**Calorie-Based Meal Advisor**

SE 403 – Software Project Management

Term Project

Team 1

A picture containing logo, symbol, graphics, font

Description automatically generated

*Instructor*

Ensar Gül

*Group Members*

Canberk Kızılkaya *(Scrum Master), ID: 190706314*

Emin Mert Demirci, *ID: 180706023*

Tolga Emre Koraş, *ID: 200706042*

Melisa Işıtır, *ID: 180706022*

Aleyna Narin, *ID: 190706034*

Yunus Emre Tezcan, *ID: 170706030*

**1. Aim of the Project:**

Today, the use of diet applications has become very popular. Everyone wants to reach their ideal weight and a healthy individual. However, the unhealthy food we eat causes us to gain weight and deteriorate our health. We will try to fix this situation.

To summarize briefly, the aim of this project is to make users consume healthy foods according to their personal characteristics and as many calories as they need. Based on the age, height and weight of the user, we present how many calories he should consume per day and what he should eat for this. In this way, we have made it possible for users to reach us for a healthy diet very easily.

As a result, we have achieved our goal in our Calorie - Based Meal Suggestion project. It has become a web application where people can get healthy food suggestions safely and easily whenever they want.

**2. Functionality of the Project:**

**2. a. Functions available at the beginning of the project:**

2. a. 1. Home Page:

The application provides a home page (index.html) that serves as the landing page for users.

2. a. 2. BMR Calculator Page:

There is a page (BMRpage.html) that allows users to calculate their Basal Metabolic Rate (BMR) based on inputs such as gender, age, weight, height, and activity level.

2. a. 3. About Us Page:

The application includes an About Us page (about\_us.html) that provides information about the project or the organization behind it.

2. a. 4. Contact Page:

There is a Contact page (contact.html) that allows users to get in touch or send inquiries.

**2. b. Functions added during the development:**

2. b. 1. Meal Recommendations:

A new feature was introduced to provide meal recommendations. Users can enter their input in a text area, and the application utilizes the OpenAI language model to generate suggested meals. The meal recommendations are then displayed as a response in JSON format.

2. b. 2. Harris-Benedict Equation Calculator:

Another functionality added was the Harris-Benedict Equation Calculator. Users can input their gender, age, weight, height, and activity level, and the application calculates the daily calorie intake based on the Harris-Benedict equation.

2. b. 3. Results Page:

After using the Harris-Benedict Equation Calculator, the application generates a meal plan suggestion. The prompt for the meal plan is dynamically generated based on the calculated calorie intake. The response from the OpenAI language model is then displayed on the results page (results.html).

These added functionalities enhance the capabilities of the Flask application by providing users with personalized meal recommendations based on their calorie intake goals and the ability to calculate their BMR. The integration of OpenAI's language model allows for dynamic and customized responses, making the application more interactive and user-friendly.

**3. Missing Parts:**

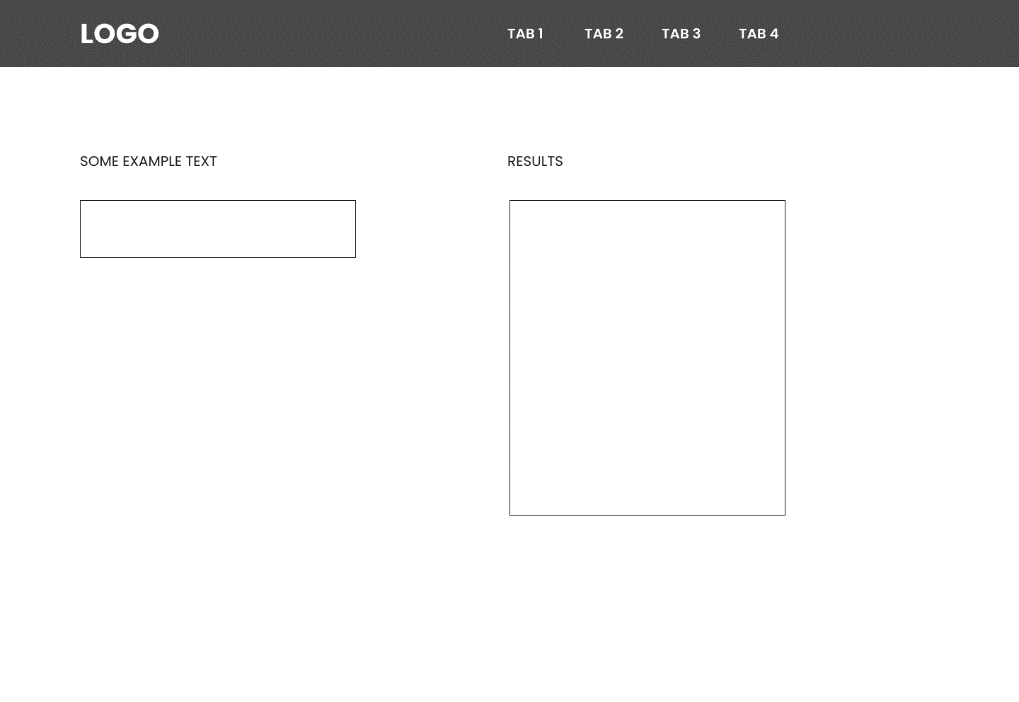
3. a. E-mail System:

