

# **NUTRITION ASSISTANT APPLICATION USING CLOUD TECHNOLOGY**

## **CLOUD APP DEVELOPMENT**

**TEAM ID: PNT2022TMID07477**

### **A PROJECT REPORT**

*Submitted by*

**PALAGIRI SANA**

**VUSTHILI VIMALA**

**M.S.NANDHINI**

**S.VIDARSHANA**

**COMPUTER SCIENCE AND ENGINEERING**

**P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

**Pollachi, Coimbatore Dt. - 642 002**



**NOVEMBER 2022**

# **P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**NUTRITION ASSISTANT APPLICATION USING CLOUD TECHNOLOGY**” is the work of “**PALAGIRI SANA (721719104060), VUSTHILI VIMALA(721719104092),M.S.NANDHINI(721719104055), S.VIDARSHANA (721719104090)**” who carried out the project work under our supervision.

### **SIGNATURE**

**Dr. D. CHITRA**

Professor

**HEAD OF THE DEPARTMENT**

Computer Science and Engineering

P. A. College of Engineering and  
Technology

### **SIGNATURE**

**FACULTY MENTOR**

Dr. M. UMASELVI

Associate Professor

Computer Science and  
Engineering

P. A. College of Engineering and  
Technology

### **SIGNATURE**

**FACULTY EVALUATOR**

Mrs. P. SANGEETHA

Assistant Professor

Computer Science and  
Engineering

P. A. College of Engineering and  
Technology

Submitted to the Viva- Voce Examination held on -----

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# **CONTENTS**

## **1. INTRODUCTION**

- 1.1 Project Overview
- 1.2 Purpose

## **2. LITERATURE SURVEY**

- 2.1 Existing Problem
- 2.2 References
- 2.3 Problem Statement Definition

## **3. IDEATION & PROPOSED SOLUTION**

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

## **4. REQUIREMENT ANALYSIS**

- 4.1 Functional requirement
- 4.2 Non-Functional requirement

## **5. PROJECT DESIGN**

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

## **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

- 7.1 Feature1
- 7.2 Feature2
- 7.3 Database Schema

## **8. TESTING**

8.1 Test cases

8.2 User Acceptance Testing

## **9. RESULTS**

9.1 Performance Metrics

## **10.ADVANTAGES & DISADVANTAGES**

## **11.CONCLUSION**

## **12.FUTURE SCOPE**

## **13.APPENDIX**

Source Code

GitHub Link & Project Demo Link

# CLOUD APP DEVELOPMENT

## Nutrition Assistant Application

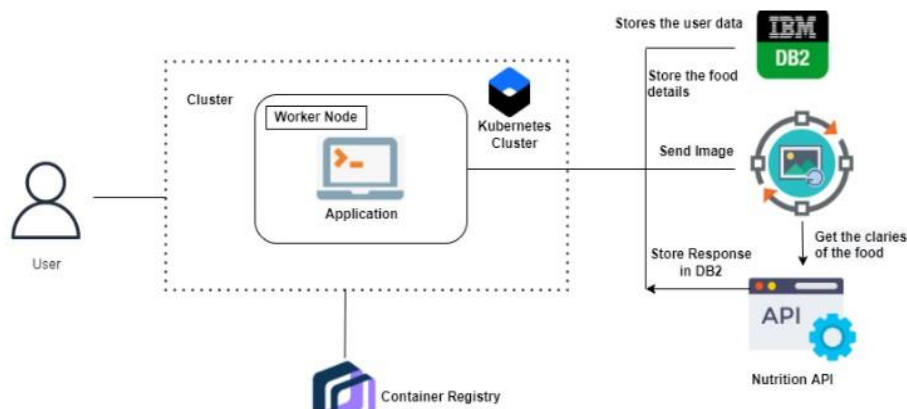
### 1.INTRODUCTION

#### 1.1 Project Overview:

Building a WEB APP that automatically estimates food attributes such as ingredients and nutritional value of food by classifying the input of the image. Our method employs **Clarifai's AI-Driven Food Detection Model** for food identification and Food API's to give the nutritional value of the identified food.

- First user interacts with WEB APP to upload image
- IBM DB2 stores the user data and stores the food details
- API gets the claries of the food and those responses stored in DB2

**Skills Required:** IBM Cloud, HTML, Javascript, IBM Cloud Object Storage, Python-Flask, Kubernetes, Docker, IBM DB2, IBM Container Registry.



#### 1.2 Purpose:

WEB APP provides Nutritional values of image.

(Nutritional values is a part of food quality is the measure of a well-balanced ratio of essential nutrients proteins, vitamins, minerals, fat, carbohydrates in item of food) Those Nutritional values stored in Database.

Due to ignorance of healthy food habits, obesity rates are increasing. So, people must follow dietary plans to avoid future health problems.

Our AI-Driven Method analyzes the real-time images of a meal with the help of this application users are able to maintain healthy life-style.



## 2.LITERATURE SURVEY

## 2.1 Existing problem:

Healthy nutrition contributes to prevent Non communicable and diet related diseases. Recommender systems, as an integral part of mHealth technologies, address this task by supporting users with healthy food recommendations. However, knowledge about the effects

of the long-term provision of health-aware recommendations in real-life situations is limited. This study investigates the impact of a mobile, personalized recommender system named Nutrilize. Our system offers automated personalized visual feedback and recommendations based on individual dietary behaviour, phenotype, and preferences.

By using quantitative and qualitative measures of 34 participants during a study of 2-3 months, we provide a deeper understanding of how our nutrition application affects the user's physique, nutrition behaviour, system interactions and system perception.

Our results shown that Nutrilize positively affects nutritional behaviour (conditional  $R^2=.342$ ) measured by optimal intake of each nutrient. The analysis of different application features shows that reflective visual feedback has a more substantial impact on healthy behaviour than the recommender (conditional  $R^2=.342$ ) we further identify system limitations influencing this result, such as a lack of diversity, mistrust in healthiness and personalization, real life contexts, and personal user characteristics with a qualitative analysis of semi-structured in depth interviews.

Finally, we discuss general knowledge acquired on the design of personalized mobile nutrition recommendations by identifying important factors, such as the user's acceptance of the recommender's taste, health, and personalization.

## **2.2 References:**

### **1. Development Of A Cloud Based Solution For Effective Nutrition Intervention In The Management Of Lifestyle Diseases**

**Author and Year :** Manju P George, C.A.Kalpana. November 2020 **Source:**  
Asian Journal of Multidimensional Research (AJMR)

#### **Findings:**

- The cloud based system would have the ability to calculate the nutritional requirements and to guide first line nutritional management to patients and clients automatically.

- Also, it serves as an electronic medical and dietetic record, and a personalized nutrition consultation approach can enable clients to converse to his/ her personal dietitian at their own convenient setting.
- Authenticity of the consultant dietitian would also be ensured by the responsible team providing nutrition support.

## **2. Food calorie estimation using machine learning and image processing**

**Author and Year:** Shaikh Mohd. Wasif ,Swapnil Thakery, Amir Nagauri, Sheetal Ignatius Pereira

**Source:** International Journal of Advance Research, Ideas and Innovations in Technology

### **Findings:** •

This paper focuses on creating software which gives the calorie of the food which the user is going to consume.

- In order to achieve this, the software will take two images as input from the user, the top view and the side view.
- The food item in the image will be detected with the help of Faster R-CNN algorithm.
- After segmentation of images, the volume of the food item is calculated using the known volume of the probe object.
- After the calculation of volume, the mass of the food item is calculated with the help of formulas and then the calories of the food item will be calculated using the relation between mass and calories.

## **2.3 Problem Statement Definition:**

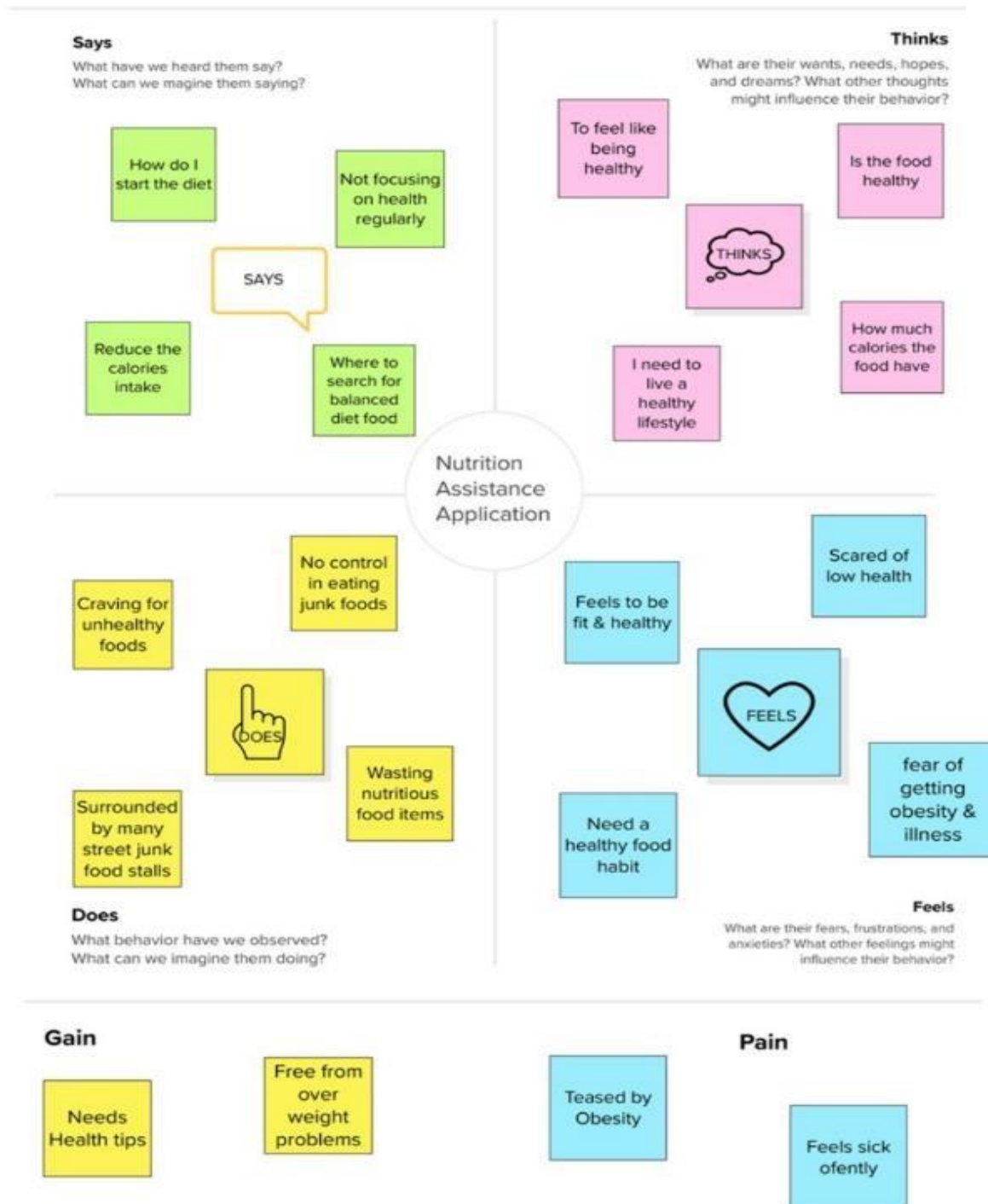
Due to the ignorance of healthy food habits, Obesity rates are increasing at an alarming speed, and this is reflective of the risks to the people's health. People need to control the daily Calories intake by eating Healthier foods, which is the most basic method to avoid obesity. However, although food packing comes with Nutrition (and Calories) labels, it's still not very convenient for people to refer to Appbased Nutrient dashboard Systems which can analyze Real-time Images of a Meal and Analyze it for Nutritional content which can be very handy and improves the Dietary habits, and therefore, helps in maintaining a Healthy Lifestyle.

The main Objective of this project is to building a Web App that automatically estimates Food attributes such as Ingredients and Nutrition Value by classifying the input image of food .



## 3.IDEATION & PROPOSED SOLUTION

### 3.1 Empathy Map Canvas:



## 3.2 Ideation& Brainstorming:



## 3.3 Proposed Solution:

S.NO	Parameter	Description
1	Aim	Building a WEB APP, user load image to application nutritional values are stored in Database.
2	Problem Statement (problem to be solved)	App-based nutrient dashboard systems which can analyze real-time images of a meal and analyze it for nutritional content.
3	Idea / Solution description	The solution can be brought by using Clarifai's AI-Driven food detection model to obtain precise food identification and food APIs to give the nutritional value of the identified food.
4	Uniqueness	Providing a user-friendly environment to access the nutritional information about the food by 1. Capturing the food 2. Uploading image from the gallery 3. Feed-in manually 4. Choosing from the provided list
5	Customer Satisfaction	By providing custom diet and meal plans to the user, getting user feedbacks for the product enhancement and longevity.
6	Business Model	By introducing not Paid membership plans and Ad's related to the food products and supplements.
7	Scalability of the Solution	1. Providing regular updates 2. Making the application user friendly 3. Ease of access

