

REPORT ON
AI-Enhanced Health & Wellness Application

Submitted by

SAADANA SHYAMSUNDER – 11724719
PUJA MADHAV – 11714654
KAVYA KUMARY JAYANTHI NAIR – 11689984
KANAKA SRI BRAHMINI BANDLA - 11595731
Group Number – 5

In partial fulfillment of the credit requirements in
BCIS 5140

Under the guidance of
Dr. Anna Sidorova



DECEMBER 2024

Table Of Contents

Table Of Contents	1
1. Executive Summary	3
2. Objective	5
2.1 Strategic Importance of Solving the Problem.....	5
3. Data Overview & Preparation	7
3.1 Introduction to the Dataset	7
3.1.1 Activity Dataset.....	7
3.1.2 Nutrition Dataset	7
3.2 Data Preparation	8
3.3 Data Cleaning and Processing	8
3.3.1 Big Data Analysis and Modeling	9
3.3.2 Tools and Techniques	9
4. Model Implementation Process	11
4.1 Technology Stack	12
4.2 Core Functionalities and Features	13
4.2.1 Personalized BMI Calculator.....	13
4.2.2 Meal Recommendations:	13
4.2.3 Fitness Activity Recommendations:.....	15
4.2.4 Weekly Meal and Fitness Plan Generator.....	16
4.3 User Interface	18
4.4 Deploying Streamlit via Jupyter and Google Vertex AI.....	21
5 Business Implications and Recommendations.....	27
6. Conclusion	28
7 Additional References.....	29

Table of Figures

Fig 4.1: Architecture diagram	11
Fig 4.2: Function used for calculating BMI.....	13
Fig 4.3: Function used for generating meal response using prompts	14
Fig 4.4: Function used for generating fitness response using prompts.....	16
Fig 4.5: Function used for generating weekly plan response using prompts.....	17
Fig 4.6: Function used for generating weekly fitness plan response using prompts	18
Fig 4.7: Screenshot of the App interface	19
Fig 4.8: App interface Personal details option	19
Fig 4.9: App interface preference option.....	20
Fig 4.10: App output screenshots	21
Fig 4.11: Uploading datasets to Cloud	22
Fig 4.12: Creating new bucket named project group number	22
Fig 4.13: Confirming the required datasets for the application inside bucket.....	23
Fig 4.14: Setting Up Google Vertex AI Workbench for Project Development	24
Fig 4.15: Setting Up of Requirements	25
Fig 4.16: Setting Up of Dockerfile	25
Fig 4.17: Cloudrun Dashboard	26

1. Executive Summary

The AI-Enhanced Health & Wellness Application project aims to revolutionize personal health management by integrating meal planning, dietary tracking, and habit coaching into a single, user-friendly platform powered by AI. The motivation for this solution stems from the growing demand for personalized health tools in a post-pandemic world, where users face challenges managing multiple apps for various health-related activities.

Our application leverages the OpenAI API for intelligent recommendations and natural language interactions, enhancing user experience with tailored dietary suggestions and actionable strategies for habit formation. By utilizing advanced datasets such as MyFitnessPal for nutrition information, and fitness dataset, the system provides a robust framework for data-driven health management. AI models like collaborative filtering and natural language processing (NLP) ensure personalized and accurate recommendations.

Key findings include:

- A unified platform significantly reduces user reliance on disparate apps, improving usability and engagement.
- AI-driven personalization enhances the relevance and effectiveness of dietary and fitness recommendations.
- The integration of habit formation strategies promotes sustained behavioral changes in users.

The project followed a structured methodology, from requirement gathering and dataset preparation to AI model integration and rigorous user acceptance testing (UAT). Early testing demonstrated promising results in user engagement and recommendation accuracy, underscoring the application's potential to redefine health management.

This project matters because it addresses the inefficiencies and fragmented user experience of existing health solutions. By providing a cohesive, intelligent platform, the AI-Enhanced Health & Wellness Application empowers individuals to take control of their health journeys, fostering better habits and improved overall well-being.

Looking ahead, the application will undergo continuous iteration based on user feedback, with plans to expand its feature set to include advanced analytics and cross-platform compatibility.

1.1 Project Motivation/Background

Motivation for the Project:

Many people aspire to lead healthier lives, whether it's through losing weight, improving fitness, or adopting a better diet. However, despite their best intentions, they often face significant challenges. Finding the right meals to prepare or choosing an effective workout routine can be overwhelming. The effort required to research recipes, calculate calorie requirements, and plan fitness activities often deters people from staying consistent.

Existing health and wellness applications claim to help but often fall short of meeting individual needs. Most provide generic recommendations that fail to consider personal factors such as dietary preferences,

specific fitness goals, or health conditions. This lack of personalization leaves users frustrated, as the solutions feel disconnected from their unique circumstances.

Recognizing this gap, we decided to develop a personalized meal and fitness recommendation application. Our goal was to create a platform that simplifies the journey to better health by providing tailored, actionable insights. By incorporating easy-to-use features like pre-built prompts and input fields for height, weight, age, and fitness goals, we eliminate the guesswork. Users don't need to calculate complex metrics or search endlessly for the right information—our application does the work for them. What sets this application apart is its ability to combine structured data processing with AI-powered recommendations. Using advanced technologies like Google Cloud Storage and OpenAI GPT, it delivers highly accurate and engaging meal plans and fitness routines. Instead of just providing numbers or generic advice, the application offers detailed, personalized guidance that users can easily implement in their daily lives.

Moreover, our cloud-based architecture ensures accessibility and scalability, allowing users to interact with the platform seamlessly from anywhere. Whether their goal is weight loss, muscle gain, or maintaining overall wellness, the app adapts to individual needs, making the process intuitive and results-driven. In essence, this project was built to remove the common barriers people face in leading a healthier lifestyle. By combining simplicity, personalization, and advanced AI insights, we aim to empower users to achieve their goals with ease and confidence.

2. Objective

2.1 Strategic Importance of Solving the Problem

Addressing the challenges in health and wellness is more critical than ever, given the growing prevalence of lifestyle-related illnesses and the need for accessible, personalized tools to promote healthier living. The project highlights several key areas of importance in solving this problem, which are elaborated below:

1. Health Empowerment

The project's primary objective is to empower individuals to take control of their health by providing actionable, data-driven insights. By leveraging advanced AI technologies, such as predictive analytics and personalized recommendations, the application enables users to make informed decisions about their nutrition, fitness, and overall well-being. This empowerment fosters self-awareness and accountability, helping individuals adopt healthier behaviors tailored to their unique needs.

2. Sustainability

Health improvement is not a short-term goal but a lifelong journey. The application emphasizes habit formation and long-term wellness strategies, addressing the critical need for sustainable health solutions. By encouraging consistency and providing continuous feedback, the project aims to help users maintain healthy routines and avoid the common pitfalls of short-lived wellness programs. This sustainable approach ensures that users can achieve lasting health benefits rather than temporary results.

3. Technological Advancement

The integration of advanced AI techniques, including OpenAI APIs and Natural Language Processing (NLP), sets this project apart as a benchmark for innovation in the health and wellness domain. These cutting-edge technologies enable the application to deliver highly personalized and interactive experiences, such as real-time feedback, habit tracking, and customized recommendations. By pushing the boundaries of what health applications can achieve, the project contributes to the evolution of digital health tools and enhances their potential impact.

