**PROJECT REPORT**

**1. INTRODUCTION**

**1.1 PROJECT OVERVIEW:**

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.

The spoonacular Nutrition, Recipe, and Food API allow you to access over 365,000 recipes and 86,000 food products. Our food ontology and semantic recipe search engine make it possible to search for recipes using natural language queries, such as "gluten-free brownies without sugar" or "low-fat vegan cupcakes." 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. With our powerful API, you can create many kinds of food and especially nutrition apps.

**1.2PURPOSE:**

* One of the most basic functions of this app is to guide its users towards a healthy diet and assist them to achieve their health goals. So, once your user specifies the goal like desired weight goal, body type, food habits, and preferred food items, this app suggests you with a proper diet accordingly.
* Nutrition assistants help dieticians with providing proper nutrition at healthcare facilities. They determine patients' nutritional needs, assess risk factors, and plan meals and menus.
* They also ensure proper sterilization of plates and utensils**.**

**2. LITERATURE SURVEY**

**2.1 EXISTING PROBLEM:**

**Decreased appetite**

Lack of appetite, or decreased hunger, is one of the most troublesome nutrition problems you can experience. Although it's a common problem, its cause is often unknown. Appetite-stimulating medicines are available. Ask your provider if such medicines would help you.

**Snack guidelines**

Wasting your energy eating foods that provide little or no nutritional value (such as potato chips, candy bars, colas and other snack foods).

A nutritional problem or deficiency refers to a condition when an individual's body experiences a shortage of essential nutrients or some specific nutrient. Such problems can give rise to several health issues such as anaemia.

**2.2 REFERENCES**:

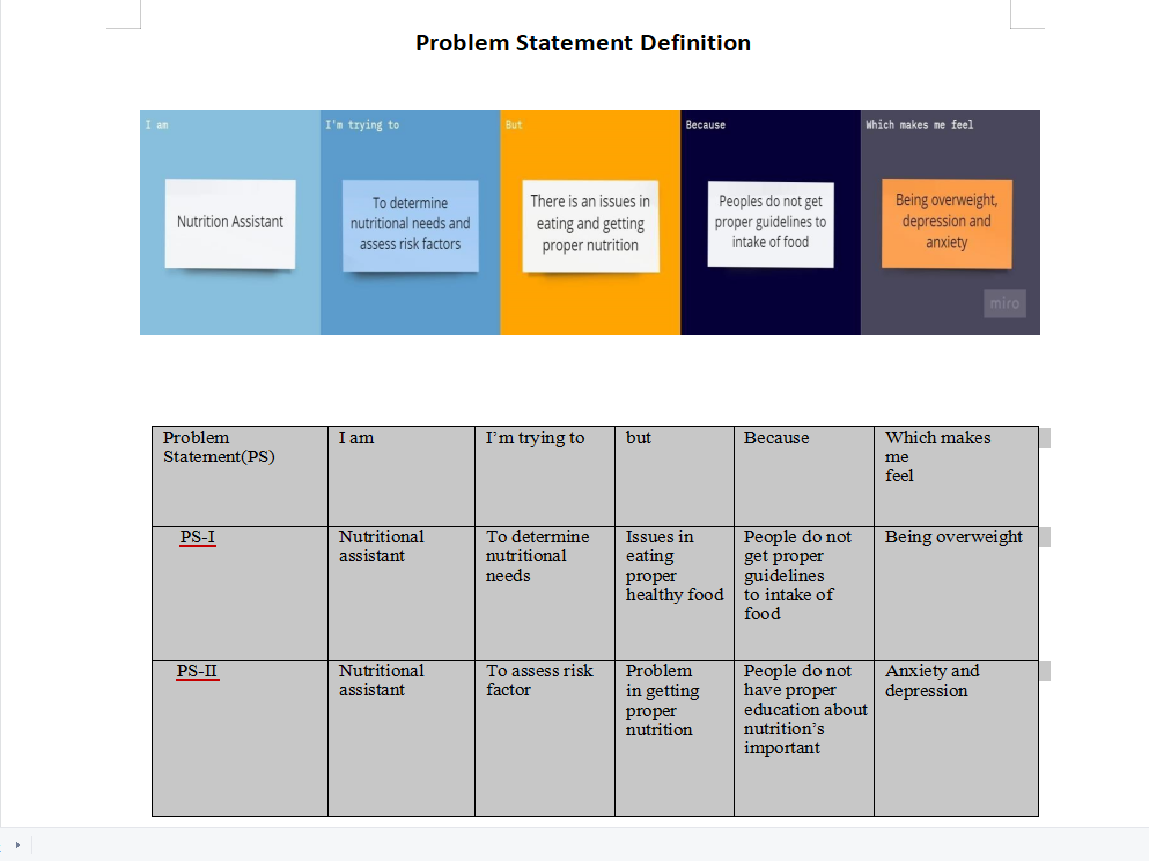
<https://www.researchgate.net/publication/346411010_DEVELOPMENT_OF_A_CLOUD_BASED_SOLUTION_FOR_EFFECTIVE_NUTRITION_INTERVENTION_IN_THE_MANAGEMENT_OF_LIFESTYLE_DISEASES>

<http://www.msspublishers.com/storage/articles/2021/08/17/hSNmr1Fp4NJ6urMGODrY10xuGQfJwV0Bk1JYnNPZ.pdf>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4985610/>

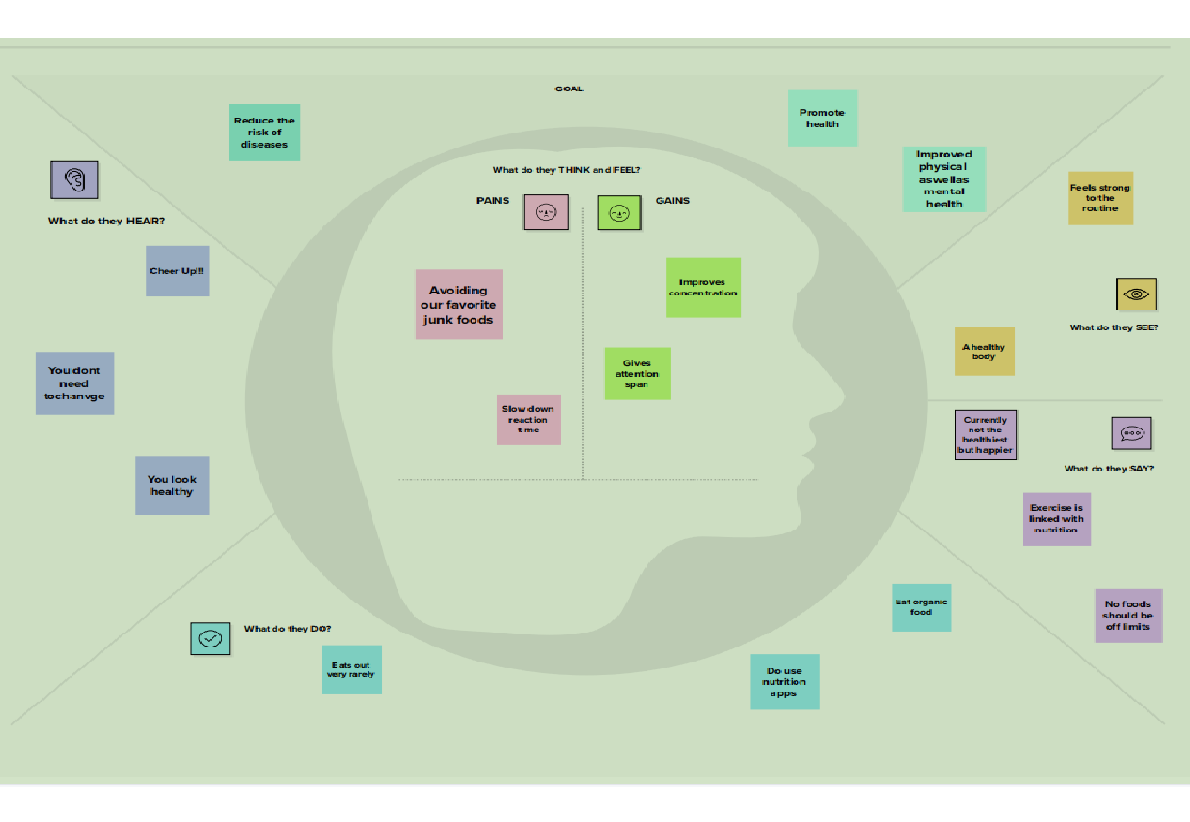
<https://www.researchgate.net/publication/268334408_Personal_Health_Assistant_on_Android_Mobile_Device_Sleeping_Nutrition_and_Exercise>

