**IBM NALAIYA THIRAN**

**PROJECT REPORT**

**NUTRITION ASSISTANT APPLICATION**

|  |  |
| --- | --- |
| **Team Id** | **PNT2022TMID02126** |
| **Project Name** | **NUTRITION ASSISTANT APPLICATION** |
| **Team Members** | * **Gokul D (2116190701049)** * **Akash S (2116190701009)** * **Gokulakrishnan K (2116190701054)** * **Boobalan D (2116190701027)** |

# Table Of Contents

|  |  |  |
| --- | --- | --- |
| **SI.**  **No** | **Title** | **Page No** |
| 1 | **INTRODUCTION**   * 1. Project Overview   2. Purpose | 4  4 |
| 2 | **LITERATURE SURVEY**   * 1. Existing problem   2. References   3. Problem Statement Definition | 5  5  5 |
| 3 | **IDEATION & PROPOSED SOLUTION**   * 1. Empathy Map Canvas   2. Ideation & Brainstorming   3. Proposed Solution   4. Problem Solution fit | 6  7  8  10 |
| 4 | **REQUIREMENT ANALYSIS**   * 1. Functional requirement   2. Non-Functional requirements | 11  11 |
| 5 | **PROJECT DESIGN**   * 1. Data Flow Diagrams   2. Solution & Technical Architecture | 12  13 |

|  |  |  |
| --- | --- | --- |
| 6 | **PROJECT PLANNING & SCHEDULING**   * 1. Sprint Planning & Estimation   2. Sprint Delivery Schedule   3. Reports from JIRA | 15  16  16 |
| 7 | **CODING & SOLUTIONING**   * 1. Feature 1   2. Feature 2   3. Database Schema (if Applicable) | 17  18  18 |
| 8 | **TESTING**   * 1. User Acceptance Testing   2. Load Testing | 20  21 |
| 9 | **RESULTS**  9.1 Performance Metrics | 22 |
| 10 | **ADVANTAGES & DISADVANTAGES** | 28 |
| 11 | **CONCLUSION** | 29 |
| 12 | **FUTURE SCOPE** | 29 |
| 13 | **APPENDIX**   * 1. Source Code   2. Github & Demo link | 30  42 |

**1.INTRODUCTION**

**1.1 Project Overview**

This project is aimed at developing a desktop-based application named Nutrition Assistant Application for estimating food attributes such as ingredients and nutritional value by classifying the input images of food. The Nutrition Assistant Application refers to the system and processes to help the user to analyze the intake of food with the involvement of a Technology system. This system can be used to store the details of the user’s health, calculating the BMI, Classifying the food image to know the nutritional value, update the status of their health condition based on the information provided, and generate health reports weekly or monthly based. This project is categorizing the individual health condition of the user. The Nutrition Assistant Application is important to control their daily calorie intake by eating healthier foods, which is the most basic method to avoid obesity. Without proper diet control, this is reflective of the risks to people’s health. A good Nutrition Assistant Application will alert the users when it is time to avoid. 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.

**1.2 Purpose**

Users still insist on learning how much nutrition their food contains. Users get knowledge about how various diets affect human health. This application's major goal is clearly to show users how to manage their diets while living healthy lives. By snapping a photo of the item, uploading a photo from the gallery, or manually entering the data, the user can obtain the nutritional information**.**

So, The Nutrition Assistant Application helps the users to eat nutritional rich food which yields to lead a healthy life.

**2. LITERATURE SURVEY**

**2.1 EXISTING PROBLEM**

In this pandemic situation, we need to lead a healthier life by means of taking healthier intake of foods .But in our fast moving world while we taking food we can’t find a chart and check whether the food is nutritional food or not . Thus to overcome that risky we created a application known as nutrition assistant application.

**2.2 REFERENCE**

[1] Adnan Shehzad , Hui Su , Jin Lui , Si Chin , Zhidong Shen (2019) “Machine Learning Based Approach on Food Recognition and `Nutrition Estimation”, International Conference On Identification, Information And Knowledge In The Internet Of Things(IOT)(IIKI2019).

[2] Alisha Lalani, Md.Riyazudin, Mousmi Ajay Chaurasia, Salva Fathima, Syed Ibrahim Ibaad (2022) “Estimation of Quantity and Nutritional Information Using Image Processing”, International Journal Of Scientific And Engineering Research.

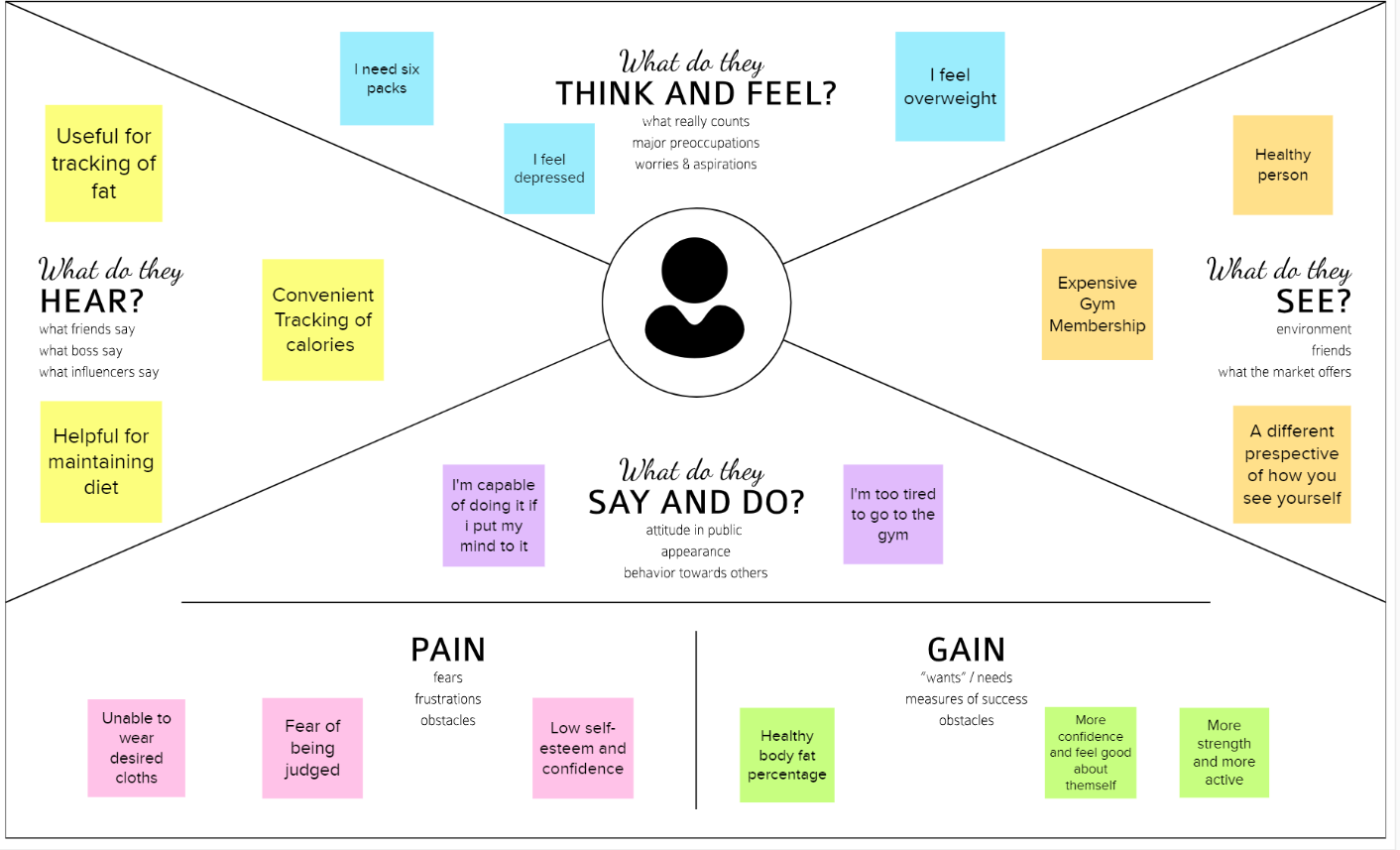
[3] Bojia Qiu , Chenxi Huang , Kunhui Lin , Landu Jiang , Xue Liu (2022) “Deep Food: Food Image Analysis and Dietary Assessment via Deep Model”, International Journal Of Scientific And Engineering Research.

