Nutrition Assistant Application

A PROJECT REPORT

TEAM ID : PNT2022TMID11404

Submitted by

PRADHEEKSHASRI A.S. 910619104054

PRADICSHA C.A. 910619104056

PREETHI K 910619104058

RHEENA R 910619104063

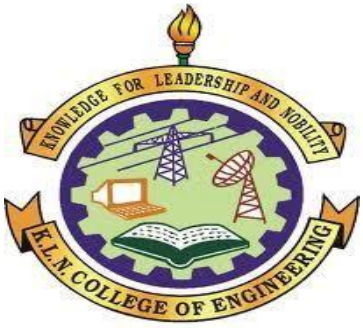
In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



K.L.N. COLLEGE OF ENGINEERING

(An Autonomous Institution, Affiliated to Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

NOV 2022

INDEX

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 App that automatically estimates food attributes such as ingredients and nutritional value by classifying the input image of food and provide it's nutritional values. Our method employs **"Clarifai's AI-Driven Food Detection Model"** for accurate food identification and "Spoonacular Nutrition API" 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 Spoonacular API which provides the nutritional value of the identified food.

1.2 Purpose

Basically, a diet and nutrition app comes with lots of benefits. It helps users in:

- To keep track of daily intake
- To monitor calories intake
- To provide facility to upload meal image
- To get nutritional value of the uploaded image
- To keep track of BMI

2. LITERATURE SURVEY

2.1 Existing solutions

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 followthrough 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 add 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.

2.2 References

i. Name of the Paper : Personal Dietary Assisstant
   Author : Gabriella Simon-Nagy, BalazsTusor

ii. Name of the Paper : Effects and challenges of a using a nutrition assistance system
Author : Monika Wintergerst , Markus Bohm

iii. Name Of The Paper : Primary Nutrition Health Care
Author : Christian Kraef et al.Bull World Health Organ

iv. Name Of The Paper : Perioperative Nutrition
Author : Michael Scott etal. AnesthAnalg

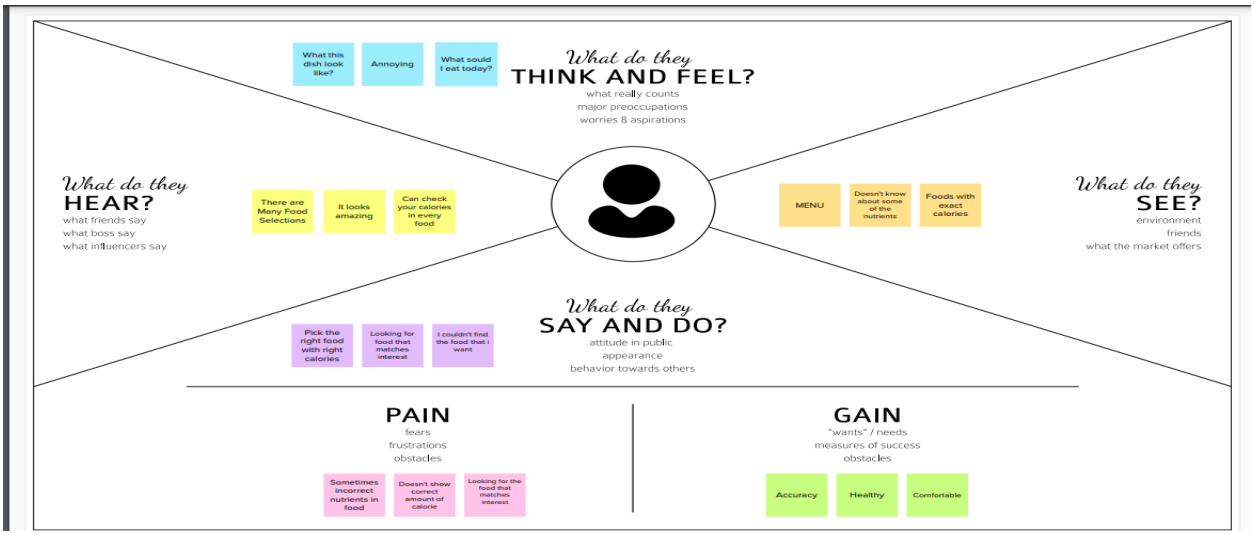
v. Name Of The Paper : Virtual Nutrionist using AI
Author : Siddarthan Chitra Suseendran

2.3 Problem Statement Definition

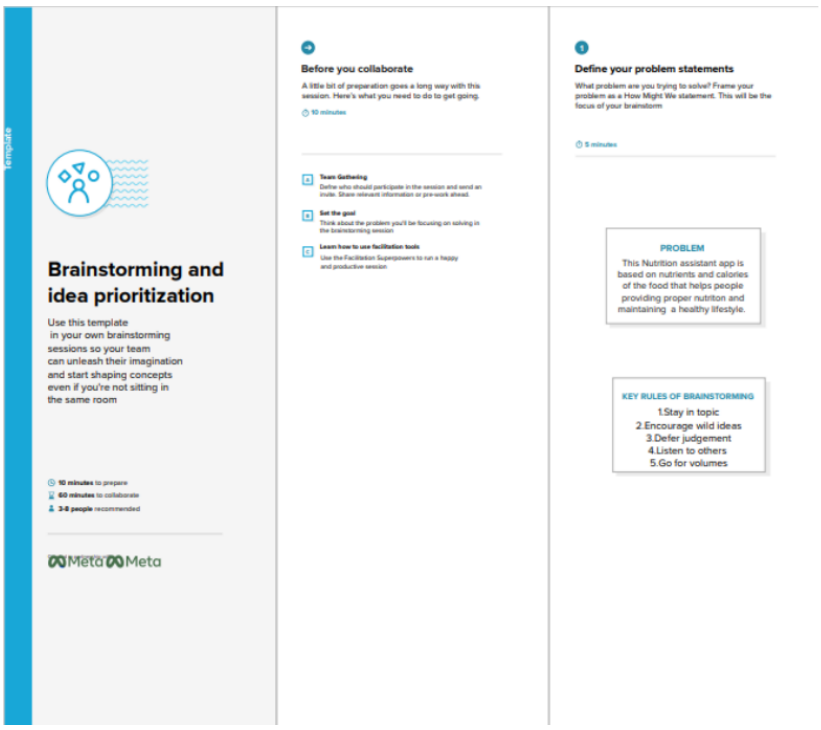
To develop Artificial Intelligence powered food AI image recognition app which automatically identify & quantify thousands of food categories and pair the food items with the relevant nutritional information for individuals to monitor and maintain the level of calorie intake. In modern days people are having a lot of fast foods which causes many diseases. To avoid these problems doctors suggests us to follow a diet. People nowadays are busy in their daily life so they cannot follow a proper diet without the nutrition chart.While travelling it is very difficult to follow the diet.Medicines often cause these taste changes and the unhealthy food habits are being practiced or being followed now a days due to fast moving world,it makes humans to lead a unhealthy lifestyle which leads to health issues.So this Nutrition Application provides us the nutrients that provides in the food. By Scanning the image it provides the nutrients present the meal such as proteins , calories etc.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



2

Brainstorm

Write down any ideas that come to mind that address your problem statement

10 minutes

3

Group ideas.

