GOVERNMENT COLLEGE OF ENGINEERING
(Formerly IRTT)
ERODE-638 316

BONAFIDE CERTIFICATE

Certified that this project titled "**NUTRITION ASSISTANT APPLICATION**" is the bonafide work of "**ABDUL SIKKANTHAR A (731119205001), AL HAMEED FARVAZ S (731119205002), EMAYAKEERTHI N (731119205008), RITHIK S (731119205038)**" who carried out the project work under my supervision.

SIGNATURE OF HOD
Dr.P.KALYANI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF IT,
GOVERNMENT COLLEGE OF
ENGINEERING,ERODE-638316

SIGNATURE OF SPOC
Dr.G.GOWRISON, M.E.,Ph.D.,
ASSISTANT PROFESSOR(SR)
DEPARTMENT OF ECE,
GOVERNMENT COLLEGE OF
ENGINEERING,ERODE – 638316

SIGNATURE OF FACULTY MENTOR
Dr.I.BHUVANESHWARRI,M.E.,Ph.D.,
ASSISTANT PROFESSOR(SR)
DEPARTMENT OF IT,
GOVERNMENT COLLEGE OF
ENGINEERING,ERODE-638316

SIGNATURE OF FACULTY EVALUATOR
Dr.S.MOHANASUNDARAM,M.TECH.,Ph.D.,
ASSISTANT PROFESSOR(SR)
DEPARTMENT OF IT,
GOVERNMENT COLLEGE OF
ENGINEERING, ERODE – 638316

Project Report Format

NUTRITION ASSISTANT APPLICATION

1.INTRODUCTION

1.1 PROJECT OVERVIEW

- 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 analyze the images and Nutrition API to provide nutritional information about the analyzed Image.
- Nutritional information of the analyzed 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 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.

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. LITERATURE SURVEY

2.1 EXISTING PROBLEM

In this existing world, there are no any proper handy nutrition assistant which helps people to maintain their health and fitness.

2.2 REFERENCES

- ➤ Enhancing Cloud and healthy Food Nutrition Information systems practice- Paul,PK and Aithal,PS and Bhuimali,A
- ➤ Mobile cloud based system recognizing nutrition and freshness of food image- Kumbhar, Diptee and Patil,Sarita
- ➤ Predicting calorific value for mixed food using image processing- Kohila, R and Meenakumari, R
- ➤ Use of artificial intelligence in precision nutrition and fitness- de Moraes Lopes, Maria Helena Baena and Ferreira, Danton Diego and Ferreira, Ana Claudia Barbosa Honorio and da Silva, Giuliano Roberto and Caetano, Aletha Silva and Braz.

2.3 PROBLEM STATEMENT DEFINITION

1.The user who wants to maintain a healthy and fit body but has no one to guide them on their diet.

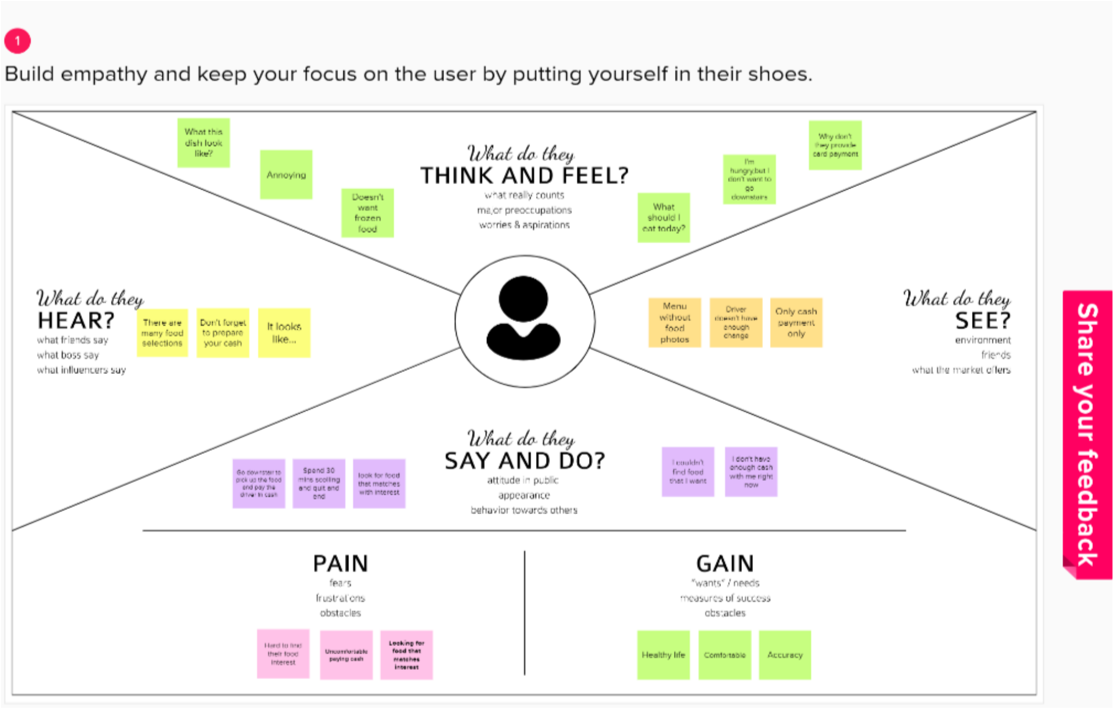
2.The user wants to develop a deep learning model that basically help athlete, body builders or other game players to keep up with their health and fitness by suggesting them with proper
nutrition plan.

3.The user wants an AI software to maintain a healthy diet rather than having a physical consultant.

4.People who are obese and overweight are more likely to have high-risk factors for heart disease, diabetes,hypertension.

3. IDEATION AND PROPOSED SOLUTION

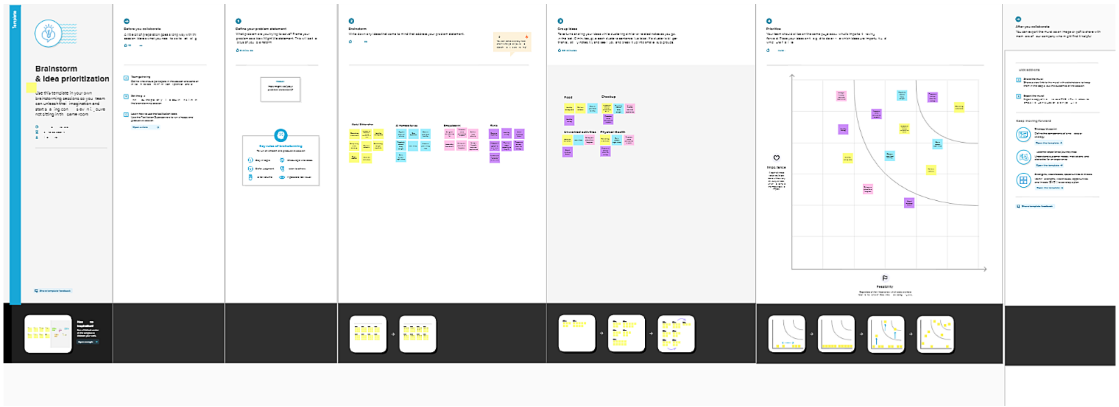
3.1 EMPATHY MAP CANVAS



1

Build empathy and keep your focus on the user by putting yourself in their shoes.

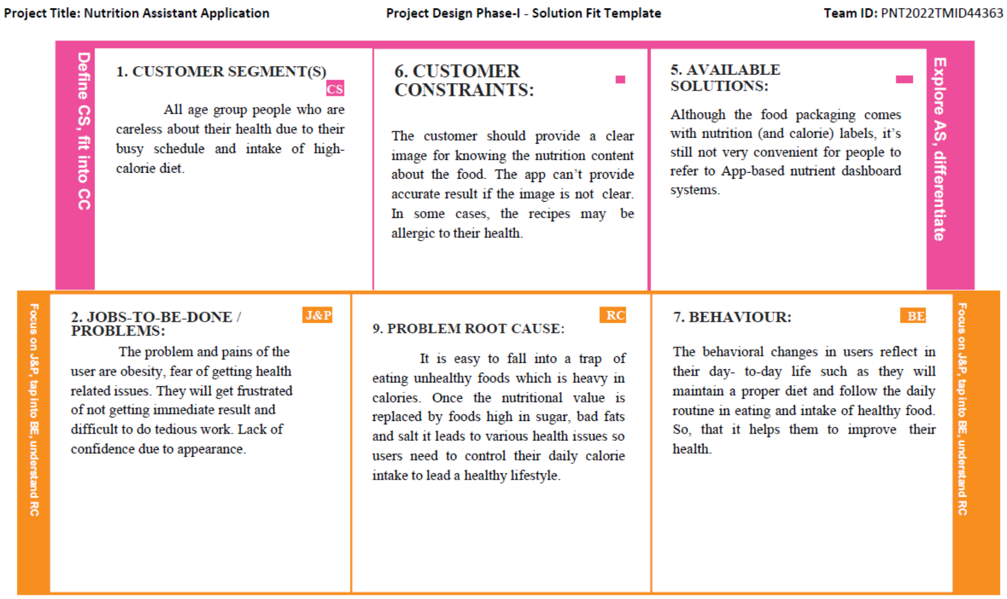
3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

S.NO.	PARAMETER	DESCRIPTION
1.	Problem Statement (Problem to be solved)	Due to improvement in people standards of living, obesity rates are increasing at an alarming speed, and this is reflected to the risk in health. People need to control their daily calories intake by eating healthy food,to avoid obesity.
2.	Idea / Solution description By creating an application,we can recommend diet plans for the users and measure sugar level ,BP level.	By creating an application,we can recommend diet plans for the users and measure sugar level ,BP level.
3.	Novelty / Uniqueness	It can realize real time images of meal and and analyze it for nutritional content can be handy and improve dietary habit
4.	Social Impact / Customer Satisfaction	It will help people with providing proper nutrition and helps in maintaining a healthy lifestyle.
5.	Business Model (Revenue Model)	Social media is the best way to spread the word about our application and with influencers we can attract normal people.
6.	Scalability of the Solution	Different diet charts can be planned for different aspects of people.

3.4 PROBLEM SOLUTION OUTFIT



Project Title: Nutrition Assistant Application

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID44363

Define CS, fit into CC

Explore AS, differentiate

1. CUSTOMER SEGMENT(S) CS All age group people who are careless about their health due to their busy schedule and intake of high-calorie diet.	**6. CUSTOMER CONSTRAINTS:** The customer should provide a clear image for knowing the nutrition content about the food. The app can't provide accurate result if the image is not clear. In some cases, the recipes may be allergic to their health.	**5. AVAILABLE SOLUTIONS:** Although the food packaging comes with nutrition (and calorie) labels, it's still not very convenient for people to refer to App-based nutrient dashboard systems.
2. JOBS-TO-BE-DONE / PROBLEMS: J&P The problem and pains of the user are obesity, fear of getting health related issues. They will get frustrated of not getting immediate result and difficult to do tedious work. Lack of confidence due to appearance.	**9. PROBLEM ROOT CAUSE:** RC 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.	**7. BEHAVIOUR:** BE The behavioral changes in users reflect in their day- to-day life such as they will maintain a proper diet and follow the daily routine in eating and intake of healthy food. So, that it helps them to improve their health.

