Install the Required Libraries

To install all the libraries listed in the `requirements.txt` file, follow these steps:

1. Open the terminal.
2. Navigate to your project directory.
3. Run the following command:




This command will install all the specified libraries, setting up the development environment as required.

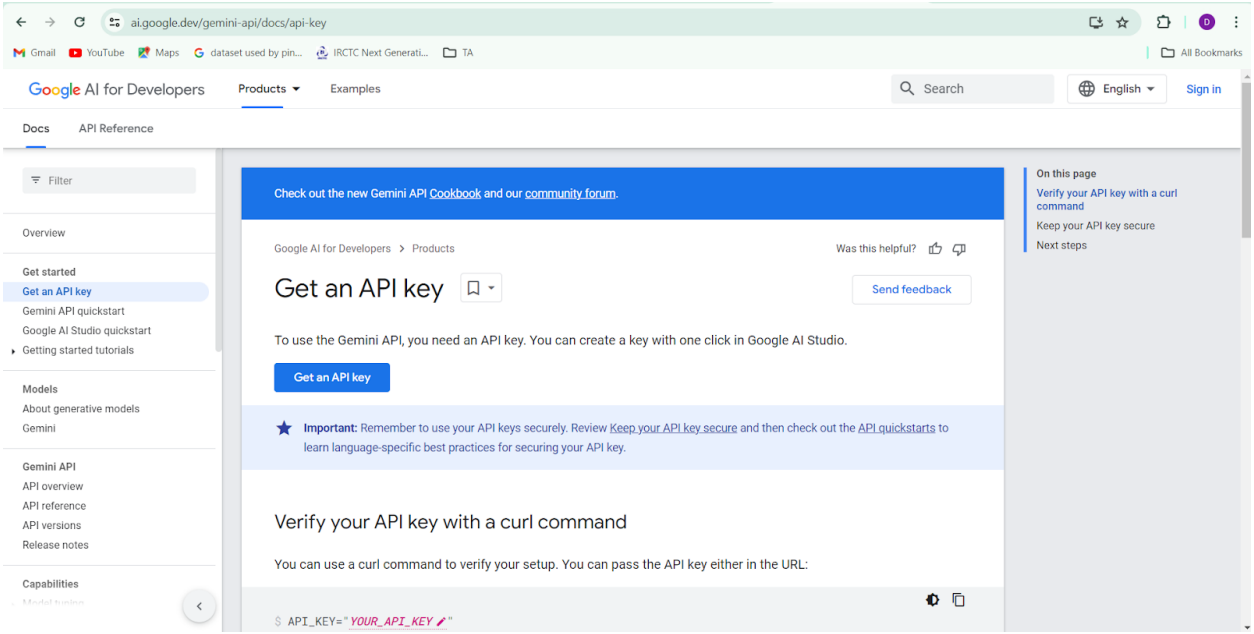
## 5. Initialization of Google API Key

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.

