PROJECT TITLE: NUTRITION ASSISTANT APPLICATION

DOMAIN: CLOUD APP DEVELOPMENT

A PROJECT REPORT

Submitted by:

Angelin Star J (950019106004)

Jeyasundari B (950019106015)

Josephin Derin E (950019106017)

Suba Lakshmi M (950019106042)

Team ID:PNT2022TMID49640

ANNA UNIVERSITY REGIONAL CAMPUS TIRUNELVELI

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

1.INTRODUCTION

1.1 Project Overview

This project aims at building a web App that automatically estimates food attributes such as ingredients and nutritional value by classifying the input image of food. Our method employs **Clarifai's AI-Driven Food Detection Model** for accurate food identification and Food API's to give the nutritional value of the identified food.

Work Flow of the Project:

- User should create an account using mail and receive a confirmation mail.
- User can calculate his BMI by providing height and weight.
- User interacts with the Web App to Load an image.
- The image is passed to the server application, which uses Clarifai's AI-Driven Food Detection Model Service to analyse the images and Nutrition API to provide nutritional information about the analysed Image.
- Nutritional information of the analysed image is returned to the app for display.

1.2 Purpose

Due to the ignorance of healthy food habits, obesity rates are increasing at an alarming speed, and this is reflective of the risks to people's health. People need to control their daily calorie intake by eating healthier foods, which is the most basic method to avoid obesity. However, although food packaging comes with nutrition (and calorie) labels, it's still not very convenient for people to refer to App-based nutrient dashboard systems which can analyse real-time images of a meal and analyse it for nutritional content which can be very handy and improves the dietary habits, and therefore, helps in maintaining a healthy lifestyle.

2.LITERATURE SURVEY

2.1 Existing problem

TITLE	YEAR	AUTHOR	OBJECTIVES	TECHNIQUES	RESULT	DISADVANTAGE
My fitnesspal	2005	Mike Lee, Albert Lee	Track weight and recommend calorie intake.	• Largest food database in a diet tracker. • Extensive recipe and exercise databases. • Syncs with fitness devices	This app helps user to log and count calories, track exercise, view weight loss progress, and connect with friends for support.	There was no significant difference between intervention and control groups in weight changes
Noom	2008	Artom Petakov, Saeju Jeong	Track number of calories consume by the user per day.	• Provides a weight loss plan based on a psychology-based evaluation. • No food or food type is off-limits. • Focuses on cresting lifestyle changes	Weight loss	No specific meal plans
Lose it!	2008	Charles Teaque	It helps to use food dairy and exercise log easily.	• Expert-verified food, restaurant, grocery store, and brand-name foods database. • Includes an active community feature. • Syncs with health apps.	Gives calories and nutritional information for whatever food logging by user and that goes toward daily total	Does not track micro nutritions

Life sum	2013	Henrik, MarcusGners	It provides a food and meal rating system that explains whether a food is nutritious and whether your meal is healthy or imbalanced.	• Includes educational content. • Provides food and meal ratings to encourage. healthier choices • Offers vegan, keto, paleo, and intermittent fasting meal plans, among others.	Building healthy habits	Food entries uploaded by users may be inaccurate

2.2 Reference link :

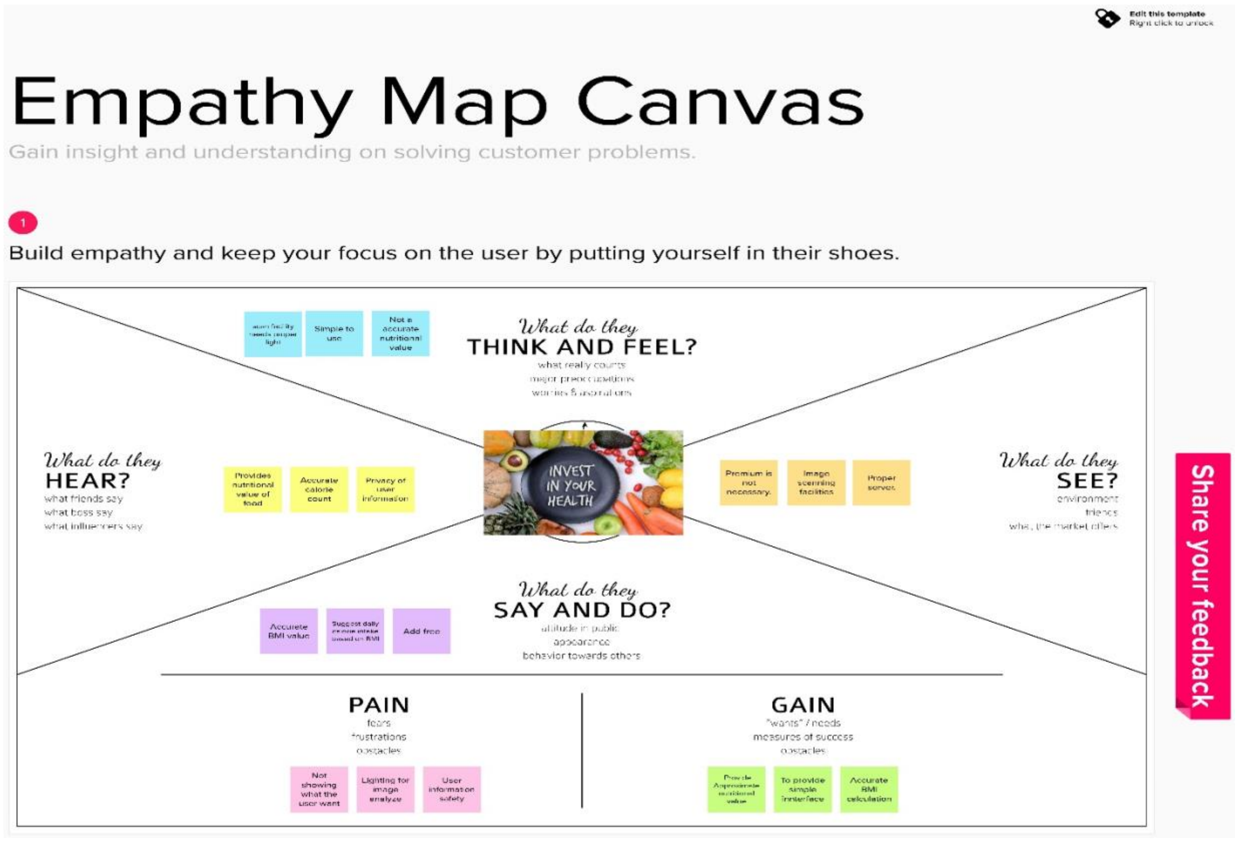
https://www.healthline.com/nutrition/5-best-calorie-counters

2.3 Problem Statement Definition

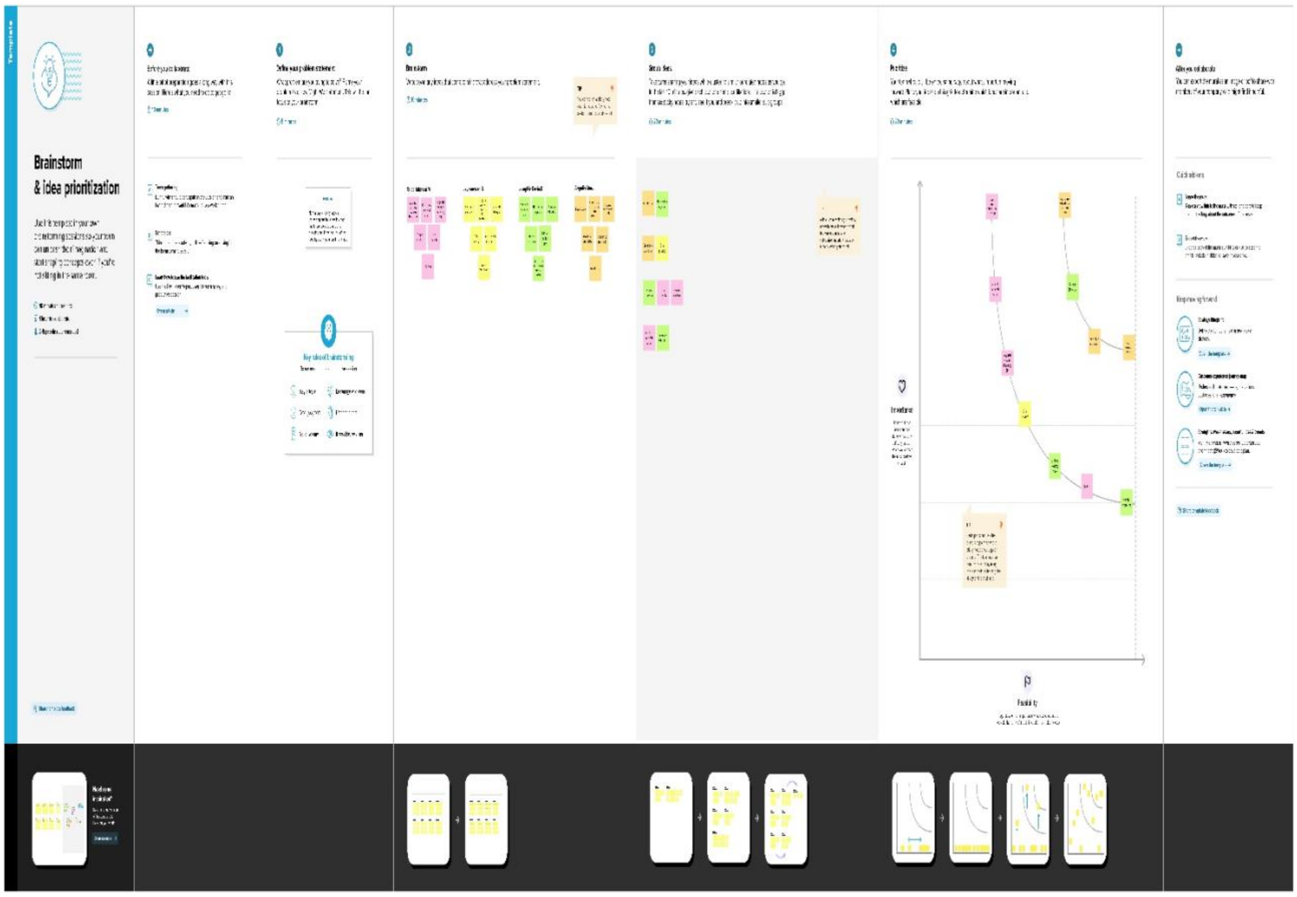
Now a days obesity rates are increasing due to unhealthy food habits and lack of exercise. Being obese has many disadvantages from lack of confidence to chronic diseases. We provide a app that analyses the food image and provide its nutritional value of the food.