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

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

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

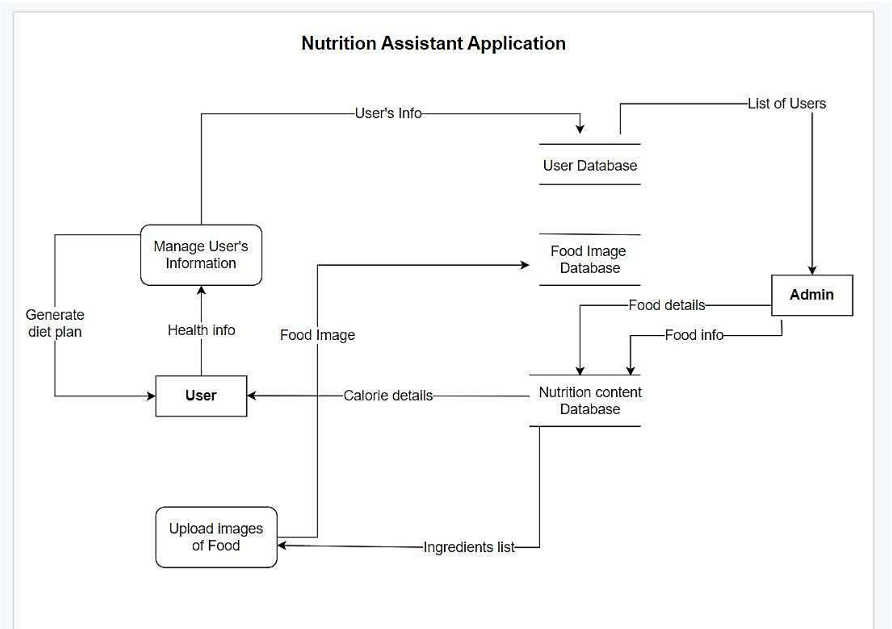
This project is aimed at developing a desktop-based application named Nutrition Assistant Application for estimates food attributes such as ingredients and nutritional value by classifying the input images of food. The Nutrition Assistant Application refers to the system and processes to help the user to analyse 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 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, and 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.

4.2 NON-FUNCTIONAL REQUIREMENTS

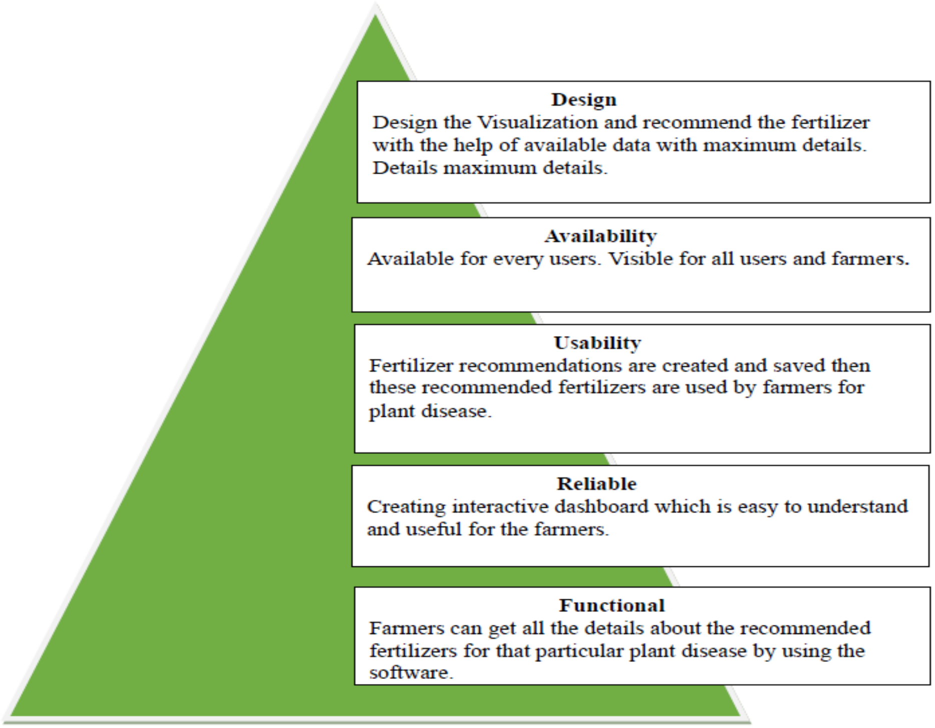
- **Maintains good health**: The application can help in guiding them on how to remain healthy and how to take good nutrition. The application will help them without personally going to the doctor. Promote better nutrition in the community by educating about better diet and nutrition.
- **Functional limitation:** The user to be specific can’t access the web or admin module, whereas the administrator has all the rights to modify and manage the contents such as news, tips, etc
- **Improve Usability:** In the part of user’s just the internet connection is enough in order to access the news, updates and other contents provided by the admin regarding their health condition.
- **Health conscious:** This will provide convenience to persons/users who wants to learn about nutrition and other related health topics by just using the Nutrition Assistant Application

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE



Design
Design the Visualization and recommend the fertilizer with the help of available data with maximum details. Details maximum details.

Availability
Available for every users. Visible for all users and farmers.

Usability
Fertilizer recommendations are created and saved then these recommended fertilizers are used by farmers for plant disease.

Reliable
Creating interactive dashboard which is easy to understand and useful for the farmers.

Functional
Farmers can get all the details about the recommended fertilizers for that particular plant disease by using the software.

5.3 USER STORIES

user type	Functional Requirements (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	I can access my account / dashboard	High	Sprint-1

			confirming my password.			
		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 user, I can register for the application through G-mail	I can receive confirmation email & click confirm to login	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
		USN-5	If I forgot my password or username, I can reset it again through my email	I can receive reset Mail to the registered Email Id	High	Sprint-2
Customer (Web user)	Registration	USN-6	As a user, I can register by entering my email, password, and confirming my password	I can access my account / dashboard	High	Sprint-2
		USN-7	As a user, I	I can receive		Sprin

			will receive confirmation email once I have registered for the application	confirmation email & click confirm	High	t-2
		USN-8	As a user, I can register for the application through G-mail	I can receive confirmation email & click confirm to login	Medium	Sprint-2
		USN-9	As a user, I can log into the application by entering email & password		High	Sprint-2

6.PROJECT PLANNING AND SHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional requirements(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USDN-1	As a user, I can register for the application by entering the name, mail, and confirming my password.	2	High	Abdul Sikkanthar A, Al Hameed Farvaz S, Emayakeerthi N, Rithik S
Sprint-1		USDN-2	As a user, I will receive confirmation email once	1	High	Abdul Sikkanthar A, Al Hameed

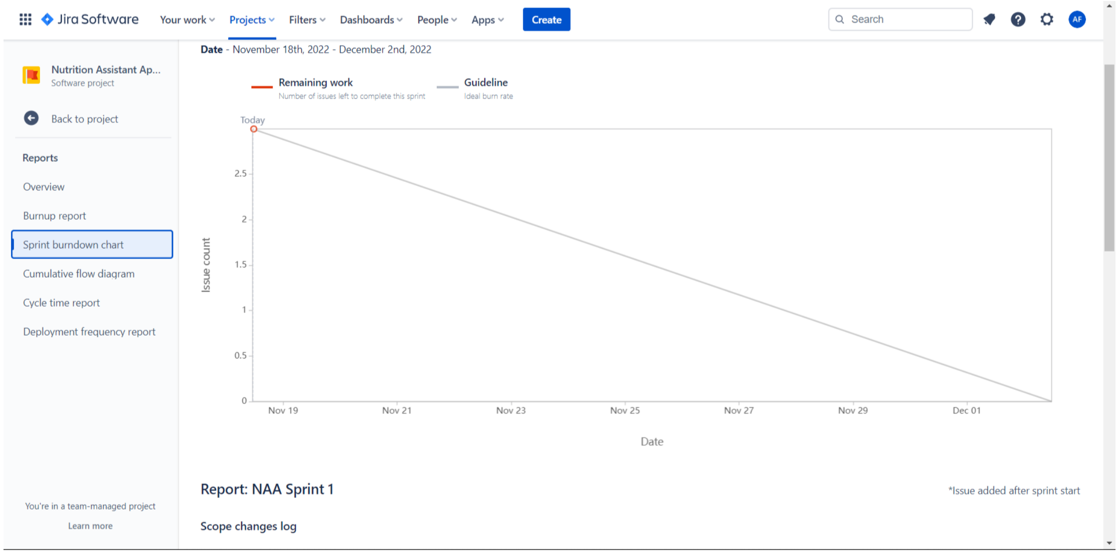
			I have registered for the application			Farvaz S, Emayakeerthi N, Rithik S
Sprint-1	Login	USDN-3	As a user, I can log into the application by entering email & password	1	High	Abdul Sikkanthar A, Al Hameed Farvaz S, Emayakeerthi N, Rithik S
Sprint-2	User details	USDN-4	As a user , I can fill the Details.	2	High	Abdul Sikkanthar A, Al Hameed Farvaz S, Emayakeerthi N, Rithik S
Sprint-3	Push notification	USDN-5	As a user, I will search the food items	2	Medium	Abdul Sikkanthar A, Al Hameed Farvaz S, Emayakeerthi N, Rithik S
Sprint-4	Shown the nutrition and Recipe	USDN-6	As a user, I can scan the food an get the details and Recipe for nutrition details and recipe for related scanned	1	High	Abdul Sikkanthar A, Al Hameed Farvaz S, Emayakeerthi N, Rithik S

6.2 SPRINT DELIVERY SHEDULE

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

6.3 REPORTS FROM JIRA

SPRINT BURN DOWN CHART



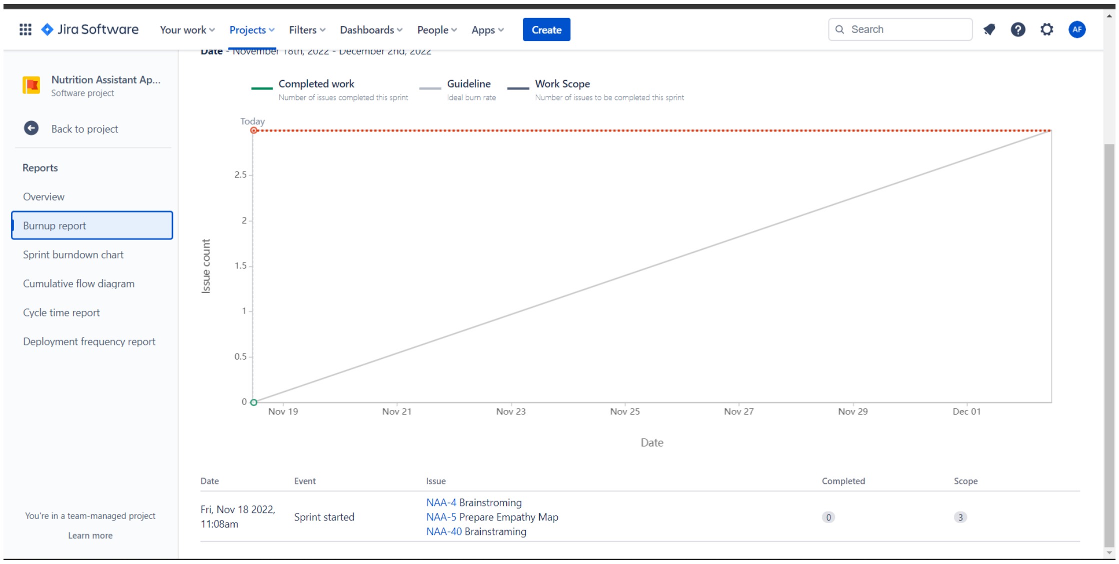
Date - November 18th, 2022 - December 2nd, 2022

