Project Report

Cloud based Nutrition Assistant Application

Team - Id : PNT2022TMID53138

NUTRITION ASSISTANT APPLICATION

1. INTRODUCTION

1.1 Project Overview

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 Application that automatically estimates food attributes such as ingredients and nutritional value like - calories, proteins, carbohydrates etc., by classifying the input image of food and providing its nutritional values. Our method employs "Clarifai's AI-Driven Food Detection Model" for accurate food identification to give the nutritional value of the identified food. Clarifai AI-Driven Food Detection Model is an API that classifies the ingredients of the meal and provide the name of the meal. That name will be provided as an input to the Nutrition API which provides the nutritional value of the identified food.

Work Flow of the Project:

- ➢ User should create an account using mail and receive a confirmation mail.
- ➢ User can calculate his BMI by providing height and weight.
- ➢ User interacts with the Web App to Load an image.
- ➢ The image is passed to the server application, which uses Clarifai's AI-Driven Food Detection Model Service to 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 fast paced environment we live in and the ignorance of physical health, diseases and obesity rates are increasing at an exponential rate. Awareness and habits of tracking calorie intake must be built to lead a disease free lifestyle. However, food packaging comes with nutrition (and calorie) labels, it is very difficult to keep track of food ingredients we consume. Having an application which can analyze real-time images of a meal and give its nutritional content can be very useful and makes us self conscious of the food we eat. It ultimately improves the dietary habits, and therefore, helps in maintaining a healthy lifestyle.

There are many applications currently that suggest various diet plans for users to follow. However, most of them don't cater to what a user particularly needs. The application gives general diet plans. This doesn't actually work perfectly in day to day scenarios where users actually might not have the correct access to the mentioned food and are going for alternatives which might change the actual nutrition value. So we would want to modify this by taking into account what types of food the user usually wants to have and make sure we give suggestions on the nutrition value for it and suggest on the same basis.

Basically, our nutrition assistant application comes with lots of benefits.

It helps users:

⇒ To keep track of daily intake.
⇒ to monitor calories intake.
⇒ To provide a facility to upload meal images.
⇒ To get the nutritional value of the uploaded image and to keep track of BMI.

2. LITERATURE SURVEY

2.1 Existing solutions

1) **MY FITNESS PAL:**

Proposed in the year 2005 by Dr. Mike Lee and Albert Lee.

,

Mobile applications that count calories, such as My Fitness Pal, are frequently employed on a daily basis. Recent research has shown that in undergraduates, calorie tracking is associated with eating disorder pathology. In the current study (N = 105 individuals diagnosed with an eating disorder), we assessed usage of My Fitness Pal to track calories.

We also assessed perceptions that My Fitness Pal contributed to eating disorder symptoms and if these perceptions were associated with eating disorder symptoms.

We found that a substantial percentage (~ 75%) of participants used Fitness Pal and that 73% of these users perceived the app as contributing to their eating disorder. Furthermore, we found that these perceptions were correlated with eating disorder symptoms.

\ This research suggests that My Fitness Pal is widely used in an eating disorder population and is perceived as contributing to eating disorder symptoms. MyFitnessPal's community aspect basically consists of a forum, where other fellow users of the app are free to exchange tips and advice, as well as to create relationships through sharing personal experiences or struggles.

MyFitnessPal has more than 350 exercises stored in its database, and it shows how much each person burns during each activity, based on their specific height, weight, and gender. It includes most cardio and strength training workouts, as well as yoga and Pilates. The free application is available for Blackberry, Android, Windows, and the iPhone.

Its main objectives are to track weight and recommends calorie intake. This application has one of the largest food databases in the diet tracker ecosystem. It also has extensive recipe and exercise databases. This app helps users to log and count calories, track exercises , view weight loss progress and many other features. One of the biggest disadvantages of this system is that there is no significant difference between intervention and control groups in weight changes.

2) PERSONALIZED DIETARY ASSISTANT

As the Internet gains dominance as the primary source of information in the daily life of people, it is naturally among the first places one would start looking for such information, although numerous online sources have been shown to lack accuracy considering dietary guidelines.

Nowadays, there are numerous types of diets that aim to improve the quality of life, health and longevity of people. However, these diets typically involve a strictly planned regime, which can be hard to get used to or even to follow through at all, due to the sudden nature of the change.

In this paper, the framework for an Intelligent Space application is proposed that helps its users to achieve a healthier diet in the long term by introducing small, gradual changes into their consumption habits. The application observes the daily nutrition intake of its users, applies data mining in order to learn their personal tastes, and educates them about the effects of their current diet on their health.

Then it analyzes the knowledge base to find different food or drink items that align with the perceived preferences, while also adding to the balance of the daily nutrition of the users considering their physical properties, activities, and health conditions (e.g. diabetes, celiac disease, food allergies, etc). Finally, the system uses the findings to make suggestions about adding items from the consumption list, or change one item to another.

3) NUTRITRACK: ANDROID-BASED FOOD RECOGNITION APP FOR NUTRITION AWARENESS

The use of smartphone technology has created new opportunities for people to be aware about health and wellness using diet monitoring applications. Proliferation of such applications have been manifested in the society and that using a smartphone and mobile technology nowadays become universal. One of the emergent concerns of human life is about health and wellness.

Undeniably, health and nutrition are one of the valuable aspects of life. Thus, technological innovations to help enhance and even promote health awareness is essential. With the advent of mobile computing, it is much easier to be aware of health information because of its mobility and availability. Many mobile applications are being developed to serve as a tool for health monitoring and nutritional guidance.

Mobile applications have the ability to support health needs like detecting heart rate, classifying food, and many more. Taking advantage of technology, utilization of it hereby addresses certain issue and problems of human life, especially in health. In this study, the researcher attempts to design and develop an Android-based food recognition application that could be used as a health awareness tool for non-health conscious individual.

The application lets the user take the photo of the food and show its nutritional contents. Implementing Mifflin-St Jeor method in determining daily calorie consumption, users shall be aware of their required calorie intake. Moreover, the researchers' have studied its effect on people's health awareness on food nutrition by the randomly selected respondents. Finally, this paper presents an analysis of the impact of the food recognition app to change people's concept of food nutrition.

2.2 References

1) https://www.myfitnesspal.com/

2) https://www.nutriassistant.com

3) 1. T. Ege and K. Yanai, “Image-based food calorie estimation using knowledge on food categories, ingredients and cooking directions,” in Proc. Thematic Workshops ACM Multimedia-Thematic Workshops, 2017, pp. 367–375.

4) J. Cade, “Evaluation of new technology-based tools for dietary intake assessment—An ilsi europe dietary intake and exposure task force evaluation,” Nutrients, vol. 11, no. 1, p. 55, 2019.

2.3 Problem Statement Definition

Many people have their own methods or apps to analyze their daily intake of nutrition, which they feel is one of the main factors for maintaining a healthy body and one of the important steps among many towards fitness. It is a good habit for a person to record daily intake of nutrition but due to unawareness and lack of proper applications to suit their privacy, lacking proper predefined plans based on actual data of nutrition present in various foods, they tend to either give up, or use methods which are not helpful. Due to the lack of a tracking system, there is a constant struggle to properly know the necessary amount of nutrition needed and the amount we intake, then the total estimation till the end of a certain period.

We aim to develop a web application that automatically estimates food attributes such as ingredients and nutritional value by classifying the input image of food provided.

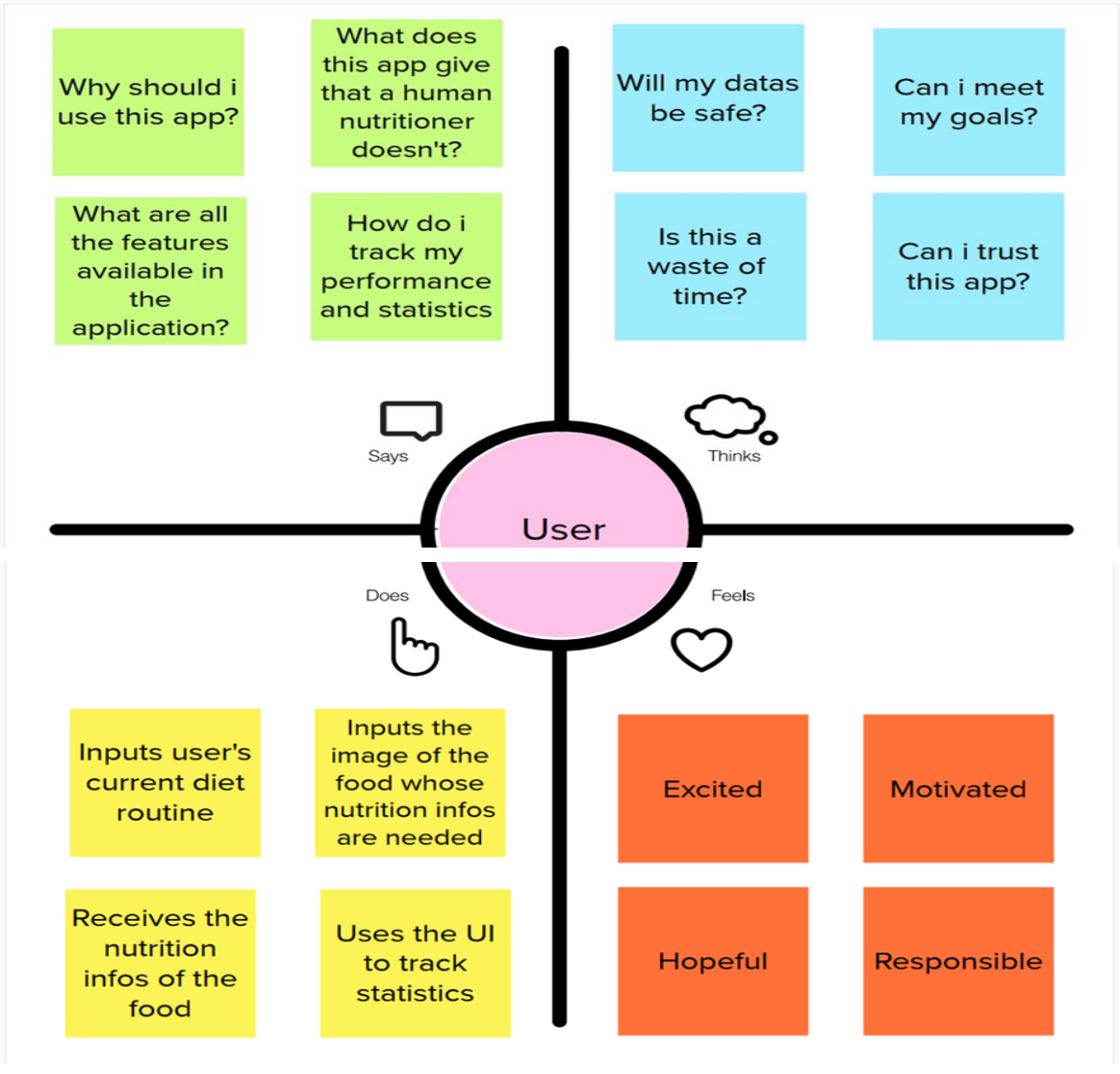
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Empathy Map

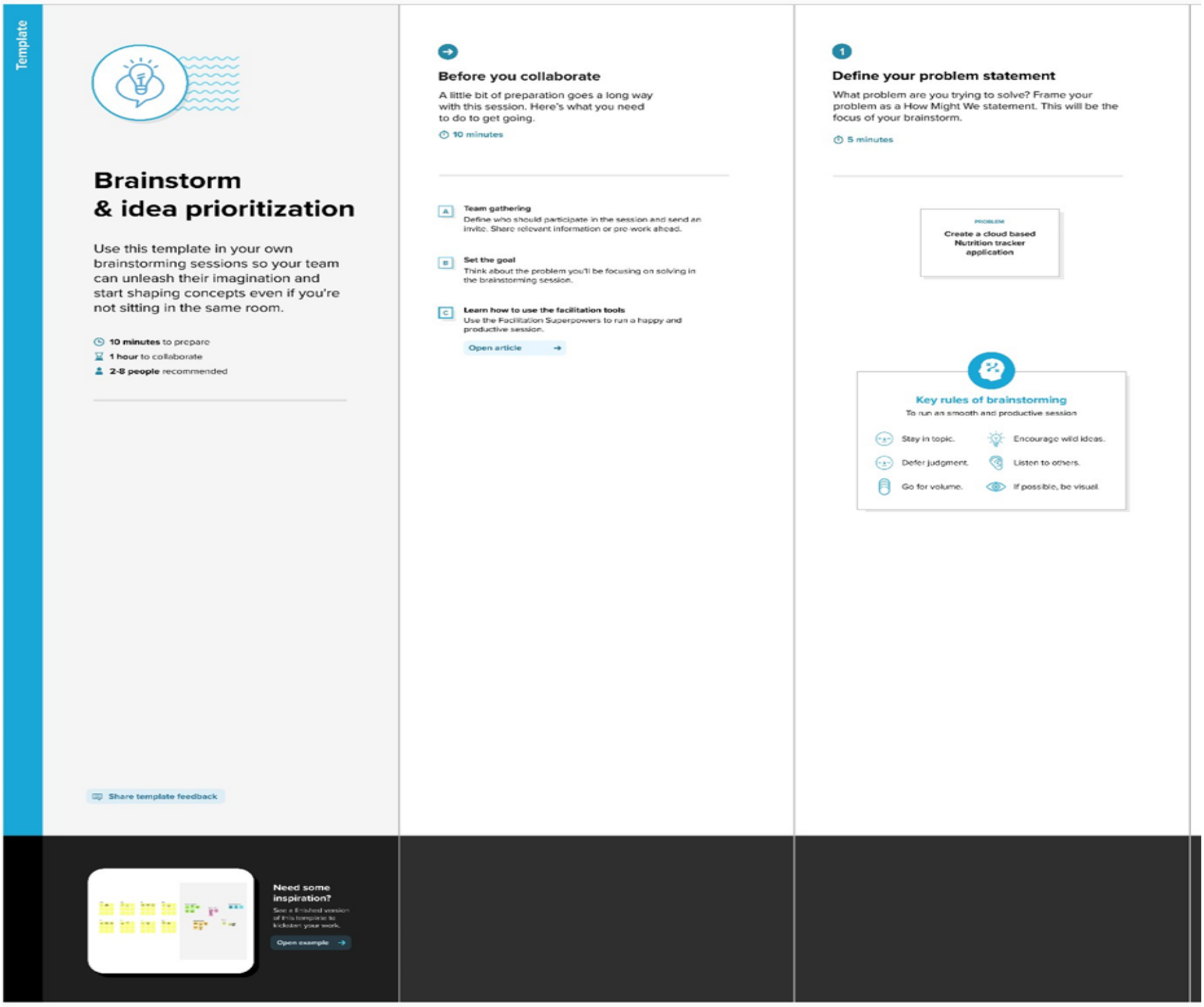
Dive into the mind of the user for focused product development