**2.3 PROBLEM STATEMENT DEFINITION:**

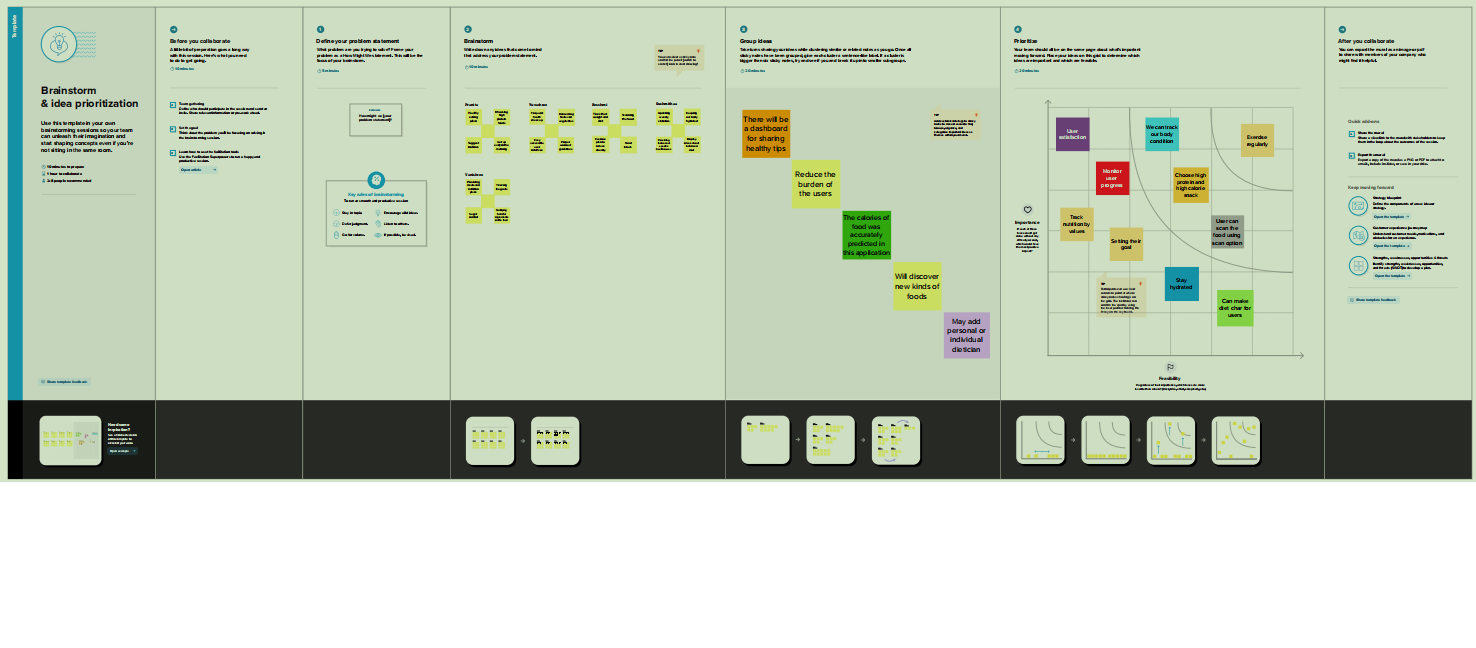
****

**3. IDEATION & PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS:**

****

**3.2 IDEATION & BRAINSTORMING:**

****

**3.3 PROPOSED SOLUTION:**

IDEA/SOLUTION DESCRIPTION:

People can easily track the Nutrition and calories by scanning an real-time images of a food and examine it’s nutritional content which will improves the dietary habits. Smart nutrition and foods can prevent diseases. This app will provide proper nutrition helps in maintaining a healthy lifestyle and also recommended diet plans for users.

NOVELITY/UNIQUENESS:

This solution has the uniqueness that we can realize real time images of meal and can easily analyssse its nutritional content. A web app that can automatically estimate food attributes such as ingredients and nutrition value by classifying the input image.

SCALABILITY OF THE SOLUTION:

People can access from anywhere at any time to track the calories and nutrition value that will improve a healthy eating pattern. This App will improve the dietary habits and helps in maintaining a healthy weight and healthy lifestyle.

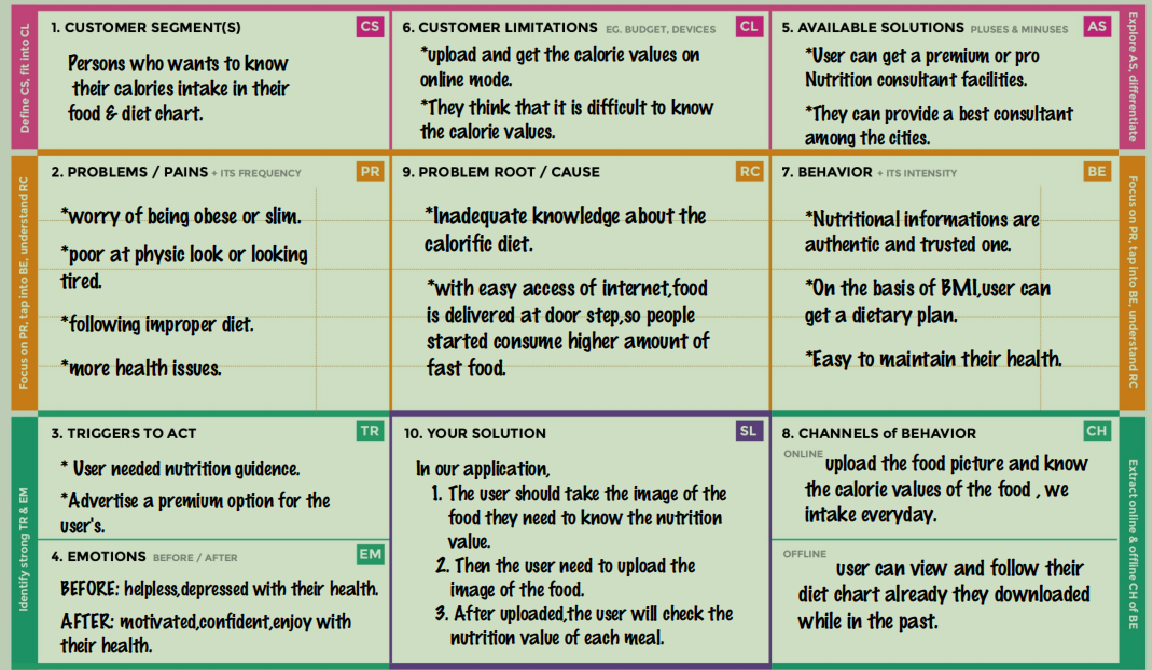
BUSINESS MODEL:

Social media is the best way to develop this application. This application will increase the confidence among the people. It is great to use, amazing convenience and also have subscription once user hit certain services.

SOCIAL IMPACT/CUSTOMER SATISFACTION:

The Obesity rate will get reduced and people will be able to lead a healthy life. It helps to achieve and maintain a healthy weight balancing in their routine life.

**3.4 PROBLEM SOLUTION FIT:**

****

**4. REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENTS:**

**1. Add Health Information**

This application will allow to add health related information of the user.

**2. Delete health information**

This application will allow to delete the unwanted details about their health.

**3. Categories of nutritional food**

The categories of food.

**4. View of Dashboard**

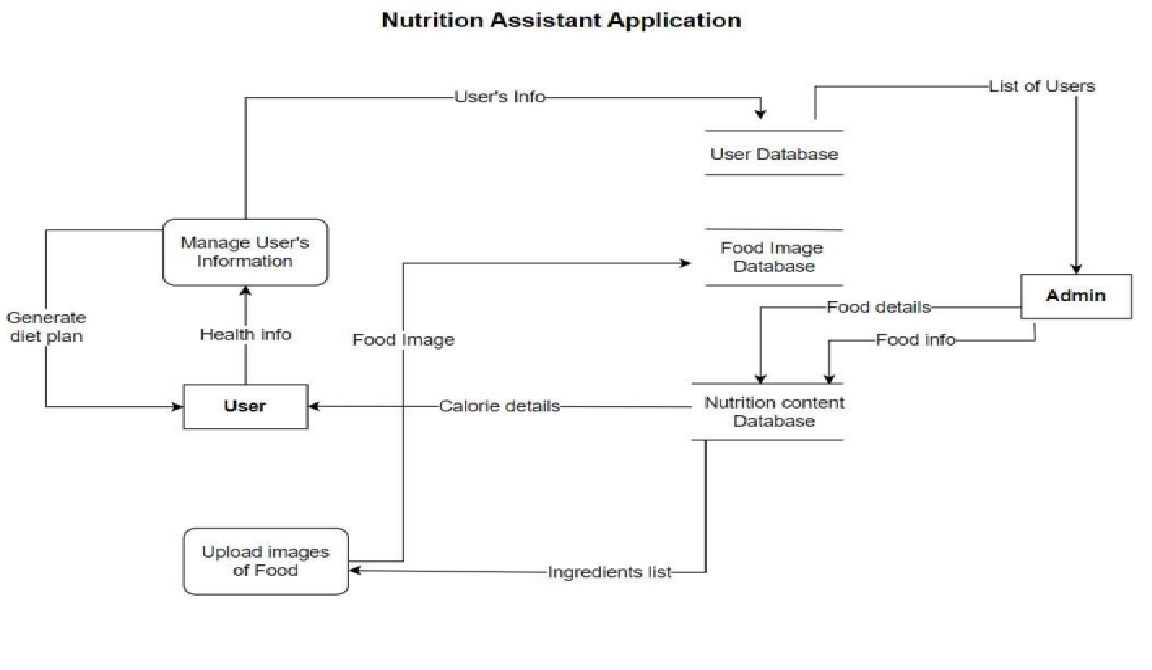
Application will allow user to view the dashboard containing nutrition details.

**5. Mail Notification**

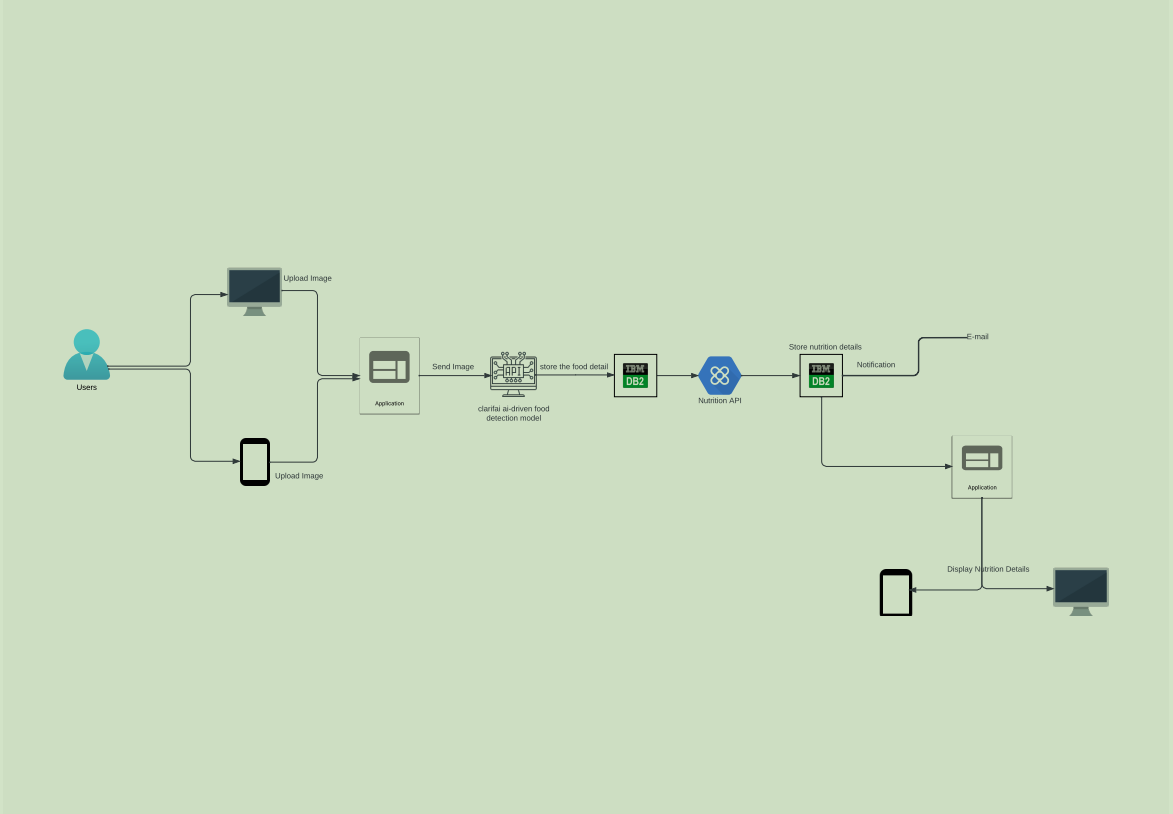
This application will allow to send mail notification to user when there are any issues regarding their health.