**2.3 Problem Statement Definition**

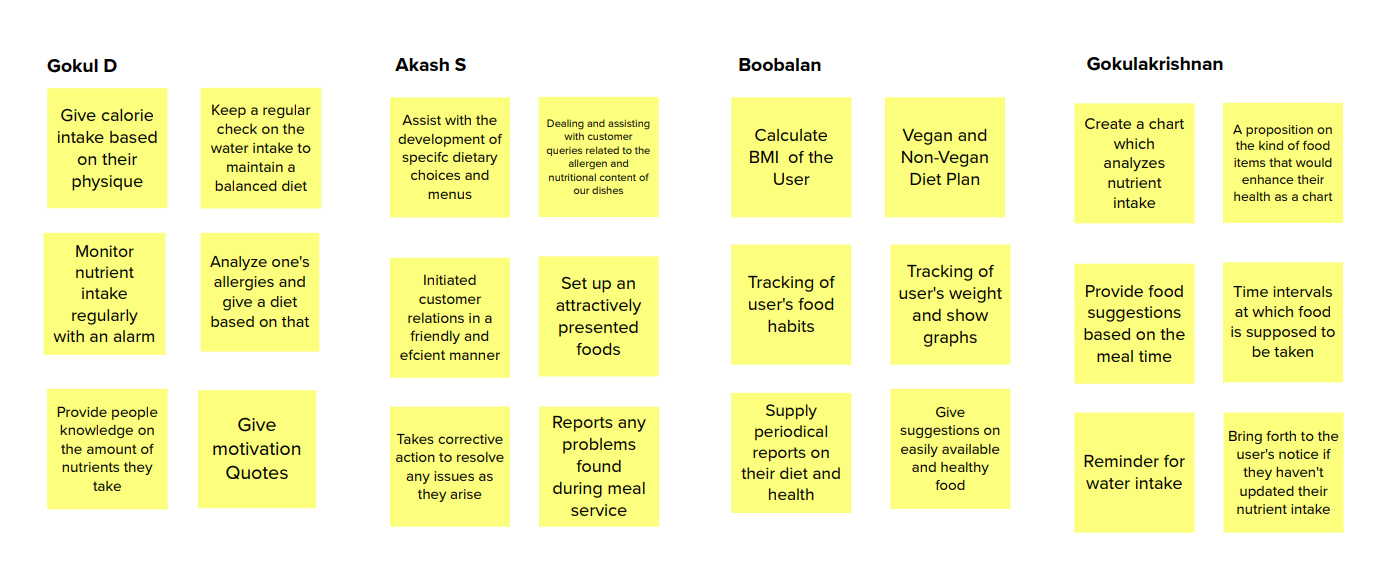
Due to the ignorance of healthy food habits, obesity rates are increasing at an alarming speed, and this is reflective of the risks to people’s health. People need to control their daily calorie intake by eating healthier foods, which is the most basic method to avoid obesity. However, although food 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 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.

**3. IDEATION AND PROPOSED SYSTEM**

**3.1 Empathy Map Canvas**

****

**3.2 Ideation and Brainstorming**

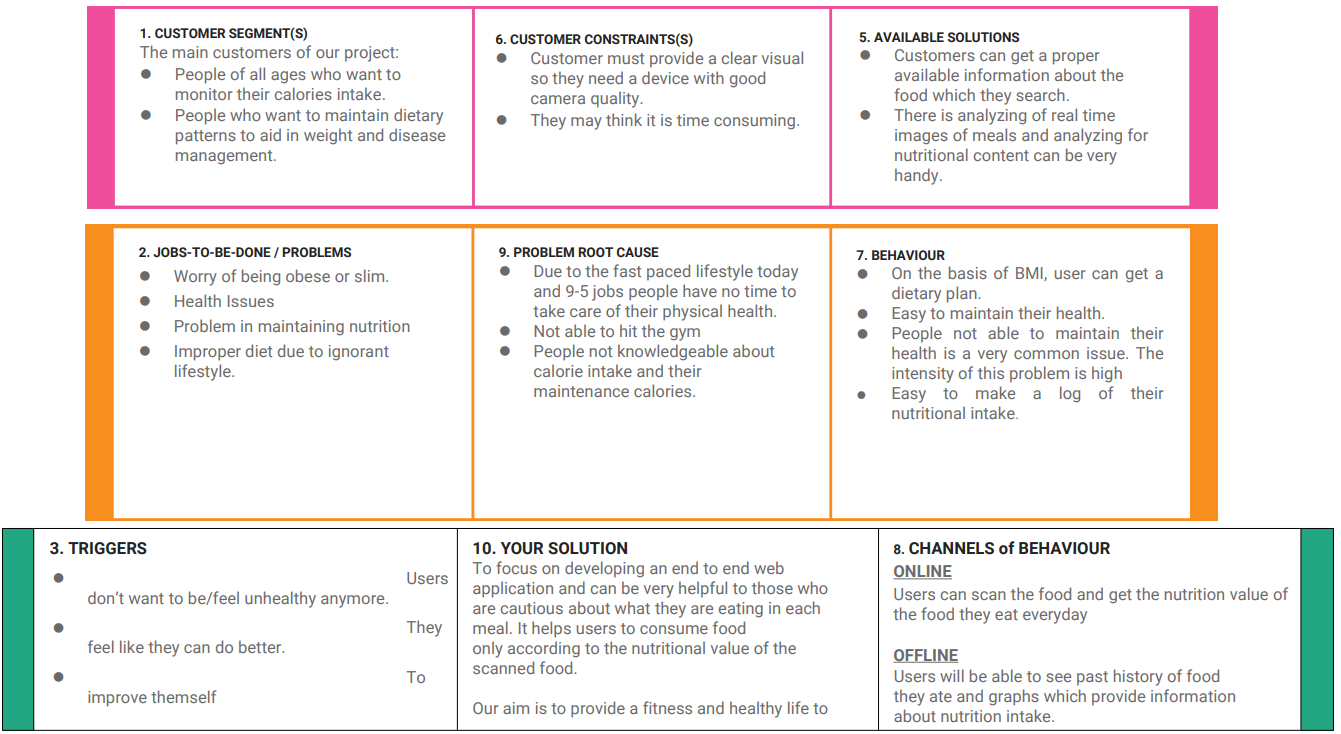
****

**3.3 Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be  solved) | 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 |

|  |  |  |
| --- | --- | --- |
| 2. | Idea / Solution description | The solution is a responsive web page that can be used in both mobile and computers. Cumulative results of pictures of food as input and provide nutritional information of food are used to achieve accurate prediction. The website provides a user-friendly interface and accepts multiple samples predicting them simultaneously. A detailed report of the concerned person’s health will be generated. |
| 3. | Novelty / Uniqueness | ● Patients to More Easily Monitor their caloric intake and dietary Pattern to aid in weight and disease Management.  ● Our method uses Clarifai's AI-driven food recognition model to accurately identify foods.  ● A food API that reports the nutritional value of identified foods.  ● Frequent checking of nutritional value and Customized food suggestions.  ● Water and medicine monitoring  ● Keep a food journal |
| 4. | Social Impact / Customer Satisfaction | Calculate the basal metabolic rate, body mass index, ideal weight and caloric intake. Nutrition Focused Food Banking. Targeted Food Assistance Programs. |
| 5. | Business Model (Revenue Model) | Revenue is generated on a subscription basis, with big data processing and targeted in-depth reporting reviews that paid subscriptions the best. |
| 6. | Scalability of the Solution | Furthermore, features can be extended in our application. Additional features such as sleep tracking, menstruation tracking can be done. |