4. Social Relevance

As the rates of lifestyle-related illnesses, such as obesity, diabetes, and cardiovascular diseases, continue to rise globally, personalized health tools play a crucial role in improving public health. This application aligns with broader societal goals by promoting better nutrition, fitness, and lifestyle habits. By addressing individual needs, the project has the potential to create a ripple effect, encouraging communities to adopt healthier behaviors, reducing the burden on healthcare systems, and improving overall quality of life.

5. Bridging Gaps in Existing Systems

Current health and wellness systems often fall short in providing intuitive, accessible, and personalized solutions. Many existing applications lack a comprehensive approach, focusing on singular aspects of health rather than holistic well-being. This project aims to fill those gaps by offering a user-friendly and engaging platform that integrates multiple dimensions of health, including diet, exercise, and mental well-

being. By addressing these shortcomings, the application strives to foster healthier behaviors in a way that is both technologically advanced and practically accessible.

By tackling these critical aspects, the project positions itself as a transformative tool that not only empowers individuals but also contributes to long-term societal well-being. Its emphasis on sustainability, innovation, and social relevance underscores the importance of solving the challenges in health and wellness, ensuring a meaningful and lasting impact.

3. Data Overview & Preparation

3.1 Introduction to the Dataset

Kaggle Link to Dataset (Nutrition Data) – [Click Here!](#)

Kaggle Link to Dataset(Fitness Data) – [Click Here!](#)

Colab Notebook Link – [Click here!](#)

3.1.1 Activity Dataset

This dataset provides information about various physical activities, their caloric expenditure, and indoor/outdoor classification.

Column Name	Description
Activity, Exercise or Sport (1 hour)	Names of various activities, exercises, or sports performed for one hour.
130 lb	Caloric expenditure for individuals weighing 130 lbs during one hour of the activity.
155 lb	Caloric expenditure for individuals weighing 155 lbs during one hour of the activity.
180 lb	Caloric expenditure for individuals weighing 180 lbs during one hour of the activity.
205 lb	Caloric expenditure for individuals weighing 205 lbs during one hour of the activity.
Calories per kg	Caloric expenditure per kilogram of body weight for one hour of the activity.
Indoor/Outdoor	Categorization of activities as Indoor, Outdoor, or Both.

3.1.2 Nutrition Dataset

This dataset contains detailed nutritional information about various food items.

Column Name	Description
Name	Names of food items.
Food Group	The category to which the food belongs (e.g., Meats, Vegetables, Fruits).
Calories	Caloric value of the food item.
Fat (g)	Amount of fat (in grams) in the food item.
Protein (g)	Amount of protein (in grams) in the food item.
Carbohydrate (g)	Amount of carbohydrates (in grams) in the food item.

Column Name	Description
Sugars (g)	Amount of sugars (in grams) in the food item.
Fiber (g)	Amount of dietary fiber (in grams) in the food item.
Cholesterol (mg)	Amount of cholesterol (in milligrams) in the food item.
Sodium (mg)	Amount of sodium (in milligrams) in the food item.
Potassium (mg)	Amount of potassium (in milligrams) in the food item.
Vegetarian	Classifies the food as vegetarian or non-vegetarian.
Cooking Suggestions	Personalized suggestions for cooking methods based on food type.

3.2 Data Preparation

The datasets for this project required significant preparation to ensure suitability for analysis and integration into the AI-Enhanced Health & Wellness Application. Data cleaning, transformation, and classification processes were conducted using Python and Google Cloud Platform's Vertex AI for storage and access.

Big Data Ecosystem and Tools

Cloud Storage: The datasets were stored in Google Cloud Storage, providing secure, scalable, and reliable access for analysis and modeling.

Access and Processing: Vertex AI on Google Cloud Platform (GCP) was utilized for accessing the data, enabling seamless integration with machine learning pipelines and tools.

Processing Frameworks: Pandas were employed for data cleaning and transformation due to its robust capabilities for handling structured data.

3.3 Data Cleaning and Processing

Nutrition Dataset

1. **Irrelevant Columns:** Columns with over 80% missing data were removed to reduce noise and improve efficiency.
2. **Missing Values:** Remaining missing values were imputed with zeros to standardize the dataset and maintain consistency.
3. **Vegetarian Classification:**
 - A predefined set of keywords (e.g., "chicken," "fish," "egg") was used to classify food items as vegetarian or non-vegetarian.
 - Non-vegetarian items containing these keywords were labeled as "No," and others as "Yes."

4. Irrelevant Item Filtering:

- Items unrelated to human consumption (e.g., pet food, baby formula) were removed using another set of filtering keywords.

5. Dynamic Cooking Suggestions:

- Custom suggestions were added for cooking methods based on the item's vegetarian classification and food type (e.g., "Grill or stir-fry with spices for tofu").

6. Storage and Output: The cleaned and enriched dataset was saved back to Google Cloud Storage for further analysis and modeling.

Activity Dataset

1. Activity Classification:

- Keywords for indoor and outdoor activities were defined (e.g., "yoga" and "cycling").
- Each activity was classified as "Indoor," "Outdoor," or "Unclassified" based on the presence of these keywords.

2. Missing Data Handling:

- Missing values were imputed where possible, with non-relevant rows removed.

3. Storage and Output: The processed dataset was saved back to Google Cloud Storage for modeling.

3.3.1 Big Data Analysis and Modeling

• Non-Normal Distribution Handling:

- For numeric features such as calories and protein content, log transformations or scaling were applied to address skewness and ensure model compatibility.

• Outlier Treatment:

- Extreme outliers (e.g., unusually high caloric values) were identified and either capped or removed to maintain data integrity.

• Integration with AI Models:

- The cleaned datasets were fed into AI models for meal and fitness recommendations, leveraging collaborative filtering and natural language processing.

3.3.2 Tools and Techniques

- **Vertex AI:** Facilitated efficient data pipeline integration for real-time model training and prediction.
- **DataFrame Operations:** Pandas was used extensively for slicing, filtering, and enriching data.

- **Cloud Infrastructure:** Google Cloud's scalable architecture supported large-scale data operations and analysis.

The prepared datasets were integral to the success of the AI models, providing clean and structured inputs for personalized recommendations. The use of a cloud-based big data ecosystem ensured efficiency, scalability, and reliability.