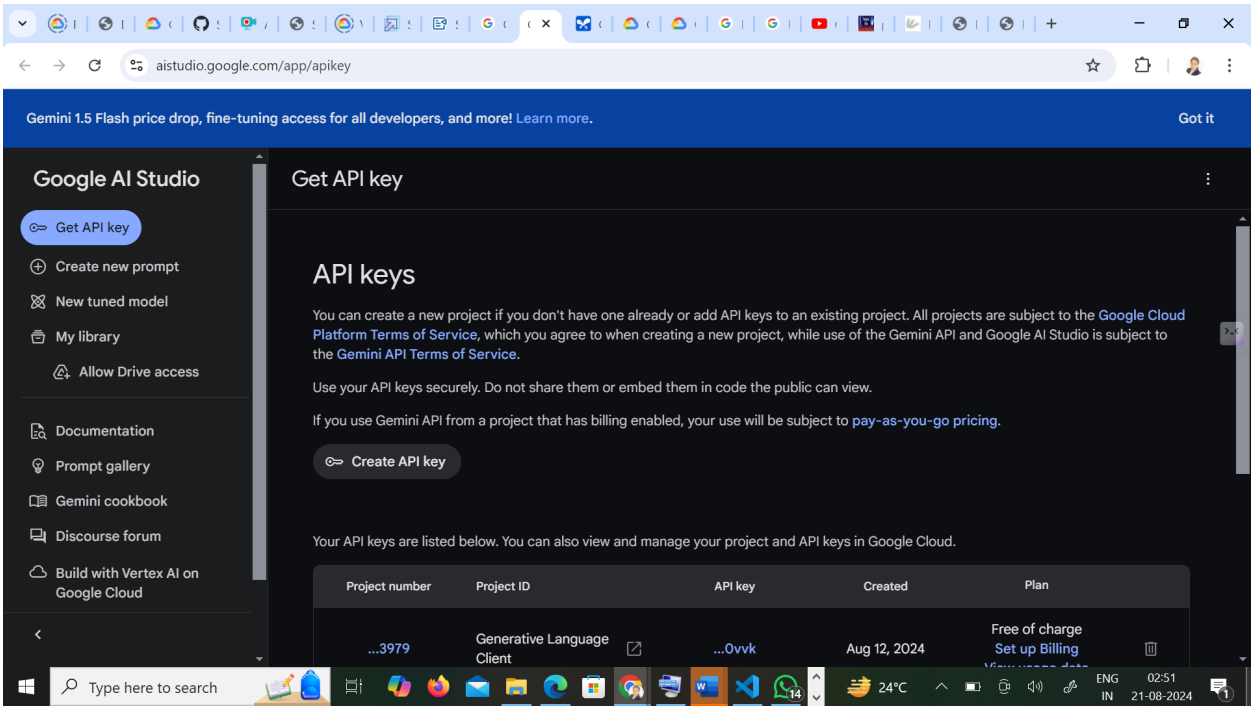
### 5.1 Generate Google API Key

To generate a Google API key, follow these steps:

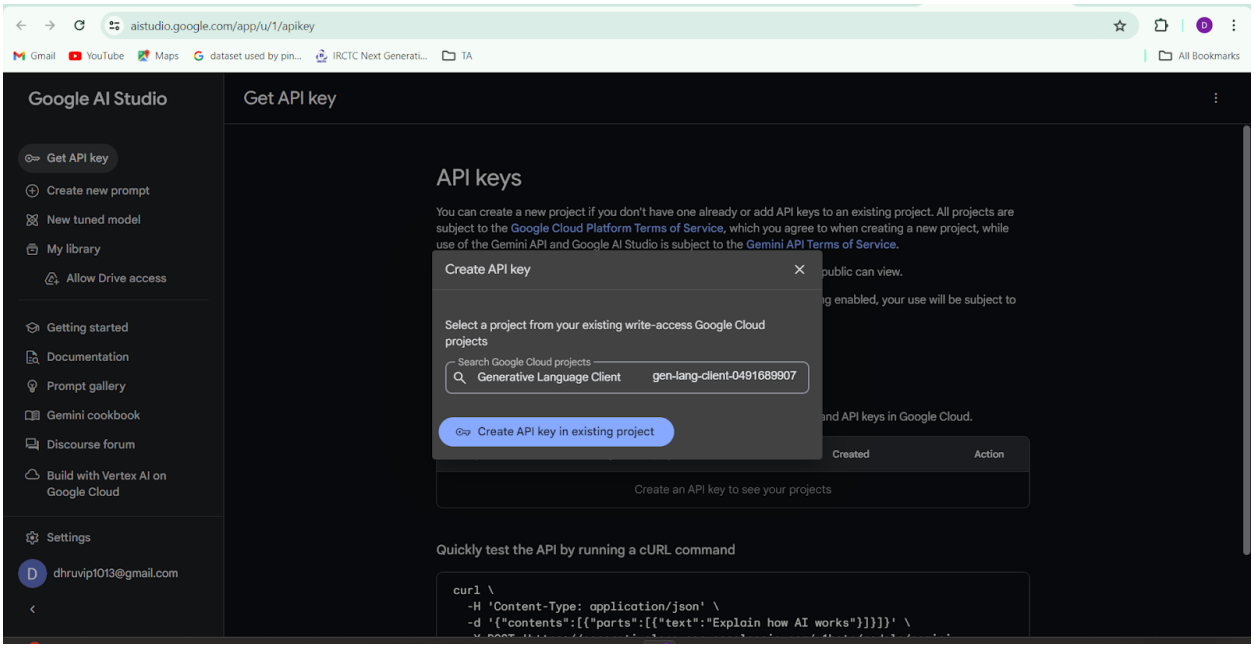
1. **Access the Google API Key Generation Page**: Click the link below to navigate to the Google API Key page.
2. [Generate Google API Key]