Report: NAA Sprint 1

*Issue added after sprint start

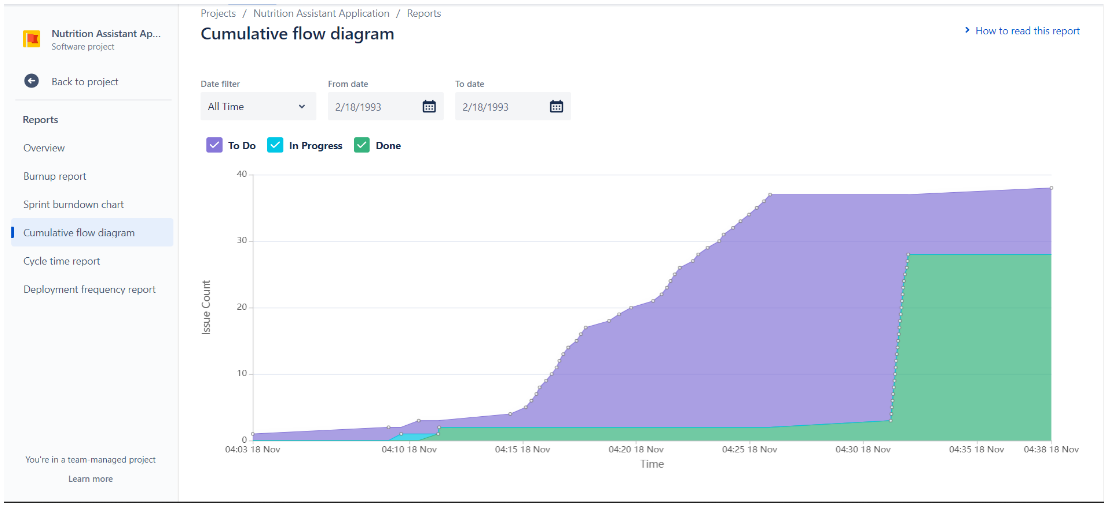
Scope changes log

BURNUP REPORT

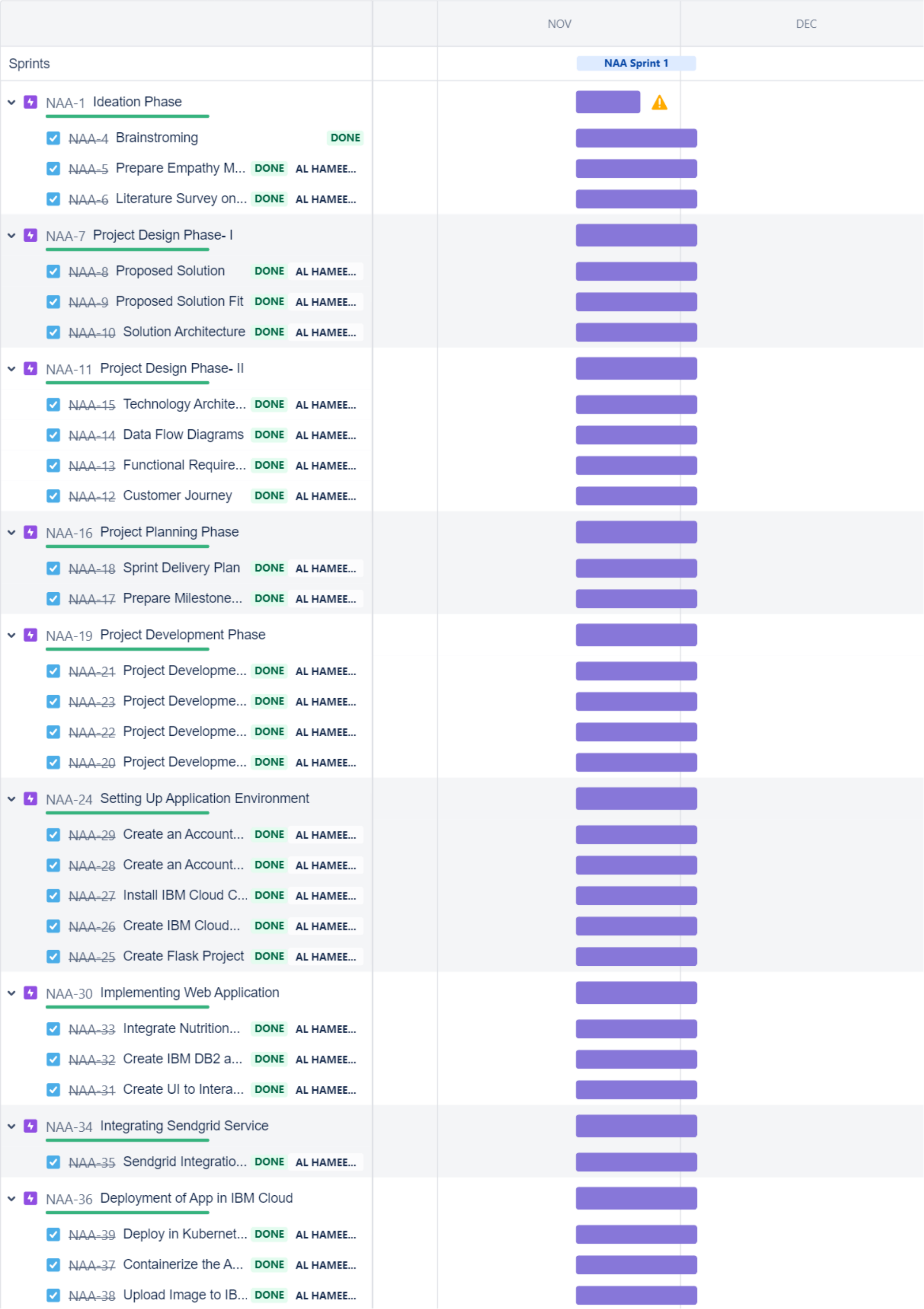


Date	Event	Issue	Completed	Scope
Fri, Nov 18 2022, 11:08am	Sprint started	NAA-4 Brainstroming NAA-5 Prepare Empathy Map NAA-40 Brainstraming	0	3

CUMMULATIVE FLOW DIAGRAM



ROAD MAP

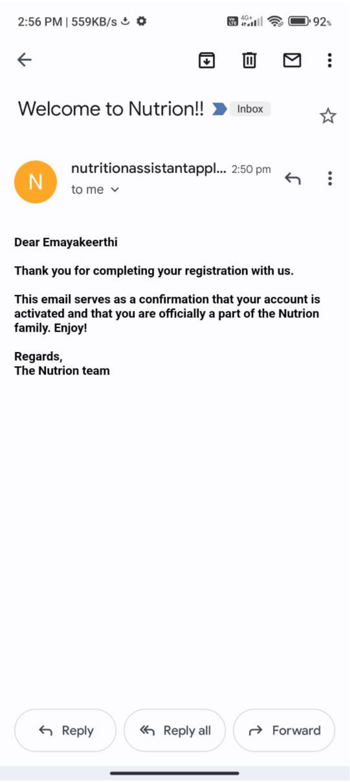


			NOV	DEC
Sprints			NAA Sprint 1	
NAA-1 Ideation Phase			⚠	
☑ ~~NAA-4~~ Brainstroming	DONE			
☑ ~~NAA-5~~ Prepare Empathy M...	DONE	AL HAMEE...		
☑ ~~NAA-6~~ Literature Survey on...	DONE	AL HAMEE...		
NAA-7 Project Design Phase- I				
☑ ~~NAA-8~~ Proposed Solution	DONE	AL HAMEE...		
☑ ~~NAA-9~~ Proposed Solution Fit	DONE	AL HAMEE...		
☑ ~~NAA-10~~ Solution Architecture	DONE	AL HAMEE...		
NAA-11 Project Design Phase- II				
☑ ~~NAA-15~~ Technology Archite...	DONE	AL HAMEE...		
☑ ~~NAA-14~~ Data Flow Diagrams	DONE	AL HAMEE...		
☑ ~~NAA-13~~ Functional Require...	DONE	AL HAMEE...		
☑ ~~NAA-12~~ Customer Journey	DONE	AL HAMEE...		
NAA-16 Project Planning Phase				
☑ ~~NAA-18~~ Sprint Delivery Plan	DONE	AL HAMEE...		
☑ ~~NAA-17~~ Prepare Milestone...	DONE	AL HAMEE...		
NAA-19 Project Development Phase				
☑ ~~NAA-21~~ Project Developme...	DONE	AL HAMEE...		
☑ ~~NAA-23~~ Project Developme...	DONE	AL HAMEE...		
☑ ~~NAA-22~~ Project Developme...	DONE	AL HAMEE...		
☑ ~~NAA-20~~ Project Developme...	DONE	AL HAMEE...		
NAA-24 Setting Up Application Environment				
☑ ~~NAA-29~~ Create an Account...	DONE	AL HAMEE...		
☑ ~~NAA-28~~ Create an Account...	DONE	AL HAMEE...		
☑ ~~NAA-27~~ Install IBM Cloud C...	DONE	AL HAMEE...		
☑ ~~NAA-26~~ Create IBM Cloud...	DONE	AL HAMEE...		
☑ ~~NAA-25~~ Create Flask Project	DONE	AL HAMEE...		
NAA-30 Implementing Web Application				
☑ ~~NAA-33~~ Integrate Nutrition...	DONE	AL HAMEE...		
☑ ~~NAA-32~~ Create IBM DB2 a...	DONE	AL HAMEE...		
☑ ~~NAA-31~~ Create UI to Intera...	DONE	AL HAMEE...		
NAA-34 Integrating Sendgrid Service				
☑ ~~NAA-35~~ Sendgrid Integratio...	DONE	AL HAMEE...		
NAA-36 Deployment of App in IBM Cloud				
☑ ~~NAA-39~~ Deploy in Kubernet...	DONE	AL HAMEE...		
☑ ~~NAA-37~~ Containerize the A...	DONE	AL HAMEE...		
☑ ~~NAA-38~~ Upload Image to IB...	DONE	AL HAMEE...		

7.CODING & SOLUTIONING

7.1 Feature 1

We incorporated an email service. This service sends email messages directly to customers' inboxes.