We have prepared a mail system to communicate with the user. But we could not use it in our project due to security errors in server access.

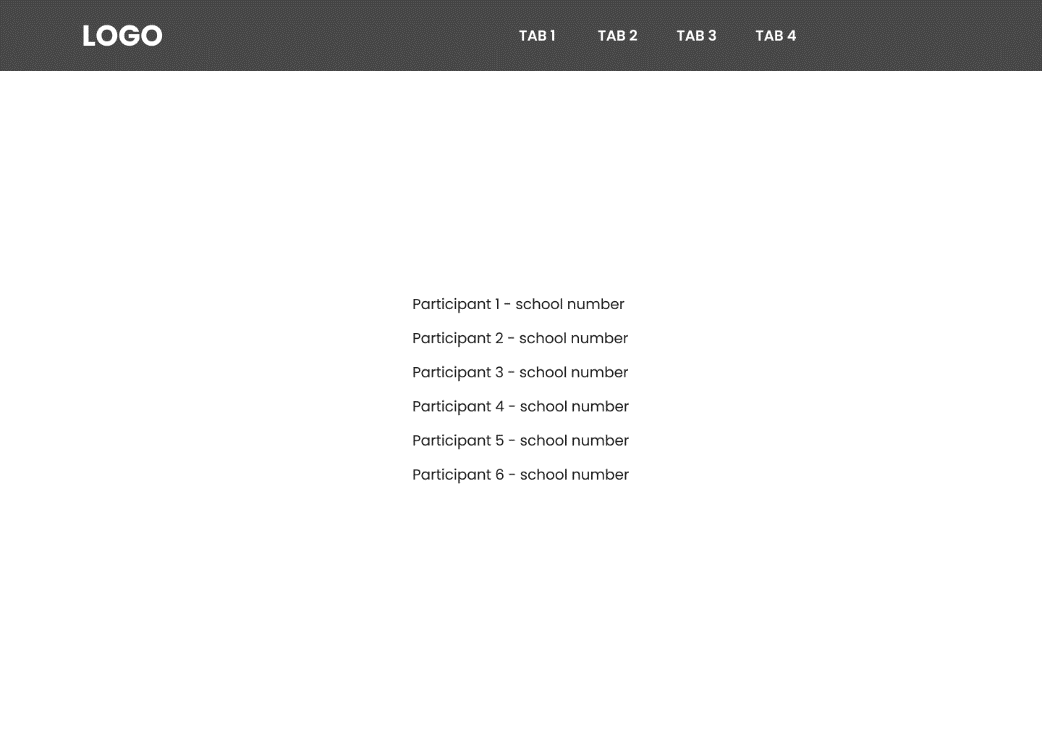
3. b. Animated Website :