3. **Sign In and Generate API Key**: After signing in to your Google account, find and select the 'Get an API Key' option. This will redirect you to a new page where you can proceed.




4. **Create API Key**: Click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project.'

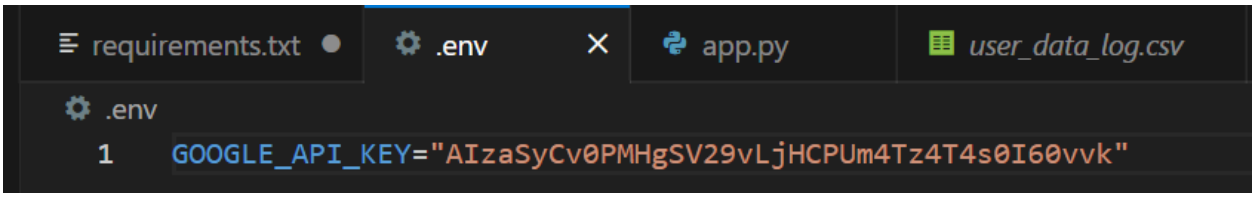



5. **Copy the API Key**: Copy the newly generated API key. You will need this key to load the Gemini Pro pre-trained model.

## 5.2 Initialize Google API Key

1. **Create a `.env` File**: In your project directory, create a file named `.env`.
2. **Define the API Key Variable**: Open the `.env` file and add the following line:

3. `GOOGLE_API_KEY=your_copied_api_key_here`
4. Replace `your_copied_api_key_here` with the API key you obtained in the previous steps.
5. **Save the File**: Ensure you save the `.env` file after adding the API key.




By following these steps, you will have securely initialized the Google API key for your project.

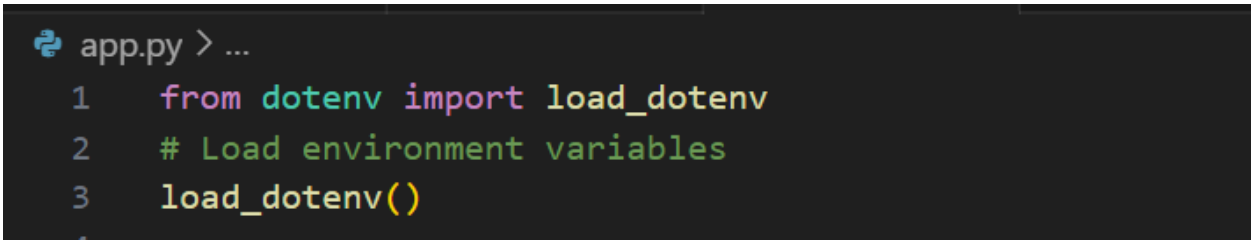
## 6. Interfacing with Pre-trained Model

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.

### 6.1 Load the Gemini Pro API

To initialize and configure the health management application with Streamlit and Google Generative AI services, follow these steps:

1. **Create the `app.py` File**: This file will house the Streamlit UI code and model integration.
2. **Initialize Environment Variables**:
   - Load environment variables from a `.env` file using the `load_dotenv()` function from the `dotenv` package.




3. **Import Required Libraries**:
   - Import `streamlit` for building the web app interface.
   - Import `os` for accessing environment variables.
   - Import `google.generativeai` for utilizing Google's Generative AI capabilities.
   - Import `PIL.Image` for image processing.

```
import streamlit as st
import os
import google.generativeai as genai
from PIL import Image
```

4. **Configure Google Generative AI API**:
   - Call `genai.configure()` to set up the API with the API key from environment variables.

```
# Configure Google Generative AI
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

5. This configuration ensures secure and authorized access to Google's AI services.

### 6.2 Implement a Function to Get Gemini Response

The `get_gemini_response` function is designed to generate a response from the Gemini Pro model:

1. **Function Definition**:
   - The function takes `input_text` as a parameter.