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



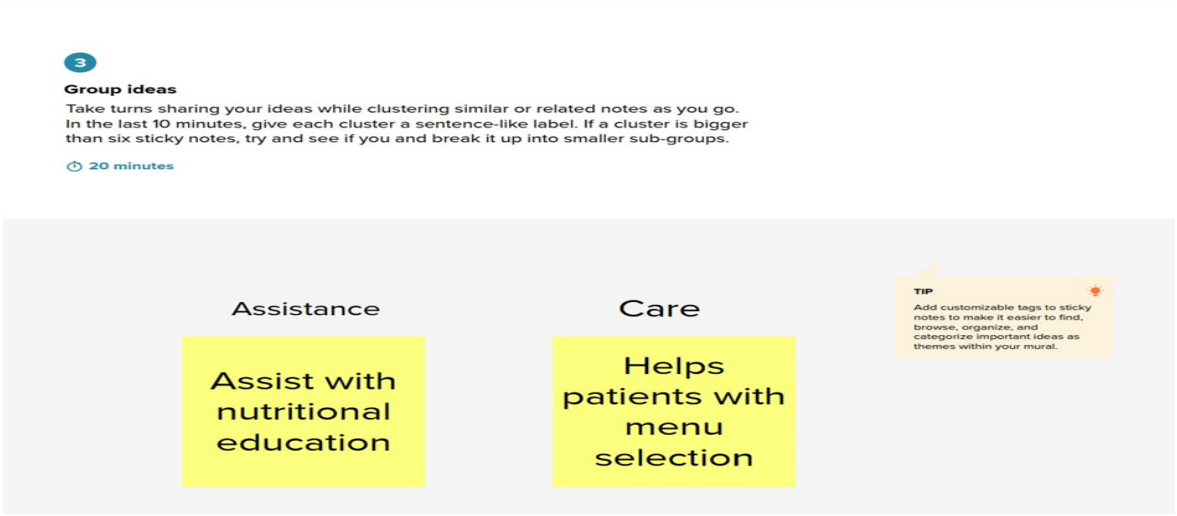
3.2 Ideation & Brainstorming

Step-1: Team Gathering, Collaboration and select the Problem Statement.

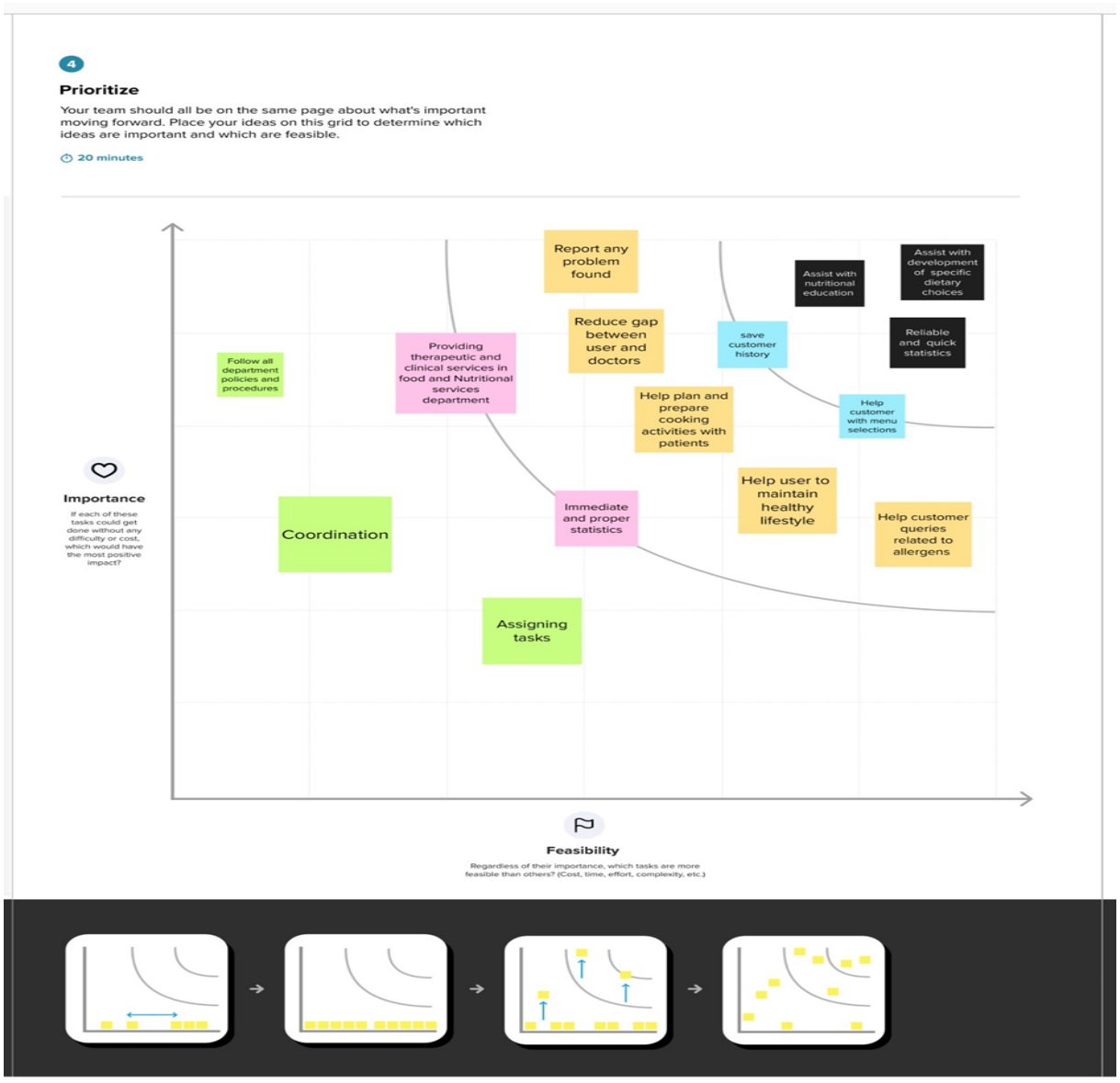


Step-2: Brainstorm, Idea Listing and Grouping

Step 3: Grouping ideas



Step-4: Idea Prioritization



3.3 Proposed Solution

1) **PROBLEM STATEMENT (PROBLEM TO BE SOLVED)**

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.

2) IDEA / SOLUTION DESCRIPTION

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

3) NOVELTY / UNIQUENESS

In this project, 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.

4) SOCIAL IMPACT / CUSTOMER SATISFACTION

It helps people with obesity by suggesting daily calorie intake and nutritional content of the food to achieve their goal of weight loss or gain.

5) BUSINESS MODEL (REVENUE MODEL)

By collaborating with many restaurants and hotel chains, we can provide them the information of the food which is very frequently scanned by the users and we can also decide the favourite food of people in different age categories.

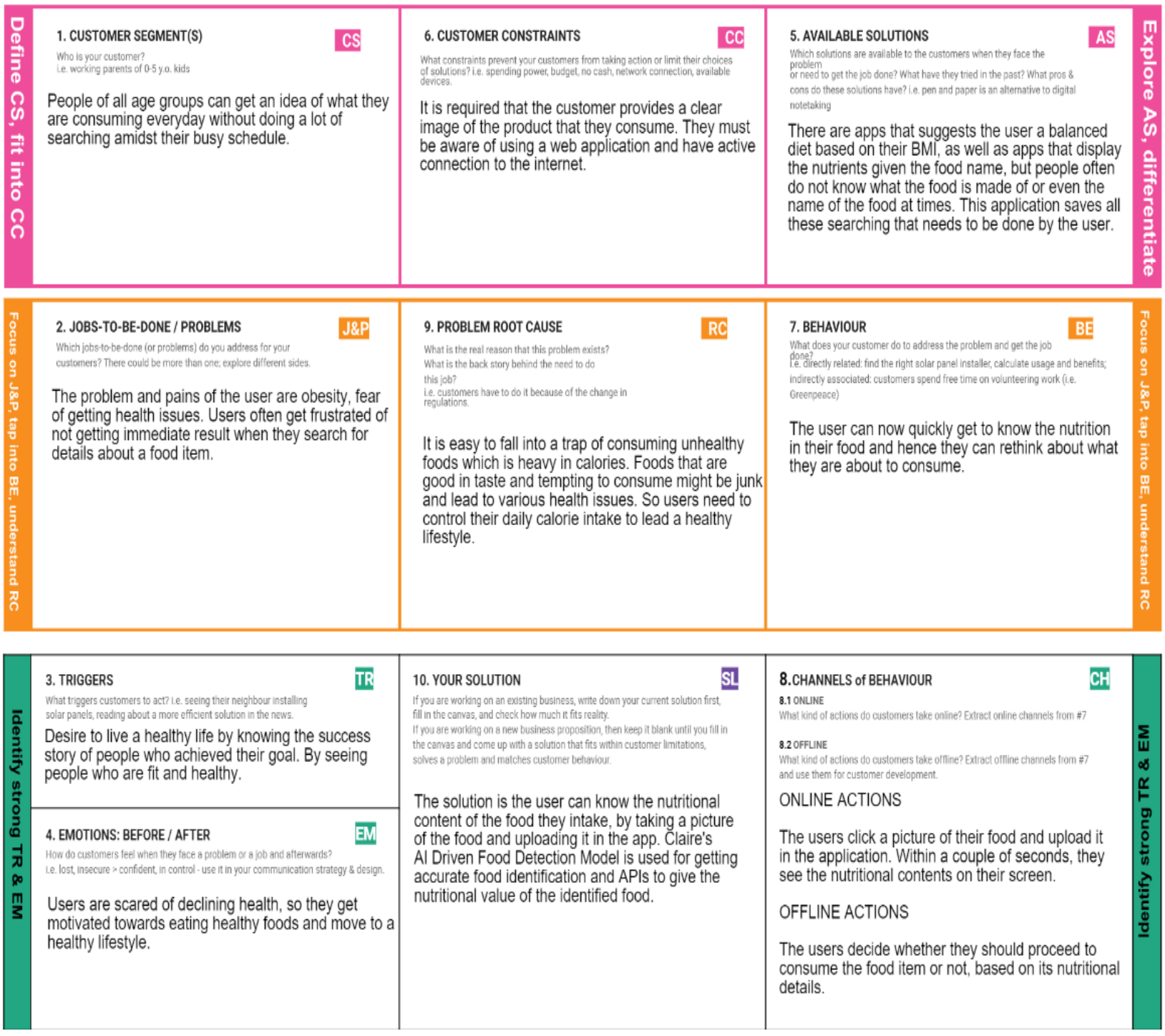
Subscription based models can be implemented which will give the users more detailed information and food recommendations.

6) SCALABILITY OF THE SOLUTION

While developing the application, as developers we kept in mind the scalability factor and made sure that our application would have large scalability. We have used Python Flask server as our backend which is one of the best when it comes to scalability. Flask by itself is only limited in terms of scaling by your application code, the data store you want to use and the Python implementation and web server you are running on. Moreover we have used IBM DB2 as our database

Db2 data sharing provides flexibility for growth and workload balancing. With the partitioned data approach to parallelism (sometimes called the shared-nothing architecture), a one-to-one relationship exists between a particular DBMS and a segment of data.