Welcome to Nutrion!!

nutritionassistantappl... 2:50 pm

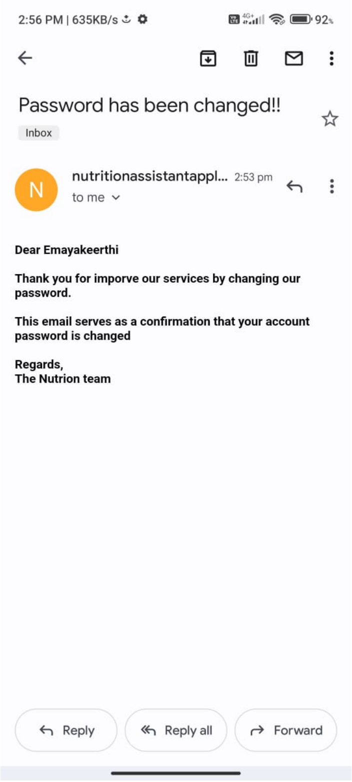
to me

Dear Emayakeerthi

Thank you for completing your registration with us.

This email serves as a confirmation that your account is activated and that you are officially a part of the Nutrion family. Enjoy!

Regards,
The Nutrion team



Password has been changed!!

nutritionassistantappl... 2:53 pm

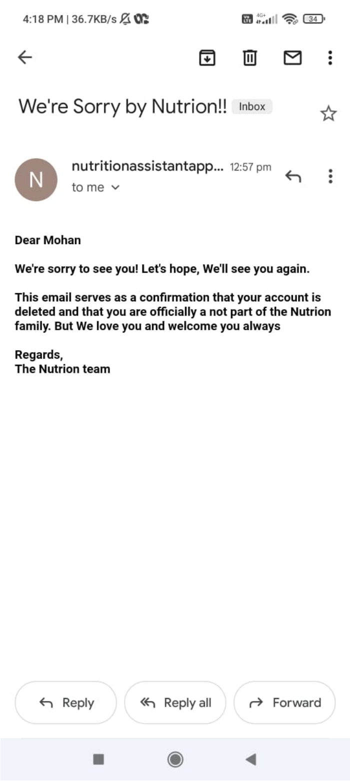
to me

Dear Emayakeerthi

Thank you for imporve our services by changing our password.

This email serves as a confirmation that your account password is changed

Regards,
The Nutrion team



We're Sorry by Nutrion!!

nutritionassistantapp... 12:57 pm

to me

Dear Mohan

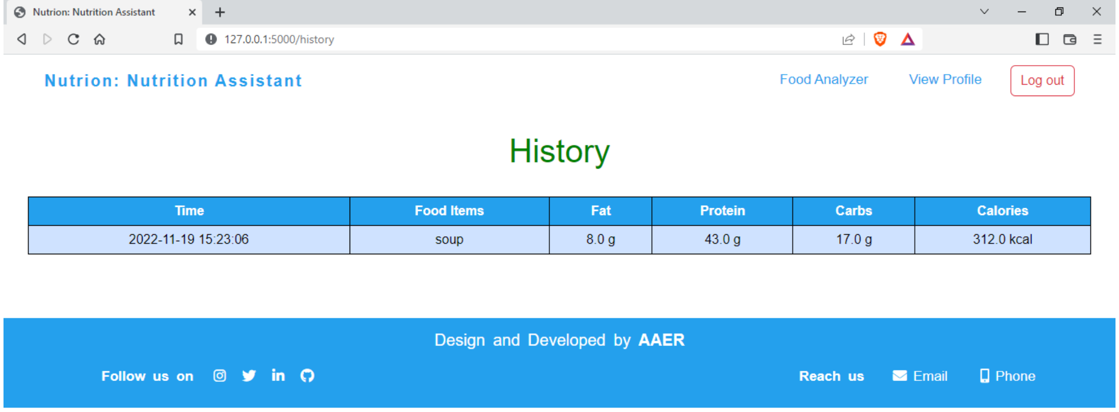
We're sorry to see you! Let's hope, We'll see you again.

This email serves as a confirmation that your account is deleted and that you are officially a not part of the Nutrion family. But We love you and welcome you always

Regards,
The Nutrion team

7.2 Feature 2

We store the nutrition-related information on the database, so users can access the data when they need it. Adding result into database.



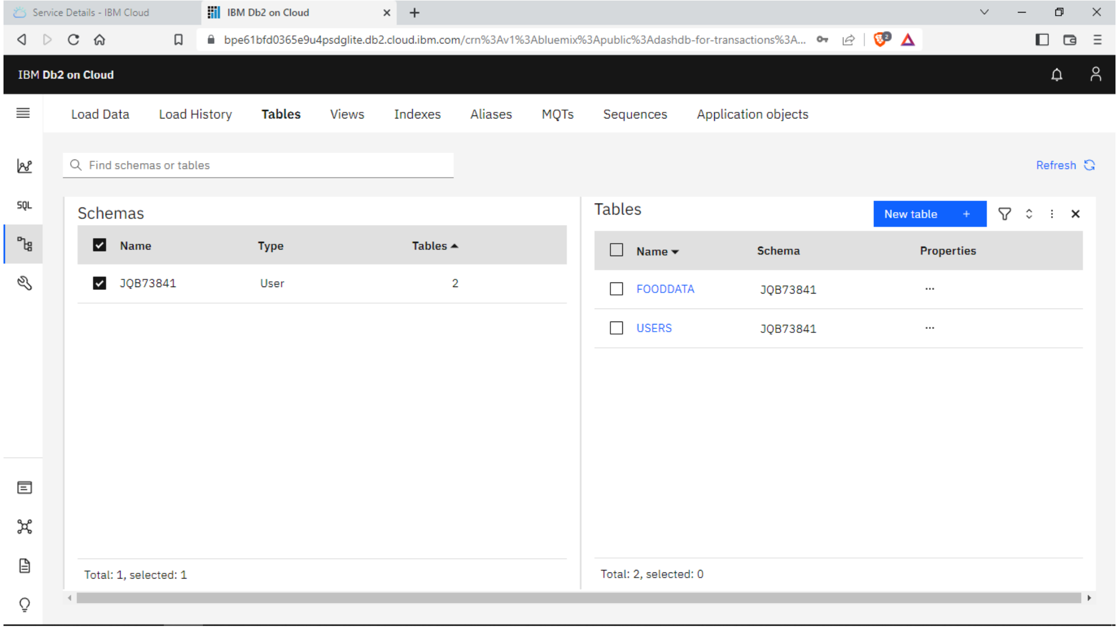
Nutrion: Nutrition Assistant
127.0.0.1:5000/history
Nutrion: Nutrition Assistant
Food Analyzer
View Profile
Log out

History

Time	Food Items	Fat	Protein	Carbs	Calories
2022-11-19 15:23:06	soup	8.0 g	43.0 g	17.0 g	312.0 kcal

Design and Developed by **AAER**
Follow us on
Reach us
Email
Phone

7.3 Database Schema



Schemas

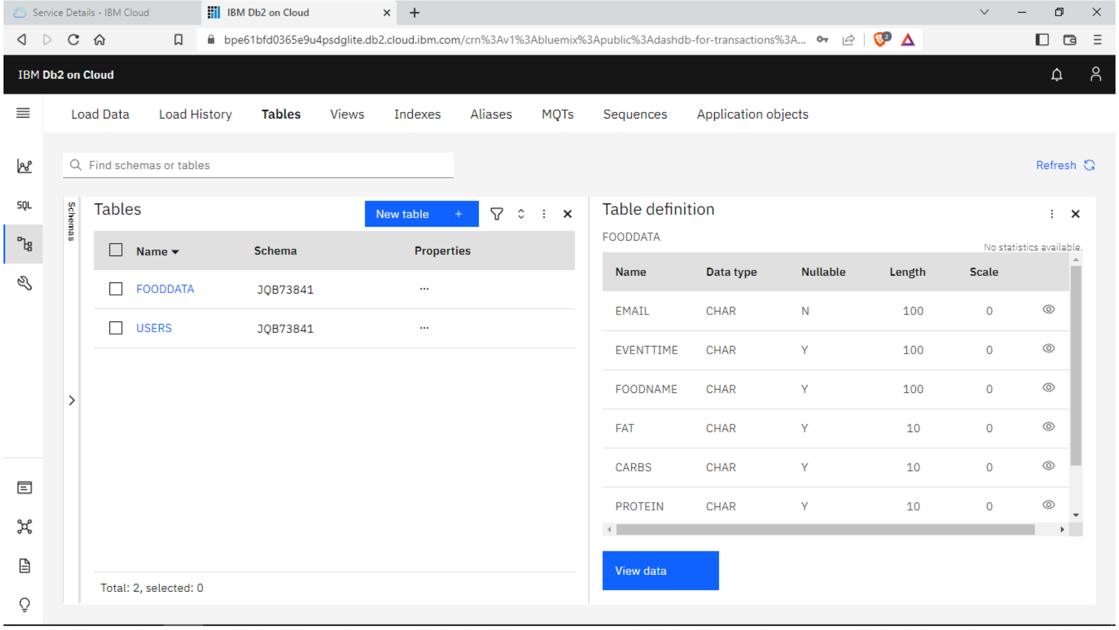
☑ Name	Type	Tables ▲
☑ JQB73841	User	2

Total: 1, selected: 1

Tables

□ Name ▾	Schema	Properties
□ FOODDATA	JQB73841	···
□ USERS	JQB73841	···

Total: 2, selected: 0



Tables

□ Name ▾	Schema	Properties
□ FOODDATA	JQB73841	···
□ USERS	JQB73841	···

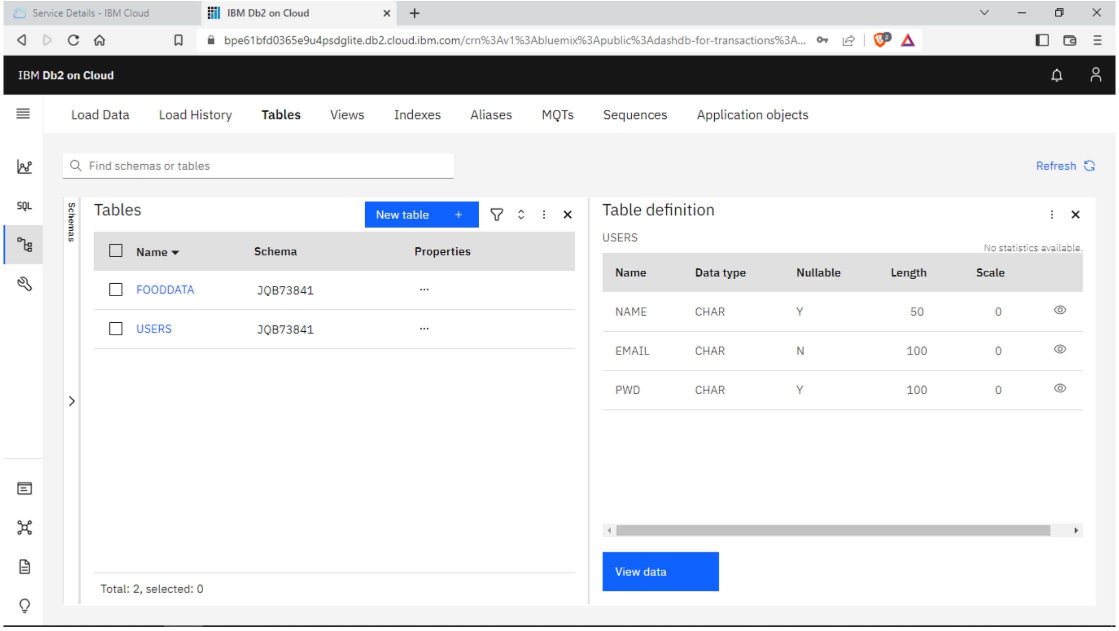
Total: 2, selected: 0

Table definition

FOODDATA

No statistics available.