4. Model Implementation Process

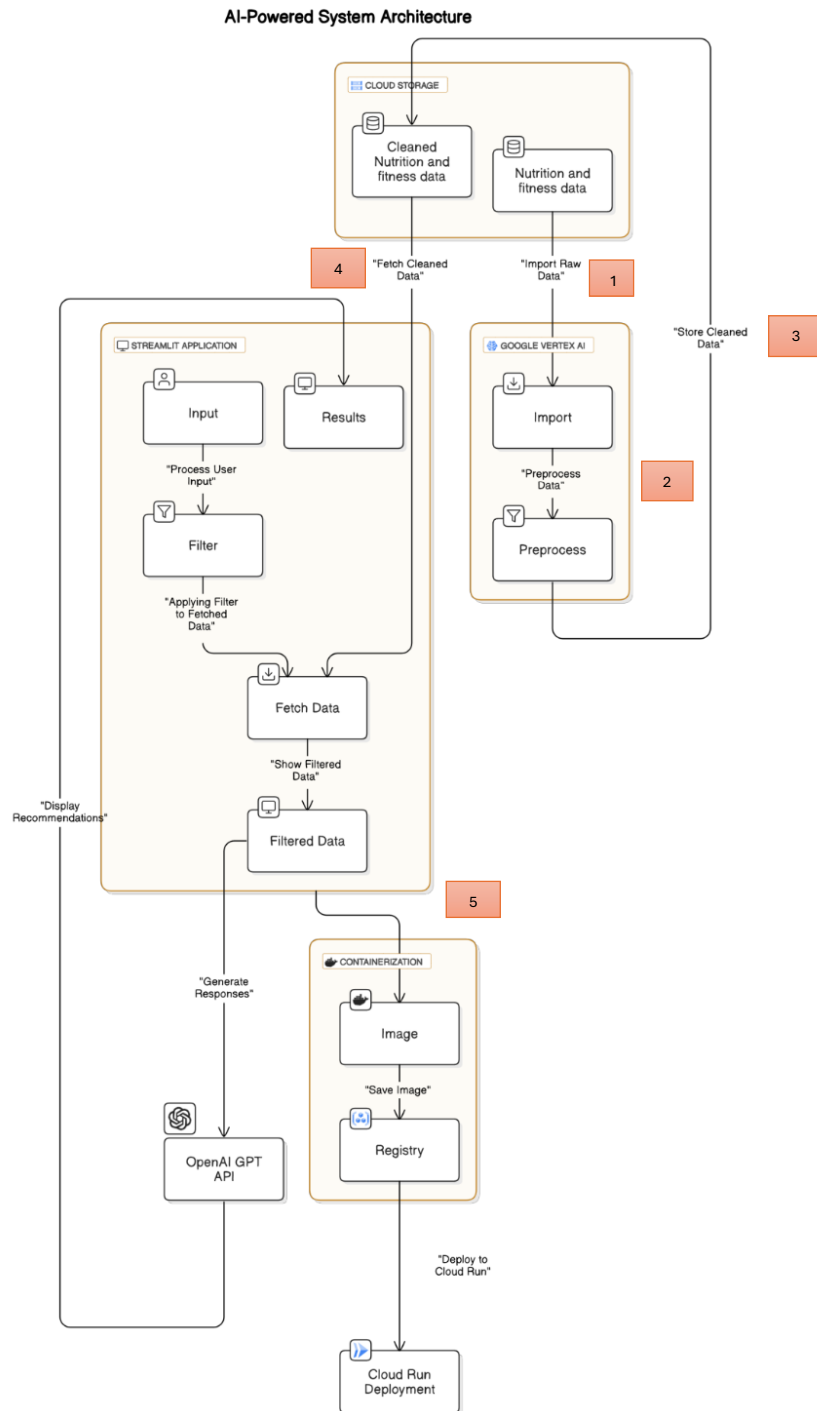


Fig 4.1: Architecture diagram

The AI-Powered Weekly Meal and Fitness Plan application was implemented using Python and Streamlit to create an interactive user interface. The application leverages Google Cloud Platform (GCP) for data access and storage and integrates OpenAI's GPT models to provide personalized recommendations, meal plans, and fitness activities. Below is a detailed breakdown of the implementation process and features:

Connections and Data Flow

Step 1: Cloud Storage

Fitness and nutrition data are initially uploaded to Google Cloud Storage, serving as the system's primary data repository.

Step 2: Google Vertex AI

Raw data is imported from Google Cloud Storage into Vertex AI, where it undergoes cleaning and transformation. This includes handling null values, extracting features, and categorizing data to ensure it is high-quality and ready for further processing.

Step 3: Cloud Storage (Cleaned Data)

After pre-processing, the refined data is uploaded back to Cloud Storage under a new "Cleaned Data" repository.

Step 4 Streamlit Application with OpenAI Integration

The Streamlit Application fetches cleaned data from Google Cloud Storage, processes user inputs (e.g., dietary preferences, calorie limits, fitness goals), and filters the dataset accordingly. The filtered data is then sent to the OpenAI GPT API, which generates personalized meal plans and fitness suggestions in natural language. These recommendations are displayed interactively within the application, providing users with tailored insights and engaging experience.

Step 5 Streamlit Application Deployment to Cloud Run

The Streamlit Application is packaged into a Docker image and stored in the Google Container Registry. It is then deployed to Google Cloud Run, a serverless platform, making the application scalable and publicly accessible via a web link. This deployment ensures seamless real-time interaction for users, enabling them to fetch data, apply filters, and view personalized recommendations efficiently.

4.1 Technology Stack

- **Streamlit:** Chosen for its simplicity in creating dynamic, user-friendly web applications. It supports rapid prototyping and visualizations, which were essential for the app.
- **Google Cloud Platform (GCP):** Used for securely storing and accessing datasets. Datasets were hosted in a Google Cloud Storage bucket and accessed dynamically.

- **OpenAI GPT-3.5 Turbo:** Integrated to provide personalized meals and fitness recommendations using natural language responses.
- **Python Libraries:** Key libraries used include:
 - pandas: For data manipulation and filtering.
 - google.cloud.storage: For accessing datasets stored in GCP.
 - random: For generating random samples of meals and activities.

4.2 Core Functionalities and Features

4.2.1 Personalized BMI Calculator

This project uses **BMI**, **age**, and **gender** to personalize recommendations for nutrition and fitness. **BMI** helps tailor calorie needs, meal plans, and workout intensity based on weight goals. **Age** ensures recommendations fit different life stages, like higher activity for younger users or low-impact routines for older individuals. **Gender** adjusts plans for metabolic differences, nutritional needs, and specific health considerations. Together, these factors create customized meal plans, targeted workouts, and actionable health advice, ensuring recommendations are practical and suited to each user's unique profile.

```
Function to calculate BMI
def calculate_bmi(weight, height):
    return weight / (height / 100) ** 2
```

Fig 4.2: Function used for calculating BMI

4.2.2 Meal Recommendations:

The Meal Recommendations feature is a key component of the AI-Powered Weekly Meal and Fitness Plan Application, designed to provide users with personalized dietary suggestions. This functionality aligns meal options with the user's health goals, preferences, and restrictions, ensuring a seamless and tailored experience.

Purpose

The Meal Recommendations simplify the process of planning meals by dynamically generating options that cater to individual needs. It considers dietary preferences, caloric goals, and exclusions, providing users with actionable recipes for convenient preparation.

Features

1. Dietary Preferences

- Description: Users can select their dietary type:
 - Vegetarian: Plant-based meal options.
 - Non-Vegetarian: Meals including animal-based ingredients.

