1. INTRODUCTION

1.1. Project Overview

Advancing Nutrition Science through Gemini-AI is an AI-powered web-based application designed to provide users with comprehensive nutritional information about food items. The system leverages Google Generative AI (Gemini) to deliver instant and detailed insights into macronutrients such as proteins, fats, and carbohydrates, as well as micronutrients including vitamins and minerals, along with calorie estimation.

The application is developed using **Streamlit** for the user interface and integrates the **Gemini 1.5 Flash** model for real-time intelligent nutrition analysis. It assists users in understanding food composition, planning healthier diets, and receiving personalized nutrition guidance.

This project demonstrates the practical use of generative AI in healthcare and nutrition by offering a fast, interactive, and user-friendly platform for dietary awareness and meal planning.

1.2. Objectives

The main objectives of this project are:

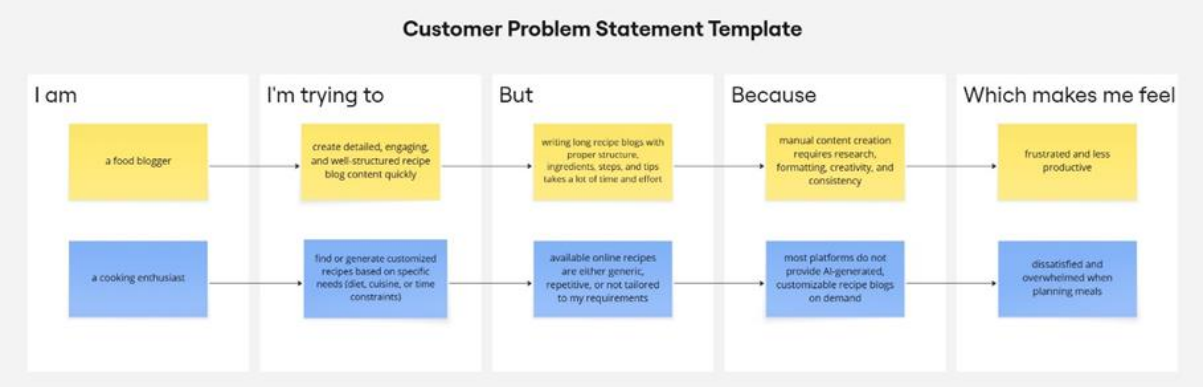
- To develop an AI-powered web application for nutritional analysis
- To provide detailed macronutrient and micronutrient information about foods
- To generate personalized weekly meal plans based on user preferences
- To provide virtual nutrition coaching using AI
- To help users make informed dietary decisions
- To create a simple and user-friendly interface using Streamlit

2. Project Initialization and Planning Phase

2.1. Defining Problem Statement

Many individuals struggle to understand the nutritional value of their daily food intake and often fail to maintain balanced diets. Manual diet planning and consulting nutrition experts can be expensive and time-consuming. Moreover, most existing tools provide generic and non-personalized nutrition advice.

This project solves the problem by providing an AI-powered system that delivers real-time personalized nutrition insights, meal planning, and virtual coaching using Gemini AI.



Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	A food blogger	Create detailed, and high-quality recipe blog content in less time	Writing long and recipe blogs manually is time-consuming	It requires research, creativity, formatting, and consistency	Frustrated and less productive
PS-2	A cooking enthusiast	Generate customized recipes	Existing recipes available online are generic and not customizable	Most platforms do not offer AI-driven recipe blog generation	Dissatisfied and overwhelmed while planning meals

2.2. Project Proposal (Proposed Solution)

Project Overview	
Objective	The main objective of this project is to develop an AI-powered web application that generates customized and high-quality recipe blogs based on user inputs.
Scope	The scope of the project includes building a Streamlit-based web interface where users can enter a recipe topic and desired word count. The system generates a complete recipe blog using an AI model and displays it to the user.
Problem Statement	
Description	Many food bloggers and users find it difficult to create detailed and well-structured recipe blogs within a short time. Manual content creation requires creativity, research, and proper formatting, making it time-consuming.
Impact	Solving this problem helps users save time and effort while creating customized recipe blogs
Proposed Solution	
Approach	The proposed solution uses a Streamlit web application integrated with the Gemini Flash Lite model. Users provide a recipe topic and word count through the UI. The AI model processes the input and generates a complete recipe blog.
Key Features	• User-friendly Streamlit interface • AI-powered nutritional analysis • Image-based food nutrition detection • Virtual AI nutrition coach

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU for application execution	Standard system CPU
Memory	RAM required to run application	8 GB
Storage	Disk space for application files	50 GB SSD
Software		
Frameworks	Python web framework	Streamlit
Libraries	AI and utility libraries	google-generativeai
Development Environment	IDE and version control	VS Code, Git
Data		
Data	User input text	Food Items names entered by user

2.3. Initial Project Planning

Product Backlog, Sprint Schedule, and Estimation

The following table represents the product backlog and sprint-wise planning for the **Nutrition Science** project.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date
Sprint 1	User Interface Setup	USN-1	As a user, I can access a Streamlit-based interface to enter a recipe topic and word count.	2	High	All Team Members	28 January 2026	31 January 2026

