

Nutrition App Using Gemini Pro: Your Comprehensive Guide To Healthy Eating And Well-Being

Milestone 1: Introduction

In today's fast-paced world, maintaining a healthy diet can be challenging. Many people struggle with accurately assessing the nutritional content of their meals and understanding how they align with their dietary needs. This is especially true for people managing chronic conditions like diabetes, where precise dietary choices are crucial. Our Nutrition App addresses these issues by offering a simple, user-friendly solution. By using advanced AI to analyze food images, the app provides instant, detailed information about calorie counts, nutritional features, and suitability for specific health conditions, empowering users to make informed dietary decisions and achieve their wellness goals with ease.

The primary objectives of our Nutrition App are to deliver precise nutritional analysis by accurately identifying calorie counts and key nutritional features of various foods from images. The app aims to assess meal suitability for specific health conditions, such as diabetes, providing users with crucial insights for making informed dietary decisions. By offering an intuitive and accessible interface, the app ensures ease of use for all individuals, supporting effective dietary management.

Milestone 2: Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

Activity 1: Define Problem Statement

Problem Statement: John, a 45-year-old with Type 2 Diabetes, relies on Nutritionist AI 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.

Ref. template: [Click Here](#)

Nutrition App Problem Statement Report: [Click here](#)

Activity 2: Project Proposal (Proposed Solution)

The proposed solution utilizes Gemini Pro's API to build a nutrition app that accepts food images and provides detailed nutritional information, aiding users, especially diabetic patients, in making informed dietary choices. The app uses advanced image recognition to identify food items and fetches comprehensive nutritional data. This solution not only supports individual dietary management but also aids healthcare providers in offering personalized dietary advice, ultimately contributing to improved public health and well-being through AI-driven nutritional guidance.

Ref. template: [Click Here](#)

Nutrition App Project Proposal Report: [Click here](#)

Activity 3: Initial Project Planning

The project planning phase for the Nutrition App Using Gemini Pro involves key actions to establish a solid foundation. Detailed requirements are specified, including necessary libraries and API key initialization. Preparations for model development are made by interfacing with pre-trained models and implementing the API, and plans for model deployment include integration with the web framework and hosting the application. These actions ensure the team has a clear direction and all necessary resources and tools are identified and prepared.

Ref. template: [Click Here](#)

Nutrition App Project Planning Report: [Click here](#)

Milestone 3: Data Collection and Preprocessing Phase

There is no need for a dataset in this project as the app will directly access a pre-trained model using an API key. This approach leverages Gemini Pro's advanced capabilities, which already include extensive training on diverse food images and nutritional data. By utilizing the API, the app can instantly identify food items and provide detailed nutritional information without the need for additional data collection or model training. This not only simplifies the development process but also ensures that the app benefits from the accuracy and reliability of a thoroughly trained and tested model, allowing the team to focus on other critical aspects such as user interface design and functionality.

Activity 1: Data Collection Plan, Raw Data Sources Identified, Data Quality Report

Ref. template: [Click Here](#)

Nutrition App Data Collection Report: [Click here](#)

Activity 2: Data Quality Report

Ref. template: [Click Here](#)

Nutrition App Data Quality Report: [Click here](#)

Activity 3: Data Exploration and Preprocessing

Ref. template: [Click Here](#)

Nutrition App Data Exploration and Preprocessing Report: [Click here](#)

Milestone 4: Model Development Phase

For our Nutrition App project, there is no need for additional model development as we are leveraging the capabilities of Gemini Pro. This advanced model already provides comprehensive analysis, including calorie counts, nutritional features, and suitability for diabetes patients, directly from food images. By utilizing Gemini Pro's existing functionalities, we can efficiently integrate these features into our app without the need for further model development.

Activity 1: Feature Selection Report

Ref. template: [Click Here](#)

Nutrition App Feature Selection Report: [Click here](#)

Activity 2: Model Selection Report

Ref. template: [Click Here](#)

Nutrition App Model Selection Report: [Click here](#)

Activity 3: Initial Model Training Code, Model Validation and Evaluation Report

Ref. template: [Click Here](#)

Nutrition App Model Development Phase Template: [Click here](#)

Milestone 5: Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining Gen AI models for peak performance.

Activity 1: Prompt Design

We focused on prompt design to tailor the Gemini Pro model to our specific application. By carefully crafting prompts that guide the model's attention to relevant details—such as identifying specific food items, estimating their calorie content, and evaluating their nutritional properties—we ensured that the model delivers precise and actionable information.