3.4 Problem Solution fit



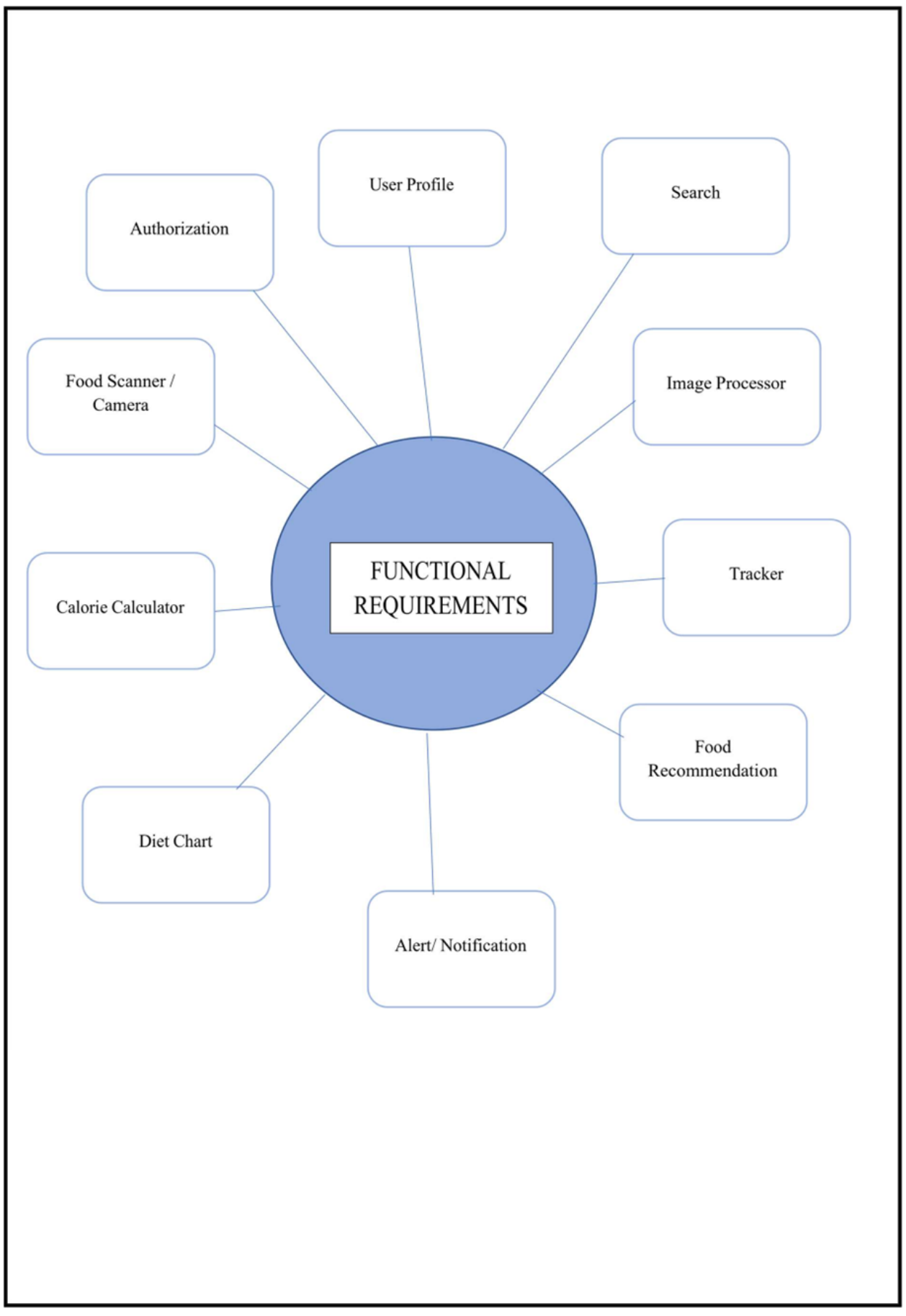
4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the major functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration / Authorisation	Registration through G-Mail Login through G-Mail
FR-2	Food Scanner	Scan the food based on image
FR-3	Calorie Calculator	Get all components of food Calculate calorie and nutritional values
FR-4	Image Processor	Input image Get various sub-components

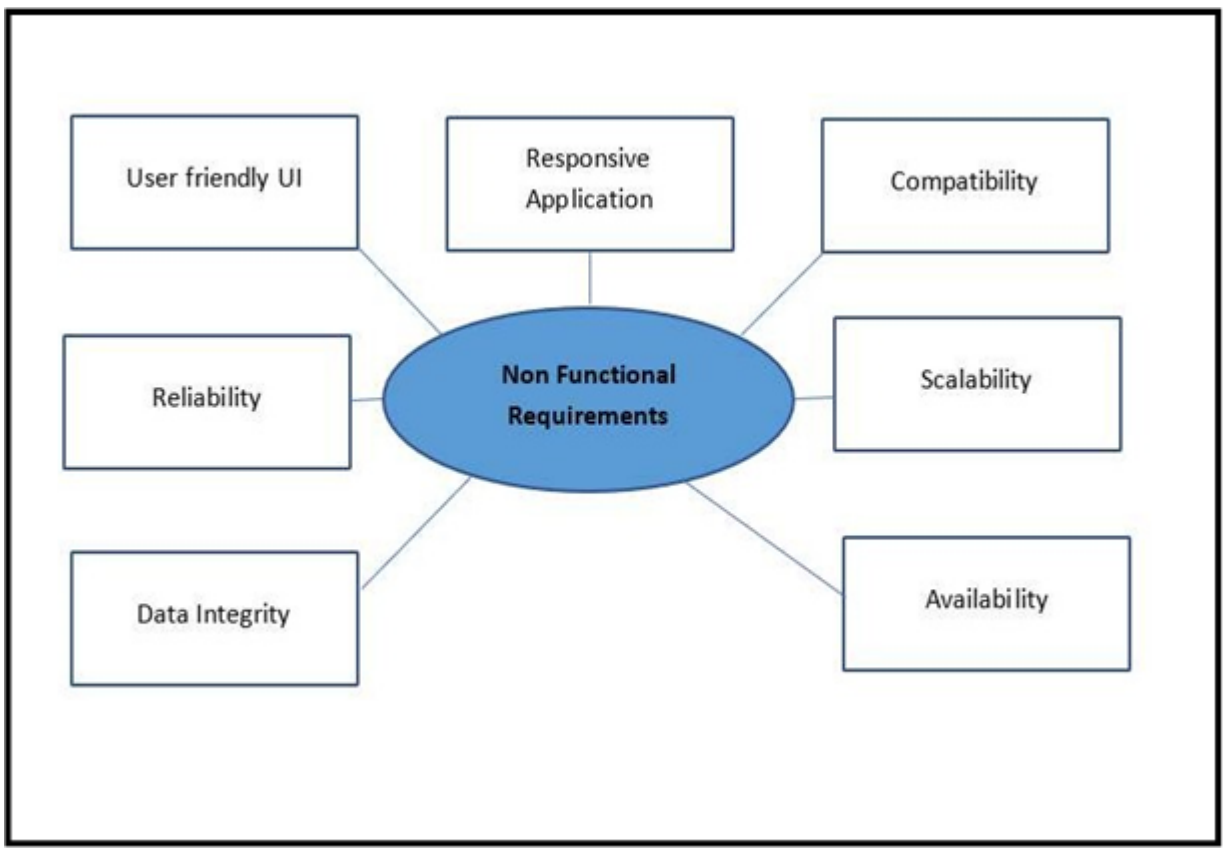
FR-5	Alert	Alert with notification for user
FR-6	Diet Recommender	Diet chart with good recommendation
FR-7	Food Tracker	Keep track of the individual food habits



4.2 Non-Functional requirements

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

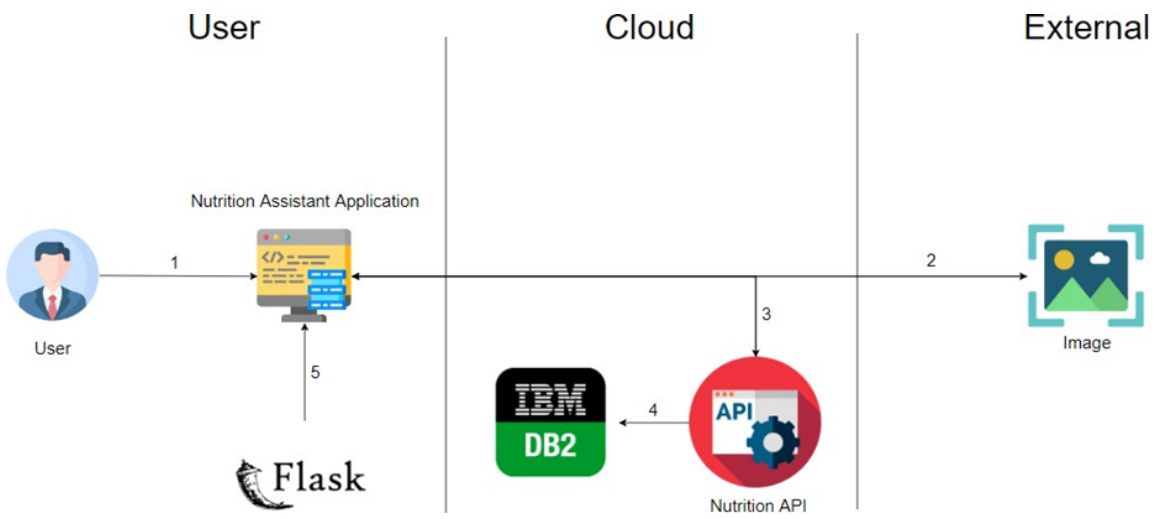
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User-friendly UI
NFR-2	**Security**	Data integrity and safeness of data in DB
NFR-3	**Reliability**	The application will be consistent and reliable
NFR-4	**Performance**	Fast image processing and nutrition extraction Quick to recommend food and alert the users.
NFR-5	**Availability**	Available at all time for usage
NFR-6	**Scalability**	Large number of users can be able to use at the same time with proper functioning.



5. PROJECT DESIGN

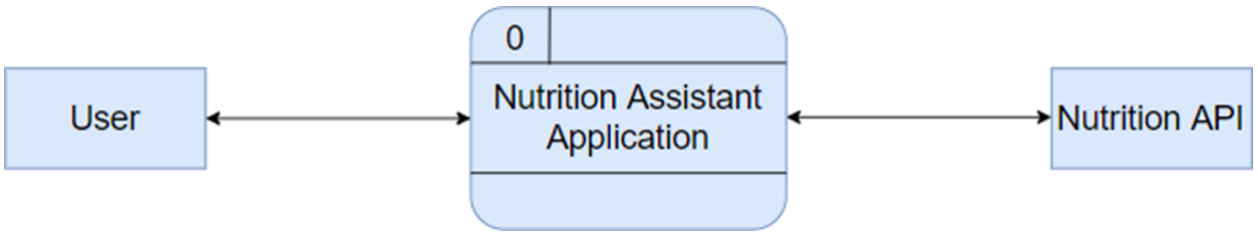
5.1 Data Flow Diagrams

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

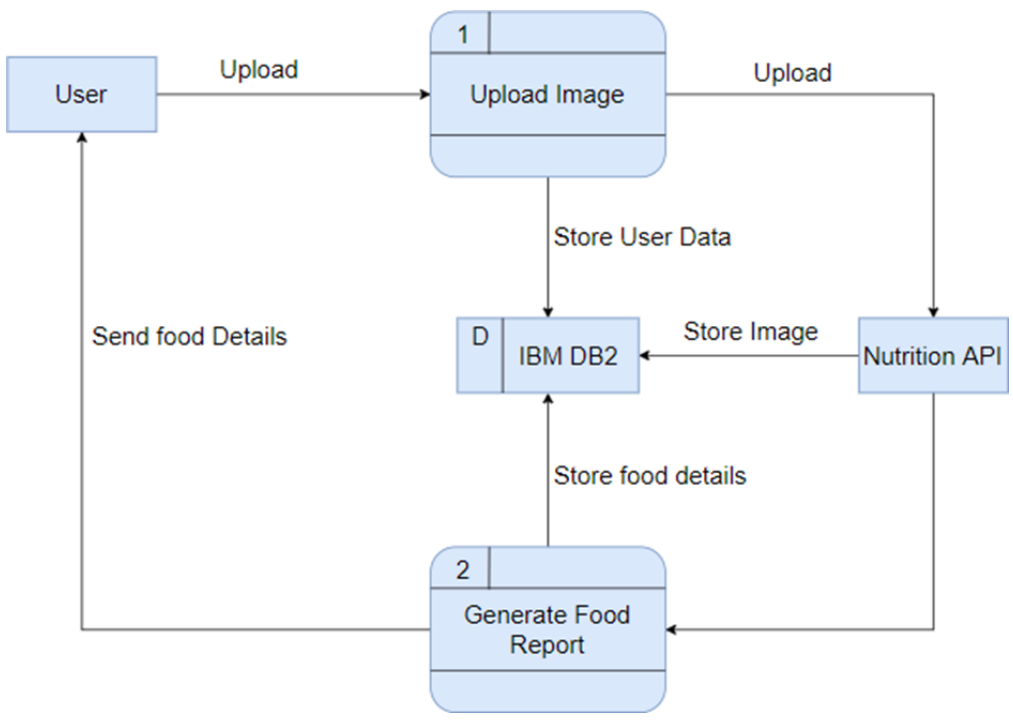


1. The User configures credentials for the Nutrition Assistant Application and starts the app.
2. User takes a picture of the food item and uploads it in the application.
3. Image is sent to the Nutrition API, which analyzes the calories present in the food.
4. User data as well as food details are stored in the IBM DB2 database and sent back to the user.
5. Data is visualized using Flask.