- Implementation: The system classifies meals based on predefined keywords (e.g., "chicken," "tofu") and filters options accordingly.
 - Benefit: Ensures meals align with cultural or personal dietary practices.
- 2. Calorie Limit**
- Description: Users can set a maximum daily calorie intake to align with their dietary goals.
 - Implementation: The application filters meals from the dataset that fall below the specified calorie threshold, but not less than the minimum calorie customer has to consume based on the Calorie consumption calculation made by the model.
 - Benefit: Supports users in maintaining a calorie deficit or adhering to caloric discipline.
- 3. Goal-Specific Recommendations**
- Description: Users can tailor meal suggestions based on the following health goals:
 - Weight Loss: Low-calorie meal options.
 - High Protein: Protein-rich meals to support muscle growth or maintenance.
 - Low Carb: Meals with reduced carbohydrate content for low-carb diets.
 - Implementation: Filters are applied based on thresholds such as protein >15g or carbohydrates <50g.
 - Benefit: Helps users align their diet with specific health objectives.
- 4. Exclusions**
- Description: Users can exclude specific ingredients or meals to accommodate allergies, dislikes, or dietary restrictions.
 - Implementation: The application dynamically removes any meals containing the excluded keywords from the dataset.
 - Benefit: Ensures meal suggestions are personalized and safe for consumption.
- 5. Dynamic Recipes**
- Description: Each recommended meal is accompanied by a detailed recipe, including:
 - Ingredients: A comprehensive list of measurements.
 - Cooking Instructions: Step-by-step preparation guidelines.
 - Implementation: OpenAI's GPT is used to generate conversational, user-friendly recipes.
 - Benefit: Empowers users to prepare meals confidently and creatively, making healthy eating accessible.

```
# Function to generate a natural language response for meal recommendations
def generate_meal_response(meal_name, calories, protein, carbs, goal, preference, age, gender, goal_weight):
    print(f"age: {age}")
    prompt = (
        f"You are a friendly dietitian. A {age}-year-old {gender} has the goal of reaching weight {goal_weight} kg with a focus on '{goal}', and prefers a '{preference}' diet. "
        f"Recommend a meal named '{meal_name}' with {calories} calories, {protein} grams of protein, and {carbs} grams of carbohydrates. "
        f"Explain why this meal is suitable in a conversational tone."
    )
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        max_tokens=200,
        temperature=0.7
    )
    return response["choices"][0]["message"]["content"].strip()
```

Fig 4.3: Function used for generating meal response using prompts

4.2.3 Fitness Activity Recommendations:

The Fitness Activity Recommendations feature is a critical part of the AI-Powered Weekly Meal and Fitness Plan Application, designed to provide personalized physical activity suggestions. It ensures that users can integrate appropriate fitness routines into their wellness plans by considering factors such as weight, workout intensity, and personal preferences or restrictions.

Purpose

The purpose of this feature is to tailor fitness activity recommendations to individual users, ensuring that the suggested exercises are effective, enjoyable, and feasible. By dynamically filtering activities and providing detailed explanations, this feature helps users make informed decisions about their fitness regimes.

Features

1. Calorie Burn Estimation

- **Description:** The feature estimates calories burned for different fitness activities based on the user's weight.
- **Implementation:**
 - The dataset includes calorie burn rates for various activities, categorized by weight classes (e.g., 130 lb, 155 lb).
 - Users' weight inputs are matched with the appropriate column to calculate calories burned for a given activity.
- **Benefit:** Allows users to select activities that align with their caloric expenditure goals, supporting weight management and fitness objectives.

2. Intensity Levels

- **Description:** Users can select their desired workout intensity:
 - Light: Activities with lower energy requirements (e.g., yoga, walking).
 - Moderate: Activities with moderate energy output (e.g., cycling, hiking).
 - Vigorous: High-intensity activities (e.g., running, swimming).
- **Implementation:**
 - Activities are categorized based on calorie burn thresholds for each weight class:
 - Light: <300 kcal/hour
 - Moderate: 300–600 kcal/hour
 - Vigorous: >600 kcal/hour
 - Users' selections filter the activity dataset to match their intensity level.
- **Benefit:** Helps users choose activities that align with their current fitness levels and goals, avoiding overexertion or insufficient effort.

3. Exclusions

- **Description:** Users can exclude specific activities that they dislike or are unable to perform.
- **Implementation:**
 - Users provide a list of activities or keywords to exclude.
 - The application dynamically removes activities matching the exclusions from the dataset.
- **Benefit:** Ensures that recommendations are personalized and relevant, improving adherence and user satisfaction.

4. Dynamic Explanations

- **Description:** For each suggested activity, the system generates a natural language explanation of why it is suitable for the user's goals and intensity level.
- **Implementation:**
 - OpenAI's GPT model is used to create friendly, conversational descriptions of the activities.
 - Example: "Yoga is an excellent light-intensity workout that improves flexibility and reduces stress. It's perfect for beginners or as a recovery exercise."
- **Benefit:** Provides users with actionable insights and context, making recommendations more engaging and informative.

```
def generate_fitness_response(activity, calories_burned, intensity, age, gender, goal_weight):  
    prompt = (  
        f"You are a fitness coach. A {age}-year-old {gender} has the goal weight of {goal_weight} kg with a focus on '{goal}', and is looking for a '{intensity}' intensity workout  
        f"Recommend an activity '{activity}' that burns {calories_burned} calories per hour. "  
        f"Explain why this activity is suitable for their fitness level in a friendly and motivating tone."  
    )  
    response = openai.ChatCompletion.create(  
        model="gpt-3.5-turbo",  
        messages=[  
            {"role": "system", "content": "You are a helpful assistant."},  
            {"role": "user", "content": prompt}  
        ],  
        max_tokens=200,  
        temperature=0.7  
    )  
    return response["choices"][0]["message"]["content"].strip()
```

Fig 4.4: Function used for generating fitness response using prompts

4.2.4 Weekly Meal and Fitness Plan Generator

The Weekly Meal and Fitness Plan Generator is a core feature of the AI-Powered Weekly Meal and Fitness Plan Application, designed to provide users with a comprehensive, structured plan for their dietary and fitness goals. This feature generates a personalized weekly schedule, ensuring variety and alignment with the user's preferences and goals.

Purpose

The purpose of this feature is to enable users to maintain consistency in their health and wellness journey by offering a structured, balanced, and goal-oriented weekly plan. By dynamically combining meal and fitness recommendations, the feature simplifies planning and enhances adherence to a healthier lifestyle.

Features

1. Randomized Selection

- **Description:**
 - The system uses randomized algorithms to select meals and fitness activities from the filtered datasets.
 - Ensures that the same meal or activity is not repeated within the week.
- **Implementation:**
 - Random sampling is applied to datasets of meals and fitness activities that match the user's preferences, goals, and exclusions.
 - Used a "memory" mechanism to avoid repetition by keeping track of previously selected options.
- **Benefit:**
 - Introduces variety, preventing monotony in meal choices and fitness routines, which increases user engagement and motivation.

2. Daily Recommendations

- **Description:**
 - The feature generates specific meal and fitness activity suggestions for each day of the week.
- **Implementation:**
 - The generator assigns one meal and one fitness activity for each day (Monday through Sunday) based on the user's inputs.
 - Each day's recommendation includes:
 - Meal name, calorie content, and protein/carb details.
 - Fitness activity name and calorie burn estimate for the user's weight and intensity level.
- **Benefit:**
 - Provides users with a clear, day-by-day structure that simplifies decision-making and supports consistent adherence to their health goals.