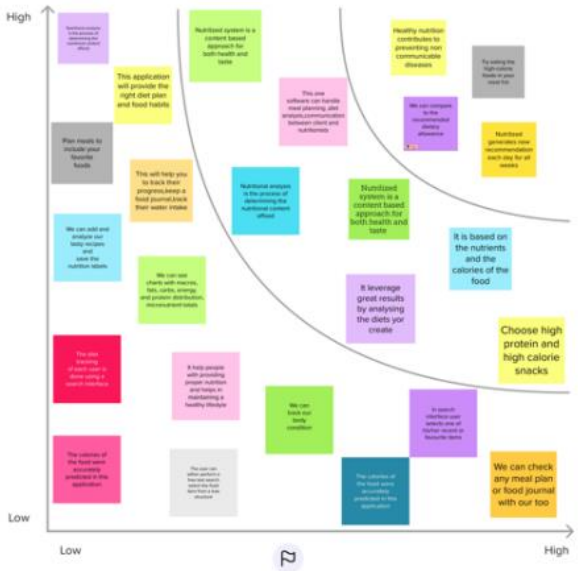
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible



Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasability

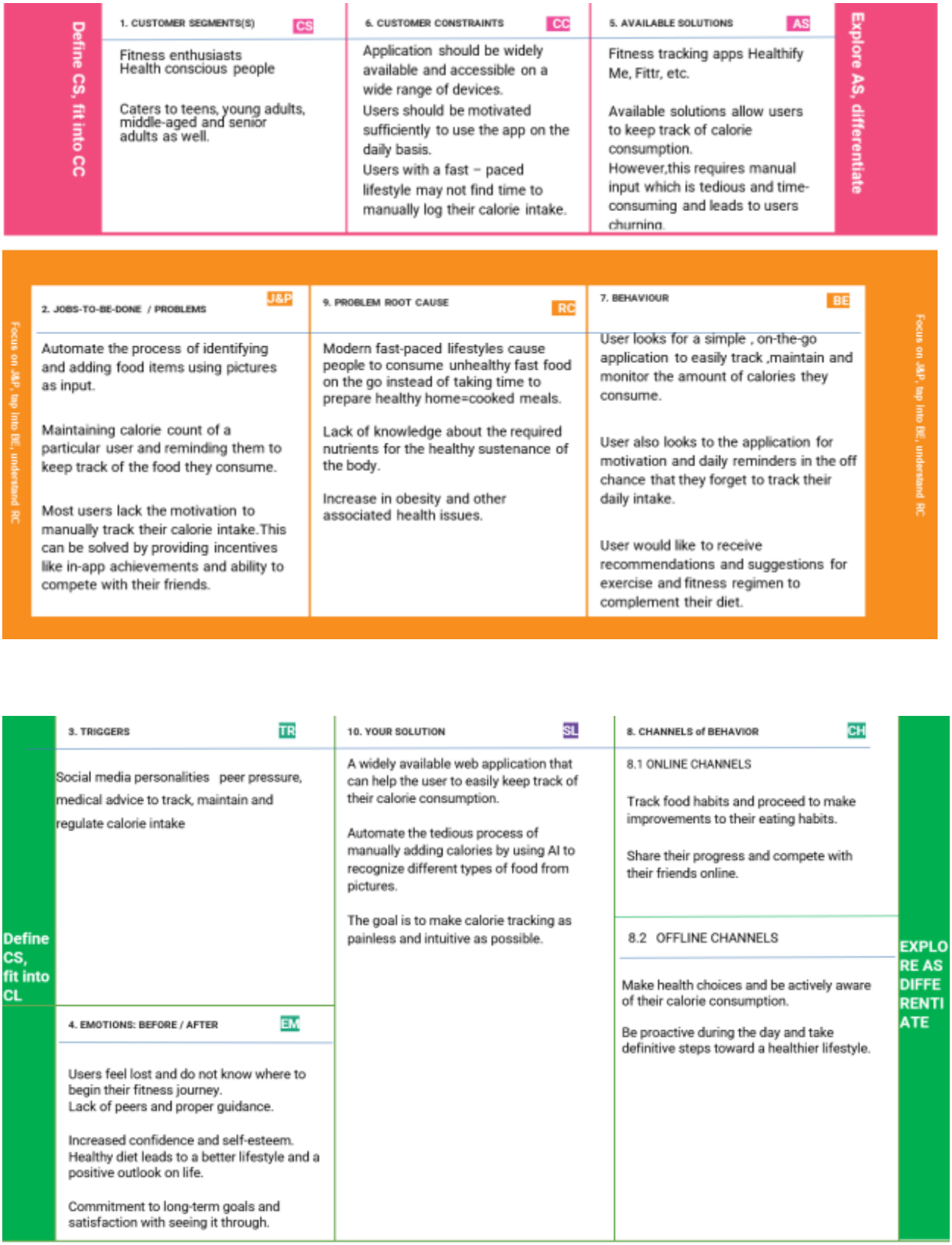
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution

S. no	Parameter	Description
1.	Problem Statement	Now a days, People are having lot of fast foods which causes many diseases likecardiac arrest etc. To avoid these diseases doctors are advised to follow a diet.
2.	Idea	The Solution to come up with the problem is , The user can know the nutritional content of the food that they are intaking, by taking picture of food and uploading it in the app. After uploading the image it gives us the nutrition present in the food like nutrients,calories etc.
3.	Novelty	This solution has the uniqueness that we can upload the images of the food which exactly looks like a real time and the nutritions of the food will be displayed. A web app that can automatically estimates food attributes such as ingredients and nutrition value by classifying the input image.

4.	Social Impact	People will lead a healthy lifestyle as the obesity rate will get reduced. It helps to achieve and maintain a healthy lifestyle. It is easy to follow without affecting their personal time and low cost expenditure.
5.	Business Model	To develop this model, Social media is the best platform As many people are spending their half of the time.So it reaches the people very easily than television.This application will boost the confidence among the people. It is a free platform for all the users for specific guidance the users have to pay.
6.	Scalability of the solution	This AI based applications are your fitness assistants. Using this people can access from anywhere at anytime to track their nutrients that will improve a healthy eating pattern. By following the proper diet people will lead a healthy lifestyle.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

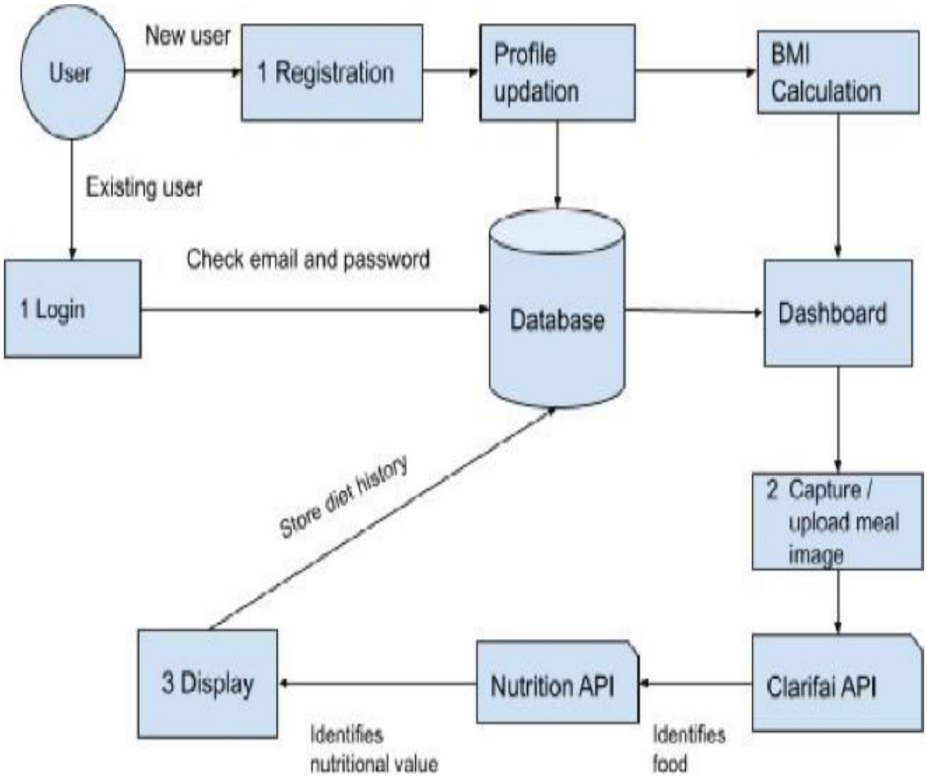
FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	☆ Registration through Form ☆ Registration through Gmail ☆ Registration through Facebook
FR-2	User Confirmation	☆ Confirmation via Email ☆ Confirmation via OTP
FR-3	User Login	☆ Login with Username ☆ Login with Password
FR-4	User Profile Update	☆ Update User's Name ☆ Update Portrait Photograph ☆ Update Date of Birth
FR-5	Uploading Food image	☆ Upload from Gallery ☆ Capture using Camera
FR-6	Enter Food name	☆ Type the name of the food
FR-7	Result	☆ Download Result ☆ Share Result through Social media
FR-8	Ratings and Reviews	☆ Share the experiences ☆ Provide Feedback

4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	☆ Accessible through INTERNET
NFR-2	Security	☆ Secure through unique Username and Password
NFR-3	Reliability	☆ Accurate result ☆ User friendly
NFR-4	Performance	☆ Using Standard algorithm to get faster and accurate results ☆ Clarifai's AI-Driven Food Detection Model is used
NFR-5	Availability	☆ Availabe for 24/7
NFR-6	Scalability	☆ It can be accessed by a greater number of users at the same time without any compromise in the performance.

