

Final Report

Project Name-Nutrition Assistant
Application using Cloud Computing

Team Members

Allan Jacob P- Team Lead
Abishek R- Team Member 1
Janesh C- Team Member 2
Achutha Mallikarjuna – Team Member 3

Project Report Format

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule

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

- 7.1 Feature 1
- 7.2 Feature 2

8. TESTING

- 8.1 Test Cases
- 8.2 User Acceptance Testing

9. ADVANTAGES & DISADVANTAGES

10. CONCLUSION

11. APPENDIX

Source Code

1 INTRODUCTION

1.1 Project Overview

In project we build a nutrition assistant application using Cloud Computing. It accepts the food that the user wants to eat and find the calories of the food, using Nutrition API. Through this ,user can find the amount of calories present in their food and can monitor and maintain their diet.

1.2 Purpose

Healthy nutrition contributes to preventing non-communicable and diet-related diseases.The World Health Organization (WHO) reported that non-communicable diseases (NCDs) accounted for 71% of deaths worldwide each year,

identifying unhealthy diets, smoking, and lack of exercise as major risk factors for NCDs. Dietary assessment and monitoring are essential steps to measure dietary intake and provide tailored advice that can improve dietary management and health.

In this

project we are going to integerate data analysis with cloud technologies which will show us accurate assessment of dietary

intake and customized feedback Using Nutrition API from rapidapi.com and integrating with cloud technologies could be used to measure dietary intake or improve the measurement of dietary. This project helps the people to guide and maintain their nurtitional health under all circumstances.

2 Literature Survey

2.1 Existing Problem

2.1 References

TITLE AND AUTHOR(S)	YEAR	TECHNIQUE (S)	FINDINGS	PROS AND CONS
Enhancing Cloud and healthy Food Nutrition Information Systems Practice- Paul, PK and Aithal, PS and Bhuimali, A	2017	Cloud Computing, Mobile Computing	Among the common mass food information systems are not yet popularized as a domain and thus there are huge potentialities to work on this.	P: Regarding manpower development there are a lot of things are pending and possible to work with. Hence cloud will do an attention on skill and manpower development for sophisticated development of food information systems.
Mobile cloud based system recognizing nutrition and freshness of food image- Kumbhar, Diptee and Patil, Sarita	2017	Cloud Computing, Image Segmentation	Mobile cloud computing (MCC) has been introduced to be a potential paradigm for mobile health services to overcome the interoperability issues over distinctive information formats. In this, we propose a mobile cloud-based food calorie measurement framework.	P: Multiple Platform Support Cost-Efficient C: Connectivity and Performance Issues

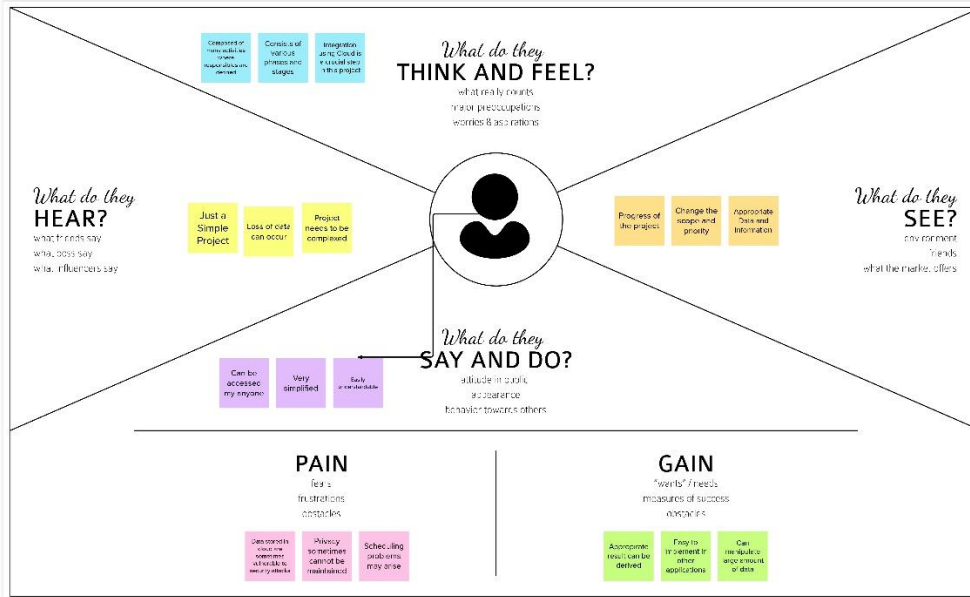
<p>Predicting calorific value for mixed food using image processing-</p> <p>Kohila, R and Meenakumari, R</p>	<p>2017</p>	<p>Cloud Computing, Image Segmentation</p>	<p>The objective of this paper is to predict and to fix diet control for various diseases by measuring the calorific value to help the patients and nutritionists. The image captured through a mobile phone/tablet camera will provide information concerning the calorie rate of the food.</p>	<p>P: Increased security Reduced cost</p> <p>C: Limited control Lacks Support</p>
<p>Use of artificial intelligence in precision nutrition and fitness-</p> <p>de Moraes Lopes, Maria Helena Baena and Ferreira, Danton Diego and Ferreira, Ana Claudia Barbosa Honorio and da Silva, Giuliano Roberto and Caetano, Aletha Silva and Braz</p>	<p>2020</p>	<p>Artificial Intelligence, Nutritional surveillance</p>	<p>Among the available computational tools, artificial intelligence (AI) has gained more and more attention recently, since it is able to learn and model linear and nonlinear relationships between variables by constructing an input-output mapping such that hidden and extremely useful information for decision-making is revealed and interpret.</p>	<p>P: A large amount of data is collected by these technologies</p> <p>C: AI is not yet widely used in the areas of nutrition and fitness</p>