3.IDEATION &PROPOSED SOLUTIONS

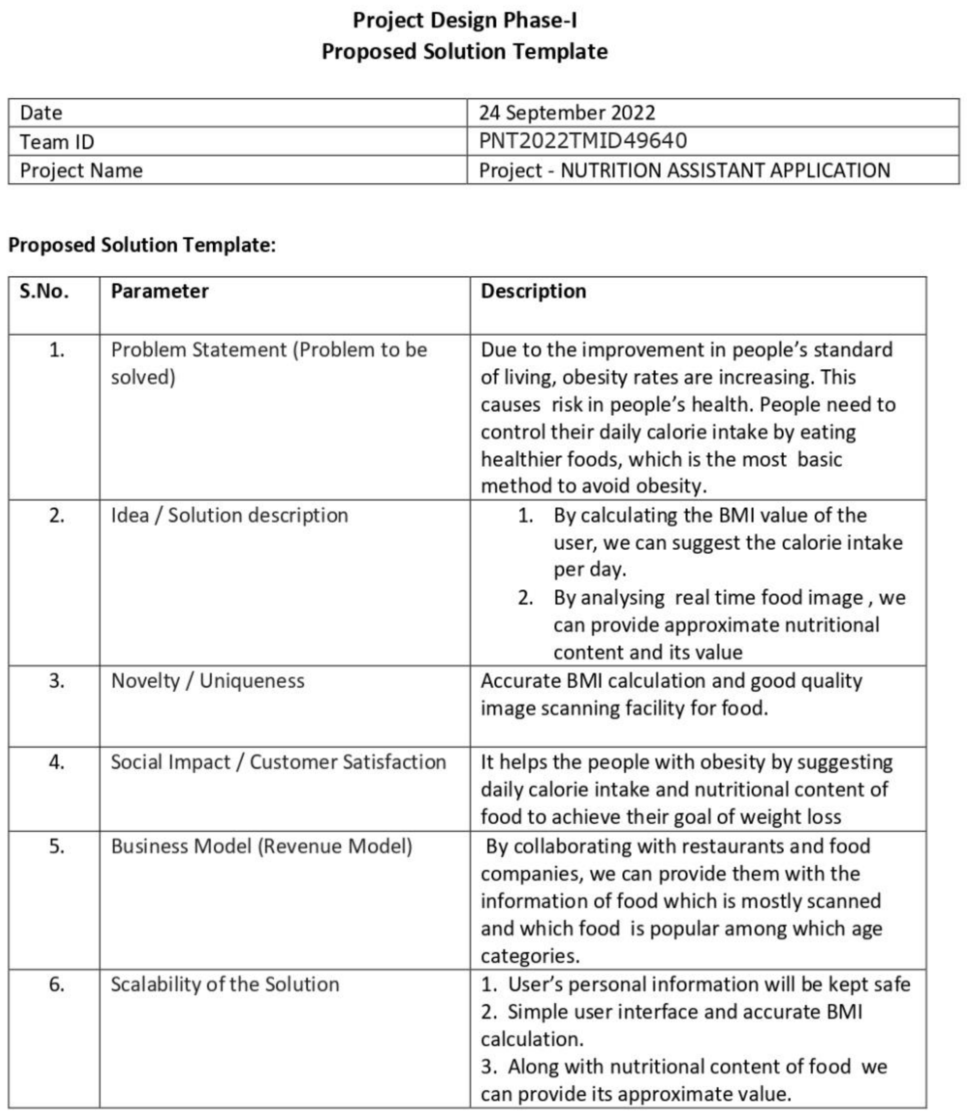
3.1 Empathy Map Canva



3.2 Ideation & Brainstorming



3.3 Proposed solution



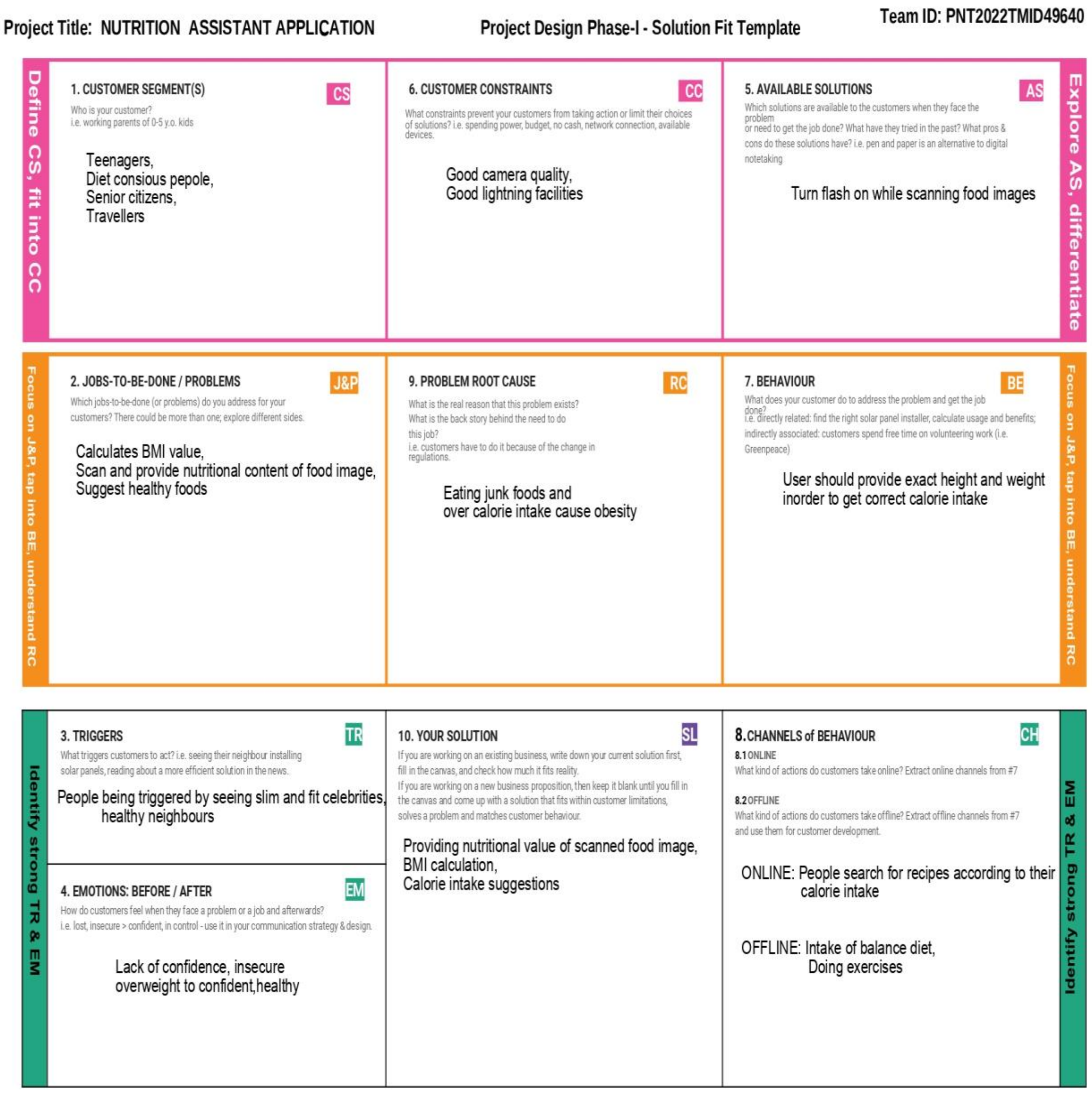
Project Design Phase-I
Proposed Solution Template

Date	24 September 2022
Team ID	PNT2022TMID49640
Project Name	Project - NUTRITION ASSISTANT APPLICATION

Proposed Solution Template:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Due to the improvement in people's standard of living, obesity rates are increasing. This causes risk in people's health. People need to control their daily calorie intake by eating healthier foods, which is the most basic method to avoid obesity.
2.	Idea / Solution description	1. By calculating the BMI value of the user, we can suggest the calorie intake per day. 2. By analysing real time food image , we can provide approximate nutritional content and its value
3.	Novelty / Uniqueness	Accurate BMI calculation and good quality image scanning facility for food.
4.	Social Impact / Customer Satisfaction	It helps the people with obesity by suggesting daily calorie intake and nutritional content of food to achieve their goal of weight loss
5.	Business Model (Revenue Model)	By collaborating with restaurants and food companies, we can provide them with the information of food which is mostly scanned and which food is popular among which age categories.
6.	Scalability of the Solution	1. User's personal information will be kept safe 2. Simple user interface and accurate BMI calculation. 3. Along with nutritional content of food we can provide its approximate value.

3.4 Problem Solution Fit



Project Title: NUTRITION ASSISTANT APPLICATION **Project Design Phase-I - Solution Fit Template** **Team ID: PNT2022TMID49640**

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Teenagers,
Diet consious pepole,
Senior citizens,
Travellers

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Good camera quality,
Good lightning facilities

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Turn flash on while scanning food images

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Calculates BMI value,
Scan and provide nutritional content of food image,
Suggest healthy foods

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Eating junk foods and
over calorie intake cause obesity

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

User should provide exact height and weight
inorder to get correct calorie intake

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

People being triggered by seeing slim and fit celebrities,
healthy neighbours

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Lack of confidence, insecure
overweight to confident,healthy

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Providing nutritional value of scanned food image,
BMI calculation,
Calorie intake suggestions