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date
Sprint 1	Input Validation	USN-2	As a user, I want the application to validate my inputs before generating the recipe.	1	High	All Team Members	28 January 2026	31 January 2026
Sprint 2	AI Model Integration	USN-3	As a user, I want the system to generate a recipe blog using the Gemini Flash Lite model.	3	High	All Team Members	02 Februar y 2026	09 February 2026
Sprint 2	Joke Generation	USN-4	As a user, I want to see a programming joke while the recipe is being generated.	1	Medium	All Team Members	02 Februar y 2026	09 February 2026
Sprint 3	Output Display	USN-5	As a user, I want to view the generated recipe blog clearly on the screen.	2	High	All Team Members	12 Februar y 2026	18 February 2026
Sprint 3	Deployment	USN-6	As a user, I want the application to be deployed and accessible through the internet.	2	Medium	All Team Members	12 Februar y 2026	18 February 2026

3. Data Collection and Preprocessing Phase

3.1. Data Collection plan and Raw Data Sources Identified

3.1.1. Data Collection Plan

Section	Description
Project Overview	The application collects data dynamically from users through the Streamlit interface based on user input. The project dynamically processes user-provided text input to generate content.
Data Collection Plan	The data for this project is collected **directly from users at runtime** through a Streamlit-based web interface. Users provide: • Dietary restrictions • Health conditions
Raw Data Sources Identified	Since this project uses real-time user input and a pre-trained AI model, no external datasets (CSV, images, or files) are collected or stored.

3.1.2. Raw Data Sources Template

Source Name	Description	Location/ URL	Format	Size	Access Permissions
User Input Data	Text entered by users (recipe topic & word count)	Streamlit Web Interface	Text	Small	Public (user-provided)
AI Model Output	Generated recipe blog content	Gemini Flash Lite API	Text	Variable	Restricted (API-based)

3.2. Data Quality Report

Data Quality Report

The Data Quality Report summarizes potential data quality issues related to user-provided input in this application. Since the project does not rely on an external dataset, data quality focuses on ensuring valid, clean, and meaningful text input before processing by the AI model.

Data Source	Data Quality Issue	Severity	Resolution Plan
User Input (Text)	Empty or missing recipe topic	High	Input validation is applied to ensure the topic field is not empty before submission.
User Input (Text)	Invalid or very low word count	Moderate	Word count is validated to ensure it falls within an acceptable range.
User Input (Text)	Special characters or extra spaces	Low	Input text is cleaned by trimming unnecessary spaces before processing.
User Input (Text)	Ambiguous or unclear prompts	Low	Clear prompt formatting is applied before sending input to the AI model.

3.3. Data Preprocessing

In this Advancing nutrition project, data preprocessing focuses on **user-provided textual input** rather than images. Since the application uses a **pre-trained generative AI model**, no traditional dataset collection or image preprocessing is required.

Section	Description
Data Overview	The data used in this project consists of user-entered text inputs such as recipe topic and desired word count. No external dataset is used.
Text Cleaning	User input is cleaned by removing unnecessary spaces and validating empty or invalid entries.

Input Validation	Ensures the recipe topic is not empty and the word count is within an acceptable range.
Token Handling	The input text is passed to the Gemini Flash Lite model, which internally handles tokenization and text processing.
Prompt Formatting	The user input is formatted into a structured prompt before being sent to the AI model for recipe generation.
Error Handling	Handles invalid inputs or API-related issues gracefully by displaying appropriate error messages.
Data Preprocessing Areas	
Loading Data	User inputs are collected directly through Streamlit text input fields.
Input Validation	Code ensures valid recipe topic and word count before processing.
Prompt Creation	Code formats the validated input into a prompt for the Gemini Flash Lite model.
Model Invocation	The formatted prompt is sent to the AI model for recipe blog generation.
Output Handling	The generated recipe text is received and displayed on the Streamlit interface.

4. Model Development Phase

4.1. Model Selection Report

In this project, the focus is on selecting a pre-trained generative language model suitable for real-time recipe blog generation. Unlike traditional deep learning projects that require training CNNs or RNNs, this application leverages an existing large language model (LLM) to generate high-quality text content. The model is selected based on performance, response time, ease of integration, and suitability for text-based content generation.

Model	Description
Gemini Flash Lite (models/gemini-flash-latest)	A lightweight pre-trained generative language model designed for fast and efficient text generation. It supports real-time content creation with good coherence, relevance, and low latency, making it ideal for recipe blog generation.

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

In this project, no custom model training is performed. Instead, a pre-trained generative AI model is integrated and used for recipe blog generation. The focus of this phase is on model selection, configuration, prompt design, and output evaluation, rather than training from scratch.

Initial Model Training Code:

Model Selection and Initialization

The **Gemini Flash Lite (models/gemini-flash-latest)** model is selected due to its lightweight architecture, faster response time, and suitability for real-time text generation tasks.

The model is initialized using the Google Generative AI API. Configuration parameters such as temperature, top-p, top-k, and maximum output tokens are defined to control creativity, response quality, and output length.

Model Validation and Evaluation Report:

Model	Summary	Training and Validation Performance Metrics
Gemini Flash Lite	Pre-trained generative language model optimized for fast text generation	Relevance of generated content, adherence to word count, coherence, clarity, and response time

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

The primary goal of the Model Optimization and Tuning phase is to enhance the accuracy, clarity, and usefulness of the responses generated by the Gemini AI model. Since the project uses a pre-trained generative AI model rather than training a custom neural network, optimization is achieved through **prompt engineering, parameter tuning, and response formatting**.