### 3.4 Problem Solution Fit:

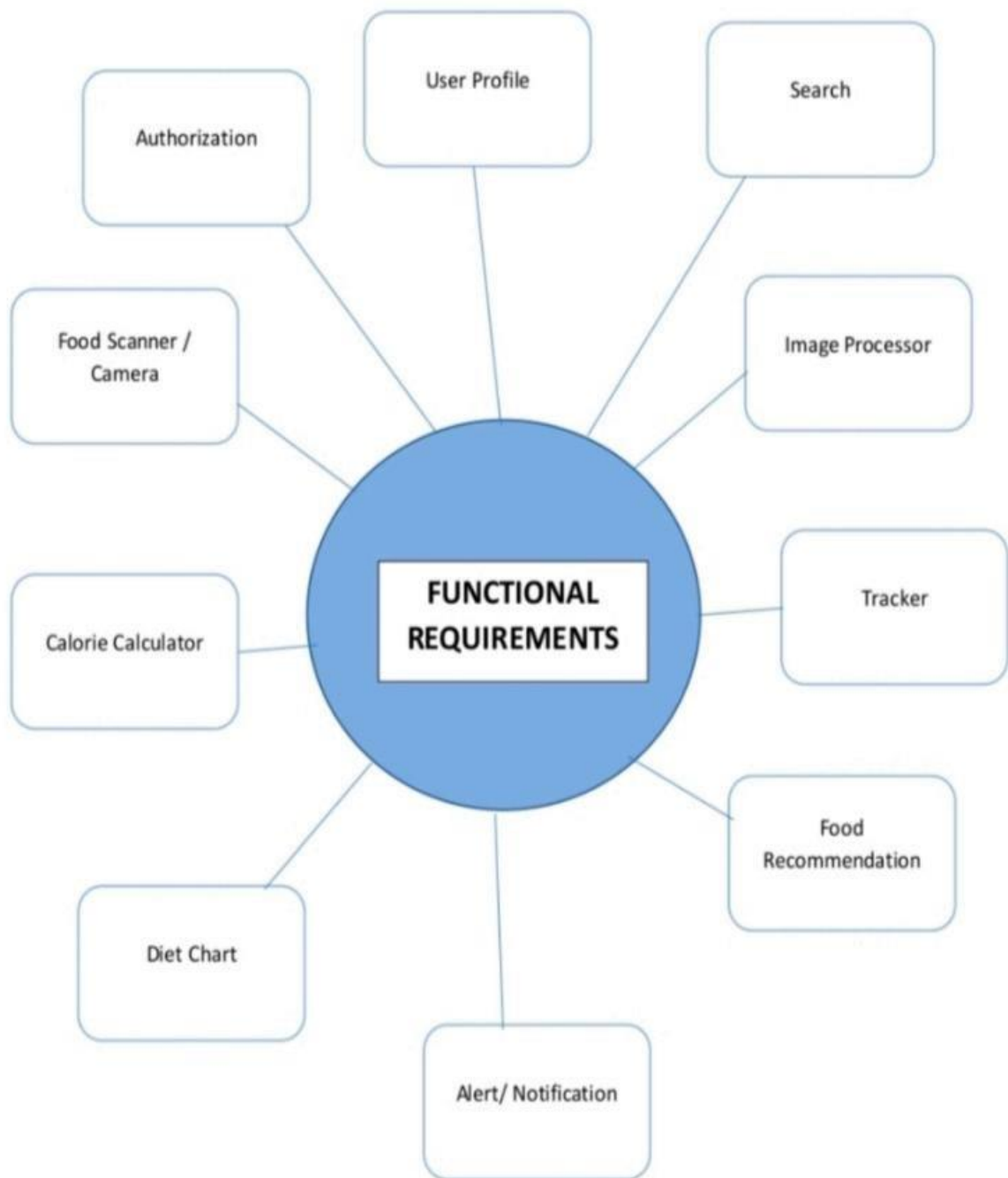
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> <p>All age group people who are careless about their health due to their busy schedule and intake of high-calorie diet.</p>	<b>6. CUSTOMER CONSTRAINTS:</b> <span>■</span> <p>The customer should provide a clear image for knowing the nutrition content about the food. The app can't provide accurate result if the image is not clear. In some cases, the recipes may be allergic to their health.</p>	<b>5. AVAILABLE SOLUTIONS:</b> <span>■</span> <p>Although the food packaging comes with nutrition (and calorie) labels, it's still not very convenient for people to refer to App-based nutrient dashboard systems.</p>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS:</b> <span>J&amp;P</span> <p>The problem and pains of the user are obesity, fear of getting health related issues. They will get frustrated of not getting immediate result and difficult to do tedious work. Lack of confidence due to appearance.</p>	<b>9. PROBLEM ROOT CAUSE:</b> <span>RC</span> <p>It is easy to fall into a trap of eating unhealthy foods which is heavy in calories. Once the nutritional value is replaced by foods high in sugar, bad fats and salt it leads to various health issues so users need to control their daily calorie intake to lead a healthy lifestyle.</p>	<b>7. BEHAVIOUR:</b> <span>BE</span> <p>The behavioral changes in users reflect in their day-to-day life such as they will maintain a proper diet and follow the daily routine in eating and intake of healthy food. So, that it helps them to improve their health.</p>	
Focus on J&P, fit into BE, understand RC	<b>3. TRIGGERS:</b> <span>TR</span> <p>Desire to live a healthy lifestyle. By knowing the success story of people who achieved their goal. By seeing people who are fit and healthy.</p>	<b>10. YOUR SOLUTION:</b> <span>SI</span> <p>The solution is user can know the nutritional content of the food they are taking, by taking picture of the food and uploading it in the app. Clarifai's AI-Driven Food Detection Model is used for getting accurate food identification and APIs to give the nutritional value of the identified food</p>	<b>8. CHANNELS of BEHAVIOUR:</b> <span>CH</span> <p><b>ONLINE:</b> The application provides a user-friendly environment that enables users to interact through chatbot to clarify their queries and a dashboard is displayed to know the activities</p> <p><b>OFFLINE:</b> Connecting all the users through offline meeting and giving some complimentary gifts. Conducting offline session by nutrition expert.</p>	Focus on J&P, fit into BE, understand RC
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> <p>They scared of declining health, so they get motivated towards eating healthy foods and move to healthy lifestyle.</p>			

## 4. REQUIREMENT ANALYSIS

**4.1 Functional requirement:** Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks.

Following are the Functional requirements of the proposed solution:

S.NO	Functional requirements	Sub-Requirements
1	User Registration	-> Registration through Form -> Registration through Gmail • -> Registration through Facebook
2	User Confirmation	-> Confirmation via Email -> Confirmation via OTP
3	User Login	-> Login with Username -> Login with Password
4	User Profile Update	-> Update user's name -> Update date of birth
5	Uploading Food Image	-> Upload from Gallery -> Capture using Camera
6	Enter Food Name	-> Type the name of the food - > Automatically it displays food name
7	Result	-> Download Results -> Share result through social media
8	Ratings and Reviews	-> Share the experiences -> Provide Feedback



## **4.2 Non-Functional Requirements:**

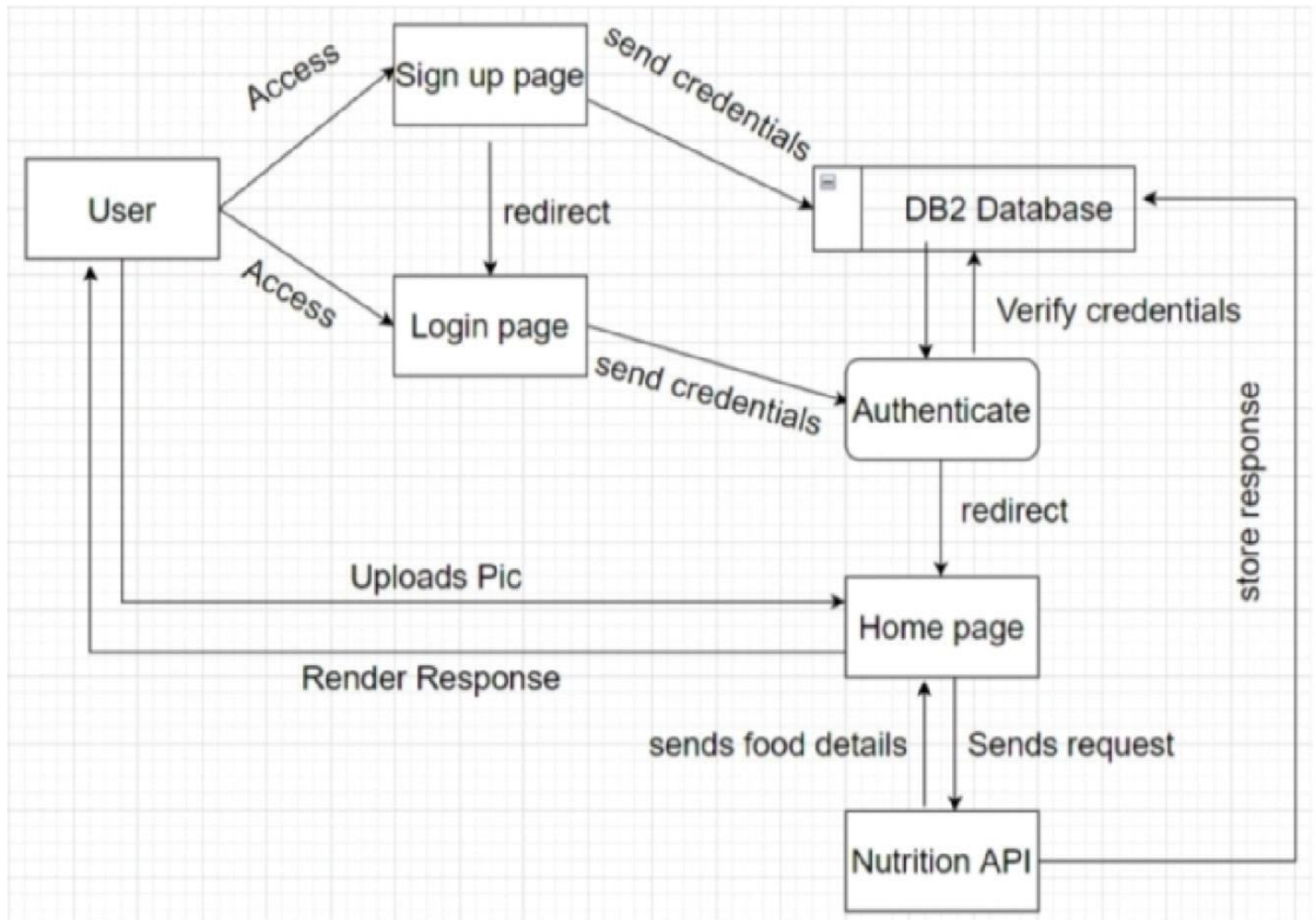
Nonfunctional Requirements (NFRs) **define system attributes such as security, reliability, performance, maintainability, scalability, and usability.**



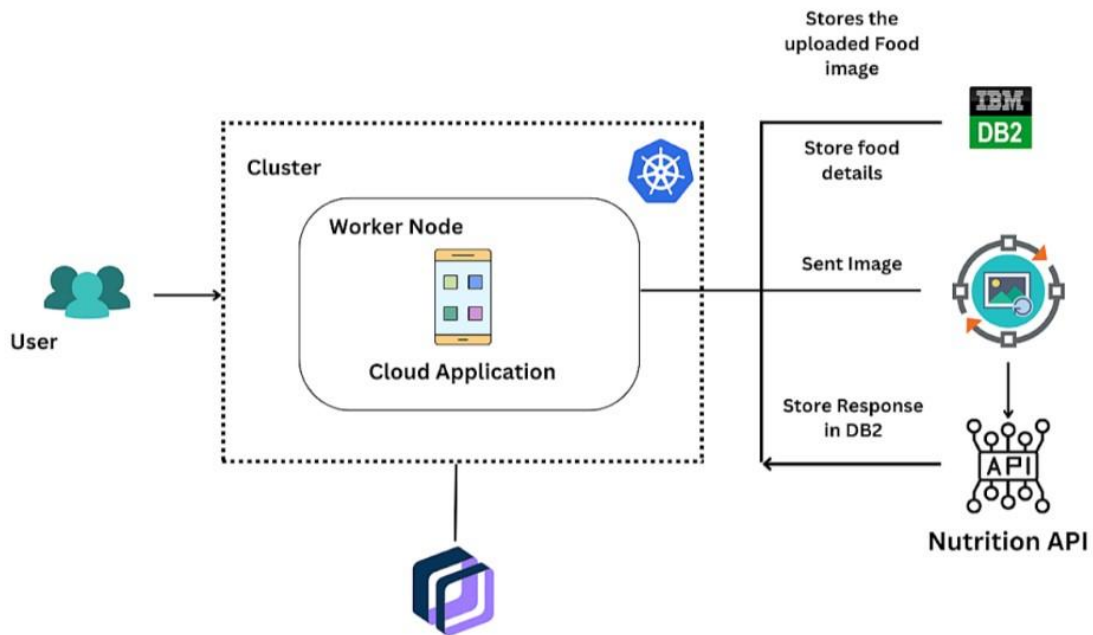
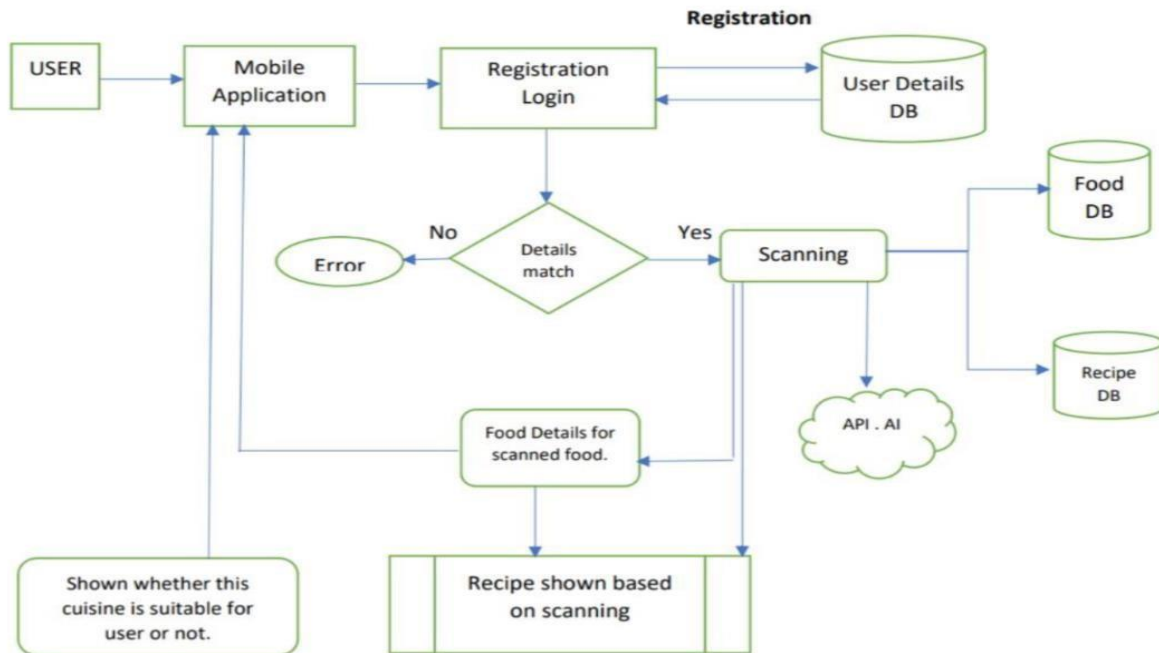
S.NO	Non-Functional Requirements	
1	Usability	Provide user friendly UI simple and Intuitive design
2	Performance	The landing page supporting several users, must provide 5 sec or less response time
3	Scalability	Provide horizontal or vertical scaling for higher workloads
4	Availability	Uninterrupted services must be available all time except the time of server updation.
5	Security	Comprehensive authorization and authentication scheme for each system actor
6	Reliability	The system must perform without failure in 95% use cases