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

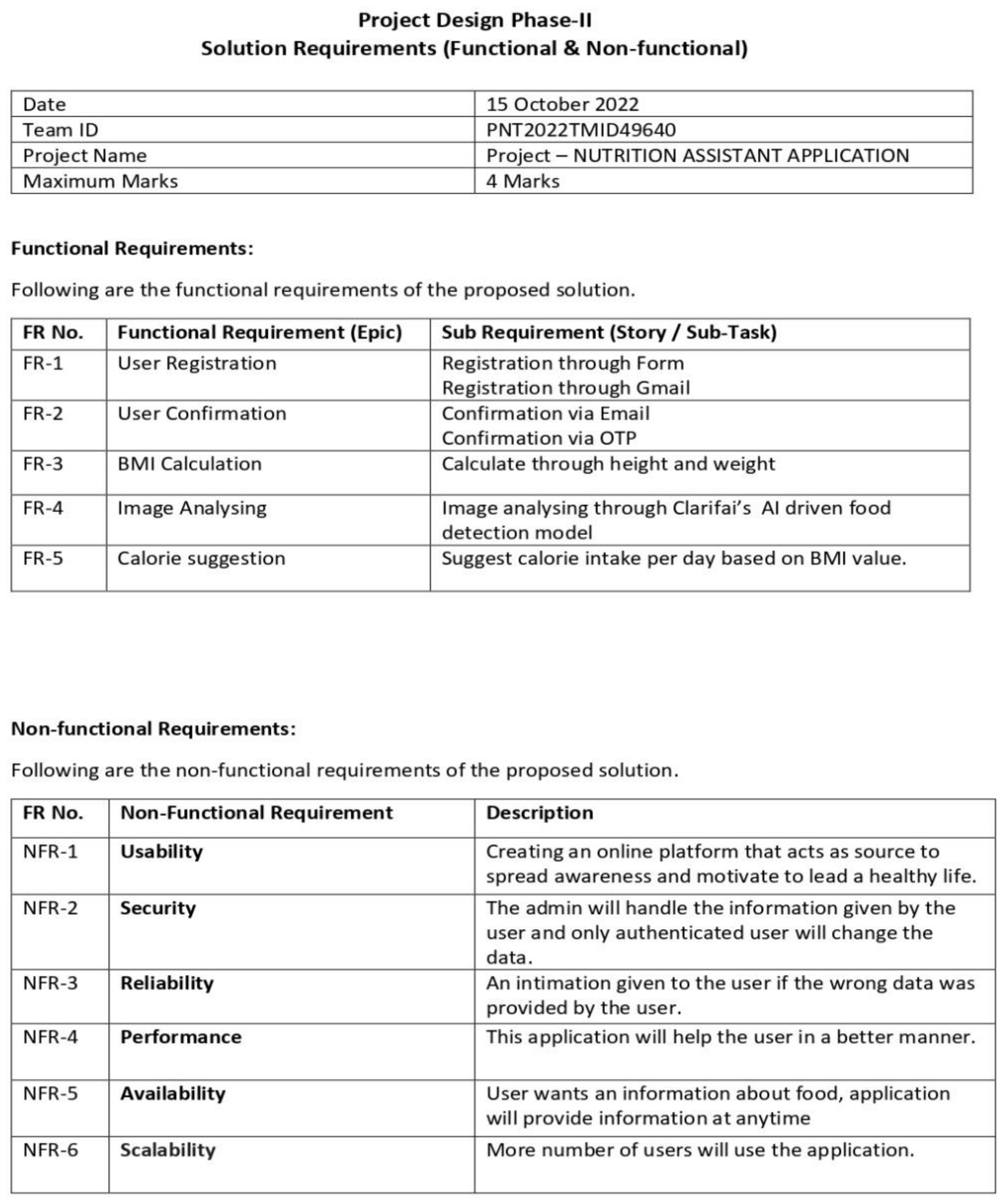
ONLINE: People search for recipes according to their
calorie intake

OFFLINE: Intake of balance diet,
Doing exercises

Identify strong TR & EM

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements & Non functional Requirements



Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	15 October 2022
Team ID	PNT2022TMID49640
Project Name	Project – NUTRITION ASSISTANT APPLICATION
Maximum Marks	4 Marks

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	BMI Calculation	Calculate through height and weight
FR-4	Image Analysing	Image analysing through Clarifai's AI driven food detection model
FR-5	Calorie suggestion	Suggest calorie intake per day based on BMI value.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Creating an online platform that acts as source to spread awareness and motivate to lead a healthy life.
NFR-2	**Security**	The admin will handle the information given by the user and only authenticated user will change the data.
NFR-3	**Reliability**	An intimation given to the user if the wrong data was provided by the user.
NFR-4	**Performance**	This application will help the user in a better manner.
NFR-5	**Availability**	User wants an information about food, application will provide information at anytime
NFR-6	**Scalability**	More number of users will use the application.

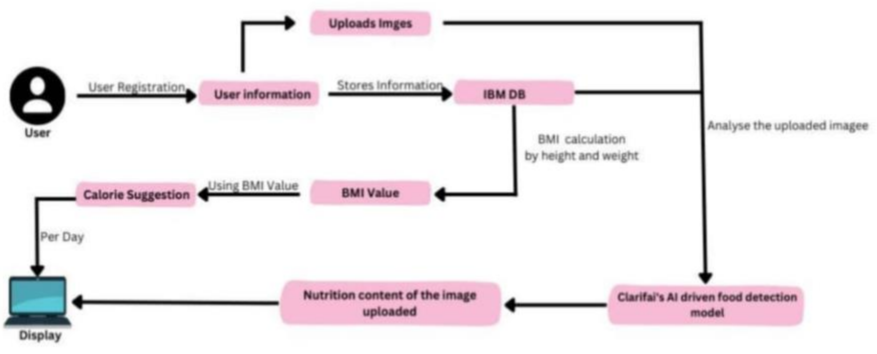
5.PROJECT DESIGN

5.1 Data Flow Diagram

Project Design Phase-II
Data Flow Diagram & User Stories

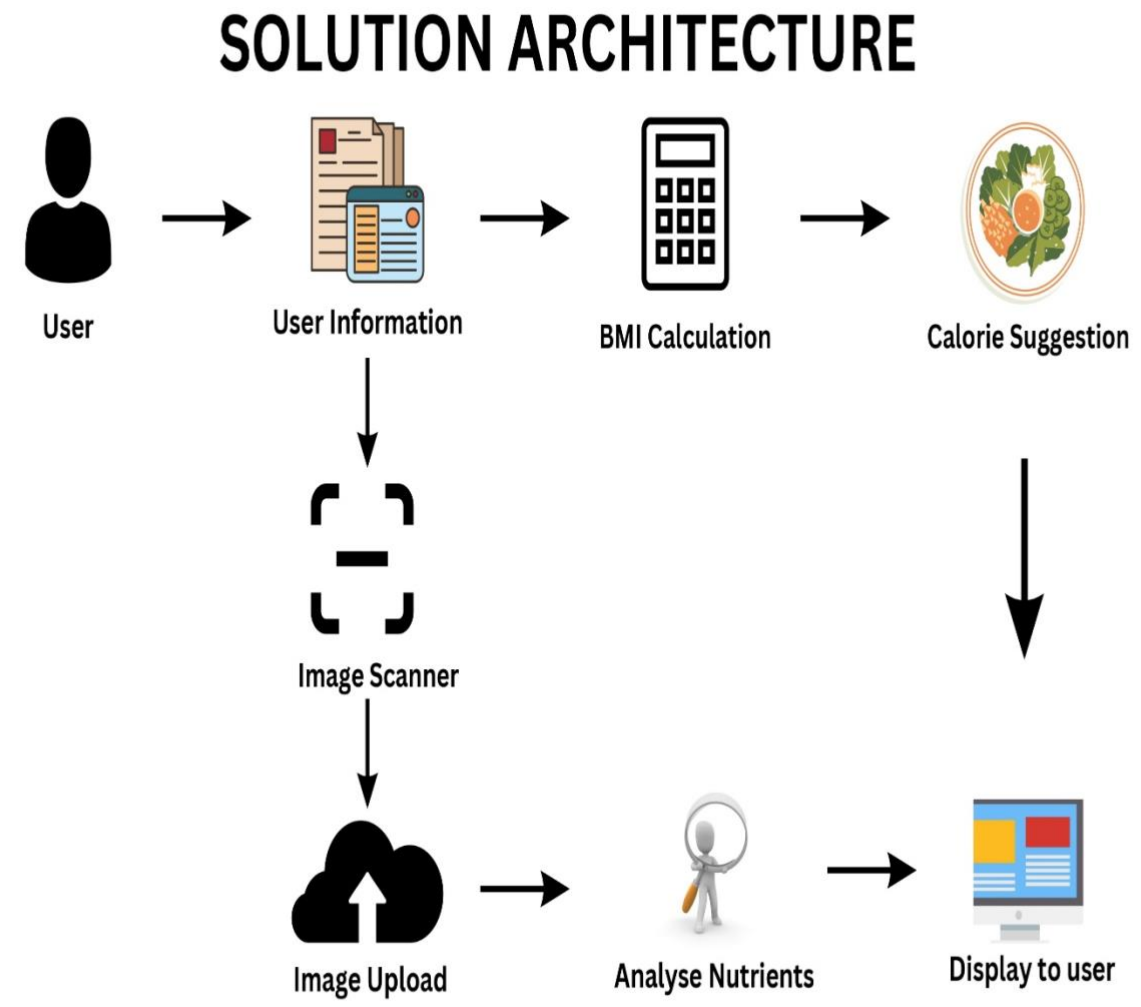
Date	16 October 2022
Team ID	PNT2022TMID49640
Project Name	Project – Nutrition Assistant Application
Maximum Marks	4 Marks

Data Flow Diagrams:

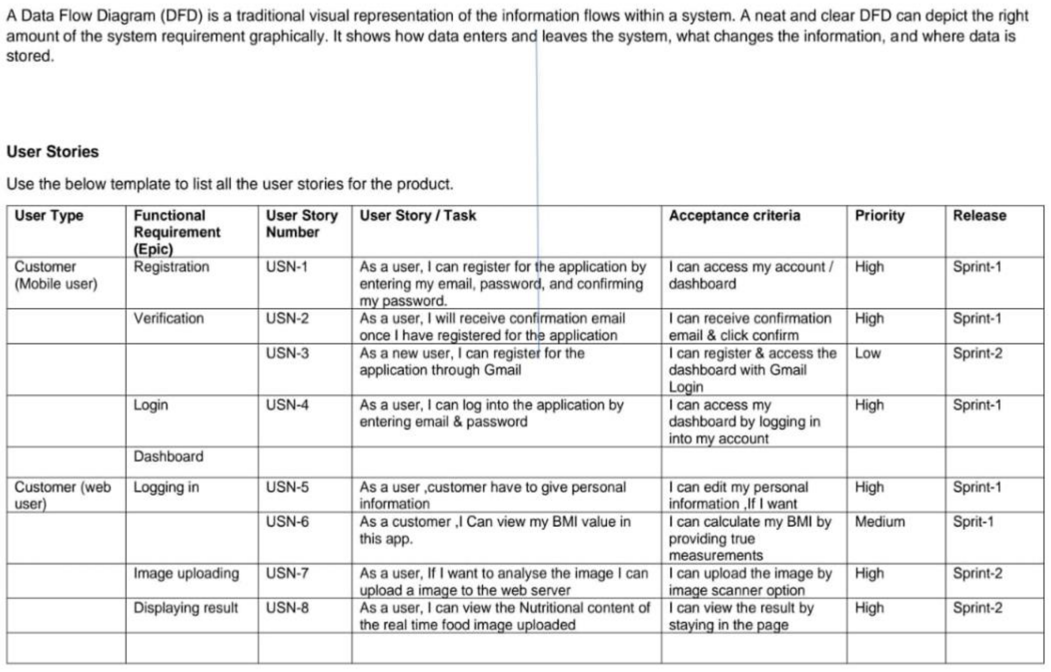


Uploads Imges
User
User Registration
User information
Stores Information
IBM DB
Analyse the uploaded imagee
BMI calculation by height and weight
Calorie Suggestion
Using BMI Value
BMI Value
Per Day
Display
Nutrition content of the image uploaded
Clarifai's AI driven food detection model