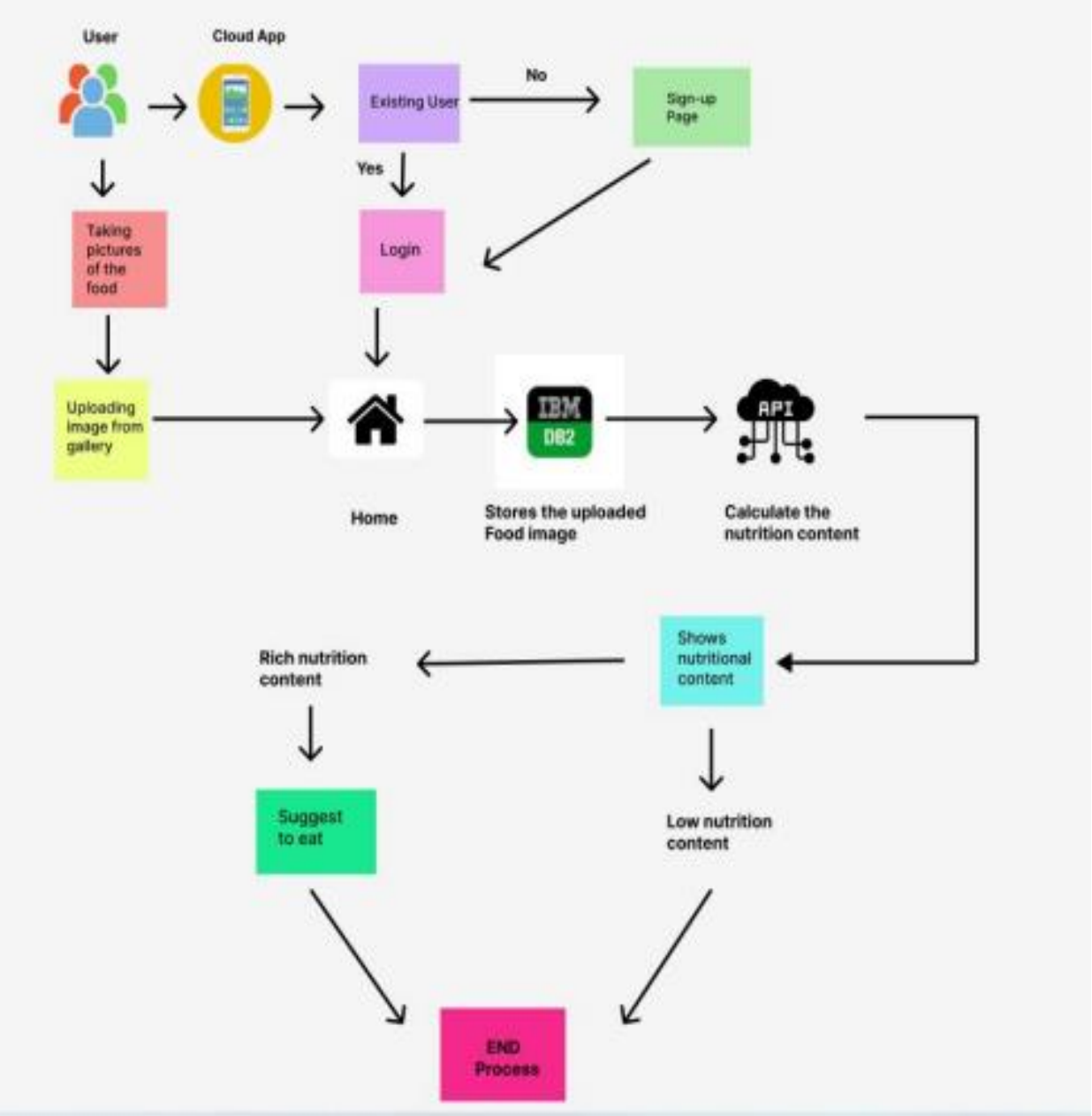
5. PROJECT DESIGN

5.1 Data Flow Diagrams

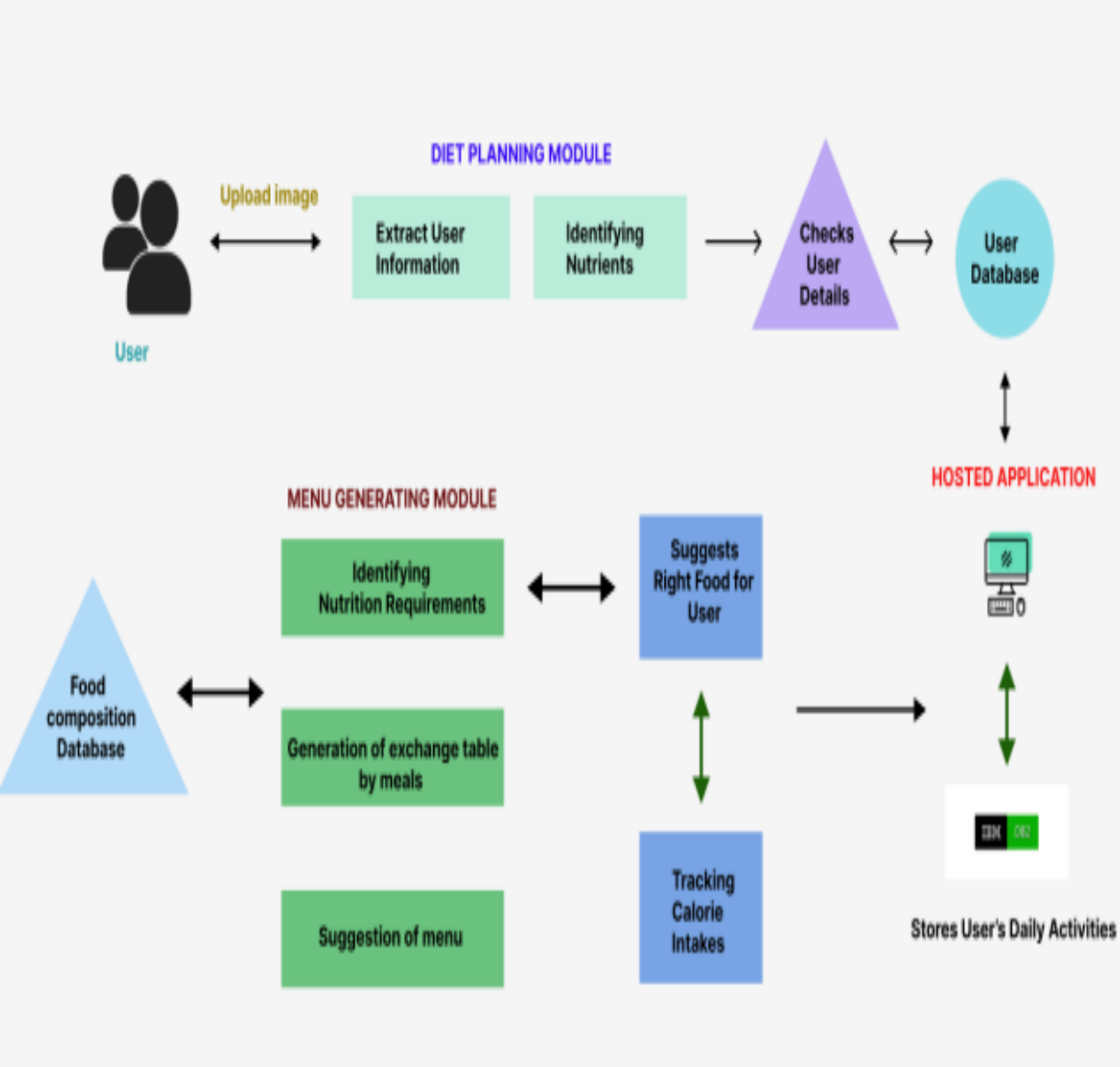


5.2 Solution & Technical Architecture

5.2.1 Solution Architecture



5.2.2 Technical Architecture



5.3 User Stories

User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer(Mobile Number)	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 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.	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password .	I can raise the issue in a ticket form.	High	Sprint-1
	Dashboard	USN-6	As a user, I will follow up with the application	I can see the agent progress on the issue being solved through mail	High	Sprint-1
Customer(Web User)	Home	USN-7	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-2
Customer Care Executive	Educational Qualification	USN-8	Act as a link between me and company	Customer care members resolves any queries which generated by customers.	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R
Sprint-1	User confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application.	1	High	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R
Sprint-2	User Details	USN-4	As a user, I can enter and update details	2	High	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R
Sprint-3	Food image scanning	USN-5	As a user, I can search the food items.	2	Medium	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R
Sprint-4	Show Nutritional Details	UNS-6	As a user, I can scan the food and get the nutritional details.	1	High	Pradheekshasri A.S Pradicsha C.A Preethi K Rheena R