3. Integration with OpenAI GPT

- **Description:**
 - Each meal and activity suggestion is enhanced with additional details dynamically generated by OpenAI GPT.
 - For meals, GPT provides:
 - A detailed recipe with ingredients and cooking instructions.
 - An explanation of why the meal is suitable for the user's goals.
 - For fitness activities, GPT generates:
 - A brief description of the activity and its benefits.
- **Implementation:**
 - GPT integration is triggered for every selected meal and activity, using carefully crafted prompts to ensure relevant and personalized outputs.
- **Benefit:**
 - Adds depth and context to the recommendations, making them actionable and engaging for users.

```

# Function to generate a weekly plan with breakfast, lunch, and dinner
def generate_detailed_weekly_plan(df_meals, df_fitness, goal, preference, intensity, calorie_limit, carb_threshold, exclude_meals, exclude_activities, gender, age, goal_weight):
    # Filter meals and fitness activities
    meal_df = filter_meals(df_meals, preference, goal, calorie_limit, carb_threshold, exclude_meals)
    fitness_df = filter_fitness(df_fitness, weight, intensity, exclude_activities)

    if meal_df.empty or fitness_df.empty:
        return "No suitable meal or fitness activity found.", []

    weekly_plan = []

    for day in ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]:
        daily_meals = []
        chat_responses = {"meals": [], "fitness": None}

        for meal_time in ["Breakfast", "Lunch", "Dinner"]:
            # Randomly select a meal
            meal = meal_df.sample(1).iloc[0]
            daily_meals.append(meal)

            # Generate a ChatGPT response for the meal
            chat_response = generate_meal_response(
                meal_name=meal['name'],
                calories=meal['Calories'],
                protein=meal['Protein (g)'],
                carbs=meal['Carbohydrate (g)'],
                goal=goal,
                preference=preference,
                age=age,
                gender=gender,
                goal_weight=goal_weight # Pass goal_weight here
            )
            chat_responses["meals"].append({"time": meal_time, "response": chat_response})

    return weekly_plan, chat_responses
```

Fig 4.5: Function used for generating weekly plan response using prompts

```

# Randomly select a fitness activity
activity = fitness_df.sample(1).iloc[0]

# Generate a ChatGPT response for the fitness activity
# Generate a ChatGPT response for the fitness activity
fitness_response = generate_fitness_response(
    activity=activity['Activity, Exercise or Sport (1 hour)'],
    calories_burned=activity['155 lb'],
    intensity=intensity,
    age=age,
    gender=gender,
    goal_weight=goal_weight
)

chat_responses["fitness"] = fitness_response

# Append to weekly plan
weekly_plan.append({
    "day": day,
    "meals": daily_meals,
    "activity": activity,
    "chat_responses": chat_responses
})

return weekly_plan

```

Fig 4.6: Function used for generating weekly fitness plan response using prompts

4.3 User Interface

The User Interface (UI) of the AI-Powered Weekly Meal and Fitness Plan Application is a critical component that allows users to seamlessly interact with the application. Built using Streamlit, the UI is designed to be intuitive and user-friendly, catering to both technical and non-technical users.

The UI provides a platform for users to input their personal details, dietary preferences, fitness goals, and exclusions. It facilitates the interaction between the user and the backend, allowing users to receive personalized weekly meal and fitness plans with minimal effort.

Benefits:

- **Ease of Use:** Ensures that users can navigate the application without requiring technical expertise.
- **Interactivity:** Allows users to actively engage with the application by providing inputs and receiving dynamic outputs.
- **Customization:** Offers tailored options to suit individual user needs, improving the relevance and value of the recommendations.
- **Streamlined Workflow:** Guides users step-by-step through the process of entering data, generating plans, and viewing results.

User Details

Age

25

-

+

Gender

Female

▼

Weight (kg)

65.00

-

+

Height (cm)

160.00

-

+

Goal Weight (kg)

60.00

-

+

Your BMI: 25.39

Goal

High Protein

▼

Diet Preference

☐ Vegetarian
☒ Non-Vegetarian

AI-Powered Weekly Meal and Fitness Planner

Generate Weekly Plan

Monday

Meals:

- Breakfast: Lamb Ribs (359.0 cal, 21.12g protein)
- Lunch: Chicken Thigh Fried Coated Skin / Coating Not Eaten From Pre-Cooked (192.0 cal, 24.07g protein)
- Dinner: Ham Fresh Cooked Lean And Fat Eaten (271.0 cal, 26.61g protein)

Fitness Activity:

Track and field (shot, discus) - 281 cal/hour

Tuesday

Meals:

- Breakfast: Chicken Broilers Or Fryers Thigh Meat Only Cooked Rotisserie Original Seasoning (196.0 cal, 24.06g protein)

Fig 4.7: Screenshot of the App interface

How the UI Guides Users

1. Step 1: Enter Personal Details

- Users input weight, height, and goal weight through st.number_input.
- BMI is calculated in real-time and displayed for user reference.

User Details

Age

25

-

+

Gender

Female

▼

Weight (kg)

65.00

-

+

Height (cm)

160.00

-

+

Goal Weight (kg)

60.00

-

+

Your BMI: 25.39

Goal

High Protein

▼

Diet Preference

☐ Vegetarian
☒ Non-Vegetarian

AI-Powered Weekly Meal and Fitness Planner

Generate Weekly Plan

Monday

Meals:

- Breakfast: Lamb Ribs (359.0 cal, 21.12g protein)
- Lunch: Chicken Thigh Fried Coated Skin / Coating Not Eaten From Pre-Cooked (192.0 cal, 24.07g protein)
- Dinner: Ham Fresh Cooked Lean And Fat Eaten (271.0 cal, 26.61g protein)

Fitness Activity:

Track and field (shot, discus) - 281 cal/hour

Tuesday

Meals:

- Breakfast: Chicken Broilers Or Fryers Thigh Meat Only Cooked Rotisserie Original Seasoning (196.0 cal, 24.06g protein)

Fig 4.8: App interface Personal details option

2. Step 2: Choose Preferences

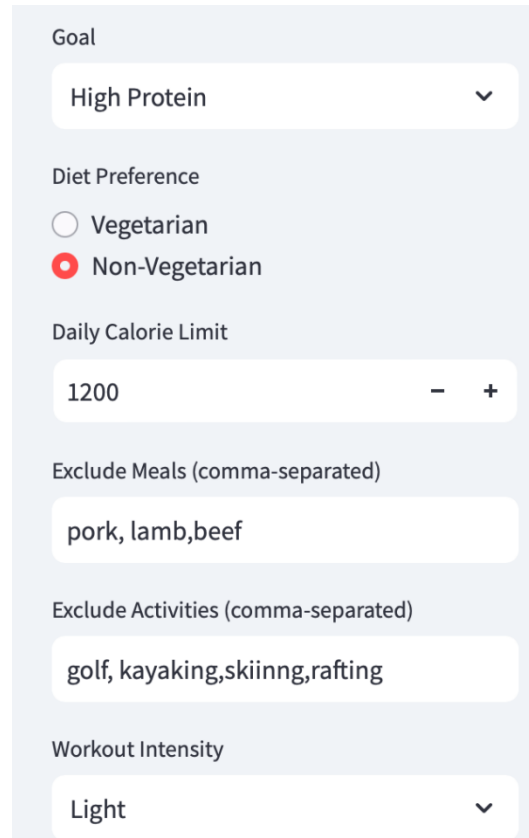
- Dietary preferences (vegetarian/non-vegetarian) are captured using st.radio.
- Calorie limits and carbohydrate thresholds are set using st.number_input and sliders.
- Fitness goals and workout intensity are selected via st.selectbox.