5.2 Solution & Technical Architecture



5.3 User Stories



A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

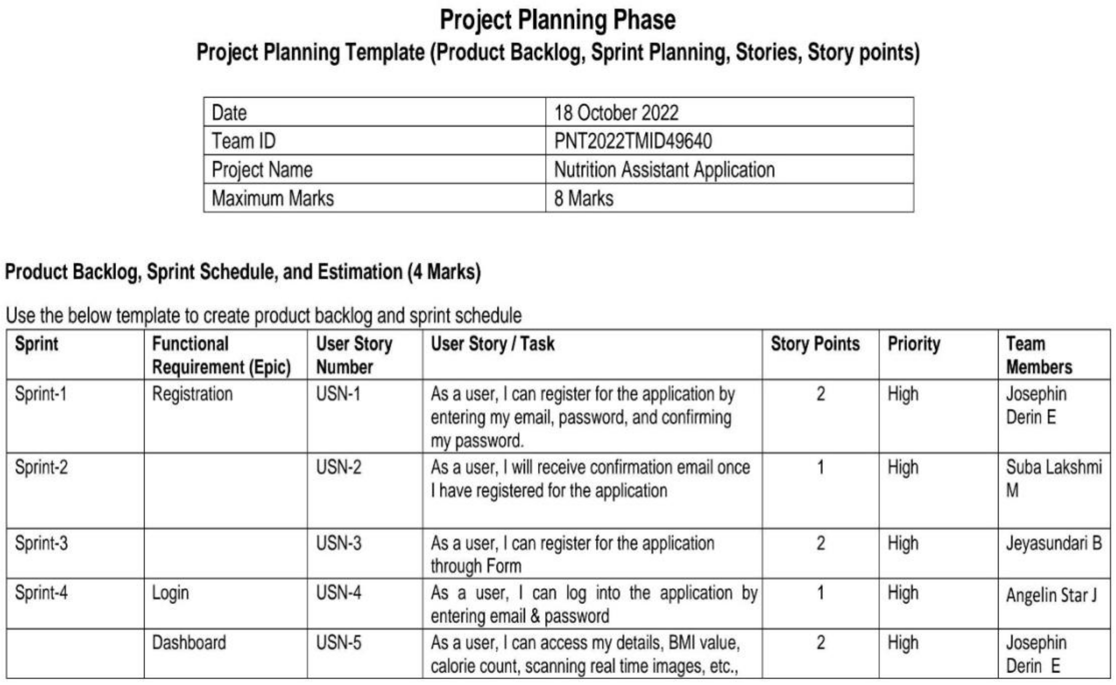
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Verification	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a new user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Low	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access my dashboard by logging in into my account	High	Sprint-1
	Dashboard					
Customer (web user)	Logging in	USN-5	As a user ,customer have to give personal information	I can edit my personal information ,If I want	High	Sprint-1
		USN-6	As a customer ,I Can view my BMI value in this app.	I can calculate my BMI by providing true measurements	Medium	Sprit-1
	Image uploading	USN-7	As a user, If I want to analyse the image I can upload a image to the web server	I can upload the image by image scanner option	High	Sprint-2
	Displaying result	USN-8	As a user, I can view the Nutritional content of the real time food image uploaded	I can view the result by staying in the page	High	Sprint-2

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation



Project Planning Phase

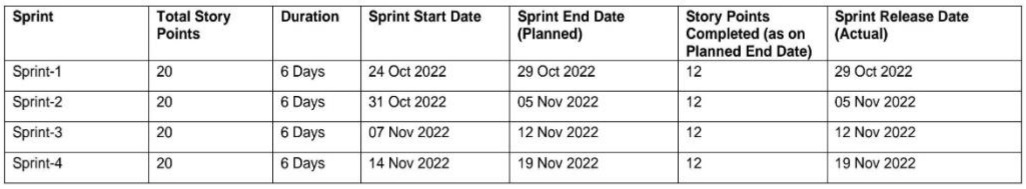
Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	18 October 2022
Team ID	PNT2022TMID49640
Project Name	Nutrition Assistant Application
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

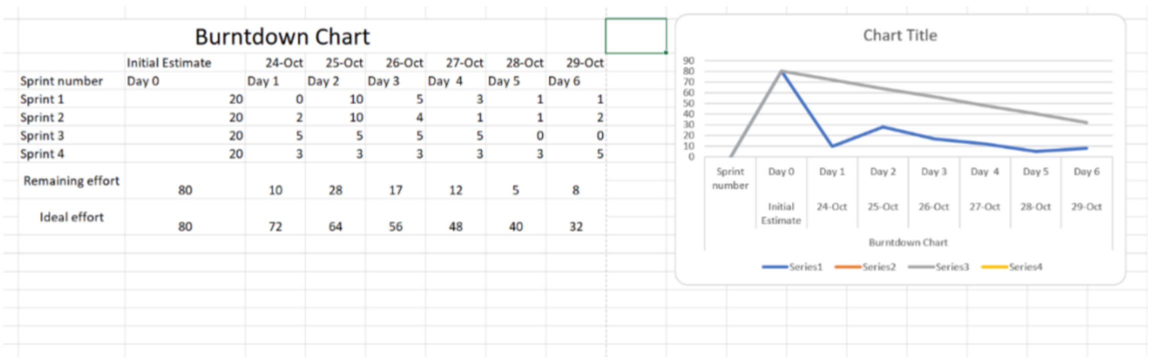
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Josephin Derin E
Sprint-2		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Suba Lakshmi M
Sprint-3		USN-3	As a user, I can register for the application through Form	2	High	Jeyasundari B
Sprint-4	Login	USN-4	As a user, I can log into the application by entering email & password	1	High	Angelin Star J
	Dashboard	USN-5	As a user, I can access my details, BMI value, calorie count, scanning real time images, etc.,	2	High	Josephin Derin E



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	12	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	12	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	12	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	12	19 Nov 2022

Burndown Chart

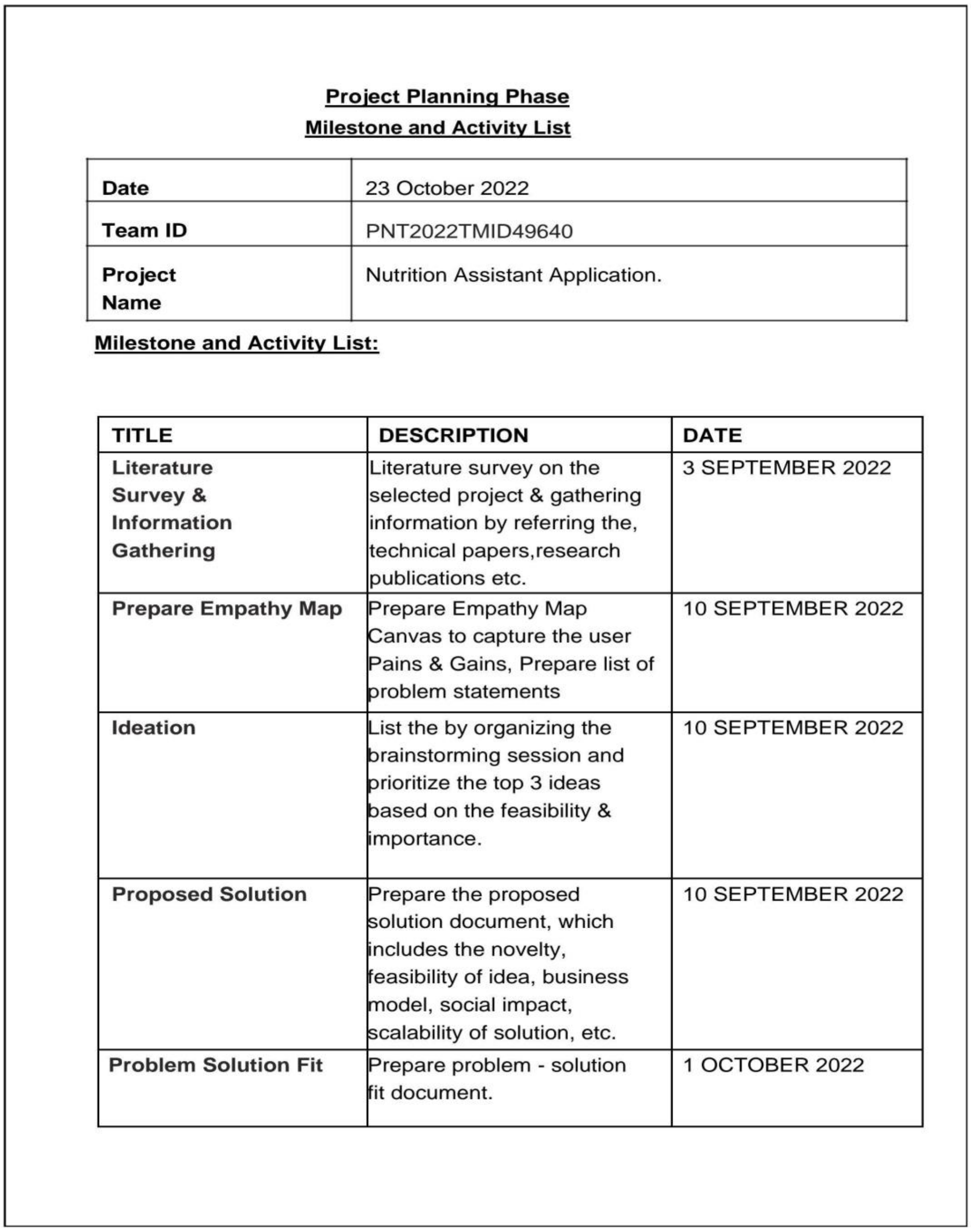
A burndown chart is a graphical representation of work left to do versus time. IT is often used in agile software develop.



Burntdown Chart

	Initial Estimate	24-Oct	25-Oct	26-Oct	27-Oct	28-Oct	29-Oct
Sprint number	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
Sprint 1	20	0	10	5	3	1	1
Sprint 2	20	2	10	4	1	1	2
Sprint 3	20	5	5	5	5	0	0
Sprint 4	20	3	3	3	3	3	5
Remaining effort	80	10	28	17	12	5	8
Ideal effort	80	72	64	56	48	40	32

6.2 Sprint Delivery Schedule



Project Planning Phase
Milestone and Activity List

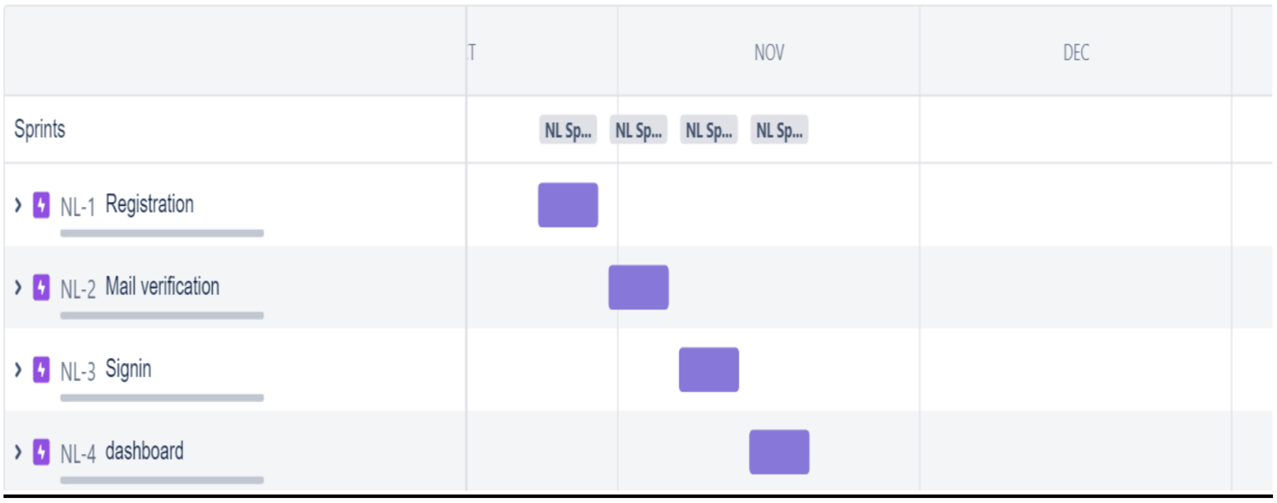
Date	23 October 2022
Team ID	PNT2022TMID49640
Project Name	Nutrition Assistant Application.