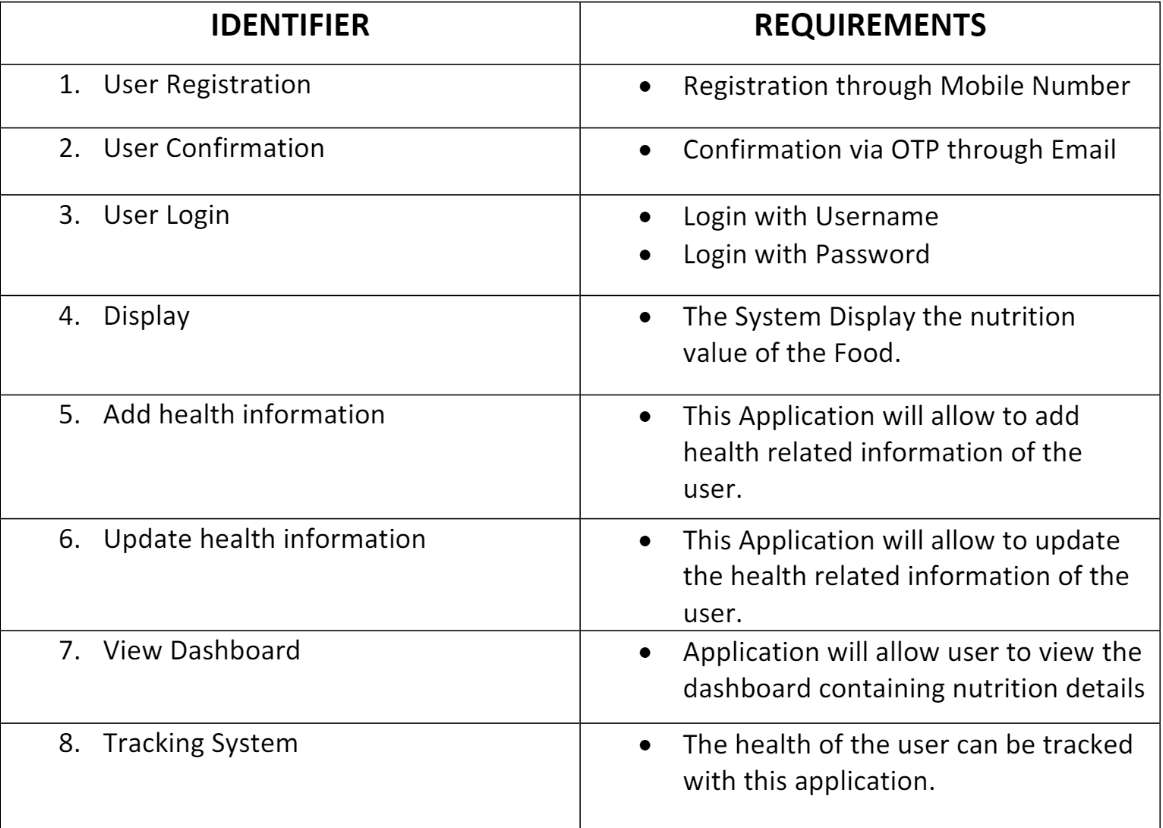
**3.4 Problem Statement Fit**

****

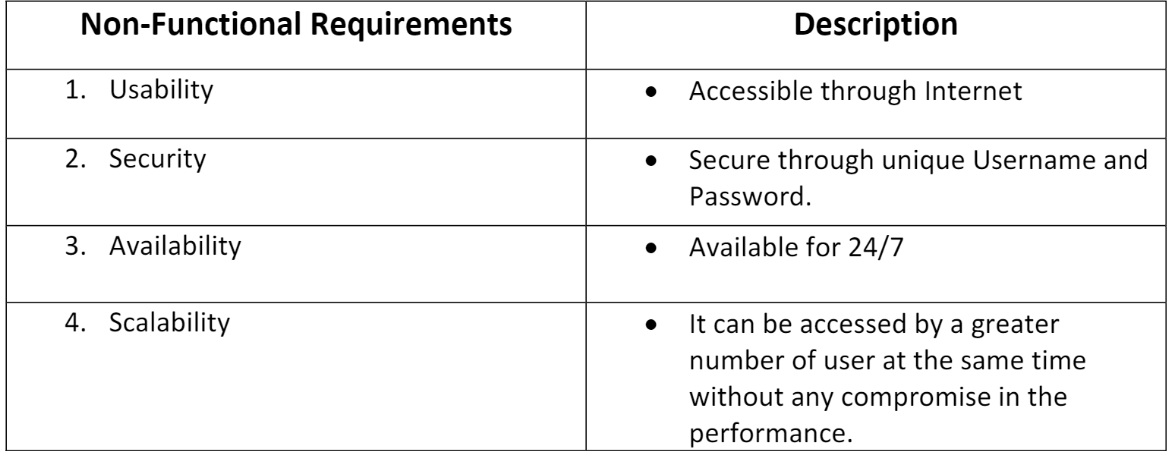
**3. REQUIREMENT ANALYSIS**

**4.1 Functional Requirements:**

Following are the functional requirements of the proposed solution.

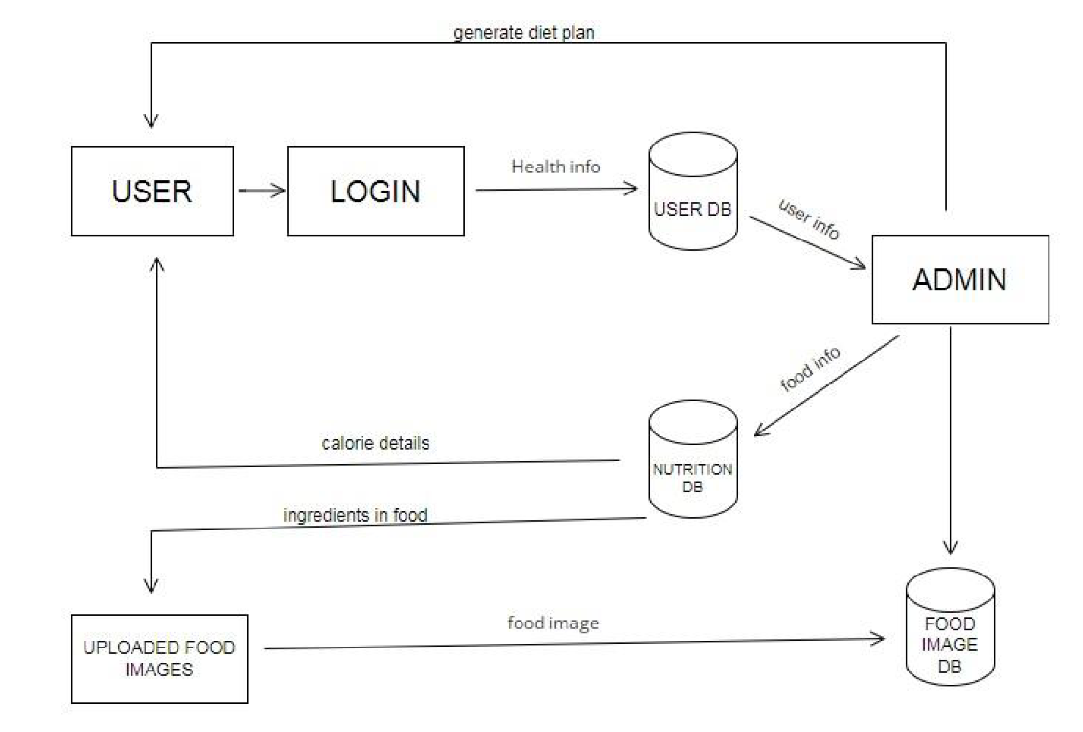


**4.2 Non-Functional Requirements:**

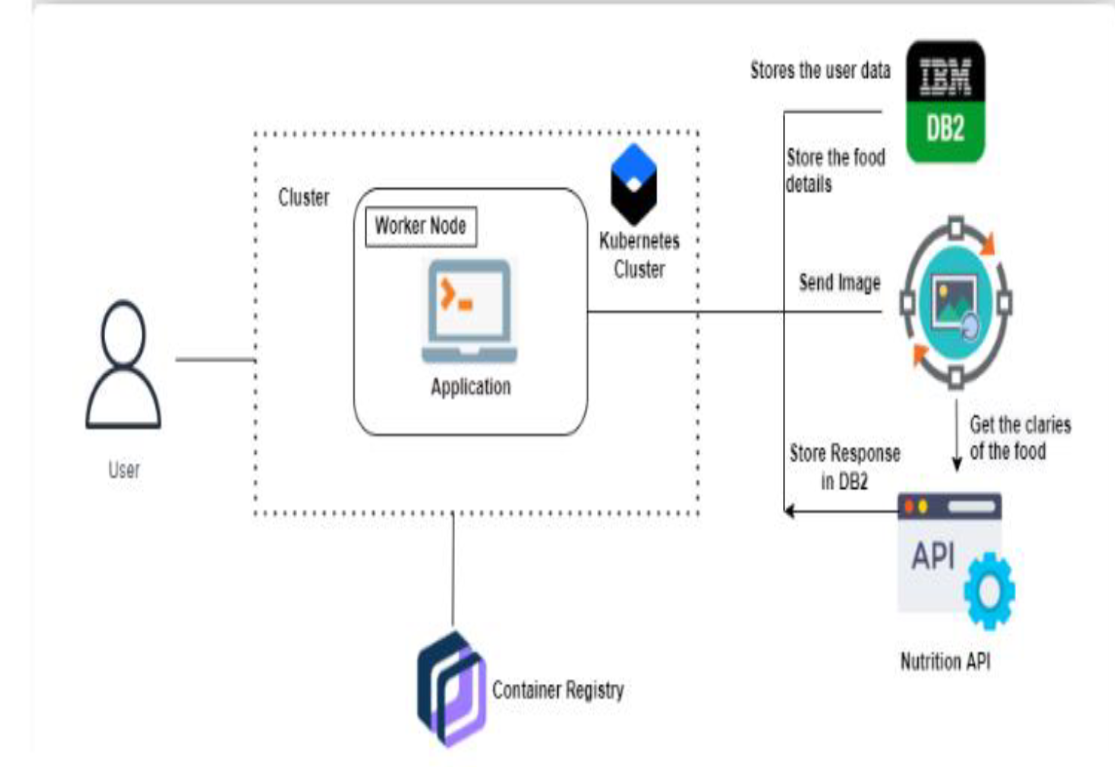
Following are the non functional requirements of the proposed solution

