Project Report

NUTRITION ASSISTANT APPLICATION

S.NO	REG.NO	NAME	DEPARTMENT	TEAM
1.	2127190701107	SRIMAN NARAYAN M	ECE	Team Lead
2.	2127190701109	SRIVANTH A	ECE	Team Member 1
3.	2127190701092	SAHITHYAN S	ECE	Team Member 2
4.	2127190701117	THARANI BALAN SK	ECE	Team Member 3

DONE BY
TEAM ID: PNT2022TMID53656

1. INTRODUCTION

The objective of this study is to identify dietary self-monitoring implementation strategies on a mobile application. Nutritional knowledge is essential for promoting good eating habits since it ensures that necessary nutrient requirements are met to avoid malnutrition.

Wellness and healthy lifestyles have become mainstream. Interest in fitness applications and revenue from them grow as fast as the number of people striving to be fit.

2. 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.

3. PURPOSE

You can automatically calculate the nutritional information for any recipe, analyze recipe costs, visualize ingredient lists, find recipes for what's in your fridge, find recipes based on special diets, nutritional requirements, or favorite ingredients, classify recipes into types and cuisines, convert ingredient amounts, or even compute an entire meal plan.

LITERATURE SURVEY

1) Existing problem

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.

2) References

Flask: https://www.youtube.com/embed/uxZuFm5tmhM

Send-Grid: https://sendgrid.com/

Rapid API: https://rapidapi.com/hub

Docker: https://www.youtube.com/embed/pTFZFxd4hOI

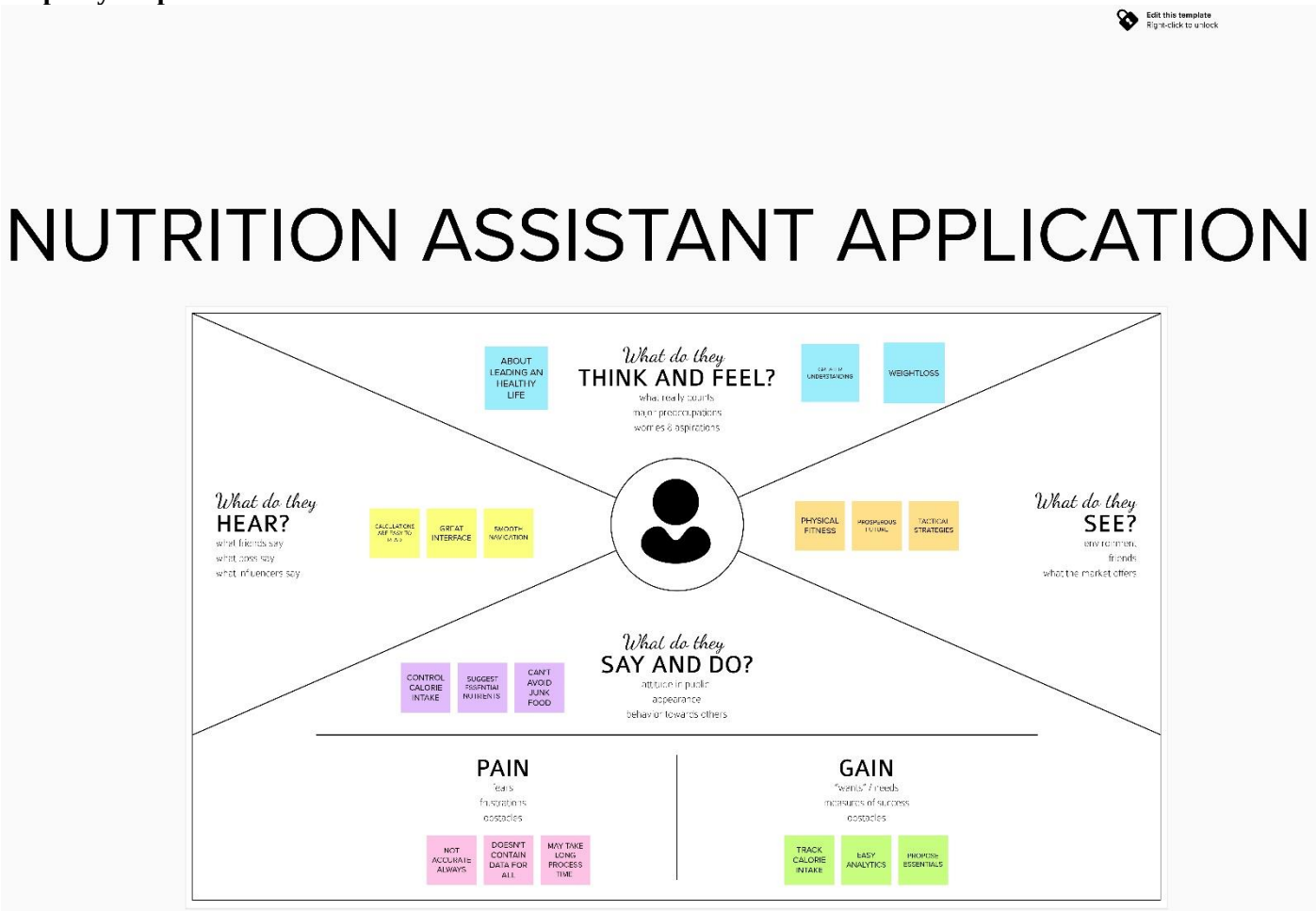
Kubernetes: https://www.youtube.com/embed/d6WC5n9G_sM