Milestone and Activity List:

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers,research publications etc.	3 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	10 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	10 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	10 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	1 OCTOBER 2022

Solution Architecture	Prepare solution architecture document.	19 OCTOBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application.	15 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	18 OCTOBER 2022
Technology Architecture	architecture diagram.	19 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	23 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	IN PROGRESS..

6.3 Reports from JIRA



7. Coding & Solutioning (Explain the features added in the project along with code)

7.1 Feature 1

Python Flask

We have used python flask to develop our project.

Python flask is a web frame work and python module that lets you develop web application easily.

Main.py

```
from flask import Flask, render_template, request, redirect, url_for, session

import ibm_db

from flask_mail import Mail, Message

import re

from werkzeug.utils import secure_filename

from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel

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

from clarifai_grpc.grpc.api.status import status_code_pb2
```

```
app = Flask(__name__)
mail = Mail(app) # instantiate the mail class


# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'derinjose886@gmail.com'
app.config['MAIL_PASSWORD'] = 'lhmjtfrjwblfbgeq'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ksm24043;PWD=ZXsdfH0rppz
tWofo",'','')
@app.route('/')
def home():
    return render_template('register.html')


@app.route('/login', methods =['GET', 'POST'])
def login():
    global userid
    msg = ''
    if request.method == 'POST'and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        stmt = ibm_db.prepare(conn,'SELECT * FROM accounts WHERE username = ?AND
password = ?')
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            session['loggedin'] = True
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'
            return render_template('index.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', a = msg)

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM accounts WHERE username = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        msgg = Message(
            'Hello',
            sender ='derinjose886@gmail.com',
            recipients = [email]
           )
        msgg.body = ' Welcome to NUTRI LITE!! Thanks for registering.  '
        mail.send(msgg)

        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Username must contain only characters and numbers !'
        elif not username or not password or not email:
            msg = 'Please fill out the form !'
        else:
            insert_sql = "INSERT INTO accounts (username,email,password) VALUES (?, ?, ?)"
            stmt = ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, password)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)

@app.route('/bmi', methods =['GET', 'POST'])
def bmi():
    if request.method == 'POST':
        height = request.form['height']
        weight= request.form['weight']
    return render_template('bmi.html')



@app.route('/img',  methods =['GET', 'POST'])
def img():
    print(request.form)
    return render_template('image.html')


@app.route('/food')
def food():
    return render_template('food.html')


@app.route("/dashboard", methods=["GET", "POST"])
def dashboard():
  global request
  if flask.request.method == "POST" and session['LoggedIn']:
    if 'file' not in flask.request.files:
      flash('No file part')
      return redirect(flask.request.url)
    file = flask.request.files['file']
    if file.filename == '':
      flash('No image selected')
      return redirect(flask.request.url)
    if file and allowed_file(file.filename):
      filename = secure_filename(file.filename)
      file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
      flash('Image successfully uploaded')
      with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), "rb") as f:
        file_bytes = f.read()

      request = service_pb2.PostModelOutputsRequest(
          model_id="food-item-v1-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=file_bytes
                )
              )
            )
          ],
      )
      response = stub.PostModelOutputs(request, metadata=metadata)

      if response.status.code != status_code_pb2.SUCCESS:
          print(response)
          raise Exception(f"Request failed, status code: {response.status}")

      foodname = response.outputs[0].data.concepts[0].name

      ingredients = ''
      for concept in response.outputs[0].data.concepts:
        ingredients += f"{concept.name}: {round(concept.value, 2)}, "

      nutritionValues = ''
    #headers = {'X-RapidAPI-Key':
"e90a2b1101msh8a9c2a55215e6b8p1b6838jsn26de2538dc24",
    #'X-RapidAPI-Host': "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"}
```

```
nutritions = {
  "recipesUsed": 10,
  "calories": {
    "value": 470,
    "unit": "calories",
    "confidenceRange95Percent": {
      "min": 408.93,
      "max": 582.22
    },
    "standardDeviation": 139.8
  },
  "fat": {
    "value": 17,
    "unit": "g",
    "confidenceRange95Percent": {
      "min": 12.81,
      "max": 21.36
    },
    "standardDeviation": 6.9
  },
  "protein": {
    "value": 15,
    "unit": "g",
    "confidenceRange95Percent": {
      "min": 9.06,
      "max": 29.78
    },
    "standardDeviation": 16.71
  },
  "carbs": {
          "value": 65,
          "unit": "g",
          "confidenceRange95Percent": {
            "min": 57.05,
            "max": 77.9
          },
          "standardDeviation": 16.81
        }
      }
      nutritions.pop('recipesUsed')
      for i in nutritions:
        nutritionValues += f"{i}: {nutritions[i]['value']} {nutritions[i]['unit']}, "



      sql = "INSERT INTO foods VALUES(?,?,?,?,?)"
      stmt=ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['userid'])
      ibm_db.bind_param(stmt, 2, datetime.datetime.now().strftime('%Y-%m-%d
%H:%M:%S'))
      ibm_db.bind_param(stmt, 3, foodname)
      ibm_db.bind_param(stmt, 4, ingredients)
      ibm_db.bind_param(stmt, 5, nutritionValues)
      ibm_db.execute(stmt)



      return render_template("dashboard.html",
        filename = filename,
        username = session['username'],
        foodname = foodname,
        ingredients = ingredients,
        nutritionValues = nutritionValues,
      )
    else:
      flash('Allowed image formats - png, jpg, jpeg')
      return redirect(flask.request.url)

if __name__ == '__main__':
 app.run(debug = True)
```

Mail Verification

When the new user registers, the confirmation mail is sent to the user’s mail.

```
from flask_mail import Mail, Message
mail = Mail(app) # instantiate the mail class
# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'derinjose886@gmail.com'
app.config['MAIL_PASSWORD'] = 'lhmjtfrjwblfbgeq'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM accounts WHERE username = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        msgg = Message(
            'Hello',
            sender ='derinjose886@gmail.com',
            recipients = [email]
           )
        msgg.body = ' Welcome to NUTRI LITE!! Thanks for registering.  '
        mail.send(msgg)
```

IBM Cloud DB2

When the new user registers into the application then the details of the user gets stored in IBM cloud DB2

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ksm24043;PWD=ZXsdfH0rppztWofo",",")
```

Registration page

The interested people can register in our website by filling the required information such as username, mail ID, password.

The registered user get verified through receiving mail.

The user can register in our website in order to get the nutrition value of the food image.

Register.html

```
<html>
 <head>
 <meta charset="UTF-8">
 <title> Register </title>
 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
 </head>
 <body style="background-image: linear-
gradient(120deg,#84e1ef,#ea7dbb);"></br></br></br></br></br>
 <div align="center">
 <div align="center" class="border">
 <div class="header">
 <h1 class="word">Register</h1>
 </div></br></br></br>
 <h2 class="word">
 <form action="{{ url_for('register') }}" method="post">
 <div class="msg">{{ msg }}</div>
 <input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/></br></br>
 <input id="email" name="email" type="email" placeholder="Enter Your
Email" class="textbox"/></br></br>
 <input id="password" name="password" type="text" placeholder="Enter Your Password"
class="textbox"/></br></br>
 <input type="submit" class="btn" value="Sign Up"></br>
 </form>
 </h2>
 <p class="bottom">Already have an account? <a class="bottom"
href="{{url_for('login')}}"> Sign In here</a></p>
 </div>
 </div>
 </body>
</html>
```

Login Page

The user can login our website by giving user name and password.

Login.html

```
<html>
```

```
 <head>
 <meta charset="UTF-8">
 <title> Login </title>
 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
 </head>
 <body style="background-image: linear-
gradient(120deg,#84e1ef,#ea7dbb);"></br></br></br></br></br>
 <div align="center">
 <div align="center" class="border">
 <div class="header">
 <h1 class="word">Login</h1>
 </div></br></br></br>
 <h2 class="word">
 <form action="{{ url_for('login') }}" method="post">
 <div class="msg">{{ msg }}</div>
 <input id="username" name="username" type="text" placeholder="Enter Your
Username" class="textbox"/></br></br>
 <input id="password" name="password" type="password" placeholder="Enter Your
Password" class="textbox"/></br></br></br>
 <input type="submit" class="btn" value="Sign In"></br></br>
 </form>
 </h2>
 <p class="bottom">Don't have an account? <a class="bottom"
href="{{url_for('register')}}"> Sign Up here</a></p>
 </div>
 </div>
 </body>
</html>
```

Index Page

User can get greetings from our website.

In index page BMI and Image icons are included.

User can also logout through the logout icon in index page.

Index.html

```
<html>
 <head>
 <meta charset="UTF-8">
 <title> Index </title>
 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
 </head>
 <body  style="background-image: linear-gradient(120deg,#84e1ef,#ea7dbb);"
></br></br></br></br></br>
    <div class="container">
    <div class="navigation-container">
    <!-- <div class="nav-BMI-button">BMI</div> -->
    <a href="{{url_for('bmi')}}" class="nav-BMI-button" >BMI</a>
</div>
<style>

    .container{
        text-decoration: none;
        display: flex;
        flex-direction: row;
        justify-content: flex-end;
    }
    .navigation-container{
        justify-content: flex-end;
        display: flex;
        align-items: center;
        flex-direction: row;
        gap: 15px;
        margin-right: 15px;
    }
    .nav-BMI-button{
        background-color: blue;
        padding: 5px;
        color: white;
        border-radius: 5px;
        border: 1px solid blue;
        cursor: pointer;