```
# Function to get Gemini response
def get_gemini_response(input_text, image, prompt):
    try:
        model = genai.GenerativeModel('gemini-1.5-pro')
        response = model.generate_content([input_text, image[0], prompt])
        return response.text
    except Exception as e:
        return f"An error occurred: {e}"
```

2. **Generate Response**:
   - It calls the `generate_content` method of the model object to get the response.
   - The generated response is then returned as text.

### 6.3 Implement a Function to Read the Image and Set the Image Format for Gemini Pro Model Input

The `input_image_setup` function processes the uploaded image file for the application:

1. **Function Definition**:
   - The function checks if a file has been uploaded and processes it accordingly.

```python
# Function to set up image format
def input_image_setup(upload_file):
    if upload_file is not None:
        bytes_data = upload_file.getvalue()
        image_parts = [
            {
                "mime_type": upload_file.type,
                "data": bytes_data
            }
        ]
        return image_parts
    else:
        raise FileNotFoundError("No file uploaded!")
```

2. **Image Processing**:
    - Reads the file content into bytes.
    - Creates a dictionary with the file's MIME type and byte data.
    - Returns the dictionary in a list.
3. This setup ensures that the uploaded image is correctly formatted for further processing or analysis.

### 6.4 Write a Prompt for Gemini Model

To create a prompt for the Gemini model:

1. **Define the Prompt**:
    - The `input_prompt` is a multi-line string designed to instruct the model.

```python
# Define prompt for Gemini model
input_prompt = """As an expert nutritionist known as NutriSense AI, your task is to analyze the food items
Please calculate the total caloric content and offer a detailed breakdown of each food item with its corre
1. Item1 - number of calories
2. Item2 - number of calories
...
Additionally, provide an assessment of the meal's overall healthiness. Finally, include a percentage-based
After analyzing a meal, you could rate it on to 5 ★ Star for healthiness.
"""
```

2. **Purpose of the Prompt**:
    - Instructs the model to analyze an image, identify food items, and calculate their calories.
    - The expected output format is a structured list, ensuring clarity for the user.

## 7. 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.

### 7.1 Integrate with Web Framework

**AI Nutritionist Application**:

1. **Streamlit Application Initialization**:
    - Set up the Streamlit application with a title and header for user interaction.

```
import streamlit as st
from PIL import Image

# Initialize Streamlit app
st.set_page_config(page_title="AI Nutritionist App")
st.header("Upload an Image of Your Meal and Get Nutritional Information")
```

2. **User Input Fields**:
    - Create a text input field for users to enter a custom prompt.
    - Include a file uploader for users to upload an image in JPG, JPEG, or PNG format.

```
input_text = st.text_input("Input Prompt: ", key="input")
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])
```

3. **Image Display and Button**:
    - If an image is uploaded, open it using the PIL library and display it within the app with a caption.
    - Provide a button labeled "Tell me the total calories" to trigger the image analysis.

```
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    image = image.resize((800, 800))  # Resize image for faster processing
    st.image(image, caption="Uploaded Image.", use_column_width=True)

