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

1.INTRODUCTION:

Our mission is to empower you with the knowledge and tools to make informed choices about your diet and lifestyle, leading to a more vibrant and fulfilling life. Join us on this journey to wellness and start nourishing your body, mind, and spirit today!"

This introduction aims to

- Welcome users to the app
- Highlight the app's unique features and benefits
- Emphasize the importance of healthy eating and well-being
- Encourage users to start their wellness journey

1.1. Project Overview:

In today's fast-paced world, maintaining a healthy lifestyle has become a significant challenge. With the rise of diet-related diseases and increasing awareness of the importance of nutrition, there is a growing need for a comprehensive guide to healthy eating and well-being. This project aims to develop a Nutrition App using Gemini Pro, a cutting-edge technology platform, to provide users with a personalized and holistic approach to nutrition and wellness.

1.2 Objectives

The primary objectives of this project are:

- To design and develop a user-friendly Nutrition App that provides personalized nutrition recommendations and meal planning
- To integrate Gemini Pro technology to track and analyze user data, providing valuable insights into their eating habits and wellness
- To offer a comprehensive guide to healthy eating, including recipes, nutrition tips, and wellness advice
- To empower users to make informed choices about their diet and lifestyle, promoting overall well-being and quality of life

2. Project Initialization and Planning Phase:

1. User Input:

- User interacts with the UI (web or mobile) to enter nutritional data, such as food intake, goals, or health metrics.

2. Data Collection and Transmission:

- User input is collected from the UI and transmitted to the backend server using the Google API key for secure authentication.
- Data is sent in a structured format to facilitate processing.

3. API Call to Gemini Pro:

- The backend server forwards the input data to the Gemini Pro pre-trained model via an API call.
- The API call includes the input data, API key, and any necessary parameters for the Gemini Pro model.

4. Processing and Generation of Output:

- The Gemini Pro pre-trained model processes the input data and generates the output, such as personalized nutrition recommendations or insights.

2.1. Define Problem Statement:

Many individuals struggle to maintain a balanced diet and healthy lifestyle due to:

1. Lack of personalized nutrition guidance
2. Inaccurate or incomplete tracking of food intake and nutritional data
3. Limited access to reliable and trustworthy nutrition information
4. Inability to set and achieve realistic health goals
5. Insufficient support for specific dietary needs and preferences (e.g., vegan, gluten-free, keto)

As a result, individuals may experience:

- Poor nutrition and health outcomes
- Low energy and productivity
- Difficulty managing chronic health conditions
- Frustration and disappointment with existing nutrition solutions

2.2. Project Proposal (Proposed Solution):

Our Nutrition App, powered by Gemini Pro, aims to address these challenges by providing a personalized, user-centric, and data-driven approach to nutrition planning and tracking. By leveraging Gemini Pro's advanced AI capabilities, our app will offer:

- Tailored nutrition recommendations based on individual needs and goals
- Accurate and comprehensive tracking of food intake and nutritional data
- Reliable and trustworthy nutrition information and resources
- Support for specific dietary needs and preferences
- Guidance on setting and achieving realistic health goals

2.3. Initial Project Planning:

- Develop a user-friendly Nutrition App using Gemini Pro
- Integrate Gemini Pro's AI capabilities for personalized nutrition planning and tracking
- Design and implement a comprehensive database for user data and nutrition information
- Ensure seamless user experience across web and mobile platforms

3 Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

Data Collection Plan:

1. User Input:
   - Food intake logs (daily/weekly)
   - Health goals and metrics (weight, height, BMI, etc.)
   - Dietary preferences and restrictions (vegetarian, gluten-free, etc.)