3. Step 3: Specify Exclusions

- Users input specific meals or activities to exclude using `st.text_input`.

4. Step 4: Generate Plan

- Clicking the `st.button` triggers the generation of a tailored weekly plan.
- Outputs are displayed dynamically, including meal details, fitness activity descriptions, and actionable steps (e.g., recipes).



The image shows a mobile app interface for setting preferences. It includes a 'Goal' dropdown set to 'High Protein', 'Diet Preference' with radio buttons for 'Vegetarian' and 'Non-Vegetarian' (the latter is selected), a 'Daily Calorie Limit' spinner set to 1200, and two text input fields for 'Exclude Meals' (containing 'pork, lamb,beef') and 'Exclude Activities' (containing 'golf, kayaking,skiinng,rafting'). At the bottom is a 'Workout Intensity' dropdown set to 'Light'.

Goal

High Protein

Diet Preference

☐ Vegetarian

☒ Non-Vegetarian

Daily Calorie Limit

1200

Exclude Meals (comma-separated)

pork, lamb,beef

Exclude Activities (comma-separated)

golf, kayaking,skiinng,rafting

Workout Intensity

Light

Fig 4.9: App interface preference option

AI-Powered Weekly Meal and Fitness Planner

Generate Weekly Plan

Monday

Meals:

- Breakfast: Lamb Ribs (359.0 cal, 21.12g protein)
- Lunch: Chicken Thigh Fried Coated Skin / Coating Not Eaten From Pre-Cooked (192.0 cal, 24.07g protein)
- Dinner: Ham Fresh Cooked Lean And Fat Eaten (271.0 cal, 26.61g protein)

Fitness Activity:

Track and field (shot, discus) - 281 cal/hour

Tuesday

Meals:

- Breakfast: Chicken Broilers Or Fryers Thigh Meat Only Cooked Rotisserie Original Seasoning (196.0 cal, 24.06g protein)

Generate Fitness Details for an Activity

Enter the activity name to generate fitness details:

table tennis

Fitness Details:

Table tennis, also known as ping pong, is a fantastic choice for a light intensity workout that can help you burn around 281 calories per hour. This activity is perfect for those looking to get moving and have fun at the same time!

Table tennis involves quick movements, hand-eye coordination, and agility, making it a great way to get your heart rate up without feeling too strenuous. It's a low-impact sport, which means it's gentle on your joints and muscles, making it suitable for individuals of all fitness levels.

Additionally, playing table tennis can improve your reflexes, concentration, and overall mental agility. It's a social activity that you can enjoy with friends or family, adding an element of fun and competition to your workout routine.

So, if you're looking for a light intensity activity that can help you burn calories while having a great time, table tennis is a fantastic choice! Give it a try and enjoy the benefits of this engaging and enjoyable workout.

Generate Recipe for a Meal

Enter the meal name to generate a recipe:

Bacon Cheeseburger

Recipe:

Bacon Cheeseburger with 1 Small Patty:

Ingredients:

- 1 small ground beef patty (approximately 2 oz)
- 2 slices of bacon
- 1 slice of cheese
- 1 hamburger bun
- Lettuce
- Tomato slices
- Pickles
- Ketchup
- Mustard
- Salt and pepper
- Cooking spray or oil

Instructions:

1. Preheat a skillet or grill over medium-high heat. If using a skillet, lightly coat it with cooking spray or oil.

Fig 4.10: App output screenshots

4.4 Deploying Streamlit via Jupyter and Google Vertex AI

Deployment process involves running the **Streamlit application** through **Jupyter Notebook**. This approach utilizes **Google Cloud Storage (GCS)** for data access and **Google Vertex AI** for hosting the application. Here’s a detailed explanation tailored for your report:

1. Storing Datasets in Google Cloud Storage

Google Cloud Storage serves as the central repository for datasets, enabling seamless integration with the application.

Steps:

1. Upload Data to GCS:

- Use the google-cloud-storage library to upload datasets from the Jupyter environment.

```
from google.cloud import storage
import pandas as pd

# Define the bucket name and file path
bucket_name = "ai_project_group_5" # Replace with your bucket name
destination_blob_name = "cleaned_nutrition_data.csv" # Desired file name in GCS

# Initialize the GCS client
client = storage.Client()

def save_dataframe_to_gcs(df, bucket_name, destination_blob_name):
    """Save a Pandas DataFrame to Google Cloud Storage as a CSV."""
    # Save the DataFrame to a temporary CSV file
    temp_csv_path = "/tmp/cleaned_nutrition_data.csv"
    df.to_csv(temp_csv_path, index=False)

    # Upload the CSV file to GCS
    bucket = client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)
    blob.upload_from_filename(temp_csv_path)

    print(f"File uploaded to {bucket_name}/{destination_blob_name}")

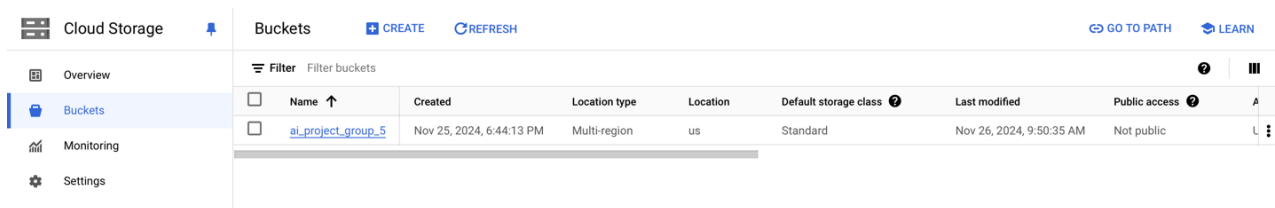
# Save the cleaned DataFrame to GCS
save_dataframe_to_gcs(df_cleaned, bucket_name, destination_blob_name)
```

1]

· File uploaded to ai_project_group_5/cleaned_nutrition_data.csv

Fig 4.11: Uploading datasets to Cloud

2. Verify Upload: Confirm the uploaded files are present in the bucket:



Cloud Storage		Buckets		CREATE	REFRESH	GO TO PATH	LEARN
Overview	Buckets	Monitoring	Settings				
Filter buckets							
<input type="checkbox"/>	Name ↑	Created	Location type	Location	Default storage class ⓘ	Last modified	Public access ⓘ
<input type="checkbox"/>	ai_project_group_5	Nov 25, 2024, 6:44:13 PM	Multi-region	us	Standard	Nov 26, 2024, 9:50:35 AM	Not public

