|  |
| --- |
| *Report on Nutrition App Using Gemini Pro**done as a part of*  **Generative AI Applications using Vertex AI**  Submitted by  **TEAM ID : SWTID1720159923**  **MIRUDHULAA M**  **SURYA R**  **KARTHIK K**  **MONISHA A**  To  Team SmartBridge |

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

**Project Description**

Nutritionist AI is a innovative mobile application created to offer customized dietary recommendations and nutritional advice, utilizing the advanced features of the Gemini Pro model. This app employs artificial intelligence to examine user data, dietary preferences, and health objectives, providing personalized meal plans, nutritional insights, and wellness tips. The main goal of Nutritionist AI is to encourage healthier eating habits and enhance overall well-being through smart, data-driven recommendations

.

**Scenario 1:** Muscle Gain and Fitness Improvement

John, a 32-year-old aiming to gain 10 pounds of muscle and improve his overall fitness, turns to Nutritionist AI for support. As an omnivore with a high activity level, he inputs his dietary preferences, fitness goals, and current workout regimen into the app. Nutritionist AI generates a high-protein, nutrient-dense meal plan tailored to his needs, ensuring he gets the right balance of macronutrients to support muscle growth. John logs his meals by taking photos or scanning barcodes, and the app provides real-time feedback on his protein intake and overall nutritional balance, suggesting necessary adjustments. By syncing his fitness tracker, the app integrates his workout data, offering comprehensive insights to help John optimize his diet and exercise routine, ensuring he stays on track with his muscle gain goals while maintaining proper nutrition.

**A screenshot of a computer

Description automatically generated**

Scenario 2: Heart Health Management

Edward, a 50-year-old with a history of heart disease, uses Nutritionist AI to manage his condition through a heart-healthy diet. He inputs his dietary preferences and cardiovascular health goals into the app, which generates meal plans that focus on low sodium, healthy fats, and high fiber content to support his heart health. Edward logs his meals by taking photos , receiving immediate feedback on their suitability for his heart condition. Detailed nutritional breakdowns highlight sodium levels, saturated fat content, and essential heart-healthy nutrients, aiding Edward in making informed food choices. Additionally, the app provides educational resources about managing heart disease through diet, keeping Edward well-informed and empowered to handle his condition better.

**A screenshot of a computer screen

Description automatically generated**

Scenario 3: Weight Loss Journey

Sophia, a 25-year-old with a goal to lose 15 pounds, uses Nutritionist AI 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. Nutritionist AI creates a calorie-controlled, nutrient-dense meal plan tailored to her vegetarian diet. Sophia 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

**A screenshot of a diagram

Description automatically generated**

**Technical Architecture**

A diagram of a computer program

Description automatically generated

**Project Flow**

* **User Interaction:** The user interacts with the UI to enter their input.
* **Data Collection:** The user input is collected from the UI and transmitted to the backend using the Google API key.
* **API Call:** The input is forwarded to the Gemini Pro pre-trained model via an API call.
* **Processing:** The Gemini Pro pre-trained model processes the input and generates the output.
* **Result Display:** 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

To complete this project, prior knowledge of the following topics is essential:

**Generative AI Concepts**

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

**Project Structure**

Create the Project folder which contains files as shown below:

A screenshot of a computer

Description automatically generated

* **.env file:** Securely stores the Google API key.
* **app.py:** The main application file containing both the model and Streamlit UI code.
* **requirements.txt:** Lists the necessary libraries to ensure proper functioning.
* 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.**

A screen shot of a computer

Description automatically generated

* **streamlit**: 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:** Allows management of environment variables stored in a .env file for Python projects.
* **PyPDF2:** A Python library for extracting text and manipulating PDF documents.
* **Pillow:** 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**

A screen shot of a computer

Description automatically generated

* 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 [Google API Key page](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.
* 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.

A screenshot of a computer

Description automatically generated

**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.

A screenshot of a computer

Description automatically generated

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**

A screenshot of a computer screen

Description automatically generated

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 the Gemini response**

A black screen with text

Description automatically generated

* 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**

A screen shot of a computer program

Description automatically generated

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 the Gemini model**

A black screen with orange text

Description automatically generated

The variable input\_prompt is a multi-line string designed as a prompt for a nutritionist AI 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

Milestone 4: Model Deployment

Deploying our model using the Streamlit framework allows us to create and share data applications quickly and easily. Streamlit enables us to build interactive web applications that allow users to interact with our models in real time, offering an intuitive and seamless user experience.

**Activity 1: Integrate with Web Framework**

We begin by integrating our model with the Streamlit framework, setting up the necessary UI components and functionality for user interaction.

**AI Nutritionist Application:**

* **Initialization and Setup:**
  + Set up the page title and create the app's header.
  + Include a text input field for users to enter a custom prompt.
  + Provide a file uploader for users to upload an image in JPG, JPEG, or PNG format.
  + Display the uploaded image within the app with a caption.
  + Provide a button labeled "Tell me the total calories" to trigger the application's functionality for analyzing the uploaded image.

A screen shot of a computer program

Description automatically generated

**Activity 2: Host the Application**

Hosting the application involves deploying the Streamlit app to a cloud platform or a local server so that users can access and interact with it.

**Steps to Host the Application:**

1. **Open the Terminal:**
   * Navigate to the project directory.
2. **Run the Streamlit Application:**
   * Execute the following command in the terminal: streamlit run app.py
   * This command launches the application and provides a URL for accessing the app in a web browser.

A screenshot of a computer program

Description automatically generated

Sample Results:

**Input Example 1:**

* User uploads an image of a meal.
* User clicks the "Tell me the total calories" button.

A screenshot of a nutrition app

Description automatically generated

**Output Example 1:**

The app analyzes the image and displays the total calorie count along with the dietery information of each food and also tells how to improve the health by intaking proper diet food.

A screenshot of a computer

Description automatically generated

**Input Example 2:**

* User enters a custom prompt and uploads another image.
* User clicks the "Tell me the total calories" button.

A screenshot of a computer

Description automatically generated

**Output Example 2:**

The app processes the input prompt and image, providing detailed nutritional information and calorie breakdown for the uploaded meal also tells it is an healthy meal or not and suggests to improve

the meal.

A screenshot of a computer

Description automatically generated

A screen shot of a recipe

Description automatically generated

----------------------------------------------------------------------------------------------------------------------------------