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart

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	23 Oct 2022	28 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	04 Nov 2022
Sprint-3	20	6 Days	05 Nov 2022	10 Nov 2022	20	10 Nov 2022
Sprint-4	20	6 Days	12 Nov 2022	18 Nov 2022	20	18 Nov 2022

7. CODING & SOLUTIONING

7.1 Feature 1

Home page:

In addition to providing essential nutrients and micronutrients to the organism, food plays a crucial role in the immune and metabolic regulation of the organism. The nutritional quality of food has always been an important indicator in the evaluation of food, and the nutrition of food is very important, and many food-related nutrients are closely related to our health. However, the way to understand the nutritional quality of food while ensuring the efficiency and integrity of food has been a hot topic of research.

Home.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" />
    <link
      rel="stylesheet"
      href="{{url_for('static',filename='home-style.css')}}"
    />
    <title>HOME</title>
    <script src="https://kit.fontawesome.com/a076d05399.js"></script>
  </head>
  <body>
```

```
<header>
  <img
    class="logo"
    src="{{url_for('static',filename='NutriAux.png')}}"
    alt="logo"
  />
  <nav>
    <input type="checkbox" id="check" />
    <label for="check" class="checkbtn">
      <i class="fas fa-bars"></i>
    </label>
    <label class="nutri">NutriAux</label>
    <ul>
      <li><a href="{{url_for('reg')}}">Register</a></li>
      <li><a href="{{url_for('login')}}">Login</a></li>
      <li><a href="{{url_for('support')}}">Support</a></li>
    </ul>
  </nav>
</header>

<div class="bg-text">
  <p>
    <span>NutriAux</span> is a web app that aims at automatically estimating
    the food attributes such as ingredients and nutritional value by
    classifing the input image of the food. The person who wishes to use the
```

app must register before using the app. The user can login to their account to see thier respective dashboard with the details of height, weight, BMI that is automatically generated and the amount of calorie that can be taken according to the BMI of the user. The user can upload the image of the food that he has taken. The app will then display the nutritional value of the food. The user can take a note of it and can alter his food habits accoring to his BMI and amount of calories that can be intaken. The user can view the history of the food and the calorie intake details in the histroy page. The user can make use of the support in case of any queries.

```
    </p>
  </div>
  <section></section>
  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "cc205492-37a9-4863-8858-ddf7b525f0e8", // The ID of this integration.
      region: "us-south", // The region your integration is hosted in.
      serviceInstanceID: "37ac1e13-ba05-4364-9892-9f93f5d59696", // The ID of your service instance.
      onLoad: function (instance) {
        instance.render();
      },
    };
    setTimeout(function () {
      const t = document.createElement("script");
```

```
      t.src =
        "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
        (window.watsonAssistantChatOptions.clientVersion || "latest") +
        "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
    });
  </script>
 </body>
</html>
```

Register Page:

The register page asks the user details like User name, Email, Password. After clicking on the register button , we will check whether the user has already registered or not. If they have already registered, they will be directed to login page. If the person does not have an account, then they will receive a confirmation email and will be directed to personal details page.

Registration.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Responsive Registration Form</title>
  <meta
   name="viewport"
   content="width=device-width,
```

```
    initial-scale=1.0"
  />
  <link
    rel="stylesheet"
    href="{{url_for('static',filename='Registration-style.css')}}"
  />
</head>
<body>
  <div class="container">
    <h1 class="form-title">
      <img
        class="logo"
        src="{{url_for('static',filename='NutriAux.png')}}"
        alt="logo"
      /> Registration
    </h1>
    <form action="/register" method="post">
      <div class="main-user-info">
        <div class="user-input-box">
          <label for="firstName">First Name</label>
          <input
            type="text"
            id="firstName"
            name="firstName"
            placeholder="Enter First Name"
```

```
    required
  />
</div>
<div class="user-input-box">
  <label for="lastName">Last Name</label>
  <input
    type="text"
    id="lastName"
    name="lastName"
    placeholder="Enter Last Name"
    required
    autocomplete="off"
  />
</div>
<div class="user-input-box">
  <label for="email">Email</label>
  <input
    type="email"
    id="email"
    name="email"
    placeholder="Enter Email"
    required
    autocomplete="off"
  />
</div>
```

```
<div class="user-input-box">
  <label for="phoneNumber">Phone Number</label>
  <input
    type="text"
    id="phoneNumber"
    name="phoneNumber"
    placeholder="Enter Phone Number"
    required
    autocomplete="off"
  />
</div>
<div class="user-input-box">
  <label for="password">Password</label>
  <input
    type="password"
    id="password"
    name="password"
    placeholder="Enter Password"
    pattern="^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{5,}$"
    required
    autocomplete="off"
  />
</div>
<div class="user-input-box">
  <label for="confirmPassword">Confirm Password</label>
```

```
        <input
          type="password"
          id="confirmPassword"
          name="confirmPassword"
          placeholder="Confirm Password"
          pattern="^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{5,}$"
          required
          autocomplete="off"
        />
      </div>
      <div>
        <span style="color: white; font-size: 13px"
          >Password must be minimum Five characters with at least one letter
          and one number</span
        >
      </div>
    </div>
    <div class="form-submit-btn">
      <input type="submit" value="Register" />
    </div>
    <div class="go-back-btn">
      <a href="{{url_for('home')}}" class="goback">Go Back</a>
    </div>
  </form>
</div>
```

```
  </body>
</html>
```

Login Page:

The user will be asked to enter their registered username and password. After entering correct email and password the user have to click the login button. After that the user will be directed to their respective dashboard.

Login.html :

```
<! DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Login Form</title>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
    />
  </head>
  <style>
    body {
      padding-top: 50px;
      background-color: #f1f375d7;
      background-image: url("{{url_for('static',filename='calorie_calc_bg.png')}}");
      background-blend-mode: color-burn;
      background-size: 430px 400px;
    }
```

```
  .login-form {
    background: #dcf1f1;
    margin-top: 40px;
    margin-bottom: 100px;
    padding: 50px;
    border-radius: 50px;
    box-shadow: 10px 10px 5px 0px rgba(0, 0, 0, 0.75);
  }

  .btn-primary {
    width: 100%;
  }
</style>
<body>
  <div class="container" style="margin-top: 50px">
    <div class="row">
      <div class="col-md-8 offset-md-2">
        <div class="login-form">
          <img
            class="navbar-brand"
            src="{{url_for('static',filename='NutriAux.png')}}"
            alt="NutriAux"
            height="60px"
            width="60px"
            style="position: absolute; margin-left: 180px"
```

```
/>
<h1
  class="text-center"
  style="margin-top: 6px; padding-bottom: 20px"
>
  Login
</h1>

<form action="/verify" method="post">
  <div class="form-group">
    <label for="exampleInputEmail1">Enter Email address </label>
    <input
      type="email"
      name="email"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="emailHelp"
      placeholder="Enter email"
      required
      autocomplete="off"
    />
    <small id="emailHelp" class="form-text text-muted">
      We'll never share your email with anyone else.
    </small>
  </div>
```

```
<div class="form-group">
  <label for="exampleInputPassword1"> Enter Password </label>
  <input
    type="password"
    class="form-control"
    id="password"
    name="password"
    placeholder="Password"
    pattern="^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{5,}$"
    required
    autocomplete="off"
  />
</div>
<div class="form-group form-check">
  <input
    type="checkbox"
    class="form-check-input"
    onclick="showpass()"
  />
  <label class="form-check-label" for="exampleCheck1">
    Show Password
  </label>
</div>
<span style="color: red">{{message}}</span>
<button
```

```
            type="submit"
            class="btn btn-primary"
            style="margin-top: 10px"
          >
            Submit
          </button>
        </form>
      </div>
    </div>
  </div>
</div>
<script>
  function showpass() {
    var x = document.getElementById("password");
    if (x.type === "password") {
      x.type = "text";
    } else {
      x.type = "password";
    }
  }
</script>
<script
  src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
  integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
```

```
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
      integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
      integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
      crossorigin="anonymous"
    ></script>
    <script></script>
  </body>
</html>
```