## 5.PROJECT DESIGN

**5.1. Data Flow Diagrams:** A data flow diagram (DFD) maps out the flow of information for any process or system.












## 5.2 Solution & Technical Architecture:





## 5.3 User Stories:

<p>SCENARIO</p> <p>Browsing, booking, attending, and rating a local city tour</p>	 <p><b>Entice</b></p> <p>How does someone initially become aware of this process?</p>	 <p><b>Enter</b></p> <p>What do people experience as they begin the process?</p>	 <p><b>Engage</b></p> <p>In the core moments in the process, what happens?</p>
 <p><b>Steps</b></p> <p>What does the person (or group) typically experience?</p>	<div> <div>knowledge through online advertisement</div> <div>To know the calorie values.</div> </div> <div> <div>A customer navigates to the page for taking image &amp; customer navigates to the page for taking image</div> <div>people want to know the caloric value of the food they intake</div> </div>	<div> <div>User friendly interface</div> </div> <div> <div>Customer can view the home page then register and upload the picture of the food</div> </div>	<div> <div>Upload a image</div> <div>View the result</div> <div>Enjoy their diet</div> </div> <div> <div>Customer wants to take a picture of the food and upload the photo to know the attributes in it</div> <div>Customer can view the caloric value for the uploaded food image</div> <div>According to their BMI customer can get a diet chart.</div> </div>
 <p><b>Interactions</b></p> <p>What interactions do they have at each step along the way?</p> <ul style="list-style-type: none"> <li>■ <b>People:</b> Who do they see or talk to?</li> <li>■ <b>Places:</b> Where are they?</li> <li>■ <b>Things:</b> What digital touchpoints or</li> </ul>	<div> <div>Interaction with a web page.</div> <div>Interaction with browsers.</div> </div>	<div> <div>Interaction with the home page.</div> <div>Interaction with a UI Login page, if they already registered.</div> <div>Interaction with a registration page, if they are new user.</div> </div>	<div> <div>People interacts with a interface to knowing about the food nutrition value easily</div> <div>Interacts with result page using the image upload the user will being engage with the software.</div> </div>
 <p><b>Goals &amp; motivations</b></p> <p>At each step, what is a person's primary goal or motivation? ("Help me..." or "Help me avoid...")</p>	<div> <div>Help me preserve my physical wellbeing.</div> <div>Assist me with avoiding junk food.</div> </div>	<div> <div>Please help me to know the food's calorie count.</div> <div>Give me suggestion to maintain my diet.</div> </div>	<div> <div>Please assist me in learning the nutritional content of each meal.</div> <div>shows the caloric value of the uploaded picture.</div> <div>It gives diet suggestions according to the BMI given.</div> <div>Customer should follow the diet plan.</div> </div>
 <p><b>Positive moments</b></p> <p>What steps does a typical person find enjoyable, productive, fun, motivating, delightful, or exciting?</p>	<div> <div>Customer feel more enjoyable and excited if it's free to use.</div> </div>	<div> <div>Customers are excited about the user friendly features.</div> </div>	<div> <div>The customer will be happy to maintain proper diet plan in their meal.</div> <div>Customer fell delightful to have a nutritional guideline.</div> <div>User feels joyful to know the calorie value of the food they intake.</div> </div>
 <p><b>Negative moments</b></p> <p>What steps does a typical person find frustrating, confusing, angering, costly, or time-consuming?</p>	<div> <div>A Customer feels upset if a application charges to utilize.</div> </div>	<div> <div>Some users may confusing about using this application.</div> </div>	<div> <div>When the caloric value of the food is incorrect, it will be helpless to the user</div> <div>If the value is inaccurate, it redirects the user's health.</div> </div>
 <p><b>Areas of opportunity</b></p> <p>How might we make each step better? What ideas do we have? What have others suggested?</p>	<div> <div>Easy to accessibility to all customer.</div> <div>Nutritional value estimate.</div> </div>	<div> <div>An user friendly interface.</div> <div>calorie counter estimation.</div> <div>Offer a food diary to let you track what you eat.</div> </div>	<div> <div>Give food nutrition value image processing.</div> <div>Provide customizable meal plan for an individual.</div> <div>Provide high nutrition food list.</div> </div>

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation:

Setting Up Application Environment To create lots of environment. Create or enrolment to the IBM Cloud, Docker CLI Installation, Create an account in SendGrid and Nutrition API etc., Implementing Web Application, Create a UI to interact with application. Create Database system DB2 and connect it with Python and Integrate with Nutrition API. Including some Rest API services for to give a Nutrition and calorie value. Deployment of APP in IBM Cloud in deploy process, deployment in Kubernetes cluster is the major task before that we need to containerize the app and upload image to IBM Container Registry

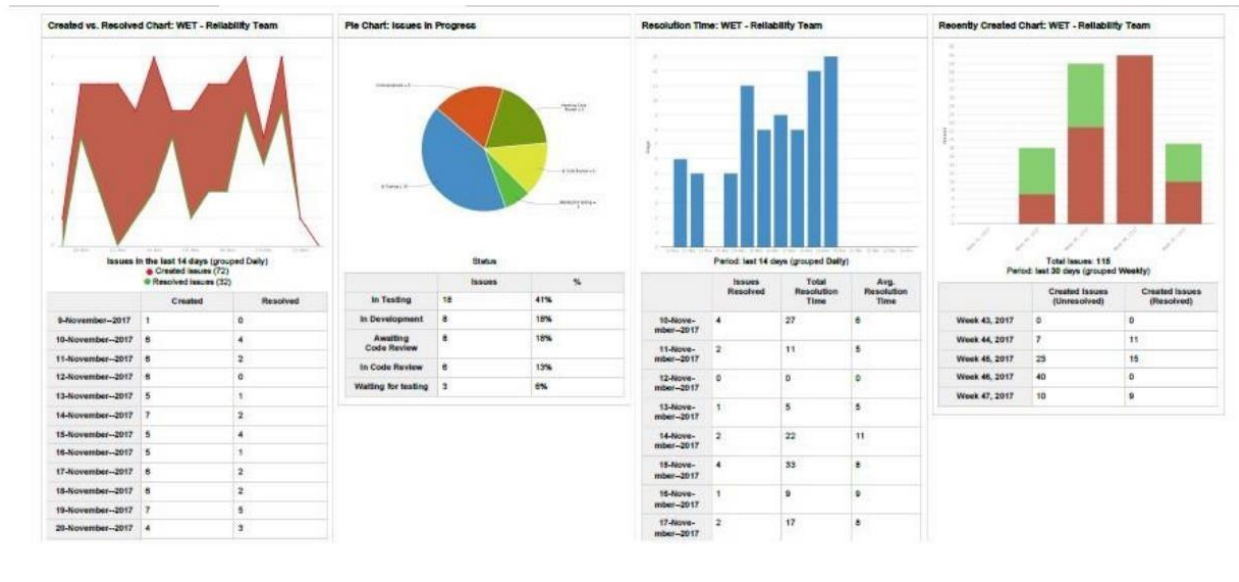
Sprint	Functional Requirement	User story number	User Story	Story points	Priorty	Team members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering email, password, and confirming password	2	High	Palagiri Sana, Vusthili Vimala, Nandhini, Vidarshana
Sprint -1		USN-2	As a user, I will receive confirmation mail once I registered for application	1	High	Palagiri Sana, Vusthili Vimala, Nandhini, Vidarshana
Sprint-1	Login	USN-3	As a user, I can login into the application by entering email& password	1		Palagiri Sana, Vusthili Vimala, Nandhini, Vidarshana
Sprint-2	User details	USN-4	The details of the user and the history of the searched food nutrition data will be available	2	High	Palagiri Sana, Vusthili Vimala, Nandhini, Vidarshana

Sprint-3	Upload Image of the food	USN-5	The user will upload the food image to get nutrition details.	2	High	Palagiri Sana, Vusthili Vimala,
						Nandhini, Vidarshana
Sprint-4	Shown the nutrition details for recipe	USN-6	The system will scanned the food image and display	2	High	Palagiri Sana, Vusthili Vimala, Nandhini, Vidarshana

## **6.2 Sprint Delivery Schedule :**

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	16 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

## **6.3 Reports from JIRA:**



## 7 CODING & SOLUTIONING

### 7.1 Feature1:

#### APP.PY

```
import binascii
```

```
import math import
```

```
random import
```

```
requests as res
```

```
import secrets
```

```
import time
```

```
from base64 import urlsafe_b64encode as b64e, urlsafe_b64decode as b64d from time import
```

```
strftime, localtime
```

```
import ibm_db import sendgrid from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
```

```

from clarifai_grpc.grpc.api import resources_pb2, service_pb2, service_pb2_grpc from

clarifai_grpc.grpc.api.status import status_code_pb2 from cryptography.fernet import

InvalidToken from cryptography.hazmat.backends import default_backend from

cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes from flask

import Flask, render_template, request, session, redirect


from markupsafe import escape from sendgrid.helpers.mail import Mail,

Email, To, Content


# clarifai

YOUR_CLARIFAI_API_KEY = "xxxxxxxxxxxxxxxxxxxxxxxx"

YOUR_APPLICATION_ID = "xxxxxxxxxxxxxxxxxxxxxxxx" channel =

ClarifaiChannel.get_json_channel() stub =

service_pb2_grpc.V2Stub(channel) metadata = (("authorization", f"Key

{YOUR_CLARIFAI_API_KEY}"),)


# sendgrid

SENDGRID_API_KEY =

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"


#      rapid      API      url      =      "https://spoonacular-recipe-

foodnutritionv1.p.rapidapi.com/recipes/parseIngredients"      querystring      =

{"includeNutrition": "true"} headers = {"content-type": "application/x-

wwwform-urlencoded",

```

"XX"

"XX"

```
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'jifif'}
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=1bbf73c5-d84a-4bb0-85b9ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=32286;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=tmj80232;PWD=XFJsY2e4yqV8KpXS", "", "")
```

```
app = Flask(__name__)
```

"\xfd{H\xe5<\x95\xf9\xe3\x96.5\xd1\x01O<!\xd5\xa2\xa0\x9fR"

`send_mail(email):`

```
sg = sendgrid.SendGridAPIClient(SENDGRID_API_KEY)    from_email =
```

Email("xxxxxxxxxxxxx.foryoy@gmail.com") # Change to your verified sender

```

to_email = To(email) # Change to your recipient    subject = "Nutrition is a
basic human need and a prerequisite for healthy life"    content =
Content("text/plain",
        "Thank you for creating an account on our platform. Now you can utilise our platform
        "to maintain a healthier life.")    mail =
Mail(from_email, to_email, subject, content)

# Get a JSON-ready representation of the Mail object

mail_json = mail.get()

# Send an HTTP POST request to /mail/send
response = sg.client.mail.send.post(request_body=mail_json)

# print(response.status_code)

# print(response.headers)

def custom_send_mail(email, otp):

    sg = sendgrid.SendGridAPIClient(SENDGRID_API_KEY)    from_email =
Email("xxxxxxxxxx@gmail.com") # Change to your verified sender    to_email =
To(email) # Change to your recipient    subject = "Nutrition is a basic human
need and a prerequisite for healthy life"

    content = Content("text/plain",
        f"OTP : '{otp}'")

    mail = Mail(from_email, to_email, subject, content)

```

# Get a JSON-ready representation of the Mail object

```
mail_json = mail.get()
```

# Send an HTTP POST request to /mail/send    response =

```
sg.client.mail.send.post(request_body=mail_json)
```

# print(response.status\_code)

# print(response.headers)

```
def generateOTP():
```

```
    digits = "0123456789"
```

```
    OTP = ""    for i in
```

```
        range(6):
```

```
            OTP += digits[math.floor(random.random() * 10)]
```

```
    return OTP
```

```
def get_history():
```

```
    history = []
```

```
    sql = f"SELECT * FROM PERSON WHERE email = '{session['email']}'"    stmt =  
    ibm_db.exec_immediate(conn, sql)
```

```
    dictionary = ibm_db.fetch_both(stmt)    while
```

```
        dictionary:
```



```
history.append(dictionary)
```

```
dictionary = ibm_db.fetch_both(stmt)
```

```
return history
```

```
def get_history_person(email):
```

```
    history = []    sql = f"SELECT * FROM PERSON WHERE
```

```
    email = '{email}'"    stmt = ibm_db.exec_immediate(conn,
```

```
    sql)    dictionary = ibm_db.fetch_both(stmt)    while
```

```
    dictionary:
```

```
        history.append(dictionary)
```

```
    dictionary = ibm_db.fetch_both(stmt)
```

```
    return history
```

```
def get_history_person_time(time):
```

```
    historys = []
```

```
    sql = f"SELECT * FROM PERSON WHERE time = '{time}'"
```

```
    stmt = ibm_db.exec_immediate(conn, sql)    dictionary
```

```
    = ibm_db.fetch_both(stmt)    while dictionary:
```

```
        historys.append(dictionary)
```

```
    dictionary = ibm_db.fetch_both(stmt)
```

```
    return historys
```

```

def get_user():

    user = []    sql = f"SELECT * FROM USER"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_both(stmt)    while

dictionary:

    user.append(dictionary)

dictionary    =    ibm_db.fetch_both(stmt)

return user


backend = default_backend() def

aes_gcm_encrypt(message: bytes, key: bytes) -> bytes:

    current_time = int(time.time()).to_bytes(8, 'big')    algorithm

= algorithms.AES(key)    iv =

secrets.token_bytes(algorithm.block_size // 8)    cipher =

Cipher(algorithm, modes.GCM(iv), backend=backend)

encryptor = cipher.encryptor()

encryptor.authenticate_additional_data(current_time)

ciphertext = encryptor.update(message) + encryptor.finalize()

return b64e(current_time + iv + ciphertext + encryptor.tag)


def aes_gcm_decrypt(token: bytes, key: bytes, ttl=None) -> bytes:

```

```

algorithm = algorithms.AES(key)    try:

    data = b64d(token)    except

(TypeError, binascii.Error):

    raise InvalidToken            timestamp, iv, tag = data[:8],

data[8:algorithm.block_size // 8 + 8], data[-16:]    if ttl is not None:

    current_time = int(time.time())    time_encrypted, = int.from_bytes(data[:8],

'big')    if time_encrypted + ttl < current_time or current_time + 60 <

time_encrypted:        # too old or created well before our current time + 1 h to

account for clock skew        raise InvalidToken    cipher = Cipher(algorithm,

modes.GCM(iv, tag), backend=backend)    decryptor = cipher.decryptor()

decryptor.authenticate_additional_data(timestamp)    ciphertext = data[8 + len(iv):-

16]    return decryptor.update(ciphertext) + decryptor.finalize()

@app.route('/', methods=['GET', 'POST']) @app.route('/home', methods=['GET',

'POST']) def homepage():    if request.method == 'POST' and 'email' in request.form

and 'pass' in request.form:

        error = None        username =

request.form['email']        password

= request.form['pass']        user =

None

        if username == "":