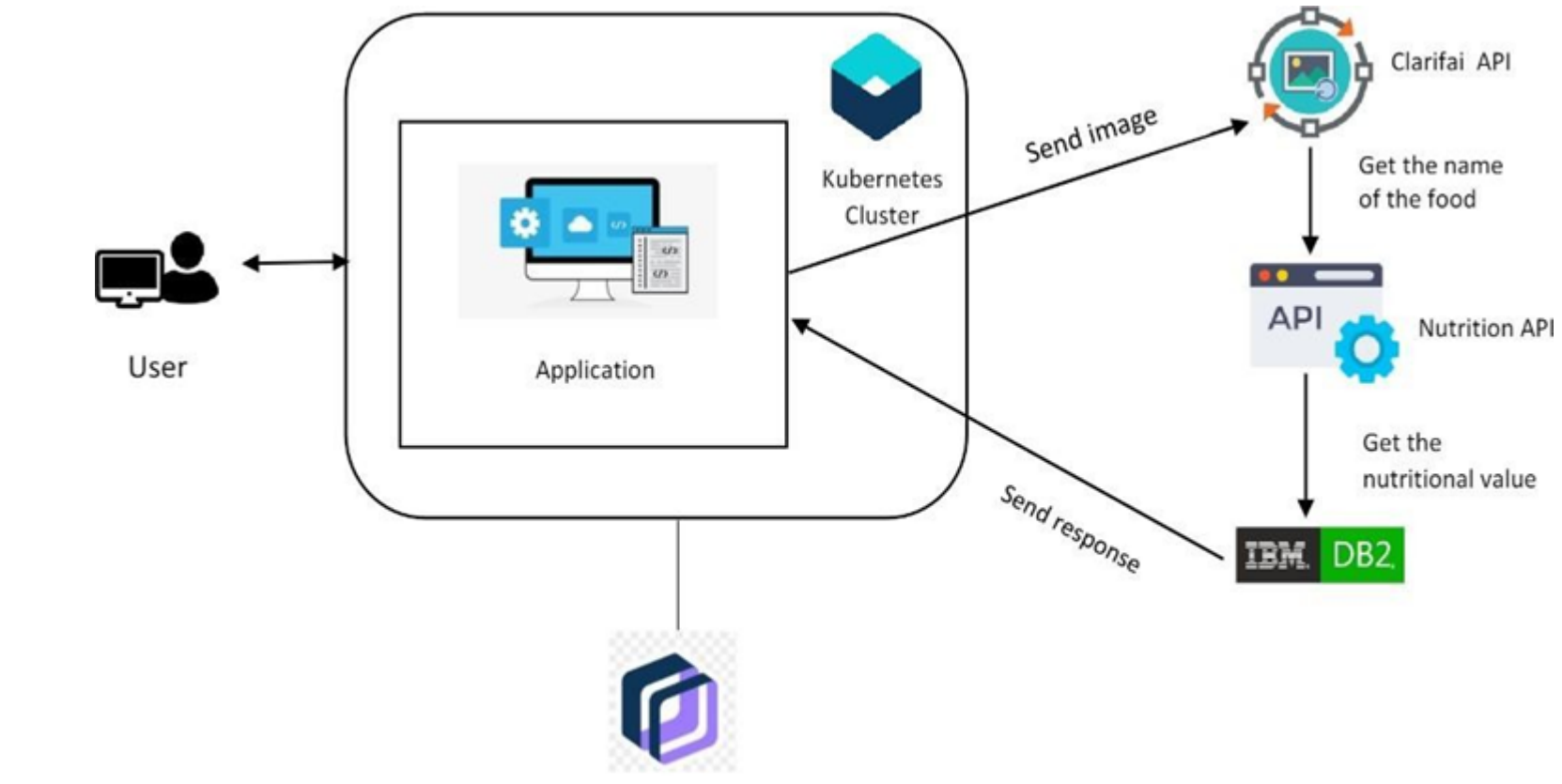
DFD Level 0



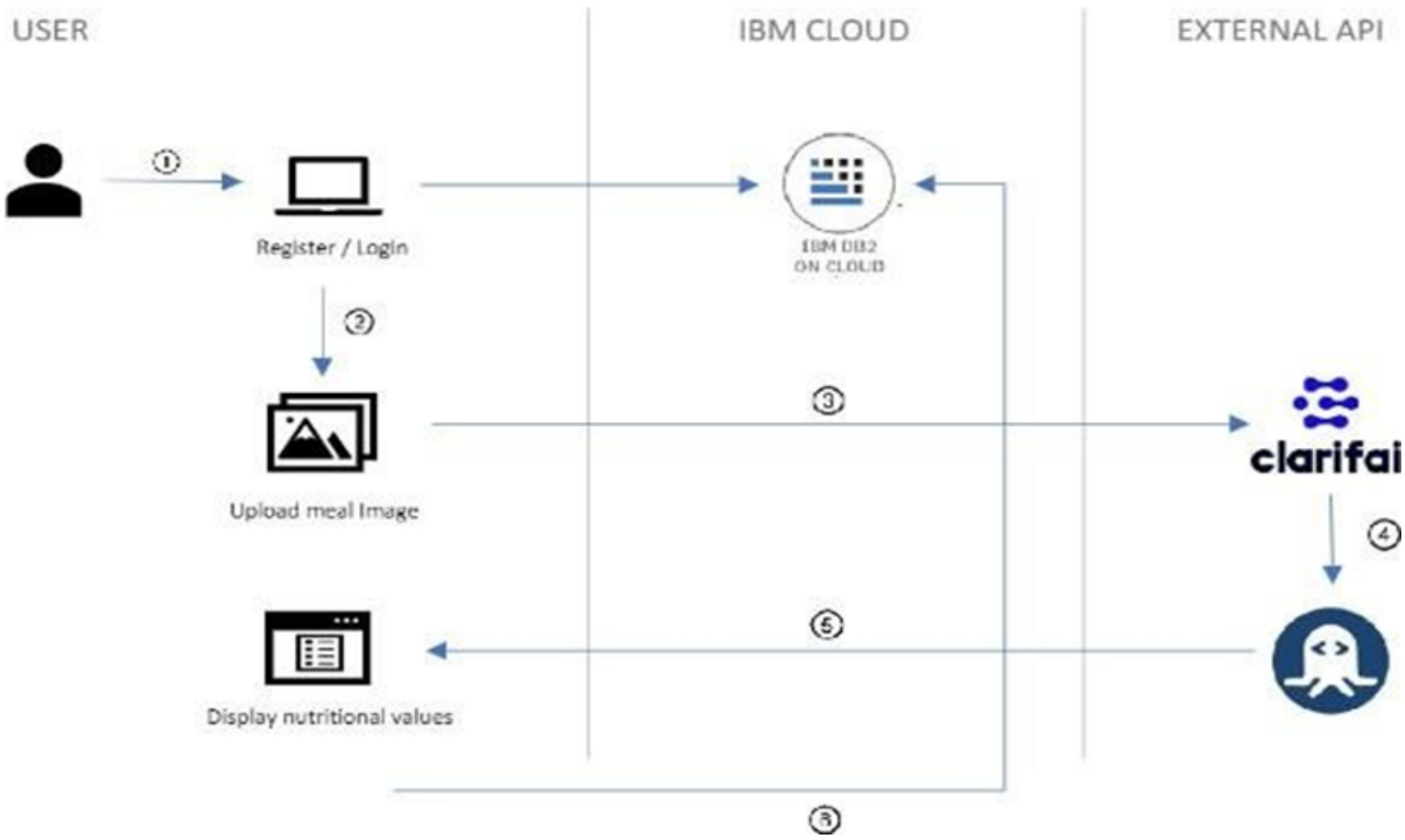
DFD Level 1



5.2 Solution & Technical Architecture



Container Registry



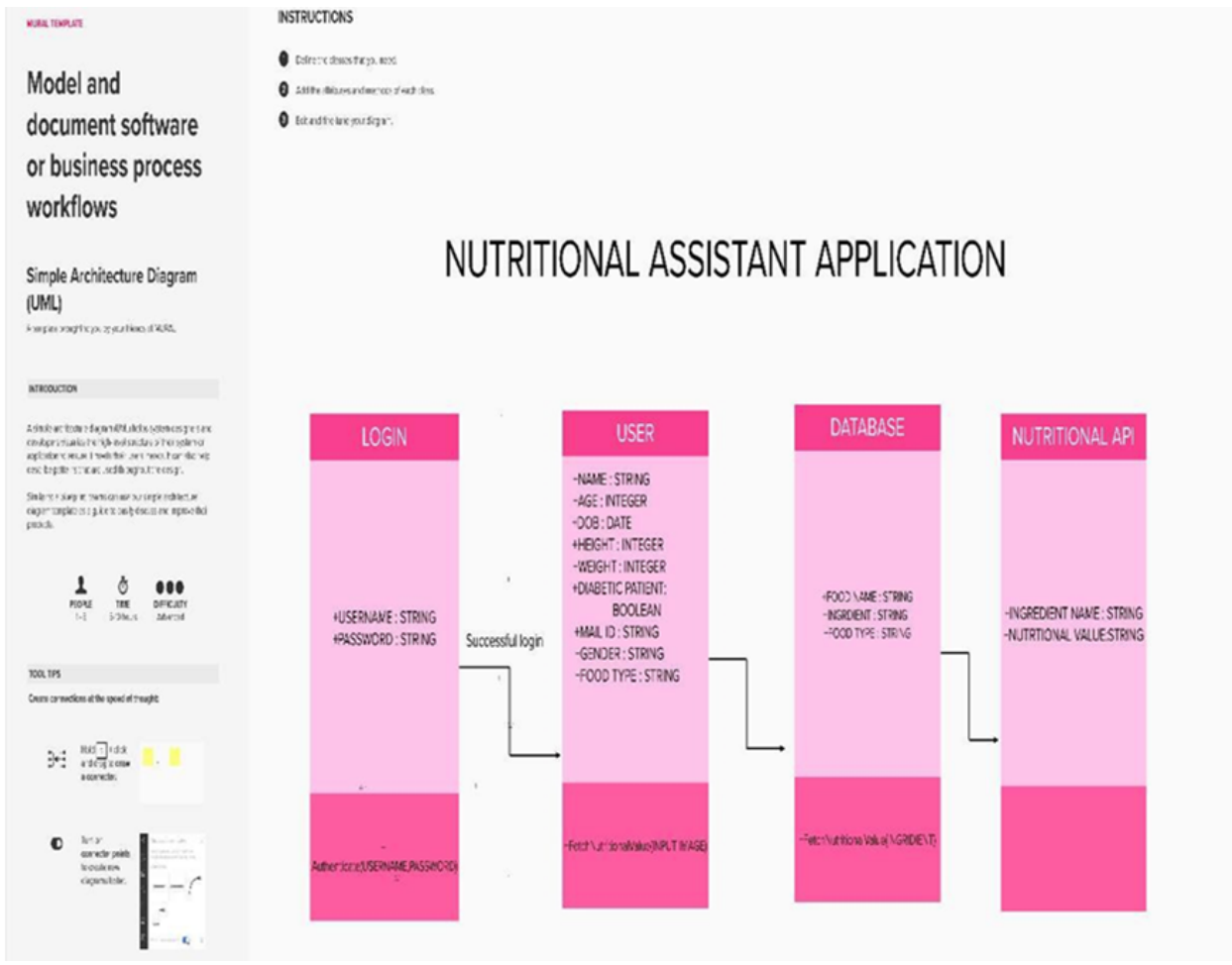


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI which gets image from the user to get the nutritional content.	HTML, CSS, Bootstrap
2.	Application Logic-1	Upload image in the application.	Python - Flask
3.	Application Logic-2	Displaying all the necessary information about the uploaded image to the user.	Python - Flask
4.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	File Storage	File storage requirements	IBM Container Registry
8.	External API-1	To give nutritional value of the determined food	Food API
10.	Machine Learning Model	For accurate food identification	Clarifai’s AI Driven Food Detection Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Cloud - Cloud Server Configuration :	Kubernetes.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Used Python-Flask for the backend system of the application.	Flask
2.	Security Implementations	All the security / access controls implemented in the application.	SHA-256, Encryptions
3.	Scalable Architecture	Applications can be transparently partitioned over multiple servers to reduce network traffic and scale up.	Event driven Architecture
4.	Availability	Availability of application.	DNS,Network Load Balancing System.
5.	Performance	Design consideration for the performance of the application.	Caching, third party CDN

5.3 User Stories

User Type	Functional Requirements	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password	I can access my account / dashboard	High	Sprint-1

		USN-2	As a user, I will receive a 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 Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medi um	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account / dashboard using my credentials	High	Sprint-1
	Home page	USN-6	As a user, I can upload a picture of my food in the home page	I am redirected to the details page on clicking submit	High	Sprint-1
	Language	USN-7	As a user, I should be able to see the details of the food in the language of my choice	I am able to view the details in my language after clicking translate button	Low	Sprint-3