    }
    .nav-BMI-button:hover{
        background-color: transparent;
        color: blue;
        border: 1px solid blue;
    }
</style>
<div class="navigation">
    <a href="{{url_for('img')}}" class="nav-Image-button">Image</a>
</div>
<style>
    .navigation{
        justify-content: flex-end;
        display: flex;
        align-items: center;
        flex-direction: row;
        gap: 15px;
        margin-right: 15px;
    }
    .nav-Image-button{
        background-color: blue;
        padding: 5px;
        color: white;
        border-radius: 5px;
        border: 1px solid blue;
        cursor: pointer;

    }
    .nav-Image-button:hover{
        background-color: transparent;
        color: blue;
        border: 1px solid blue;
    }
</style>
</div>
 <div align="center">
 <div align="center" class="border">
 <div class="header">
 <h1 class="word">Index</h1>
 </div></br></br></br>
 <h1 class="bottom">
 Hi {{session.username}}!!</br></br> Welcome to the index page...
 </h1></br></br></br>
 <a href="{{ url_for('logout') }}" class="btn">Logout</a>
 </div>
 </div>
 <script src="main.js"></script>
 </body>
</html>
```

7.2. Feature 2

BMI Page

BMI calculator available in the BMI page calculates the BMI of the user by getting the height and weight information from user and gives the health status.

BMI.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>Document</title>
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
href="https://fonts.googleapis.com/css2?family=Old+Standard+TT:wght@700&display=swa
p" rel="stylesheet">
  <div class="container">



  <style>
    .container{
      margin-left: 500px;
      margin-top: 150px;
    }
    .bmi_container {
      display: flex;
      flex-direction: column;
      width: 300px;
      gap: 8px;
```

```
        }



    </style>
    </div>
</head>

<body style="background-image: linear-gradient(120deg,#84e1ef,#ea7dbb);">
    <div class="container">
        <h1>BMI Calculator</h1>
        <div class="bmi_container">
            <input id="height" type="number" placeholder="Enter Your Height in
Centimeters:">
            <input id="weight" type="number" placeholder="Enter Your Weight in Kilograms:
">

            <button type="submit" onclick="Calculate()">Calculate BMI</button>
        </div>

        <div class="bmi value">
            <h4>BMI Value: </h4>
            <div id="bmioutput"></div>
        </div>
        <div class="status">
            <h4>Status: </h4>
            <div id="bmistatus"></div>
        </div>

    </div>
     <script>
        const bmioutput = document.getElementById('bmioutput')
        const bmistatus = document.getElementById('bmistatus')

        function Calculate() {
            var height = document.getElementById("height").value;
            var weight = document.getElementById("weight").value;

            var result = parseFloat(weight) / (parseFloat(height) / 100) ** 2;

            if (!isNaN(result)) {
                bmioutput.innerHTML = result;
                if (result < 18.5) {
                    bmistatus.innerHTML = "Underweight,take more calories";
                }
                else if (result < 25) {
                    bmistatus.innerHTML = "Healthy,take nutrition rich food";
                }
                else if (result < 30) {
                    bmistatus.innerHTML = "Overweight,reduce calorie intake and do more
exercise";
                }
                else {
                    bmistatus.innerHTML = "Obesity,avoid junk food and do more exercise ";
                }
            }
        }
    </script>

</body>

</html>
```

Image Page

The user can upload the food image by clicking the choose file option.

The food analyze icon is also available in this page.

Image.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <body style="background-image: linear-gradient(120deg,#84e1ef,#ea7dbb);">
  <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 rel="stylesheet" href="static/styles.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
  <title>Nutri Lite</title>
</head>

    <div class="row align-items-md-stretch">
      <div class="col-md-6 my-3">
        <div class="h-100 p-5 text-bg-dark rounded-3">
          <h2>Upload food image</h2>
          <form action = "/dashboard" method = "POST" enctype="multipart/form-data">
            <input id="image" class="my-3 form-control" type="file" name="file" required/>
            <a href="{{url_for('food')}}"><center>ANALYZE </center></a>



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


  </div>



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

Food Details

After analyzing the uploaded food image it gives the user the food details such as food name, ingredients and the available nutrients.

Food.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 rel="stylesheet" href="static/styles.css">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">
  <title>NUTRI LITE</title>
</head>
```

```
<body style="background-image: linear-
gradient(120deg,#84e1ef,#ea7dbb);",background="bg.png">>