3 IDEATION & PROPOSED SOLUTION

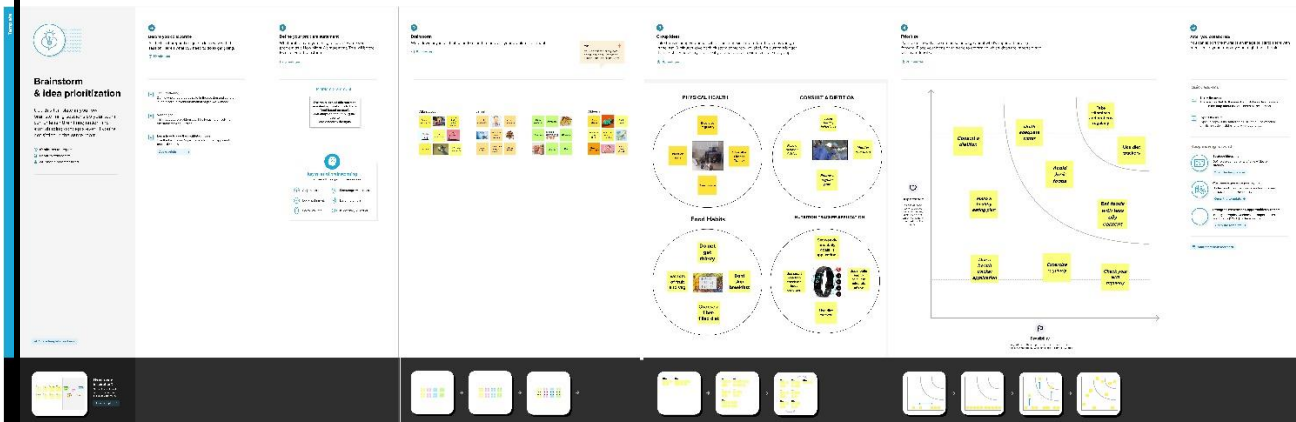
3.1 Empathy Map Canvas

IBM-Nalaiyathiran

Nutrition Assistant Application using Cloud Computing



3.2 Ideation & Brainstorming



3.3 Proposed Solution

The application will get the food, the user wants to eat and finds the amount of calories present in that food. The data received will be integrated in cloud. Using that data user can manage and regulate always.

Novelty/Uniqueness :-

- Deploys the model in Cloud
- Helps in finding the calories of food
- User can monitor regularly
- Details are being updated at time-basis

Social Impact/Customer Satisfaction :-

- Promotes Simplicity
- Promotes Self-Diagnosis
- Requires minimal effort and time
- Proposed solution abides by privacy laws and no private information of user is stored
- Delivers highly accurate results(classification of arrhythmia) in a short span of time.