```

```

        error = 'Incorrect username.'        return

render_template('index.html', error=error)

        if password ==
"""
        error = 'Incorrect password.'        return

render_template('index.html', error=error)

sql = "SELECT * FROM ADMIN WHERE email =?"

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, username)

ibm_db.execute(stmt)

account = ibm_db.fetch_assoc(stmt)    if account:

    print(aes_gcm_decrypt(account['PASSWORD'], bytes(KEY, 'utf8')))

    print(bytes(password, 'utf-8'))

    if aes_gcm_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')) ==
bytes(password, 'utf-8'):

        user = account['NAME']

email = account["EMAIL"]

session["loggedIn"] = None

session['name'] = user

session['email'] = email        msg =

None        history = get_history() # end

of user

```

```

        list =
get_user()

        return render_template('adminpanel.html', user=user, list=list, email=email, msg=msg)

return render_template('index.html', error="Wrong Password!")

sql = "SELECT * FROM USER WHERE email =?"

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, username)

ibm_db.execute(stmt)      account =

ibm_db.fetch_assoc(stmt)    if not account:

return render_template('index.html', error="Username

not found!")


print(aes_gcm_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')))

print(bytes(password, 'utf-8'))    if aes_gcm_decrypt(account['PASSWORD'], bytes(KEY,

'utf-8')) == bytes(password, 'utf-8'):

    user = account['NAME']

email = account["EMAIL"]

session["loggedIn"] = 'loggedIn'

session['name'] = user

session['email'] = email      msg =

None      history = get_history() # end

of user

```

```

        list =
get_user()

        return render_template('dashboard.html', user=user, email=email, msg=msg,
history=history)    return render_template('index.html', error="Wrong
Password!")

elif request.method == 'POST' and 'deleteHistory' in request.form:
    sql = f"SELECT * FROM PERSON WHERE email='{session['email']}'"

    print(sql)    stmt =

ibm_db.exec_immediate(conn, sql)

list_of_history = ibm_db.fetch_row(stmt)    if

list_of_history:

    sql = f"DELETE FROM PERSON WHERE email='{session['email']}'"

    stmt = ibm_db.exec_immediate(conn, sql)

    history = get_history()    if history:

        return render_template("dashboard.html", msg="Delete successfully",
user=session['name'],

        email=session['email'])

    return render_template("dashboard.html", msg="Delete successfully", user=session['name'],

        email=session['email'])

elif request.method == 'POST' and 'logout' in request.form:

    session["loggedIn"] = None    session['name'] = None

    session['email'] = None    return render_template('index.html',

```

```
error="Successfully Logged Out!") elif request.method == 'POST' and
```

```
'extra_submit_param_view' in request.form:
```

```
nutrition_list = request.form["extra_submit_param_view"]
```

```
history = get_history() splitted_nutrition =
```

```
nutrition_list.split(",")
```

```
return render_template('dashboard.html', user=session['name'], email=session['email'],  
data=splitted_nutrition,
```

```
history=history)
```

```
elif request.method == 'POST' and 'extra_submit_param_delete' in request.form:
```

```
time_identity = request.form["extra_submit_param_delete"]
```

```
history = get_history() sql = f"SELECT * FROM PERSON WHERE
```

```
time='{escape(time_identity)}'" stmt = ibm_db.exec_immediate(conn,
```

```
sql) row = ibm_db.fetch_row(stmt) if row:
```

```
sql = f"DELETE FROM PERSON WHERE time='{escape(time_identity)}'"
```

```
stmt = ibm_db.exec_immediate(conn, sql) history = get_history()
```

```
if history:
```

```
return render_template("dashboard.html", history=history, msg="Delete successfully")  
return render_template("dashboard.html", msg="Delete successfully")
```

```
return render_template("dashboard.html", history=history, msg="Something went wrong,  
Try again")
```

```
elif request.method == 'POST' and 'extra_submit_param_record' in request.form:
```

```
email_user = request.form["extra_submit_param_record"]
```

```

        return render_template('adminpanal.html', user=session['name'], email=session['email'],
list=get_user(), history=get_history_person(email_user))

    elif request.method == 'POST' and 'extra_submit_param_delete_user' in request.form:

        email_user = request.form["extra_submit_param_delete_user"]

        sql = f"SELECT * FROM USER WHERE time='{escape(email_user)}'"

        stmt = ibm_db.exec_immediate(conn, sql)      row =

        ibm_db.fetch_row(stmt)

        if row:

            sql = f"DELETE FROM USER WHERE time='{escape(email_user)}'"      stmt =

            ibm_db.exec_immediate(conn, sql)

            sql = f"SELECT * FROM PERSON WHERE time='{escape(email_user)}'"

            stmt = ibm_db.exec_immediate(conn, sql)      row =

            ibm_db.fetch_row(stmt)      if row:

                sql = f"DELETE FROM PERSON WHERE time='{escape(email_user)}'"

                stmt = ibm_db.exec_immediate(conn, sql)      return

        render_template('adminpanal.html', user=session['name'], list=get_user())

    elif request.method == 'POST' and 'extra_submit_param_nutritions' in request.form:

        user_time = request.form["extra_submit_param_nutritions"]      user_of =

        get_history_person_time(user_time)      user_dic = user_of[0]

        splitted_nutrition = user_dic['NUTRITION'].split(",")      return

        render_template('adminpanal.html', user=session['name'], list=get_user(),

```



```

        history=get_history_person(user_dic["EMAIL"]), data=splitted_nutrition)    elif

request.method == 'POST' and 'extra_submit_param_delete_record' in request.form:

    email_user = request.form["extra_submit_param_delete_record"]

    user_of = get_history_person_time(email_user)    user_dic =

    user_of[0]    sql = f"SELECT * FROM PERSON WHERE

time='{escape(email_user)}'"    stmt = ibm_db.exec_immediate(conn,

sql)    row = ibm_db.fetch_row(stmt)    if row:

        sql = f"DELETE FROM PERSON WHERE time='{escape(email_user)}'"

    stmt = ibm_db.exec_immediate(conn, sql)

    return render_template('adminpanal.html', user=session['name'], list=get_user(),

        history=get_history_person(user_dic["EMAIL"]))

    elif session.get('loggedIn'):    history = get_history()    return

render_template('dashboard.html', user=session['name'], history=history)    return

render_template('index.html')

def allowed_file(filename):    return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/dashboard', methods=['GET', 'POST']) def

upload_file():    history = []

    # sql = "SELECT * FROM Students"    sql = f"SELECT * FROM

PERSON WHERE email = '{session['email']}'"    stmt =

```

```

ibm_db.exec_immediate(conn, sql)    dictionary =

ibm_db.fetch_both(stmt)    while dictionary:

    history.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)    if request.method

== 'POST':

    # check if the post request has the file part    if 'logout' in

request.form:

    session["loggedIn"] = None        session['name'] = None

session['email'] = None        return render_template('index.html',

error="Successfully created")

    if 'file' not in request.files:

# flash('No file part')

return redirect(request.url)

file = request.files['file']

    # If the user does not select a file, the browser submits an    # empty file without a

filename.

    if file.filename ==

":

    return render_template('dashboard.html', msg="File not found", history=history)

baseimage = file.read()    if file and allowed_file(file.filename):

    requests = service_pb2.PostModelOutputsRequest(
        # This is the model ID of a publicly available General model. You may use any other

public or custom        # model ID.

```

```

        # model_id="general-image-recognition"

        # model_id="food-item-recognition"         model_id="food-item-recognition",

user_app_id=resources_pb2.UserAppIDSet(app_id=YOUR_APPLICATION_ID),

        inputs=[                resources_pb2.Input(

data=resources_pb2.Data(image=resources_pb2.Image(base64=baseimage))

        )

    ],

)

response = stub.PostModelOutputs(requests, metadata=metadata)

if response.status.code != status_code_pb2.SUCCESS:

    return render_template('dashboard.html', msg=f'Failed {response.status}',
history=history)

        calcium = 0

vitaminb5 = 0

protein = 0

vitamind = 0

vitamina = 0

vitaminb2 = 0

carbohydrates = 0

fiber = 0         fat

= 0         sodium =

0         vitaminc =

0         calories =

```

0 vitaminb1

= 0 folicacid

= 0 sugar = 0

vitamink = 0

cholesterol = 0

potassium = 0

monounsaturatedfa

t = 0

polyunsaturatedfat

= 0

saturatedfat = 0

totalfat = 0

calciumu = 'g' vitaminb5u =

'g' proteinu = 'g' vitamindu =

'g'

vitaminau = 'g' vitaminb2u = 'g'

carbohydratesu = 'g' fiberu = 'g'

fatu = 'g' sodiumu = 'g'

vitamincu = 'g' caloriesu = 'cal'

vitaminb1u = 'g' folicacidu = 'g'

sugaru = 'g' vitaminku = 'g'

cholesterolu = 'g' potassiumu = 'g'

monounsaturatedfatu = 'g'

```
polyunsaturatedfat = 'g'
```

```
saturatedfat = 'g'         totalfat = 'g'
```

```
for concept in response.outputs[0].data.concepts:         print("%12s: %.2f" %  
(concept.name, concept.value))         if concept.value > 0.5:
```

```
    payload = "ingredientList=" + concept.name +  
    "&servings=1"
```

```
    response1 = res.request("POST", url, data=payload, headers=headers,  
    params=querystring)
```

```
    data = response1.json()         for i in range(0,  
1):
```

```
        nutri_array = data[i]
```

```
nutri_dic = nutri_array['nutrition']
```

```
nutri = nutri_dic['nutrients']
```

```
    for z in range(0,  
len(nutri)):
```

```
        temp = nutri[z]         if temp['name'] == 'Calcium':
```

```
calcium += temp['amount']         calciumu = temp['unit']         elif
```

```
temp['name'] == 'Vitamin B5':         vitaminb5 += temp['amount']
```

```
vitaminb5u = temp['unit']
```

```
    elif temp['name'] == 'Protein':
```

```
protein += temp['amount']
```

```
proteinu = temp['unit']         elif
```

```
temp['name'] == 'Vitamin D':
```

```
vitamind += temp['amount'] vitamindu =
```

```
temp['unit']
```

```
elif temp['name'] == 'Vitamin A': vitamina +=
```

```
temp['amount'] vitaminau = temp['unit'] elif
```

```
temp['name'] == 'Vitamin B2': vitaminb2 += temp['amount']
```

```
vitaminb2u = temp['unit'] elif temp['name'] == 'Carbohydrates':
```

```
carbohydrates += temp['amount'] carbohydratesu = temp['unit']
```

```
elif temp['name'] == 'Fiber':
```

```
fiber += temp['amount']
```

```
fiberu = temp['unit'] elif
```

```
temp['name'] == 'Vitamin C':
```

```
vitaminc += temp['amount']
```

```
vitamincu = temp['unit'] elif
```

```
temp['name'] == 'Calories':
```

```
calories += temp['amount']
```

```
caloriesu = 'cal' elif
```

```
temp['name'] == 'Vitamin B1': vitaminb1 +=
```

```
temp['amount']
```

```
vitaminb1u = temp['unit']
```

```
elif temp['name'] == 'Folic Acid': folicacid
```

```
+= temp['amount'] folicacidu =
```

```

temp['unit']                elif temp['name'] == 'Sugar':

sugar += temp['amount']          sugaru =

temp['unit']                elif temp['name'] == 'Vitamin

K':                vitamink += temp['amount']

vitaminku = temp['unit']                elif temp['name']

== 'Cholesterol':                cholesterol +=

temp['amount']                cholesterolu =

temp['unit']                elif temp['name'] == 'Mono

Unsaturated Fat':                monounsaturatedfat +=

temp['amount']                monounsaturatedfatu =

temp['unit']                elif temp['name'] == 'Poly

Unsaturated Fat':                polyunsaturatedfat +=

temp['amount'] polyunsaturatedfatu = temp['unit']

elif temp['name'] ==

'Saturated Fat':

                saturatedfat += temp['amount']                saturatedfatu

= temp['unit']                elif temp['name'] == 'Fat':

                fat += temp['amount']                fatu =

temp['unit']                elif temp['name'] == 'Sodium':

sodium += temp['amount']                sodiumu =

temp['unit']                elif temp['name'] == 'Potassium':

```

```

potassium += temp['amount']                potassiumu =

temp['unit']                                else:

pass

totalfat += saturatedfat + polyunsaturatedfat + monounsaturatedfat

data = [calories, totalfat, saturatedfat, polyunsaturatedfat, monounsaturatedfat,
cholesterol, sodium,

potassium, sugar, protein, carbohydrates, vitamina, vitaminc, vitamind, vitaminb5,
calcium]

unit = [caloriesu, "g", saturatedfatu, polyunsaturatedfatu, monounsaturatedfatu,
cholesterolu, sodiumu,

potassiumu, sugaru, proteinu, carbohydratesu, vitaminau, vitamincu, vitamindu,
vitaminb5u, calciumu]

to_string
=
"{0},{0},{0},{0},{0},{0},{0},{0},{0},{0},{0},{0},{0},{0}"
.format(data[0], data[1], data[2], data[3],

data[4],

data[5], data[6], data[7], data[8],

data[9],

data[10], data[11], data[12], data[13],

data[14], data[15])

current_time = strftime("%a, %d %b %Y %H:%M:%S", localtime())

sql = "SELECT * FROM PERSON"

stmt = ibm_db.prepare(conn, sql)

```



```

        # ibm_db.bind_param(stmt, 1, session['email'])

    ibm_db.execute(stmt)

    # account = ibm_db.fetch_assoc(stmt)

try:

    insert_sql = "INSERT INTO PERSON VALUES (?, ?, ?, ?)"

    prep_stmt = ibm_db.prepare(conn, insert_sql)

    ibm_db.bind_param(prepare_stmt, 1, session['name'])

    ibm_db.bind_param(prepare_stmt, 2, session['email'])

    ibm_db.bind_param(prepare_stmt, 3, to_string)

    ibm_db.bind_param(prepare_stmt, 4, current_time)

    ibm_db.execute(prepare_stmt)

    return render_template('dashboard.html', user=session['name'], email=session['email'],
data=data,

history=history, unit=unit) except ibm_db.stmt_error:

print(ibm_db.stmt_error())

    return render_template('dashboard.html', msg='Something wnt wrong',
user=session['name'],

email=session['email'], data=data, history=history)

    return render_template('dashboard.html', history=history) if session['name'] is None:

    return render_template('index.html')

    return render_template('dashboard.html', user=session['name'], email=session['email'],
history=history)