Name	Data type	Nullable	Length	Scale
EMAIL	CHAR	N	100	0
EVENTTIME	CHAR	Y	100	0
FOODNAME	CHAR	Y	100	0
FAT	CHAR	Y	10	0
CARBS	CHAR	Y	10	0
PROTEIN	CHAR	Y	10	0



Tables

Name	Schema	Properties
FOODDATA	JQB73841	···
USERS	JQB73841	···

Total: 2, selected: 0

Table definition

USERS

No statistics available.

Name	Data type	Nullable	Length	Scale
NAME	CHAR	Y	50	0
EMAIL	CHAR	N	100	0
PWD	CHAR	Y	100	0

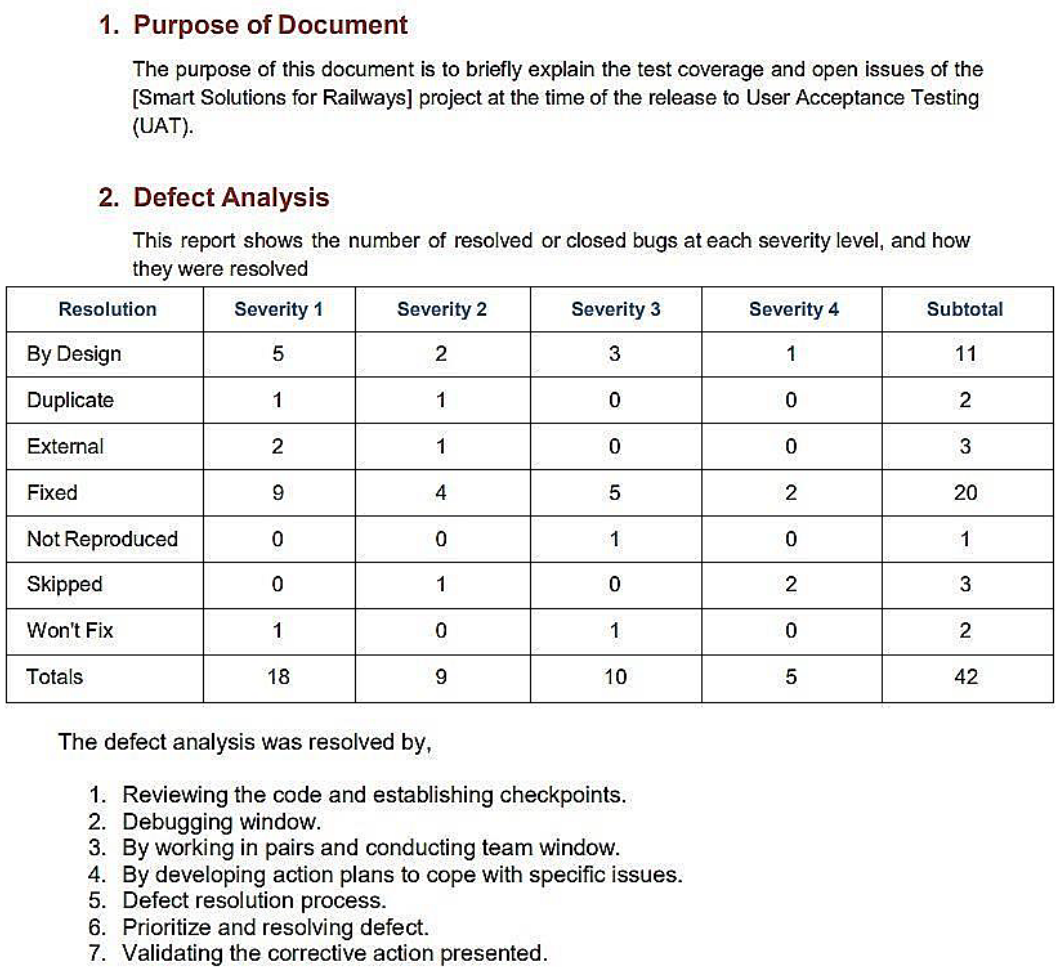
8. TESTING

8.1 Test Cases

```
import unittest
try: from app import app
except Exception as e:
        print('Some modules missing {}'.format(e))

class FlaskTest(unittest.TestCase):
         # check if response is 200 def test_index(self):
         tester = app.test_client(self)
        response = tester.get("/")
        statuscode = response.status_code self.assertEqual(statuscode, 200)
```

```
 # check content type def test_index_content(self):
tester = app.test_client(self)
response = tester.get("/")
self.assertEqual(response.content_type, 'text/html; charset=utf-8')

def test_register(self):
        tester = app.test_client(self)
        response = tester.post('/register', data=dict(email='username',
password='password'), follow_redirects=True)
        self.assertTrue(b'email' in response.data)

         # check log in def test_login(self):
        tester = app.test_client(self)
        response = tester.post('/', data=dict(email='username',password='password'),
follow_redirects=True) self.assertTrue(b'email' in response.data)

        # checking forgot function def test_forgot(self): tester = app.test_client(self)
        response = tester.post('/', data=dict(email='username'), follow_redirects=True)
self.assertTrue(b'email' in response.data)

 if __name__ == '__main__':
         unittest.main()
```

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 [Smart Solutions for Railways] 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 were resolved

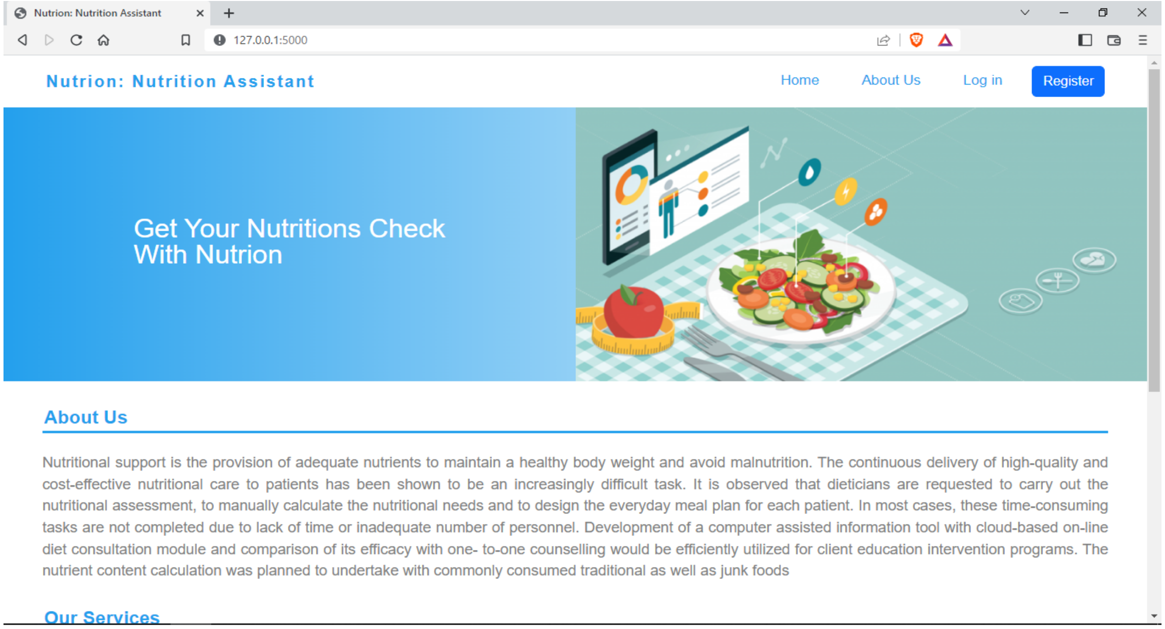
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	2	3	1	11
Duplicate	1	1	0	0	2
External	2	1	0	0	3
Fixed	9	4	5	2	20
Not Reproduced	0	0	1	0	1
Skipped	0	1	0	2	3
Won't Fix	1	0	1	0	2
Totals	18	9	10	5	42

The defect analysis was resolved by,

1. Reviewing the code and establishing checkpoints.
2. Debugging window.
3. By working in pairs and conducting team window.
4. By developing action plans to cope with specific issues.
5. Defect resolution process.
6. Prioritize and resolving defect.
7. Validating the corrective action presented.