**5. PROJECT DESIGN**

**5.1 Data Flow Diagram**



**5.2 Solution and Technical Architecture:**

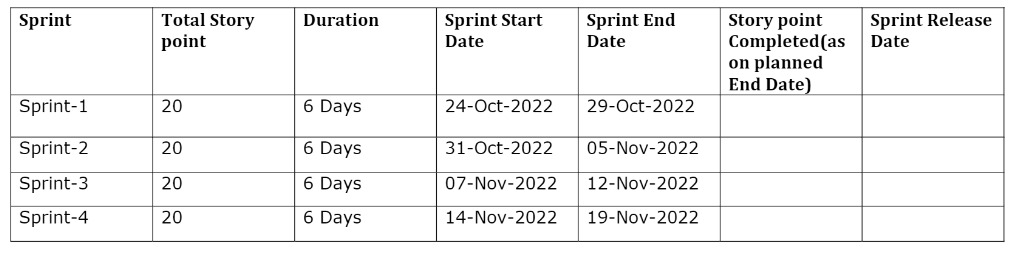


**6. PROJECT PLANNING AND SCHEDULING**

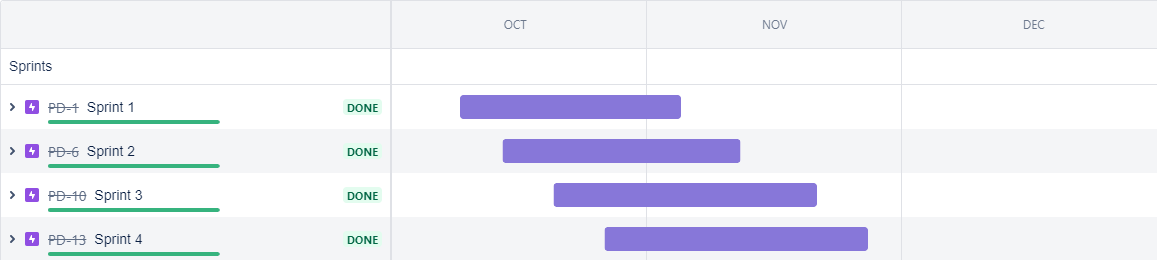
**6.1 Sprint Planning & Estimation**

****

**6.2 Sprint Delivery Schedule:**

****

**6.3 Reports from JIRA**

****

**7. CODING & SOLUTIONING**

**7.1 Feature - 1 - Email Service**

To the project, we incorporated an email service. This service

sends email messages with nutrition-related information directly to

customers' inboxes.

def custom\_send\_mail(email, data):

sg = sendgrid.SendGridAPIClient(SENDGRID\_API\_KEY)

from\_email = Email("nutritioninyourlife.foryoy@gmail.com")

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"'{data}'")

mail = Mail(from\_email, to\_email, subject, content)

# Get a JSON-ready representation of the Mail object

mail\_json = mail.get()

sg.client.mail.send.post(request\_body=mail\_json)

**7.2 Feature - 2 - Storing Records**

We store the nutrition-related information on the database, so

users can access the data when they need it.

**Adding result into database,**

insert\_sql = "INSERT INTO PERSON VALUES (?,?,?,?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, session['name'])

ibm\_db.bind\_param(prep\_stmt, 2, session['email'])

ibm\_db.bind\_param(prep\_stmt, 3, complete\_value)

ibm\_db.bind\_param(prep\_stmt, 4, current\_time)

ibm\_db.execute(prep\_stmt)

