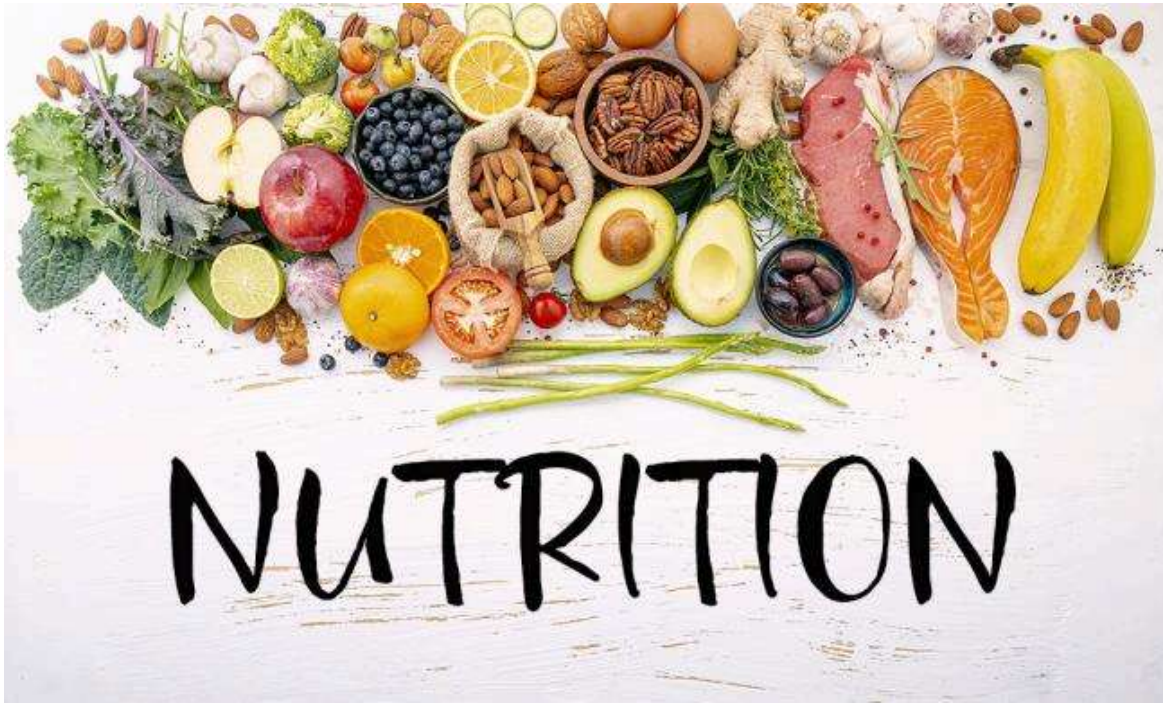


# Nutrition App Using Gemini Pro

*Your Comprehensive Guide To Enhancing Fitness and Wellness*



**Github Repo link:** [END-END-NUTRITION\\_APP-USING-GEMINI-API](#)

## **Team Members:**

1. Keerthi Krishna 21BAI1514
2. Aniruddhan N 21BRS1682
3. Ruwan Aryan 21BDS0280

## **Skills Required:**

1. Python
2. Deep Learning,
3. Streamlit

# Project Description:

Health Tracker AI is an innovative mobile application designed to revolutionize personal health management using the advanced capabilities of the Gemini Pro model. Leveraging artificial intelligence and deep learning techniques, the app analyzes user health data, fitness routines, and wellness goals to deliver personalized health plans, fitness insights, and lifestyle recommendations. The primary goal of Health Tracker AI is to empower users to achieve optimal health and wellness through data-driven guidance.

User Stories:

## Scenario 1: Weight Management

Sarah, a 30-year-old looking to manage her weight, relies on NutriPlan AI to achieve her goals. She inputs her dietary preferences, weight loss targets, and activity level into the app. NutriPlan AI generates personalized meal plans that include balanced portions of proteins, carbohydrates, and fats, tailored to Sarah's calorie needs and nutritional requirements. The app tracks Sarah's daily food intake through meal logging features, providing real-time feedback on her nutritional balance and suggesting adjustments to support her weight management journey effectively.

## Scenario 2: Vegan Lifestyle Support

Mark, a 25-year-old vegan, uses NutriPlan AI to optimize his plant-based diet. He inputs his vegan diet preferences, nutrient concerns, and fitness goals into the app. NutriPlan AI curates customized meal plans rich in plant proteins, essential vitamins, and minerals crucial for a vegan lifestyle. Mark benefits from a variety of vegan recipes with detailed cooking instructions and nutritional information. The app also offers insights into plant-based nutrition and suggests suitable supplements to ensure Mark maintains optimal health while adhering to his vegan diet.