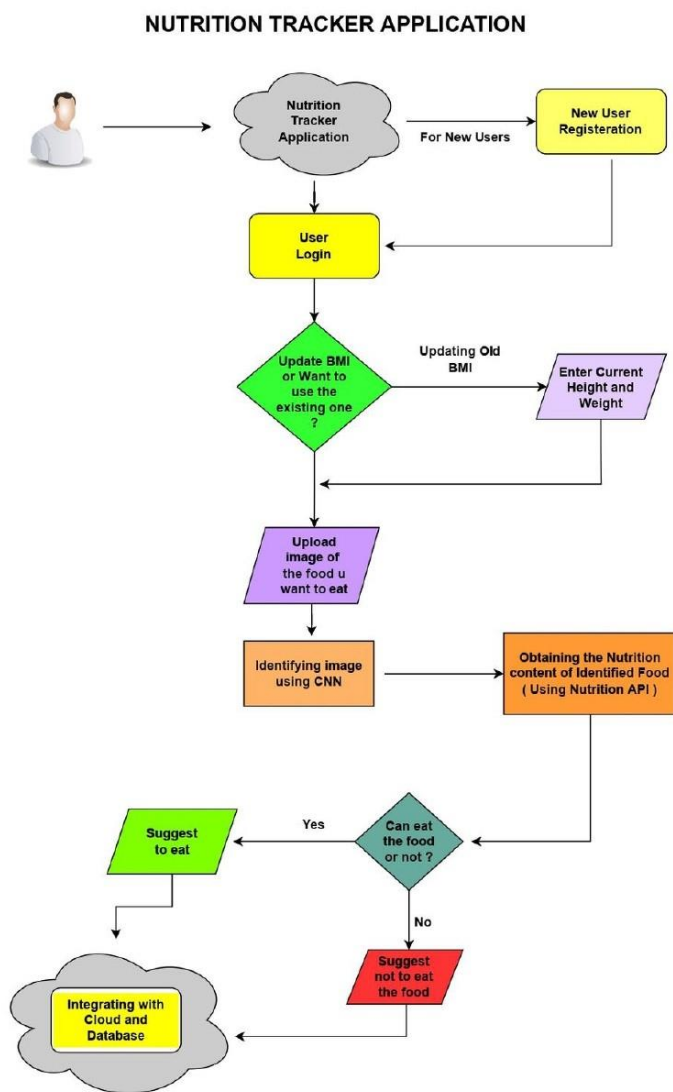
Business Model

- Our business model primarily covers the expense we incur by deploying the service in cloud platforms
- Primary consumers of our proposed service are hospitals who seek immediate consultation or use our service as a reference.
- Our service can be used by anybody who has access to internet services.
- It can be used by everybody else

Scalability of the Solution

- Can handle all type of food images
- Process all times of food and gives the calories of all types of food
- Increases Scalability and security

3.4 Problem Solution Fit



4 REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
FR-1	User Registration	Registration through email
FR-2	User Confirmation	Confirmation through email
FR-3	User Data	Collection of all required input data
FR-4	Data Analysis	Process Inputs using Nutrition API
FR-5	Data Processing	Evaluate and store in cloud containers
FR-6	Provide Output to User	Display results to user

4.2 Non - Functional Requirements

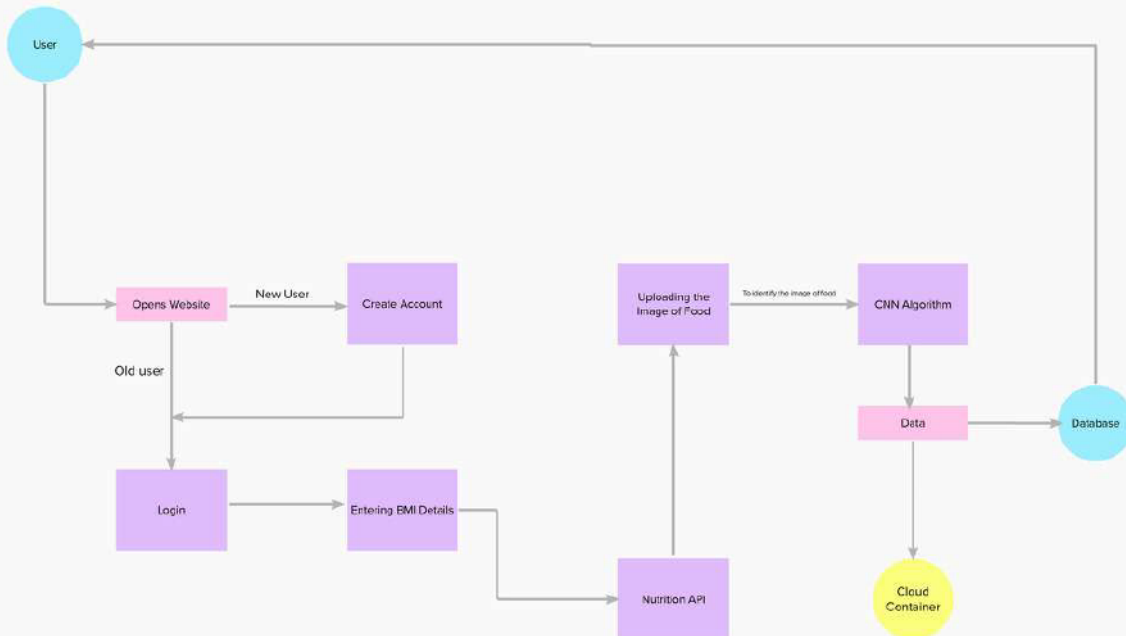
Following are the Non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
NFR-1	Usability	User-friendly and overall satisfaction of the user while using the website
NFR-2	Security	The website provides proper authentication and verification
NFR-3	Reliability	The site always provides reliable outputs and lacks failures
NFR-4	Availability	Provides 100% efficiency of the output
NFR-5	Scalability	Effective in obtaining good accuracies