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

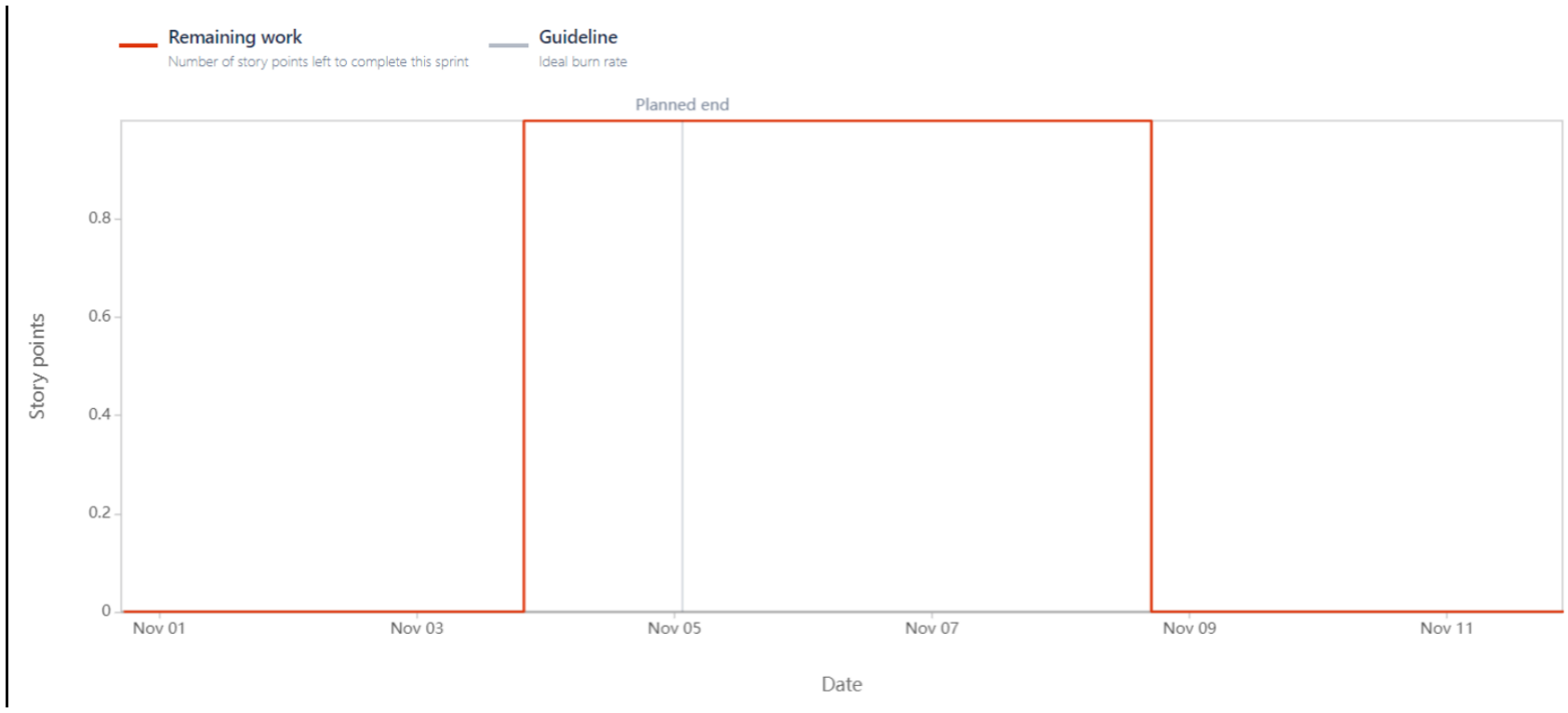
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members

Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	8	High	Sanyog
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	3	High	Madhav Prasath
Sprint-2	Registration	USN-3	As a user, I can register for the application through Facebook	2	Low	Venkat Narayan
Sprint-1	Registration	USN-4	As a user, I can register for the application through Gmail	1	Medium	Venkat Narayan
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	8	High	Hariprasad
Sprint-2	Dashboard	USN-6	As a user, I can land in the dashboard of my application	10	High	Sanyog
Sprint-3	Dashboard	USN-7	As a user, I can upload the image of my food in the dashboard	10	High	Venkat Narayan
Sprint-2	Dashboard	USN-8	As a user, I can see my profile	8	Medium	Madhav Prasath
Sprint-4	Dashboard	USN-9	As a user, I can update my profile	10	High	Sanyog
Sprint-3	Dashboard	USN-10	As a user, I can view the result of the processed image	10	High	Hariprasad
Sprint-4	Dashboard	USN-11	As a user, I can view my history of searches	10	Medium	Madhav Prasath

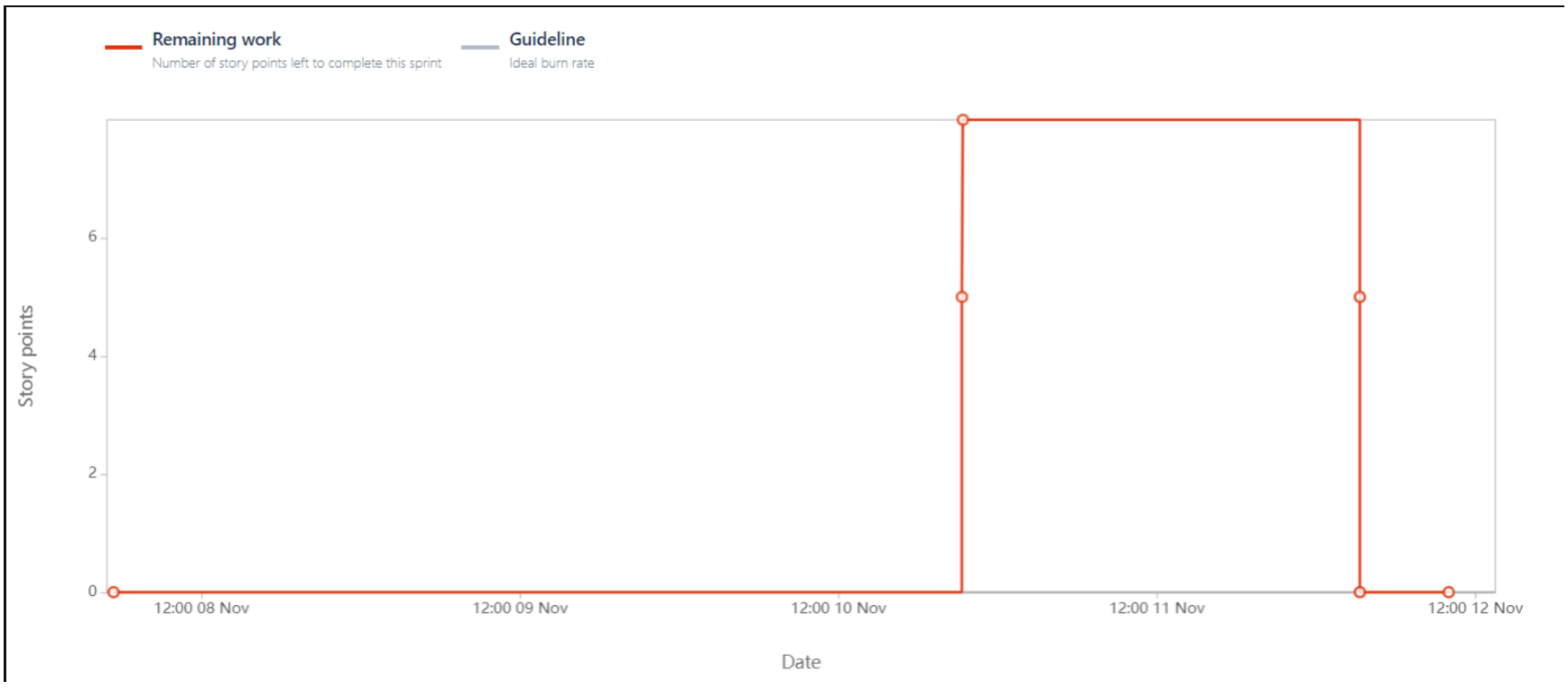
Burndown Charts:

Sprint-1:

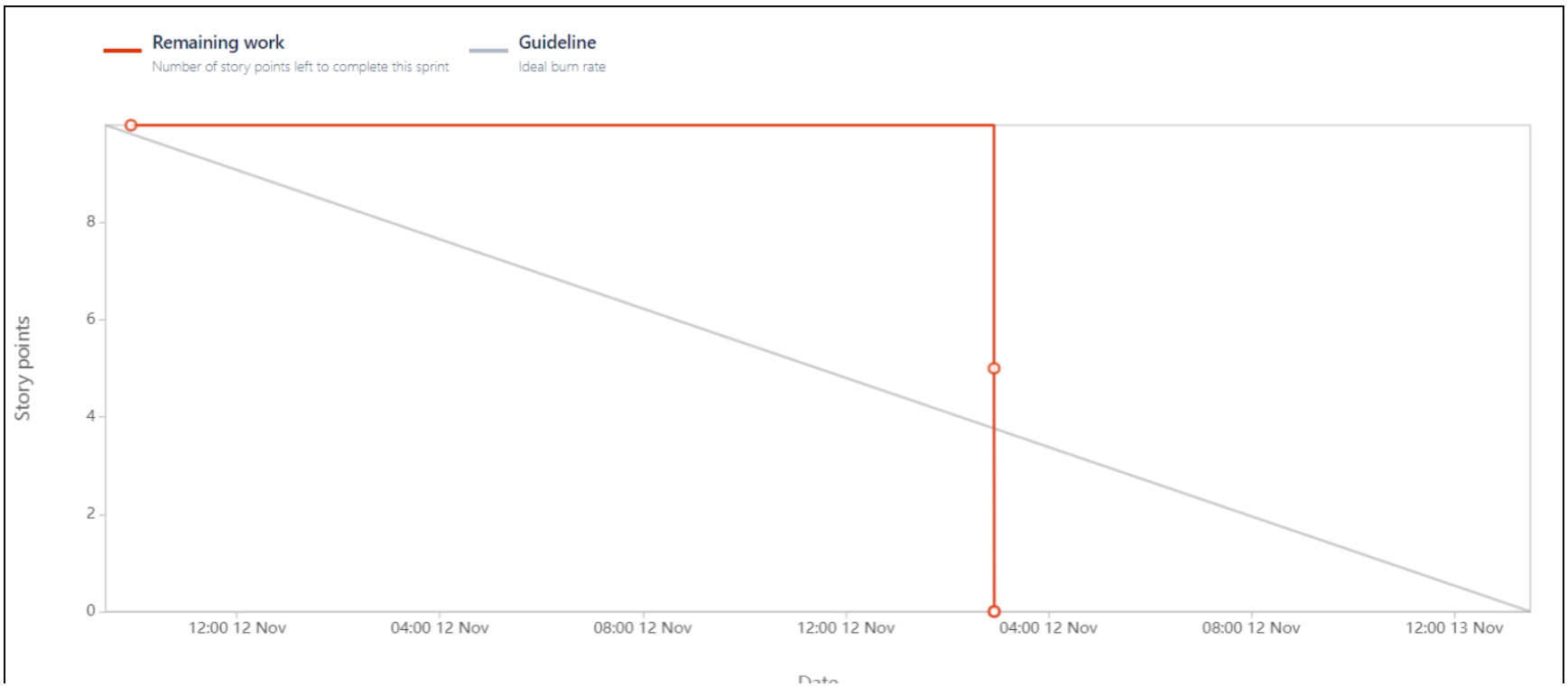
Sprint-2:



Sprint-3:



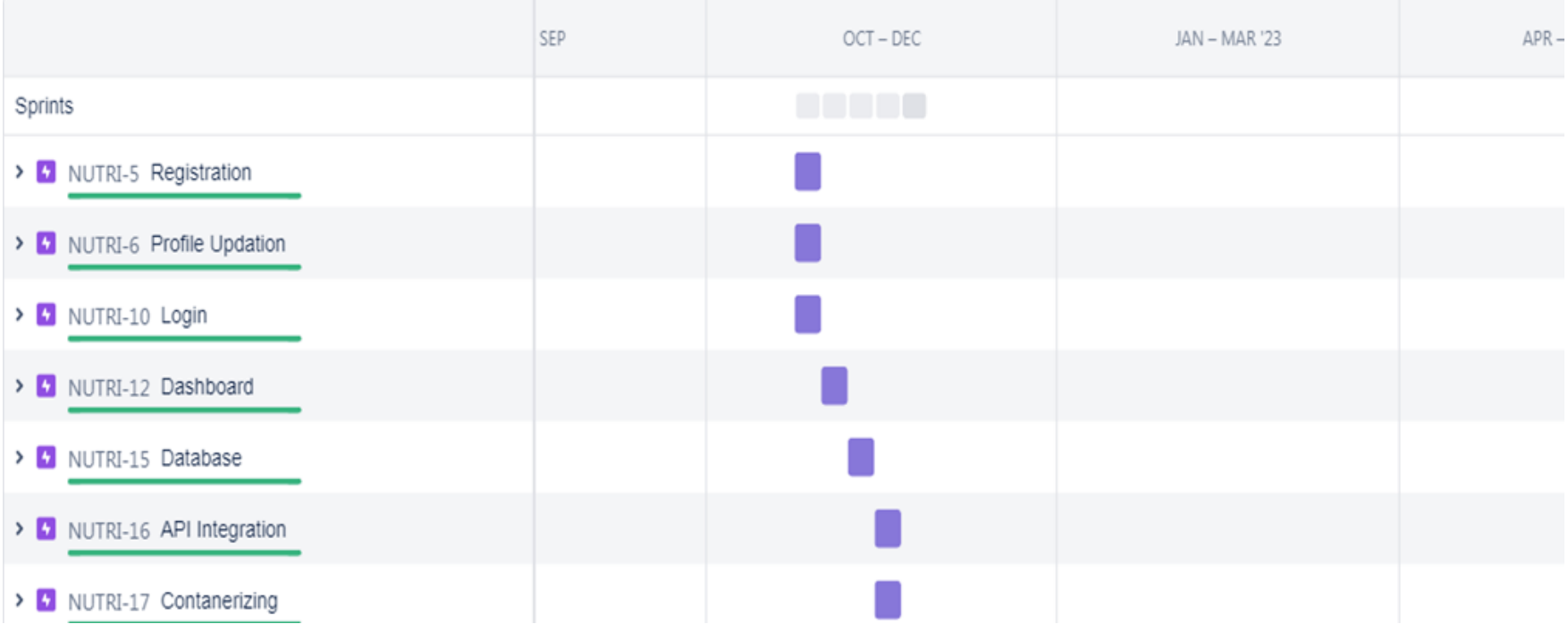
Sprint-4:

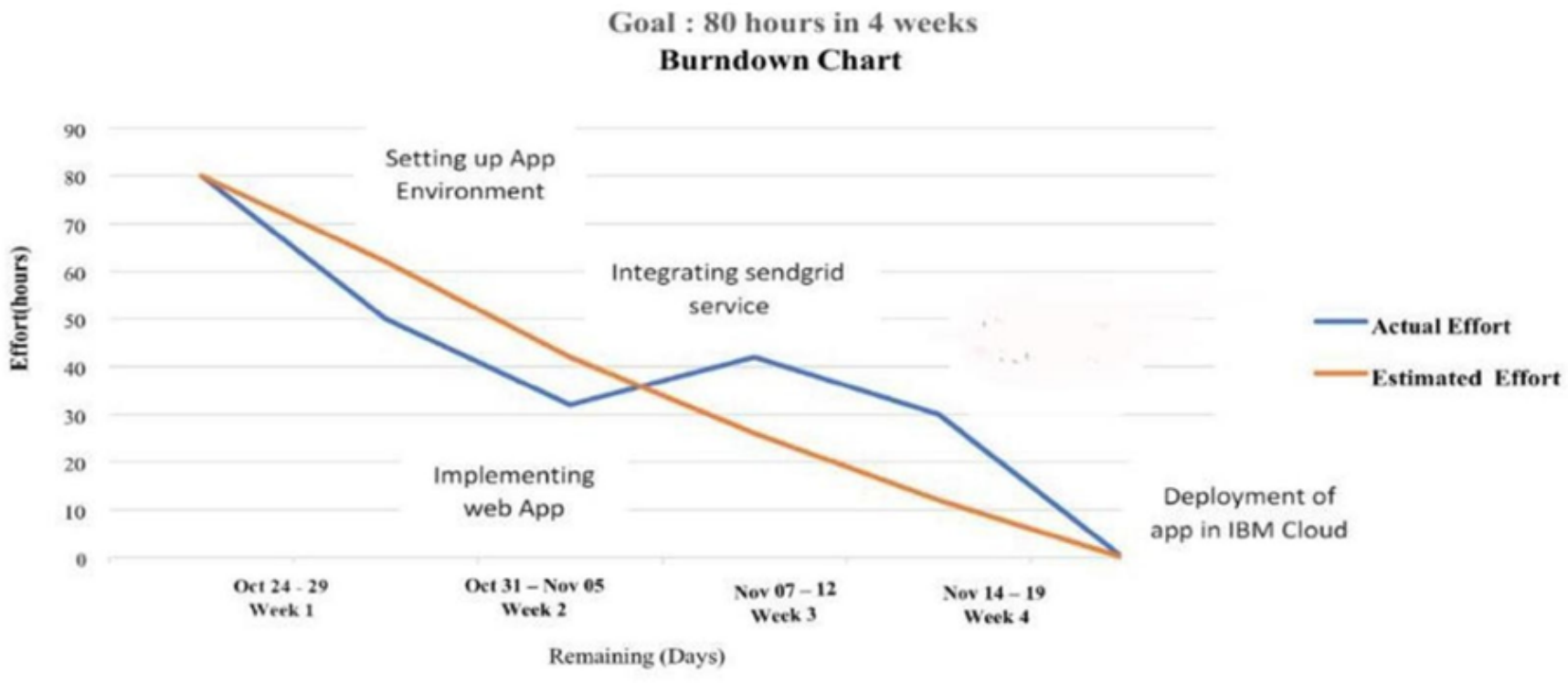


6.2 Sprint Delivery Schedule

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

6.3 Reports from JIR



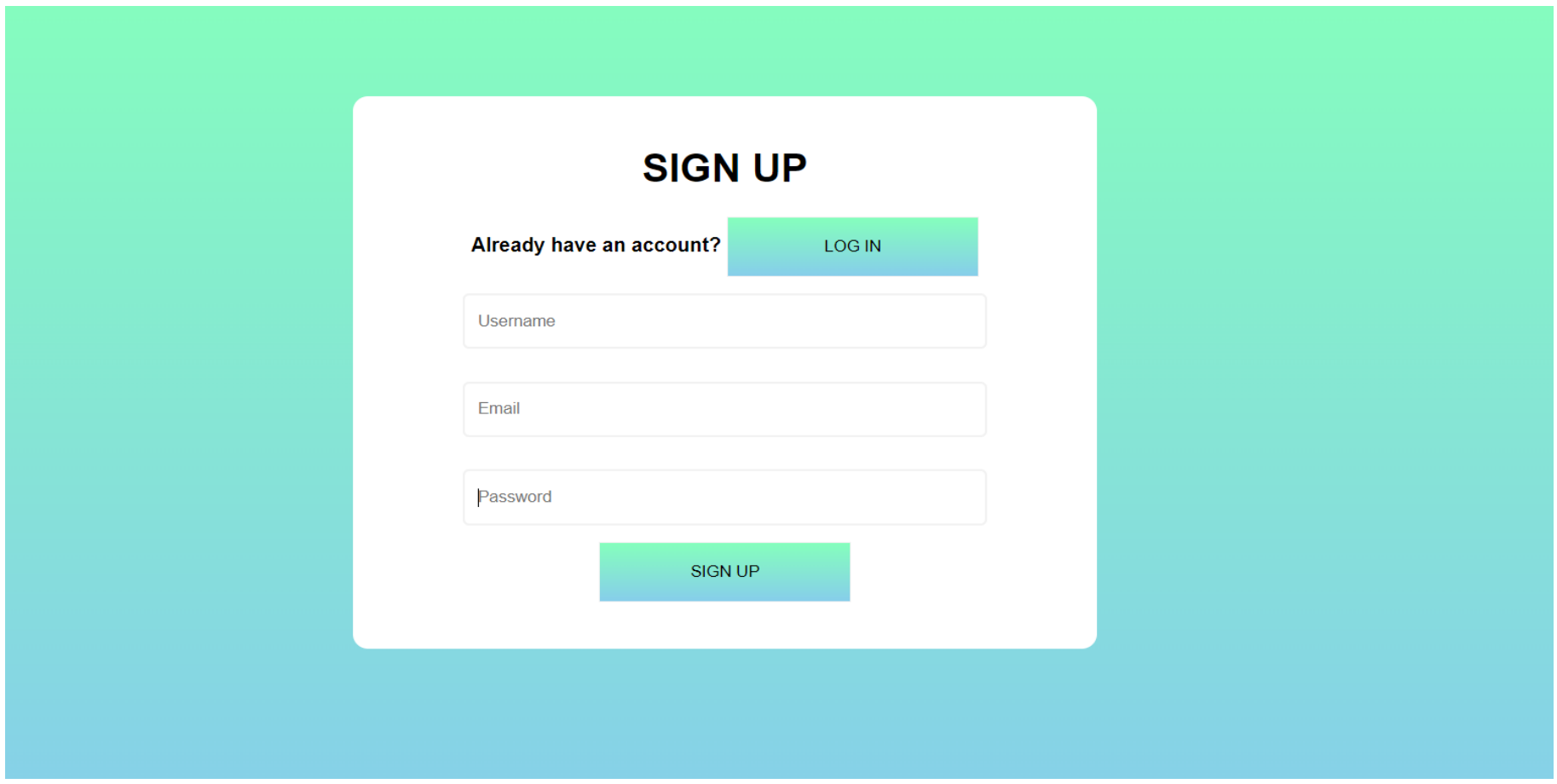


7.CODING & SOLUTIONING (Explain the features added in the project along with code)

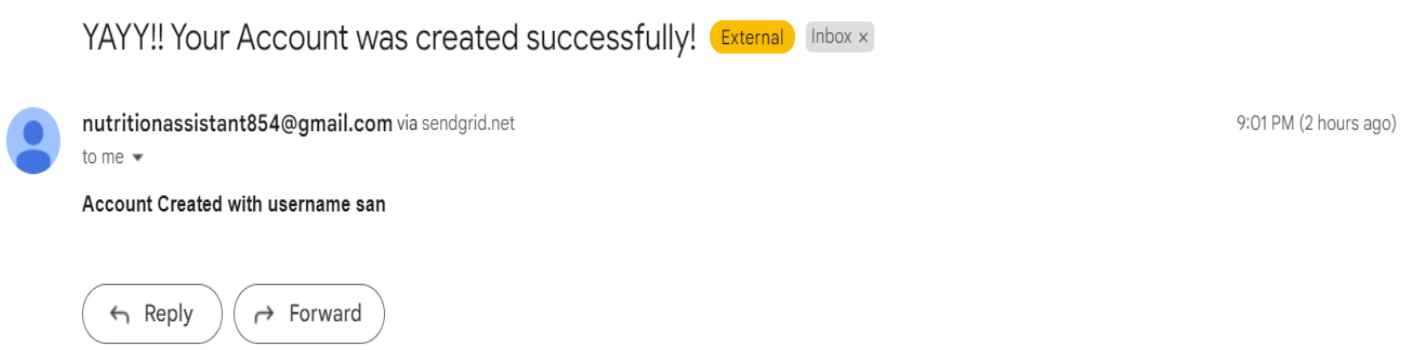
Feature 1:

The Nutrition Assistant Application is a web application which provides nutritional information about foods to the users. The Feature provides a login page which authenticates the users of the system. They have to provide their credentials, that is their username and password or via gmail. If they provide the correct credentials, they are logged in to the page, else an error message “Invalid Credentials” is displayed. If the user is visiting the application for the first time, they can sign up by giving some information like username, maid id, password.

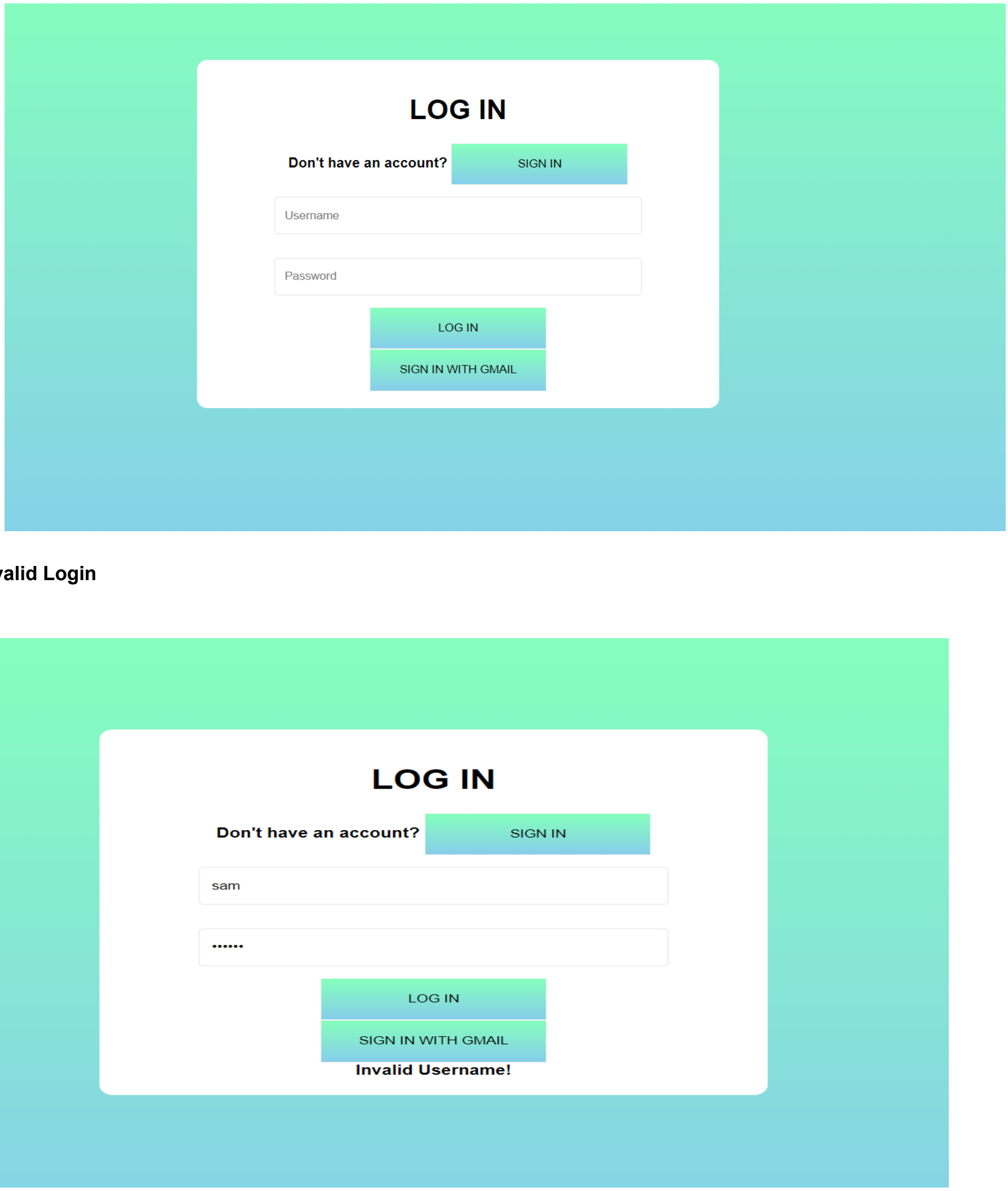
Sign Up Page



Upon signing up, a confirmation mail is sent to the user’s registered email.