**5.PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAMS:**

****

**5.2 SOLUTION AND TECHNICAL ARCHITECTURE:**

****

**5.3 USER STORIES:**

**David pickup -** My Dietitian did a great job of understanding the way I work and providing insightful comments for ways of rethinking food, preparation and planning. She was a great listener and extremely easy to open up to (very approachable). I really enjoyed how we managed to change habits, set a strong baseline with good food preparation and planning, and providing a different perspective on nutrition seemingly without trying. I now feel I have a great system in place that I can build from.

**Denis Gagne -** I came some time ago for a few nutrition counseling sessions as my blood sugar was a bit high. I am happy to report that my blood sugar is back to normal as I have made big changes in my choices and portion sizes. I have lost about 20 lbs. in less than a year and while I still have a ways to go, I am doing well. Thanks for all of your help!! I’ll apprise if I think I need any “tune up” sessions with you down the road.”

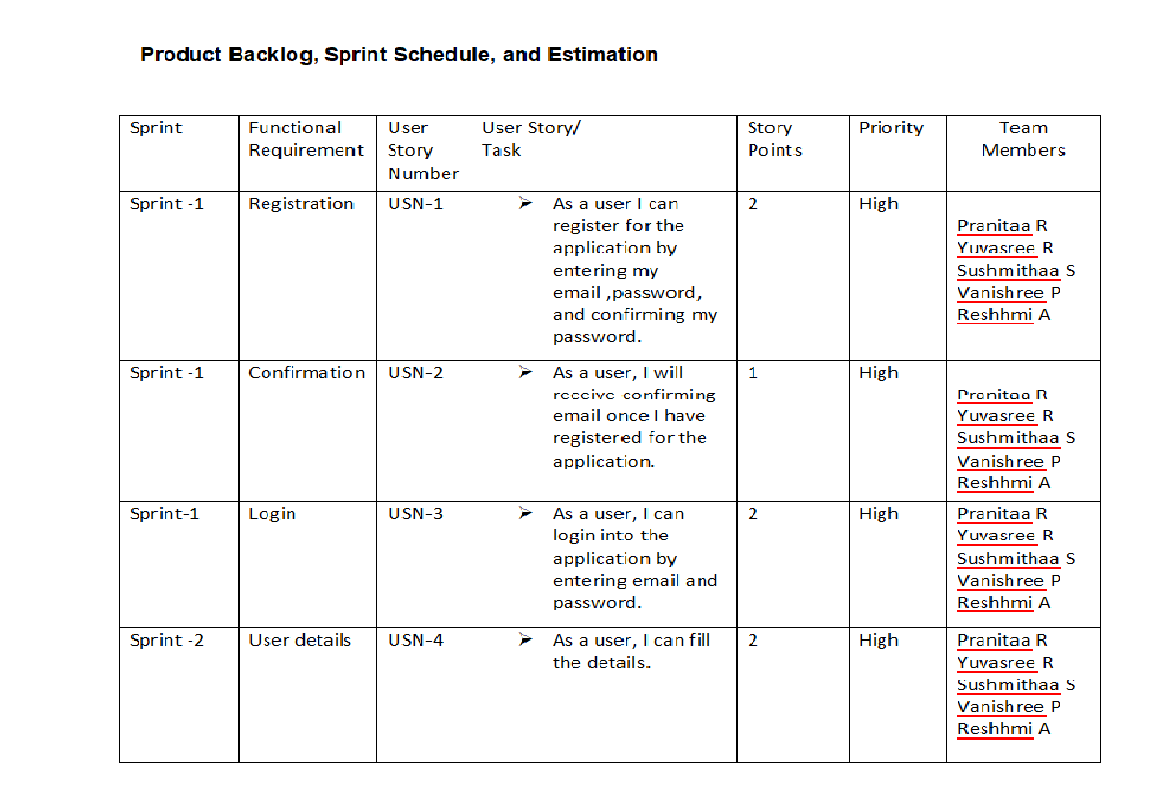
**Rhonda Jenkins -** “This is the first time I feel satisfied; my cravings have diminished dramatically, and I have a whole new relationship with food. I am eating guilt-free for the first time in my life. My energy has also dramatically increased and I feel great!”

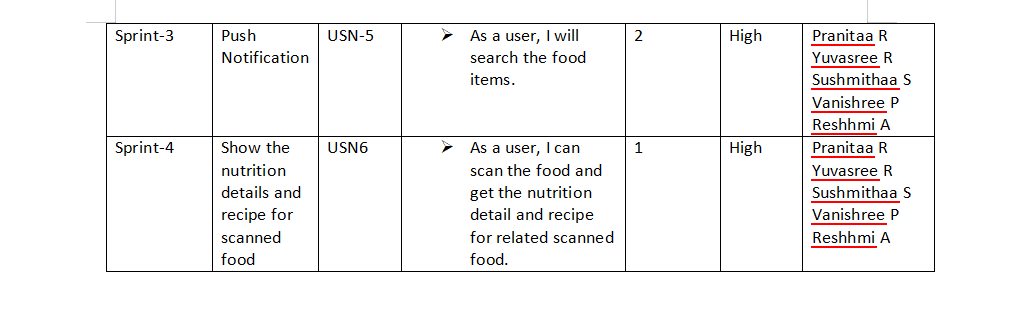
**Andre Perras** - “I’ve never been to a dietician before. What I received was far more advice and knowledge that I had initially expected. My Dietitian is extremely knowledgeable, excellent at communicating, and just plain fun to talk to. **I’ve learned so much in just three appointments.** I can actually feel the differences in my body after making the changes recommended to me. I’m far more aware of how to read my body’s reactions to certain foods. I have learned how to put together healthy delicious meals that aren’t repetitive. I don’t have to deprive myself of anything nor do I coerce myself into eating things I dislike. I’d rate this 6 stars instead of 5 stars on google reviews as my expectations were far over achieved! Thank you!”

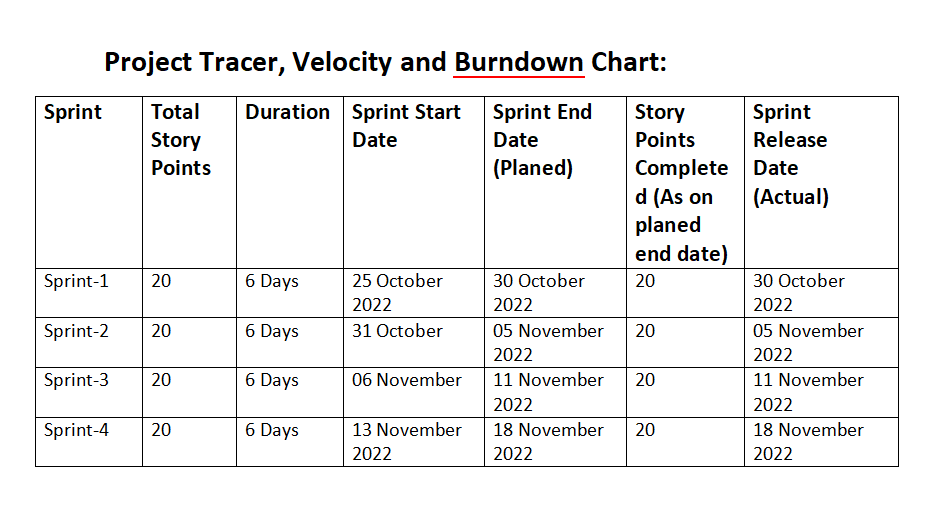
**Beatrice Wilhelm** - “By giving me a **safe place to talk**and examine what was or was not working in my life I came to see the opportunity to reduce controllable stress producing behavior which was impacting my body. Generally, I was amazed at the poise and understanding that my Dietitian provided in my sessions. She was **empathetic and supportive**of positive changes that I made, and **understanding**of those I didn’t make.”