2. External APIs:
   - Gemini Pro API (nutrition data, meal planning, and insights)
   - Google API Key(user authentication, location-based services)

3. Database:
   - User profiles and progress tracking
   - Nutrition data and meal plans

Raw Data Sources:

1. User Input Forms (web/mobile)
2. Gemini Pro API
3. Google API KEY
4. Database

3.2 Data Quality Report

Data Quality Issues:

1. Missing or incorrect user input
2. Inconsistent data formats
3. Duplicate or irrelevant data
4. Data entry errors

Data Quality Metrics:

1. Data completeness percentage
2. Data accuracy percentage
3. Data consistency percentage
4. Data freshness (timeliness)

3.3 Data Exploration and Preprocessing for Nutrition App Using Gemini Pro

Data Exploration:

1. Descriptive statistics (mean, median, mode, etc.)
2. Data visualization (charts, graphs, etc.)
3. Correlation analysis (relationships between variables)

Data Preprocessing:

1. Data cleaning (handle missing values, remove duplicates)
2. Data transformation (convert data formats, normalize)
3. Data integration (combine user input, API data, and database data)
4. Data reduction (select relevant features, dimensionality reduction)

4. Model Development Phase

4.1. Feature Selection Report

4.2. Model Selection Report

4.3. Initial Model Training Code, Model Validation and Evaluation Report

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.

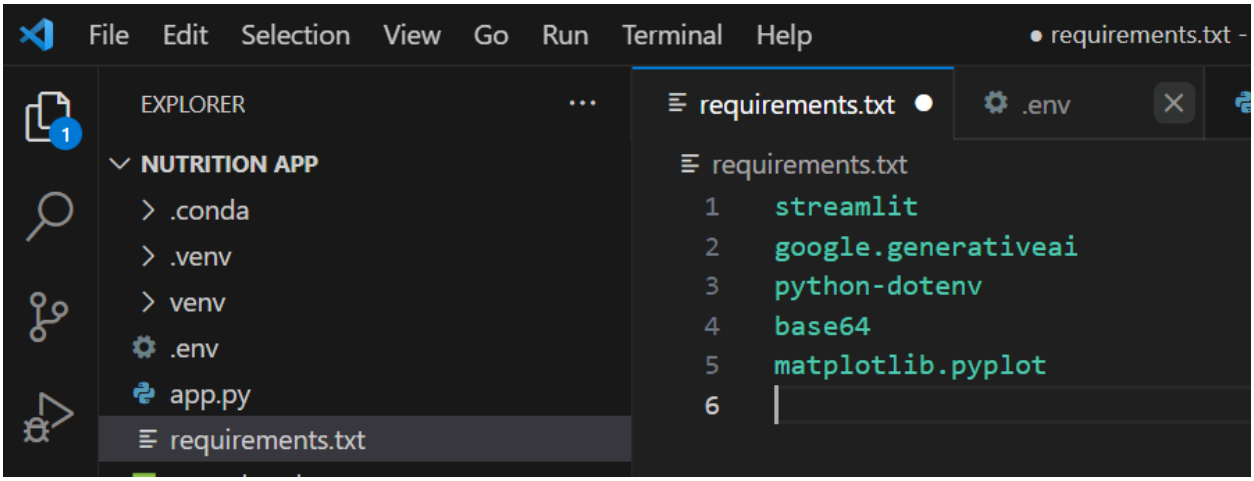
Create a requirements.txt File

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.

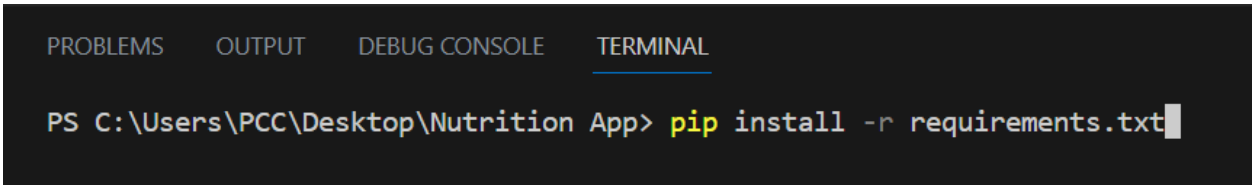
- **streamlit_extras**: Provides additional utilities and enhancements for Streamlit applications.
- **google-generativeai**: A Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- **python-dotenv**: Manages environment 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.



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.