3) Problem Statement Definition

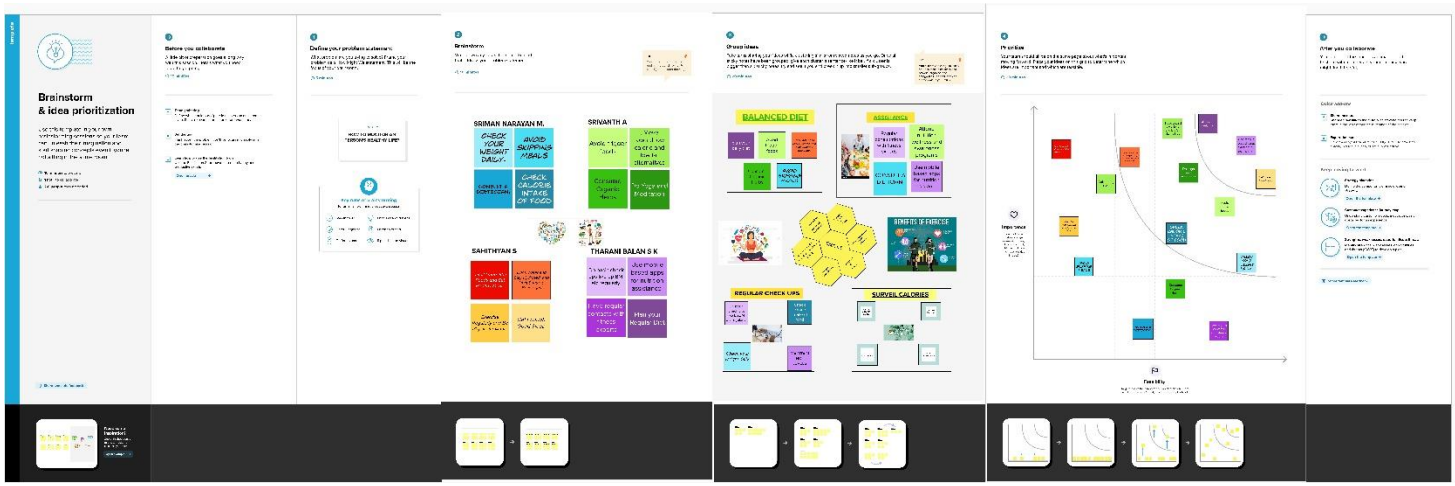
App-based nutrient dashboard systems which can analyze real-time images of a meal and analyze it for nutritional content which can be very handy and improves the dietary habits, and therefore, helps in maintaining a healthy lifestyle.

IDEATION & PROPOSED SOLUTION

1) Empathy Map Canvas



2) Ideation & Brainstorming

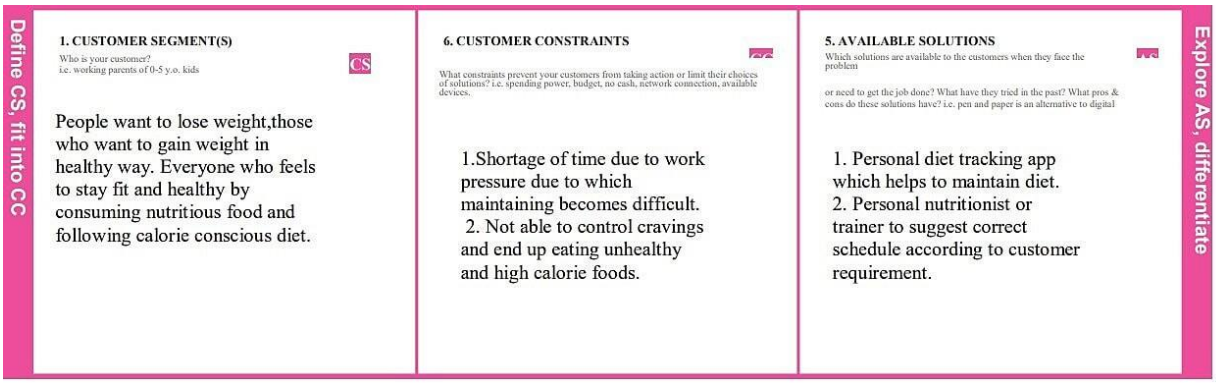


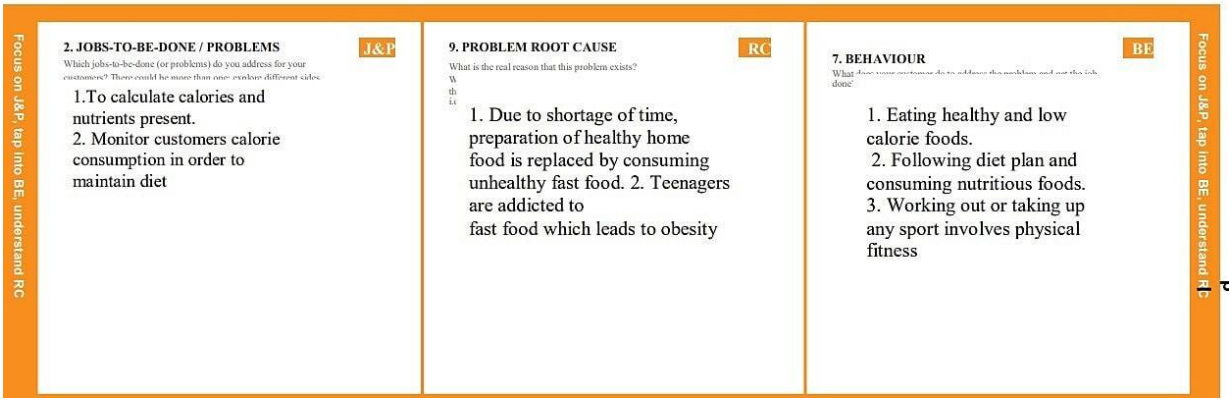
3) Proposed Solution

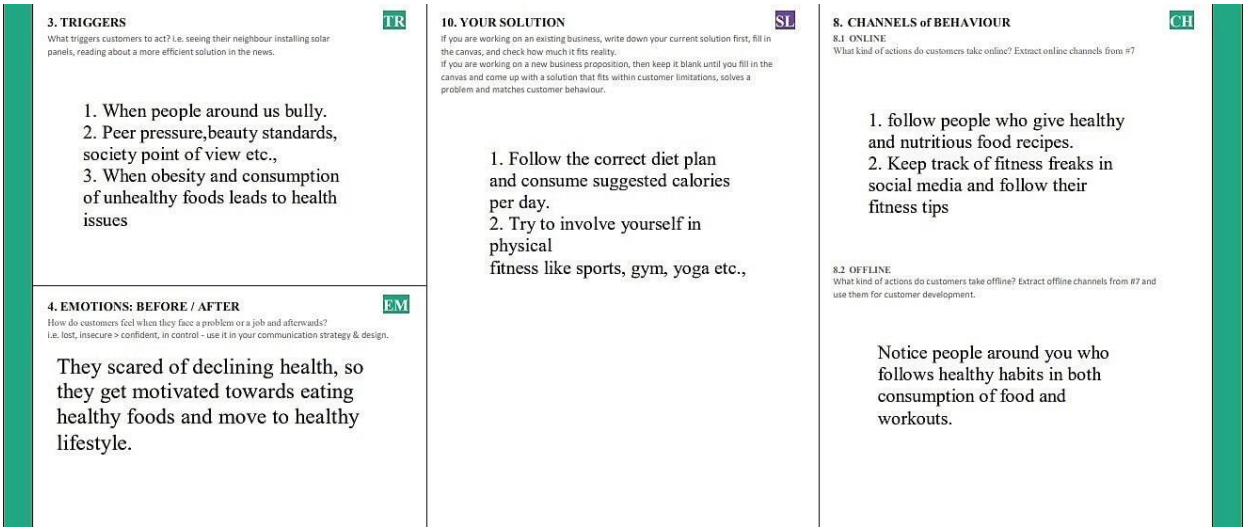
S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• It is easy to fall into a trap of eating unhealthy foods which is heavy in calories. Once the nutritional value is replaced by foods high in sugar, bad fats and salt it leads to various health issues so users need to control their daily calorie intake to lead a healthy lifestyle.
2.	Idea / Solution description	• The solution is a responsive Web application that can be used in any PCdevices. • The website provides a user-friendly interface and accepts multiple samplespredicting them simultaneously. • Our method uses Clarifai's AI- driven food recognition model to accurately identify food suggestions. • A detailed report of the concerned person's health will be generated.
3.	Novelty / Uniqueness	• Keep a food journal. • Providing individual diet charts for users based on their BMI and medical condition if any. • Provides recipes according to their diet. • Providing a user-friendly environment.
4.	Social Impact / Customer Satisfaction	• Getting feedback from the users for enhancement and giving notification on their diet plans and goal tracking. • Nutrition focused food banking & targetedin-depth reporting reviews that