Hyperparameter Tuning Documentation:

Model	**Tuned Hyper parameters**
Gemini Flash Lite	**Temperature:** Controls creativity of output (set to a moderate value for balanced creativity). **Top-p:** Limits token selection to the most probable tokens for coherent responses. **Top-k:** Restricts token sampling to reduce irrelevant content. **Max Output Tokens:** Ensures generated recipe blogs match the desired word count. **Response Format:** Set to plain text for easy display in the UI.

5.2. Final Model Selection Justification

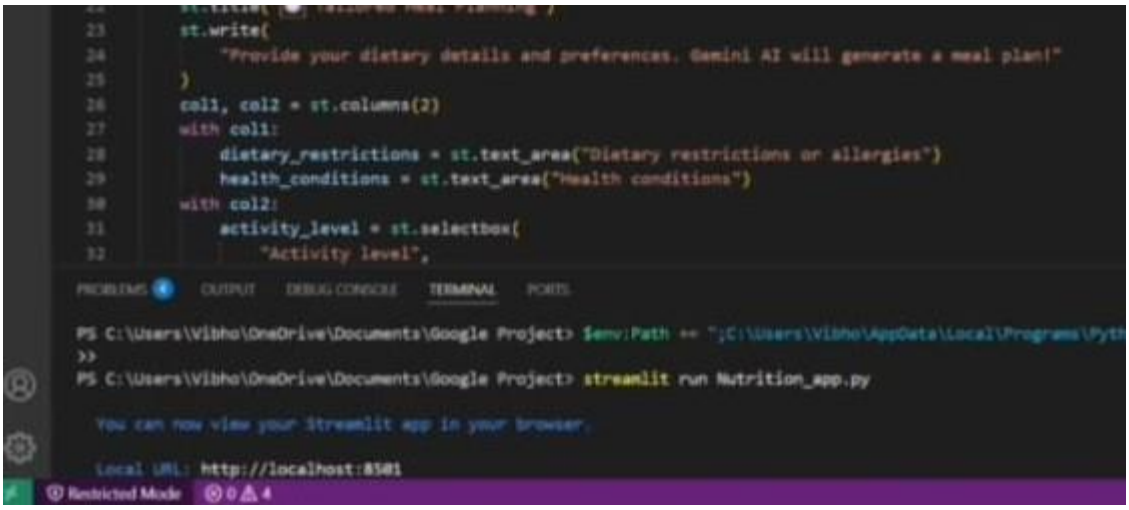
Final Model	**Reasoning**
Gemini Flash Lite (models/gemini-flash-latest)	Selected due to its fast response time, efficient resource usage, high-quality text generation, and seamless integration with Streamlit for real-time recipe suggestions.

6. RESULTS

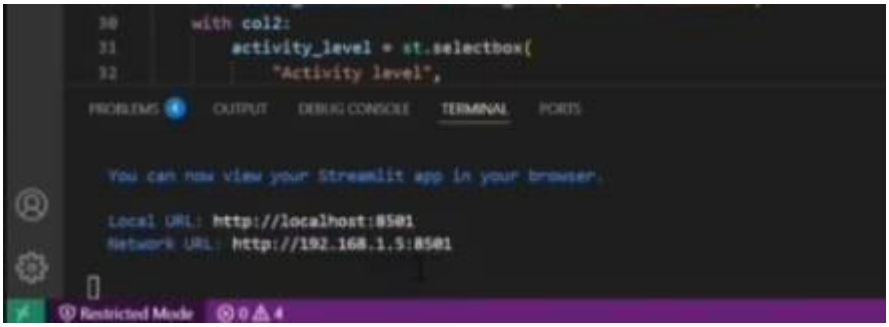
6.1. Output Screenshots

The complete execution of Advancing Nutrition Science through GeminiAI application is represented step by step in the following screenshots.

Step 1: To run the Streamlit Application we have to use the command streamlit run app.py in the terminal in path where the app.py file is located.