**Getting information from the database**

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

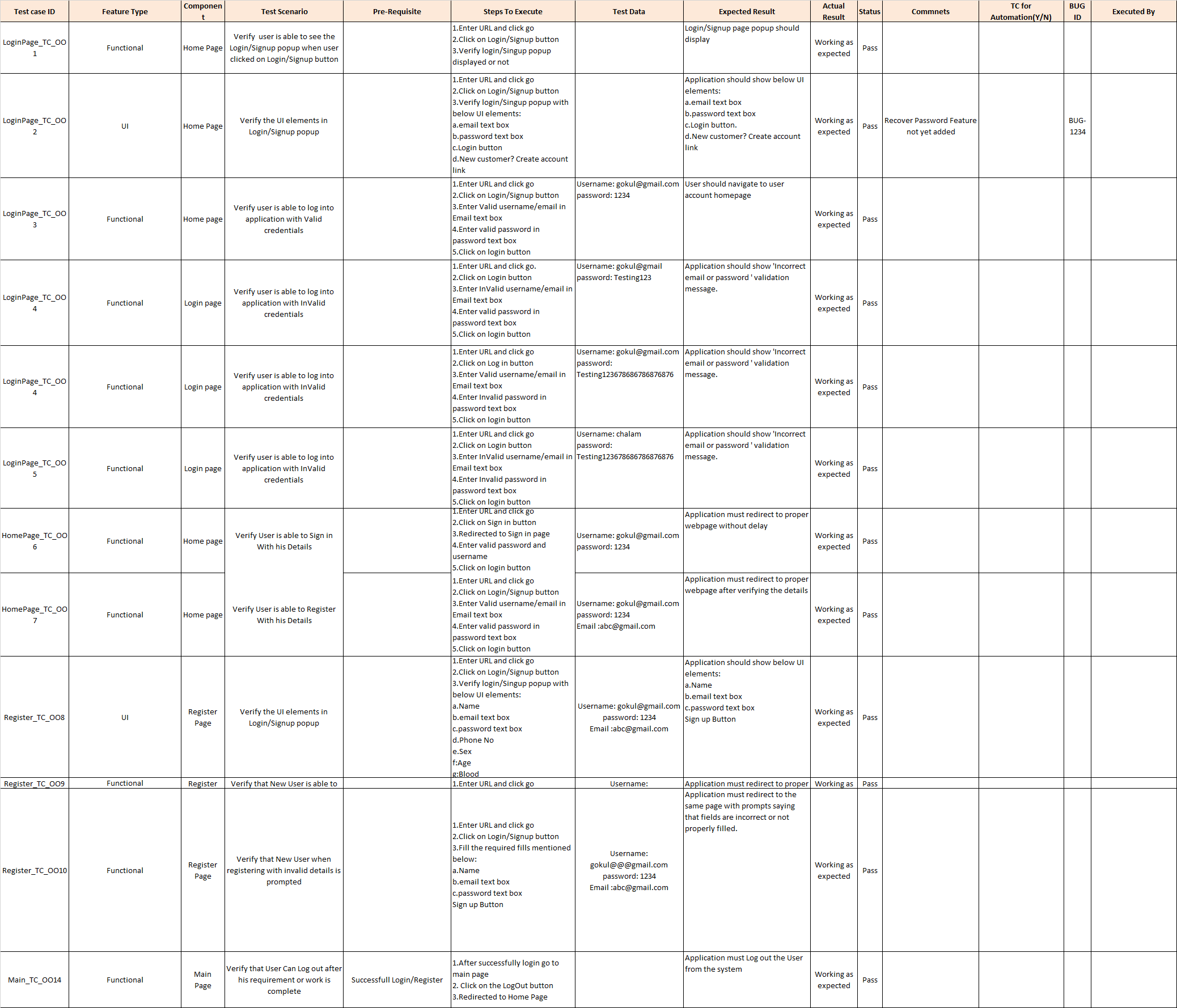
**7.3 Database Scheme**

**IBM DB2**

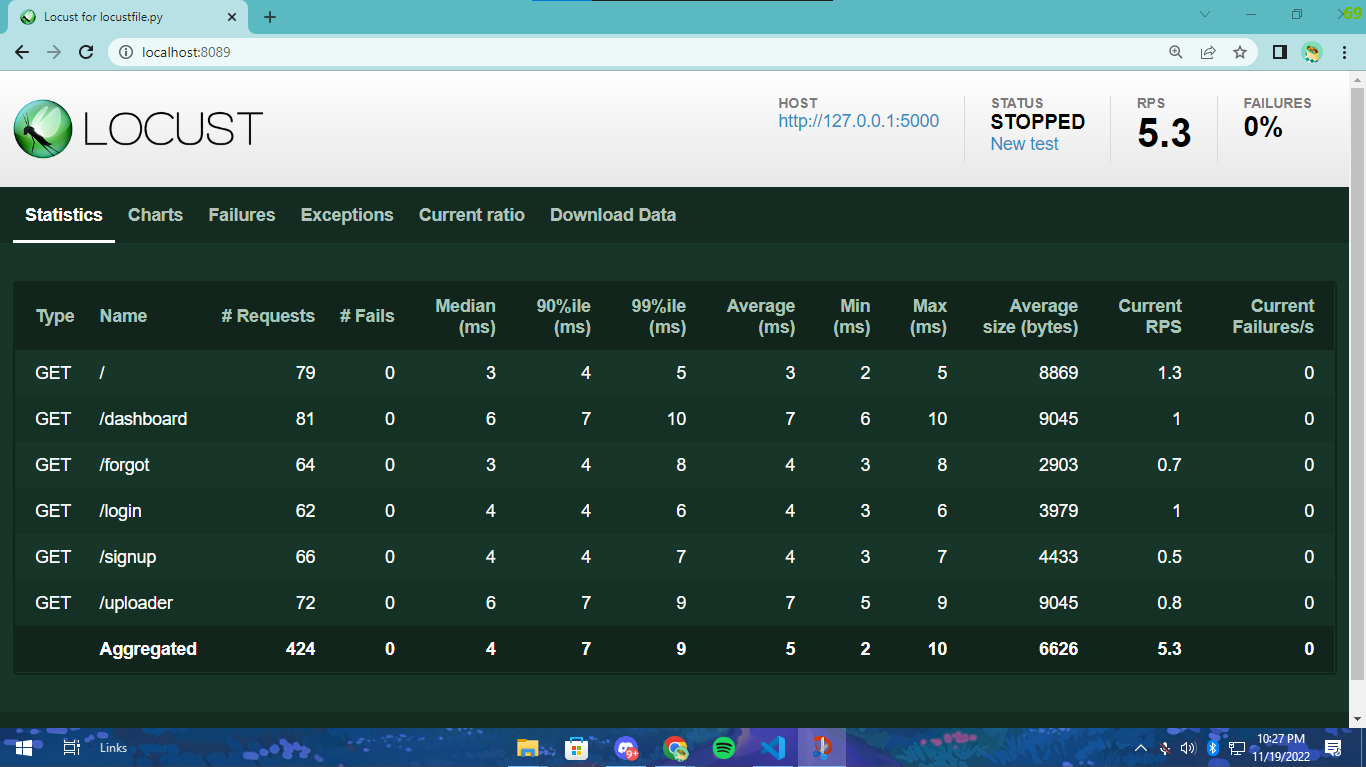
****

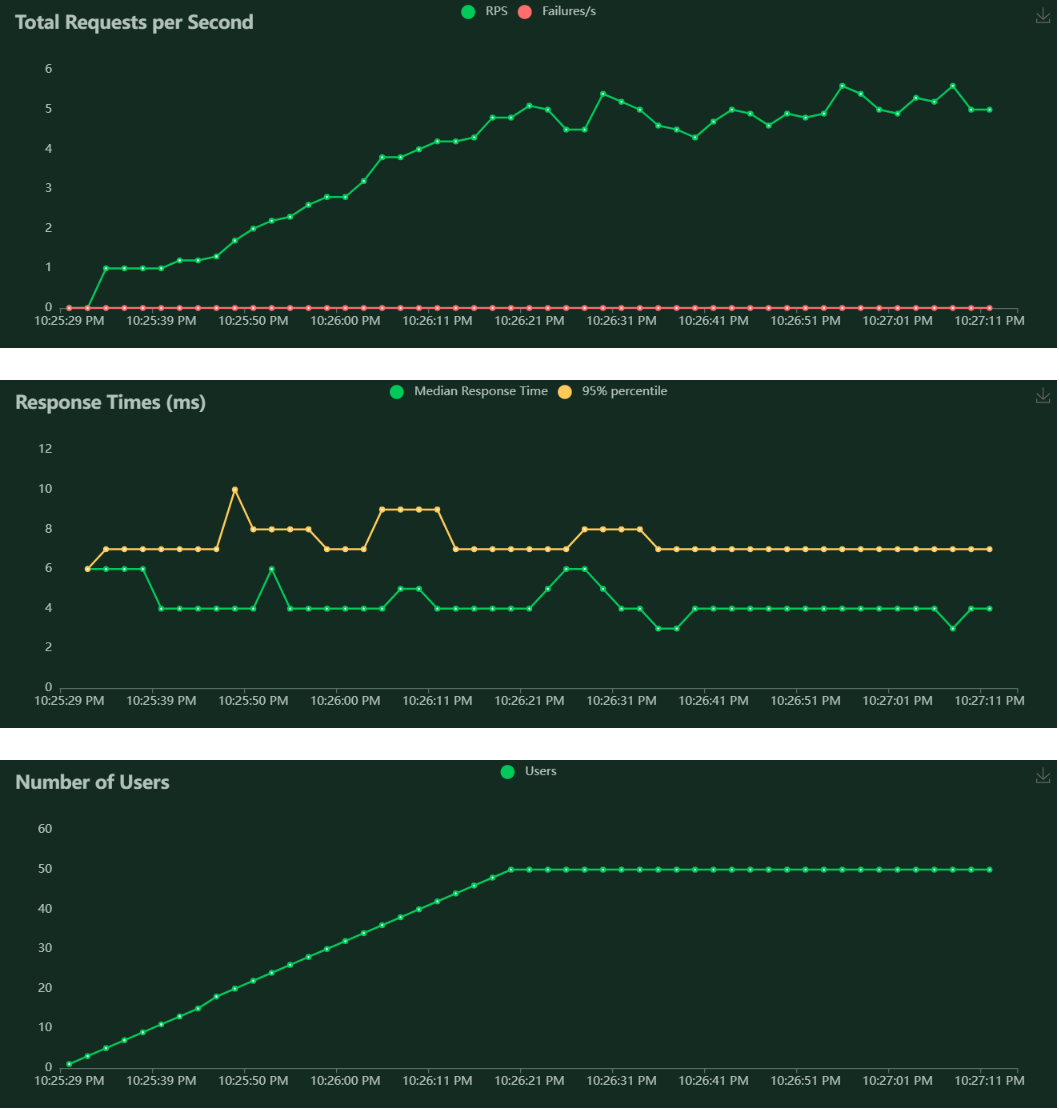
**8.TESTING**

**8.1 User Acceptance Testing**

****

**8.2 Load Testing**

****

****

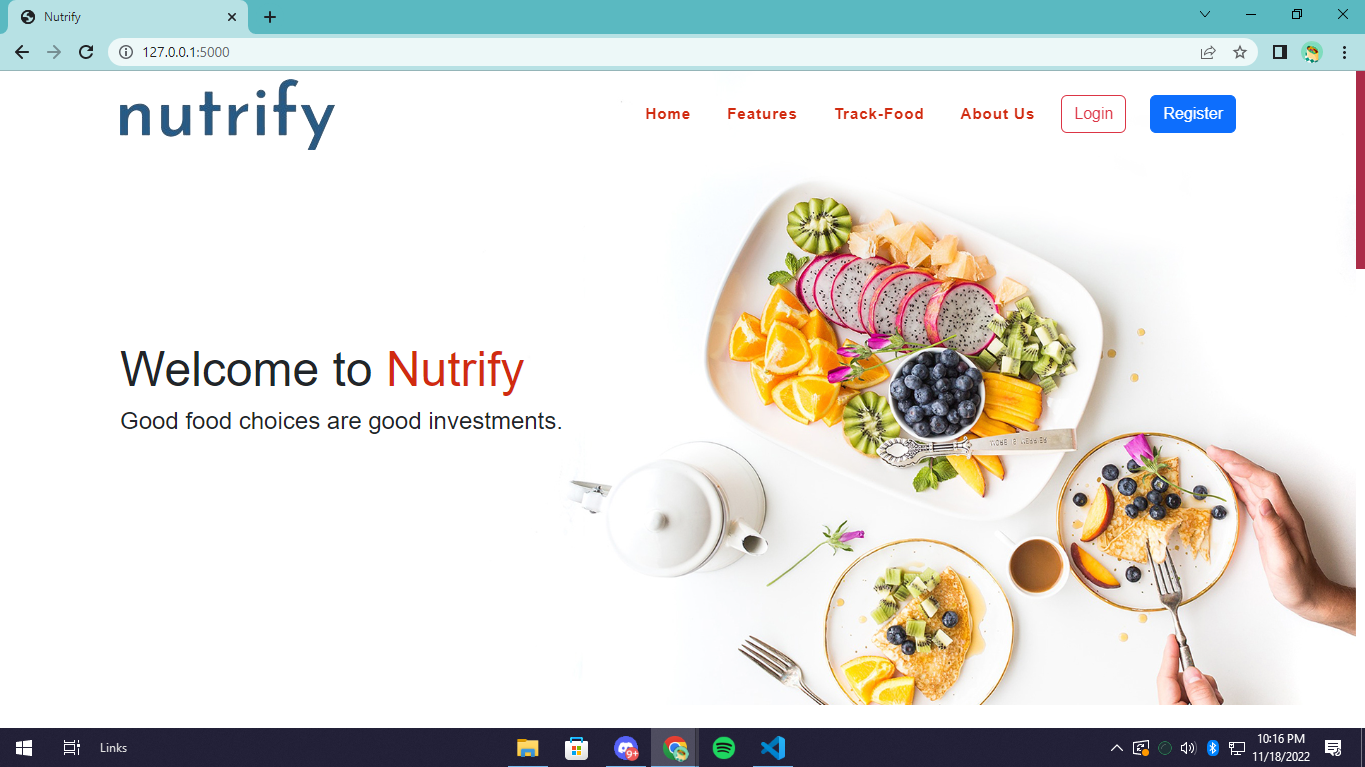