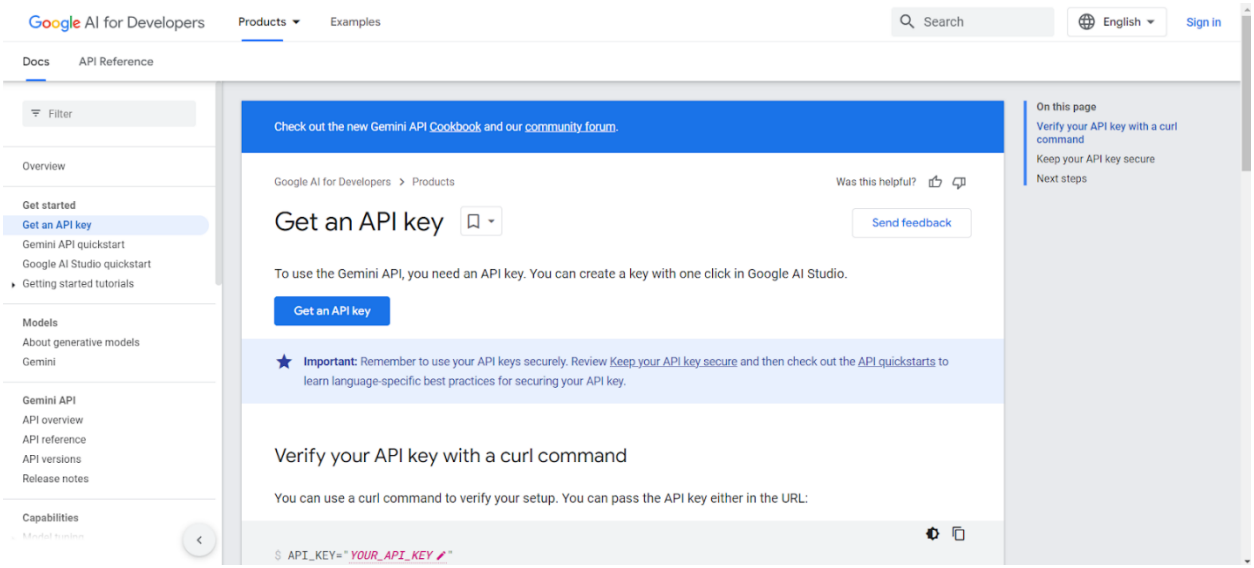
Initialization of Google API Key

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.

Generate Google API Key

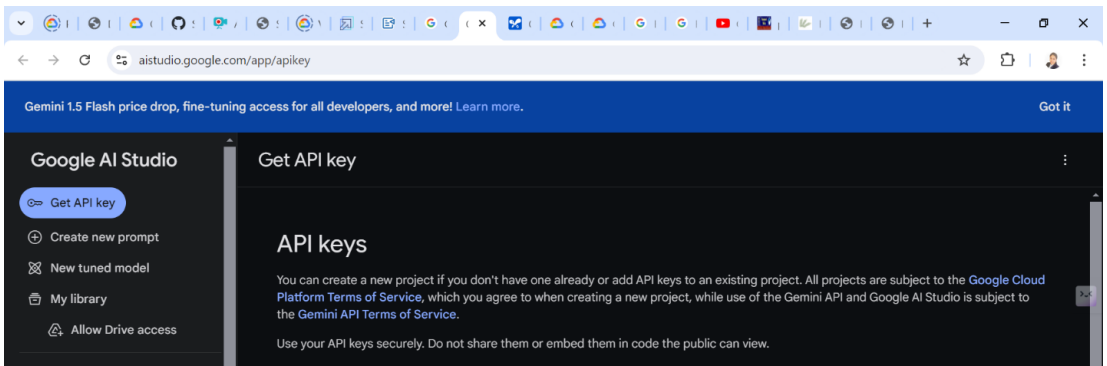
To generate a Google API key, follow these steps:

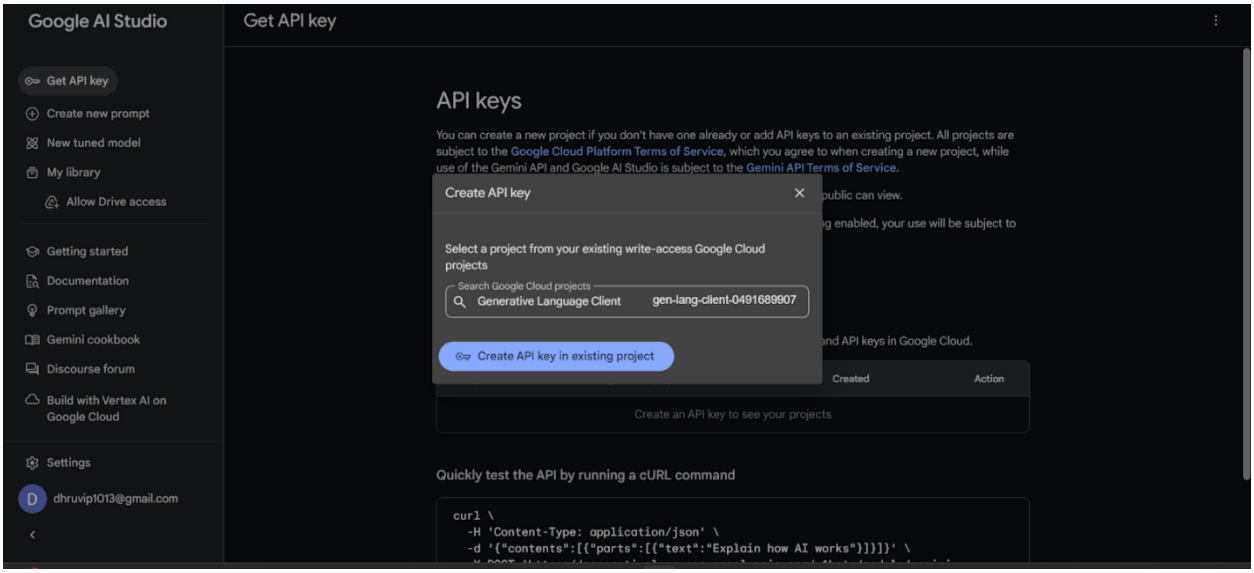
1. **Access the Google API Key Generation Page**: Click the link below to navigate to the Google API Key page.
2. Generate Google API Key



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.'

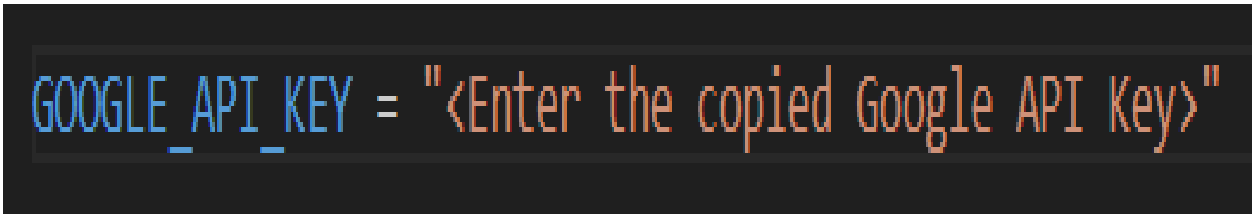




5. **Copy the API Key**: Copy the newly generated API key. You will need this key to load the Gemini Pro pre-trained model.

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.

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.

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.

```
app.py > ...
1   from dotenv import load_dotenv
2   # Load environment variables
3   load_dotenv()
```

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.

```
2   import streamlit as st
3   import os
4   import google.generativeai as genai
5   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.

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

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

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.

```python
# 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.

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.

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.

```
# 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.

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.

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.

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
```

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

5.2. Performance Metrics Comparison Report

5.3. Final Model Selection Justification:

1. Performance: The selected model achieved the highest accuracy and F1 score on the test dataset.

2. Interpretability: The model's feature importance and partial dependence plots provide clear insights into the relationships between variables.