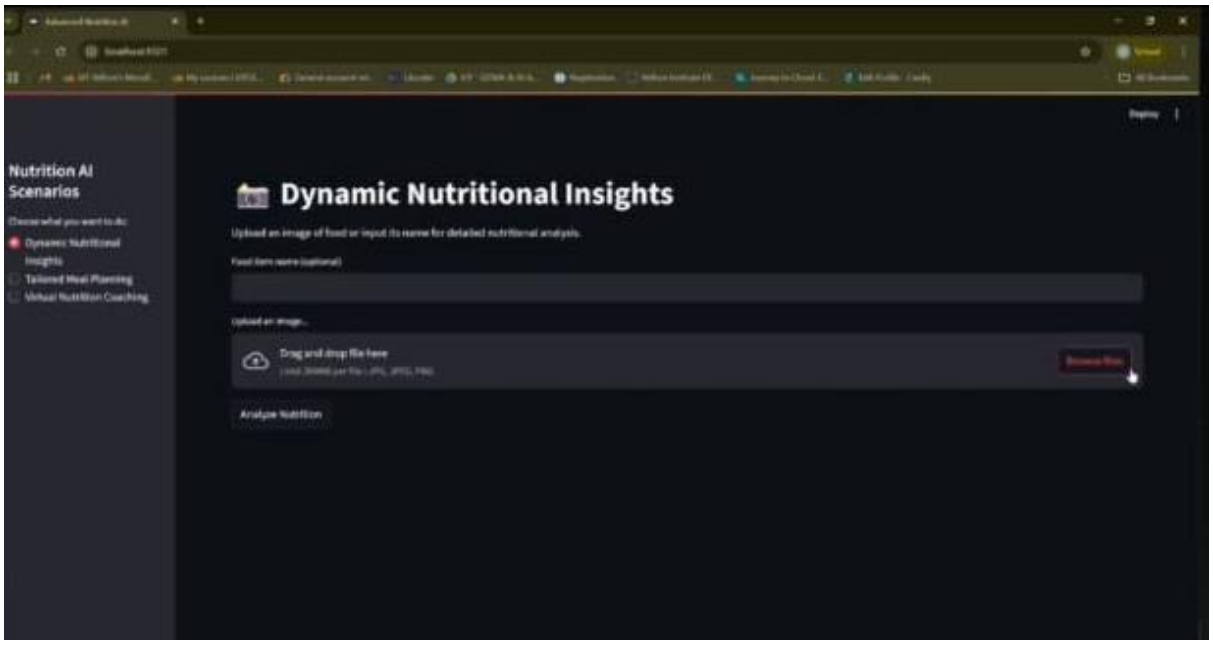


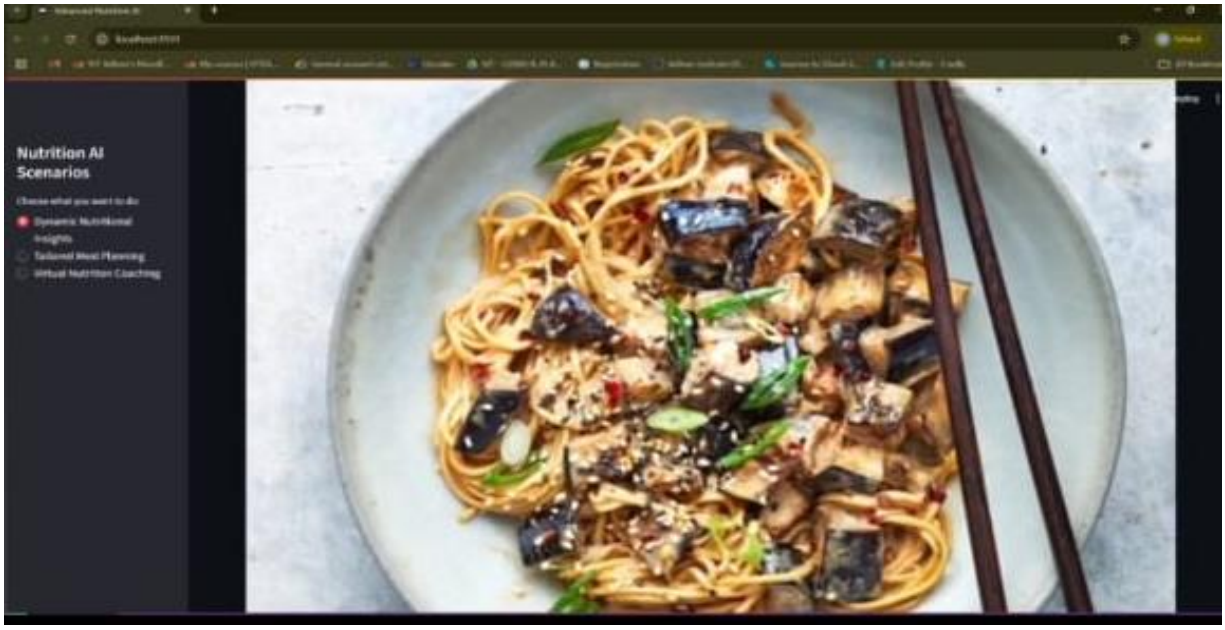
Step 2: After running the command in terminal, the code will get executed and the webpage will open directly. Another way to open webpage is that a localhost link will get generated in the terminal, we can access the webpage using that link.