Nutrition App Model Optimization and Tuning Phase Report: [Click here](#)

Milestone 6: Results

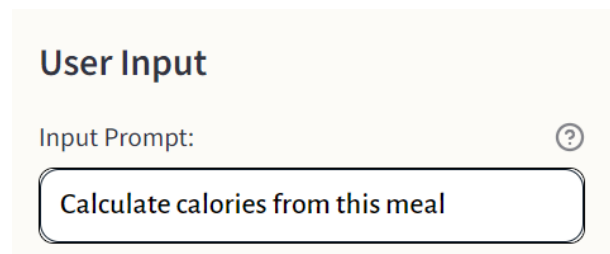
Web application UI:

The user-friendly UI makes the application easier to understand and navigate. The application uses Streamlit for the page creation and deployment. The styling was done using CSS.




The screenshot shows the main interface of the AI Nutritionist App. On the left is a sidebar titled 'User Input' containing an 'Input Prompt' text box with the default text 'Calculate calories from this meal', an 'Upload an image' section with a 'Browse files' button, and a 'Submit' button. The main area on the right has a light beige background with the title 'AI Nutritionist App' and the tagline 'Helping you manage your diet intelligently.' below it.

The input prompt for the Gemini model can be given through the input text box in the sidebar. The default prompt is to calculate and display the calories in the meal. It will also generate the basic nutrient distribution in each food item and provide a decision regarding the health factor in regards with diabetic patients. The prompt can be modified by the user to a certain extent.

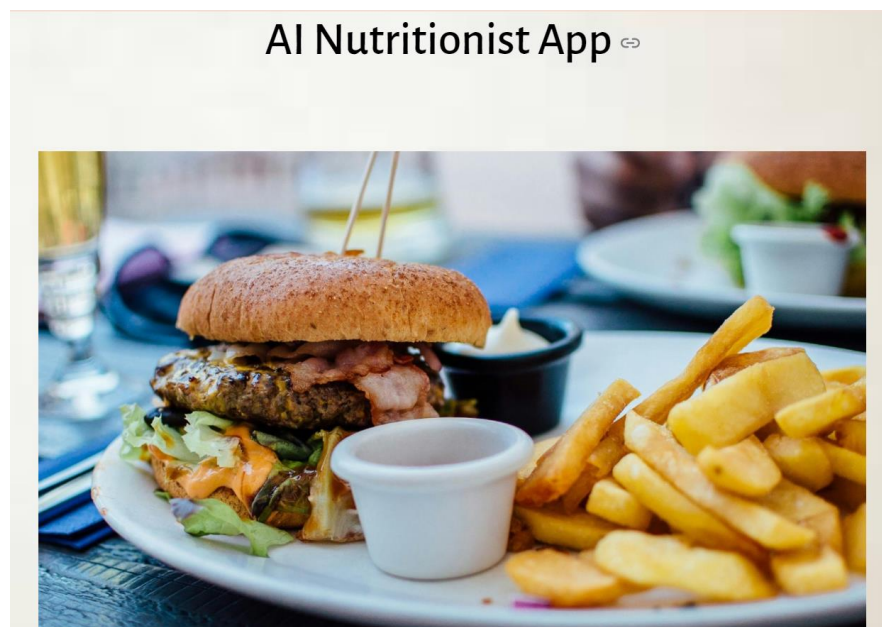


This is a close-up of the 'User Input' sidebar. It shows the 'Input Prompt:' label, a question mark icon, and a text box containing 'Calculate calories from this meal'.

The user can upload the image of the meal through the 'Browse files' option in the sidebar. The file can be in PNG, JPEG, or JPG format and must be less than 200MB. As long as the file follows these constraints, it will be successfully uploaded and the preview will be displayed. Once uploaded, the user must click 'Submit' to get the response.

Upload an image: 

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG



The response has been tailored to provide the necessary information in an understandable format. This was achieved using prompt design, where we tuned the model to provide the desired output by providing it with a base prompt.

Response:

1. Burger - 500 calories 2. French Fries - 300 calories 3. Mayonnaise - 100 calories

This meal is not suitable for a diabetes patient. It is high in carbohydrates and unhealthy fats.

- Grilled Salmon - Rich in omega-3 fatty acids and protein, good for heart health - 200 calories
- Roasted Vegetables - Rich in fiber and vitamins - 100 calories
- Brown Rice - Rich in fiber and complex carbohydrates - 200 calories

Some edge cases have also been identified and the corresponding responses were provided to the application. For example, if the image does not contain any food items, the application will request the user to verify and upload again.

AI Nutritionist App



Response:

This image contains a beach scene and no food items. Therefore, it is not possible to calculate calorie intake or provide alternative food recommendations.