Log-In with username



Invalid Login

CODE:

LOGIN

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Log In</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card">
      <h1>LOG IN</h1>
            <strong>  Don't  have  an  account?  <a  href="/"><button>Sign
in</button></a></strong>
      <form method="POST" action="">
        <div> <input type="text" placeholder="Username" name="user" required
/></div>
         <div> <input type="password" placeholder="Password" name="password"
required/></div>
        <input type="submit" value="Log In" />
      </form>
      <a href="/google_login"><button>Sign in with Gmail</button></a>
      <br />
      <strong>{{msg}}</strong>
    </div>
  </body>
</html>
```

SIGNUP:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Sign Up</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
```

```
      </head>
      <body>
        <div class="card">
          <h1>SIGN UP</h1>
              <strong>Already have an account? <a href='/login'><button>Log
in</button></a></strong>
          <form method="POST" action="">
                <div> <input type="text" placeholder="Username" name="user"
required/> </div>
            <div> <input type="text" placeholder="Email" name="email" required/>
</div>
             <div> <input type="password" placeholder="Password" name="password"
required/> </div>
            <input type="submit" value="Sign Up" />
          </form>
          <br/>
          <strong>{{msg}}</strong>
        </div>
      </body>
    </html>
```

PROFILE PAGE:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>User Profile</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card">
      <h1>Profile</h1>
      <h2>UserName : {{user}}</h2>
      <h2>Email : {{email}}</h2>
      <a href="/changepwd"><button>Change Password</button></a>
      <a href="/home"><button>Back to Home</button></a>
      <br />
      <strong>{{msg}}</strong>
    </div>
  </body>
</html>
```

FOODPAGE and HISTORY:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Dashboard</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card">
      <h1>Welcome {{user}}!</h1>
      <a href="/logout"><button>Logout</button></a>
      <a href="/delete"><button>Delete Account</button></a>
      <a href="/profile"><button>Profile</button></a>
      <a href="/history"><button>View History(limited to last 10)</button></a>
      <form method="POST" action="" enctype="multipart/form-data">
        <label for="file-upload" class="custom-file-upload">
          Upload Food Image
        </label>
        <input id="file-upload" type="file" name="food"/>
        <input type="submit" value="Submit" />
      </form>
      <h1>{{msg}}</h1>
    </div>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nutrition</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
```

```
  </head>
  <body>
    <div class="card" style="width:auto;">
      <h1>Food Nutrients</h1>
      {% autoescape false %}
        {{msg}}
      {% endautoescape %}
      <p>{{val}}</p>
    </div>
    </body>
    </html>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nutrition</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card" style="width:auto;">
      <h1>History</h1>
      {% autoescape false %}
        {{msg}}
      {% endautoescape %}
    </div>
    </body>
    </html>
```

STYLING:

```
html {
    background: linear-gradient(#85FFBD, #87CEEB);
    height: 100%;
    font-family: 'roboto', sans-serif;
    background-repeat: no-repeat;
    display: flex;
    justify-content: center;
    text-align: center;
}
```

```css
form {
    text-align: center;
    box-sizing: border-box;
}

form input[type="submit"], button {
    height: 50px;
    width: 200px;
    background: linear-gradient(#85FFBD, #87CEEB);
    border: 1px solid #f2f2f2;
    text-transform: uppercase;
    cursor: pointer;
}

form input[type="text"],
form input[type="password"] {
    max-width: 400px;
    width: 80%;
    line-height: 3em;
    margin: 1em 2em;
    border-radius: 5px;
    border: 2px solid #f2f2f2;
    outline: none;
    padding-left: 10px;
}

input[type="file"] {
    display: none;
}

.custom-file-upload {
    display: block;
    height: 50px;
    width: 200px;
    background: linear-gradient(#E61F20, #C9000B);
    border: 1px solid #f2f2f2;
    text-transform: uppercase;
    cursor: pointer;
    max-width: 400px;
    width: 80%;
    line-height: 3em;
    margin: 1em 4em;
    border-radius: 5px;
    border: 2px solid #f2f2f2;
    outline: none;
}
```

```css
.card {
    margin-top: 100px;
    background-color: white;
    width: 550px;
    height: inherit;
    padding: 20px;
    border-radius: 12px;
}

table, tr, th,td {
    text-align: center;
    padding:3px;
}
```

Feature 2:

The user can upload the image for which they wish to see the nutritional contents. Upon clicking the upload food image button, users can choose from their local computers to select the image.

Image Upload

Result

Upon uploading an image for example:

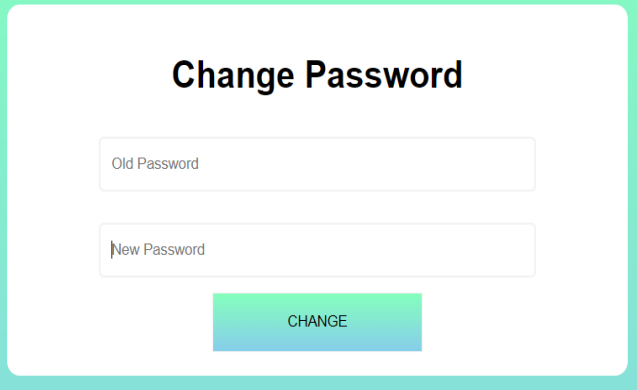
We get the results :

Food Nutrients

	Ingredients	sugar_g	fiber_g	sodium_mg	potassium_mg	fat_saturated_g	fat_total_g	calories	cholesterol_mg	protein_g	carbohydrates_total_g
0	lettuce	1.2	2.1	7.0	30.0	0.0	0.3	17.0	0.0	1.2	3.3
1	bread	5.7	2.7	495.0	98.0	0.7	3.4	261.6	0.0	8.8	50.2
2	cheese	0.5	0.0	661.0	459.0	18.9	33.0	393.9	100.0	22.7	3.2
3	sandwich	5.9	4.0	501.0	150.0	0.8	3.2	269.5	0.0	10.8	48.9
4	tomato	2.6	1.2	4.0	23.0	0.0	0.2	18.2	0.0	0.9	3.9
5	onion	4.7	1.4	2.0	35.0	0.0	0.2	44.7	0.0	1.4	10.1
6	meat	0.0	0.0	69.0	209.0	4.4	12.2	221.2	90.0	27.7	0.0
Total		20.6	11.4	1739.0	1004.0	24.8	52.5	1226.1	190.0	73.5	119.6

Also the user has the option to change their password just in case necessary.

Change Password




```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
          <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <title>Change Password</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card">
      <h1>Change Password</h1>
      <form method="POST" action="">
                <input  type="text"  placeholder="Old  Password"
name="oldpass" required />
             <input  type="password"  placeholder="New  Password"
name="newpass" required />
        <input type="submit" value="Change" />
      </form>
    </div>
  </body>
</html>
```

Users can also visit check the history of their image uploads (limited to 10):

History

	sugar_g	fiber_g	serving_size_g	sodium_mg	potassium_mg	fat_saturated_g	fat_total_g	calories	cholesterol_mg	protein_g	carbohydrates_total_g
0	4.6	2.4	100.0	1	75	0.2	1.5	93.9	0	3.4	21.0
1	0.0	0.0	0.0	0	0	0.0	0.0	0.0	0	0.0	0.0
2	0.0	0.0	0.0	0	0	0.0	0.0	0.0	0	0.0	0.0
3	19.4	9.3	600.0	1732	974	24.8	52.2	1209.1	190	72.3	116.3

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <title>Nutrition</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='style.css') }}"
    />
  </head>
  <body>
    <div class="card" style="width:auto;">
      <h1>History</h1>
      {% autoescape false %}
        {{msg}}
      {% endautoescape %}
    </div>
    </body>
    </html>
```

DB Schema

8.TESTING

8.1 Test Cases

				Date	3-Nov-22								
				Team ID	PNT2022TMID53138								
				Project Name	Project - Nutrition Assistant Application								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Prerequisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By

LoginPage_TC_OO1	Functional	Login page	Verify user is able to create a new account by giving email, username and password.		1.Enter URL(http://159.122.181.237:3121 212/) 2.Enter email, username and password. 3.Click Sign Up.	Test1@gmail.com testuser test123	Users should be able to create an account and get a confirmation email.	Working as expected	Pass				Hari Prasad
LoginPage_TC_OO2	Functional	Login Page	Verify user is able to log in through gmail.		1.Enter URL(http://159.122.181.237:3121 212/) 2.Enter gmail option, and select the authorized gmail account for logging in.	Test1@gmail.com	Application should open the google authentication service and list all logged in gmail accounts in that particular device.	Working as expected	Pass				Venkat Narayan
LoginPage_TC_OO3	Functional	Login page	Verify user is able to log into application with Valid credentials		1.Enter URL(http://159.122.181.237:3121 212/) 2.Enter valid username in the username field. 3.Enter valid password in the password field. 4.Click on the login button.	Username: testuser password: test123	User should navigate to user account homepage	Working as expected	Pass				Sanyog Kave
LoginPage_TC_OO4	Functional	Login page	Verify user is unable to log into application with InValid credentials		1.Enter URL(http://159.122.181.237:3121 212/) 2. Enter non existing username or(and) password in their respective fields. 3. Click on the login Button.	Username: kimjong password: korea	Application should show an "Invalid Credentials" message and should not allow to log in.	Working as expected	Pass				Madhava Prashath
HomePage_TC_OO1	UI	Home page	Verify users are able to see options like - Upload Food Image, Profile, Change Password, Logout, Delete Account.	User should be logged into their account.	1.Enter URL(http://159.122.181.237:3121 212/) 2. Enter valid username and password in their respective fields. 3. Click in login button.	Username: testuser password: test123	Application should show all the operations that can be performed by the user - Profile, Upload Food Image, Delete account, etc.	Working as expected	Pass				Hari Prasad
Home_TC_OO2	Functional	Home page	Verify users are able to upload an image.	User should be logged into their account Food image must be stored in local device.	1.Enter URL(http://159.122.181.237:3121 212/) 2. Enter valid username and password in their respective fields. 3. Click on the login button. 4. Click- Upload Food Image. 5. Select any food image from your local device. 6. Click Submit	Username: testuser password: test123 food image - Burger	Application should show the correct ingredients of the uploaded image and its nutritional content in a tabular format.	Working as expected	Pass				Venkat Narayan
Home_TC_003	Functional	Home page	Verify users are able to log out of their account.	User should be logged into their account	1.Enter URL(http://159.122.181.237:3121 212/) 2. Enter valid username and password in their respective fields. 3. Click in login button. 4. Click in the logout button.	Username: testuser password: test123	Application should redirect the user back to the login Page.	Working as expected	Pass				Sanyog Kave

Home_TC_004	**Functional**	**Home page**	**Verify users are able to see history of their last 10 uploaded food history details.**	**User should be logged into their account**	**1.Enter URL(http://159.122.181.237:31212/)2. Enter valid username and password in their respective fields. 3. Click on login button. 4. Click on View History Button.**	**Username: testuser password: test123**	**Application should display the information about the user's last 10 food details along with nutritional contents.**	**Working as expected**	**Pass**				**Madhava Prashath**
Profile_TC_001	**Functional**	**Profile page**	**Verify users are able to visit the change password.**	**Users should be logged into their account.**	**1.Enter URL(http://159.122.181.237:31212/)2. Enter valid username and password in their respective fields. 3. Click on login button. 4. Click on the Profile button. 5. Click on the Change password button. 6. Enter the new password. 7. Re enter the new password. 8. Click on Submit.**	**Username: testuser password: test123 new Password: test321 re enter new password: test321**	**Application should update the new password in the database successfully.**	**Working as expected**	**Pass**				**Hari Prasad**

8.2 User Acceptance Testing

Acceptance Testing
UAT Execution & Report Submission

Date	03 November 2022
Team ID	PNT2022TMID53138
Project Name	Project - Nutrition Assistant Application
Maximum Marks	4 Marks

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Nutrition Assistant Application 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	0	0	0	0	0
Duplicate	4	0	1	0	5
External	0	1	1	0	2
Fixed	9	3	7	4	23
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	1
Totals	13	5	7	4	31

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Home Page	3	0	0	3
Client Application	8	0	0	8
Registration/Login Page	4	0	0	4
Nutrition API	6	0	0	6
Clarifai's Model API	6	0	1	5
Connection between Client and Server	9	0	0	9
Kubernetes Deployment	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics

			NFT - Risk Assessment						
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Authentication	New	High	No Changes	High	Nil	>10 to 30%	ORANGE	Auth changes is crucial to the application.
2	Clarifai API	New	High	No Changes	High	Nil	>10 to 30%	RED	Clarifai's Food Detection model is the core of this application.
3	Profile Page	New	High	No Changes	High	Nil	>10 to 30%	GREEN	Profile page is required for a user.

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff

1	Nutrition Assistant Application	End to end manual testing	1. The user has internet access. 2. The user knows to operate a browser. 3. The user has to food images saved.	Approved

10.ADVANTAGES & DISADVANTAGES

Advantages:

The advantages of Nutrition Assistant Application are as follows:

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

Disadvantages:

- Sometimes it causes a level of disbalance in the balanced diet of an individual.
- It can improve the level of nutrition among individuals but delivers an inappropriate means of nutritional labeling.
- Sometimes the provided nutritional value of uploaded food images are not accurate.
- It requires an active internet connection.
- Clarifai's food Detection model is unable to identify Indian Traditional foods correctly.

11. CONCLUSION

In conclusion, our project will be helpful for the end user in the following ways:

➜ The information about the nutritional value of the food that has been printed in the food packages are not convenient to keep track of the daily calorie intake.

➜ Nutrition Assistant Application helps in finding the nutritional content present in the food with real time image processing using Clarifai Food Detection Model API and Nutrition API.

➜ The users can upload their daily meal image and get the nutritional content values.

➜ The users can check their history of uploads to view their past foods uploaded.

12. FUTURE SCOPE

The application developed fulfills the promised features and satisfies the requirements. But there can be some more improvement to it, which we would like to keep under future scope.

Future Scope and improvements:

- Send timely notification to users to take particular food to satisfy dietary requirements.
- Chatting System that connects users to nutritionists and doctors for expert advice.
- Connect with friends to share tasks and diet plan which creates a healthy competitive environment.

13. APPENDIX

13.1 Source Code