		paid subscriptions the best.
5.	Business Model (Revenue Model)	• Advertising membership option for users to get more benefits like diet- plans or consultation from experts and In-app advertisements. • Revenue is generated on a subscription basis, with big data processing and targetedin-depth reporting reviews that paid subscriptions the best.
6.	Scalability of the Solution	• Providing regular updates • Efficient goal tracking assistance • The additional features such that sleep tracking, mensuration tracking can be done.

4) Problem Solution fit







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.

1. When people around us bully.
2. Peer pressure,beauty standards, society point of view etc.,
3. When obesity and consumption of unhealthy foods leads to health issues

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.

They scared of declining health, so they get motivated towards eating healthy foods and move to healthy lifestyle.

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.

1. Follow the correct diet plan and consume suggested calories per day.
2. Try to involve yourself in physical
fitness like sports, gym, yoga etc.,

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

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

1. follow people who give healthy and nutritious food recipes.
2. Keep track of fitness freaks in social media and follow their fitness tips

8.2 OFFLINE

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

Notice people around you who follows healthy habits in both consumption of food and workouts.

REQUIREMENT ANALYSIS

1) Functional requirement

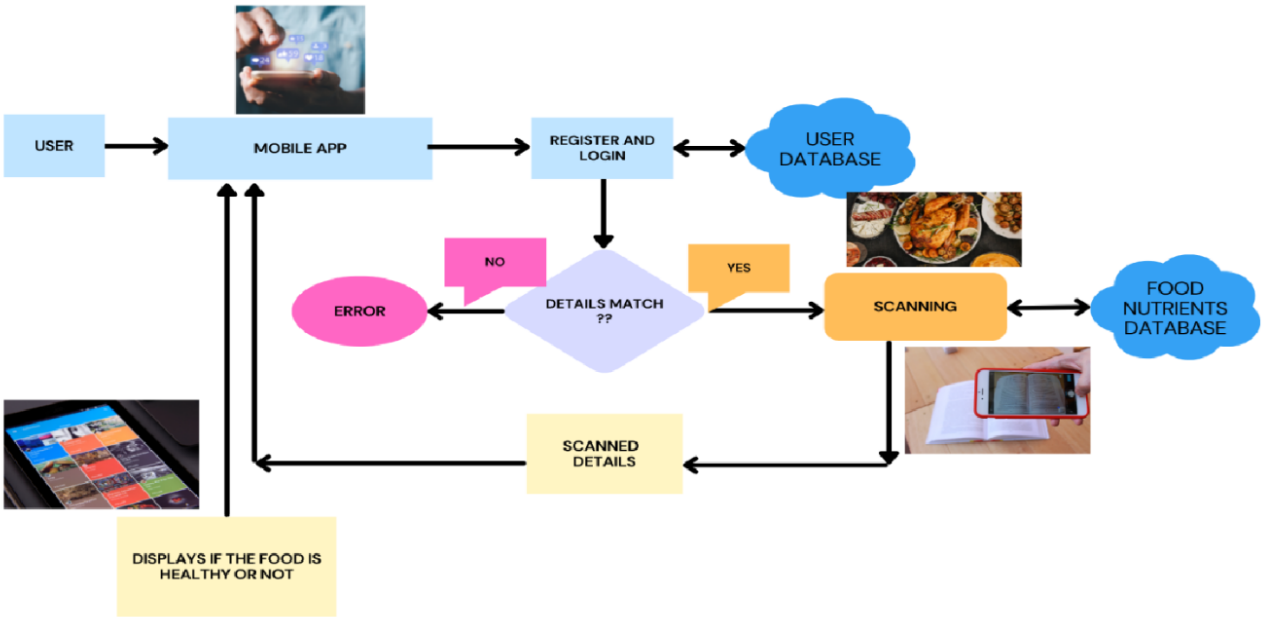
- User Registration
- User Confirmation
- Update Profile
- User Authentication
- Report