**6. SPRINT PLANNING & SCHEDULING**

**6.1 SPRINT PLANNING AND ESTIMATION:**

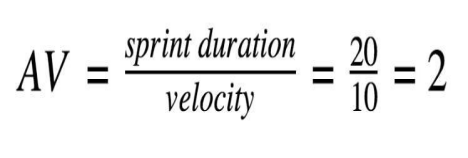
****

****

****

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

**Average Velocity = Story Points perday**

**Sprint Duration = Number of (Duration) days per Sprint**

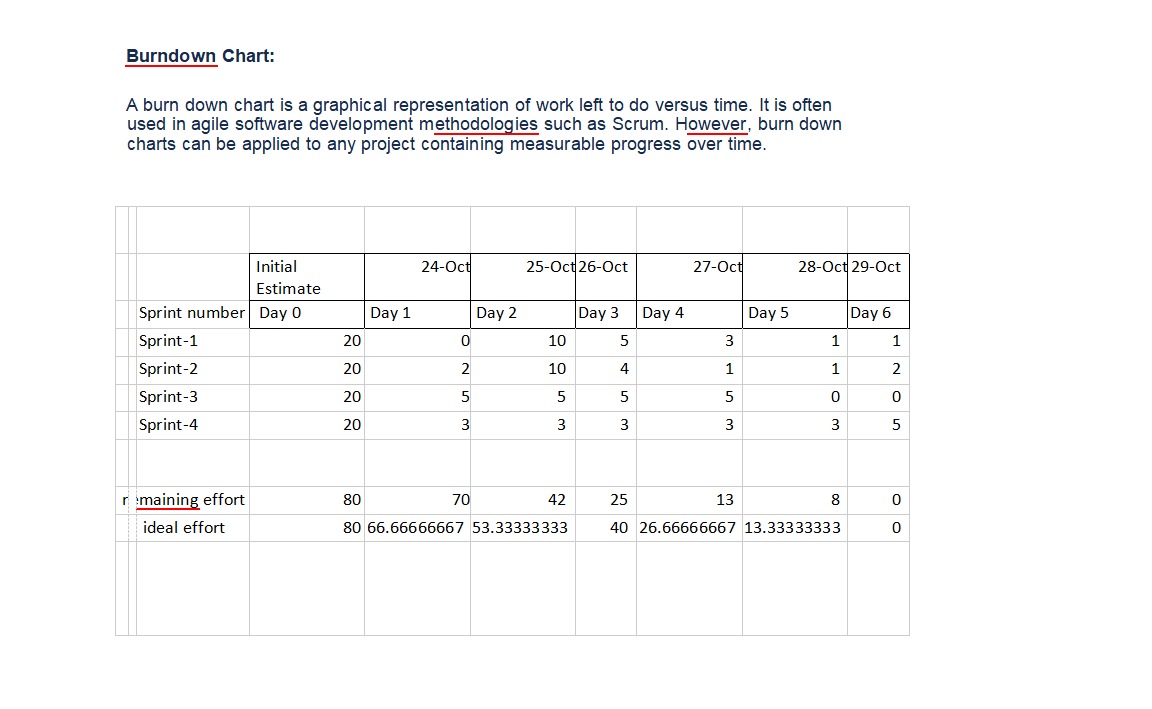
**Velocity = Points per Sprint**

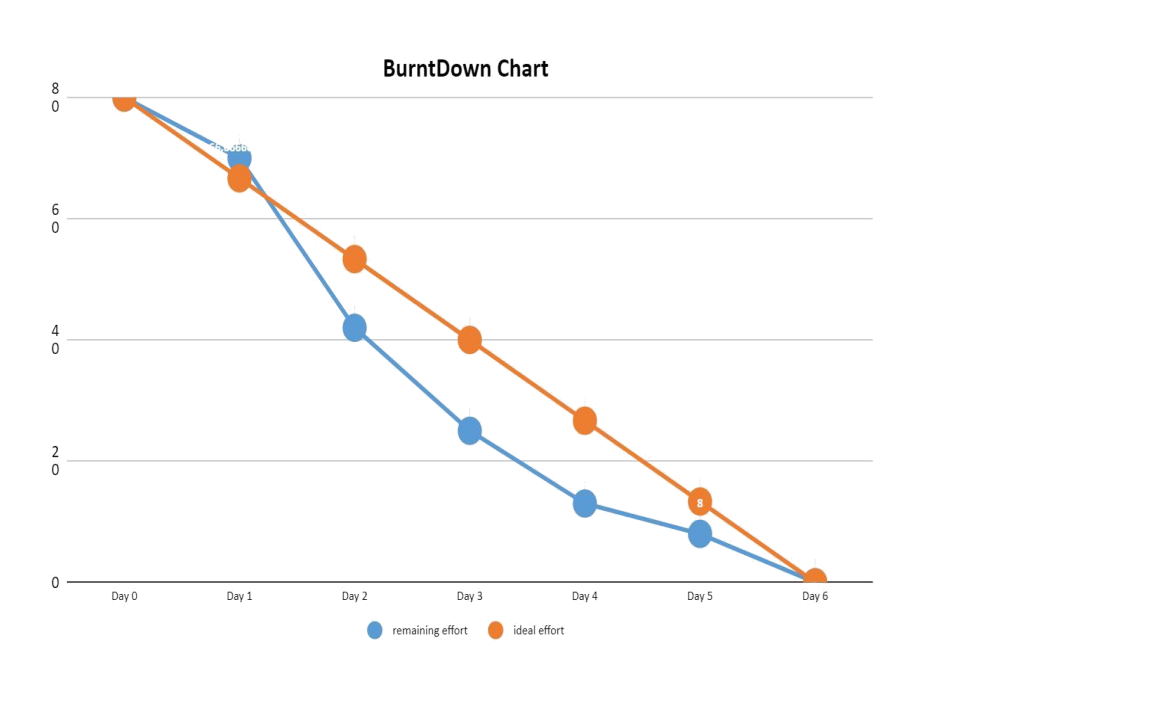
**20**

**AV**= 

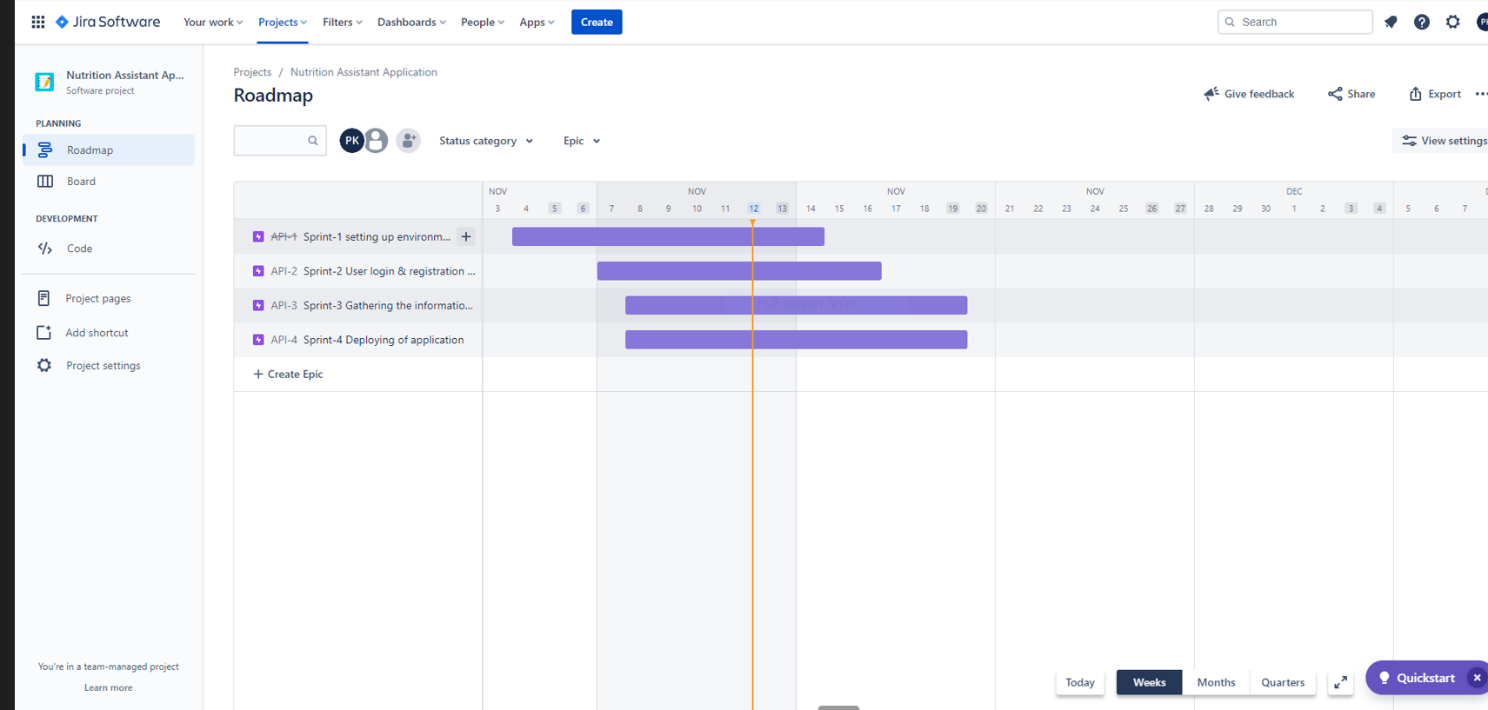
**6**

Therefore, the AVERAGE VELOCITY IS 4 POINTS PER SPRINT



****

**6.3 REPORTS FROM JIRA:**

****

**7. CODING & SOLUTIONING**

**STATIC:**

body {

background-color: rgb(192, 171, 171);;

box-sizing: border-box;

}

.dash {

padding: 50px 0;

}

.header {

font-size: 1.2rem;

background-color: rgb(192, 171, 171);

padding: 50px;

border: 1px solid black;

margin: 0 20px;

box-sizing: border-box;

border-radius: 25px;

}

.header >h4{

text-align: right;

font-family: Arial, Helvetica, sans-serif;

}

.combutton {

margin: 10px 30px;

padding: 10px 30px;

border-radius: 10px;

}

.btns {

padding: 5px 30px;

}

.btns:hover {

background-color: rgb(143, 131, 116);

}

…

body {

background-color: rgb(192, 171, 171);

box-sizing: border-box;

}

.dash {

padding: 50px 0;

}

.header {

font-size: 1.2rem;

background-color: rgb(192, 171, 171);

padding: 50px;

border: 1px solid black;

margin: 0 20px;

box-sizing: border-box;

border-radius: 25px;

}

.header >h4{

text-align: right;

font-family: Arial, Helvetica, sans-serif;

}

.rowh {

min-height: 75vh;

}

.colh {

display: flex;

flex-direction: column;

border-radius: 20px;

padding: 40px;

}

.maincon {

padding: 40px;

border-radius: 20px;

height: 100%;

text-align: start;

background-color: aliceblue;

}

.maincon > h4 {

text-decoration: underline;

text-align: center;

padding-bottom: 40px;

}

.maincon > h5 {

padding: 10px 20px;

/\*background-color: beige;\*/

border-radius: 20px;

margin-bottom: 20px;

position: relative;