**9. RESULTS**

**9.1 Performance Metrics**

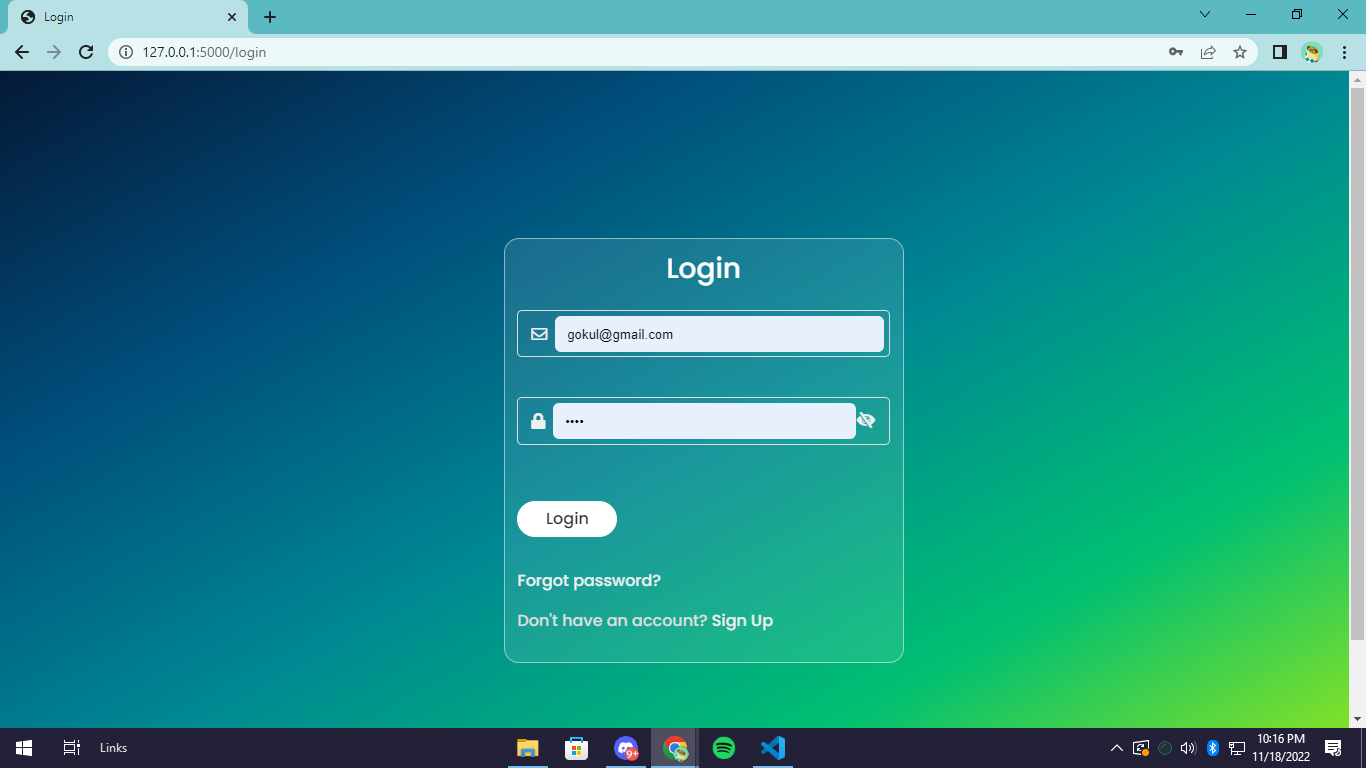
* Project metrics are used to track the progress and performance of a project.
* Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
* As a project evolves, managers need access to changing deadlines or budgets to meet their client's expectations