## Scenario 3: Managing High Blood Pressure

Emily, a 50-year-old managing high blood pressure, turns to NutriPlan AI for dietary guidance. She inputs her health condition, dietary restrictions (such as low sodium), and medication information into the app. NutriPlan AI designs personalized meal plans focused on reducing sodium intake and promoting heart-healthy nutrition. The app provides Emily with low-sodium recipes, nutritional analyses highlighting sodium content, and meal tracking capabilities to monitor her daily sodium intake. Additionally, NutriPlan AI offers educational resources on managing high blood pressure through diet, empowering Emily to make informed choices for her cardiovascular health.

#### **Scenario 4: Vegan Nutrition Guidance**

Emma, a 25-year-old vegan, uses Health Tracker AI to enhance her nutrition and overall health. With a busy schedule and specific dietary preferences, she inputs her vegan diet requirements and fitness goals into the app. Health Tracker AI generates personalized meal plans rich in plant-based proteins, essential vitamins, and minerals to support Emma's lifestyle. The app provides recipes tailored to vegan diets, nutritional breakdowns, and shopping lists, enabling Emma to maintain a balanced diet effortlessly while achieving her fitness objectives.

#### **Scenario 5: Managing Food Allergies**

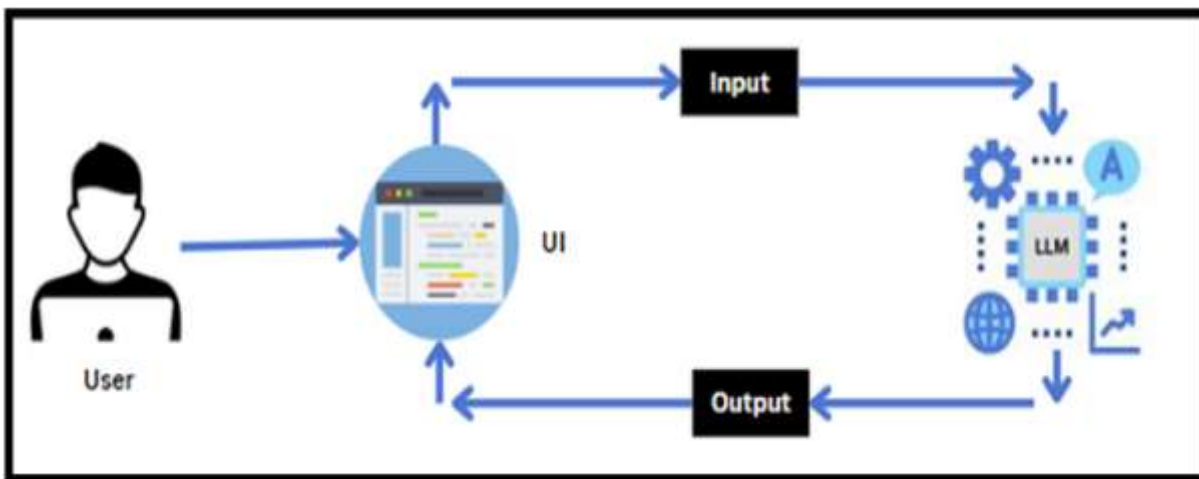
Alex, a 30-year-old with multiple food allergies, relies on Health Tracker AI to manage his dietary restrictions effectively. He inputs his allergens and nutritional requirements into the app. Health Tracker AI creates customized meal plans that exclude allergens while ensuring a balanced intake of nutrients. Alex uses the app to track his meals, receiving alerts and alternative ingredient suggestions to accommodate his allergies. The app also offers educational resources on managing food allergies through diet, empowering Alex to make informed food choices and maintain his health safely.

#### **Scenario 6: Performance Nutrition for Athletes**

Ryan, a 22-year-old competitive athlete, uses Health Tracker AI to optimize his performance nutrition. With rigorous training schedules and performance goals, he inputs his athletic requirements and dietary preferences into the app. Health Tracker AI designs personalized meal plans tailored to support Ryan's energy needs, muscle

recovery, and performance enhancement. The app recommends nutrient-dense meals, pre-workout snacks, and post-workout recovery options, providing Ryan with the nutritional insights and meal timing guidance necessary to excel in his sport.

## Technical Architecture:



## Project Flow:

- The user initiates interaction with the user interface (UI) to input their preferences and goals.
- User input is seamlessly collected from the UI and securely transmitted to the backend using the Google API key for authentication and data transmission.
- The input data is then forwarded to the Gemini Pro pre-trained model through a structured API call, ensuring efficient data transfer and compatibility.
- Leveraging its advanced machine learning algorithms, the Gemini Pro model meticulously processes the received input, analyzing intricate details and patterns to generate precise and personalized outputs.
- The results, enriched with valuable insights and recommendations, are seamlessly returned to the frontend. Here, they undergo meticulous formatting to ensure clarity and user-friendliness, enhancing the overall experience for the user.