-webkit-transition-duration: 0.4s;

transition-duration: 0.4s;

}

.normsize {

height: 100%;

width: 100%;

box-sizing: border-box;

/\* min-width: 250px;

min-height: 350px;\*/

}

.lesssize {

width: 90%;

height: 90%;

}

.normpadding {

padding: 30px;

margin: 10px;

box-sizing: border-box;

}

.roudcorner {

border-radius: 25px;

margin: 10px;

}

.comcolor {

background-color: aliceblue;

-webkit-box-shadow: 9px 10px 23px 7px rgba(0,0,0,0.75);

-moz-box-shadow: 9px 10px 23px 7px rgba(0,0,0,0.75);

box-shadow: 9px 10px 23px 7px rgba(0,0,0,0.75);

}

.comflex {

display: flex;

justify-content: center;

align-items: center;

align-content: center;

padding: 30px;

}

.comflex-col {

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

align-content: center;

padding: 30px;

}

.subflex {

align-self: center;

justify-self: center;

text-align: center;

}

.combutton {

margin: 10px 30px;

padding: 10px 30px;

border-radius: 10px;

}

.btns {

padding: 5px 30px;

}

.btns:hover {

background-color: rgb(143, 131, 116);

}

/\*floating list - view history\*/

.float {

position: absolute;

margin-inline: auto;

top: 25vh;

min-height: 30vh;

display: flex;

justify-content: center;

}

.containers {

width: min(calc(100% - 15%), 840px);

margin-inline: auto;

}

.floatcontainer {

display: flex;

flex-direction: column;

background-color: white;

border-radius: 25px;

-webkit-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

-moz-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

}

.box1 {

display: flex;

justify-content: right;

position: relative;

}

.closes {

position: absolute;

right: 32px;

top: 32px;

width: 32px;

height: 32px;

opacity: 0.3;

}

.closes:hover {

opacity: 1;

}

.closes:before, .closes:after {

position: absolute;

left: 15px;

content: ' ';

height: 33px;

width: 2px;

background-color: #333;

}

.closes:before {

transform: rotate(45deg);

}

.closes:after {

transform: rotate(-45deg);

}

.box2 {

margin: 20px 40px;

display: flex;

flex-direction: column;

}

.bcol{

padding: 10px;

margin-bottom: 5px;

}

.inline {

display: inline;

}

.link-button {

background: none;

border: none;

color: blue;

text-decoration: underline;

cursor: pointer;

font-size: 1em;

font-family: serif;

}

.link-button:focus {

outline: none;

}

.link-button:active {

color:red;

}

….

body {

height: 100vh;

background-color: aqua;

}

.maincontainer {

min-height: 100vh;

}

.maxhight {

height: 100%;

width: 100%;

}

.displaytype {

display: flex;

justify-content: center;

}

.floatcontainers {

display: flex;

background-color: white;

border-radius: 25px;

-webkit-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

-moz-box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

box-shadow: 6px 6px 21px 4px rgba(0,0,0,0.75);

}

.maincontainers {

display: flex;

justify-content: center;

align-items: center;

align-content: center;

}

.boxfor {

display: flex;

flex-direction: row;

justify-content: center;

}

….

@media only screen and (max-width :768px ) {

.colh {

height: auto;

}

.lists {

height: 330px;

overflow: auto;

flex-direction: column;

}

}

body {

background-color: rgb(255, 255, 255);

font-family: Futura ,Trebuchet MS,Arial,sans-serif;

min-height: 100vh;

font-size: 1.3rem;

}

.heading {

color: rgb(25, 23, 23);

}

.colh {

height: auto;

display: flex;

justify-content: center;

align-items: center;

padding: 50px 0;

}

.colh > div {

width: 100%;

}

.spaceimp {

margin-bottom: 200px;

}

.lists {

list-style-type: none;

display: flex;

font-size: 1.2rem;

justify-content: space-around;

align-items: center;

align-content: center;

transition: 2.8s;

}

.lists > li {

text-decoration: none;

}

.lists > li > a {

text-decoration: none;

padding: 10px 40px;

color: rgb(11, 9, 9);

transition: .8s ease-in;

border-radius: 26px;

background: #e3e3e3;

box-shadow: inset -17px 17px 33px #d1d1d1,

inset 17px -17px 33px #f5f5f5;

}

.lists > li > a:hover {

color: rgb(251, 249, 249);

border-radius: 26px;

background: #131111;

box-shadow: -11px 11px 22px #8a8484,

11px -11px 22px #a49d9d;

}

p {

text-align: justify;

}

.info {

min-height: 50vh;

color: rgb(1, 9, 9);

min-width: 10vw;

padding-left: 30px;

}

.content {

margin-top: 50px;

padding-bottom: 50px;

box-sizing: border-box;

}

.content > info > p {

font-size: 2.5rem;

}

.cont\_1 {

width: 100%;

height: 100%;

display: flex;

justify-content: top;

align-items: flex-end;

flex-direction: column;

flex-grow: 1;

padding: 0 10px 0 0;

position: relative;

}

.login {

width: 100%;

height: 100%;

left: 0;

top: 0;

position: absolute;

z-index: 1;

display: none;

margin-bottom: 50px;

animation-name: loginhidder;

animation-duration: 2s;

}

.about {

background-color: rgba(188, 196, 196,0.8);

min-height: 200px;

width: 80%;

border-radius: 25px;

margin-right: 10%;

margin-top: 100px;

padding: 20px;

-webkit-box-shadow: 3px 3px 5px 6px rgb(89, 82, 82); /\* Safari 3-4, iOS 4.0.2 - 4.2, Android 2.3+ \*/

-moz-box-shadow: 3px 3px 5px 6px rgb(89, 82, 82); /\* Firefox 3.5 - 3.6 \*/

box-shadow: 3px 3px 5px 6px rgb(89, 82, 82);

display: none;

animation-name: loginhidder;

animation-duration: 2s;

}

@keyframes loginhidder {

0% {opacity: 0;}

50% {opacity: .5;}

100% {opacity: 1;}

}

.list {

padding-left: 80px;

}

.options{

padding: 0px 15px;

background-color: rgba(188, 196, 196,0.8);

box-sizing: border-box;

text-align: center;

margin-bottom: 20px;

border-radius: 25px;

display: flex;

width: 90%;

transition: opacity .9s;

}

.options > p {

display: flex;

justify-content: center;

align-items: center;

align-content: center;

font-size: 1.2rem;

font-style: bold;

}

#l1 {

position: relative;

animation-name: example;

animation-duration: 1s;

}

#l2 {

position: relative;

animation-name: example;

animation-duration: 1.5s;

}

#l3 {

position: relative;

animation-name: example;

animation-duration: 2s;

}

#l4 {

position: relative;

animation-name: example;

animation-duration: 2.5s;

}

@keyframes example {

0% {left:800px;}

1% {left:792px;}

2% {left:784px;}

3% {left:776px;}

4% {left:768px;}

5% {left:760px;}

6% {left:752px;}

7% {left:744px;}

8% {left:736px;}

9% {left:728px;}

10% {left:720px;}

11% {left:712px;}

12% {left:704px;}

13% {left:696px;}

14% {left:688px;}

15% {left:680px;}

16% {left:672px;}

17% {left:664px;}

18% {left:656px;}

19% {left:648px;}

20% {left:640px;}

21% {left:632px;}

22% {left:624px;}

23% {left:616px;}

24% {left:608px;}

25% {left:600px;}

26% {left:592px;}

27% {left:584px;}

28% {left:576px;}

29% {left:568px;}

30% {left:560px;}

31% {left:552px;}

32% {left:544px;}

33% {left:536px;}

34% {left:528px;}

35% {left:520px;}

36% {left:512px;}

37% {left:504px;}

38% {left:496px;}

39% {left:488px;}

40% {left:480px;}

41% {left:472px;}

42% {left:464px;}

43% {left:456px;}

44% {left:448px;}

45% {left:440px;}

46% {left:432px;}

47% {left:424px;}

48% {left:416px;}

49% {left:408px;}

50% {left:400px;}

51% {left:392px;}

52% {left:384px;}

53% {left:376px;}

54% {left:368px;}

55% {left:360px;}

56% {left:352px;}

57% {left:344px;}

58% {left:336px;}

59% {left:328px;}

60% {left:320px;}

61% {left:312px;}

62% {left:304px;}

63% {left:296px;}

64% {left:288px;}

65% {left:280px;}

66% {left:272px;}

67% {left:264px;}

68% {left:256px;}

69% {left:248px;}

70% {left:240px;}

71% {left:232px;}

72% {left:224px;}

73% {left:216px;}

74% {left:208px;}

75% {left:200px;}

76% {left:192px;}

77% {left:184px;}

78% {left:176px;}

79% {left:168px;}

80% {left:160px;}

81% {left:152px;}

82% {left:144px;}

83% {left:136px;}

84% {left:128px;}

85% {left:120px;}

86% {left:112px;}

87% {left:104px;}

88% {left:96px;}

89% {left:88px;}

90% {left:80px;}

91% {left:72px;}

92% {left:64px;}

93% {left:56px;}

94% {left:48px;}

95% {left:40px;}

96% {left:32px;}

97% {left:24px;}

98% {left:16px;}

99% {left:8px;}

100% {left:0px;}

}

h1 {

font-weight: bold;

margin: 0;

}

h2 {

text-align: center;

}

p {

font-size: 14px;

font-weight: 100;

line-height: 20px;

letter-spacing: 0.5px;

margin: 20px 0 30px;

}

.info > p {

font-size: 1.2rem ;

font-weight: 20;

line-height: 30px;

letter-spacing: 1px;

margin: 20px 0 30px;

}

a {

color: #333;

font-size: 14px;

text-decoration: none;

margin: 15px 0;

}

button {

border-radius: 20px;

border: 1px solid #FF4B2B;

background-color: #FF4B2B;

color: #FFFFFF;

font-size: 12px;

font-weight: bold;

padding: 12px 45px;

letter-spacing: 1px;

text-transform: uppercase;

transition: transform 80ms ease-in;

}

button:active {

transform: scale(0.95);

}

button:focus {

outline: none;