Fig 4.12: Creating new bucket named project group number

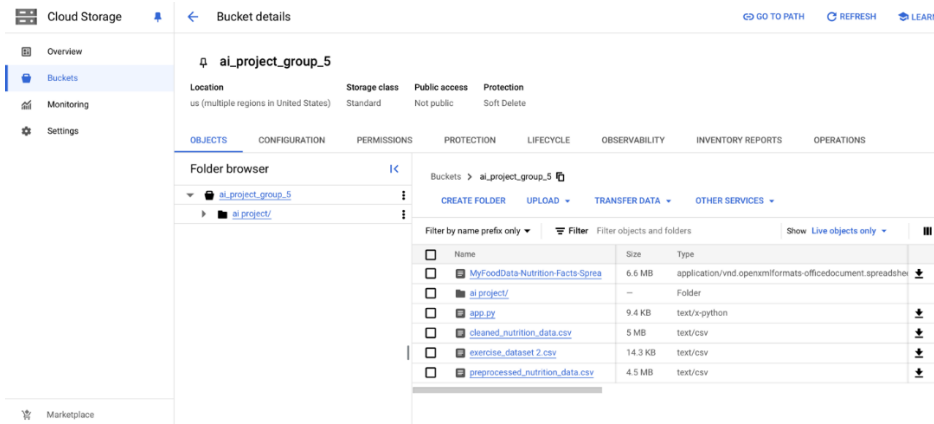


Fig 4.13: Confirming the required datasets for the application inside bucket

2. Accessing Data in the Application

Dynamic Data Loading:

The Streamlit app reads datasets dynamically from GCS during runtime.

a. Define a Function to Load Data:

b. Ensure Dataset Accessibility:

- Use Google Cloud authentication credentials in the Jupyter Notebook to ensure secure access to the storage bucket.

Development of the Streamlit Application

The application was designed to provide dynamic meal and fitness plans with advanced features:

1. Development Environment:

- Used Jupyter Notebook within Google Vertex AI Workbench for development.
- Built an interactive Streamlit app (app.py) to generate:
 - Weekly meal plans.
 - Fitness activity recommendations.
 - Recipes and explanations using OpenAI GPT integration.

2. Key Features of the App:

- Real-time recommendations tailored to user inputs.
- GPT-powered recipe generation and activity descriptions.
- Dynamic and interactive interface for better user engagement.

3. Integrating with Google Vertex AI

The top screenshot displays the Vertex AI console interface. On the left is a sidebar with navigation options: TOOLS (Dashboard, Model Garden, Pipelines), NOTEBOOKS (Colab Enterprise, Workbench), VERTEX AI STUDIO, BUILD WITH GEN AI (Extensions), DATA (Feature Store, Datasets, Labeling tasks), and MODEL DEVELOPMENT. The main panel shows '1 instance selected' with buttons for START, RESET, STOP, and DELETE. Below this, there are tabs for INSTANCES, EXECUTIONS, and SCHEDULES. The 'INSTANCES' tab is active, showing a table of instances. The table has columns for Instance name, Zone, Auto upgrade, Version, Machine Type, GPUs, and Owner. One instance is listed: 'instance-20241126-090639' in the 'us-central1-a' zone, version 'M126', machine type 'Efficient Instance: 4 vCPUs, 16 GB RAM', and owned by '440820212665-compute@developer.gserviceaccount.com'. Below the table, there is a section for 'Workbench Instances have JupyterLab 3 pre-installed and are configured with GPU-enabled machine learning frameworks. Learn more'.

The bottom screenshot shows a JupyterLab notebook environment. The top bar indicates 'instance-20241126-090639' and 'e2-standard-4'. The left sidebar shows a file explorer with 'Project group 5.ipynb' selected. The main area displays a code cell with the command `!pip install google-cloud-storage pandas`. The output of the command is shown below the code cell, indicating that the requirements are already satisfied for the specified versions of the packages.

Instance name	Zone	Auto upgrade	Version	Machine Type	GPUs	Owner
instance-20241126-090639	us-central1-a	—	M126	Efficient Instance: 4 vCPUs, 16 GB RAM	None	440820212665-compute@developer.gserviceaccount.com

```
[1]: !pip install google-cloud-storage pandas

Requirement already satisfied: google-cloud-storage in /opt/conda/lib/python3.10/site-packages (2.14.0)
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages (2.2.3)
Requirement already satisfied: google-auth<3.0dev,>=2.23.3 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.36.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (1.34.1)
Requirement already satisfied: google-cloud-core<3.0dev,>=2.3.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.4.1)
Requirement already satisfied: google-resumable-media>=2.6.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.7.2)
Requirement already satisfied: requests<3.0dev,>=2.18.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (2.32.3)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /opt/conda/lib/python3.10/site-packages (from google-cloud-storage) (1.6.0)
Requirement already satisfied: numpy>=1.22.4 in /opt/conda/lib/python3.10/site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.10/site-packages (from pandas) (2024.2)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in /opt/conda/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-cloud-storage) (1.66.0)
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<4.0.0dev,>=3.19.5 in /opt/conda/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-cloud-storage) (3.20.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0dev,>=2.23.3->google-cloud-storage) (5.5.0)
Requirement already satisfied: oauthlib>=3.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3.0dev,>=2.23.3->google-cloud-storage) (3.2.1)
```

Note: This figure illustrates the setup and configuration of the Google Vertex AI Workbench for project execution. The first image shows the installation of necessary Python libraries (google-cloud-storage and pandas) in a Jupyter Notebook environment, ensuring the required dependencies are in place for cloud storage operations and data processing. The second image depicts the Vertex AI Workbench interface, highlighting the creation of a compute instance with 4 vCPUs and 16 GB RAM for executing machine learning workflows, demonstrating the infrastructure used for efficient project execution

Fig 4.14: Setting Up Google Vertex AI Workbench for Project Development

4. Containerization with Docker

To ensure seamless deployment, the app was containerized using Docker:

1. Preparation:

- Created a requirements.txt file to specify dependencies.

```
streamlit==1.40.2
openai==0.27.8
pandas==2.2.3
google-cloud-storage==2.14.0
pyngrok==7.2.1
numpy>=1.22.4
```

Fig 4.15: Setting Up of Requirements

- Wrote a Dockerfile to containerize the application.

Docker file

```
# Use an official Python runtime as the base image
FROM python:3.9-slim

# Set environment variables
ENV PYTHONUNBUFFERED=1 \
    PYTHONONWRITEBYTECODE=1

# Set the working directory in the container
WORKDIR /app

# Copy application files to the working directory
COPY app.py /app/
COPY requirements.txt /app/

# Install system dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc \
    && apt-get clean && rm -rf /var/lib/apt/lists/*

# Install Python dependencies
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

# Expose the port the app runs on
EXPOSE 8080

# Command to run the application
CMD ["streamlit", "run", "app.py", "--server.port=8080", "--server.address=0.0.0.0"]
```

Fig 4.16: Setting Up of Dockerfile

2. Dockerfile Overview:

- Specified a lightweight Python base image.
- Installed required dependencies from requirements.txt.
- Configured the app to listen on the port specified by the PORT environment variable.