9. RESULTS

9.1 Performance Metrics



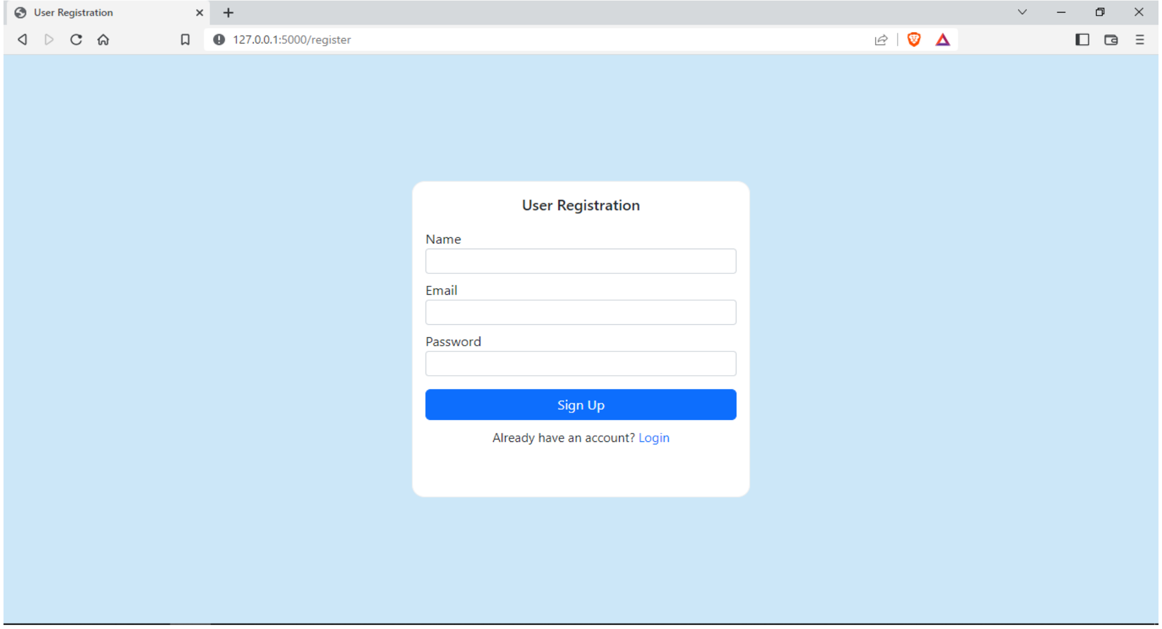
Nutrion: Nutrition Assistant

Get Your Nutritions Check With Nutrion

About Us

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

Our Services



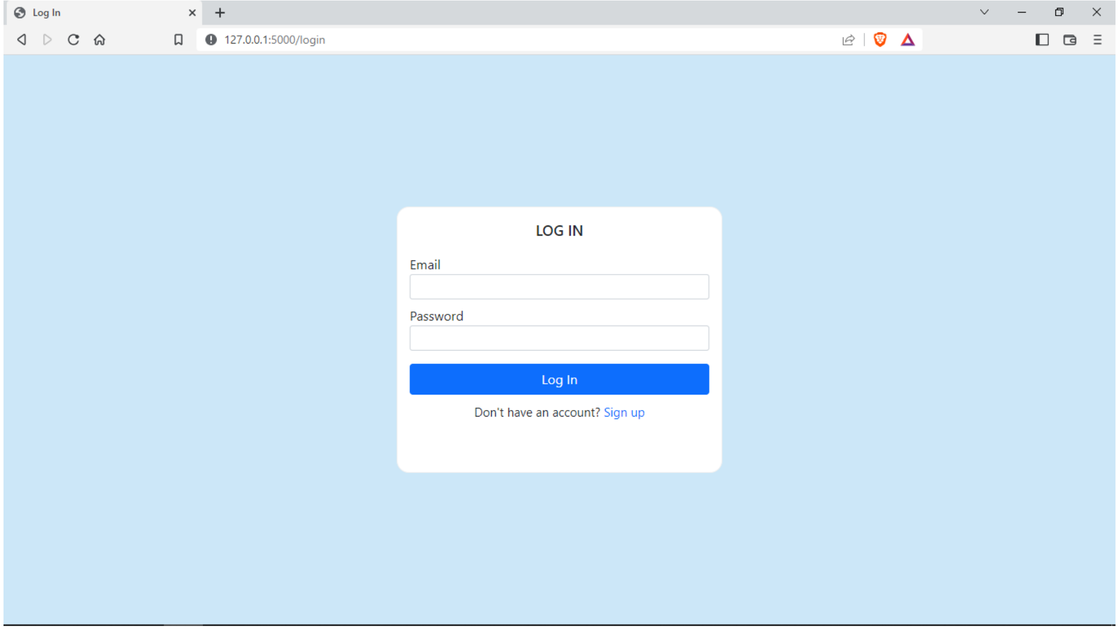
User Registration

Name

Email

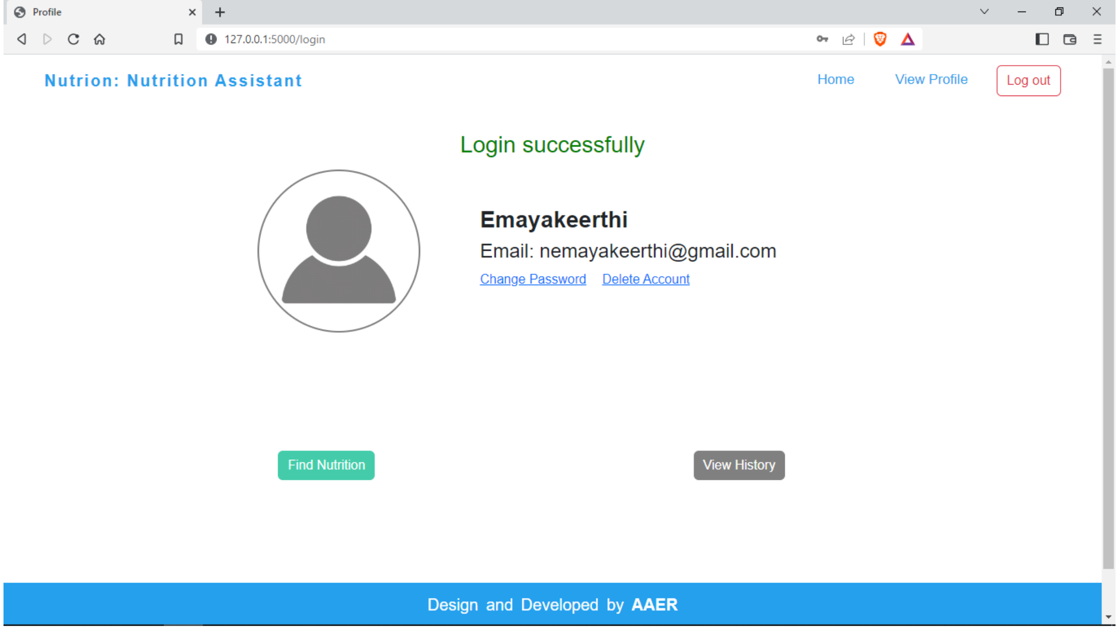
Password

Already have an account? Login



LOG IN

Don't have an account? Sign up



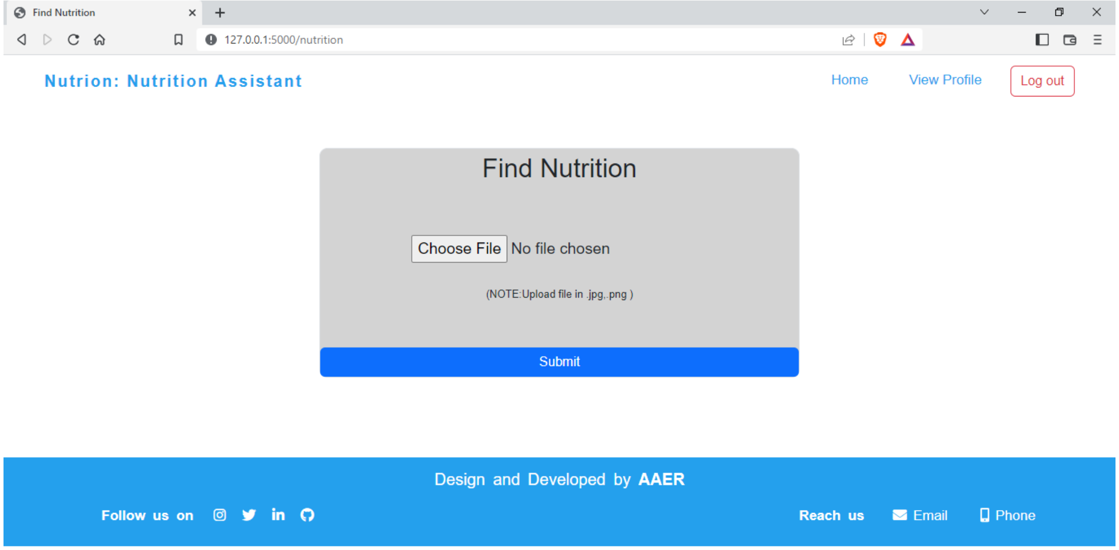
Nutrion: Nutrition Assistant

Login successfully

Emayakeerthi

Email: nemayakeerthi@gmail.com

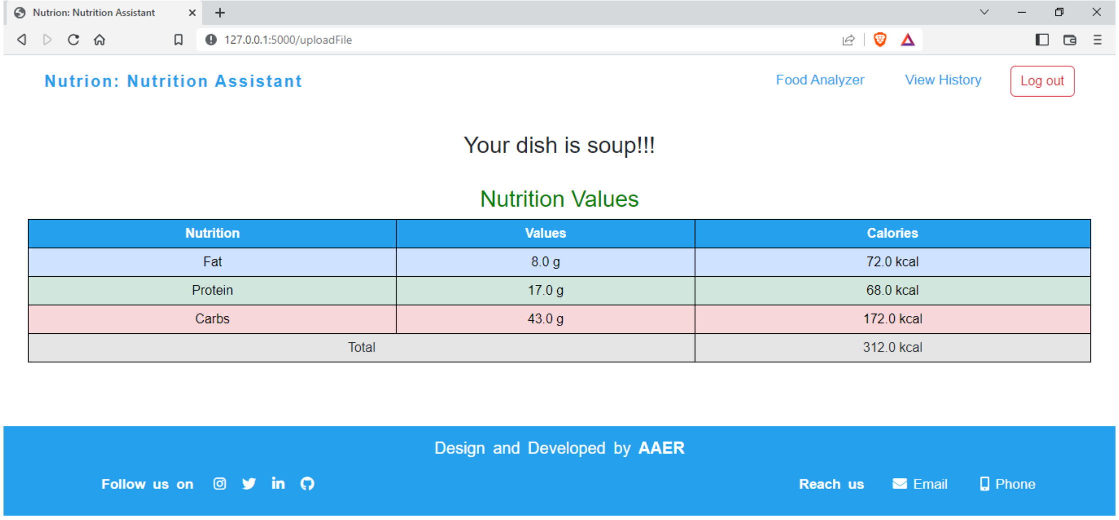
Design and Developed by **AAER**



Find Nutrition

Choose File No file chosen

(NOTE:Upload file in .jpg,.png)

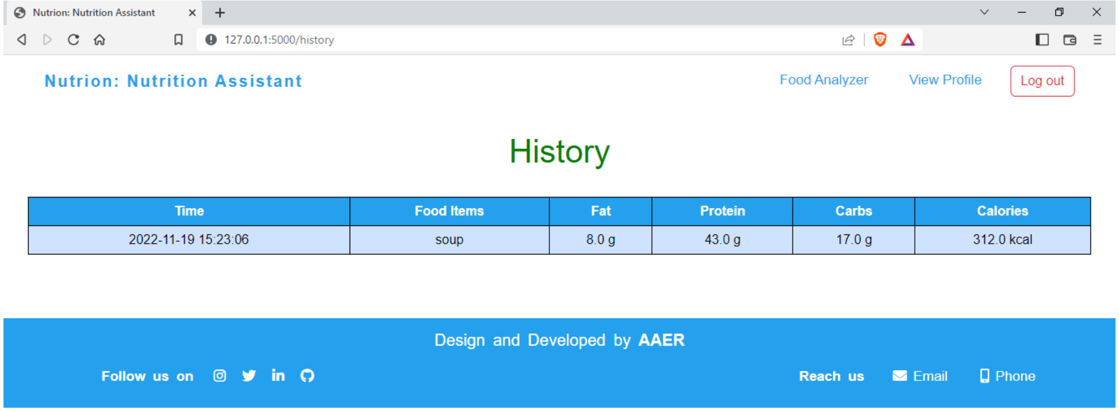

Nutrion: Nutrition Assistant
127.0.0.1:5000/uploadFile
Nutrion: Nutrition Assistant
Food Analyzer
View History
Log out

Your dish is soup!!!