}

button.ghost {

background-color: transparent;

border-color: #FFFFFF;

}

form {

background-color: #FFFFFF;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

padding: 0 50px;

height: 100%;

text-align: center;

}

input {

background-color: #eee;

border: none;

padding: 12px 15px;

margin: 8px 0;

width: 100%;

}

.container {

background-color: #fff;

border-radius: 10px;

box-shadow: 0 14px 28px rgba(0,0,0,0.25),

0 10px 10px rgba(0,0,0,0.22);

position: relative;

overflow: hidden;

width: 768px;

max-width: 100%;

min-height: 480px;

}

.form-container {

position: absolute;

top: 0;

height: 100%;

transition: all 0.6s ease-in-out;

}

.sign-in-container {

left: 0;

width: 50%;

z-index: 2;

}

.container.right-panel-active .sign-in-container {

transform: translateX(100%);

}

.sign-up-container {

left: 0;

width: 50%;

opacity: 0;

z-index: 1;

}

.container.right-panel-active .sign-up-container {

transform: translateX(100%);

opacity: 1;

z-index: 5;

animation: show 0.6s;

}

@keyframes show {

0%, 49.99% {

opacity: 0;

z-index: 1;

}

50%, 100% {

opacity: 1;

z-index: 5;

}

}

.overlay-container {

position: absolute;

top: 0;

left: 50%;

width: 50%;

height: 100%;

overflow: hidden;

transition: transform 0.6s ease-in-out;

z-index: 100;

}

.container.right-panel-active .overlay-container{

transform: translateX(-100%);

}

.overlay {

background: #FF416C;

background: -webkit-linear-gradient(to right, #FF4B2B, #FF416C);

background: linear-gradient(to right, #FF4B2B, #FF416C);

background-repeat: no-repeat;

background-size: cover;

background-position: 0 0;

color: #FFFFFF;

position: relative;

left: -100%;

height: 100%;

width: 200%;

transform: translateX(0);

transition: transform 0.6s ease-in-out;

}

.container.right-panel-active .overlay {

transform: translateX(50%);

}

.overlay-panel {

position: absolute;

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

padding: 0 40px;

text-align: center;

top: 0;

height: 100%;

width: 50%;

transform: translateX(0);

transition: transform 0.6s ease-in-out;

}

.overlay-left {

transform: translateX(-20%);

}

.container.right-panel-active .overlay-left {

transform: translateX(0);

}

.overlay-right {

right: 0;

transform: translateX(0);

}

.container.right-panel-active .overlay-right {

transform: translateX(20%);

}

/\*input[type=text], input[type=password] {

width: 100%;

padding: 2px 10px;

margin: 8px 0;

display: inline-block;

border: 0px solid #ccc;

box-sizing: border-box;

border: 1;

}

button {

width: 100%;

margin-top: 30px;

background-color: green;

border: 0;

padding: 5px;

}\*/

import binascii

import math

import random

import requests as res

import secrets

import time

from base64 import urlsafe\_b64encode as b64e, urlsafe\_b64decode as b64d

from time import strftime, localtime

import ibm\_db

import sendgrid

from clarifai\_grpc.channel.clarifai\_channel import ClarifaiChannel

from clarifai\_grpc.grpc.api import resources\_pb2, service\_pb2, service\_pb2\_grpc

from clarifai\_grpc.grpc.api.status import status\_code\_pb2

from cryptography.fernet import InvalidToken

from cryptography.hazmat.backends import default\_backend

fssrom cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

from flask import Flask, render\_template, request, session, redirect

from markupsafe import escape

from sendgrid.helpers.mail import Mail, Email, To, Content

# clarifai

YOUR\_CLARIFAI\_API\_KEY = "xxxxxxxxxxxxxxxxxxxxxx"

YOUR\_APPLICATION\_ID = "xxxxxxxxxxxxxxxxxxxxxxx"

channel = ClarifaiChannel.get\_json\_channel()

stub = service\_pb2\_grpc.V2Stub(channel)

metadata = (("authorization", f"Key {YOUR\_CLARIFAI\_API\_KEY}"),)

# sendgrid

SENDGRID\_API\_KEY = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

# rapid API

url = "https://spoonacular-recipe-food-nutrition-v1.p.rapidapi.com/recipes/parseIngredients"

querystring = {"includeNutrition": "true"}

headers = {"content-type": "application/x-www-form-urlencoded",

           "X-RapidAPI-Key": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

           "X-RapidAPI-Host": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

           }

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'jfif'}

KEY = "24803877913464067088963527689231"

conn = ibm\_db.connect(

    "DATABASE=bludb;HOSTNAME=ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT"

    "=31505;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=xnc98967;PWD=wQioPYLq4Oanh0Vm",

    '', '')

print(conn)

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

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

# sendgrid

def send\_mail(email):

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

    from\_email = Email("xxxxxxxxxxxxxx.foryoy@gmail.com")  # Change to your verified sender

    to\_email = To(email)  # Change to your recipient

    subject = "Nutrition is a basic human need and a prerequisite for healthy life"

    content = Content("text/plain",

                      "Thank you for creating an account on our platform. Now you can utilise our platform "

                      "to maintain a healthier life.")

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

    # Get a JSON-ready representation of the Mail object

    mail\_json = mail.get()

    # Send an HTTP POST request to /mail/send

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

    # print(response.status\_code)

    # print(response.headers)

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

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

    from\_email = Email("xxxxxxxxxx@gmail.com")  # Change to your verified sender

    to\_email = To(email)  # Change to your recipient

    subject = "Nutrition is a basic human need and a prerequisite for healthy life"

    content = Content("text/plain",

                      f"OTP : '{otp}'")

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

    # Get a JSON-ready representation of the Mail object

    mail\_json = mail.get()

    # Send an HTTP POST request to /mail/send

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

    # print(response.status\_code)

    # print(response.headers)

def generateOTP():

    digits = "0123456789"

    OTP = ""

    for i in range(6):

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

    return OTP

def get\_history():

    history = []

    sql = f"SELECT \* FROM PERSON WHERE email = '{session['email']}'"

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

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

    while dictionary:

        history.append(dictionary)

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

    return history

def get\_history\_person(email):

    history = []

    sql = f"SELECT \* FROM PERSON WHERE email = '{email}'"

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

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

    while dictionary:

        history.append(dictionary)

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

    return history

def get\_history\_person\_time(time):

    historys = []

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

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

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

    while dictionary:

        historys.append(dictionary)

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

    return historys

def get\_user():

    user = []

    sql = f"SELECT \* FROM USER"

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

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

    while dictionary:

        user.append(dictionary)

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

    return user

backend = default\_backend()

def aes\_gcm\_encrypt(message: bytes, key: bytes) -> bytes:

    current\_time = int(time.time()).to\_bytes(8, 'big')

    algorithm = algorithms.AES(key)

    iv = secrets.token\_bytes(algorithm.block\_size // 8)

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

    encryptor = cipher.encryptor()

    encryptor.authenticate\_additional\_data(current\_time)

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

    return b64e(current\_time + iv + ciphertext + encryptor.tag)

def aes\_gcm\_decrypt(token: bytes, key: bytes, ttl=None) -> bytes:

    algorithm = algorithms.AES(key)

    try:

        data = b64d(token)

    except (TypeError, binascii.Error):

        raise InvalidToken

    timestamp, iv, tag = data[:8], data[8:algorithm.block\_size // 8 + 8], data[-16:]

    if ttl is not None:

        current\_time = int(time.time())

        time\_encrypted, = int.from\_bytes(data[:8], 'big')

        if time\_encrypted + ttl < current\_time or current\_time + 60 < time\_encrypted:

            # too old or created well before our current time + 1 h to account for clock skew

            raise InvalidToken

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

    decryptor = cipher.decryptor()

    decryptor.authenticate\_additional\_data(timestamp)

    ciphertext = data[8 + len(iv):-16]

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

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

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

def homepage():

    if request.method == 'POST' and 'email' in request.form and 'pass' in request.form:

        error = None

        username = request.form['email']

        password = request.form['pass']

        user = None

        if username == "":

            error = 'Incorrect username.'

            return render\_template('index.html', error=error)

        if password == "":

            error = 'Incorrect password.'

            return render\_template('index.html', error=error)

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

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt, 1, username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        if account:

            print(aes\_gcm\_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')))

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

            if aes\_gcm\_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')) == bytes(password, 'utf-8'):

                user = account['NAME']

                email = account["EMAIL"]

                session["loggedIn"] = None

                session['name'] = user

                session['email'] = email

                msg = None

                history = get\_history()  # end of user

                list = get\_user()

                return render\_template('adminpanal.html', user=user, list=list, email=email, msg=msg)

            return render\_template('index.html', error="Wrong Password!")

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

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt, 1, username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        if not account:

            return render\_template('index.html', error="Username not found!")

        print(aes\_gcm\_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')))

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

        if aes\_gcm\_decrypt(account['PASSWORD'], bytes(KEY, 'utf-8')) == bytes(password, 'utf-8'):

            user = account['NAME']

            email = account["EMAIL"]

            session["loggedIn"] = 'loggedIn'

            session['name'] = user

            session['email'] = email

            msg = None

            history = get\_history()  # end of user

            list = get\_user()

            return render\_template('dashboard.html', user=user, email=email, msg=msg, history=history)

        return render\_template('index.html', error="Wrong Password!")