PERSONAL DETAILS.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
```

```
    />

    <!-- Bootstrap CSS -->
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
      integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
      crossorigin="anonymous"
    />
    <title>Calorie Calculator</title>

    <style>
      body {
        background-color: #f1f375d7;
        background-image: url("{{url_for('static',filename='calorie_calc_bg.png')}}");
        background-blend-mode: color-burn;
        background-size: 430px 400px;
      }
      .results {
        display: none;
      }

      .btn {
        background: lighten(#06837f, 3%);
```

```
    font-weight: 600;
    letter-spacing: 1px;
   }
  </style>
</head>
<body>
  <div class="container" style="margin-top: 100px">
    <div class="row">
      <div class="col-lg-7 mx-auto">
        <div class="card card-body text-center mt-5">
          <h3 class="heading display-5 pb-3 mb-4">
            <img
              class="navbar-brand"
              src="{{url_for('static',filename='NutriAux.png')}}"
              alt="NutriAux"
              height="60px"
              width="60px"
            />
            Enter your personal details
          </h3>
          <form id="calorie-form" method="post" action="/addpersonaldetails">
            <div class="form-group row">
              <label for="age" class="col-sm-2 col-form-label">Age</label>
              <div class="col-sm-10">
                <input
```

```
      type="text"
      class="form-control"
      id="age"
      name="age"
      placeholder="Age > 17"
      pattern="(?:1[01][0-9]|120|1[7-9]|[2-9][0-9])$"
      required
      autocomplete="off"
    />
  </div>
</div>

<fieldset class="form-group">
  <div class="row">
    <legend class="col-form-label col-sm-2 pt-0">Gender</legend>
    <div class="col-sm-10" id="form-radio">
      <div
        class="custom-control custom-radio custom-control-inline"
      >
        <input
          type="radio"
          id="male"
          name="Gender"
          value="male"
          class="custom-control-input"
```

```
          checked="checked"
        />
        <label class="custom-control-label" for="male"
          >Male</label
        >
      </div>
      <div
        class="custom-control custom-radio custom-control-inline"
      >
        <input
          type="radio"
          id="female"
          name="Gender"
          value="female"
          class="custom-control-input"
        />
        <label class="custom-control-label" for="female"
          >Female</label
        >
      </div>
    </div>
  </div>
</fieldset>

<div class="form-group row">
```

```
  <label for="weight" class="col-sm-2 col-form-label"
    >Weight</label
  >
  <div class="col-sm-10">
    <input
      type="number"
      class="form-control"
      id="weight"
      name="weight"
      placeholder="In kilograms"
      required
    />
  </div>
</div>

<div class="form-group row">
  <label for="height" class="col-sm-2 col-form-label"
    >Height</label
  >
  <div class="col-sm-10">
    <input
      type="number"
      class="form-control"
      id="height"
      name="height"
```

```
      placeholder="In centimeters"
      required
    />
  </div>
</div>

<div class="form-group row">
  <legend class="col-form-label col-sm-2 pt-0">Activity</legend>
  <select
    class="custom-select col-sm-10 col-lg-9 ml-3"
    id="list"
    name="activity"
  >
    <option selected value="1">
      Sedentary (little or no exercise)
    </option>
    <option value="2">
      Lightly active (light exercise/sports 1-3 days/week)
    </option>
    <option value="3">
      Moderately active (moderate exercise/sports 3-5 days/week)
    </option>
    <option value="4">
      Very active (hard exercise/sports 6-7 days a week)
    </option>
```

```
                <option value="5">
                  Extra active (very hard exercise/sports & physical job or 2x
                  training)
                </option>
              </select>
            </div>

            <div class="form-group" style="margin-top: 70px">
              <input
                type="submit"
                value="Proceed to login page"
                class="btn btn-block"
                style="background-color: #06837f; color: white"
              />

          </form>
        </div>
      </div>
    </div>
  </div>

  <script
    src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
```

```
    crossorigin="anonymous"
  ></script>
  <script
    src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
    integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
    crossorigin="anonymous"
  ></script>
  <script
    src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
    integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
    crossorigin="anonymous"
  ></script>
 </body>
</html>
```

Upload.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Upload image</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```
    <link rel="stylesheet" href="{{url_for('static',filename='upload.css')}}">


  </head>

  <body >

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>


      <img class="navbar-brand" src="{{url_for('static',filename='NutriAux.png')}}" style="position:absolute" alt="NutriAux" height="60px" width="60px" >

    <form action="/getnutri" method="post" enctype="multipart/form-data">

    <div class="container uploadOuter ">

      <table class="uploadimage">

        <tr>

      <th><label for="uploadFile" id="file" ><h2>Choose your image to upload</h2><i class="fa fa-file-photo-o" style="font-size:48px;color:red"></i></i>


      </label></th>

      <th>  </th>

      <th><h3>OR</h3></th>


     <th><span class="dragBox" >

      <h2>Drag and Drop image here</h2><i class="fa fa-file-photo-o" style="font-size:48px;color:red"></i>

      <input name="file" type="file" onChange="dragNdrop(event)" ondragover="drag()" ondrop="drop()" id="uploadFile" required/>

    </span></th>
```

```
</tr>
</table>

    <button class="btn btn-primary btn-lg" id="b1" onclick="window.history.go(-2)">Go Back
</button>
    <button class="btn btn-primary btn-lg" id="b1" type="submit" >Submit </button>

    </div>
   </form>

    <div>
     <table class="table table-bordered">
        <tr>
          <th scope="col" style="background-color:#000c66;color:white;">Calories</th>
          <td style="background-color:white;font-weight: bolder;">{{calories}}</td>
        </tr>
        <tr>
          <th scope="col" style="background-color:#000c66;color:white;">Protein</th>
          <td style="background-color:white;font-weight: bolder;">{{protein}}</td>

        </tr>
        <tr>
          <th scope="col" style="background-color:#000c66;color:white;">Fat</th>
          <td style="background-color:white;font-weight: bolder;">{{fat}}</td>
```

```
            </tr>
            <tr>
              <th scope="col" style="background-color:#000c66;color:white;">Carbs</th>
              <td style="background-color:white;font-weight: bolder;">{{carbs}}</td>
            </tr>

          </table>
        </div>

          <div id="preview" style="height:105px">  </div>

        </div>
        </div>
        <script>
          "use strict";
function dragNdrop(event) {
    var fileName = URL.createObjectURL(event.target.files[0]);
    var preview = document.getElementById("preview");
    var previewImg = document.createElement("img");
    previewImg.setAttribute("src", fileName);
    previewImg.setAttribute("width", "100px");
    previewImg.setAttribute("height", "100px");
    preview.innerHTML = "";
    preview.appendChild(previewImg);
}
```

```
function drag() {
    document.getElementById('uploadFile').parentNode.className = 'draging dragBox';
}
function drop() {
    document.getElementById('uploadFile').parentNode.className = 'dragBox';
}
        </script>
        <script>
          window.watsonAssistantChatOptions = {
            integrationID: "cc205492-37a9-4863-8858-ddf7b525f0e8", // The ID of this integration.
            region: "us-south", // The region your integration is hosted in.
            serviceInstanceID: "37ac1e13-ba05-4364-9892-9f93f5d59696", // The ID of your service instance.
            onLoad: function (instance) {
              instance.render();
            },
          };
          setTimeout(function () {
            const t = document.createElement("script");
            t.src =
              "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
              (window.watsonAssistantChatOptions.clientVersion || "latest") +
              "/WatsonAssistantChatEntry.js";
            document.head.appendChild(t);
          });
```

```
        </script>
    </body>
</html>
```