```
import ibm_db as db
from flask import Flask, render_template, request, redirect,
session, abort
import os
import pathlib
import requests
from dotenv import load_dotenv
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from google.oauth2 import id_token
from google_auth_oauthlib.flow import Flow
from pip._vendor import cachecontrol
import google.auth.transport.requests
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 werkzeug.utils import secure_filename
from datetime import date
import json
import pandas as pd
UPLOAD_FOLDER='/uploads'

# Configure Flask app
app = Flask(__name__)
SECRET_KEY = os.urandom(32)
app.config['SECRET_KEY'] = SECRET_KEY
```

```
# Load .env file
load_dotenv()

# Connect to the Database
HOSTNAME = os.getenv('HOSTNAME')
PORT_NUMBER = os.getenv('PORT_NUMBER')
DATABASE_NAME = os.getenv('DATABASE_NAME')
USERNAME = os.getenv('USER')
PASSWORD = os.getenv('PASSWORD')
GOOGLE_CLIENT_ID = os.getenv('GOOGLE_AUTH_CLIENT_ID')
NUTRITION_API_KEY = os.getenv('NUTRITION_API')

connection_string                                                    =
"DATABASE={0};HOSTNAME={1};PORT={2};SECURITY=SSL;SSLServerCertif
icate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID={3};PWD={4};".
format(DATABASE_NAME, HOSTNAME, PORT_NUMBER, USERNAME, PASSWORD)
conn = db.connect(connection_string, "", "")

# Frequently used variables
SIGN_UP_PAGE_URL = '/'
LOG_IN_PAGE_URL = '/login'
HOME_PAGE_URL = '/home'
GOOGLE_LOGIN_PAGE_URL = '/google_login'
PROFILE_PAGE_URL = '/profile'
CHANGE_PASSWORD_URL = '/changepwd'
FOOD_URL = '/food'
HISTORY_PAGE_URL = '/history'

# Google Auth Configuration
os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"

client_secrets_file                                                  =
os.path.join(pathlib.Path(__file__).parent,
"client_secret.json")

flow = Flow.from_client_secrets_file(
    client_secrets_file=client_secrets_file,
      scopes=["https://www.googleapis.com/auth/userinfo.profile",
"https://www.googleapis.com/auth/userinfo.email", "openid"],
    redirect_uri="http://127.0.0.1:5000/callback"
)
```

```
#Clarifai api
C_USER_ID = os.getenv('C_USER_ID')
# Your PAT (Personal Access Token) can be found in the portal
under Authentification
C_PAT = os.getenv('C_PAT')
C_APP_ID = 'main'
C_MODEL_ID = 'food-item-recognition'
IMAGE_URL = 'https://samples.clarifai.com/metro-north.jpg'

# Helper Function to execute SQL queries
def execute_sql(statement, **params):
    global conn
    stmt = db.prepare(conn, statement)

    param_id = 1
    for key, val in params.items():
        db.bind_param(stmt, param_id, val)
        param_id += 1

    result = ''
    try:
        db.execute(stmt)
        result = db.fetch_assoc(stmt)
    except:
        pass

    return result


def execute_Multisql(statement):
    print(statement)
    result = []
    global conn
    stmt = db.exec_immediate(conn, statement)
    dictionary = db.fetch_assoc(stmt)
    result.append(dictionary)
    while dictionary != False:
        dictionary = db.fetch_assoc(stmt)
        result.append(dictionary)
    # param_id = 1
    # for key, val in params.items():
```

```
    #     db.bind_param(stmt, param_id, val)
    #     param_id += 1

    # result = []
    # try:
    #     dictionary = db.fetch_assoc(stmt)
    #     print(dictionary)
    #     while dictionary != False:
    #         print(dictionary)
    #         result.append(dictionary)
    #         dictionary = db.fetch_assoc(stmt)
    # except:
    #     print('error in multisql')
    #     pass

    return result

# Creates user table if not exists
create_table   =   "CREATE   TABLE   IF   NOT   EXISTS   user(email
varchar(30), username varchar(30) NOT NULL, password varchar(30)
, PRIMARY KEY(username))"
execute_sql(statement=create_table)
create_table = "CREATE TABLE IF NOT EXISTS stats(id integer NOT
NULL,   username   varchar(30),   uploadedOn   DATE   ,   result
VARCHAR(32074),   PRIMARY   KEY(id),   FOREIGN   KEY(username)
REFERENCES user(username) ON DELETE CASCADE)"
execute_sql(statement=create_table)

# Helper function to send confirmation mail on sign in
def send_confirmation_mail(user, email):
    message = Mail(
        from_email="nutritionassistant854@gmail.com",
        to_emails=email,
        subject="YAYY!! Your Account was created successfully!",
           html_content= "<strong>Account Created with username
{0}</strong>".format(user)
    )

    try:
                                                sg      =
SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
```

```
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)

# Sign up page
@app.route(SIGN_UP_PAGE_URL, methods=['GET', 'POST'])
def signup():
    msg = ''

    if session.get('user'):
        return redirect(HOME_PAGE_URL)

    if request.method == 'POST':
        user = request.form['user']
        email = request.form['email']
        password = request.form['password']

        duplicate_check = "SELECT * FROM user WHERE username=?"
                account  =  execute_sql(statement=duplicate_check,
user=user)

        if account:
                msg = "There is already an account with this
username!"
        else:
            insert_query = "INSERT INTO user values(?, ?, ?)"
                execute_sql(statement=insert_query,  email=email,
user=user, password=password)

            send_confirmation_mail(user, email)
            return redirect(LOG_IN_PAGE_URL)
    return render_template('signup.html', msg=msg)

# Login page
@app.route(LOG_IN_PAGE_URL, methods=['GET', 'POST'])
def login():
    msg = ''
```

```
    if session.get('user'):
        return redirect(HOME_PAGE_URL)

    if request.method == "POST":

        user = request.form['user']
        password = request.form['password']

        duplicate_check = "SELECT * FROM user WHERE username=?"
              account  =  execute_sql(statement=duplicate_check,
user=user)

        print(account)
        if account and account['PASSWORD'] == password:
            session['user'] = user
            return redirect(HOME_PAGE_URL)
        elif account and account['PASSWORD'] != password:
            msg = 'Invalid Password!'
        else:
            msg = "Invalid Username!"

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

# Login using Gmail
@app.route(GOOGLE_LOGIN_PAGE_URL , methods=['GET','POST'])
def google_login():
    authorization_url, state = flow.authorization_url()
    session["state"] = state
    return redirect(authorization_url)

# Configuring user credentials after gmail login
@app.route("/callback")
def callback():
    flow.fetch_token(authorization_response=request.url)

    if session["state"] != request.args["state"]:
        abort(500)  # State does not match!

    credentials = flow.credentials
    request_session = requests.session()
    cached_session = cachecontrol.CacheControl(request_session)
```

```
                                                token_request          =
google.auth.transport.requests.Request(session=cached_session)

    id_info = id_token.verify_oauth2_token(
        id_token=credentials._id_token,
        request=token_request,
        audience=GOOGLE_CLIENT_ID,
        clock_skew_in_seconds=10
    )

    session["user"] = id_info.get("email")
    session["google_id"] = id_info.get("sub")
    session["name"] = id_info.get("name")
    return redirect(HOME_PAGE_URL)

#Clarify and nutrion application
@app.route(FOOD_URL,methods=['GET','POST'])
def foodpage():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)
    msg=''
    user = session.get('user')
    channel = ClarifaiChannel.get_grpc_channel()
    stub = service_pb2_grpc.V2Stub(channel)

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

                                          userDataObject          =
resources_pb2.UserAppIDSet(user_id=C_USER_ID, app_id='main')

    #print(FILE_NAME)
    with open(FILE_NAME, "rb") as f:
        file_bytes = f.read()

    post_model_outputs_response = stub.PostModelOutputs(
        service_pb2.PostModelOutputsRequest(
            user_app_id=userDataObject,
            model_id=C_MODEL_ID,

            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]
    query = ''
    #print("Predicted concepts:")
    for concept in output.data.concepts:
        #print("%s %.2f" % (concept.name, concept.value))
        if(concept.value>0.3):
            if len(query) > 0 and query[-1] != '&':
                query += " and "
            query += concept.name

    # Uncomment this line to print the full Response JSON
    #print(post_model_outputs_response)

                                        api_url            =
'https://api.calorieninjas.com/v1/nutrition?query='

            response    =    requests.get(api_url    +    query,
headers={'X-Api-Key': NUTRITION_API_KEY})
    if response.status_code != requests.codes.ok:
        print("Error:", response.status_code, response.text)
        abort(500)

    obj = json.loads(response.text)
```

```
    totalDict = {}
    for item in obj['items']:
        for key, value in item.items():
            if type(value) == str:
                continue
            if totalDict.get(key, -1) != -1:
                totalDict[key] += value
            else:
                totalDict[key] = 0
    data = json.dumps(totalDict, indent=2)

    df = pd.DataFrame(obj["items"])
    df.insert(0, "Ingredients", query.split(" and "))
    df.drop('name', axis=1, inplace=True)
    df.drop('serving_size_g', axis=1, inplace=True)

    df.loc['Total']=df.sum(axis=0, numeric_only=True)
    df.iloc[-1, df.columns.get_loc('Ingredients')] = ''

    sqlst = "SELECT count(*) from stats"
    id = execute_sql(statement=sqlst)
    newId = int(id['1'])+1
    today = date.today()
    sqlst = "INSERT INTO stats values(?,?,?,?)"
    execute_sql(statement=sqlst , id = newId , username = user ,
date = today , result = data)


    return render_template('foodpage.html', msg=df.to_html())

# History
@app.route(HISTORY_PAGE_URL , methods=['GET','POST'])
def history():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)
    msg = ''
    user = session.get('user')
      sqlst = f"SELECT * from stats where username = '{user}'
ORDER BY id desc limit 10"
    result = execute_Multisql(statement = sqlst)
    #print(type(result))
```

```
    print(result)
    # outputStats = result['RESULT']
    # dateUploaded = result['UPLOADEDON']
    # #print(outputStats)
    totalDict = {}
    for item in result:
        if type(item) == bool:
            continue
        nutritionValues = json.loads(item['RESULT'])
        print(nutritionValues)
        tempdict = {}
        for key, value in nutritionValues.items():
            if tempdict.get(key, -1) != -1:
                tempdict[key] += value
            else:
                tempdict[key] = value
        for key, value in tempdict.items():
            if totalDict.get(key, -1) == -1:
                totalDict[key] = [value]
            else:
                totalDict[key].append(value)
    print(totalDict)
    # dictString = json.loads(outputStats)
    df = pd.DataFrame(totalDict)
    #df.insert(0, "Date", dateUploaded)


    return render_template('history.html', msg=df.to_html())



# Home page
@app.route(HOME_PAGE_URL, methods=['GET', 'POST'])
def homepage():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)
    global FILE_NAME
    msg = ''
    if request.method == 'POST':
        if request.files['food']:
            img=request.files['food']
            #print(img.filename)
```

```
            #print("ABCD",)

FILE_NAME=os.path.join('./uploads/',secure_filename(img.filename
))
            print(FILE_NAME)
            img.save(FILE_NAME)

            msg = 'Image Uploaded Successfully!'
            return redirect(FOOD_URL)
        else:
            msg = "Image wasn't uploaded, Try again!"

                    return      render_template('homepage.html',
user=session.get('user'), msg=msg)

# Profile page
@app.route(PROFILE_PAGE_URL, methods=['GET', 'POST'])
def profile():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    sqlst = "select email from user where username=?"
    user = session.get('user')
    email = execute_sql(statement=sqlst, user=user)

            return   render_template('profile.html',   user=user,
email=email['EMAIL'])

#change password
@app.route(CHANGE_PASSWORD_URL, methods=['GET', 'POST'])
def changepwd():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

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

```
        sqlst = 'SELECT password from user where username = ?'
            dbpass  =  execute_sql(statement  =  sqlst  ,  username  =
user)['PASSWORD']
        sqlst = 'SELECT email from user where username = ?'
             email  =  execute_sql(statement  =  sqlst  ,username  =
user)['EMAIL']

        if dbpass == oldpass:
             sqlst = 'UPDATE user SET password = ? where username
= ?'
             execute_sql(statement = sqlst , password = newpass ,
username = user)
             msg = 'Updated Successfully!'
        else:
             msg = 'Old Password Incorrect!'

               return  render_template('profile.html',  user=user,
email=email, msg=msg)

    return render_template('passwordChange.html')


# Logout user
@app.route('/logout')
def logout():
    session['user'] = ''
    return redirect(LOG_IN_PAGE_URL)


# Delete user account
@app.route('/delete')
def delete():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    user = session['user']
    delete_query = "DELETE FROM stats where username=?"
    execute_sql(statement=delete_query, username=user)
    delete_query = "DELETE FROM user WHERE username=?"
    execute_sql(statement=delete_query, username=user)
```

```
    session.clear()
    return redirect(SIGN_UP_PAGE_URL)

# Run the application
if __name__ == '__main__':
    app.run(debug=True)
```

13.2 GitHub & Project Demo Link

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-16823-1659623579

Demo Link: **https://drive.google.com/file/d/1LEUus9YHfP75B5RHYcCju2sqoTY4ZmvL/view?usp=sharing**