## **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 Mode\***

- 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.

## **Project Structure**

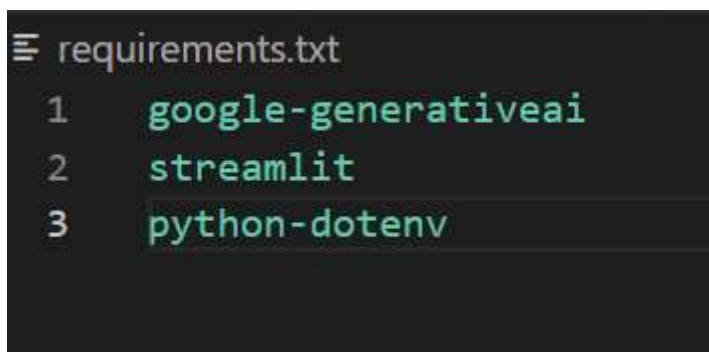
Create the Project folder, which contains files as shown below:

- **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.

## 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.

A screenshot of a code editor with a dark background. The file name 'requirements.txt' is visible in the top left corner. The file contains three lines of code, each on a new line and numbered 1, 2, and 3 respectively. The code is: 1 google-generativeai, 2 streamlit, and 3 python-dotenv. The text is in a light green color.

```
requirements.txt
1  google-generativeai
2  streamlit
3  python-dotenv
```

Here's the list aligned with proper spacing:

- **streamlit:** Streamlit is a powerful framework for building interactive web applications with Python.
- **streamlit\_extra:** 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 necessary Libraries

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

```
> pip install -r requirements.txt
```

## 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: [Maker suite google for API key and code](#)



## Get API key

### API keys

You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the [Google Cloud Platform Terms of Service](#), which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the [Gemini API Terms of Service](#).

Use your API keys securely. Do not share them or embed them in code the public can view.

If you use Gemini API from a project that has billing enabled, your use will be subject to [pay-as-you-go pricing](#).

[Create API key](#)

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Project number	Project ID	API key	Created	Plan
...5133	My First Project <a href="#">↗</a>	...nnaM	Jul 6, 2024	Free of charge <a href="#">Set up Billing</a> <a href="#">View usage data</a>

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

```
1 google_api_key = "AIzaSyAQJDw6xpKOBzKbyy2wqwe4bFQGt7nnnaM"
```

- 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

```
app.py > ...
1  ### Health Management APP
2  from dotenv import load_dotenv
3
4  load_dotenv() ## load all the environment variables
5  import streamlit as st
6  import os
7  import google.generativeai as genai
8  from PIL import Image
9
10 genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
```

### Activity 2: Implement a function to get Gemini response

```
def get_gemini_response(input,image,prompt):
    model=genai.GenerativeModel('gemini-pro-vision')
    response=model.generate_content([input,image[0],prompt])
    return response.text
```

- 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

```
def input_image_setup(uploaded_file):
    # Check if a file has been uploaded
    if uploaded_file is not None:
        # Read the file into bytes
        bytes_data = uploaded_file.getvalue()

        image_parts = [
            {
                "mime_type": uploaded_file.type,  # Get the mime type of the uploaded file
                "data": bytes_data
            }
        ]
        return image_parts
    else:
        raise FileNotFoundError("No file uploaded")
```

## Activity 4: Write a prompt for gemini model

```
input_prompt = """
You are an expert nutritionist. Analyze the food items in the image and calculate the total calories. Provide the details of each food item and its calories.

1. Item 1 - no of calories
2. Item 2 - no of calories
----
----
Total Calories:
"""

st.set_page_config(page_title="AI Nutritionist App")
st.header("AI Nutritionist App")

uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

image = ""

if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image.", use_column_width=True)

submit = st.button("Analyze Food & Calculate Calories")
```

## 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

AI Nutritionist Application:

```
st.set_page_config(page_title="AI Nutritionist App")

st.header("AI Nutritionist App")
input=st.text_input("Input Prompt: ",key="input")
uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])
image=""
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image.", use_column_width=True)

submit=st.button("Tell me the total calories")
```