2) Non-Functional requirements

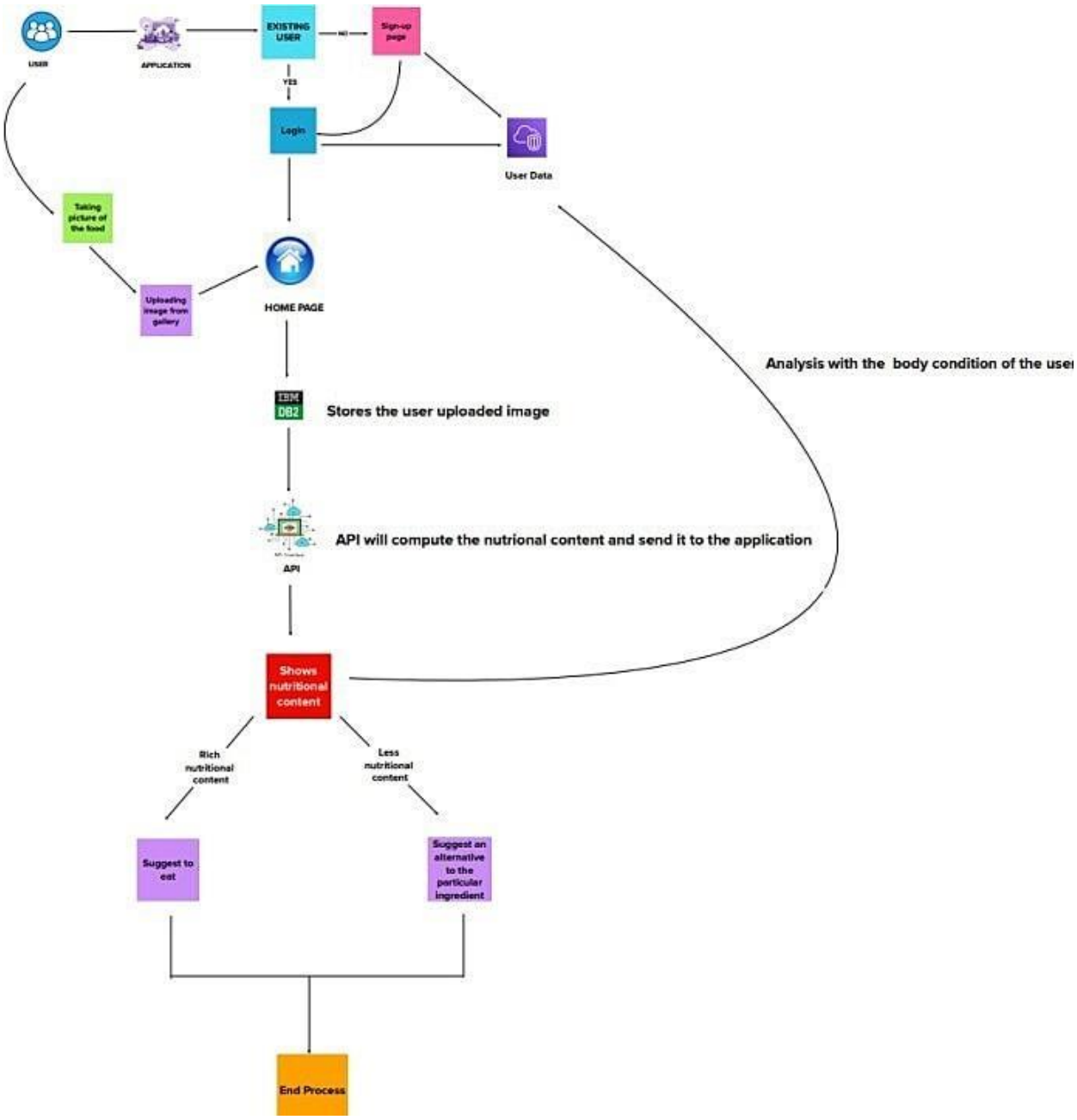
- Usability
- Security
- Reliability
- Performance
- Availability
- Scalability

PROJECT DESIGN

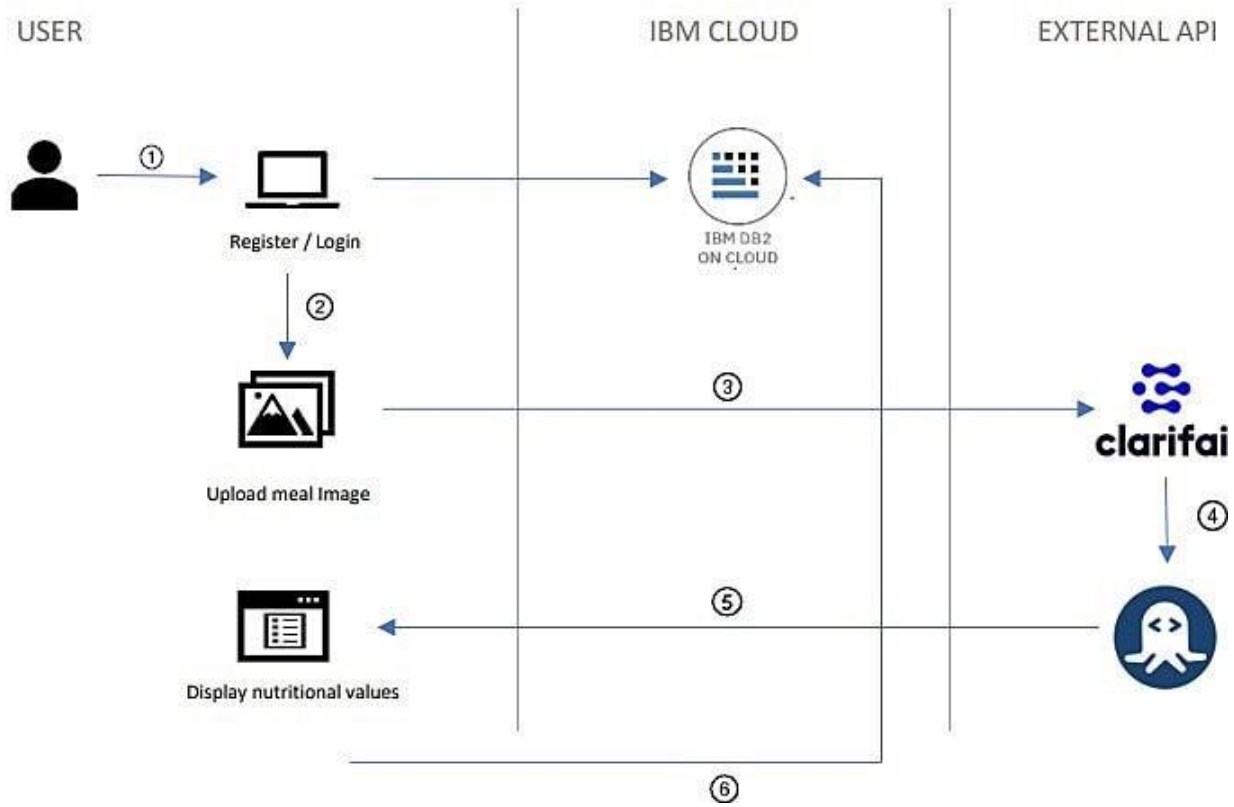
1) Data Flow Diagrams



2) Solution & Technical Architecture



Technical Architecture:



3) User Stories

- As a user, I can register for the application by entering my email, password, and Confirming my password
- As a user, I will receive confirmation email once I have registered for the application
- As a user, I can log into the application by entering email & password
- As a user, I can fill the details.
- As a user ,I will search the food items.
- As a user, I can scan the food an get the nutrition details and recipe for related scanned food.