<h1><center>FOOD DETAILS</center></h1>

<h3><center><b>FOOD NAME</b>: </center></h3>

<h3><li><b>INGREDIENTS: </b></li></h3>

<h4><li><ol></ol></li></center></h4>

<h4><li><ol></ol></li></h4>

<h4><li><ol></ol></li></h4>

<h4><li><ol></ol></li></h4>

<h4><li><ol></ol></li></h4>

<h4><li><ol></ol></li></h4>

<h4><li><ol></ol></li></h4>

<h4><li><b>NUTRITION VALUE: </b></li></h4>

<h3><li><ol>CALORIES : </ol></li> </h3>

<h4><li><ol>CALORIES FROM FAT : </ol></li> </h4>

<h4><li><ol>CHOLESTEROL 54mg : </ol></li> </h4>

<h4><li><ol>TOTAL CARBOHYDRATES 18g : </ol></li> </h4>

<h4><li><b>PROTEIN 15g : </b></li> </h4>

<h4><li><ol>VITAMIN A : </ol></li></h4>

<h4><li><ol>VITAMIN C : </ol></li></h4>

<h4><li><ol>CALCIUM : </ol></li></h4>

<h4><li><ol>IRON : </ol></li></h4>
```

8.Testing

8.1. Test cases

Testcase ID	Feature Type	Component	Test scenario	Prerequisite	Steps to execute	Test data	Expected Result	Actual Result	Status	Comments	TC for automation (Y/N)	BUG ID	Executed by

Login page_ TC_0 01	Functional	Register page	User can sign up in register page and redirected to sign in page	Login page	1.Enter user name. 2.Enter password. 3.User is redirected to index page	Login.html	Sign in/sign up button will display.	Working as expected	Pass				
Login page_ TC_0 02	UI	Register page	Verify the UI elements in sign in.	Sign in page	1.Enter user name. 2.Enter password. 3.User is redirected to index page 4.Verify the registration with below UI element a) user	Register.html	Application should show below UI elements a) user name text box b)password text box. c)sign in button. d)don’t have an	Working as expected	Pass				

					name text box b)password text box. c)sign in button. d)don’t have an account.		account.						

8.2. User Acceptance Testing

1.Purpose of document

The purpose of this document is to briefly explain the test coverage and open issues of the project at the time of the release to user acceptance testing(UAT).

2.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level and how they resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub Total
By design	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won’t fix	0	5	0	1	8
Totals	22	14	11	22	51

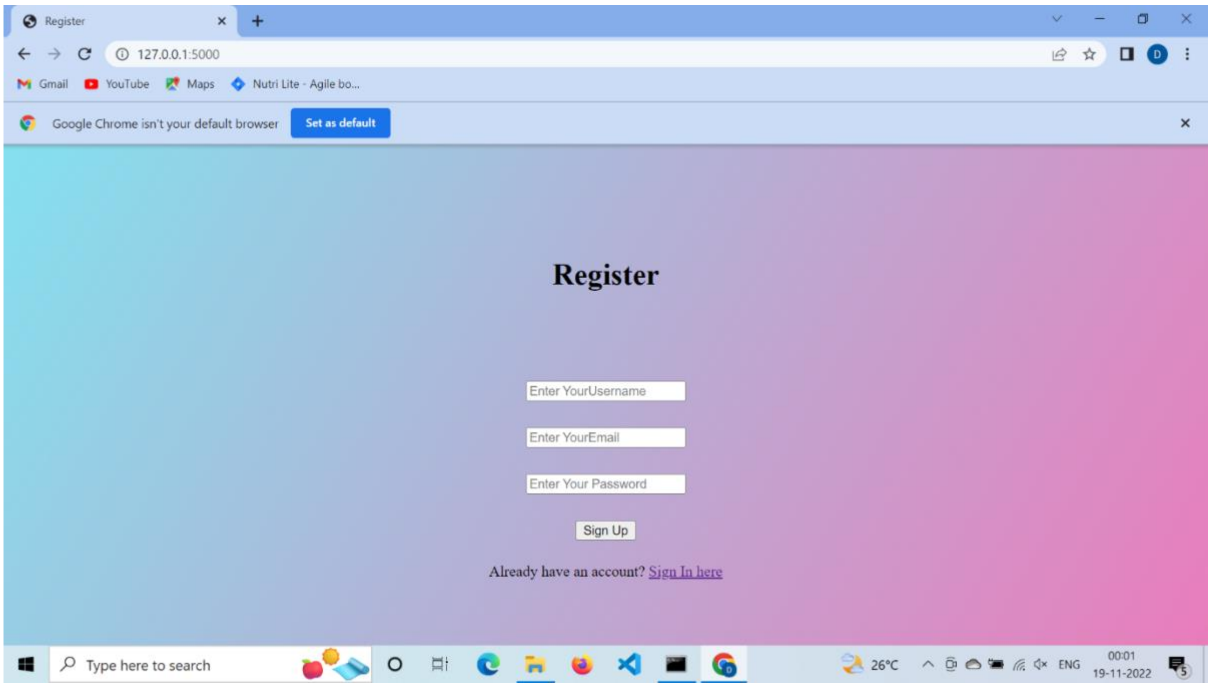
3.Test Case Analysis

This report shows the number of test cases that have passed, failed and untested.

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	43	0	0	43
Logout	2	0	0	2
Limit	3	0	0	3
Sign up	8	0	0	8
Final Report Output	4	0	0	4

9.Results

9.1. Performance Metrics



Register

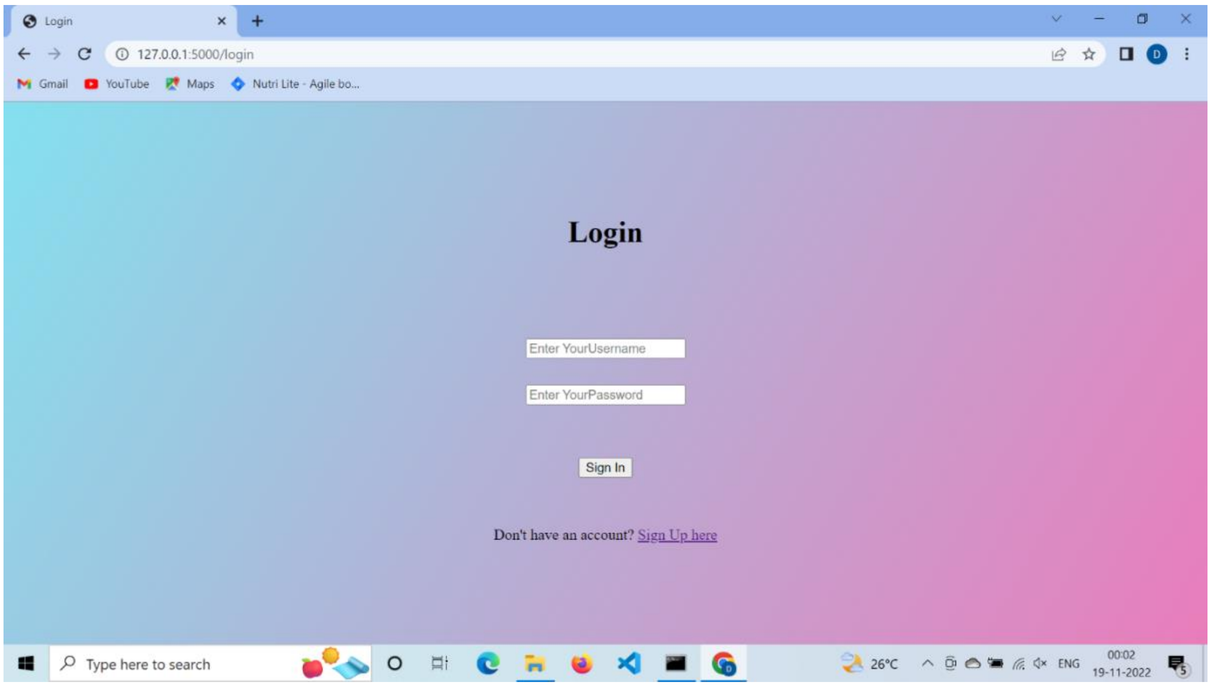
Enter YourUsername

Enter YourEmail

Enter Your Password

Sign Up

Already have an account? Sign In here



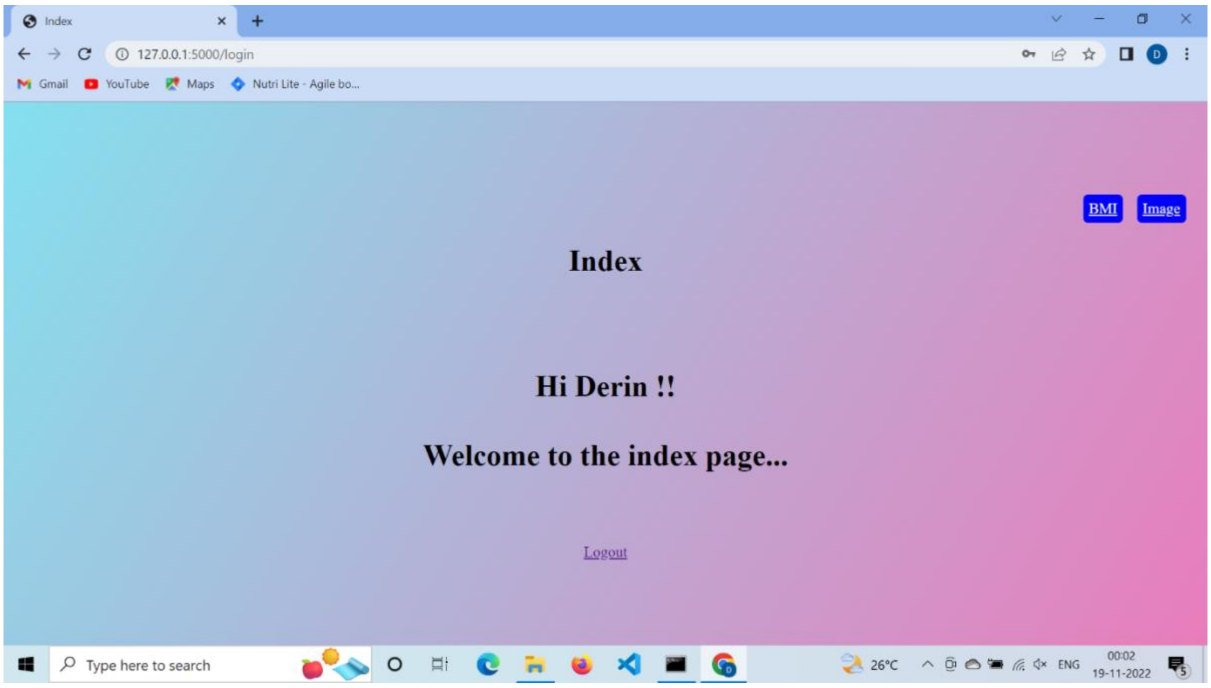
Login

Enter YourUsername

Enter YourPassword

Sign In

Don't have an account? Sign Up here

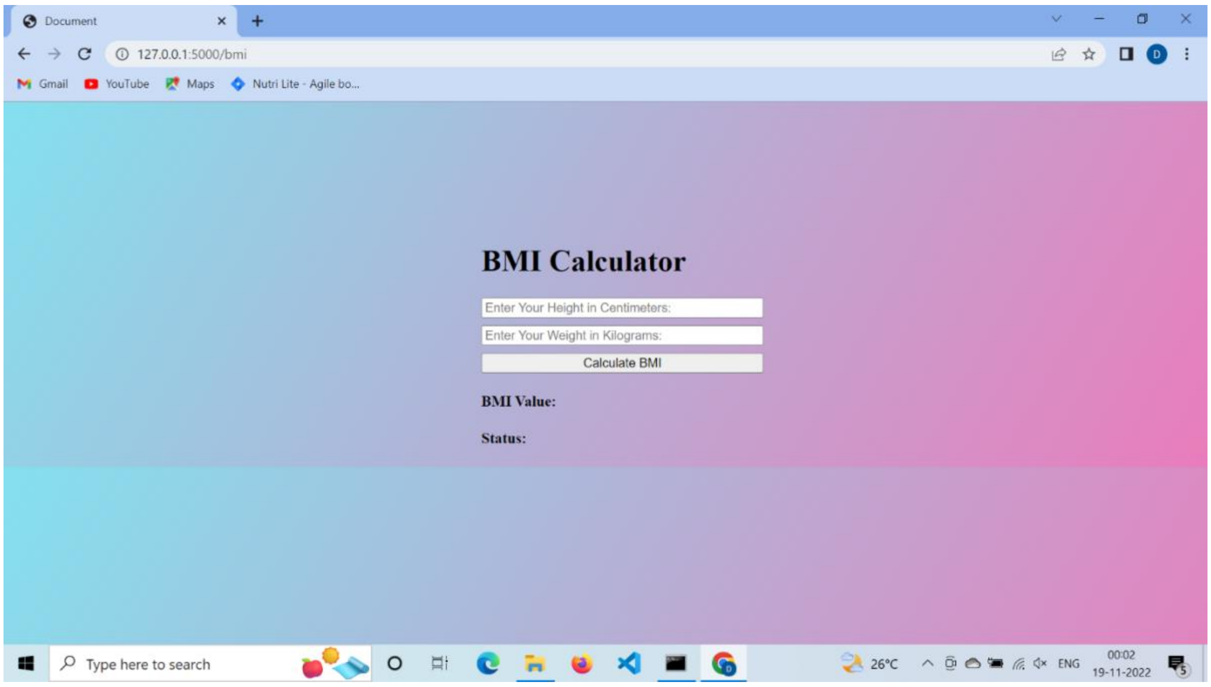


Index

Hi Derin !!

Welcome to the index page...

Logout



BMI Calculator

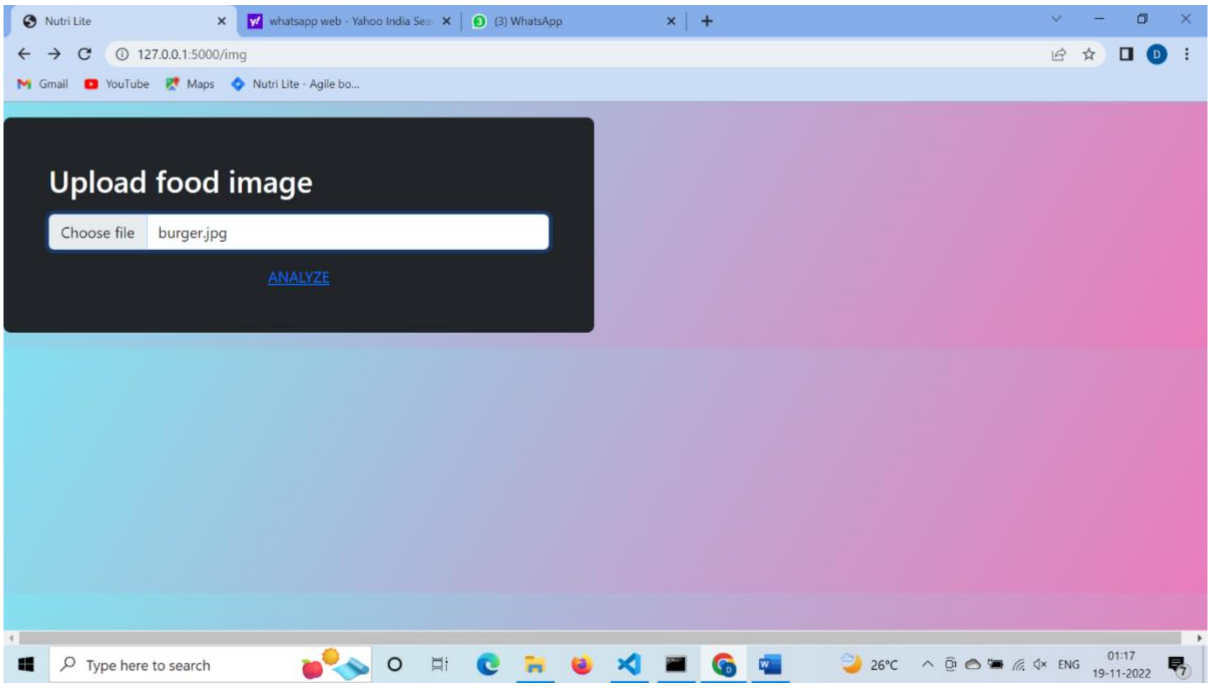
Enter Your Height in Centimeters:

Enter Your Weight in Kilograms:

Calculate BMI

BMI Value:

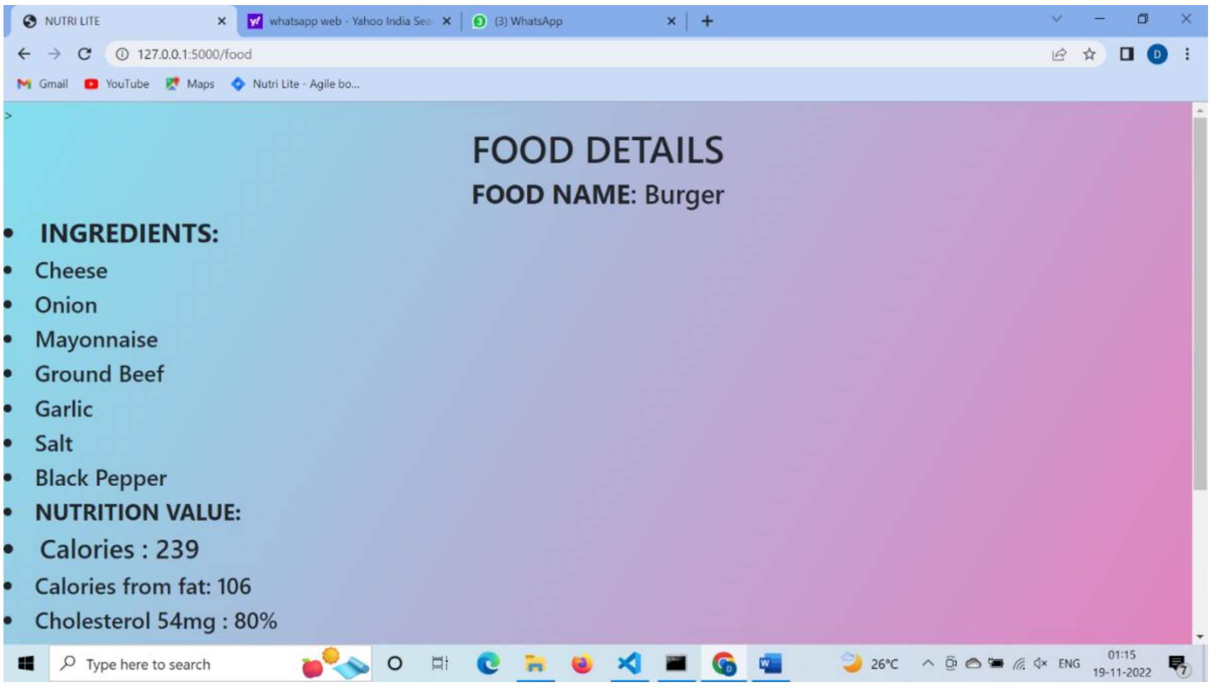
Status:



Upload food image

Choose file burger.jpg

ANALYZE



FOOD DETAILS

FOOD NAME: Burger

- **INGREDIENTS:**
- Cheese
- Onion
- Mayonnaise
- Ground Beef
- Garlic
- Salt
- Black Pepper
- **NUTRITION VALUE:**
- Calories : 239
- Calories from fat: 106
- Cholesterol 54mg : 80%

10. Advantages & Disadvantages

The advantages of Nutrition Assistant Application are as follows:

- It provides a maintained strategy of healthy eating habits.
- It delivers information on the nutritional value of foods and how balance and healthy eating habits are important for us.
- It limits the amount of unnecessary foods which contains fat that people consumes a lot.
- Increase health literacy.
- User information are highly secured.
- It provides the accurate BMI value.
- Easy to use the application.

The disadvantages of Nutrition Assistant Application are as follows:

- Sometimes it makes a level of disbalance in the balanced diet of an individual.
- It can improve the level of nutrition among individuals but delivers an inappropriate means of nutritional labeling.
- Sometimes it is considered as one of the major factors of weight gain.
- Sometimes the provided nutritional value of uploaded food image are not accurate.

11.Conclusion

In this study, we conducted a critical review of mobile apps from popular app stores. Our search results identified a total of 473 related apps, from which we selected and evaluated 80 apps using our modified app rating tool. We devised this app rating tool specifically for analyzing food consumption tracking and recommendation apps by adopting and extending existing mobile app rating scales. Using this rating tool we evaluated the selected 80 apps and analyzed and identified their design faults. According to our evaluation most of the existing mobile apps in the app stores do not meet the essential requirements for correctly tracking food consumption and recommendation.

Although a few apps had some of the expected features, none met all the required functionalities. For most of the apps, tracking information required manual data input. The data bases that are used in the apps are not enriched. We also observed that there are very few evidence -based apps. Because there have been numerous studies about automatic food recognition, food portion, size estimates and nutritional value assessments, these aspects must be included in modern food consumption tracking and recommendation apps. Also there has been much research on food recommendations but this feature is absent in most of the evaluated apps, that is why this feature is needs to be included in future apps. These apps suggest diet plans recommended foods to users, estimate nutrient values, so an expert dietitian or nutritionist should be involved in their development. Also, enrichment of the data base is required as nowadays multiple food data sets are available. Software qualities also play a vital role in commercial apps and thus developers thus need to consider these matters. Nonetheless, the analysis provided here covers a variety of general quality features and specific functional features that can be used in food consumption tracking and recommendation apps to provide consumers with a realistic and evidenced-based experience. This study will open the door to future researchers to focus on the implementation, effectiveness and performance measurement of food computing apps.

12. Future Scope

- Send notification to user’s to intake food at the correct time.
- Send notification to user’s as which food is taken for a particular time.
- To chat with experts and get suggestions about food intake as per their health conditions.
- User’s with medical condition can consult the experts through this app.

13. Appendix