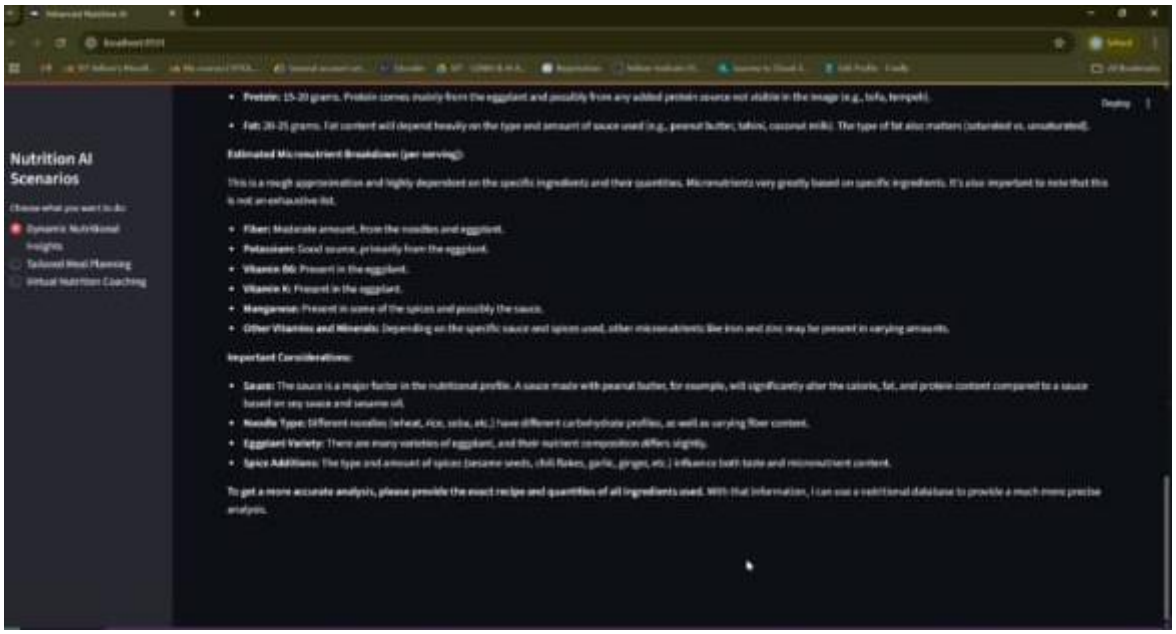


Step 3: The Streamlit webpage opens as shown in the figure given below. This is an automated webpage. No secondary HTML codes required to build this webpage. Python code itself consists the webpage building code.

Scenario 1: Tailored Meal Planning Many individuals struggle with creating healthy and satisfying meal plans that align with their specific needs and preferences. NutriGen addresses this challenge by generating personalized meal plans based on user input. Users can provide information about their dietary restrictions, allergies, health conditions, activity levels, and taste preferences. The AI then crafts a week-long meal plan with recipes and grocery lists, ensuring nutritional balance, variety, and enjoyment.