PROJECT PLANNING & SCHEDULING

1) Sprint Planning & Estimation

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	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B
Sprint-2	User details	USN-4	As a user , I can fill the Details.	2	High	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B
Sprint-3	Push notification	USN-5	As a user, I will search the food items.	2	Medium	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B
Sprint-4	Shown the nutrition details and Recipe for scanned food	USN-6	As a user, I can scan the food an get the nutrition details and recipe for related scanned food	1	High	Arivanantha Pandian R Arunkumar S Antony Kevin S Balaji B

2) Sprint Delivery Schedule

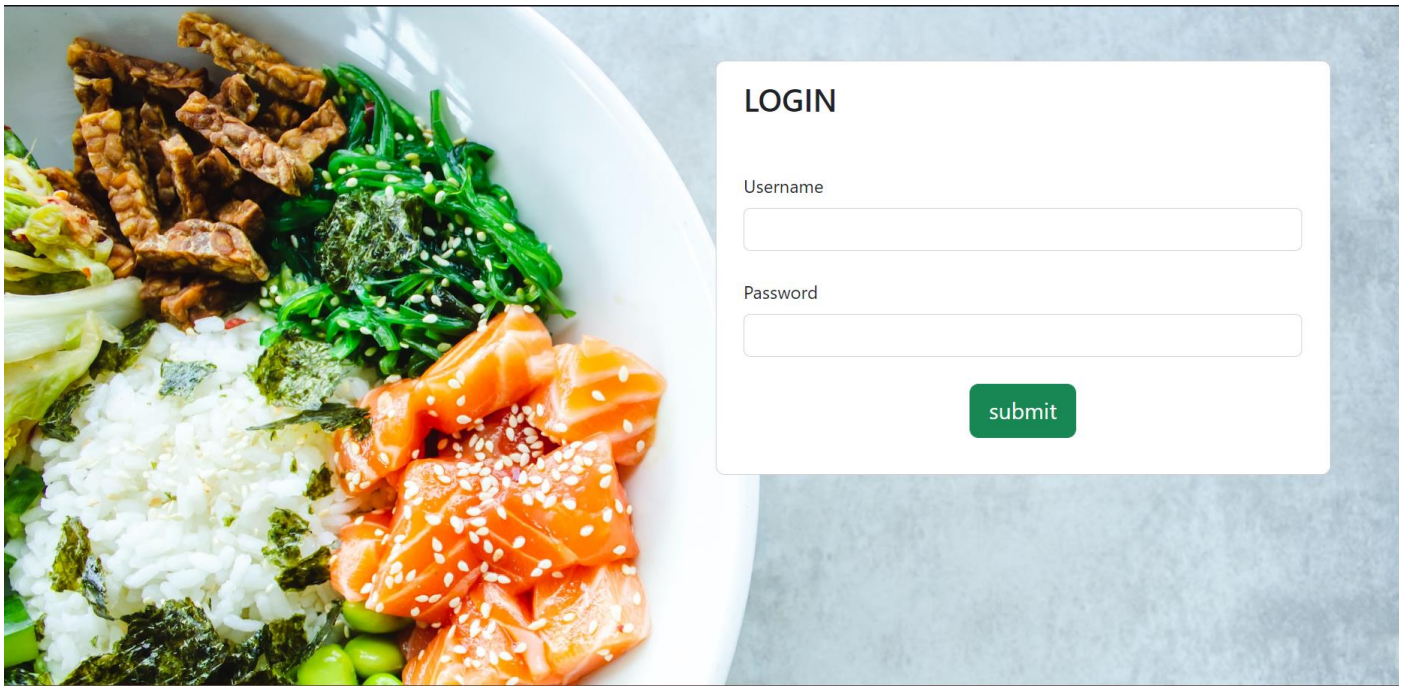
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CODING & SOLUTIONS

FEATURE 1:

LOGIN Algorithm :

1. Enter the credentials and hit enter (email and password).
2. If already logged in user is taken to home page
3. Else , check for validity of credentials entered using query to cloudant db.
4. If wrong credentials entered , notification displayed to user and user stays in login page.
5. On correct credentials , user is taken to home page.

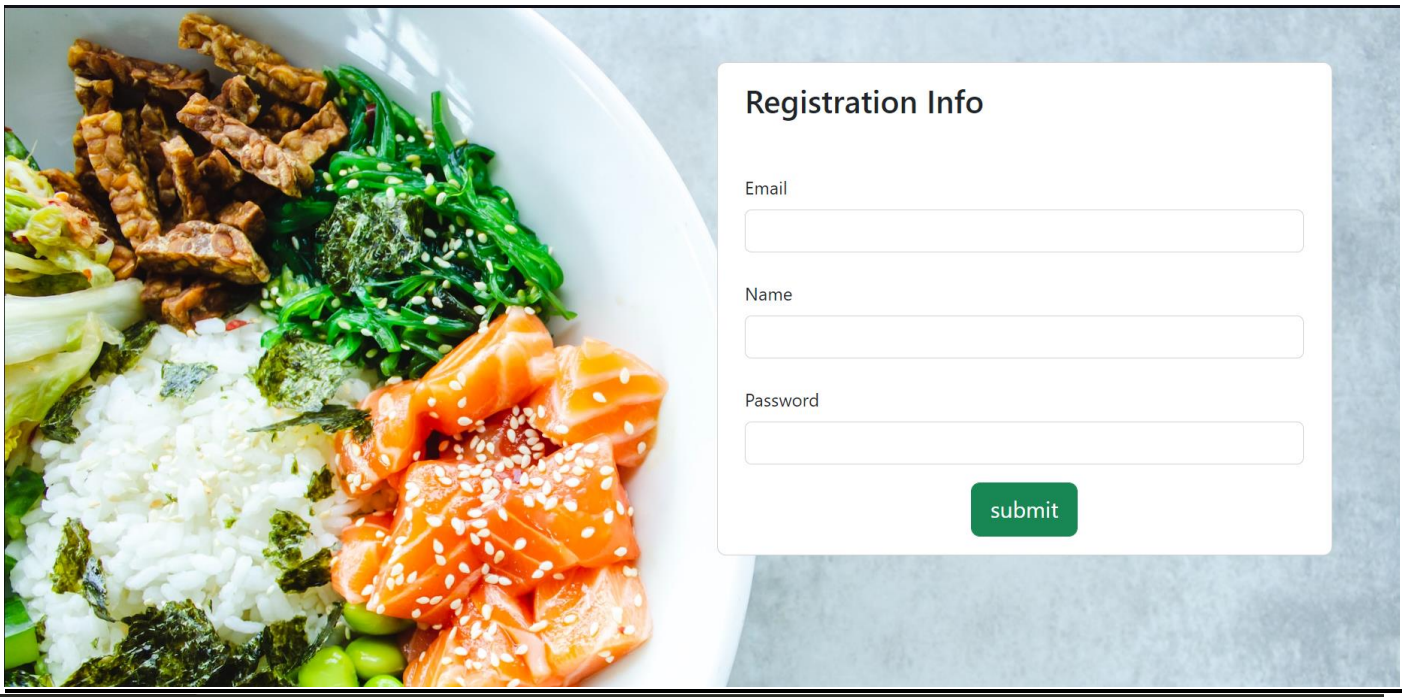


FEATURE 2

SIGNUP

Algorithm :