On clicking, we get the desired result

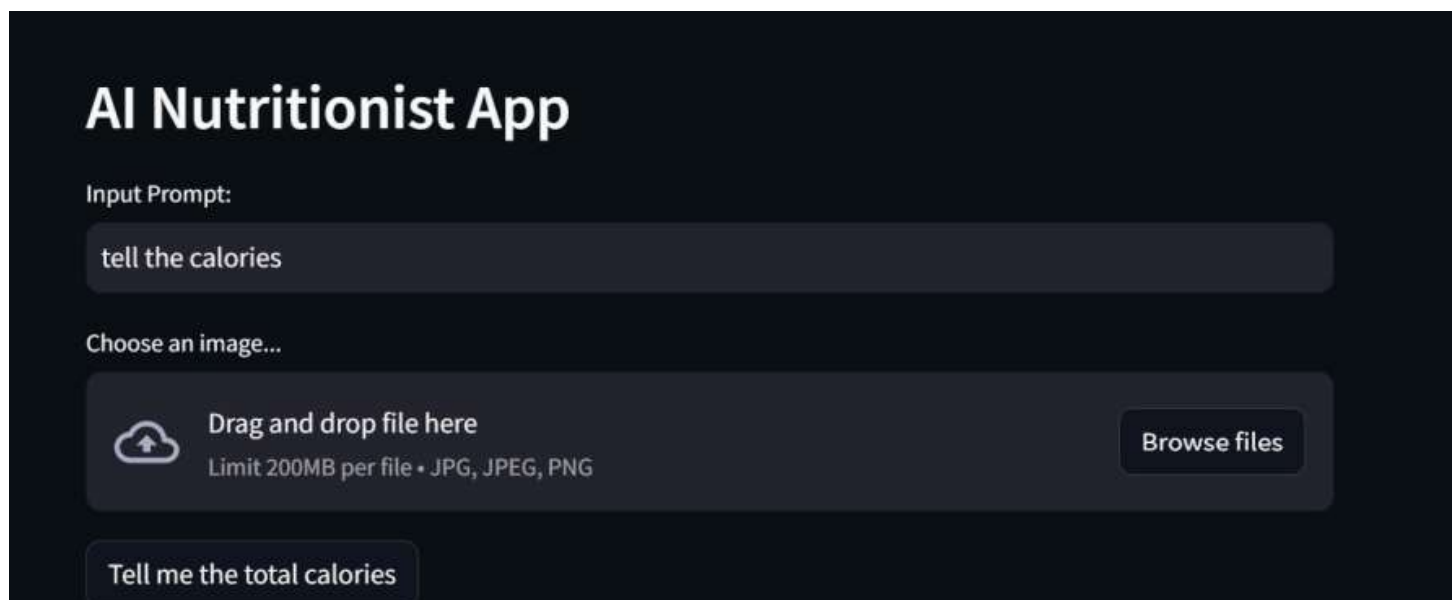
This code initializes a Streamlit application titled "AI Nutritionist App" by setting the page title and creating the app's header. It includes a text input field for users to enter a custom prompt and a file uploader for users to upload an image in JPG, JPEG, or PNG format. If an image is uploaded, 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

```
PS C:\Users\aniru\OneDrive\Pictures\Documents\GenAI_Course\Ch...  
  
You can now view your Streamlit app in your browser.  
  
Local URL: http://localhost:8501  
Network URL: http://192.168.29.180:8501
```

Launching the Application:

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



## Sample Images of the Web application

Sample 1 Input and Output:



Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files



nutrution image.jpg 304.1KB



Uploaded image.

Analyze Food & Calculate Calories

## Nutrition Analysis:

1. Hard-boiled egg - 78 calories
2. Whole wheat bread - 80 calories
3. Tomatoes - 16 calories
4. Mozzarella cheese - 87 calories
5. Basil leaves - 2 calories

Total Calories: 263 calories



Input 2 and Output2:

## AI Nutritionist App

Choose an image...



Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files



anna-pelzer-IGflGP5ONV0-unsplash.jpg 4.1MB



Uploaded image.

Analyze Food & Calculate Calories

### Nutrition Analysis:

1. Lettuce - 5 calories
2. Bell pepper - 30 calories
3. Tomatoes - 25 calories
4. Red cabbage - 11 calories
5. Sweet potato - 103 calories
6. Avocado - 160 calories
7. Chickpeas - 269 calories
8. Dressing - 100 calories

Total Calories: 703 calories

## Conclusion

Upon completing the development of the app, we have successfully implemented a streamlined process for predicting outcomes using the Gemini Pro model. The following key milestones were achieved:

### 1. Requirements Specification:

- Created and populated the `requirements.txt` file with all necessary libraries.
- Successfully installed the required libraries.

### 2. Initialization of Google API Key:

- Generated and initialized the Google API key, ensuring secure and authenticated communication with the backend.

### 3. Interfacing with Pre-trained Model:

- Loaded the Gemini Pro pre-trained model.
- Developed functions to retrieve responses from the Gemini model and read PDF content.
- Created effective prompts for interacting with the Gemini model.

### 4. Model Deployment:

- Seamlessly integrated the model with the web framework.
- Successfully hosted the application, making it accessible for users.

By following this structured approach, we have built a robust app that allows users to enter inputs via a user-friendly interface. These inputs are securely transmitted to the backend, processed by the Gemini Pro model, and returned to the frontend for display. This comprehensive solution ensures accurate and efficient predictions, enhancing the user experience and showcasing the capabilities of the Gemini Pro model.