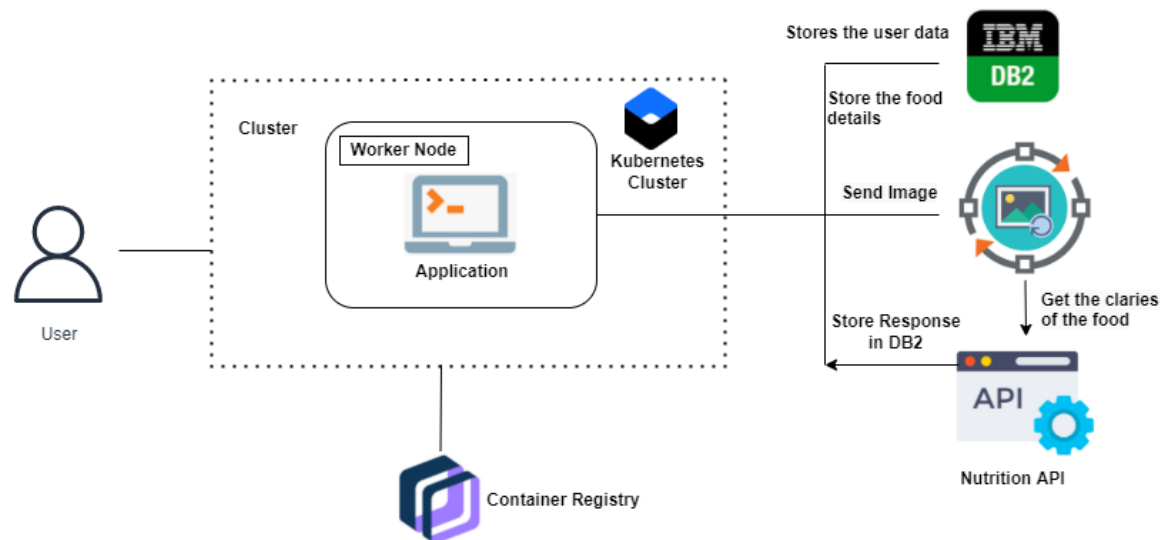
5 PROJECT DESIGN

5.1 Data Flow Diagrams

DATA FLOW DIAGRAM



5.2 Solution Technical Architecture



5.3 User Stories

User Stories

User Type	Functional Requirements (Epic)	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
User(All common people)	User Registration	USN-1	As a user ,I can register for the application by entering my Name, email, password.	I can access my dashboard.	High	Sprint-1
	Login	USN-2	As a user, I can login to the application using my given credentials.	I can access my dashboard.	High	Sprint-1
	BMI Calculation	USN-3	As a user, I enter my height and weight details.	I can get to know about my BMI.	High	Sprint-1
	Uploading the Image	USN-4	As a user, I will upload the image of food that I want to eat.	I can upload the image to decide whether to eat or not.	High	Sprint-1
	Providing output to user	USN-5	As a user, I will get to know the results of the inputs I've given.	I will get to know if I can eat the food or not.	Medium	Sprint-2
Administrator	Data Analysis	USN-6	As an admin, I will develop algorithms and modules to process the data.	I can store the result in database.	High	Sprint-1
	Integrating with Cloud	USN-7	As a admin, I integrate the results in cloud containers.	I can deploy the data in cloud.	High	Sprint-1