1. Enter the signup form fields (name , email , password) and hit enter.
2. All credentials are validated at client side.
3. Email is checked if already registered or not in the database.
4. If already registered , notification displayed. Or else, the user is taken to the successful signup page




```python
@app.route('/home', methods=['GET', 'POST'])
def homepage():
    if request.method == 'POST' and 'email' in request.form and 'pass' in request.form:
        return render_template('index.html', error="Wrong Password!")

    return render_template('index.html')


@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST' and 'name' in request.form and 'email' in request.form and 'pass' in request.form:

        name = request.form['name']
        email_up = request.form['email']
        pass_up = request.form['pass']

        if name == "":
            error = 'Enter a valid Name.'
            return render_template('index.html', error=error)

        if email_up == "":
            error = 'Enter a valid E-mail.'
            return render_template('index.html', error=error)

        if pass_up == "":
            error = 'Enter a valid Password.'
            return render_template('index.html', error=error)

        sql = "SELECT * FROM USER WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email_up)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return render_template('index.html', error="You are already a member, please login using your details")
        else:
            try:
                insert_sql = "INSERT INTO USER VALUES (?,?,?)"
                prep_stmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(prep_stmt, 1, name)
                ibm_db.bind_param(prep_stmt, 2, email_up)
```

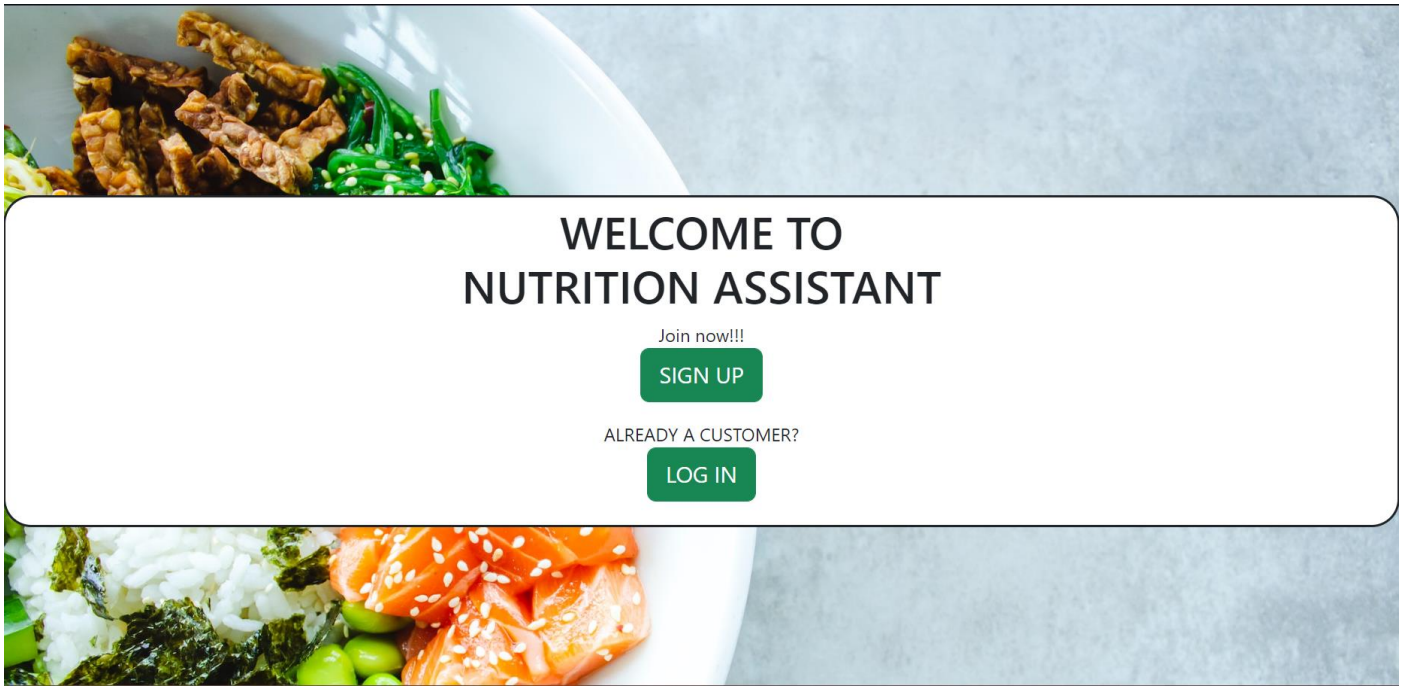
TESTING

1) Test Cases

i. Our code was tested on various food to check whether it gives the correctoutput
ii. To satisfy the customer's expectations we tested it fully.

2) User Acceptance Testing

Our project was tested by an end user to verify that it's working correctly.