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

        sql = f"SELECT \* FROM PERSON WHERE email='{session['email']}'"

        print(sql)

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

        list\_of\_history = ibm\_db.fetch\_row(stmt)

        if list\_of\_history:

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

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

            history = get\_history()

            if history:

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

                                       email=session['email'])

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

                               email=session['email'])

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

        session["loggedIn"] = None

        session['name'] = None

        session['email'] = None

        return render\_template('index.html', error="Successfully Logged Out!")

    elif request.method == 'POST' and 'extra\_submit\_param\_view' in request.form:

        nutrition\_list = request.form["extra\_submit\_param\_view"]

        history = get\_history()

        splitted\_nutrition = nutrition\_list.split(",")

        return render\_template('dashboard.html', user=session['name'], email=session['email'], data=splitted\_nutrition,

                               history=history)

    elif request.method == 'POST' and 'extra\_submit\_param\_delete' in request.form:

        time\_identity = request.form["extra\_submit\_param\_delete"]

        history = get\_history()

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

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

        row = ibm\_db.fetch\_row(stmt)

        if row:

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

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

            history = get\_history()

            if history:

                return render\_template("dashboard.html", history=history, msg="Delete successfully")

            return render\_template("dashboard.html", msg="Delete successfully")

        return render\_template("dashboard.html", history=history, msg="Something went wrong, Try again")

    elif request.method == 'POST' and 'extra\_submit\_param\_record' in request.form:

        email\_user = request.form["extra\_submit\_param\_record"]

        return render\_template('adminpanal.html', user=session['name'], email=session['email'], list=get\_user(),

                               history=get\_history\_person(email\_user))

    elif request.method == 'POST' and 'extra\_submit\_param\_delete\_user' in request.form:

        email\_user = request.form["extra\_submit\_param\_delete\_user"]

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

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

        row = ibm\_db.fetch\_row(stmt)

        if row:

            sql = f"DELETE FROM USER WHERE time='{escape(email\_user)}'"

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

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

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

        row = ibm\_db.fetch\_row(stmt)

        if row:

            sql = f"DELETE FROM PERSON WHERE time='{escape(email\_user)}'"

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

        return render\_template('adminpanal.html', user=session['name'], list=get\_user())

    elif request.method == 'POST' and 'extra\_submit\_param\_nutritions' in request.form:

        user\_time = request.form["extra\_submit\_param\_nutritions"]

        user\_of = get\_history\_person\_time(user\_time)

        user\_dic = user\_of[0]

        splitted\_nutrition = user\_dic['NUTRITION'].split(",")

        return render\_template('adminpanal.html', user=session['name'], list=get\_user(),

                               history=get\_history\_person(user\_dic["EMAIL"]), data=splitted\_nutrition)

    elif request.method == 'POST' and 'extra\_submit\_param\_delete\_record' in request.form:

        email\_user = request.form["extra\_submit\_param\_delete\_record"]

        user\_of = get\_history\_person\_time(email\_user)

        user\_dic = user\_of[0]

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

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

        row = ibm\_db.fetch\_row(stmt)

        if row:

            sql = f"DELETE FROM PERSON WHERE time='{escape(email\_user)}'"

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

        return render\_template('adminpanal.html', user=session['name'], list=get\_user(),

                               history=get\_history\_person(user\_dic["EMAIL"]))

    elif session.get('loggedIn'):

        history = get\_history()

        return render\_template('dashboard.html', user=session['name'], history=history)

    return render\_template('index.html')

def allowed\_file(filename):

    return '.' in filename and \

           filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

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

def upload\_file():

    history = []

    # sql = "SELECT \* FROM Students"

    sql = f"SELECT \* FROM PERSON WHERE email = '{session['email']}'"

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

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

    while dictionary:

        history.append(dictionary)

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

    if request.method == 'POST':

        # check if the post request has the file part

        if 'logout' in request.form:

            session["loggedIn"] = None

            session['name'] = None

            session['email'] = None

            return render\_template('index.html', error="Successfully created")

        if 'file' not in request.files:

            # flash('No file part')

            return redirect(request.url)

        file = request.files['file']

        # If the user does not select a file, the browser submits an

        # empty file without a filename.

        if file.filename == '':

            return render\_template('dashboard.html', msg="File not found", history=history)

        baseimage = file.read()

        if file and allowed\_file(file.filename):

            requests = service\_pb2.PostModelOutputsRequest(

                # This is the model ID of a publicly available General model. You may use any other public or custom

                # model ID.

                # model\_id="general-image-recognition"

                # model\_id="food-item-recognition"

                model\_id="food-item-recognition",

                user\_app\_id=resources\_pb2.UserAppIDSet(app\_id=YOUR\_APPLICATION\_ID),

                inputs=[

                    resources\_pb2.Input(

                        data=resources\_pb2.Data(image=resources\_pb2.Image(base64=baseimage))

                    )

                ],

            )

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

            if response.status.code != status\_code\_pb2.SUCCESS:

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

            calcium = 0

            vitaminb5 = 0

            protein = 0

            vitamind = 0

            vitamina = 0

            vitaminb2 = 0

            carbohydrates = 0

            fiber = 0

            fat = 0

            sodium = 0

            vitaminc = 0

            calories = 0

            vitaminb1 = 0

            folicacid = 0

            sugar = 0

            vitamink = 0

            cholesterol = 0

            potassium = 0

            monounsaturatedfat = 0

            polyunsaturatedfat = 0

            saturatedfat = 0

            totalfat = 0

            calciumu = 'g'

            vitaminb5u = 'g'

            proteinu = 'g'

            vitamindu = 'g'

            vitaminau = 'g'

            vitaminb2u = 'g'

            carbohydratesu = 'g'

            fiberu = 'g'

            fatu = 'g'

            sodiumu = 'g'

            vitamincu = 'g'

            caloriesu = 'cal'

            vitaminb1u = 'g'

            folicacidu = 'g'

            sugaru = 'g'

            vitaminku = 'g'

            cholesterolu = 'g'

            potassiumu = 'g'

            monounsaturatedfatu = 'g'

            polyunsaturatedfatu = 'g'

            saturatedfatu = 'g'

            totalfatu = 'g'

            for concept in response.outputs[0].data.concepts:

                print("%12s: %.2f" % (concept.name, concept.value))

                if concept.value > 0.5:

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

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

                    data = response1.json()

                    for i in range(0, 1):

                        nutri\_array = data[i]

                        nutri\_dic = nutri\_array['nutrition']

                        nutri = nutri\_dic['nutrients']

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

                            temp = nutri[z]

                            if temp['name'] == 'Calcium':

                                calcium += temp['amount']

                                calciumu = temp['unit']

                            elif temp['name'] == 'Vitamin B5':

                                vitaminb5 += temp['amount']

                                vitaminb5u = temp['unit']

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

                                protein += temp['amount']

                                proteinu = temp['unit']

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

                                vitamind += temp['amount']

                                vitamindu = temp['unit']

                            elif temp['name'] == 'Vitamin A':

                                vitamina += temp['amount']

                                vitaminau = temp['unit']

                            elif temp['name'] == 'Vitamin B2':

                                vitaminb2 += temp['amount']

                                vitaminb2u = temp['unit']

                            elif temp['name'] == 'Carbohydrates':

                                carbohydrates += temp['amount']

                                carbohydratesu = temp['unit']

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

                                fiber += temp['amount']

                                fiberu = temp['unit']

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

                                vitaminc += temp['amount']

                                vitamincu = temp['unit']

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

                                calories += temp['amount']

                                caloriesu = 'cal'

                            elif temp['name'] == 'Vitamin B1':

                                vitaminb1 += temp['amount']

                                vitaminb1u = temp['unit']

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

                                folicacid += temp['amount']

                                folicacidu = temp['unit']

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

                                sugar += temp['amount']

                                sugaru = temp['unit']

                            elif temp['name'] == 'Vitamin K':

                                vitamink += temp['amount']

                                vitaminku = temp['unit']

                            elif temp['name'] == 'Cholesterol':

                                cholesterol += temp['amount']

                                cholesterolu = temp['unit']

                            elif temp['name'] == 'Mono Unsaturated Fat':

                                monounsaturatedfat += temp['amount']

                                monounsaturatedfatu = temp['unit']

                            elif temp['name'] == 'Poly Unsaturated Fat':

                                polyunsaturatedfat += temp['amount']

                                polyunsaturatedfatu = temp['unit']

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

                                saturatedfat += temp['amount']

                                saturatedfatu = temp['unit']

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

                                fat += temp['amount']

                                fatu = temp['unit']

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

                                sodium += temp['amount']

                                sodiumu = temp['unit']

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

                                potassium += temp['amount']

                                potassiumu = temp['unit']

                            else:

                                pass

            totalfat += saturatedfat + polyunsaturatedfat + monounsaturatedfat

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

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

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

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

            to\_string = "{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}".format(data[0], data[1], data[2], data[3],

                                                                                 data[4],

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

                                                                                 data[9],

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

                                                                                 data[14], data[15])

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

            sql = "SELECT \* FROM PERSON"

            stmt = ibm\_db.prepare(conn, sql)

            # ibm\_db.bind\_param(stmt, 1, session['email'])

            ibm\_db.execute(stmt)

            # account = ibm\_db.fetch\_assoc(stmt)

            try:

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

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

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

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

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

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

                ibm\_db.execute(prep\_stmt)

                return render\_template('dashboard.html', user=session['name'], email=session['email'], data=data,

                                       history=history, unit=unit)

            except ibm\_db.stmt\_error:

                print(ibm\_db.stmt\_error())

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

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

        return render\_template('dashboard.html', history=history)

    if session['name'] is None:

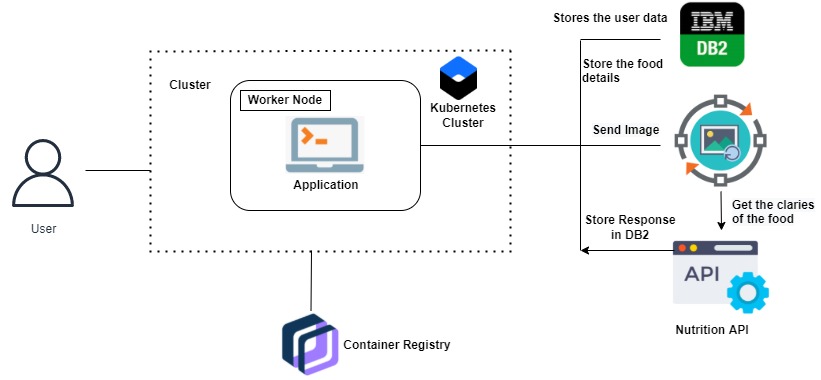
        return render\_template('index.html')