```
from flask import Flask, render_template, request, redirect, url_for, session

import ibm_db

from flask_mail import Mail, Message
```

```
import re
from werkzeug.utils import secure_filename
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2, resources_pb2, service_pb2_grpc
from clarifai_grpc.grpc.api.status import status_code_pb2

app = Flask(__name__)
mail = Mail(app) # instantiate the mail class

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'derinjose886@gmail.com'
app.config['MAIL_PASSWORD'] = 'lhmjtfrjwblfbgeq'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ksm24043;PWD=ZXsdfH0rppz
tWofo",",")
@app.route('/')
def home():
    return render_template('register.html')

@app.route('/login', methods =['GET', 'POST'])
def login():
    global userid
    msg = "
    if request.method == 'POST'and 'username' in request.form and 'password' in request.form:
        username = request.form['username']

        password = request.form['password']

        stmt = ibm_db.prepare(conn,'SELECT * FROM accounts WHERE username = ?AND
password = ?')

        ibm_db.bind_param(stmt,1,username)

        ibm_db.bind_param(stmt,2,password)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        if account:

            session['loggedin'] = True

            session['username'] = account['USERNAME']

            msg = 'Logged in successfully !'

            return render_template('index.html', msg = msg)

        else:

            msg = 'Incorrect username / password !'

    return render_template('login.html', a = msg)


@app.route('/logout')

def logout():

    session.pop('loggedin', None)

    session.pop('id', None)

    session.pop('username', None)

    return redirect(url_for('login'))


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

def register():

    msg = ''

    if request.method == 'POST':

        username = request.form['username']

        email = request.form['email']

        password = request.form['password']
        sql = "SELECT * FROM accounts WHERE username = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        msgg = Message(
                'Hello',
                sender ='derinjose886@gmail.com',
                recipients = [email]
               )
        msgg.body = ' Welcome to NUTRI LITE!! Thanks for registering.  '
        mail.send(msgg)


        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Username must contain only characters and numbers !'
        elif not username or not password or not email:
            msg = 'Please fill out the form !'
        else:
            insert_sql = "INSERT INTO accounts (username,email,password) VALUES (?, ?, ?)"
            stmt = ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, password)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
    elif request.method == 'POST':
            msg = 'Please fill out the form !'
        return render_template('register.html', msg = msg)

    @app.route('/bmi', methods =['GET', 'POST'])
    def bmi():
        if request.method == 'POST':
            height = request.form['height']
            weight= request.form['weight']
        return render_template('bmi.html')



    @app.route('/img',  methods =['GET', 'POST'])
    def img():
        print(request.form)
        return render_template('image.html')

    @app.route('/food')
    def food():
        return render_template('food.html')

    @app.route("/dashboard", methods=["GET", "POST"])
    def dashboard():
      global request
      if flask.request.method == "POST" and session['LoggedIn']:
        if 'file' not in flask.request.files:
          flash('No file part')
          return redirect(flask.request.url)
        file = flask.request.files['file']
        if file.filename == '':
          flash('No image selected')
      return redirect(flask.request.url)
    if file and allowed_file(file.filename):
      filename = secure_filename(file.filename)
      file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
      flash('Image successfully uploaded')

      with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), "rb") as f:
        file_bytes = f.read()

      request = service_pb2.PostModelOutputsRequest(
          model_id="food-item-v1-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=file_bytes
                )
              )
            )
          ],
      )
      response = stub.PostModelOutputs(request, metadata=metadata)

      if response.status.code != status_code_pb2.SUCCESS:
          print(response)
          raise Exception(f"Request failed, status code: {response.status}")

      foodname = response.outputs[0].data.concepts[0].name

      ingredients = ''
```

```
    for concept in response.outputs[0].data.concepts:
      ingredients += f"{concept.name}: {round(concept.value, 2)}, "


    nutritionValues = ''
  #headers = {'X-RapidAPI-Key':
"e90a2b1101msh8a9c2a55215e6b8p1b6838jsn26de2538dc24",
  #'X-RapidAPI-Host': "spoonacular-recipe-food-nutrition-v1.p.rapidapi.com"}
    nutritions = {
      "recipesUsed": 10,
      "calories": {
        "value": 470,
        "unit": "calories",
        "confidenceRange95Percent": {
          "min": 408.93,
          "max": 582.22
        },
        "standardDeviation": 139.8
      },
      "fat": {
        "value": 17,
        "unit": "g",
        "confidenceRange95Percent": {
          "min": 12.81,
          "max": 21.36
        },
        "standardDeviation": 6.9
      },
      "protein": {
        "value": 15,
        "unit": "g",
        "confidenceRange95Percent": {
```

```
        "min": 9.06,
        "max": 29.78
      },
      "standardDeviation": 16.71
    },
    "carbs": {
      "value": 65,
      "unit": "g",
      "confidenceRange95Percent": {
        "min": 57.05,
        "max": 77.9
      },
      "standardDeviation": 16.81
    }
  }
  nutritions.pop('recipesUsed')
  for i in nutritions:
    nutritionValues += f"{i}: {nutritions[i]['value']} {nutritions[i]['unit']}, "


  sql = "INSERT INTO foods VALUES(?,?,?,?,?)"
  stmt=ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, session['userid'])
  ibm_db.bind_param(stmt, 2, datetime.datetime.now().strftime('%Y-%m-%d
%H:%M:%S'))
  ibm_db.bind_param(stmt, 3, foodname)
  ibm_db.bind_param(stmt, 4, ingredients)
  ibm_db.bind_param(stmt, 5, nutritionValues)
  ibm_db.execute(stmt)
```

```
      return render_template("dashboard.html",
        filename = filename,
        username = session['username'],
        foodname = foodname,
        ingredients = ingredients,
        nutritionValues = nutritionValues,
      )
    else:
      flash('Allowed image formats - png, jpg, jpeg')
      return redirect(flask.request.url)


if __name__ == '__main__':
 app.run(debug = True)
```

GitHub & Project demo link

GitHub link

https://github.com/IBM-EPBL/IBM-Project-24838-1659949758

https://github.com/IBM-EPBL/IBM-Project-24838-1659949758

Project demo link

https://drive.google.com/file/d/1-YDMSkcAn2v46EG-eU3KksFrkLjSqoJI/view?usp=share_link