6 PROJECT PLANNING SCHEDULING

6.1 Sprint Planning & Estimation

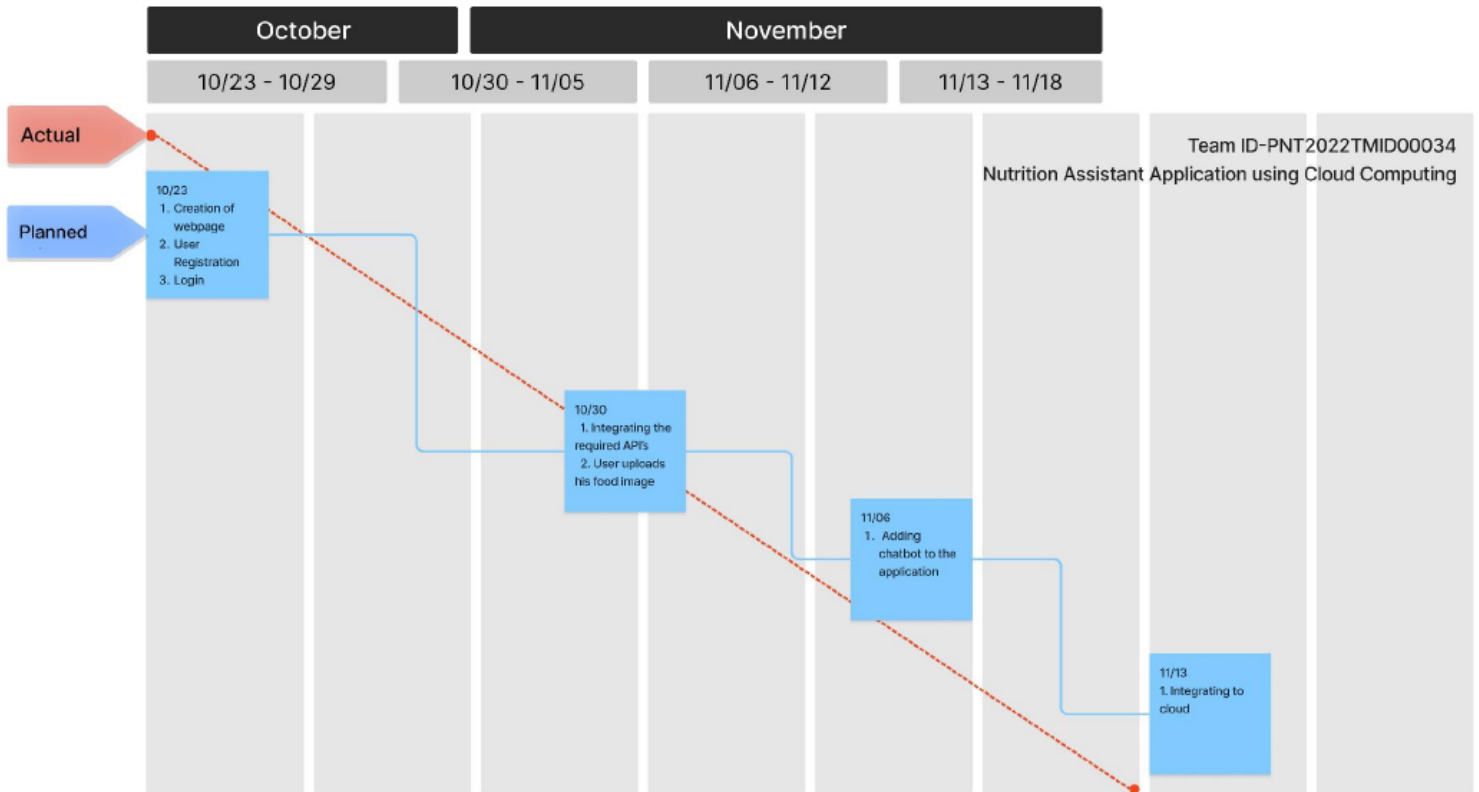
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website enter his height & weight and also uploads the food he wants to eat on the website.	20	High	ALLAN JACOB P ABISHEK R ACHUTHA MALLIKARJUNA JANESH C
Sprint-2	Admin Panel	USN-2	The admin uses the Clarifai API to identify the food and Nutrition API to find the amount of nutrition present in that food.	20	High	ALLAN JACOB P ABISHEK R ACHUTHA MALLIKARJUNA JANESH C
Sprint-3	Chat Bot	USN-3	The user can also directly talk to the webpage and ask questions using the chatbot	20	High	ALLAN JACOB P ABISHEK R ACHUTHA MALLIKARJUNA JANESH C
Sprint-4	Final Delivery	USN-4	Integrate the application to Cloud using Docker and Kubernetes. Submit the report of the final application.	20	High	ALLAN JACOB P ABISHEK R ACHUTHA MALLIKARJUNA JANESH C