```

```
if __name__ == '__main__':
```

```
    app.debug = True
    app.run()
```

## Index.html

```
<!DOCTYPE html>
```

```
<html lang="en"> <head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle
.min.js" integrity="sha384-
OERcA2EqJJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>

  <title>Nutrition Assistant</title>
  <link rel="icon" href="{{ url_for( 'static', filename =
'src/cardiogram.png') }}">
  <link rel="stylesheet" href="{{ url_for( 'static', filename =
'css/index.css') }}">
  <style>

</style> </head> <body>

{% if error %}
  <p style="font-size:20px;padding:20px;background-color:red;borderradius:20px;margin: 10px
40px">{{error}}</p>
{% endif %}
```

```

<div class="container-fluid">
  <div class="row">
    <div class="col-lg-5 colh heading">
      <p class="display-3">Nutrition Assistant</p>
    </div>
    <div class="col-lg-7 colh">
      <div>
        <ul class="lists">
          <li><a id="c1" href="#cont_1">Home</a></li>
          <li onmouseover = "signinOne(this)"
onmouseout="unsigin(this)"><a id="c2" href="#logins">SignIn</a></li>
          <li onmouseover = "signinTwo(this)"
onmouseout="unsigin(this)"><a id="c3" href="#logins">SignUp</a></li>
          <li onmouseover = "visible(this)"
onmouseout="unsigin(this)"><a id="c4" href="#abouts">About</a></li>
        </ul>
      </div>
    </div>
  </div>
</div>

```

```

<div class="container-fluid content">
  <div class="row">
    <div class="col-lg info">
      <p>
        Nutritional support is the provision of adequate nutrients to maintain a
        healthy body weight and avoid malnutrition. The continuous delivery of

```

high-quality and cost-effective nutritional care to patients has been shown to be an increasingly difficult task. It is observed that dieticians are requested to carry out the nutritional assessment, to manually calculate the nutritional needs and to design the everyday meal plan for each patient. In most cases, these time-consuming tasks are not completed due to lack of time or inadequate number of personnel. Development of a computer assisted information tool with cloud-based on-line diet consultation module and comparison of its efficacy with one- to-one counselling would be efficiently utilized for client education intervention programs. The nutrient content calculation was planned to undertake with commonly consumed traditional as well as junk foods

</p>

</div>

<div class="col-lg info" id="infos">

<div class="cont\_1 pb-0">

<div onclick="ont()" id="l1" class="options">

<p><p class="list">Home</p> </div>

<div id="l2" class="options">

<p><p class="list">Sign

In</p>

</div>

<div id="l3" class="options">

```

        <p><p class="list">Sign Up</p>

    </div>

    <div id="l4" class="options">

        <p><p class="list">about</p>        </div>


    <div id="logins" class="login">

        <div class="container" id="container">

            <div class="form-container sign-up-container">                <form action="{{ url_for('signup')
}}" method="post" enctype="multipart/form-data">

                <h1>Create Account</h1>

                <input type="text" name="name" placeholder="Name" required />

                <input type="email" name="email" placeholder="Email" required />

                <input type="password" name="pass" placeholder="Password" required />

                <button>Sign Up</button>

            </form>

        </div>

        <div class="form-container sign-in-container">

            <form action="{{ url_for('homepage') }}" method="post" enctype="multipart/form-data">

<h1>Sign in</h1>

```

```

    <input STYLE="display:none" type="text" name="name" placeholder="Name"/>
    <input type="email" name="email" placeholder="Email" required />
    <input type="password" name="pass" placeholder="Password" required />
    <a href="{{url_for('homepage')}}">Forgot your password?</a>
    <button>Sign In</button>
  </form>
</div>
<div class="overlay-container">
  <div class="overlay">
    <div class="overlay-panel overlay-left">
      <h1>Welcome Back!</h1>
      <p>To keep connected with us please login with your personal info</p>
<button class="ghost" id="signIn">Sign In</button>
    </div>
    <div class="overlay-panel overlay-right">
      <h1>Hello, Friend!</h1>
      <p>Enter your personal details and start journey with us</p>
<button class="ghost" id="signUp">Sign Up</button>
    </div>
  </div>
</div>
<div class="about" id="abouts">
  <p>
    A web based tool is being planned for therapeutic nutrition

```

prescriptions in clinical settings.

The cloud based system would have the ability to calculate the nutritional requirements and to

```

        guide first line nutritional management to patients and clients automatically.
    </p>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

</body>
<script src="{{ url_for('static', filename='js/index.js') }}">

function ont() {
    document.getElementById('info').style.color = 'red';
}
</script>
</html>

```

## index.css

```

@media only screen and (max-width :768px ) {
    .colh {
height: auto;
    }

    .lists {
height: 330px;
    }
}

```

```
        overflow: auto;    flex-
direction: column;
    }

}

body {
    background-color: rgb(255, 255, 255);
    font-family: Futura ,Trebuchet MS,Arial,sans-serif;
    min-height: 100vh;    font-size: 1.3rem;

}

.heading {
    color: rgb(25, 23, 23);
}

.colh    {
height:    auto;
display: flex;
    justify-content: center;
align-items: center;    padding:
50px 0;
}

.colh > div {
width: 100%;
}
```



```
.spaceimp {  
  margin-bottom: 200px;  
}  
  
.lists {  
  list-style-type: none;  
display: flex;  font-  
size: 1.2rem;  
  justify-content: space-around;  
align-items: center;  align-  
content: center;  transition: 2.8s;  
}  
  
.lists > li {  
  text-decoration: none;  
}  
  
.lists > li > a {  
  text-decoration: none;  
padding: 10px 40px;  
color: rgb(11, 9, 9);  
transition: .8s ease-in;  
border-radius: 26px;  
background: #e3e3e3;  
  box-shadow: inset -17px 17px 33px #d1d1d1,  
inset 17px -17px 33px #f5f5f5;  
}
```

```
.lists > li > a:hover {
color: rgb(251, 249, 249);
border-radius: 26px;
background: #131111;
    box-shadow: -11px 11px 22px #8a8484,
        11px -11px 22px #a49d9d;
}

p {
    text-align: justify;
}

.info {
    min-height: 50vh;
color: rgb(1, 9, 9);    min-
width: 10vw;    padding-
left: 30px;
}

.content {    margin-top:
50px;    padding-bottom:
50px;        box-sizing:
border-box;
}

.content > info > p {    font-
size: 2.5rem;
}

.cont_1 {
width: 100%;
```

```
height: 100%; display: flex;
justify-content: top; align-
items: flex-end; flex-
direction: column; flex-
grow: 1; padding: 0 10px 0
0; position: relative;
}
```

```
.login {
width: 100%;
height: 100%;
left: 0;
top: 0;
position: absolute; z-
index: 1; display: none;
margin-bottom: 50px;
animation-name: loginhider;
animation-duration: 2s;
}
```

```
.about {
background-color: rgba(188, 196, 196, 0.8);
min-height: 200px; width: 80%;
border-radius: 25px;
margin-right: 10%; margin-top: 100px;
padding: 20px;
```

```
-webkit-box-shadow: 3px 3px 5px 6px rgb(89, 82, 82); /* Safari 3-4, iOS 4.0.2 - 4.2, Android 2.3+ */
-moz-box-shadow: 3px 3px 5px 6px rgb(89, 82, 82); /* Firefox 3.5 3.6 */
box-shadow: 3px 3px 5px 6px rgb(89, 82, 82); display: none;

  animation-name: loginhider;      animation-
duration: 2s;
}

@keyframes loginhider {
  0% {opacity: 0;}
  50% {opacity: .5;} 100% {opacity:
1;}
}

.list {
  padding-left: 80px;
}

.options{
  padding: 0px 15px;
  background-color: rgba(188, 196, 196,0.8);
box-sizing: border-box; text-align: center;
margin-bottom: 20px; border-radius:
25px; display: flex; width: 90%;
  transition: opacity .9s;
}
```

```
.options > p {  
display: flex;  
justify-content: center;  
align-items: center; align-  
content: center; font-size:  
1.2rem; font-style: bold;  
}
```

```
#l1 {  
position: relative;  
animation-name: example;  
animation-duration: 1s;  
}
```

```
#l2 {  
position: relative;  
animation-name: example;  
animation-duration: 1.5s;  
}
```

```
#l3 {  
position: relative;  
animation-name: example;  
animation-duration: 2s;  
}
```

```
#l4 {  
position: relative;
```

```
    animation-name: example;  
animation-duration: 2.5s;  
}
```

```
@keyframes example {
```

```
0% {left:800px;}
```

```
    1% {left:792px;}
```

```
    2% {left:784px;}
```

```
    3% {left:776px;}
```

```
    4% {left:768px;}
```

```
    5% {left:760px;}
```

```
    6% {left:752px;}
```

```
    7% {left:744px;}
```

```
    8% {left:736px;}
```

```
    9% {left:728px;}
```

```
    10% {left:720px;}
```

```
    11% {left:712px;}
```

```
    12% {left:704px;}
```

```
    13% {left:696px;}
```

```
    14% {left:688px;}
```

```
    15% {left:680px;}
```

```
    16% {left:672px;}
```

```
    17% {left:664px;}
```

```
    18% {left:656px;}
```

```
    19% {left:648px;}
```

```
    20% {left:640px;}
```

```
    21% {left:632px;}
```

```
    22% {left:624px;}
```

```
    23% {left:616px;}
```

```
    24% {left:608px;}
```

```
    25% {left:600px;}
```

26% {left:592px;}

27% {left:584px;}

28% {left:576px;}

29% {left:568px;}

30% {left:560px;}

31% {left:552px;}

32% {left:544px;}

33% {left:536px;}

34% {left:528px;}

35% {left:520px;}

36% {left:512px;}

37% {left:504px;}

38% {left:496px;}

39% {left:488px;}

40% {left:480px;}

41% {left:472px;}

42% {left:464px;}

43% {left:456px;}

44% {left:448px;}

45% {left:440px;}

46% {left:432px;}

47% {left:424px;}

48% {left:416px;}

49% {left:408px;}

50% {left:400px;}

51% {left:392px;}

52% {left:384px;}

53% {left:376px;}

54% {left:368px;}

55% {left:360px;}

56% {left:352px;}

57% {left:344px;}

58% {left:336px;}  
59% {left:328px;}  
60% {left:320px;}  
61% {left:312px;}  
62% {left:304px;}  
63% {left:296px;}  
64% {left:288px;}  
65% {left:280px;}  
66% {left:272px;}  
67% {left:264px;}  
68% {left:256px;}  
69% {left:248px;}  
70% {left:240px;}  
71% {left:232px;}  
72% {left:224px;}  
73% {left:216px;}  
74% {left:208px;}  
75% {left:200px;}  
76% {left:192px;}  
77% {left:184px;}  
78% {left:176px;}  
79% {left:168px;}  
80% {left:160px;}  
81% {left:152px;}  
82% {left:144px;}  
83% {left:136px;}  
84% {left:128px;}  
85% {left:120px;}  
86% {left:112px;}  
87% {left:104px;}  
88% {left:96px;}  
89% {left:88px;}



```
90% {left:80px;}
91% {left:72px;}
92% {left:64px;}
93% {left:56px;}
94% {left:48px;}
95% {left:40px;}
96% {left:32px;}
97% {left:24px;}
98% {left:16px;} 99%
{left:8px;}
100% {left:0px;}
}

h1 {
font-weight: bold;
margin: 0;
}

h2 {
text-align: center;
}

p {
font-size: 14px; font-
weight: 100; line-height:
20px; letter-spacing: 0.5px;
margin: 20px 0 30px;
}

.info > p {
```

```
font-size: 1.2rem ; font-  
weight: 20; line-height:  
30px;  
letter-spacing: 1px; margin: 20px 0 30px;  
} a { color: #333;  
font-size: 14px; text-  
decoration: none;  
margin: 15px 0;  
}  
button {  
border-radius: 20px;  
border: 1px solid #FF4B2B; background-color: #FF4B2B;  
color: #FFFFFF; font-size: 12px; font-weight: bold;  
padding: 12px 45px; letter-spacing: 1px; text-  
transform: uppercase; transition: transform 80ms ease-  
in;  
}  
button:active {  
transform: scale(0.95);  
}  
  
button:focus { outline: none;  
}
```

```
button.ghost {
  background-color: transparent; border-
color: #FFFFFF;
}

form {
  background-color:
#FFFFFF; display: flex;
align-items: center;
justify-content: center;
flex-direction: column;
padding: 0 50px; height:
100%; text-align: center;
}

input {
  background-color: #eee;
border: none; padding:
12px 15px; margin: 8px
0; width: 100%;
}

.container {
  background-color: #fff; border-
radius: 10px;
  box-shadow: 0 14px 28px rgba(0,0,0,0.25),
  0 10px 10px rgba(0,0,0,0.22);
position: relative; overflow:
hidden; width: 768px;
```

```
max-width: 100%; min-
height: 480px;
}

.form-container {
position: absolute;
top: 0; height:
100%;
transition: all 0.6s ease-in-out;
}

.sign-in-container {
left: 0;
width: 50%; z-index: 2;
}

.container.right-panel-active .sign-in-container {
transform: translateX(100%);
}

.sign-up-container {
left: 0;
width: 50%; opacity: 0; z-index:
1;
}

.container.right-panel-active .sign-up-container {
transform: translateX(100%); opacity: 1; z-
index: 5;
```

```
animation: show 0.6s;
}

@keyframes show {
0%, 49.99% {
    opacity: 0; z-index: 1;
}

50%, 100% {
    opacity: 1; z-index: 5;
}
}

.overlay-container {
position: absolute;
top: 0; left: 50%;
width: 50%;
height: 100%;
overflow: hidden;
transition: transform 0.6s ease-in-out; z-
index: 100;
}

.container.right-panel-active .overlay-container{
transform: translateX(-100%);
}

.overlay {
background: #FF416C;
```

```
background: -webkit-linear-gradient(to right, #FF4B2B, #FF416C);
background: linear-gradient(to right, #FF4B2B, #FF416C);
background-repeat: no-repeat; background-size: cover;
background-position: 0 0; color: #FFFFFF; position: relative;
left: -100%; height: 100%; width: 200%;
transform: translateX(0);
transition: transform 0.6s ease-in-out;
}
```

```
.container.right-panel-active .overlay {
transform: translateX(50%);
}
```

```
.overlay-panel {
position: absolute;
display: flex; align-
items: center; justify-
content: center; flex-
direction: column;
padding: 0 40px; text-
align: center; top: 0;
height: 100%; width:
50%;
transform: translateX(0);
transition: transform 0.6s ease-in-out;
}
```

```

.overlay-left {
transform: translateX(-20%);
}

.container.right-panel-active .overlay-left {
transform: translateX(0);
}

.overlay-right {
right: 0;
transform: translateX(0);
}

.container.right-panel-active .overlay-right {
transform: translateX(20%);
}

/*input[type=text], input[type=password] {
width: 100%; padding: 2px 10px;
margin: 8px 0; display: inline-block;
border: 0px solid #ccc; box-sizing: border-box;
border: 1;
}

button {
width: 100%;

margin-top: 30px; background-color: green; border: 0; padding: 5px;
}*/