7.2 FEATURE 2

Home-Style.css

```
* {
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  font-family: verdana;
  box-sizing: border-box;
}
body {
  background-color: #fdc100;
}
span {
  font-size: 30px;
  text-align: center;
  font-weight: bold;
}
img.logo {
  height: 60px;
  width: 80px;
  margin: 10px;
```

```
}
header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
nav {
  background-color: #fdc100;
  height: 80px;
  width: 100%;
}
label.nutri {
  padding: 0 30px;
  color: black;
  font-size: 25px;
  line-height: 80px;
  font-weight: bold;
}
nav ul {
  float: right;
  margin-right: 20px;
}
nav ul li {
  display: inline-block;
  line-height: 80px;
```

```
  margin: 0 5px;
}
nav ul li a {
  color: black;
  font-size: 17px;
  border: 1px solid transparent;
  padding: 7px 13px;
  border-radius: 3px;
}
a.active,
a:hover {
  background: #80cc66;
  border: 1px solid white;
  transition: 0.5s;
}
.checkbtn {
  font-size: 30px;
  color: crimson;
  float: right;
  line-height: 80px;
  margin-right: 40px;
  cursor: pointer;
  display: none;
}
#check {
```

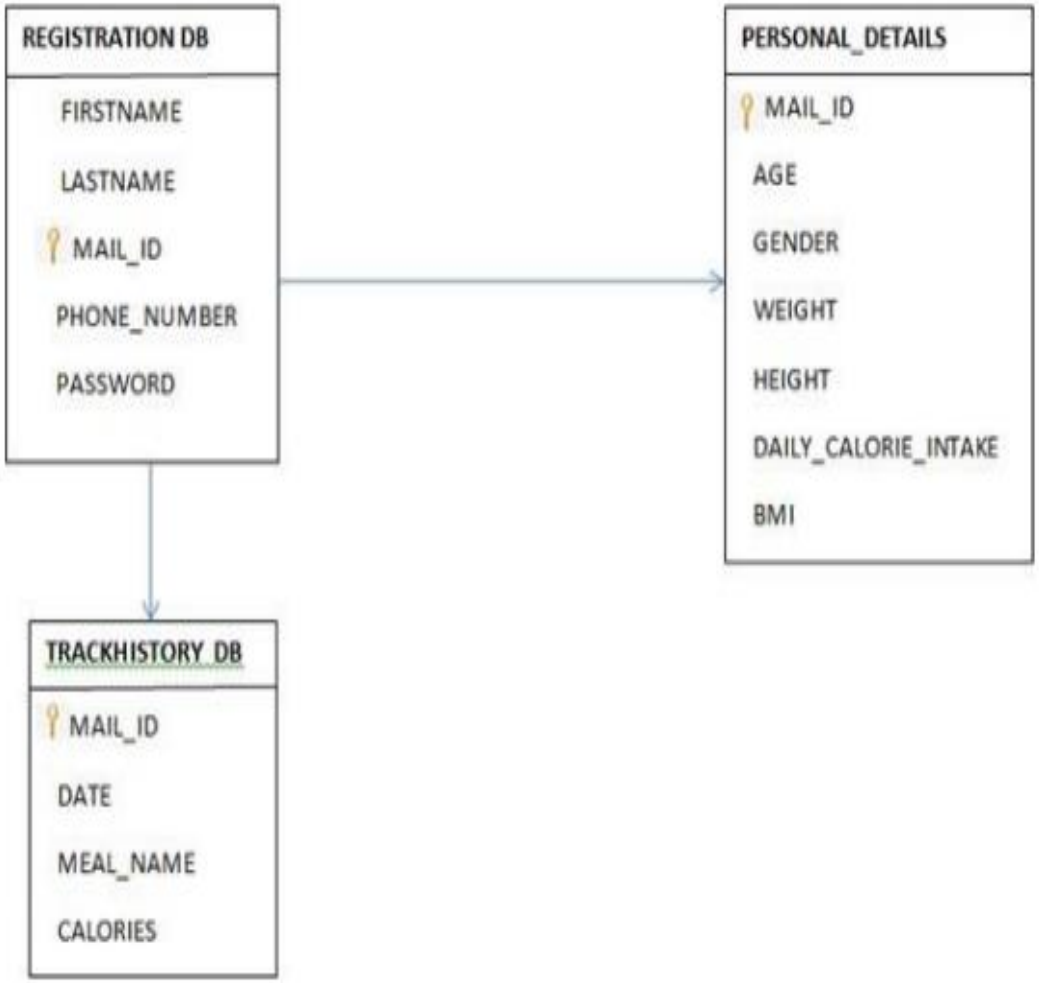
```
  display: none;
}
.bg-text {
  background-color: rgb(0, 0, 0); /* Fallback color */
  background-color: rgba(0, 0, 0, 0.7); /* Black w/opacity/see-through */
  color: white;
  border: 3px solid #f1f1f1;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  z-index: 2;
  width: 80%;
  padding: 10px;
  text-align: justify;
  font-size: 22px;
}

@media (max-width: 992px) {
  label.nutri {
    font-size: 30px;
    padding-left: 50px;
  }
  nav ul li a {
    font-size: 16px;
```

```
  }
  .bg-text {
    margin-top: 50px;
    top: 50%;
    left: 50%;
  }
}
@media (max-width: 858px) {
  .checkbtn {
    display: block;
  }
  label.nutri {
    display: none;
  }
  .bg-text {
    font-size: 15px;
    top: 50%;
    left: 50%;
  }
  ul {
    position: fixed;
    width: 100%;
    height: 100vh;
    background: #ffee99;
    z-index: 20;
```

```
    top: 80px;
    left: -100%;
    text-align: center;
    transition: all 0.5s;
  }
  nav ul li {
    display: block;
    margin: 50px 0;
    line-height: 30px;
  }
  nav ul li a {
    font-size: 20px;
    color: black;
  }
  a.active,
  a:hover {
    background: none;
    border: none;
    color: #f26d1b;
  }
  #check:checked ~ ul {
    left: 0;
  }
}
section {
```

```
  width: 100%;
  background: url(food2.jpg);
  background-position: center;
  background-size: cover;
  height: calc(100vh - 80px);
  background-repeat: no-repeat;
}
```

7.3 DATABASE SCHEMA



8. TESTING

8.1 Test Cases

Sprint 2:

Test case ID	Test Scenario	Expected Result	Status
Dashboard_TC_OO1	Verify user is able to see their height, weight, bmi and calorie intake	Verify these are available 1.Height 2.Weight 3.BMI 4.Calorie intake 5.Upload image button 6.Track history button	Pass
Dashboard_TC_OO2	Verify whether upload image button works	Redirected to upload image page	Pass
Dashboard_TC_OO3	Verify whether track history button works	Redirected to History page	Pass
Uploadimage_TC_OO1	Check the choose file option available	Able to view the 1.Choose file 2.Submit button 3.Go back button	Pass
Uploadimage_TC_OO2	Verify whether food image can be uploaded	Preview of the image uploaded will be displayed	Pass
Uploadimage_TC_OO3	Verify whether it alerts when no image is uploaded	The alert will show "Please upload the file"	Pass
Trackhistory_TC_OO1	Verify whether history table displayed	1.Date picker text box 2.Food name text box 3.Calorie text box 4.Add button 5.The track history table with date,food name and calorie value	Pass
Trackhistory_TC_OO2	Verify whether add button works	1.Chose Date 2.Enter food 3.Enter Calorie 4.Click add button 5.The data will be added to the table displayed	Pass

Sprint 3:

Test case ID	Test Scenario	Expected Result	Status
Registrationdatabase_TC_OO1	Verify whether registration credentials are added to the cloud database	The given credentials by the user should be same as the credentials stored in ibm cloud database	Pass
Personaldetailsdatabase_TC_OO1	Verify whether personal details credentials are added to the cloud database	The given credentials by the user should be same as the credentials stored in ibm cloud database	Pass
Track_historydatabase_TC_OO1	Verify whether added food details are added to the cloud database	The added food details by the user should be same as the details stored in ibm cloud database	Pass
Trackhistory_TC_OO2	Verify whether add button works	1.Chose Date 2.Enter food 3.Enter Calorie 4.Click add button 5.The data will be added to the table displayed	Pass