Nutrition Values

Nutrition	Values	Calories
Fat	8.0 g	72.0 kcal
Protein	17.0 g	68.0 kcal
Carbs	43.0 g	172.0 kcal
Total		312.0 kcal

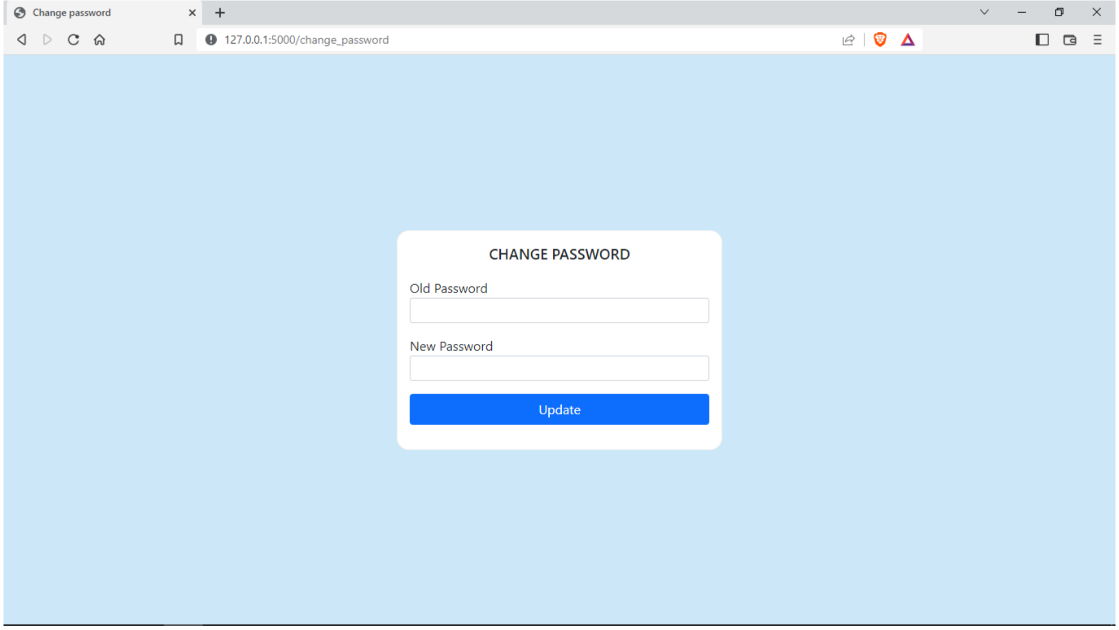
Design and Developed by AAER
Follow us on
Reach us
Email
Phone



History

Time	Food Items	Fat	Protein	Carbs	Calories
2022-11-19 15:23:06	soup	8.0 g	43.0 g	17.0 g	312.0 kcal

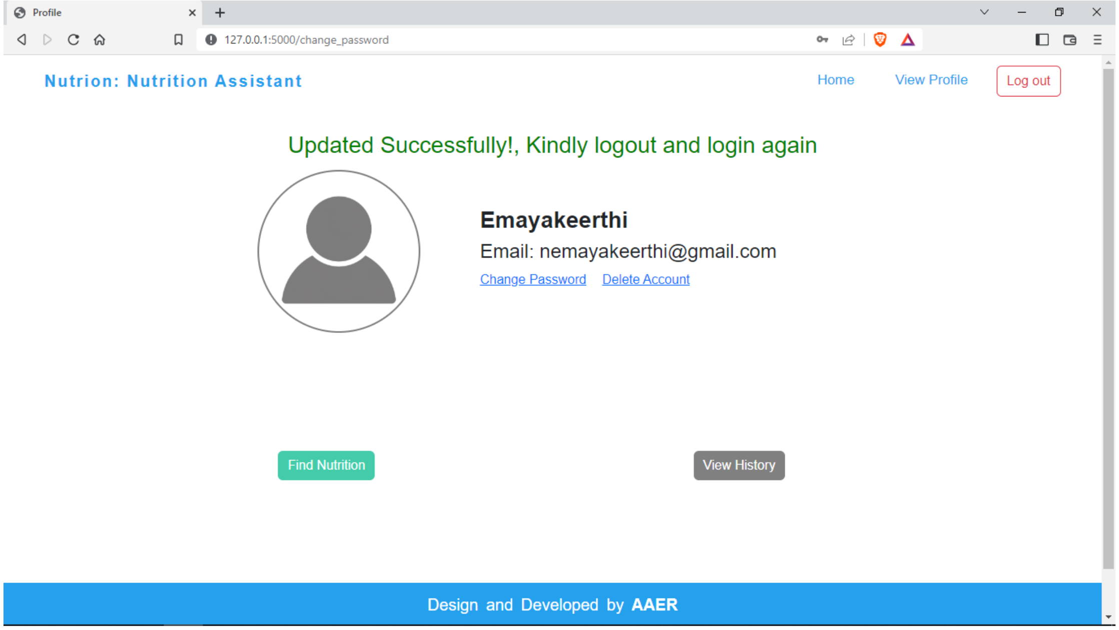
Design and Developed by **AAER**



CHANGE PASSWORD

Old Password

New Password

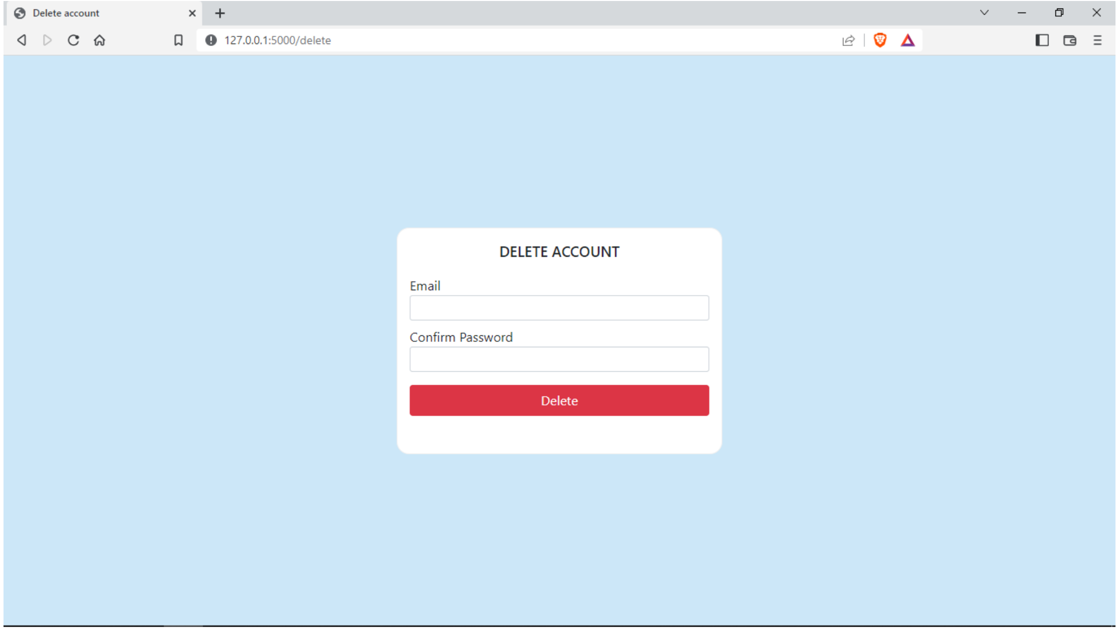


Updated Successfully!, Kindly logout and login again

Emayakeerthi

Email: nemayakeerthi@gmail.com

Design and Developed by **AAER**



DELETE ACCOUNT

Email

Confirm Password

10. ADVANTAGES

- Used to see fat, protein, carbs, and total calories just with the pictures of your meal.
- Easy to use.
- Works under low data connection.
- Obesity can reduced.

DISADVANTAGES

- Nutrition content rate is not accurate.
- It cannot be used without internet connection.

11. CONCLUSION

- The user who want to maintain a healthy and fit body but have no one to guide on their dieting.
- The user wants to develop a deep learning model that basically help athlete, body builders or other game players to keep up with their health and fitness by suggesting them with proper nutrition plan.
- The user wants an AI software to maintain healthy diet rather than having a physical consultant.
- People who are obese and overweight are more likely to have high-risk factors for heart disease, diabetes,hypertension. The goal of the application is to create a healthy lifestyle for its user.
- User has obesity who needs to follow diet to improve his health without the feeling that he's following diet.

12. FUTURE SCOPE

- Project scope is a way to set boundaries on your project and define exactly what goals, deadlines, and project deliverables you'll be working towards. By clarifying your project scope, you can ensure you hit your project goals and objectives without delay or overwork. Defining your project scope isn't a one-person job.
- You can work as a Nutritionist/Dietitian there and take control of the food intake and also the food quality consumed by the people. With a degree in food and nutrition, you can act as a Public.

- Health Nutritionist in non governmental organizations and play your part in spreading some good in the world.
- Future Scope is for the Undergraduates, Graduates and the Working Professionals. They may want to review or reconsider their future options and goals in terms of its suitability now; may be with a different perspective of their options in terms of time, resources, inclination etc.