3. Building and Tagging the Docker Image:

- Tagged the image for Google Container Registry (GCR):
- Uploaded the image to GCR for deployment:

4. Deployment to Google Cloud Run

Google Cloud Run was used to host the containerized application for public access:

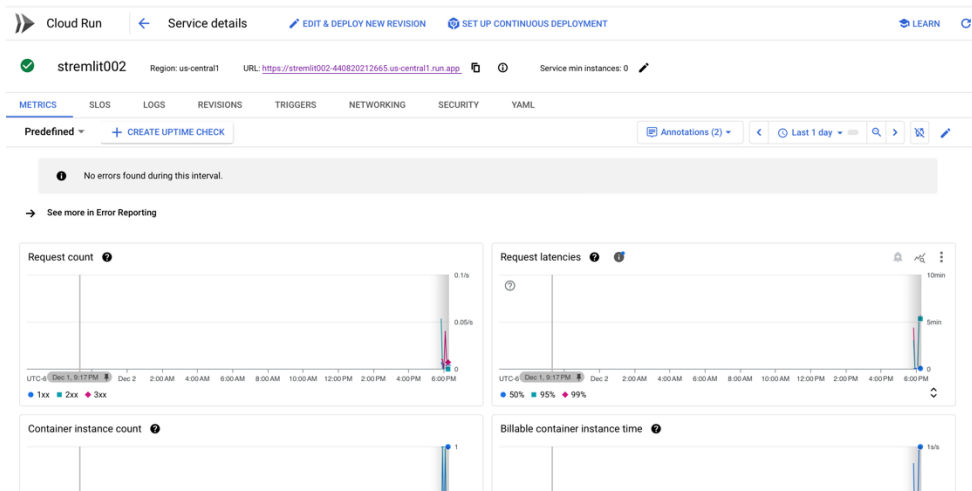


Fig 4.17: Cloudrun Dashboard

1. Cloud Run Configuration:

- The application was configured to listen on the port specified by the PORT environment variable.
- The deployment ensured serverless hosting with auto-scaling capabilities.

2. Verification:

- Verified successful deployment through the public URL generated by Cloud Run.
- Tested the application to ensure proper functioning of all features, including GPT-based recommendations.

Benefits of Using Google Cloud Run

- **Serverless Infrastructure:** Eliminated the need to manage servers manually.
- **Automatic Scaling:** The app scales based on traffic, optimizing resource usage and cost.
- **Simplified Deployment:** Cloud Run made it easy to deploy and manage the containerized application.

5 Business Implications and Recommendations

The AI-Powered Weekly Meal and Fitness Plan App is designed to make health and wellness more personalized and engaging. By understanding user preferences, it creates tailored meal plans and fitness routines that boost satisfaction and encourage regular use. The app can grow by offering more options, like gluten-free, vegan, or keto diets, and filters for specific groups like seniors or kids. Using Google Cloud for data storage keeps everything scalable and consistent while keeping costs in check by archiving older data and focusing on what users access most. Dynamic content, such as recipes and tips powered by OpenAI, makes the experience fun and interactive. Adding multilingual support can help the app expand to global markets and create customized marketing campaigns.¹

With more people focusing on health trends like high-protein diets and sustainable fitness, the app can partner with fitness brands or nutrition companies to offer recommended products. Educational content like blogs or videos can also make the app a go-to resource for wellness advice². Data analytics play a big role, helping track what users like and improving recommendations to meet their needs. Making the app available on mobile and adding offline features ensures it's accessible to everyone, even with limited internet.

Looking ahead, the app could collect more insights about user behavior to improve its algorithms and recommendations. New features like allergy detection, regional cuisines, and activity tracking can make the experience even more personal, and partnerships with health experts can boost its credibility. Connecting with wearable devices like Apple Watch or Fitbit could provide real-time fitness feedback. Marketing can focus on personalization, using real success stories to attract more users. Regular updates to the app's data and features, guided by user feedback, will keep it fresh and relevant. These steps will make the app a trusted and comprehensive tool for anyone looking to improve their health and wellness.

-
1. Samsung Newsroom. "Samsung Announces Global Launch of Samsung Food, an AI-Powered Personalized Food and Recipe Service." Available at: <https://news.samsung.com/global/samsung-announces-global-launch-of-samsung-food-an-ai-powered-personalized-food-and-recipe-service>
 2. Wekerle, M. "How Health and Wellness Apps are Reshaping Consumer Engagement." Forbes. Available at: <https://www.forbes.com/sites/healthcare/health-wellness-apps-reshaping-consumer-engagement>

6. Conclusion

The AI-Powered Weekly Meal and Fitness Plan Application showcases how advanced technology can revolutionize health and wellness by providing personalized, actionable, and engaging recommendations. By leveraging robust backend logic, efficient data preprocessing, and cloud storage with Google Cloud, the application ensures scalability and consistent access for users. OpenAI GPT integration enhances the user experience by dynamically generating meal descriptions, recipes, and fitness activity explanations, creating a highly engaging and practical tool. The user-friendly interface simplifies the input process, with intuitive components like number inputs, dropdown menus, and real-time BMI calculations, making it accessible for users of all technical backgrounds.

This application is designed specifically for nutrition and fitness, making it far more focused and practical than a general tool like ChatGPT. One key difference is that it doesn't rely on you to know exactly what to ask or input. Most of us don't know the precise details, like calorie needs or fitness metrics, to type into ChatGPT. Instead, this app offers pre-built input fields, prompts, and dropdowns for things like height, weight, dietary preferences, and fitness goals. This makes it much easier and more intuitive to use, guiding you step by step to get the most personalized and accurate recommendations.

This application empowers users to take control of their health goals by offering tailored weekly meal and fitness plans that align with their preferences and objectives, promoting consistency and sustainable habits. Its scalable design supports a growing user base and positions the project as a valuable solution in the health-tech space. With the increasing demand for personalized wellness tools, this application has strong potential for market expansion, monetization, and partnerships, making it a forward-thinking and impactful innovation in the wellness industry.

7 Additional References

1. Colab Notebook Link – [Click here!](#)
2. Google Cloud. (n.d.). *Cloud Storage Documentation*. Retrieved from <https://cloud.google.com/storage/docs>
3. OpenAI. (n.d.). *ChatGPT API Documentation*. Retrieved from <https://platform.openai.com/docs/>
4. Streamlit. (n.d.). *Streamlit API Reference*. Retrieved from <https://docs.streamlit.io/library>
5. Pandas Development Team. (2023). *Pandas Documentation*. Retrieved from <https://pandas.pydata.org/docs/>
6. Fitness and Wellness Trends. (2023). *The Rise of Personalized Fitness and Nutrition Plans*. Retrieved from <https://www.healthline.com>
7. Kaggle. (n.d.). *Public Datasets for Nutrition and Fitness*. Retrieved from <https://www.kaggle.com>
8. Streamlit Community. (2023). *Building Applications for Wellness Using Streamlit*. Retrieved from <https://discuss.streamlit.io>
9. American Council on Exercise (ACE). (n.d.). *Caloric Burn Data by Weight Class*. Retrieved from <https://www.acefitness.org>

ChatGPT was utilized to structure, refine, and enhance the project report and presentation. It provided guidance on summarizing data analytics techniques, crafting professional narratives, and ensuring clear explanations of complex topics such as predictive modeling and clustering. Additionally, it assisted in generating insights and actionable business implications based on the project's findings.