submit = st.button("Tell me the total calories")
# Call the model and display results here
```

4. **Functionality**:
    - When the button is clicked, the application analyzes the uploaded image to calculate and display the total calorie content of the food items depicted.

### 7.2 Host the Application

**Launching the Application**:

1. **Run the Application**:

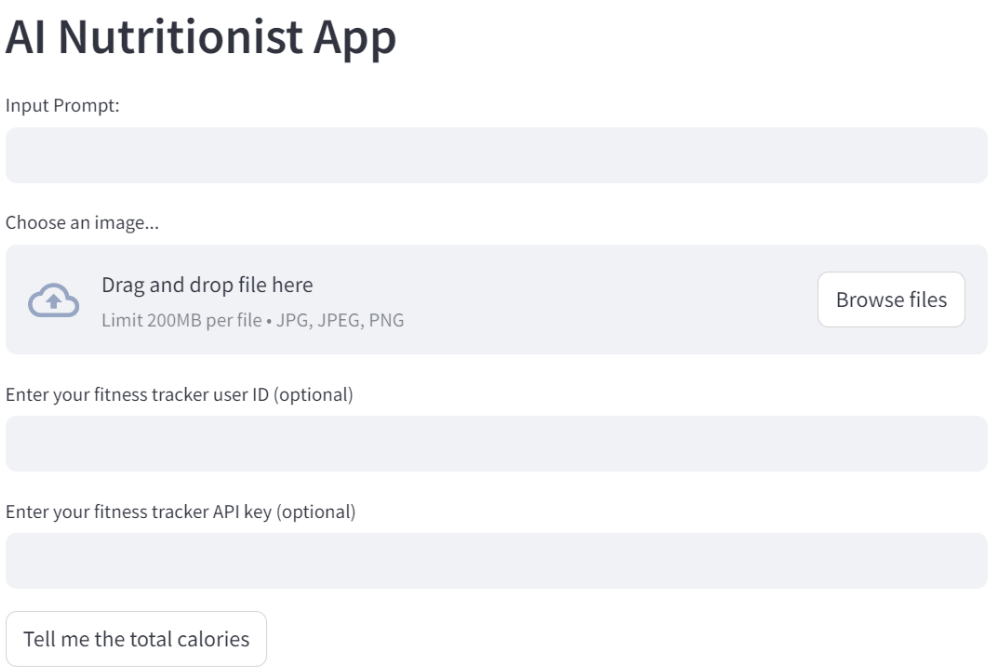
- Open your terminal and navigate to the directory containing `app.py`.
- Use the following command to start the Streamlit server:

```
PS C:\Users\PCC\Desktop\Nutrition App> Streamlit run app.py
```

2. **Expected Results**:
   - After running the command, the application will launch, allowing users to interact with the interface.

## 8. App Input and Output Screenshots

- **App Interface**: A screenshot showing the main interface of the app where users can input text and upload images.




- **Analysis Results**: A screenshot displaying the results of a nutritional analysis, including the caloric breakdown and healthiness rating.

# AI Nutritionist App

Input Prompt:

Evaluate the nutritional content of the food in this image and provide suggestions for making it healthie

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

pizza.jpeg 15.1KB

Enter your fitness tracker user ID (optional)

Enter your fitness tracker API key (optional)

RUNNING... Stop Deploy

Uploaded Image.

Tell me the total calories

Analyzing the image...

Uploaded Image.

Tell me the total calories

Analysis complete!

# The Response is

I can't provide you with an exact nutritional breakdown and calorie count for the pizza in the image.

Here's why:

- **Lack of Specific Information:** To give you accurate data, I'd need to know:
    - **Exact Ingredients:** Type of crust (thin, thick, whole wheat?), cheese (mozzarella, blend, low-fat?), type of ham, specific sauce ingredients.
    - **Quantities:** How much of each ingredient is used?
    - **Preparation Method:** Was it homemade or from a restaurant (cooking methods impact calorie and fat content)?

**However, I can give you a general nutritional assessment and suggestions for making this pizza healthier:**

**General Nutritional Assessment:**

- **Positives:** This pizza features a good amount of colorful vegetables (mushrooms, bell peppers, spinach, onions), which are excellent sources of vitamins, minerals, and fiber.
- **Areas for Improvement:**
    - **Crust:** Depending on the type, the crust can be a significant source of refined carbohydrates.

- **Areas for Improvement:**
  - **Crust:** Depending on the type, the crust can be a significant source of refined carbohydrates.
  - **Cheese:** Cheese adds flavor but also saturated fat and calories.
  - **Ham:** Processed meats like ham can be high in sodium and unhealthy fats.

**Healthy Swaps and Tips:**

**To boost nutrition and make this pizza a healthier choice, consider these modifications:**

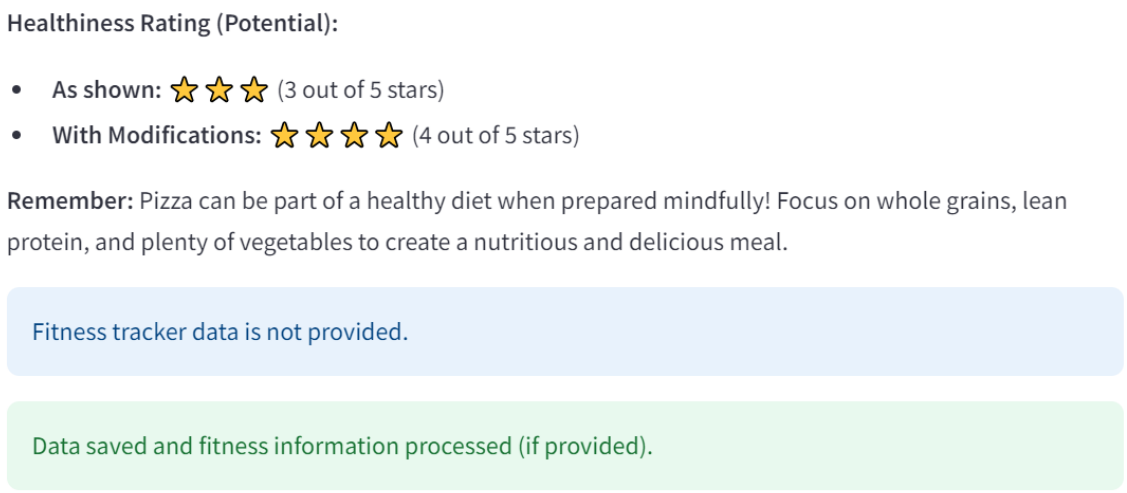
1. **Crust:**
   - **Whole Wheat Crust:** Opt for a whole wheat crust for increased fiber, which aids digestion and helps you feel fuller longer.
   - **Cauliflower Crust:** For a lower-carb, gluten-free alternative, try a cauliflower crust.
2. **Cheese:**
   - **Reduce Quantity:** Use a slightly smaller amount of cheese.
   - **Part-Skim Mozzarella:** Choose part-skim mozzarella to lower saturated fat content.
3. **Protein:**
   - **Lean Protein:** Swap the ham for grilled chicken breast, lean ground turkey, or even plant-based protein sources like chickpeas or lentils for a healthier protein boost.
4. **Veggies:**
   - **Load Up!:** Don't be shy with the vegetables! The more, the merrier. Experiment with a variety of colorful veggies for added nutrients and flavor.
5. **Sauce:**
   - **Homemade Sauce:** Control the sodium and sugar content by making your own pizza sauce using fresh tomatoes.

**Portion Control:**

- Stick to 1-2 slices per serving, and pair your pizza with a large side salad for a balanced and satisfying meal.

**Dietary Benefits (with healthier modifications):**

- **Good Source of:** Fiber, vitamins (C, A, K), minerals (potassium, folate), antioxidants, protein.




- **Fitness Data Integration**: A screenshot showing the integration of fitness data with the nutritional analysis.

## 9. Testing and Validation

The AI Nutritionist App was tested using a variety of meal images and fitness tracker data to ensure accuracy and reliability. The results were validated against known nutritional databases and fitness tracker outputs to confirm the accuracy of the AI-generated content.

## 10. Challenges and Solutions

Several challenges were encountered during the development of the AI Nutritionist App:

- **Image Processing**: Handling different image formats and ensuring accurate image recognition was a significant challenge. This was addressed by resizing images and standardizing input formats.
- **API Integration**: Integrating the fitness tracker API required careful handling of user authentication and data retrieval. This was solved by securely managing API keys and using error handling to manage API request failures.
- **Data Logging**: Ensuring that all user interactions were accurately logged required robust data handling. This was addressed by using pandas to structure and save data efficiently.

## 11. Conclusion

The AI Nutritionist App successfully combines image recognition, natural language processing, and fitness data integration to provide a comprehensive nutritional analysis tool. This project

demonstrates the potential of AI in improving personal health and wellness by offering users detailed insights into their dietary habits.

## 12. References


- **Google Generative AI Documentation**: Google AI Documentation
- **Streamlit Documentation**: Streamlit Documentation
- **Pandas Documentation**: Pandas Documentation