13. APPENDIX

Source Code

```
from flask import Flask, render_template, request, redirect, url_for, flash, session
import ibm_db
import re
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import os
from werkzeug.utils import secure_filename
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
from clarifai_grpc.grpc.api import service_pb2, resources_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2
import spoonacular as sp
import datetime
import requests



#creating instance of flask class
app = Flask(__name__)

#connecting with ibm db2
app.secret_key = "nutritionassistantapplication"
conn = ibm_db.connect("DRIVER={IBM DB2 ODBC DRIVER}; DATABASE=bludb;
HOSTNAME=xxxx; PORT=xxxx;
```

```
SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt; UID=xxxx;
PWD=xxxxxxx;",",")

# Defining upload folder path
UPLOAD_FOLDER = os.path.join('static', 'uploads')
# # Define allowed files
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

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

            def send_confirmation_reg(user, mail):
                url = "https://rapidprod-sendgrid-v1.p.rapidapi.com/mail/send"

                text = f"""
                        <b>Dear {user}<b><br><br>
                        Thank you for completing your registration with us.<br><br>
                        This email serves as a confirmation that your account is activated and that you are
            officially a part of the Nutrion family.
                        Enjoy!<br><br>
                        <b>Regards,<br>
                        The Nutrion team</b>
                        """
                payload = {
                        "personalizations": [
                                    {
                                                "to": [{"email": mail}],
                                                "subject": "Welcome to Nutrion!!"
                                    }
                        ],
        "from": {"email": "nutritionassistantapplication@gmail.com"},
        "content": [
            {
                "type": "text/html",
                "value": text
            }
        ]
    }

    headers = {
        "content-type": "application/json",
        "X-RapidAPI-Key": "xxxxx",
        "X-RapidAPI-Host": "rapidprod-sendgrid-v1.p.rapidapi.com"
    }

    response = requests.request("POST", url, json=payload, headers=headers)


def send_confirmation_del(user, mail):
    url = "https://rapidprod-sendgrid-v1.p.rapidapi.com/mail/send"

    text = f"""
        <b>Dear {user}<b><br><br>

        We're sorry to see you! Let's hope, We'll see you again.<br><br>

        This email serves as a confirmation that your account is deleted and that you are officially a not part of the Nutrion family. But We love you and welcome you always<br><br>

        <b>Regards,<br>
        The Nutrion team</b>
        """

    payload = {
        "personalizations": [
            {
                "to": [{"email": mail}],
                "subject": "We're Sorry by Nutrion!!"
            }
        ],
        "from": {"email": "nutritionassistantapplication@gmail.com"},
        "content": [
            {
                "type": "text/html",
                "value": text
            }
        ]
    }

    headers = {
        "content-type": "application/json",
        "X-RapidAPI-Key": "xxxxxx",
        "X-RapidAPI-Host": "rapidprod-sendgrid-v1.p.rapidapi.com"
    }

    response = requests.request("POST", url, json=payload, headers=headers)



def send_confirmation_cp(user, mail):
    url = "https://rapidprod-sendgrid-v1.p.rapidapi.com/mail/send"

    text = f"""
```

```
        <b>Dear {user}<b><br><br>
        Thank you for imporve our services by changing our password.<br><br>
        This email serves as a confirmation that your account password is changed<br><br>
        <b>Regards,<br>
        The Nutrion team</b>
        """
    payload = {
        "personalizations": [
            {
                "to": [{"email": mail}],
                "subject": "Password has been changed!!"
            }
        ],
        "from": {"email": "nutritionassistantapplication@gmail.com"},
        "content": [
            {
                "type": "text/html",
                "value": text
            }
        ]
    }

    headers = {
        "content-type": "application/json",
        "X-RapidAPI-Key": "xxxxx",
        "X-RapidAPI-Host": "rapidprod-sendgrid-v1.p.rapidapi.com"
    }

    response = requests.request("POST", url, json=payload, headers=headers)
```

```
@app.route('/register', methods=["POST", "GET"])
def register():
    msg = ''
    #render_template("register.html")
    if request.method=="POST":
        #getting data from register form
        name = request.form.get("name")
        email = request.form.get("email")
        password = request.form.get("password")

        #checking an account existing
        sql = "SELECT * FROM USERS WHERE email = ? "
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)

        account  = ibm_db.fetch_assoc(stmt)

        if(account):
            msg = "Account already registered!"
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = "Invalid email address"
        elif not re.match(r'[A-Za-z\s]*$', name):
            msg = "Name should contain characters and space only"
        elif password:
            if(len(password)<8):
                msg = "Make sure your password is at lest 8 letters"
            elif (re.search('[0-9]',password) is None):
                msg = "Make sure your password has a number in it"
            elif (re.search('[a-z]', password) is None):
                msg = "Make sure your password has a small letter in it"
            elif (re.search('[a-z]', password) is None):
                msg = "Make sure your password has a Capital letter in it"
            elif (re.compile('[^0-9a-zA-Z]+').search(password) is None):
                msg = "Make sure your password has a special character in it"
            else:
                #inserting the data into db2 database
                sql = "INSERT INTO USERS VALUES(?,?,?)"
                stmt = ibm_db.prepare(conn, sql)
                ibm_db.bind_param(stmt, 1, name)
                ibm_db.bind_param(stmt, 2, email)
                ibm_db.bind_param(stmt, 3, password)
                ibm_db.execute(stmt)

                msg = "Account created successfully, Kindly login"

                #sending confirmation mail to user
                send_confirmation_reg(name, email)

                return redirect(url_for("login", msg=msg, clr="green"))
    else:
        if "user" in session:
            return render_template("profile.html", msg=", name=session['name'],
email=session['user'])

    return render_template("register.html", msg=msg)

@app.route("/profile")
def profile():
    if "user" not in session:
        return redirect(url_for("login", msg="Kindly login", clr="red"))

    return render_template("profile.html", msg=", name=session['name'],
email=session['user'])



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

        #retrieving the user details
        sql = "SELECT * FROM users WHERE email = ? AND pwd = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        if account:
            name = account['NAME']
            session['loggedin'] = True
            session['user'] = email
            session['name'] = name

            msg = "Login successfully"

            return render_template("profile.html", msg=msg, name=name, email=email,
clr="green")

        elif "user" in session:
            return redirect(url_for('profile'))
        else:
            msg = "Incorrect password or email address"

    else:
        if "user" in session:
            return render_template("profile.html", msg=", name=session['name'],
email=session['user'], clr="green")

    return render_template("login.html", msg=msg, clr="red")

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('name', None)
    session.pop('user', None)
    flash("Loged out successfully")
    return render_template("index.html", msg="Loged out successfully", clr="red",
cls="msg")

@app.route("/delete", methods=['POST', 'GET'])
def delete():
    if "user" not in session:
        return redirect(url_for('login', msg="Kindly login", clr="red"))

    msg="
    color="
    if request.method=="POST":
        email = request.form['email']
        password = request.form["password"]
        sql = "SELECT * FROM USERS WHERE email = ? and pwd = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            #deleting user data from users table
            sql = "DELETE FROM USERS WHERE email = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, email)
            ibm_db.execute(stmt)

            #deleting user data from food data table
            sql = "DELETE FROM FOODDATA WHERE email = ?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, email)
            ibm_db.execute(stmt)

            name = session['name']
            #remove the session
            session.pop('loggedin', None)
            session.pop('name', None)
            session.pop('user', None)

            #sending confirmation mail to user
            send_confirmation_del(name, email)

            return render_template("index.html", msg="Account deleted successfully",
clr="green", cls="msg")
        else:
            msg = "Incorrect email/password"
            color = 'red'
```

```
    return render_template("delete.html", msg=msg, clr=color)



@app.route('/uploadFile',  methods=("POST", "GET"))
def uploadFile():

    if request.method == 'POST':
        # Upload file flask
        uploaded_img = request.files['uploaded-file']
        # Extracting uploaded data file name
        img_filename = secure_filename(uploaded_img.filename)
        # Upload file to database (defined uploaded folder in static path)
        path = os.path.join(app.config['UPLOAD_FOLDER'], img_filename)
        #save image in local directory
        uploaded_img.save(path)
        food = predictConcept(path)
        data = getNutritions(food)

        # Storing uploaded file path in flask session
        session['uploaded_img_file_path'] = os.path.join(app.config['UPLOAD_FOLDER'],
img_filename)
        print("File name: ",img_filename)
        fat = data['fat']
        carbs = data['carbs']
        protein = data['protein']

        fat_value = str(fat['value'])+" "+fat['unit']
        carb_value = str(carbs['value'])+" "+carbs['unit']
        protein_value = str(protein['value'])+" "+protein['unit']
        fat_cal = fat['value']*9
        carbs_cal = carbs['value']*4
        protein_cal = protein['value']*4
        total = str(fat_cal+carbs_cal+protein_cal)+" kcal"
        fat_cal = str(fat_cal)+" kcal"
        carbs_cal = str(carbs_cal)+" kcal"
        protein_cal = str(protein_cal)+" kcal"

        crttime = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        sql = "INSERT INTO FOODDATA VALUES(?,?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session['user'])
        ibm_db.bind_param(stmt, 2, crttime)
        ibm_db.bind_param(stmt, 3, food)
        ibm_db.bind_param(stmt, 4, fat_value)
        ibm_db.bind_param(stmt, 5, carb_value)
        ibm_db.bind_param(stmt, 6, protein_value)
        ibm_db.bind_param(stmt, 7, total)
        ibm_db.execute(stmt)

    else:
        if "user" in session:
            return render_template("profile.html", msg=", name=session['name'],
email=session['user'])

    return render_template("nutrition_table.html", food=food, fat=fat_value,
carbs=carb_value, protein=protein_value, fat_cal= fat_cal, protein_cal=protein_cal,
carbs_cal=carbs_cal, total_calories=total)


#Predict the food item in the given using the image recognition in clarifai model
```

```
def predictConcept(path):
    USER_ID = 'xxxxxx'
    # Your PAT (Personal Access Token) can be found in the portal under Authentification
    PAT = 'xxxxxx'
    APP_ID = 'Nutrion'
    # Change these to whatever model and image input you want to use
    MODEL_ID = 'general-image-recognition'
    IMAGE_FILE_LOCATION = path
    # This is optional. You can specify a model version or the empty string for the default
    MODEL_VERSION_ID = ''

    channel = ClarifaiChannel.get_grpc_channel()
    stub = service_pb2_grpc.V2Stub(channel)

    metadata = (('authorization', 'Key ' + PAT),)

    userDataObject = resources_pb2.UserAppIDSet(user_id=USER_ID, app_id=APP_ID)

    with open(IMAGE_FILE_LOCATION, "rb") as f:
        file_bytes = f.read()

    post_model_outputs_response = stub.PostModelOutputs(
        service_pb2.PostModelOutputsRequest(
            user_app_id=userDataObject,  # The userDataObject is created in the overview
and is required when using a PAT
            model_id=MODEL_ID,
            version_id=MODEL_VERSION_ID,  # This is optional. Defaults to the latest
model version
            inputs=[
                resources_pb2.Input(
                    data=resources_pb2.Data(
                        image=resources_pb2.Image(
                            base64=file_bytes
                        )
                    )
                )
            ]
        ),
        metadata=metadata
    )
    if post_model_outputs_response.status.code != status_code_pb2.SUCCESS:
        print(post_model_outputs_response.status)
        raise Exception("Post model outputs failed, status: " +
post_model_outputs_response.status.description)

    # Since we have one input, one output will exist here
    output = post_model_outputs_response.outputs[0]

    for concept in output.data.concepts:
        print("%s %d"%(concept.name, concept.value))
        return concept.name



    """   text = text+" "+str(concept.name)
    response = api.detect_food_in_text(text).json()
    for data in response['annotations']:
        if(data['tag']=='ingredient'):
            print(data['annotation'])
    """



#search the nutrition in given food using FatSecret API
api = sp.API("xxxxxx")
def getNutritions(food_item):
    response = api.guess_nutrition_by_dish_name(food_item)
    data = response.json()
    return data


@app.route('/nutrition')
def nutrition():
    if 'user' not in session:
        return redirect(url_for('login', msg="Kindly login", clr="red"))

    return render_template('nutrition.html')

@app.route('/history')
def history():
    if 'user' in session:
        email = session['user']
        sql = "SELECT * FROM FOODDATA WHERE EMAIL = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)


        history_data = ibm_db.fetch_assoc(stmt)
        history = []
        if history_data:
            while history_data!=False:
                history.append(history_data)
                history_data = ibm_db.fetch_assoc(stmt)

            return render_template("history.html", history=history)
        else:
            history=[{'EVENTTIME':'-', 'FOODNAME':'-', 'FAT':'-', 'CARBS':'-',
'PROTEIN':'-', 'TOTALCAL':'-'}]

            print(history)
            return render_template("history.html", history=history)



@app.route('/change_password', methods=['GET', 'POST'])
def change_password():
    if not session.get('user'):
        return redirect(url_for('login', msg="Kindly login", clr="red"))



    msg=''
    color = ''

    if request.method == 'POST':
        email = session['user']
        oldpass = request.form.get('oldpass')
        newpass = request.form.get('newpass')

        sql = 'SELECT * from users where email = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        dbpass = account['PWD'].strip()
        print(dbpass, oldpass, newpass)

        if dbpass == oldpass and oldpass!=newpass:
            sql = 'UPDATE USERS SET PWD = ? WHERE EMAIL = ?'
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, newpass)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.execute(stmt)
            msg = 'Updated Successfully!, Kindly logout and login again'
            color = "green"

            #sending confirmation mail
            send_confirmation_cp(session['name'], email)
            return render_template('profile.html', name=session['name'], email=email,
msg=msg, clr=color)
        elif oldpass==newpass:
            msg = "Old password and new password should not be same"
            color = "red"
        else:
            msg = 'Old Password Incorrect!'
            color = "red"

    return render_template('changePassword.html', msg=msg, clr=color)
if __name__=="__main__":
    app.run(debug=True)
```

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-1423-1658387310

Demo Video Link

http://shorturl.at/fhqx3