RESULTS

Performance Metrics

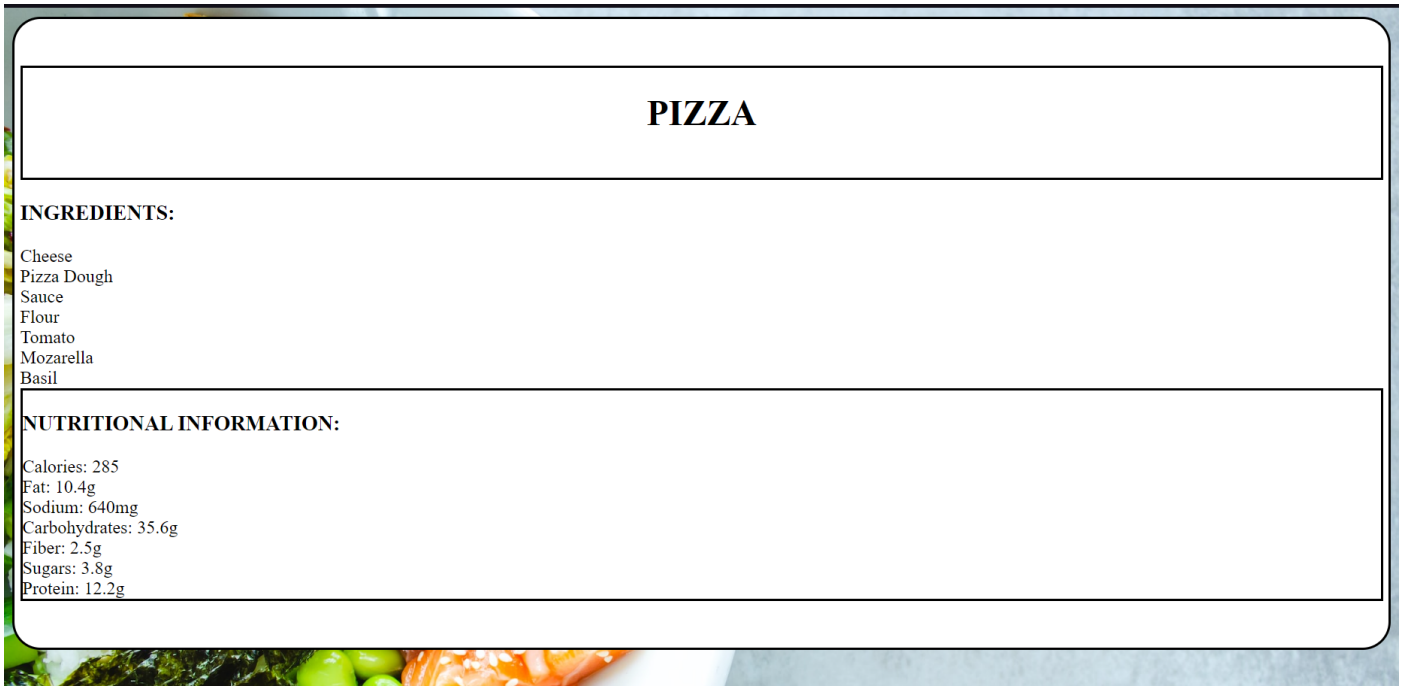
The proposed procedure was implemented and tested set of images. The training database consists of various images of food items. Once a food is recognized the equivalent **Nutrition** in shown on the screen

Upload image of your food

TO VIEW THE NUTRIONAL INFORMATION OF A FOOD, UPLOAD THE IMAGE OF A FOOD IN ORDER TO SEE NUTRITIONAL INFORMATION

Choose File pizza.jpg

Submit



TESTING:

A) TEST CASES

1.Login button click with wrong credentials entered.

2.Signup with already registered mail ID.

3.Signup with wrong form data entered.

4.Entering home page with logged out session.

5.Clicking home page buttons with logged out session.

6.Invalid data entered in change password page and requested for change in password.

8.2 USER ACCEPTANCE TESTING

S.NO	TEST CASE	REQUIRED OUTPUT	RESULT OUTPUT	STATUS
1	Login button click with wrong credentials	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
2	Signup with already registered mail ID.	Email already registered notification	Email already registered notification	ACCEPTED
3	Signup with wrong form data entered.	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
4	Entering home page with logged out session.	Take user to login page	Take user to login page	ACCEPTED
5	Clicking home page buttons with logged out session.	Take user to login page	Take user to login page	ACCEPTED
6	Invalid data entered in change password page and requested for change in password.	Wrong form data entered notification	Wrong form data entered notification	ACCEPTED

ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

1.Low cost.

2.Simple UI.

3.Faster response due to single page web page.

4.Capability of adding many features with ease and less cost.

DISADVANTAGES :

1.Lack of efficiency . Efficiency of the product needs to be improved.
2.Consistency of the product is not 100%.

3.Not a compact sized product. Size needs to be decreased.

CONCLUSION

Nutrition education is important because it has the potential to improve the health and extend the lives of generations of Americans. The results of this survey indicate that nutrition education is of great interest to educators. It is offered by most public schools, is covered in many grades, and a wide range of topics are covered. However, even though nutrition education is an active area, the intensity and quality of the nutrition messages students are receiving is not known. In addition, because nutrition education is concentrated in the health curriculum, science classes, and school health programs, the proportion of students participating at each grade level is not known.

There appears to be room for additional coordination of nutrition education across different subjects within the curriculum, across grade levels, and between the curriculum and the school meals program. The survey findings also indicate that schools are focusing on increasing students' knowledge about what is meant by good nutrition, with less emphasis on influencing students' motivation, attitudes, and eating behaviors. One objective of nutrition education is to increase knowledge. Other objectives are to change unhealthy attitudes so students have the motivation to establish healthy eating practices and teach positive skills so students have all the tools to accomplish their nutritional goals. However, less than one-third of schools that covered topics related to motivation, attitudes, or behavior provided thorough coverage of those topics.

Dietary tracking is an essential task in chronic disease management and intervention. Food photo taking and image recognition significantly reduce the burden of food entering on personal mobile devices. In this work, we have developed a dietary tracking system that applies the deep-based image recognition to accurately and efficiently log food and nutrition intake. Through real user food photo testing and user study, we found that laboratory models form the foundation of the solution but miss out some of the key challenges. The diversity of real food photos is higher than the lab trained model. An ingredient based recognition is a promising way of tracking the free style and homemade food recognition problems in which training data is sparse and not representative. Moreover, the proposed photo based portion selection method is shown to be more accurate and engages the users better than the existing 25 methods.

FUTURE SCOPE

In future we'll be adding more features which will benefit the users. The ui/ux of the web application will be improved. Scaling the project for more use cases and customers. Implementing distributed computing for efficient processing. Making encryption standard for cloud storage.