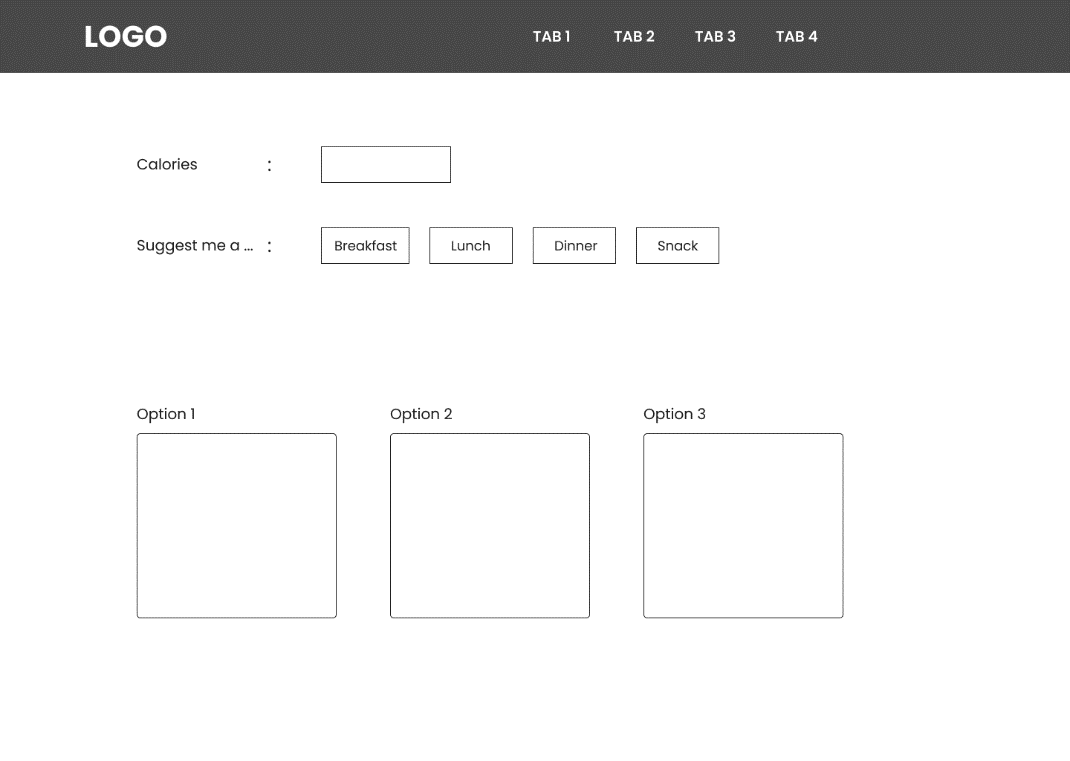
We have prepared a mail system to communicate with the user. But we could not use it in our project due to security errors in server access.

**4. Design Documents :**



* Home Page :



* About Us Page :
* BMR Page :

**5. Deployment:**

Set up the environment :

Make sure you have Python installed on your system. If not, download and install Python from the official website (https://www.python.org).

Create a new directory for your project and navigate to it using the command line/terminal.

Install the required dependencies :

Create a virtual environment by running the following command:

python -m venv venv

Activate the virtual environment:

For Windows:

venv\Scripts\activate

For Unix/Linux :

source venv/bin/activate

Install Flask and OpenAI Python packages :

pip install flask openai

Save your Flask application code in a Python file, for example, app.py.

Create a directory called ‘templates’ in same directory where you ‘app.py’ file is located. This directory will contain HTML templates.

Create a directory called static in the same directory where your app.py file is located. This directory will contain static files, including CSS files.

Create a new JavaScript file in the same directory where your ‘app.py’ located. ‘.js’

Extension. This file contains JavaScript code.

Export the OpenAI API key as an environment variable:

For Windows:

set OPENAI\_API\_KEY=KEY

For Unix/Linux:

export OPENAI\_API\_KEY=KEY

Run the Flask application:

In the command line/terminal, navigate to the directory where your app.py file is located.

Run the following command to start the Flask development server:

flask run

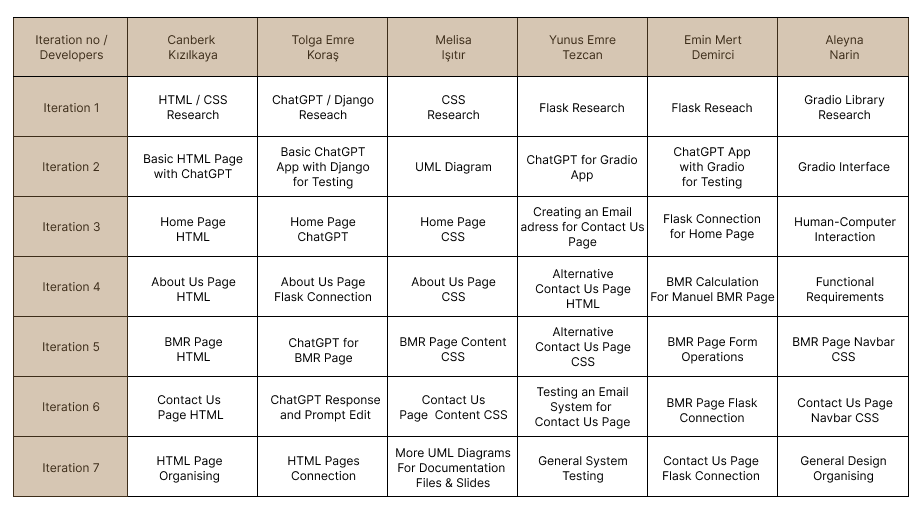
Access the application:

Once the Flask application is running, you can access it in your web browser by visiting http://localhost:5000 or http://127.0.0.1:5000.

Note: Make sure to replace the openai.api\_key value in your code with your actual OpenAI API key.

Remember, this deployment setup is suitable for development purposes. For a production deployment, you may need to consider additional configuration and security measures.

**6. Tasks and Responsibilities:**



**7. Risk Management:**

7. 1. Invalid User Input:

Users may provide incorrect or invalid input when using the BMR Calculator or submitting forms, leading to unexpected behavior or errors.

7. 2. Error Handling:

The application needs to handle errors gracefully and provide meaningful error messages to users when validation or other errors occur.

7. 3. Input Validation:

Implemented robust input validation mechanisms to ensure that user input is properly validated before processing. Validated input fields such as age, weight, height, and activity level to ensure they meet the required criteria. We used Flask's form validation features and custom validation functions to validate user input.

7. 4. Error Messages:

Created clear and informative error messages to guide users when they provide invalid input. Displayed these error messages on the corresponding pages or forms to help users understand and correct their input. We used Flask's flash messages and render the error messages directly in the templates.

**8. Tests :**

8. 1. Unit Testing:

Unit testing focuses on testing individual units or components of the application in isolation. The unittest or pytest frameworks in Python can be used to write and execute unit tests. Developers can create test cases for functions, methods, and classes to validate their behavior and expected outputs. Mocking and patching techniques can be employed to isolate dependencies and simulate different scenarios.

Create a new Python file for your unit tests, for example, test\_app.py.

Import the necessary modules for testing:

import unittest

from flask import Flask, request

from app import app, generate\_meal\_recommendations, harris\_benedict\_equation\_calculation

Define a test class that inherits from unittest.TestCase:

class FlaskAppTests(unittest.TestCase):

def setUp(self):

app.testing = True

self.app = app.test\_client()

def tearDown(self):

pass

Write individual test methods for each functionality you want to test. Use the self.app object to simulate requests and assert the expected responses. Here's an example test method for the get\_meal\_recommendations endpoint:

def test\_get\_meal\_recommendations(self):

data = {'user\_input': 'Sample user input'}

response = self.app.post('/get\_meal\_recommendations', data=data)

self.assertEqual(response.status\_code, 200)

self.assertIn('meal\_recommendations', response.get\_json())

Add test methods for other functionalities, such as results or harris\_benedict\_equation, using similar patterns.

Add the following code at the bottom of the test\_app.py file to run the tests when the file is executed directly:

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Open a terminal or command prompt and navigate to the directory where your test\_app.py file is located.

Run the tests by executing the following command:

python test\_app.py

Note: Make sure you have the necessary dependencies installed, including unittest and Flask, in the same Python environment where you run the tests.

**9. Experience Gained:**

9. 1. Canberk Kızılkaya:

9. 2. Emin Mert Demirci :

9. 3. Yunus Emre Tezcan :

9. 4. Melisa Işıtır :

9. 5. Aleyna Narin :

9. 6. Tolga Emre Koraş :

9. 7. As A Group :

**10. Source Code Link :**

10. 1. GitHub:

https://github.com/LogitheInsane/se-403-spring-22-team-1

10. 2. Trello:

https://trello.com/b/U66Ajgoe/calorie-based-meal-suggestion