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

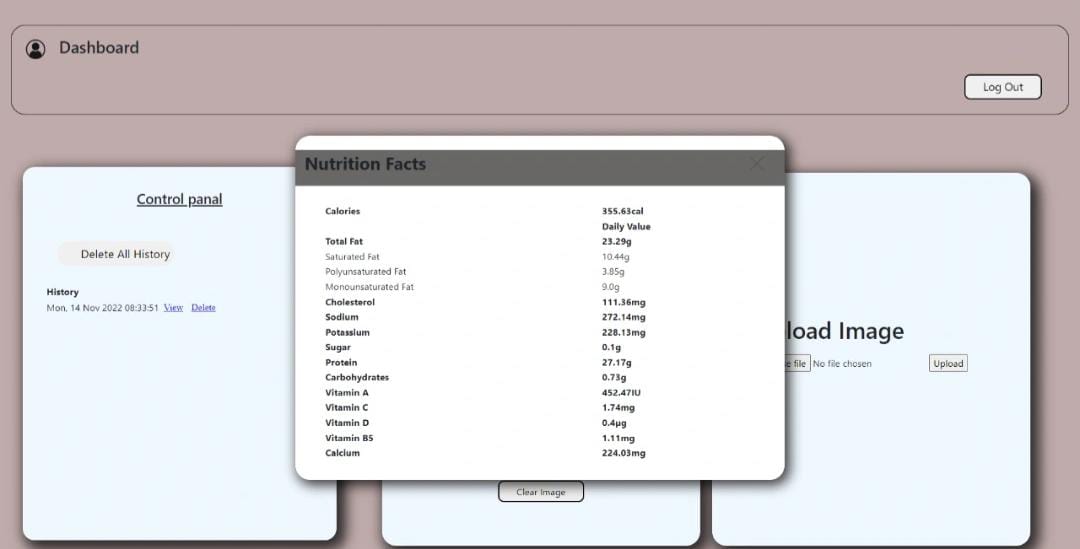
if \_\_name\_\_ == '\_\_main\_\_':

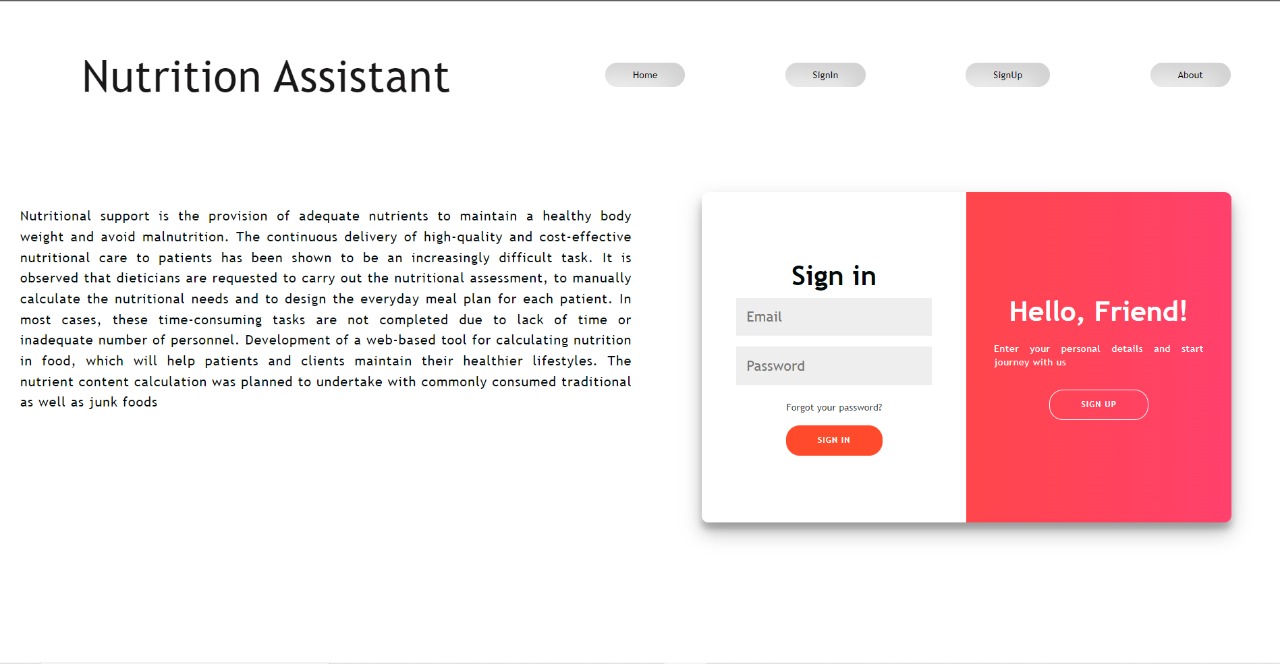
    app.debug = True

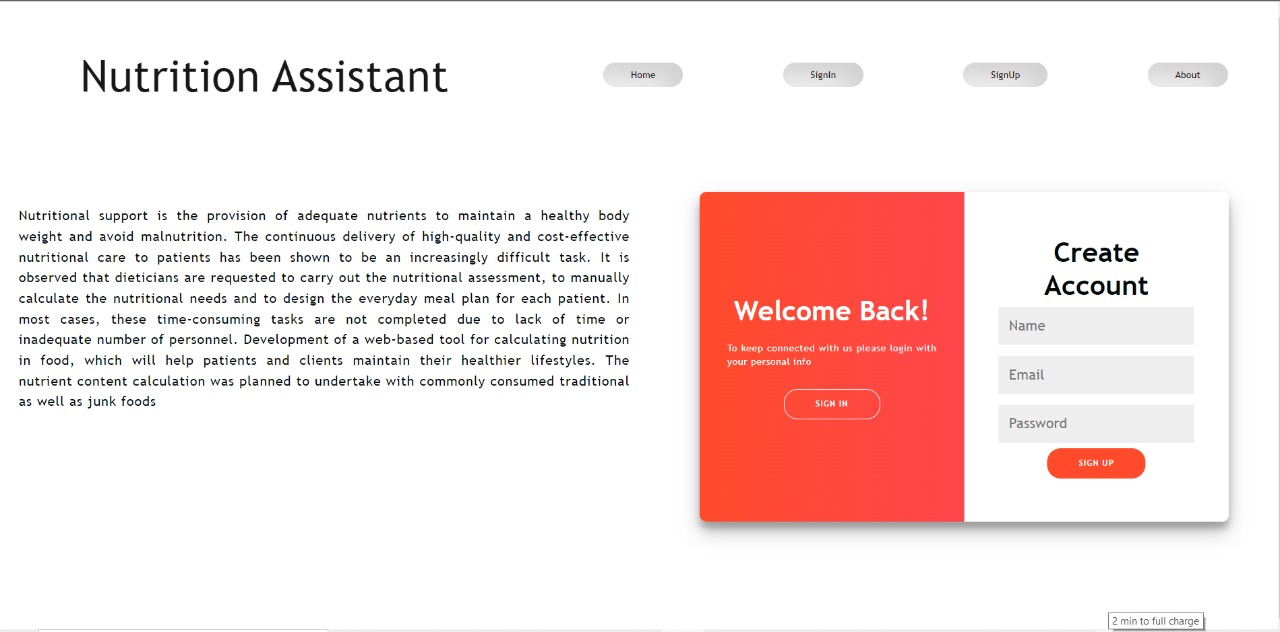
    app.run()

****

****

****



****

****

**9. RESULTS**

As our results have shown that participants specifically appreciated the visual feedback, we suggest that recommender systems should consider visual feedback as an integral part to serve awareness, reflection on behaviour, and educational content to enhance nutrition-related knowledge. Further investigations into the potential of automated personalized feedback towards users’ action  would be helpful to determine which visual feedback serves users with the most decisional support. The personalized recommendations provided diverse daily sets of recipes but were repetitive over time and could not always meet user preferences. The repetitiveness is partially attributed to the dependency on users’ food diaries, which were sometimes incomplete due to high tracking effort. Alternative methods to elicit user information could help mitigate this issue if they are less laborious for users but might come with a decrease in accuracy.

**10. ADVANTAGES AND DISADVANTAGES**

**ADVANTAGES:**

### They Can Help You Eat Healthier

### They can help you improve your diet so you are eating healthier. Using a nutrition app will inherently help you eat a better and more optimized diet. Not only do these apps typically have calorie counters in them, but they can show you how many macro and micronutrients you are getting into your diet.

### Track Your Progress

### One of the most important things that you can do when you are looking to improve your diet choices and your weight loss efforts is to track your progress. This is something that you will either need to do manually or in an automated fashion using an app like this.

### Some Can be Used for Free

### You will find a lot of nutrition apps that are completely free to use. You don’t necessarily have to use one that is paid. There are a lot of free apps that are readily available across all of the major mobile stores. Some of them will include premium features that are blocked off to free users.

### DISADVANTAGES:

### 1.Inaccuracy - One of the downsides to using an app for calorie counting is the fact that it’s not always going to be accurate. Whether you mistype som ething or it tracks incorrectly, it can completely throw off your true macronutrient count.

### 2. Time-Consuming - Another con associated with using and relying on these apps has to do with the time-consuming nature of logging all of your data. Many people won’t want to spend more than a couple of minutes logging their information.

### 3. It Can Get Addicting - One of the other downsides to these types of apps is that they can promote addictive behavior. While there are certainly worse things to be addicted to than nutrition apps, that doesn’t mean it’s necessarily a good thing.

### 11. CONCLUSION:

### The application comes in response to a current problem that is increasingly common - physical appearance. Control over our diet makes an important contribution to our physical and mental state. By saying this we mean avoiding certain food intolerances, frequent mood swings, or storing energy in larger quantities. The application has an interface that can be easily interacted with. It has combined the features that are most popular and used in the process of weight loss or weight gain. It is based on a database large enough to demonstrates purpose and workﬂow.

### 12. FUTURE SCOPE

### Nutrition apps market is expected to grow in the forecast period of 2021 to 2028. Data Bridge Market Research analyses that the market is growing with a CAGR of 30.4% in the forecast period of 2021 to 2028 and is expected to reach USD 17,450.56 million by 2028. Nutrition apps are software applications used for tracking nutritional intakes and managing diets for healthy eating, weight loss, weight maintenance, weight gain, and fitness. Mobile phone applications are used as tools by many users to get help in their daily tasks. Increasing awareness about health disorders due to unhealthy lifestyles has led to increased diet and nutrition apps.