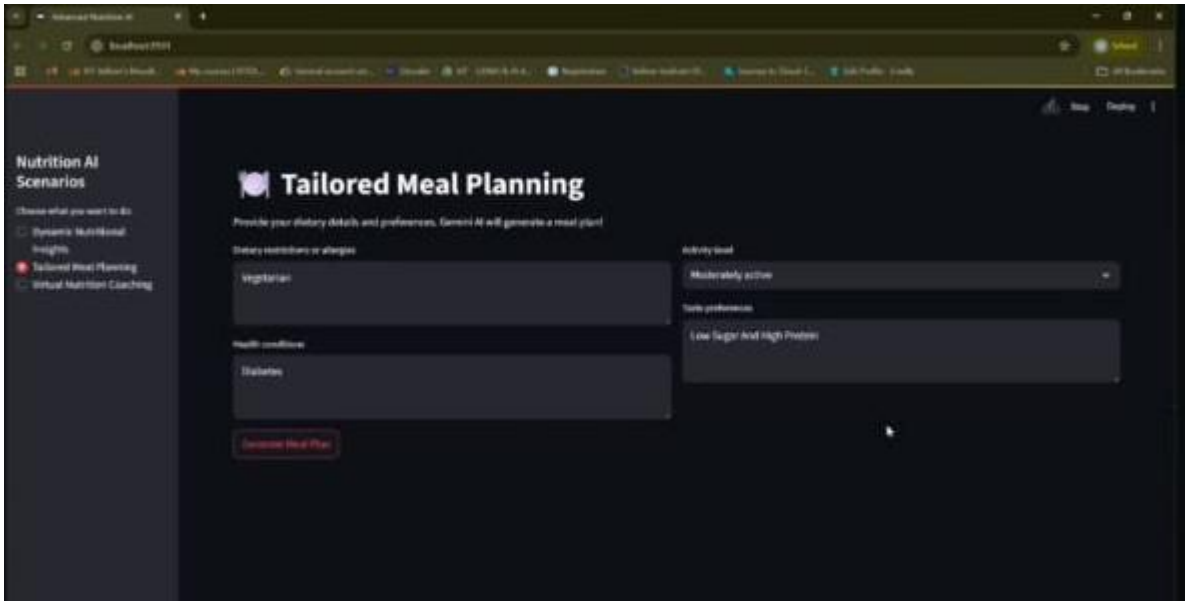


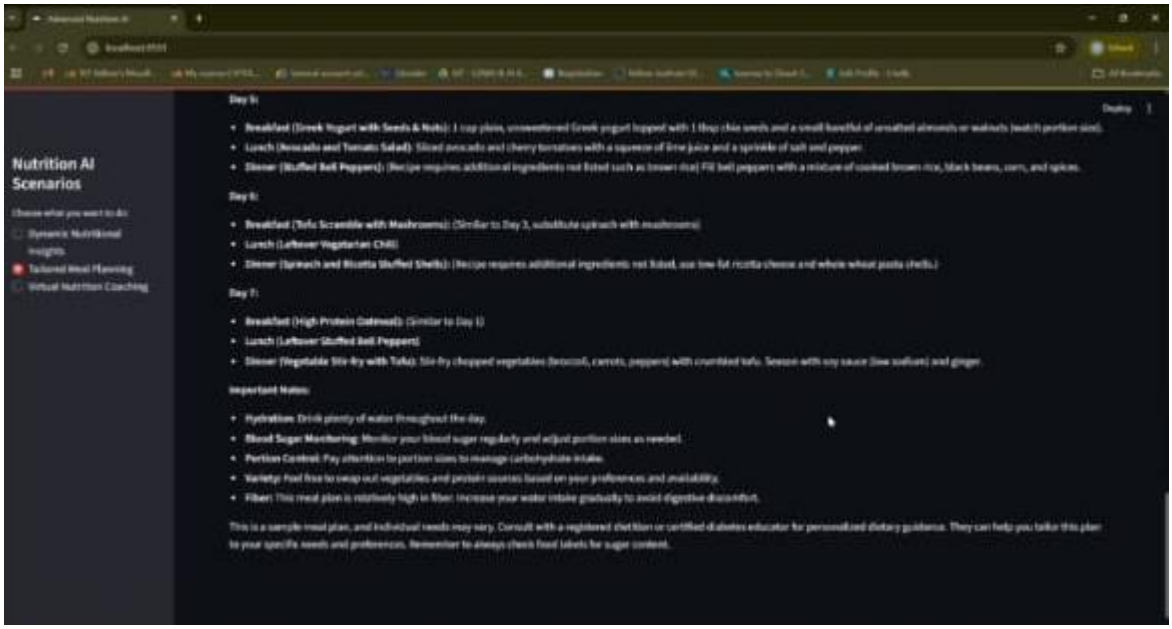




Step 4:

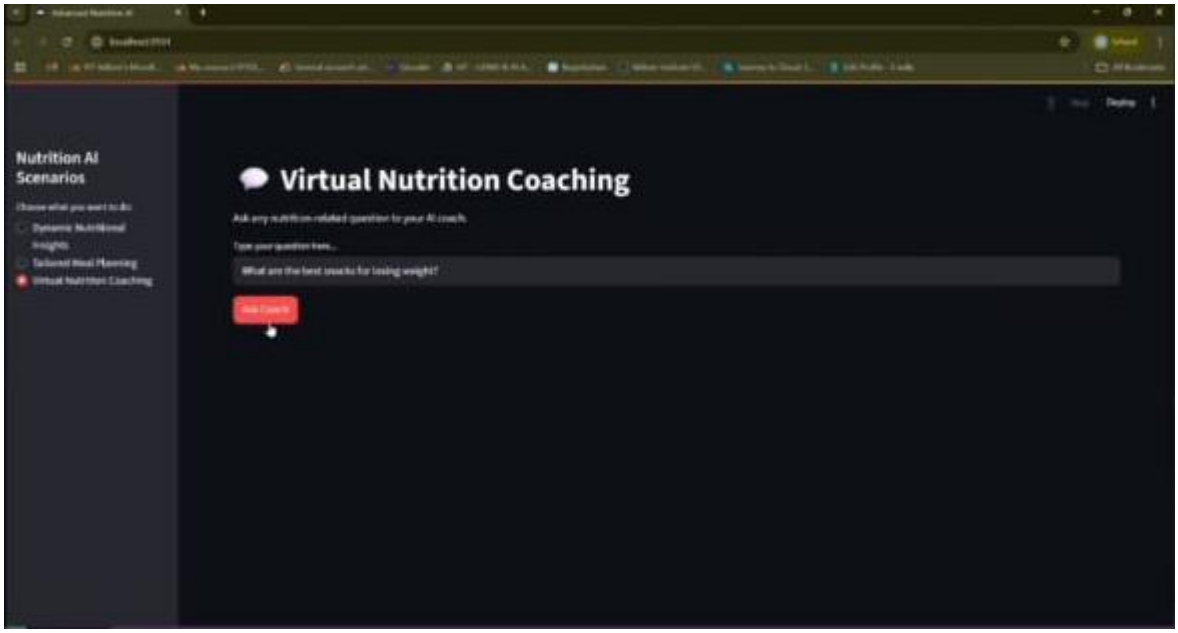
Scenario 2: Dynamic Nutritional Insights Understanding the nutritional content of food is essential for making healthy choices. NutriGen provides users with dynamic nutritional insights about their meals and snacks. By inputting food items or scanning barcodes, users can instantly receive detailed information about macronutrients (protein, fat, carbohydrates), micronutrients (vitamins, minerals), and calorie content. This empowers users to make conscious decisions about their food intake and track their progress toward their nutritional goals.