**Outputs**

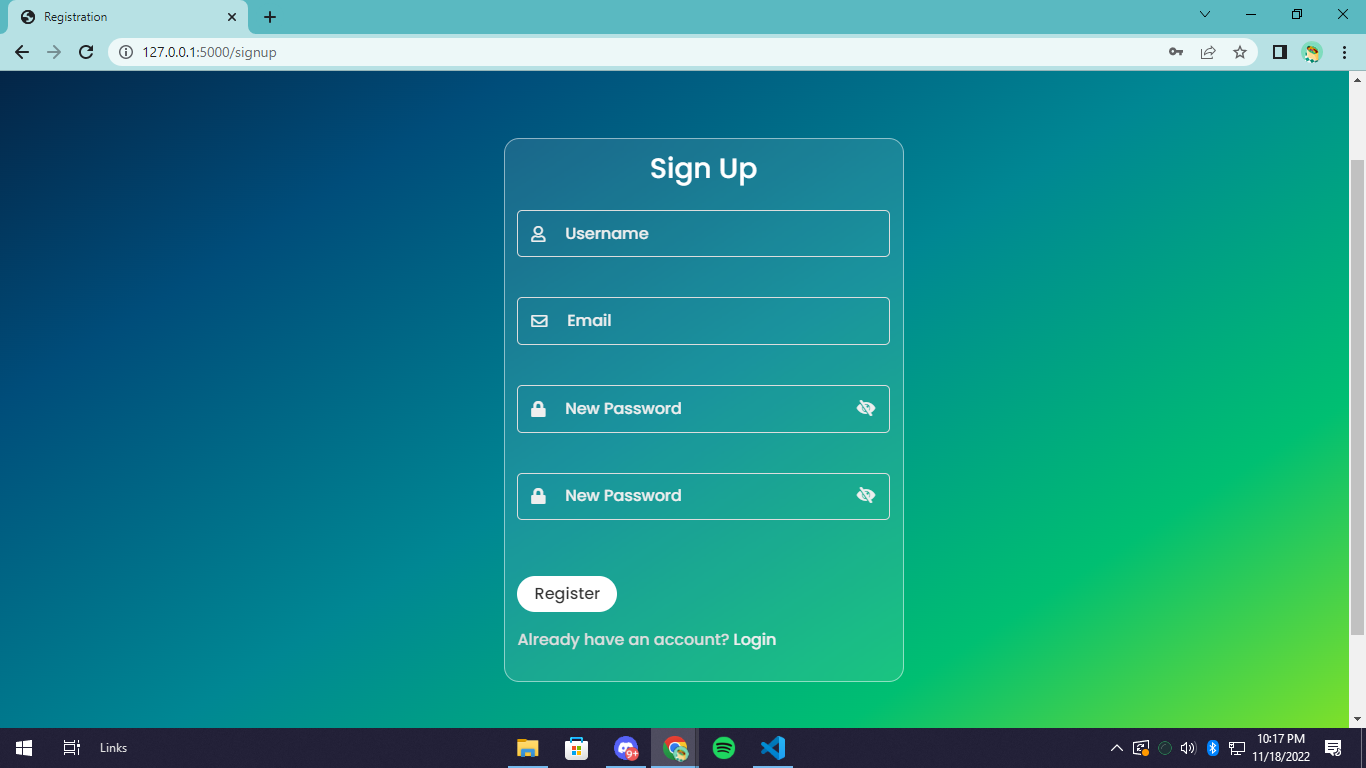
**Home Page**

****

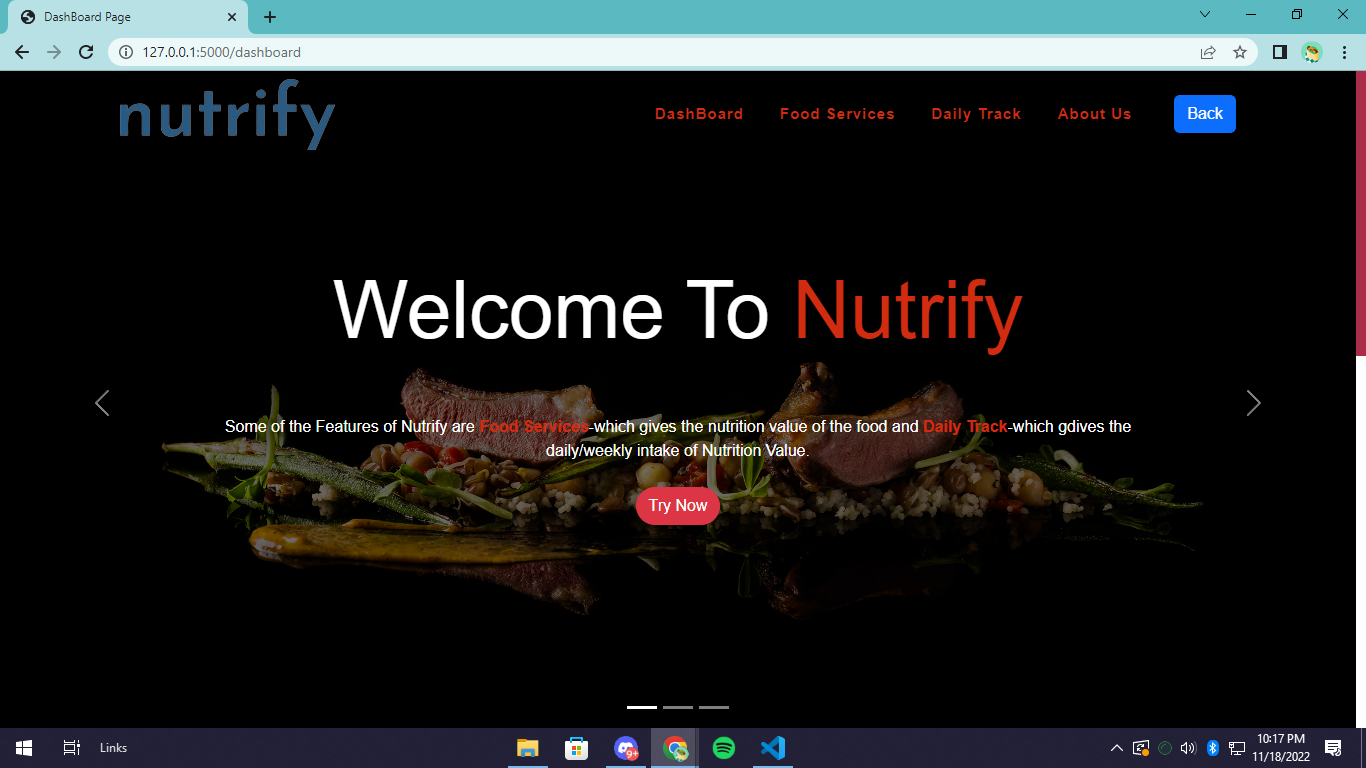
**Login Page**

****

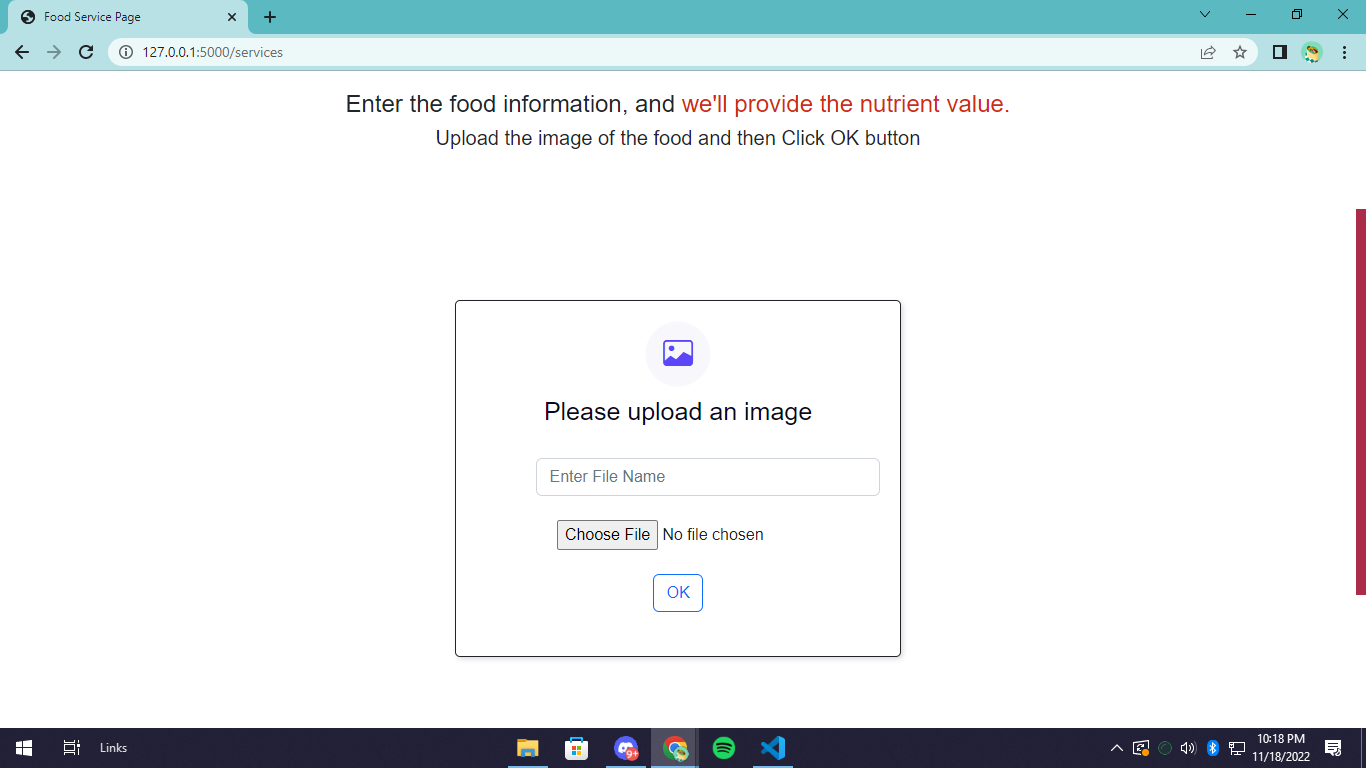
**Register Page**

****

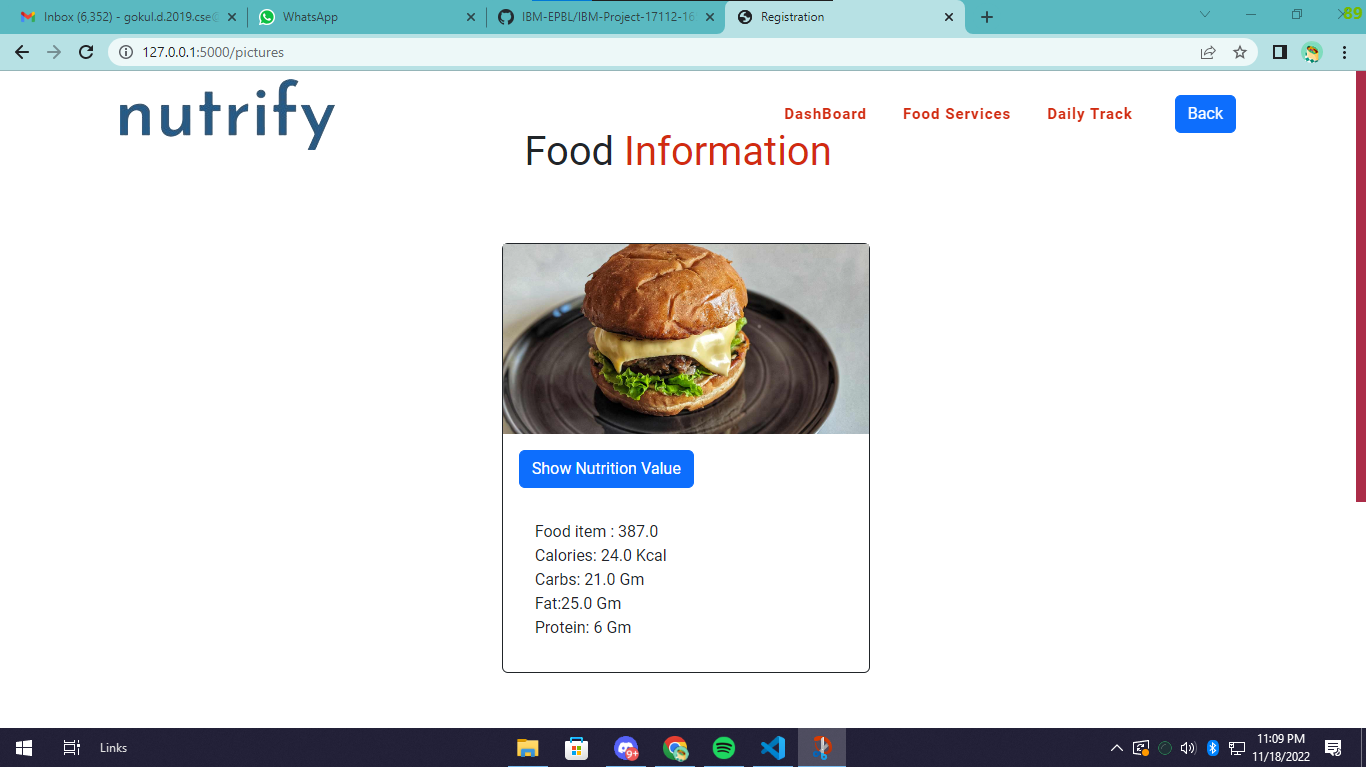
**Dashboard**

****

**Service Page**

****

**Food Details Page**

****

**10. ADVANTAGES & DISADVANTAGES**

**ADVANTAGES:**

* Low Energy Consumption.
* Works Under Low Data Connection.
* User Friendly Web Application.
* Data Privacy.
* Easy to Understand.

**DISADVANTAGES:**

* It Cannot be Used Without Internet Connection.
* Usage of 3rd party API may cause the time delay.

**11. CONCLUSION**

* The efficient way of tracking a user’s diet and knowing the nutrient intake of any food is implemented using the nutrition assistant website that is hosted on IBM Cloud platform.
* To ensure the smooth functioning of the web site operation. I have hosted the website in IBM Db2 & Kubernetes Cluster to make sure the operations are running successfully Cloud lambda function is used and to deploy the application IBM Db2 service is used.

**12. FUTURE SCOPE**

* Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures to keep track of their nutrition value in their diet.
* Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime

**13. APPENDIXES**

**13.1 Sample Source Code**

from flask import Flask, render\_template, redirect, url\_for, request, flash, escape, session

from flask\_wtf import FlaskForm

from wtforms import StringField, PasswordField, EmailField

from wtforms.validators import InputRequired, Length, Email, EqualTo

import openapi\_client

from com.spoonacular import misc\_api

import testmail

import ibm\_db

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import os

from dotenv import load\_dotenv

load\_dotenv()

DB\_HOSTNAME = os.getenv("DB\_HOSTNAME")

DB\_PORT = os.getenv("DB\_PORT")

DB\_USERNAME = os.getenv("DB\_USERNAME")

DB\_PASS = os.getenv("DB\_PASS")

COS\_ENDPOINT = os.getenv("COS\_ENDPOINT")

COS\_API\_KEY\_ID = os.getenv("COS\_API\_KEY\_ID")

COS\_INSTANCE\_CRN = os.getenv("COS\_INSTANCE\_CRN")

cos = ibm\_boto3.resource("s3",

    ibm\_api\_key\_id=COS\_API\_KEY\_ID,

    ibm\_service\_instance\_id=COS\_INSTANCE\_CRN,

    config=Config(signature\_version="oauth"),

    endpoint\_url=COS\_ENDPOINT

)

conn = ibm\_db.connect(f"DATABASE=bludb;HOSTNAME={DB\_HOSTNAME};PORT={DB\_PORT};SECURITY=SSL;SSLServerCertificate=SSLCertificate.crt;UID={DB\_USERNAME};PWD={DB\_PASS}",'','')