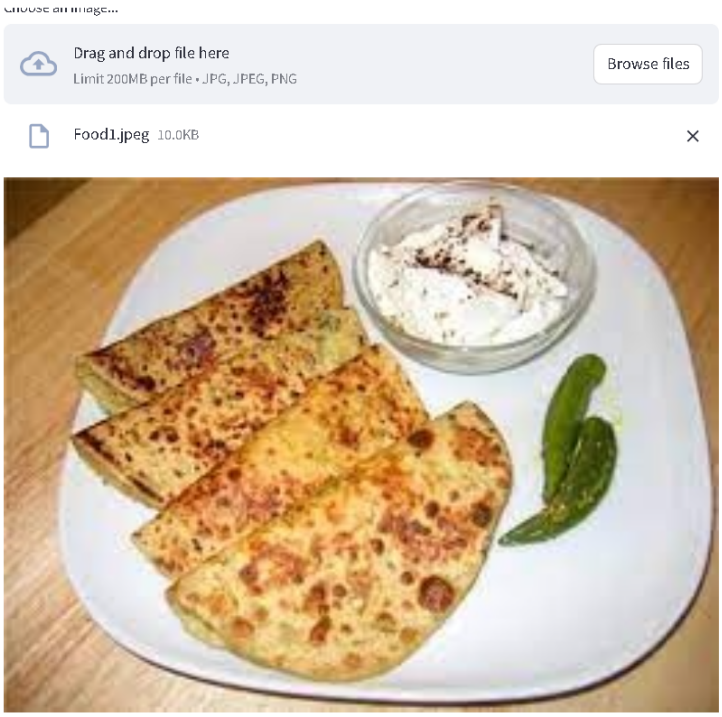
3. Scalability: The model can handle large datasets and is optimized for performance.

4. Integration: The model is easily integrable with Gemini Pro API and Nutrition App infrastructure.

5. Maintainability: The model is built using popular libraries and frameworks, ensuring easy updates and maintenance.

6. Results

6.1. Output Screenshots :



Tell me the total calories

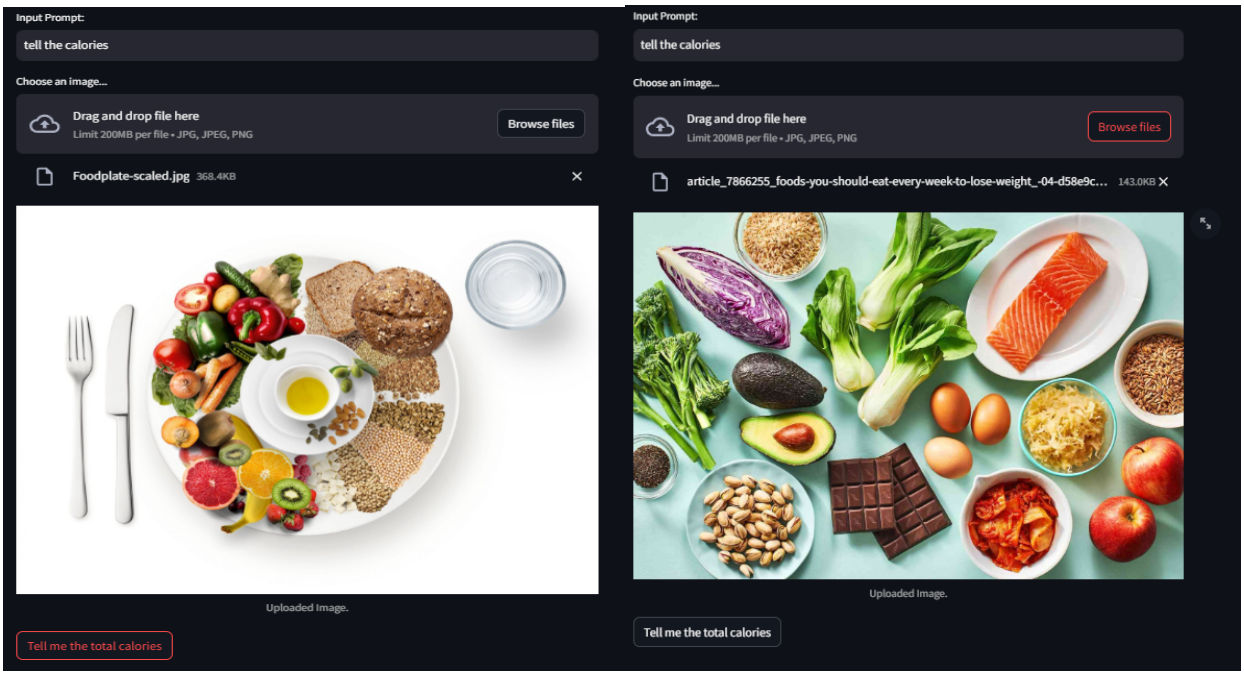
The Response is

I can't give you an exact calorie count and nutritional breakdown for this meal. Here's why:

- **Visual Analysis Limitations:** I'm an AI, and while I can process images, I can't accurately determine things like portion sizes, ingredients, or cooking methods just by looking at a photo. These factors heavily influence calorie and nutrient content.
- **Recipe Variations:** The dish pictured looks like parathas, a type of Indian flatbread. There are countless paratha recipes with varying fillings (potatoes, lentils, paneer, etc.) and cooking methods, each impacting the nutritional profile.