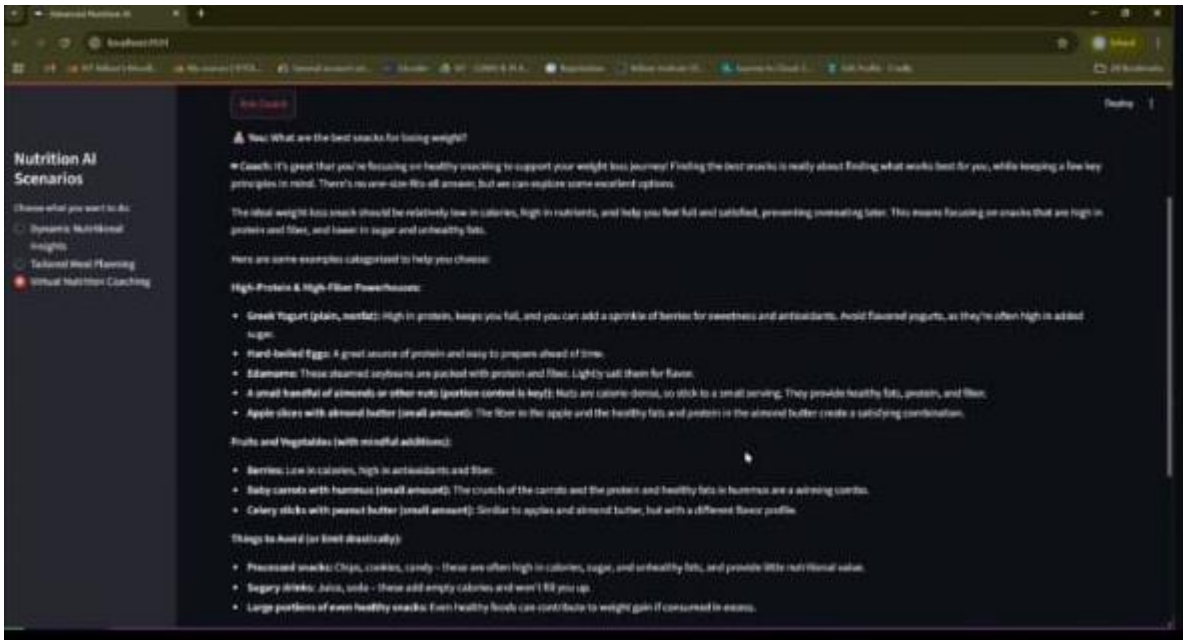




Step 5:

Scenario 3: Virtual Nutrition Coaching Receiving personalized guidance from a nutrition expert can be costly and time-consuming. NutriGen democratizes access to nutritional expertise by offering virtual nutrition coaching. The AI acts as a virtual coach, providing users with personalized advice, answering questions, and offering support throughout their wellness journey. This interactive coaching experience helps users stay motivated, make sustainable lifestyle changes, and achieve long-term health improvements.


Nutrition AI Scenarios
Virtual Nutrition Coaching
Ask any nutrition-related question to your AI coach.
What are the best snacks for losing weight?


Nutrition AI Scenarios
You: What are the best snacks for losing weight?
Coach: It's great that you're focusing on healthy snacking to support your weight loss journey! Finding the best snacks is really about finding what works best for you, while keeping a few key principles in mind. There's no one-size-fits-all answer, but we can explore some excellent options.
The ideal weight loss snack should be relatively low in calories, high in nutrients, and help you feel full and satisfied, preventing overeating later. This means focusing on snacks that are high in protein and fiber, and lower in sugar and unhealthy fats.
Here are some examples categorized to help you choose:
High-Protein & High-Fiber Powerhouses:
Greek Yogurt (plain, nonfat): High in protein, keeps you full, and you can add a sprinkle of berries for sweetness and antioxidants. Avoid flavored yogurts, as they're often high in added sugar.
Hard-boiled Eggs: A great source of protein and easy to prepare ahead of time.
Edamame: These steamed soybeans are packed with protein and fiber. Lightly salt them for flavor.
A small handful of almonds or other nuts (portion control is key!): Nuts are calorie-dense, so stick to a small serving. They provide healthy fats, protein, and fiber.
Apple slices with almond butter (small amount): The fiber in the apple and the healthy fats and protein in the almond butter create a satisfying combination.
Fruits and Vegetables (with mindful additions):
Berries: Low in calories, high in antioxidants and fiber.
Baby carrots with hummus (small amount): The crunch of the carrots and the protein and healthy fats in hummus are a winning combo.
Celery sticks with peanut butter (small amount): Similar to apples and almond butter, but with a different flavor profile.
Things to Avoid (or limit drastically):
Processed snacks: Chips, cookies, candy – these are often high in calories, sugar, and unhealthy fats, and provide little nutritional value.
Sugary drinks: Juice, soda – these add empty calories and won't fill you up.
Large portions of even healthy snacks: Even healthy foods can contribute to weight gain if consumed in excess.

7. ADVANTAGES AND DISADVANTAGES

Advantages

- Provides real-time nutrition insights
- Generates personalized meal plans
- Offers AI-powered virtual nutrition coaching
- User-friendly and interactive interface
- No dataset required for training
- Fast and efficient response generation
- Supports both text and image input
- Helps users make healthier lifestyle decisions

Disadvantages

- Requires continuous internet connection
- Depends on third-party AI API services
- Accuracy depends on clarity of user input
- Cannot replace professional medical consultation
- Limited offline functionality

8. CONCLUSION

The **Advancing Nutrition Science through GeminiAI** project successfully demonstrates the application of generative AI in the field of nutrition and healthcare. The system provides intelligent nutritional insights, personalized meal planning, and virtual coaching through an easy-to-use web interface.

By leveraging Gemini AI, the application helps users understand food composition, make informed dietary decisions, and adopt healthier lifestyle habits. The project highlights how AI can make personalized nutrition guidance accessible, affordable, and efficient for everyone.