8.2 User Acceptance Testing

Sprint 2:

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	2	3	2	4	11
Duplicate	1	0	1	0	2
External	1	1	1	1	4
Fixed	7	3	2	10	22
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	11	7	6	15	39

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
Dashboard	3	0	0	3
Upload Image	4	0	0	4
TrackHistory Page	4	0	0	4

Sprint 3:

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	3	3	2	4	12
Duplicate	2	0	3	0	5
External	1	1	1	1	4
Fixed	9	3	2	11	25
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	15	7	8	16	46

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
Personal details Database	1	0	0	1
Track History Database	2	0	0	2
Registration Database	1	0	0	1
Track History Page	4	0	0	4

Sprint 4:

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	1	4	2	4	11
Duplicate	1	0	1	0	2
External	1	0	0	1	2
Fixed	10	3	4	10	27
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	13	7	7	15	42

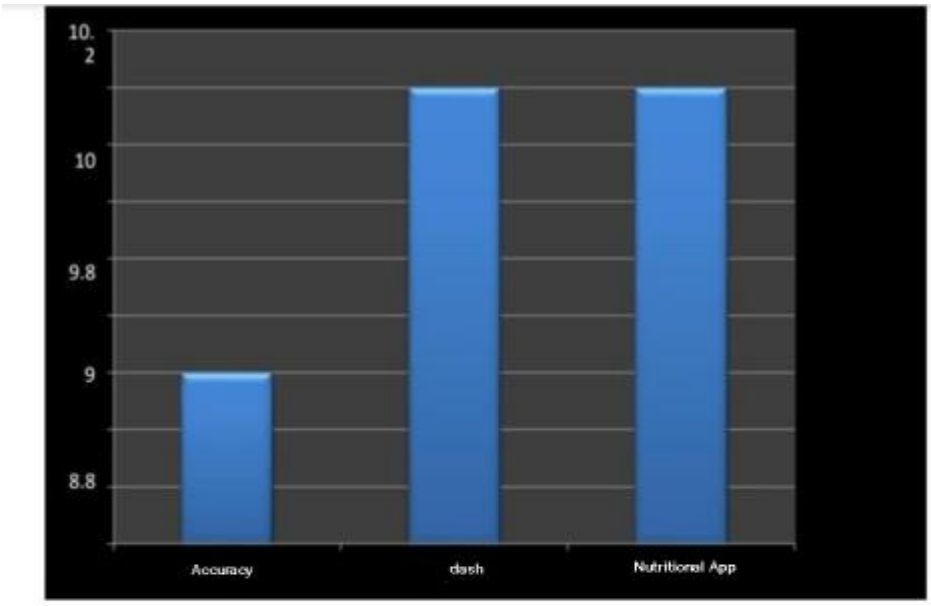
3. Test Case Analysis

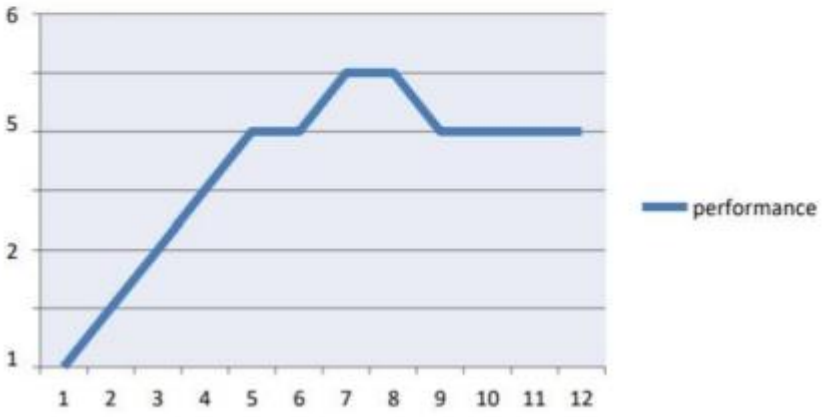
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Upload Image	3	0	0	3
Clarifai API	1	0	0	1
Spoonacular Nutrition API	1	0	0	1

9. RESULTS

9.1 Performance Metrics





10. ADVANTAGES & DISADVANTAGES

Advantages:

- By using our webapp, the user can know their BMI, which will lead the user to decide whether he has to gain weight or lose weight.
- User can know their daily calorie intake, which can help them to know amount of calorie they can consume for that particular day.
- The user can upload the image of the meal which will provide them the nutritional value of that particular meal.
- Nutrition Application is a user friendly and easy to use application.
- The user can track the daily calorie intake which will help them to know their progress towards their fitness goal.

Disadvantages:

- It requires an active internet connection.
- Not all types of foods can be detected correctly by Clarifai Food Detection Model API.
- The user cannot update their personal details once it has been registered.

11. CONCLUSION

Since obesity rate has become a major problem in this decade, the diet management is very important. 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 Application helps in finding the nutritional content present in the food with real time image processing using Clarifai Food Detection Model API and Spoonacular Nutrition API. The user can upload his daily meal image and get the nutritional value. They can also track their daily calorie intake.

12. FUTURE SCOPE

Nutrition Application will be upgraded in the following years with the feature of “Profile Updation”. The user can update his personal details like height, weight and age which will help them to keep track of the daily calorie intake and the BMI. “Dietary Recommendation” facility and “Water Reminder” facility will also be added in the future.