```

## index.js

```
const signUpButton = document.getElementById('signUp');
const signInButton = document.getElementById('signIn'); const container =
document.getElementById('container');

const switchone = document.getElementById('c1');
const switchtwo = document.getElementById("c2");
const switchthree = document.getElementById('c3');
const switchfour = document.getElementById('c4');

const Fswitchone = document.getElementById('l1');
const Fswitchtwo = document.getElementById("l2");
const Fswitchthree = document.getElementById('l3');
const Fswitchfour = document.getElementById('l4');

const space = document.getElementById('infos');

var pre_state = 0;
```



```

var stateone = 0;
var statetwo = 0;
var statethree = 0;

signupButton.addEventListener('click', () => { container.classList.add("right-panel-active");
});

signInButton.addEventListener('click', () => { container.classList.remove("right-panel-
active"); });

switchone.addEventListener('click', remover); switchtwo.addEventListener('click', signin);
switchthree.addEventListener('click', Signup) switchfour.addEventListener('click', about);

Fswitchone.addEventListener('click', remover); Fswitchtwo.addEventListener('click', signin);
Fswitchthree.addEventListener('click', Signup)
Fswitchfour.addEventListener('click', about);

function remover() {

if(pre_state == 1){
pre_state = 0;
space.classList.remove("spaceimp");
document.getElementById("abouts").style.display = "none";
document.getElementById("logins").style.display = "none";

```

```

    document.getElementById("l1").style.display = "flex";
document.getElementById("l2").style.display = "flex";
    document.getElementById("l3").style.display = "flex";    document.getElementById("l4").style.display =
"flex";
}
}

function div_adder () {
    space.classList.add("spaceimp");
    document.getElementById("abouts").style.display = "none";
document.getElementById("logins").style.display = "block";
document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
    document.getElementById("l3").style.display = "none";    document.getElementById("l4").style.display =
"none";
}

function about_adder () {
    //space.classList.add("spaceimp");
    // remover();
    document.getElementById("abouts").style.display = "block";
document.getElementById("l1").style.display    =    "none";
document.getElementById("l2").style.display = "none";
    document.getElementById("l3").style.display = "none";    document.getElementById("l4").style.display =
"none";
}

function signin() {

    if(pre_state == 0) {

```

```

        pre_state = 1;
stateone = 1;    statetwo
= 0;    statethree = 0;
        container.classList.remove("right-panel-active");
div_adder();
    }else {
        if(stateone == 0) {
pre_state = 1;
stateone = 1;    statetwo
= 0;    statethree = 0;
        container.classList.remove("right-panel-active");
div_adder();
    }else {
remover();
    }
    }
}

function Signup() {

    if(pre_state == 0) {
pre_state = 1;
stateone = 0;    statetwo
= 1;    statethree = 0;
        container.classList.add("right-panel-active");
div_adder();
    }else {

```

```

    if(statetwo == 0) {
pre_state = 1;
stateone = 0;    statetwo
= 1;    statethree = 0;
        container.classList.add("right-panel-active");
div_adder();
    }else {
remover();
    }
}
}
}

```

```

function about() {
if(pre_state == 0){
pre_state = 1;
stateone = 0;    statetwo
= 0;    statethree = 3;
about_adder();
}else{
    if(statethree == 0){
remover();    pre_state
= 1;    stateone = 0;
statetwo = 0;
statethree = 3;
about_adder();
    }else{
remover();
    }
}
}

```

```

}

}

function invisible(x) {
if(pre_state == 0) {
    document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
document.getElementById("l3").style.display = "none";    document.getElementById("l4").style.display
= "none";
    }

}

function visible(x){
if(pre_state == 0) {
    document.getElementById("abouts").style.display = "block";
    //space.classList.add("spaceimp");
    container.classList.add("right-panel-active");
    document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
document.getElementById("l3").style.display = "none";    document.getElementById("l4").style.display
= "none";
    }
}

function unsigin(x) {
if(pre_state == 0){
    container.classList.remove("right-panel-active");
space.classList.remove("spaceimp");
    document.getElementById("logins").style.display = "none";

```

```

    document.getElementById("abouts").style.display = "none";
document.getElementById("l1").style.display = "flex";
document.getElementById("l2").style.display = "flex";
document.getElementById("l3").style.display = "flex";    document.getElementById("l4").style.display
= "flex";
    }

}

function signinOne(x){    if(pre_state == 0) {    container.classList.remove("right-
panel-active");    space.classList.add("spaceimp");

    document.getElementById("logins").style.display = "block";
document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
document.getElementById("l3").style.display = "none";    document.getElementById("l4").style.display
= "none";
    }
}

function signinTwo(x){    if(pre_state == 0) {

    document.getElementById("logins").style.display = "block";
space.classList.add("spaceimp");

    container.classList.add("right-panel-active");

    document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
document.getElementById("l3").style.display = "none";
document.getElementById("l4").style.display = "none";

```

```

}

function setcon(x) {
if(pre_state == 0) {
    document.getElementById("abouts").style.display = "block";
    //space.classList.add("spaceimp");    container.classList.add("right-panel-
active");
    document.getElementById("l1").style.display = "none";
document.getElementById("l2").style.display = "none";
document.getElementById("l3").style.display = "none";
document.getElementById("l4").style.display = "none";
    }
}
}

```

## 7.2 Feature 2 dashboard.html

```

<!DOCTYPE html> <html lang="en"> <head>
3  <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle

```

```

.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbaw3"
crossorigin="anonymous"></script>

<link rel="stylesheet" href="{{url_for('static', filename='css/dashboard.css')}}">
<link rel="icon" href="{{ url_for( 'static', filename =
'src/cardiogram.png')}}">
<title>Welcome {{user}}</title>

</head>
<body>

    {% if msg %}
    <div class="msg bg-info" style="padding: 0px 0 0px 50px;margin: 20px
20px 0 20px;border-radius: 20px;">
        <h4>{{msg}}</h4>
    </div>
    {% endif %}

    <div class="container-fluid dash">
        <div class="header p-3">
            <h3>&nbsp;&nbsp;&nbsp;Dashboard</h3>
            <div style="display:flex; justify-content: right;align-items: center;">Welcome
{{user}},&nbsp;&nbsp;&nbsp;<form action="" method="post" enctype="multipart/form-data"><button
type="submit" name="logout" class="combutton btns">Log Out</button></form></div>
        </div>
    </div>

```



```

<div class="container-fluid">
  <div class="row rowh">
    <div class="col-lg-4 colh">
      <div class="maincon comcolor">
        <h4>Control panal</h4>
        <h5><form action="" name="deleteHistory" method="post"
enctype="multipart/form-data"><input style="width:
200px;border:0px;padding:10px 40px;border-radius:20px;" type="submit"
name="deleteHistory" value="Delete All History"></form></h5>

        <table>
          <tr>
            <th>History</th>
          </tr>
          {% for row in history %}
            <tr>
              <td>{{ row['TIME'] }}</td>
              <td><form
method="post" action="{{url_for('homepage')}}" class="inline">
<input type="hidden" name="extra_submit_param_view"
value="{{row['NUTRITION']}}">
              <button type="submit"
name="submit_param" value="submit_value" class="link-button">
                View
              </button>
            </form>
          </td>
          <td><form method="post"
action="{{url_for('homepage')}}" class="inline">
<input type="hidden" name="extra_submit_param_delete"
value="{{row['TIME']}}">
            <button type="submit" name="submit_param" value="submit_value"
class="linkbutton">
              Delete

```

```

        </button>                </form>

    </td>

</tr>

{% endfor %}

</table>

</div>    </div>

<div class="col-lg-8 row colh">

    <div class="row normsize">

        <div class="col-lg normsize roudcorner comcolor">

            <div class="comflex-col">

<button class="combutton btns" onclick="setImage()" >Clear Image</button>

            </div>                </div>

        <div class="col-lg normsize roudcorner comcolor">

            <div class="comflex lesssize normpadding">

                <div>

                    <h1>Upload Image</h1>

                    <form action="{{url_for('upload_file')}}" method="post" enctype="multipart/formdata">

                        <input type=file onchange="readURL(this);" name="file">

```

```

        <input style="margin: 10px 0px;" onclick="setImage()" type=submit value=Upload
name="upload">

    </form>

</div>

</div>

</div>

</div>

</div>

</div>

{ % if data %}

<div class="container-fluid float">

    <div class="containers floatcontainer ">

        <div class="box1">

            <div class="close">

                <a href="{{url_for('upload_file',methods='POST')}}" class="closes"></a>

            </div>

        </div>

        <div style="background-color: rgb(105, 102, 102);margin-top: 25px;font-size: 30px;font-weight:
bold;padding-left: 15px;"><p>Nutrition Facts</p></div>

        <div class="box2">

            <div class="bcol">

                <table style="width:100%;">

                    <tr>

                        <th>Calories</th>

                        <th>{{data[0]}}{{unit[0]}}</th>

```

```
</tr>

<tr>

  <th></th>

  <th>Daily Value</th>
</tr>

<tr>

  <th>Total Fat</th>

  <th>{{data[1]}}{{unit[1]}}</th>
</tr>

<tr>

  <td>Saturated Fat</td>

  <td>{{data[2]}}{{unit[2]}}</td>
</tr>

<tr>

  <td>Polyunsaturated Fat</td>

  <td>{{data[3]}}{{unit[3]}}</td>
</tr>

<tr>

  <td>Monounsaturated Fat</td>

  <td>{{data[4]}}{{unit[4]}}</td>
</tr>

<tr>

  <th>Cholesterol</th>

  <th>{{data[5]}}{{unit[5]}}</th>
</tr>

<tr>

  <th>Sodium</th>

  <th>{{data[6]}}{{unit[6]}}</th>
</tr>

<tr>

  <th>Potassium</th>

  <th>{{data[7]}}{{unit[7]}}</th>
```

```
</tr>

<tr>
  <th>Sugar</th>
  <th>{{data[8]}}{{unit[8]}}</th>
</tr>

<tr>
  <th>Protein</th>
  <th>{{data[9]}}{{unit[9]}}</th>
</tr>

<tr>
  <th>Carbohydrates</th>
  <th>{{data[10]}}{{unit[10]}}</th>
</tr>

<tr>
  <th>Vitamin A</th>
  <th>{{data[11]}}{{unit[11]}}</th>
</tr>

<tr>
  <th>Vitamin C</th>
  <th>{{data[12]}}{{unit[12]}}</th>
</tr>

<tr>
  <th>Vitamin D</th>
  <th>{{data[13]}}{{unit[13]}}</th>
</tr>

<tr>
  <th>Vitamin B5</th>
  <th>{{data[14]}}{{unit[14]}}</th>
</tr>

<tr>
  <th>Calcium</th>
  <th>{{data[15]}}{{unit[15]}}</th>
```

```

        </tr>
    </table>    </div>
</div>
</div> </div>
{% endif %}

<script>
    //image = document.getElementById('myImage');
function clearImage() {
    image.src = "{{url_for('static',filename='src/user.jpg')}}";
//onclick="document.getElementById('myImage').src='src/omplate.png'"    }

    function setImage() {
        image.src = "{{url_for('static',filename='src/food.jpg')}}";
    }
</script>
<script src="{{url_for('static', filename='js/dashboard.js')}}">

</script> </body> </html>

```

## dashboard.css

```
body {
```

```
background-color: rgb(192, 171, 171);
box-sizing: border-box;
}

.dash {
padding: 50px 0;
}

.header {
font-size: 1.2rem;
background-color: rgb(192, 171, 171);
padding: 50px; border: 1px solid
black; margin: 0 20px; box-sizing:
border-box; border-radius: 25px;
}

.header >h4{
text-align: right;
font-family: Arial, Helvetica, sans-serif;
}

.rowh {
min-height: 75vh;
}

.colh {
display: flex;
flex-direction: column; border-
radius: 20px;
```

```
padding: 40px;
}

.maincon {
padding: 40px;
border-radius: 20px;
height: 100%; text-
align: start;
background-color: aliceblue;
}

.maincon > h4 {
text-decoration: underline;
text-align: center; padding-
bottom: 40px;
}

.maincon > h5 {
padding: 10px 20px;
/*background-color: beige;*/
border-radius: 20px; margin-
bottom: 20px; position:
relative;
-webkit-transition-duration: 0.4s;
transition-duration: 0.4s;
}

.normsize {
height: 100%;
width: 100%;
box-sizing: border-box;
```



```
/* min-width: 250px; min-
height: 350px;*/
}

.lesssize {
width: 90%;
height: 90%;
}

.normpadding {
padding: 30px;
margin: 10px; box-
sizing: border-box;
}

.rouddcorner {
border-radius: 25px;
margin: 10px;
}

.comcolor {
background-color: aliceblue;
-webkit-box-shadow: 9px 10px 23px 7px rgba(0,0,0,0.75);
-moz-box-shadow: 9px 10px 23px 7px rgba(0,0,0,0.75); box-shadow: 9px 10px 23px 7px
rgba(0,0,0,0.75);
}

.comflex {
display: flex;
justify-content: center; align-
items: center;
```

```
}

/*floating list - view history*/

.float {
    position: absolute;
margin-inline: auto;
top: 25vh;    min-height:
30vh;    display: flex;
    justify-content: center;
}

.containers {
    width: min(calc(100% - 15%), 840px);    margin-
inline: auto;
}

.floatcontainer {    display: flex;    flex-
direction: column;    background-color:
white;    border-radius: 25px;
    -webkit-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);
    -moz-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);    box-shadow: 6px 6px 21px 4px
rgba(0,0,0,0.75);
}

.box1    {
display: flex;
```

justify-content: right;

```
    position: relative;
}

.closes {
    position: absolute;
    right: 32px;    top: 32px;
    width: 32px;    height:
32px;    opacity:
0.3; }

.closes:hover {
opacity: 1;
}

.closes:before, .clos
{    position: absolute;
left: 15px;    content: '';
height: 33px;    width: 2px;
    background-color: #333;
}

.closes:before {
    transform: rotate(45deg);
}

.closes:after {
    transform: rotate(-45deg);
}

.box2 {
    margin: 20px 40px;
```

```
display: flex; flex-direction: column;
}

.bcol{
padding: 10px; margin-bottom: 5px;
}

.inline      {
display: inline;
}

.link-button {
background: none;
border: none;
color: blue;
text-decoration: underline;
cursor: pointer;
font-size: 1em; font-family: serif;
}

.link-button:focus {
outline: none;
}

.link-button:active {
color:red;
}
```