APPENDIX

1) Source Code

```
from flask import Flask,render_template,request,redirect,url_for ,session
import ibm_db
import re
import os
import math
import random
import smtplib
import requests
app=Flask(__name__,template_folder='templates',static_folder='static')
app.secret_key='a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=3
print("successfully connected")
@app.route('/')
def home():
    return render_template('index.html')


@app.route('/login',methods=['GET','POST'])
def login():
    global userid
    msg=''

    if request.method=='POST':
        username=request.form.get('username',False)
        password=request.form.get('password',False)
        sql='SELECT * FROM USER WHERE username=? AND password=?'
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Logged in']=True
            session['id']=account['USERNAME']
            userid=account['USERNAME']
            session['username']=account['USERNAME']
            msg='Logged in successfully'
```

```
            msg="You have successfully registered"
            return render_template('verify.html',msg=msg)
    elif request.method=="POST":
        msg="Please fill out the form"
    return render_template('register.html',msg=msg)

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

@app.route('/verify')
def verify():
        email=request.args.get('email', None)
        server=smtplib.SMTP('smtp.gmail.com',587)
        server.starttls()
        password="nsgeuedwbzptosyp"
        server.login(email,password)
        otp=''.join([str(random.randint(0,9))for i in range(4)])
        msg=' YOUR OTP IS'+str(otp)
        server.sendmail(email,email,msg)
        server.quit()
        if request.method=='POST':
            verify=request.method['code']
            if verify==otp:
                return render_template('login.html')
        return render_template('verify.html')

@app.route('/frgpwd', methods=['GET','POST'])
def frgpwd():
    msg =" "
    print(request.form)
    username1=request.form.get("uname", False)
    oldpassword=request.form.get("oldpassword", False)
    newpassword=request.form.get("newpassword", False)
    sql='SELECT * FROM USER WHERE username=?'
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,username1)
```

```python
            return render_template('dash.html')
        else:
            msg='Incorrect username/password'
    return render_template('login.html',msg=msg)

@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']
        Firstname=request.form['firstname']
        lastname=request.form['lastname']
        #phoneno=request.form['phoneno']
        sql='SELECT * FROM USER WHERE username=?'
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        #ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg="Account already exist!"
        elif not re.match(r'[^@]+@[^@]+\.[^@]+',email):
            msg="Invalid email address"
        elif not re.match(r'[A-Za-z0-9]+',username):
            msg="name must contain character and numbers"

        else:
            insert_sql='INSERT INTO USER values(?,?,?,?,?)'
            prep_stmt=ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt,1,username)
            ibm_db.bind_param(prep_stmt,2,email)
            ibm_db.bind_param(prep_stmt,3,password)
            ibm_db.bind_param(prep_stmt,4,Firstname)
            ibm_db.bind_param(prep_stmt,5,lastname)
            ibm_db.execute(prep_stmt)
```

```python
        chgpwd_sql='UPDATE USER SET password = ? WHERE username = ?'
        prep_stmt=ibm_db.prepare(conn, chgpwd_sql)
        ibm_db.bind_param(prep_stmt,1,newpassword)
        ibm_db.bind_param(prep_stmt,2,username1)
        ibm_db.execute(prep_stmt)
        msg="You have successfully changed password"
        return render_template('forgot password.html',msg=msg)
    return render_template('forgot password.html',msg=msg)

url = "https://low-carb-recipes.p.rapidapi.com"

headers = {
  "x-rapidapi-key": "ad933ea36amsh6b0a83e514b1a58p14bc9ejsne745a5851a1b",
  "x-rapidapi-host":  "low-carb-recipes.p.rapidapi.com"
  }

searchForRecipes = "/search"
getRecipe="/recipes/"
getImage="/images/2807982c-986a-4def-9e3a-153a3066af7a.jpeg"
getRandomRecipe="/random"

@app.route('/login/dash')
def dashboard():
    return render_template('dash.html')

@app.route('/login/dash/viewprofile')
def viewprofile():
    username=session['id']
    sql='SELECT * FROM USER 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)
    print(account)
    if account:
        return  render_template('viewprofile.html')
    else:
```

```python
@app.route('/login/dash/viewprofile/personinfo',methods=['GET','POST'])
def per_info():
    msg=''
    if request.method =='POST':
        Name=request.form['Name']
        gender=request.form['gender']
        tar_weight=request.form['Target Weight']
        Age=request.form['Age']
        Height=request.form['Height']
        Weight=request.form['Weight']
        email=request.form['email']
        location=request.form['location']
        phoneno=request.form['phoneno']
        sql='SELECT * FROM USER WHERE username=?'
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,Name)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            insert_sql='INSERT INTO USER values(?,?,?,?,?,?,?)'
            prep_stmt=ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt,1,Name)
            ibm_db.bind_param(prep_stmt,2,gender)
            ibm_db.bind_param(prep_stmt,3,Age)
            ibm_db.bind_param(prep_stmt,4,Height)
            ibm_db.bind_param(prep_stmt,5,Weight)
            ibm_db.bind_param(prep_stmt,7,location)
            ibm_db.execute(prep_stmt)
            msg="Your details are successfully stored"
            return render_template('viewprofile.html',msg=msg)
    elif request.method=="POST":
        msg="Please fill out the form"
    return render_template('personal info.html',msg=msg)
```

```python
        if account:
            insert_sql='INSERT INTO USER values(?,?,?)'
            prep_stmt=ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt,1,Name)
            ibm_db.bind_param(prep_stmt,2,email)
            ibm_db.bind_param(prep_stmt,3,Feedback)
            ibm_db.execute(prep_stmt)
            msg="Your Feedback has been stored"
            return render_template('ratings.html',msg=msg)
    elif request.method=="POST":
        msg="Please fill out the form"
    return render_template('ratings.html',msg=msg)

@app.route('/dash/view recipe')
def search_page():
  #session ['item']=request.form.get("Ingridients", False)
  return render_template('search.html')
@app.route('/recipes')
def get_recipes():
  #food=session['item']
  if (str(request.args['ingridients']).strip() != ""):
      print(request.args['ingridients'])
      # If there is a list of ingridients -> list
      querystring = {"name":request.args['ingridients'],"tags":request.args['tag'],"includeIngredients":request.args['included'],"exclude
      response = requests.request("GET", url + searchForRecipes, headers=headers, params=querystring)
      data=response.json()
      return render_template('recipes.html', recipes=data)
  else:
      # Random recipes
      response = requests.request("GET", url+ getRandomRecipe  , headers=headers)
      data=response.json()
      return render_template('recipes.html', recipes=data)

@app.route('/recipe')
def get_recipe():
  recipe_id = request.args['id']
  recipe_info_endpoint = "/recipes/{0}".format(recipe_id)
```

```
        data=response.json()
        return render_template('recipes.html', recipes=data)

@app.route('/recipe')
def get_recipe():
  recipe_id = request.args['id']
  recipe_info_endpoint = "/recipes/{0}".format(recipe_id)
  print(recipe_info_endpoint)
  recipe_info = requests.request("GET", url + recipe_info_endpoint, headers=headers)
  data=recipe_info.json()
  return render_template('recipe.html', recipe=data)


@app.route('/logout')
def logout():
    session.pop('loggedin',None)
    session.pop('id',None)
    session('username',None)
    return render_template("index.html")

if __name__=="__main__":
    app.run(debug=True ,host='0.0.0.0',use_reloader=False)
```

2) GitHub

https://github.com/IBM-EPBL/IBM-Project-3304-1658539564

3) Project Demo Link

https://drive.google.com/file/d/1CBz048_V2b6j6wPIKTfvr6lLMUFN22_C/view?usp=sharing