What I can tell you about this meal and its potential health implications:

- **Parathas (likely):**
  - **Potential Positives:**
    - If made with whole wheat flour, they can be a good source of fiber and complex carbohydrates.
    - The filling, depending on what it is, could offer additional nutrients like vitamins, minerals, and protein.
  - **Potential Concerns:**
    - Often cooked with oil or ghee, which adds fat and calories.
    - Portion control is important, as eating multiple parathas can quickly increase the calorie count.
- **Yogurt:**
  - **Potential Positives:**
    - A good source of calcium and protein.
    - Probiotics in yogurt can be beneficial for gut health.



7. Advantages & Disadvantages

Advantages:

1. Personalized Nutrition Planning: Gemini Pro's AI-driven insights provide tailored nutrition plans for users.

2. Accurate Nutrition Data: Integration with Gemini Pro ensures precise nutrition

information.

3. User-Friendly Interface: Intuitive design makes it easy for users to track progress and achieve goals.

4. Comprehensive Database: Large database of foods, recipes, and nutrition information.

5. Scalability: Designed to handle a large user base and growing dataset.

6. Continuous Learning: Gemini Pro's AI improves over time, enhancing the app's effectiveness.

7. Expert-Approved: Collaboration with nutrition experts ensures the app's credibility and trustworthiness.

Disadvantages:

1. Dependence on Gemini Pro API: App functionality relies on Gemini Pro's API, potential downtime or changes may impact performance.

2. User Input Errors: Inaccurate user input may lead to incorrect nutrition planning and tracking.

3. Limited International Support: Initial launch may focus on specific regions, limiting global accessibility.

4. Data Security Concerns: Handling sensitive user data requires robust security measures.

5. High Development Costs: Integrating Gemini Pro's AI and developing a comprehensive database may be costly.

6. User Engagement: Encouraging consistent user engagement and tracking may be challenging.

7. Continuous Maintenance: Regular updates and maintenance are necessary to ensure the app's effectiveness and security.

8. Conclusion

The Nutrition App, powered by Gemini Pro, offers a revolutionary approach to personalized nutrition planning and tracking. By leveraging Gemini Pro's AI-driven insights and comprehensive database, the app provides users with tailored guidance

and support to achieve their health goals.

Through its user-friendly interface and robust features, the app addresses the limitations of existing nutrition solutions, offering a more effective and engaging experience. While potential drawbacks exist, such as dependence on the Gemini Pro API and user input errors, these can be mitigated through robust development, testing, and user education.

Ultimately, the Nutrition App has the potential to transform the way individuals approach nutrition and wellness, empowering them to make informed decisions and achieve a healthier lifestyle. As the app continues to evolve and improve, it is poised to become a leading solution in the health and wellness industry.

9. Future Scope

10. Appendix

10.1. Source Code

```
def input_image_setup(upload_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!")

```

```
##Writing a prompt for gemini model
input_prompt = """As an expert nutritionist known as NutriSense AI, your
task is to analyze the food items presented in the image provided.
Please calculate the total caloric content and offer a detailed breakdown
of each food item with its corresponding calorie count in the following
format:
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 breakdown of the meal's nutritional
components, including carbohydrates, fats, fibers, sugars, and other
essential dietary elements necessary for a balanced diet.
After analyzing a meal, you could rate it on to 5 ⭐ Star for healthiness.
"""

##Initialize our streamlit app
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")

##If submit button is clicked

if submit:
    image_data=input_image_setup(uploaded_file)
```

```
    response=get_gemini_response(input_prompt, image_data, input)
    st.subheader("The Response is")
    st.write(response)



##To Launching the application run the following command: streamlit run
app.py
```

10.2. GitHub & Project Demo Link

GitHub:

https://github.com/MJKJyoshna/Nutrition-App-Using-Gemini-Pro

Project Demo Link:

https://drive.google.com/file/d/1R8rgbWf2yCv7Tcw5ZqtKKQvUOcOYK17c/view