Advantages:

- **Convenience:** Users can quickly analyze their meals by simply taking a photo, making it easy to track nutritional information on the go.
- **Accurate Insights:** Provides detailed information on calorie counts, nutritional features, and suitability for specific health conditions, helping users make informed dietary choices.
- **User-Friendly:** Designed with an intuitive interface that is accessible to users of all technical levels, promoting ease of use.
- **Health Management:** Assists individuals in managing their dietary needs, particularly those with specific health conditions like diabetes.
- **Continuous Improvement:** Regular updates and optimizations based on user feedback ensure the app remains effective and accurate.

Disadvantages:

- **Image Quality Dependency:** The accuracy of the analysis may depend on the quality and clarity of the food images provided by users.
- **Limited Food Database:** The app's effectiveness is contingent on the breadth of its food database; less common or unique foods may not be accurately analyzed.
- **Nutritional Variability:** Variations in food preparation and ingredients can affect the accuracy of nutritional information.

Conclusion

Our Nutrition App is a powerful tool designed to address the common challenges of dietary management in today's fast-paced world. By leveraging advanced AI technology, the app provides users with accurate nutritional analysis and valuable insights into meal suitability for specific health conditions, such as diabetes. Its user-friendly interface ensures accessibility for all, making it easy for individuals to make informed dietary decisions. Despite potential limitations such as dependency on image quality and database scope,

the app's continuous improvement approach ensures it remains a reliable and effective solution. Ultimately, our Nutrition App empowers users to take control of their health and wellness with confidence and convenience.

Future Scope

The future scope for our Nutrition App includes expanding the food database to cover a wider variety of foods, improving image recognition technology for greater accuracy, and integrating more personalized health recommendations. We also aim to incorporate features like meal planning, dietary tracking, and integration with wearable health devices. Additionally, enhancing data privacy measures and offering multilingual support will broaden the app's accessibility and user base, ensuring it meets the evolving needs of users globally.