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = os.getenv("SECERT\_KEY")

class LoginForm(FlaskForm):

    email = EmailField("email", validators=[InputRequired("Email is required"), Email()])

    password = PasswordField("password", validators=[InputRequired("Password is required")])

class RegisterForm(FlaskForm):

    username = StringField("username", validators=[InputRequired("Username is required")])

    email = EmailField("email", validators=[InputRequired("Email is required"), Email()])

    pass1 = PasswordField("pass1", validators=[InputRequired("Password is required"), EqualTo('pass2', message="Passwords must match"), Length(min=4, max=30, message="Length must be between 4 and 30")])

    pass2 = PasswordField("pass2")

class ForgetPassword(FlaskForm):

    email = EmailField("email", validators=[InputRequired("Email is required"), Email()])

@app.route("/")

def home():

    username = request.cookies.get('username')

    return render\_template("home.html", username=username)

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

def login():

    form = LoginForm()

    if request.method=='POST' and form.validate\_on\_submit():

        email = request.form['email']

        password = request.form['password']

        sql = f"SELECT \* FROM USERS WHERE EMAIL='{escape(email)}'"

        stmt = ibm\_db.exec\_immediate(conn, sql)

        dic = ibm\_db.fetch\_both(stmt)

        if not dic or password != dic['PASSWORD']:

            flash("Incorrect email or password", "error")

            return redirect(url\_for('login'))

        session['username'] =  dic['USERNAME']

        session['uid'] =  dic['UID']

        return redirect(url\_for('dashboard'))

    else:

        return render\_template("login.html", form=form)

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

def signup():

    form = RegisterForm()

    if request.method=='POST' and form.validate\_on\_submit():

        username = request.form['username']

        email = request.form['email']

        pass1 = request.form['pass1']

        sql = f"SELECT \* FROM USERS WHERE EMAIL='{escape(email)}'"

        stmt = ibm\_db.exec\_immediate(conn, sql)

        dic = ibm\_db.fetch\_both(stmt)

        if dic:

            flash("User with the email already exist", "error")

            return redirect(url\_for('login'))

        sql = "INSERT INTO USERS (USERNAME, EMAIL, PASSWORD) VALUES (?, ?, ?)"

        prep\_stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(prep\_stmt, 1, username)

        ibm\_db.bind\_param(prep\_stmt, 2, email)

        ibm\_db.bind\_param(prep\_stmt, 3, pass1)

        ibm\_db.execute(prep\_stmt)

        testmail.SendEmail(email, username)

        flash("Registration Successful", "success")

        response = redirect(url\_for('login'))

        return response

    else:

        return render\_template("register.html", form=form)

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

def forgot\_password():

    form = ForgetPassword()

    if request.method=='POST' and form.validate\_on\_submit():

        email = request.form['email']

        sql = f"SELECT \* FROM USERS WHERE EMAIL='{escape(email)}'"

        stmt = ibm\_db.exec\_immediate(conn, sql)

        dic = ibm\_db.fetch\_both(stmt)

        if dic:

            flash("Email has been sent if user exist", "success")

            return redirect(url\_for('forgot\_password'))

        return render\_template("forgot\_password.html", form=form)

    return render\_template("forgot\_password.html", form=form)

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

def logout():

    session.pop('username')

    session.pop('uid')

    return redirect(url\_for('home'))

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

def dashboard():

    if not 'username' in session:

        flash("Login to access pages", "error")

        return(redirect(url\_for('home')))

    return render\_template("dashboard.html")

def get\_item(bucket\_name, item\_name):

    print("Retrieving item from bucket: {0}, key: {1}".format(bucket\_name, item\_name))

    try:

        file = cos.Object(bucket\_name, item\_name).get()

        print("File Contents: {0}".format(file["Body"].read()))

    except ClientError as be:

        print("CLIENT ERROR: {0}\n".format(be))

    except Exception as e:

        print("Unable to retrieve file contents: {0}".format(e))

def get\_bucket\_contents(bucket\_name):

    print("Retrieving bucket contents from: {0}".format(bucket\_name))

    try:

        files = cos.Bucket(bucket\_name).objects.all()

        files\_names = []

        for file in files:

            files\_names.append(file.key)

            print("Item: {0} ({1} bytes).".format(file.key, file.size))

        return files\_names

    except ClientError as be:

        print("CLIENT ERROR: {0}\n".format(be))

    except Exception as e:

        print("Unable to retrieve bucket contents: {0}".format(e))

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

    try:

        print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

        # set 5 MB chunks

        part\_size = 1024 \* 1024 \* 5

        # set threadhold to 15 MB

        file\_threshold = 1024 \* 1024 \* 15

        # set the transfer threshold and chunk size

        transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

            multipart\_threshold=file\_threshold,

            multipart\_chunksize=part\_size

        )

        # the upload\_fileobj method will automatically execute a multi-part upload

        # in 5 MB chunks for all files over 15 MB

        with open(file\_path, "rb") as file\_data:

            cos.Object(bucket\_name, item\_name).upload\_fileobj(

                Fileobj=file\_data,

                Config=transfer\_config

            )

        print("Transfer for {0} Complete!\n".format(item\_name))

    except ClientError as be:

        print("CLIENT ERROR: {0}\n".format(be))

    except Exception as e:

        print("Unable to complete multi-part upload: {0}".format(e))

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

def upload():

    if not 'username' in session:

        flash("Login to access pages", "error")

        return(redirect(url\_for('home')))

    if request.method == 'POST':

        bucket=os.getenv("BUCKET\_NAME")

        f = request.files['file']

        filename = f.filename

        file\_path = os.path.join('static/uploads', filename)

        f.save(file\_path)

        multi\_part\_upload(bucket,filename,file\_path)

        uid = session['uid']

        sql = f"INSERT INTO imagedetails(img\_link, uid) VALUES(?, ?)"

        imagelink = f"https://flask-app-test.s3.jp-tok.cloud-object-storage.appdomain.cloud/{escape(filename)}"

        prep\_stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(prep\_stmt, 1, imagelink)

        ibm\_db.bind\_param(prep\_stmt, 2, uid)

        ibm\_db.execute(prep\_stmt)

        sql = f"SELECT ID FROM imagedetails WHERE img\_link='{escape(imagelink)}'"

        stmt = ibm\_db.exec\_immediate(conn, sql)

        image\_id = ibm\_db.fetch\_both(stmt)

        nutitionapi(imagelink,image\_id)

        return redirect(url\_for('pictures'))

    if request.method == 'GET':

        return render\_template('foodservices.html')

def nutitionapi(imagelink,image\_id):

    configuration = openapi\_client.Configuration(

        host = "https://api.spoonacular.com"

    )

    configuration.api\_key['apiKeyScheme'] = os.getenv("NUTRITION\_API\_KEY")

    with openapi\_client.ApiClient(configuration) as api\_client:

        api\_instance = misc\_api.MiscApi(api\_client)

        image\_url = imagelink

    try:

        api\_response = api\_instance.image\_analysis\_by\_url(image\_url)

        y = api\_response

        cal= y["nutrition"]["calories"]["value"]

        Carb= y["nutrition"]["carbs"]["value"]

        fat= y["nutrition"]["fat"]["value"]

        protein= y["nutrition"]["protein"]["value"]

        name=y["category"]["name"]

        imageid=image\_id["ID"]

        sql = f"INSERT INTO nutritiondetails(calories,carbs,fat,protein,ref\_id,name) VALUES('{escape(cal)}','{escape(Carb)}','{escape(fat)}','{escape(protein)}','{escape(imageid)}','{escape(name)}')"

        prep\_stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.execute(prep\_stmt)

        flash("Successful db operation", "success")

    except openapi\_client.ApiException as e:

        print("Exception when calling MiscApi->image\_analysis\_by\_url: %s\n" % e)

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

def pictures():

    if not 'username' in session:

        flash("Login to access pages", "error")

        return(redirect(url\_for('home')))

    uid = session['uid']

    sql = f"SELECT \* FROM imagedetails where imagedetails.uid='{escape(uid)}'"

    stmt = ibm\_db.exec\_immediate(conn, sql)

    pic = ibm\_db.fetch\_both(stmt)

    pics=[]

    while pic != False:

        x=[pic["ID"], pic["IMG\_LINK"]]

        pics.append(x)

        pic = ibm\_db.fetch\_both(stmt)

    x = []

    for i in pics:

        sql = f"SELECT \* FROM nutritiondetails,imagedetails where nutritiondetails.ref\_id=imagedetails.id and nutritiondetails.ref\_id  = {escape(i[0])}"

        print(i[0])

        stmt = ibm\_db.exec\_immediate(conn, sql)

        pic = ibm\_db.fetch\_both(stmt)

        x.append(pic)

    print(x)

    return render\_template('storage.html', foods = x)

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

    app.run(debug=True)

**13.2 GITHUB**

<https://github.com/IBM-EPBL/IBM-Project-17112-1659628423>

**DEMO VIDEO:**

<https://drive.google.com/file/d/1FqE3NqrQQZLvwlyCotqAezNMo-bke0pi/view?usp=sharing>

**LIVE APP:**

http://169.51.204.41:31301/