13.APPENDIX

Source Code

```
# -*- coding: utf-8 -*-

"""

Created on Wed Nov  2 19:40:29 2022

@author: admin

"""

from flask import Flask,render_template,request,url_for,redirect,session

from clarifai_grpc.grpc.api import service_pb2,resources_pb2

from clarifai_grpc.grpc.api.status import status_code_pb2

import ibm_db

import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

import requests

from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel

from clarifai_grpc.grpc.api import service_pb2_grpc

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

YOUR_CLARIFAI_API_KEY="95609b99ed7b4f25a095ce5e9c1d171d"

YOUR_APPLICATION_ID="NutritionAssistantApp"
```

```
app=Flask(__name__)

app.secret_key='a'

try:

 conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=kpd37336;PWD=KsEjKpkM6P5Kz7Lk","","")

except:

    print("Unable to connect: ",ibm_db.conn_error())

  @app.route("/")

def home():

    session['status_msg']=' '

    return render_template('Home.html')

@app.route("/reg")

def reg():

    return render_template('Registration.html')

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

def register():

     if request.method == 'POST' :

        firstName = request.form['firstName']

        lastName = request.form['lastName']
```

```
        session['email'] = request.form['email']
        phoneNumber = request.form['phoneNumber']
        password = request.form['password']
        sql = "SELECT * FROM registration WHERE EMAIL_ID=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,session['email'])
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        message =
Mail(from_email='pradheeksha15@gmail.com',to_emails=session['email'],subject="NutritionAs
sitant - Registration",html_content='<b>NutritionAssitant welcomes you</b><br/><p>Your
account has been registered successfully</p>')
        try:
            #USe the API key given in this link for security purposes -
https://docs.google.com/document/d/1xrF_chjAgbNJOCcsrGuVXtKWexPb5Ff5vmLpkscbgbU/e
dit?usp=sharing
            #sg=SendGridAPIClient('##USE THE API GIVEN IN THE ABOVE LINK##')
            response=sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)
```

```
        except Exception as e:
            print(e)
       if account:
            session['status_msg']= 'Account already exists ! Kindly login'
            return redirect(url_for('login'))
        else :
            insert_sql = "INSERT INTO registration VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, firstName)
            ibm_db.bind_param(prep_stmt, 2, lastName)
            ibm_db.bind_param(prep_stmt, 3, session['email'])
            ibm_db.bind_param(prep_stmt, 4, phoneNumber)
            ibm_db.bind_param(prep_stmt, 5, password)
            ibm_db.execute(prep_stmt)
        print('You have successfully registered !')
        return redirect(url_for('personaldetails'))
@app.route("/personaldetails")
def personaldetails():
    return render_template("personaldetails.html")
@app.route("/addpersonaldetails",methods=["POST","GET"])
def addpersonaldetails():
```

```
if request.method == 'POST' :
    age=float(request.form.get('age'))
    gender=request.form.get('Gender')
    weight=float(request.form.get('weight'))
    height=float(request.form.get('height'))
    activity=request.form.get('activity')
    print(age,gender,weight,height,activity)
    if(gender == 'male'and activity == "1"):
        totalCalories = 1.2 * (66.5 + (13.75 * weight) + (5.003 * height) - (6.755 * age))
    elif(gender == 'male' and activity == "2"):
        totalCalories = 1.375 * (66.5 + (13.75 * weight) + (5.003 * height) - (6.755 * age))
    elif (gender == 'male' and activity == "3"):
        totalCalories = 1.55 * (66.5 + (13.75 * weight) + (5.003 * height) - (6.755 * age))
    elif(gender == 'male' and activity == "4"):
        totalCalories = 1.725 * (66.5 + (13.75 * weight) + (5.003 * height) - (6.755 * age))
    elif(gender == 'male' and activity == "5"):
        totalCalories = 1.9 * (66.5 + (13.75 * weight) + (5.003 * height) - (6.755 * age))
    elif(gender == 'female' and activity == "1"):
        totalCalories = 1.2 * (655 + (9.563 * weight) + (1.850 * height) - (4.676 * age))
    elif(gender == 'female' and activity == "2"):
        totalCalories = 1.375 * (655 + (9.563 * weight) + (1.850 * height) - (4.676 * age))
```

```
elif(gender == 'female' and activity == "3"):
    totalCalories = 1.55 * (655 + (9.563 * weight) + (1.850 * height) - (4.676 * age))
elif(gender == 'female' and activity == "4"):
    totalCalories = 1.725* (655 + (9.563 * weight) + (1.850 * height) - (4.676 * age))
else:
    totalCalories = 1.9 * (655 + (9.563 * weight) + (1.850 * height) - (4.676 * age))
print(int(totalCalories))
BMI = (weight / (height/100)**2 )
if BMI <= 18.5:
    BMI_message="underweight"
elif BMI <= 24.9:
    BMI_message="healthy"
elif BMI <= 29.9:
    BMI_message="overweight"
else:
    BMI_message="obese"
print(BMI)
insert_query="INSERT INTO personal_details VALUES(?,?,?,?,?,?,?)"
prep_stmt=ibm_db.prepare(conn,insert_query)
ibm_db.bind_param(prep_stmt,1,session['email'])
ibm_db.bind_param(prep_stmt,2,str(int(age)))
```

```
        ibm_db.bind_param(prep_stmt,3,gender)
        ibm_db.bind_param(prep_stmt,4,str(weight))
        ibm_db.bind_param(prep_stmt,5,str(height))
        ibm_db.bind_param(prep_stmt,6,str(totalCalories))
        ibm_db.bind_param(prep_stmt,7,str(BMI))
        ibm_db.execute(prep_stmt)
    return redirect(url_for('login'))
@app.route("/login")
def login():
    return render_template("login.html",message=session['status_msg'])
@app.route("/verify",methods=["POST","GET"])
def verify():
    session['email'] = request.form.get("email")
    password = request.form.get("password")
    get_query="SELECT * FROM registration WHERE EMAIL_ID=? AND PASSWORD=?"
    prep=ibm_db.prepare(conn,get_query)
    ibm_db.bind_param(prep,1,session['email'])
    ibm_db.bind_param(prep,2,password)
    result=ibm_db.execute(prep)
    login = ibm_db.fetch_assoc(prep)
    if login:
```

```
        get_query="SELECT weight,height,daily_calorie_intake,BMI FROM personal_details
WHERE EMAIL_ID=?"
        prep=ibm_db.prepare(conn,get_query)
        ibm_db.bind_param(prep,1,session['email'])
        result=ibm_db.execute(prep)
        data = ibm_db.fetch_tuple(prep)
        global weight
        weight= data[0]
        global height
        height= data[1]
        global daily_calorie_intake
        daily_calorie_intake=data[2]
        daily_calorie_intake=daily_calorie_intake[0:7]
        global BMI
        BMI=data[3]
        BMI=BMI[0:4]
        return redirect((url_for('dashboard')))
    print("Wrong password" , session['email'],password)
    return render_template("login.html",message="Incorrect Email ID or Password! Try again")
@app.route("/dashboard")
def dashboard():
```

```
    return
render_template('dashboard.html',weight=weight,height=height,daily_calorie_intake=daily_calor
ie_intake,BMI=BMI)

@app.route("/upload")

def upload():

    return render_template('upload.html',calories="0 calories",fat="0 g",protein="0 g",carbs="0
g")

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

def getnutri():

    if request.method == "POST":

        try:

img = request.files['file']

            print("working")

            path='./static/'+session['email']+'.jpg'

            img.save(path)

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

            with open(path,"rb") as f:

                file_bytes=f.read()

                request1=service_pb2.PostModelOutputsRequest(

                    model_id='329f535676154599b313b413a71f73b4',

                    inputs=[

                        resources_pb2.Input(
```

```
                    data=resources_pb2.Data(
                        image=resources_pb2.Image(
                            base64=file_bytes
                        )
                    )
                )
            ])
        response =stub.PostModelOutputs(request1, metadata=metadata)
        if response.status.code != status_code_pb2.SUCCESS:
            raise Exception("Request failed, status code: " + str(response.status.code))
for concept in response.outputs[0].data.concepts:
            print('%12s: %.2f' % (concept.name, concept.value))
        api_url = 'https://api.spoonacular.com/recipes/guessNutrition?title='
        query = response.outputs[0].data.concepts[0].name
        response = requests.get(api_url + query, headers={'X-Api-Key':
'8f123f2f983b4b69bfe1e4a25f7bfb06'})
        if response.status_code == requests.codes.ok:
            fullresponse=response.json()
            calories=str(fullresponse['calories']['value'])+' '+str(fullresponse['calories']['unit'])
            protein=str(fullresponse['protein']['value'])+' '+str(fullresponse['protein']['unit'])
            fat=str(fullresponse['fat']['value'])+' '+str(fullresponse['fat']['unit'])
```

```
            carbs=str(fullresponse['carbs']['value'])+' '+str(fullresponse['carbs']['unit'])
            print(calories,protein,fat,carbs)
            print(type(fullresponse['calories']['value']))
            print(fullresponse)
        else:
            print("Error:", response.status_code, response.text)
        return
render_template('upload.html',calories=calories,fat=fat,protein=protein,carbs=carbs)
    except Exception as e:
        print(e)
    return "Error Occured"
@app.route("/history")
def history():
    get_query="SELECT CHAR(DATE_OF_CONSUMPTION,EUR)AS
DATE,MEAL_NAME,CALORIES FROM TRACKHISTORY WHERE EMAIL_ID=?"
    prep=ibm_db.prepare(conn,get_query)
    ibm_db.bind_param(prep,1,session['email'])
    result=ibm_db.execute(prep)
    if result==False:
        print("not working")
    history=[]
    dictionary = ibm_db.fetch_assoc(prep)
```

```
    while dictionary != False:
        history.insert(0,dictionary["DATE"])
        history.insert(1,dictionary["MEAL_NAME"])
        history.insert(2,dictionary["CALORIES"])
      print("The date is : ", dictionary["DATE"])
        print("The name is : ", dictionary["MEAL_NAME"])
        print("The calories is : ", dictionary["CALORIES"])
        dictionary = ibm_db.fetch_assoc(prep)
    print(history)
    no_of_rows=len(history)//3
    print(no_of_rows)
    return render_template("History.html",history=history,no_of_rows=no_of_rows)
@app.route("/addhistory",methods=["POST","GET"])
def addhistory():
    if request.method=='POST':
        date=request.form['date']
        meal_name=request.form['meal_name']
        calories=request.form['calories']
        insert_query="INSERT INTO trackhistory VALUES(?,?,?,?)"
        prep_stmt=ibm_db.prepare(conn,insert_query)
        ibm_db.bind_param(prep_stmt,1,session['email'])
```

```
        ibm_db.bind_param(prep_stmt,2,date)
        ibm_db.bind_param(prep_stmt,3,meal_name)
        ibm_db.bind_param(prep_stmt,4,calories)
        ibm_db.execute(prep_stmt)
    return redirect(url_for('history'))
 @app.route("/support")
def support():
    return render_template("support.html")
if __name__ == '__main__':
    app.run(debug=True)
```

GitHub & Project Demo Link :

IBM-EPBL/IBM-Project-37270-1660302929 - github.com