## dashboard.js

```
const image = document.getElementById('myImage');

function readURL(input)
{
    //document.getElementById("bannerImg").style.display = "block";

    if (input.files && input.files[0]) {    var reader = new FileReader();

        reader.onload    =    function    (e)    {
image.src = e.target.result;
        }

        reader.readAsDataURL(input.files[0]);
    }
}
```

## forgot\_password.html

```
<!DOCTYPE html> <html lang="en"> <head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Forgot Password</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
```

```

crossorigin="anonymous">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle
.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8QbSw3"
crossorigin="anonymous"></script>

  <link rel="icon" href="{{ url_for( 'static', filename =
'src/cardiogram.png') }}">
  <link rel="stylesheet" href="{{ url_for( 'static', filename =
'css/forgot.css') }}">
  <link rel="stylesheet" href="{{ url_for( 'static', filename = 'css/dashboard.css') }}">

</head>
<body>
  {% if error %}
    <p style="font-size:20px;padding:20px;background-color:red;borderradius:20px;margin: 10px
40px">{{error}}</p>
  {% endif %}
  <div class="container-fluid float">
    <div class="containers floatcontainers">
      <div class="row maxheight displaytype p-3">

        <div class="col-lg-6 maincontainers">
          <div class="boxfor">
            <form action="{{url_for('forgot')}}" method="post">
              <input type="email" required name="f_email" value="" placeholder="abc@mail.com">
              <input type="submit" value="Send OTP">

```

```
        </form>
    </div>
</div>

<div class="col-lg-6 maincontainers">
    <div class="boxfor">
        <form action="{{url_for('forgot')}}" method="post">
            <input type="OTP" required name="f_otp" placeholder="OTP">
            <input type="password" required name="f_psw" placeholder="new password">
            <input type="password" required name="f_psws" placeholder="confirm password ">
            <input type="submit" value="Submit">
        </form>
    </div>
</div>

</div>
</div>
</div>
</body>
</html>
```

```
<!--<div class="col-lg-6 box1 maxheight displaytype">
    <div class="displaytype bg-danger ">
        <form action="" method="post">
            <input type="email" required name="f_email" placeholder="abc@mail.com">
```

```
            <input type="submit" value="Send OTP">
        </form>
    </div>
```



```
</div>

<div class="col-lg-6 box1 maxheight displaytype">

  <form action="" method="post">

    <input type="email" required name="f_email" placeholder="abc@mail.com">

    <input type="submit" value="Send OTP">

  </form>

</div> -->
```

## forgot.css

```
body {
  height: 100vh;  background-color: aqua;
}

.maincontainer { min-height: 100vh;
}

.maxheight {
  height: 100%;  width: 100%;
}

.displaytype {
  display: flex;  justify-content: center;
```

```

}

.floatcontainers {
display: flex;
background-color: white; border-
radius: 25px;
-webkit-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);
-moz-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);
box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75); }

.maincontainers {
display: flex;
justify-content: center; align-
items: center; align-content:
center;
}

.boxfor { display:
flex; flex-direction:
row; justify-content:
center;
}

```

## adminpanel.html

```

<!DOCTYPE html> <html lang="en"> <head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <title>Admin Panal</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
ZenH87qX5JnK2JI0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">

```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle
.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
```

```
<link rel="stylesheet" href="{{url_for('static', filename =
'css/admin.css')}}">
```

```
<link rel="stylesheet" href="{{url_for('static', filename=
'css/dashboard.css')}}">
```

```
<link rel="icon" href="{{url_for('static', filename=
'src/admin_panal.png')}}">
```

```
</head> <body>
```

```
{% if msg %}
```

```
<div class="msg bg-info" style="padding: 0px 0 0px 50px;margin: 20px
20px 0 20px;border-radius: 20px;">
```

```
<h4>{{msg}}</h4>
```

```
</div>
```

```
{% endif %}
```

```
<div class="container-fluid dash">
```

```
</div>
```

```
</div>
```

```
<div class="container-fluid">
```

```
<div class="header p-3">
```

```
<h3>&nbsp;&nbsp;&nbsp;Admin Panal</h3> <div style="display:flex; justify-content:
right;align-items: center;">Welcome {{user}},&nbsp;&nbsp;&nbsp;<form action="" method="post"
enctype="multipart/form-data"><button type="submit" name="logout" class="combutton
btns">Log Out</button></form></div>
```

```

<div class="row rowh">
  <div class="col-lg-4 colh">
    <div class="maincon comcolor">
      <h4>Control panal</h4>
      <table style="width: 100%;">
        <tr>
          <th>Users</th>
          </tr>
          {% for row in list %}
            <tr>
              <td>{{ row['NAME'] }}</td>
              <td><form method="post"
action="" class="inline">
                <input type="hidden" name="extra_submit_param_record"
value="{{row['EMAIL']}}">
                <button type="submit" name="submit_param"
value="submit_value" class="link-button">
                  View Record
                </button>
              </form>
            </td>
          </tr>
        </table>
      </div>
    </div>
  </div>
</div>

```

```

        </td>
        <td><form method="post" action="" class="inline">
            <input type="hidden" name="extra_submit_param_delete_user"
value="{{row['EMAIL']}}">
            <button type="submit" name="submit_param"
value="submit_value" class="link-button">
                Delete User
            </button>
        </form>
    </td>
</tr>
{% endfor %}
</table>

```

```

</div>
</div>
<div class="col-lg-3 colh">
    <div class="maincon comcolor">
        <h4>User History</h4>
        <table>
            <tr>
                <th>Records</th>
            </tr>
            {% for row in history %}
            <tr>
                <td>{{ row['TIME'] }}</td>
                <td><form method="post"
action="{{url_for('homepage')}}" class="inline">
                    <input type="hidden" name="extra_submit_param_nutritions"
value="{{row['TIME']}}">
                    <button type="submit"

```

```

name="submit_param" value="submit_value" class="link-button">
                View
            </td>
        </tr>
    </table>

```

```

                </button>
            </form>
        </td>
        <td><form method="post" action="{{url_for('homepage')}}" class="inline">
            <input
type="hidden" name="extra_submit_param_delete_record" value="{{row['TIME']}}">
            <button
type="submit" name="submit_param" value="submit_value" class="link-button">

                Delete
            </button>
        </form>
        </td>
    </tr>
{% endfor %}
</table>

</div>

</div>

<div class="col-lg-5 colh">
    <div class="normsize roudcorner comcolor">
        <div class="maincon comcolor">
            <h4>Nutrition Chart</h4>
            {% if data %}
            <table style="width:100%;">
                <tr>
                    <th>Calories</th>

```

```
<th>{{data[0]}}</th>
</tr>
<tr>
<th></th>
<th>Daily Value</th>
</tr>
<tr>
<th>Total Fat</th>
<th>{{data[1]}}mg</th>
</tr>
<tr>
<td>Saturated Fat</td>
<td>{{data[2]}}mg</td>
</tr>
<tr>
<td>Polyunsaturated Fat</td>
<td>{{data[3]}}mg</td>
</tr>
<tr>
<td>Monounsaturated Fat</td>
<td>{{data[4]}}mg</td>
</tr>
<tr>
<th>Cholesterol</th>
<th>{{data[5]}}mg</th>
</tr>
<tr>
<th>Sodium</th>
<th>{{data[6]}}mg</th>
</tr>
<tr>
<th>Potassium</th>
```

```
<th>{{data[7]}}mg</th>
</tr>
<tr>
<th>Sugar</th>
<th>{{data[8]}}mg</th>
</tr>
<tr>
<th>Protein</th>
<th>{{data[9]}}mg</th>
</tr>
<tr>
<th>Carbohydrates</th>
<th>{{data[10]}}mg</th>
</tr>
<tr>
<th>Vitamin A</th>
<th>{{data[11]}}mg</th>
</tr>
<tr>
<th>Vitamin C</th>
<th>{{data[12]}}mg</th>
</tr>
<tr>
<th>Vitamin D</th>
<th>{{data[13]}}mg</th>
</tr>
<tr>
<th>Vitamin B5</th>
<th>{{data[14]}}mg</th>
</tr>
<tr>
<th>Calcium</th>
<th>{{data[15]}}mg</th>
</tr>
</table>
```

```
{% endif %}
```



</div>

</div>

</div>

</div>

</div>

</body> </html>

## admin.css

```
body {  
  background-color: rgb(192, 171, 171);;  box-  
sizing: border-box;  
}  
  
.dash {  
  padding: 50px 0;  
}  
  
.header {  
  font-size: 1.2rem;  background-color: rgb(192, 171, 171);  padding: 50px;  
border: 1px solid black;  margin: 0 20px;  box-sizing: border-box;  
border-radius: 25px;  
}
```

```

.header >h4{
    text-align: right;
    font-family: Arial, Helvetica, sans-serif;
}

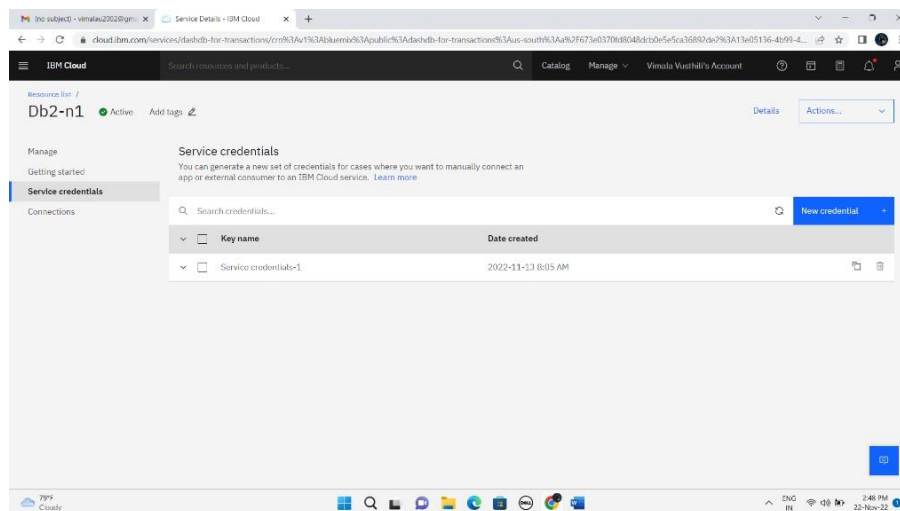
.combutton {
margin: 10px 30px;
padding: 10px 30px;
border-radius: 10px;
}