9. FUTURE SCOPE

The project can be further enhanced with the following improvements:

- Barcode-based food scanning system
- Mobile application development (Android/iOS)
- Multi-language support for wider accessibility
- Diet tracking and progress monitoring dashboard
- Integration with wearable fitness and health devices
- AI-based calorie tracking system
- Voice-based interaction with nutrition coach
- Personalized diet recommendations based on medical history

These future enhancements will further improve usability, accuracy, and real-world applicability.

10. APPENDIX

10.1. Source Code

The project is implemented using Python and Streamlit and integrates Google Gemini AI for nutrition analysis, meal planning, and coaching.

CODE

```
import streamlit as st
import google.generativeai as genai
from dotenv import load_dotenv
import os
# Load environment variables
load_dotenv()
# Get API key
api_key = os.getenv("GOOGLE_API_KEY")
# Configure Gemini with error handling for the API Key
if api_key:
    genai.configure(api_key=api_key)
    model = genai.GenerativeModel("gemini-1.5-flash")
else:
    st.error("API Key not found. Please check your .env file.")
    st.stop()
# Streamlit Page Config
st.set_page_config(page_title="Advanced Nutrition AI", layout="wide", page_icon="🍏")
# Sidebar
st.sidebar.title("Nutrition AI Scenarios")
```

```
scenario = st.sidebar.radio(
    "Choose what you want to do:",
    ["Tailored Meal Planning",
     "Dynamic Nutritional Insights",
     "Virtual Nutrition Coach"]
)

# ------------------------------------------------
# 1. Tailored Meal Planning
# ------------------------------------------------
if scenario == "Tailored Meal Planning":
    st.title("🍲 Tailored Meal Planning")
    st.write("Provide your dietary details and preferences. Gemini AI will generate a meal plan.")
    col1, col2 = st.columns(2)
    with col1:
        dietary_restrictions = st.text_input("Dietary Restrictions or Allergies", placeholder="e.g. Peanuts, Gluten-free")
        health_conditions = st.text_input("Health Conditions", placeholder="e.g. Type 2 Diabetes, Hypertension")
    with col2:
        activity_level = st.selectbox(
            "Activity Level",
            ["Sedentary", "Moderately Active", "Very Active"]
        )
```

```
    calorie_goal = st.number_input(
        "Daily Calorie Goal",
        min_value=1200,
        max_value=5000,
        value=2000
    )
if st.button("Generate Meal Plan"):
    with st.spinner("Creating your plan..."):
        prompt = f"""
        Create a personalized meal plan.
        Dietary Restrictions: {dietary_restrictions if dietary_restrictions else 'None'}
        Health Conditions: {health_conditions if health_conditions else 'None'}
        Activity Level: {activity_level}
        Daily Calorie Goal: {calorie_goal}
        Provide:
        - Breakfast
        - Lunch
        - Dinner
        - Snacks
        Include estimated calories for each meal.
        """
        try:
            response = model.generate_content(prompt)
            st.subheader("🍽️ Your Personalized Meal Plan")
```

```
            st.markdown(response.text)
        except Exception as e:
            st.error(f"An error occurred: {e}")

# ------------------------------------------------
# 2. Dynamic Nutritional Insights
# ------------------------------------------------
elif scenario == "Dynamic Nutritional Insights":
    st.title("📊 Dynamic Nutritional Insights")
    food_input = st.text_area("Enter food items consumed today:", placeholder="e.g. 2 eggs, 1 avocado toast, and a medium latte")
    if st.button("Analyze Nutrition"):
        if food_input:
            with st.spinner("Analyzing..."):
                prompt = f"""
                Analyze the following food intake and provide:
                - Total calories
                - Macronutrient breakdown (Protein, Carbs, Fats)
                - Improvement suggestions
                Food consumed:
                {food_input}
                """
                try:
                    response = model.generate_content(prompt)
```

```
            st.subheader("📈 Nutrition Analysis")
            st.markdown(response.text)
        except Exception as e:
            st.error(f'An error occurred: {e}")
    else:
        st.warning("Please enter some food items first.")

# ----------------------------------------------------
# 3. Virtual Nutrition Coach
# ----------------------------------------------------
elif scenario == "Virtual Nutrition Coach":
    st.title("🤖 Virtual Nutrition Coach")
    user_question = st.text_area("Ask your nutrition question:", placeholder="e.g. Is it okay to eat fruit at night?")
    if st.button("Ask Coach"):
        if user_question:
            with st.spinner("Consulting coach..."):
                prompt = f"""
                You are a certified nutrition coach.
                Answer the following question professionally and concisely:
                {user_question}
                """
                try:
                    response = model.generate_content(prompt)
```

```
            st.subheader("□ Coach Advice")
            st.markdown(response.text)
        except Exception as e:
            st.error(f"An error occurred: {e}")
    else:
        st.warning("Please enter a question.")
```

10.2. Github & Project Demo Link

Github :- viswanath-3317/Advancing_Nutrition_Science_Through_GeminiAI

Github link:- https://github.com/viswanath-3317

Demo Video Link :- https://youtu.be/oOgfEyGd_BY?si=bE7yyt_DcgZJlHGm