Appendix

Code:

```
import streamlit as st
from dotenv import load_dotenv
load_dotenv()
import os
import google.generativeai as genai
from PIL import Image
import time
import base64

load_dotenv()
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

def get_base64(file):
    with open(file, 'rb') as f:
        data= f.read()
    return base64.b64encode(data).decode()

image_path = "bg2.jpg" # Adjust the path to where your image is located
base64_image = get_base64(image_path)
```

```
input_prompt ="""
You are an expert nutritionist. Analyze the food items in the provided image to
calculate the total calories and provide details of each item with its calorie
intake. Follow this format:

1. Item 1 - number of calories

2. Item 2 - number of calories
...
...

Finally, assess whether the meal is suitable for a diabetes patient. Provide
alternative food recommendations suitable for diabetes management, focusing on
low carbohydrates and high fiber, in the following format:

- Item: benefits - number of calories

The user has Type 2 Diabetes and prefers low-carb options. Provide response only
for questions asked. Do not add extra explanations or creativity.
"""

def get_gemini_response(input, image, prompt):
    model= genai.GenerativeModel('gemini-1.5-flash')
    try:
        response= model.generate_content([input, image[0], prompt])
        return response.text
    except Exception as e:
        st.error(f"An error occurred: {e}")
        return None

def input_image_setup(uploaded_file):
    if uploaded_file is not None:
        bytes_data= uploaded_file.getvalue()
        image_parts= [{
            "mime_type": uploaded_file.type,
            "data": bytes_data
        }]
        return image_parts
    else:
        raise FileNotFoundError("No file uploaded")
```

```
# Set page configuration
st.set_page_config(page_title="AI Nutritionist App", page_icon=":apple:",
layout="wide")

import streamlit as st

# CSS to hide the top menu, footer, and deploy button
hide_streamlit_style= """
<style>
#MainMenu {visibility: hidden;}
footer {visibility: hidden;}
header {visibility: hidden;} /* This hides the entire header including the
"Deploy" button */
</style>
"""

st.markdown(hide_streamlit_style, unsafe_allow_html=True)

# Create a custom sidebar menu

# Add background styling
st.markdown(
    f"""
    <style>
    .stApp {{
        background-image: url("data:image/png;base64,{base64_image}");
        background-size: cover;
        background-repeat: no-repeat;
        background-position: center;
        color: #E0E1DD;
        overflow: hidden;
    }}
    </style>
    """,
    unsafe_allow_html=True
)

# Add background image and custom styles
st.markdown(
    f"""
    <style>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Alegreya+Sans:ital,wght@0,100;0,300;0,400;0,500;0,700;0,800;0,900;1,100;1,300;1,400;1,500;1,700;1,800;1,900&display=swap');

.reportview-container {{
    background: rgba(255, 255, 255, 0.8);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
    backdrop-filter: blur(10px);
    font-family: "Alegreya Sans", sans-serif;
    font-weight: 500;
    font-style: normal;
    color: black;
}}

.header {{

    color: white;
    padding: 10px 0;
    text-align: center;

    margin-bottom: 20px;
}}

.title {{
    font-family: "Alegreya Sans", sans-serif;
    font-weight: 500;
    font-style: normal;
    color: black;
}}

.footer {{
    text-align: center;
    padding: 5px;

    font-family: "Alegreya Sans", sans-serif;
    font-weight: 500;
    font-style: normal;
    color: black;

    width: 100%;
    bottom: 0;
}}
```

```
.response-box {{
    width: 100%;
    padding: 20px;
    margin-top: 20px;
    background: #ffffff;
    border-radius: 7px;
    border: 1px solid #000000;
    font-family: "Alegreya Sans", sans-serif;
    font-weight: 500;
    font-style: normal;
    color: black;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
}}

</style>
""",
    unsafe_allow_html=True
)

# Header
st.markdown(
    """
    <div class="header">
        <h1 class="title">AI Nutritionist App</h1>
    </div>
    """,
    unsafe_allow_html=True
)

# Sidebar
with st.sidebar:
    st.header("User Input")
    input= st.text_input("Input Prompt: ", key="input", value="Calculate calories
from this meal", help="Enter a prompt for the AI Nutritionist")
    uploaded_file= st.file_uploader("Upload an image:", type=["jpg", "jpeg",
"png"], help="Upload an image of your meal")
    submit= st.button("Submit", key="submit", help="Click to analyze the uploaded
image")

st.markdown(
```

```

"""
<style>

.stSidebar {
    background: linear-gradient(to right, #EFE9E1, #ffffff) !important;
}

.stSidebar .stFileUploader > div > div {
    background: linear-gradient(to right, #EFE9E1, #ffffff) !important;
    border-radius: 10px;
    padding: 10px;
    border: 1px solid #0D1B2A !important;
}

/* Style the text input widget */
.stTextInput > div > div > input {
    background-color: #ffffff;
    border-radius: 10px; /* Change border radius */
    padding: 10px; /* Add padding */
    font-family: "Alegreya Sans", sans-serif;
    font-weight: 450;
    font-style: normal;
    color: black;
    border: 1px solid #0D1B2A; /* Change border color */
}
</style>
"""
unsafe_allow_html=True
)

st.markdown(
    """
    <style>
    /* Ensure the button is properly displayed and overrides config commands */
    .st-emotion-cache-mtzh7f {
        background-color: #ffffff !important;
        color: black !important;
        border: 1px solid #000000 !important;
        padding: 10px 20px !important;
        text-align: center !important;
        display: inline-block !important;
        font-family: "Alegreya Sans", sans-serif;
    
```

```

        font-weight: 450;
        font-style: normal;
        margin: 4px 2px !important;
        cursor: pointer !important;
        border-radius: 10px !important;
    }
    .st-emotion-cache-mtzh7f:hover {
        border-color: #d00000;
        color: #d00000;
    }
</style>
"""',
    unsafe_allow_html=True
)
# Apply custom CSS to the submit button

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

# Response box
response_container= st.empty()

if submit and uploaded_file is not None:
    try:
        with st.spinner("Processing..."):
            image_data= input_image_setup(uploaded_file)
            response= None
            retries= 3
            while retries > 0 and response is None:
                response =get_gemini_response(input_prompt, image_data, input)
                if response is None:
                    retries-= 1
                    time.sleep(2) # Wait before retrying
            if response:
                response_container.markdown(
                    f"""
                    <div class="response-box">
                        <h2>Response:</h2>
                        <p>{response}</p>
                    </div>
                """
                )
    except Exception as e:
        st.error(f"Error: {e}")

```

```
        """
        unsafe_allow_html=True
    )
    else:
        st.error("Failed to get a response from the API after multiple
attempts.")
    except Exception ase:
        st.error(f"An unexpected error occurred: {e}")

# Footer
st.markdown(
    """
    <div class="footer">
        <p> Helping you manage your diet intelligently.</p>
    </div>
    """
    ,
    unsafe_allow_html=True
)
```

Project Files Submission and Documentation

For the documentation, kindly refer to the link. [Click Here](#)

Project Demonstration

For the demonstration, kindly refer to the link. [Click Here](#)