.btns {
    padding: 5px 30px;
}

.btns:hover {
    background-color: rgb(143, 131, 116);
}

```

## 7.3 Database Schema:



## 8 TESTING

**8.1 Test Cases:** With a concerted effort, I conducted research on general wellbeing to have a rudimentary grasp on health management, as well as the existing nutrient tracking apps in order to get an understanding of what is already existing in the market, the characteristics, specialties, and usability. There are a considerable number of nutrition tracking apps existing in the market. They aim to track daily calories/macros intake by logging meals to achieve users' preset goals. To log meals, users can input the food in the app, or scan the barcode of a package. Most apps allow users to connect with associated activities apps to track exercise progress. With a premium upgrade, users can get access to tailor-made recipes according to health goals or specified diets. In order to build a realistic initial target group, I wanted to conduct some usability tests with 5 users that regularly engage in physical activity and food tracking, including both first-time and regular users of meal planning and fitness apps. I asked these individuals to perform tasks related to general usage of the MyFitnessPal, Lifesum, and Nutrition Coach apps (such as food logging, searching, and checking their caloric breakdown.)

	Test Scenario
1	Verify if the user is able to open and view the homepage
2	Verify if the user is able to interact with the elements in the homepage
3	Verify if the user is able to navigate to the other pages of the application from the homepage
	Upload Image Page Actions
1	User is able to upload image
2	User is able to submit the image and obtain results
	View History of Items Related Actions
1	User is able to view all past uploaded images
2	User is able to see the nutritional breakdown of the previously uploaded images
	User is able to log in and sign up
1	User is able to create an account and log in

## **8.2 User Acceptance Testing:**

Must-have features of a diet and nutrition app I wanted to address the user pain points by including (and improving) the core features of the application. Personal profiles After downloading the app, a user needs to register and create an account. At this stage, users should fill in personal information like name, gender, age, height, weight, food preferences, allergies, and level of physical activity. Food logging and dashboard Allowing users to analyze their eating habits. They should be able to log food and water intake and see their progress on a dashboard that can track calories, fat, protein, and carbs. Push notifications Push notifications are an effective tool for increasing user engagement and retention.

To motivate users to keep moving toward their goals, it's pertinent to deliver information on their progress toward the current goal and remind them to log what they eat. Calorie counter Enabling the application to calculate calories users have burned and eaten based on the data they've logged. Barcode scanner Let users count calories and see accurate nutrition information via a built-in barcode scanner. Recipe book Users will appreciate the opportunity to find healthy recipes in the app. Including pictures, videos, and even voice instructions in your recipes would be a valuable feature. Also, allowing users to rate recipes and sort them by keywords, ingredients, categories, and calories.

Diet plans Help users maintain a healthy diet by offering diet plans. Usually, a diet plan includes meal suggestions, nutritional tips, recipes, and recommended total calorie intake per week or day. Gamification Including game elements to increase user engagement and retention. Using ranks, badges, and points to reward users for achievements such as losing weight or completing goals. Integration with wearables There are different trackers and wearables to integrate with. For example, Apple Watch, Android Wear, Jawbone, Fitbit, and Samsung Gear to synchronize data on physical activity and health metrics. Nice-to-have features of a diet and nutrition app Since nutrition apps can have different purposes, their functionality can differ accordingly. Below are features that I considered including later on or that could be useful for some nutrition apps.

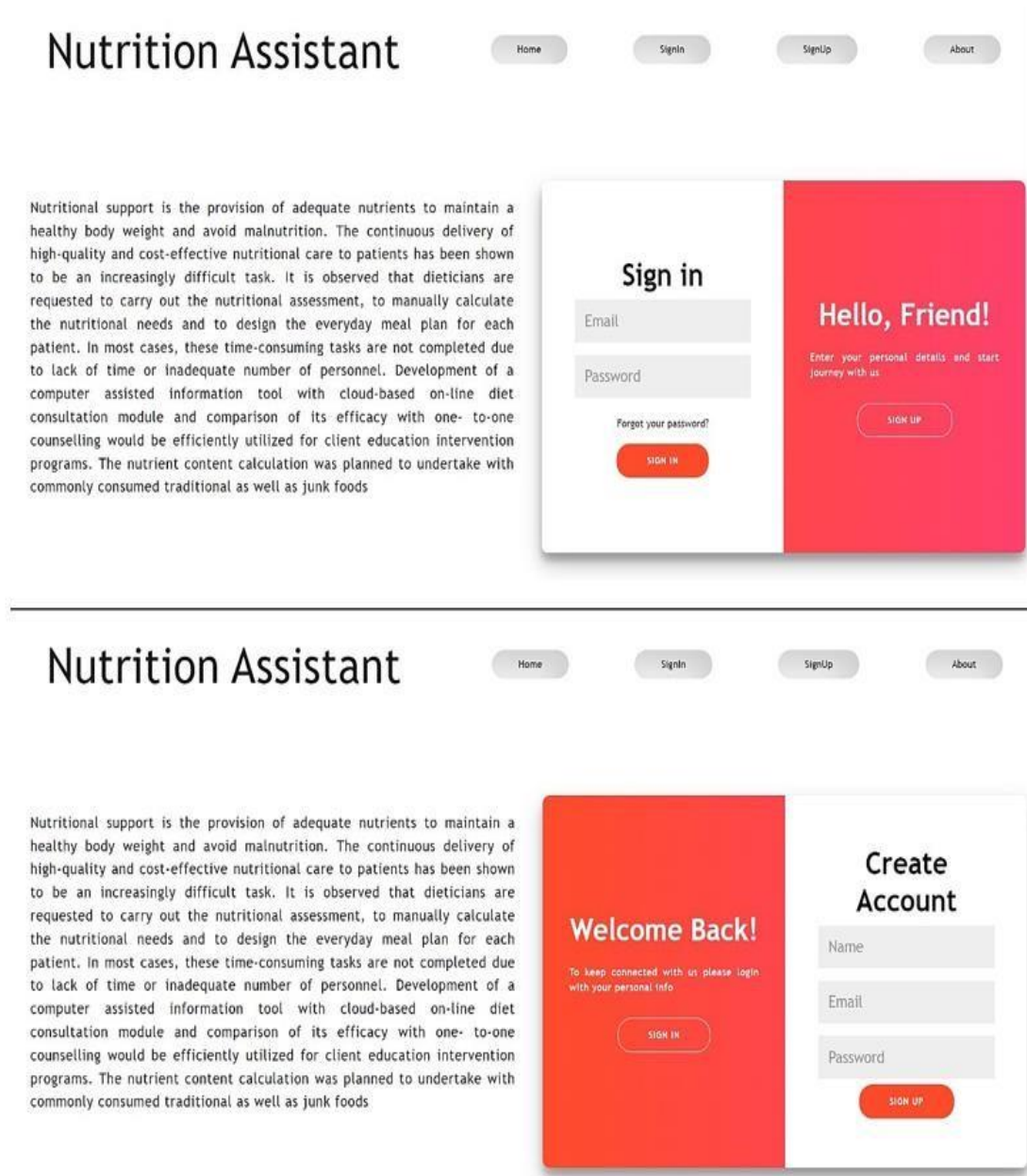
Blog A lot of users want a diet and nutrition application to not only count calories but also share some diet tips to help them improve eating habits. This is where blogs come in handy. There, the latest food and nutrition research, news, and health tips could be shared. Shopping list Letting users import ingredients from a diet plan or a recipe to a shopping list or add groceries manually. Experts Users will definitely appreciate being able to get in touch with diet coaches for expert advice. This could be a paid feature. User Personas From the interview and observation sessions, I collected and synthesized some information that can be used in the upcoming design process. I could clearly define 2 user profiles from the research data. A casual dieter who does not follow a health plan regularly: Enise is a full-time student who needs reminders, suggestions, and coaching to cook more often with fresh ingredients because they want to stay on top of their health and make it a part of their routine. dieter who does not follow a health plan regularly: Enise is a full-time student who needs reminders, suggestions,

Should alert the user for their daily plan	A alarm must be set off	Passed
Should create log of their diet plan	A text file should be created with previous diet plans	Passed
Should suggest new plans	Notification rolling for important messages	Passed
During the completion of successful diet plan a new thanking window should open	A new popup should display- saying u have completed your diet plans	Passed
Nutrition amount display	Total nutrients consumed must be displayed	Passed
Incomplete display	A alert and a popup display should be displayed	Passed
Logout display	Should successfully logoff the credentials	Passed

and coaching to cook more often with fresh ingredients because they want to stay on top of their health and make it a part of their routine.

# 9 RESULTS

## 9.1 Performance Metrics:

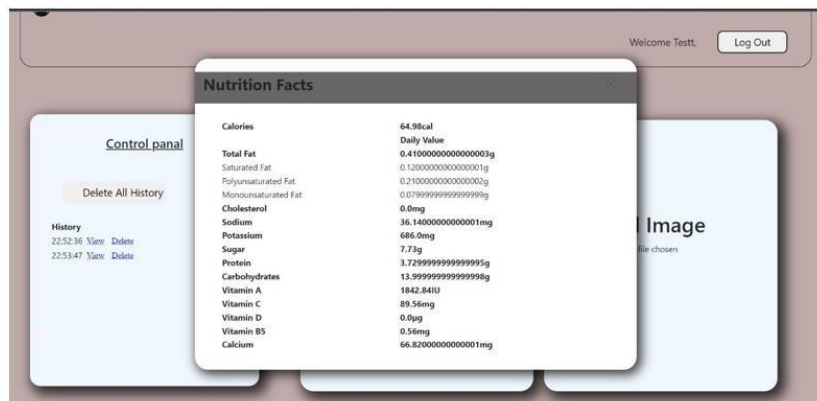
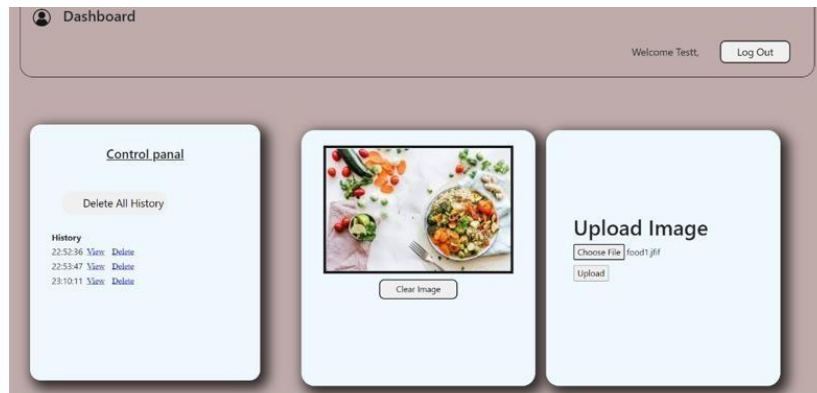


# Nutrition Assistant

[Home](#)[Signin](#)[Signup](#)[About](#)

Nutritional support is the provision of adequate nutrients to maintain a healthy body weight and avoid malnutrition. The continuous delivery of high-quality and cost-effective nutritional care to patients has been shown to be an increasingly difficult task. It is observed that dietitians are requested to carry out the nutritional assessment, to manually calculate the nutritional needs and to design the everyday meal plan for each patient. In most cases, these time-consuming tasks are not completed due to lack of time or inadequate number of personnel. Development of a computer assisted information tool with cloud-based on-line diet consultation module and comparison of its efficacy with one- to-one counselling would be efficiently utilized for client education intervention programs. The nutrient content calculation was planned to undertake with commonly consumed traditional as well as junk foods

A web based tool is being planned for therapeutic nutrition prescriptions in clinical settings. The cloud based system would have the ability to calculate the nutritional requirements and to guide first line nutritional management to patients and clients automatically.



## **10 ADVANTAGES & DISADVANTAGES**

### **Advantages:-**

1. This device is user friendly.
2. Its only required the image of the food.
3. To know the different type of nutrients present in food.
4. And also know that how much composition of the nutrients are present.
5. Output of the screen is easy understandable.
6. The user is now able to track his daily calorie intake.

### **Disadvantages:-**

1. This device is not able to predict the multiple image as input.
2. The internet is only necessary for opening the web application.(After converting the mobile app internet is not necessary for opening .)
3. It cannot be used without an Internet Connection.
4. Usage of 3rd party API may cause a time delay.

## **11 CONCLUSION**

During this assignment we were able to take a closer look at our daily eating habits. From here we can now improve our application so that we can help clients to eat and grow healthier as a person and athlete. I can truly say that I learnt a lot from this assignment. I was able to point out changes I needed to make and how to move forward and make it work in my life. I am now more educated on the powers of food and how they control our body. I hope that people will use our application to lead a healthy life.

This application allows people to get to know the nutrients of foods at any time which makes it more convenient for the users. This can be scaled to include APIs that have a larger variety of foods to have it cater to larger audiences of different backgrounds and ethnicities.



When choosing the right foods for yourself you should be focused on what is the healthiest choice. Eating healthy and feeling good go hand in hand, eating better will automatically give you a better functioning body.

## **12 FUTURE SCOPE**

### **1. ADDING GRAPHICAL DATA ON THE FOODS CONSUMES**

Adding a pie chart or a breakdown of what nutritional components are being consumed can give more insight into the food habits of a user. This can help the user to make changes and increase or decrease their consumption of a particular nutrient or

### **food. 2. CREATING A PERSONALIZED FOOD RECOMMENDATION SYSTEM**

Based on the previously uploaded images we can provide recommendations for the kinds of foods to eat to have a balanced diet.

The device will also assist you determine the quantity and degree of flavour of the food. Future goals include increasing the accuracy of our machine learning model and expanding the types of food categories so that we can better meet user needs. We are also increasing dataset of categories of images and nutrition to better efficiency to get output. Our research essentially identifies simply the nutrients, but our team members raise the bar for our project so that we also understand the ingredients and the amount of nutrients in a particular cuisine.

## **13 APPENDIX**

### **Source code:**

```
from flask import Flask, render_template, Response, request
import cv2

import datetime, time
import os, sys
import numpy as np

from threading import Thread
## csv code
import pandas as pd
```

```

read_file = pd.read_excel ("C:\\Users\\anish\\Desktop\\IBM2\\Book.xlsx")
read_file.to_csv ("Test.csv", index = None, header=True) df =
pd.DataFrame(pd.read_csv("Test.csv")) df.to_csv("Test.csv")
df=df.set_index("Food Name") def Nutrients(Name):
name=Name return(df.loc[(name),:])
##
global capture,rec_frame, grey, switch, neg, face, rec, out,p,d capture=0 grey=0
neg=0 face=0 switch=1 rec=0
# ML
import keras import cv2
import tensorflow as tf #import PIL.Image
#from tensorflow.keras.utils import to_categorical
#from tensorflow.keras.preprocessing.image import load_img, img_to_array from
keras_preprocessing.image import load_img,img_to_array
#from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
#from keras.preprocessing.image import ImageDataGenerator
#import tensorflow.compat.v2 as tf from keras.models import load_model model
=
keras.models.load_model('C:\\Users\\vimala\\Desktop\\IBM2\\Daiyan.h5')
import numpy as np
##
import numpy as np
CATEGORIES = ['Vegetable-Fruit', 'Egg', 'Bread', 'Soup', 'Seafood', 'Meat', 'vada
pav', 'Fried food', 'pizza', 'Dessert', 'Dairy product', 'Rice', 'burger',
'NoodlesPasta'] def image(path):
img = cv2.imread(path, cv2.IMREAD_GRAYSCALE) new_arr = cv2.resize(img, (60,
60))
new_arr = np.array(new_arr)
new_arr = new_arr.reshape(-1, 60, 60, 1) return new_arr

```

```

##

#make shots directory to save pics try: os.mkdir('./shots')
except OSError as error:

pass

#instantiate flask app

app = Flask( name , template_folder='./templates') camera
= cv2.VideoCapture(0) # def Path(d):

# a=d

# return a

def gen_frames(): # generate frame by frame from camera global out,
capture,rec_frame,d while True:

success, frame = camera.read() if success:

if(capture): capture=0

now = datetime.datetime.now()

p = os.path.sep.join(['shots', "shot_{}.png".format(str(now).replace(":", ""))])
#d=("C:\\Users\\anish\\Desktop\\IBM2\\"+p)

cv2.imwrite(p, frame) d=p try:

ret, buffer = cv2.imencode('.jpg', cv2.flip(frame,1)) frame = buffer.tobytes() yield
(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n') except
Exception as e:

pass else:

pass

@app.route('/') def index():

return render_template('index.html')

@app.route('/uplod') def uplod(): return

render_template('index.html')

@app.route('/video_feed') def video_feed():

```

```

return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame') @app.route('/requests',methods=['POST','GET']) def tasks():
global switch,camera if
request.method == 'POST': if
request.form.get('click') == 'Capture':
global capture capture=1
elif request.form.get('detect') == 'Detect':
# prediction =
model.predict([image("C:\\Users\\anish\\Desktop\\IBM2\\download.jfif")])
path = os.getcwd() print(d) p=os.path.join(path, "", d ) prediction =
model.predict([image(p)])
name=(CATEGORIES[prediction.argmax()]) Product_name=name
data=Nutrients(Product_name)
return render_template('Preduct.html',name=name,data=data) elif
request.form.get('stop') == 'Stop/Start': if(switch==1):
switch=0 camera.release() cv2.destroyAllWindows() else:
camera = cv2.VideoCapture(0) switch=1 elif request.method=='GET':
return render_template('index.html') return render_template('index.html')
if name == ' main ': app.run() camera.release() cv2.destroyAllWindows()

```

### **GitHub link:**

<https://github.com/IBM-EPBL/IBM-Project-53184-1661317626>

### **Project Demo link:**

<https://www.mediafire.com/file/kdbbf0r3uppdzww/DEMO+VIDEO.mp4/file>