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		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

Burndown Chart

October / November 2022



7 CODING

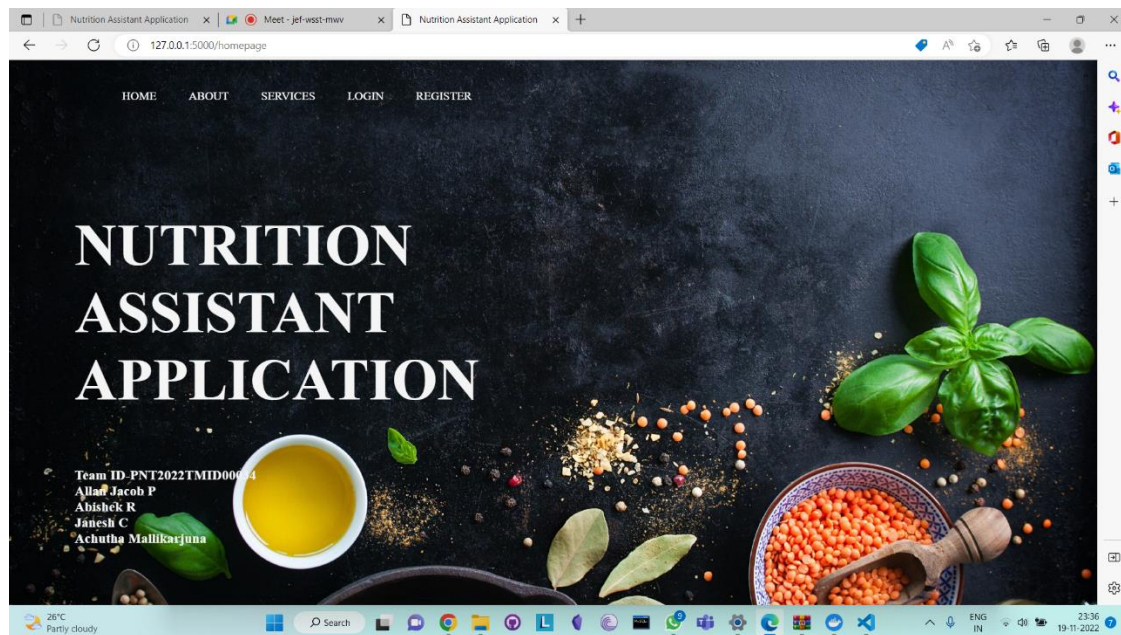
Login Feature

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/login". The page has a solid green background. In the center, there is a white login form titled "Login". The form contains two input fields: "Username" with the text "abishek" and "Password" with three dots. Below these fields is a green "Login" button. At the bottom of the form, there is a link that says "Don't have an account? [Sign Up here](#)". The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates the date and time as "19-11-2022 23:37".

Register

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/register". The page has a solid green background. In the center, there is a white registration form titled "Registration". The form contains four input fields: "Full Name" with the text "abishek", "Email" with the text "abishek@gmail.com", "Username" with the text "abishek", and "Password" with three dots. There is also a "Confirm Password" field with three dots and an eye icon to toggle visibility. Below these fields is a green "Register" button. At the bottom of the form, there is a link that says "Already have an account? [Login here](#)". The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates the date and time as "19-11-2022 23:37".

Home Page



Nutrition Calculation

127.0.0.1:5000/window

What are the nutrition value present in your food just type to know.

Food name :
burger

Upload picture:
Choose File burger.jpeg Submit

26°C Partly cloudy 23:37 19-11-2022

127.0.0.1:5000/window

```
{
  "items": [
    {
      "sugar_g": 0.0,
      "fiber_g": 0.0,
      "serving_size_g": 100.0,
      "sodium_mg": 356,
      "name": "burger",
      "potassium_mg": 137,
      "fat_saturated_g": 4.7,
      "fat_total_g": 11.5,
      "calories": 237.7,
      "cholesterol_mg": 53,
      "protein_g": 15.2,
      "carbohydrates_total_g": 18.1
    }
  ]
}
```

26°C Partly cloudy 23:37 19-11-2022

8 TESTING

8.1 Test Cases

8.1.1 Functional Testing

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

8.1.2 White Box Testing

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done using the percentage value of load and energy. The tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

8.1.3 Black Box Testing

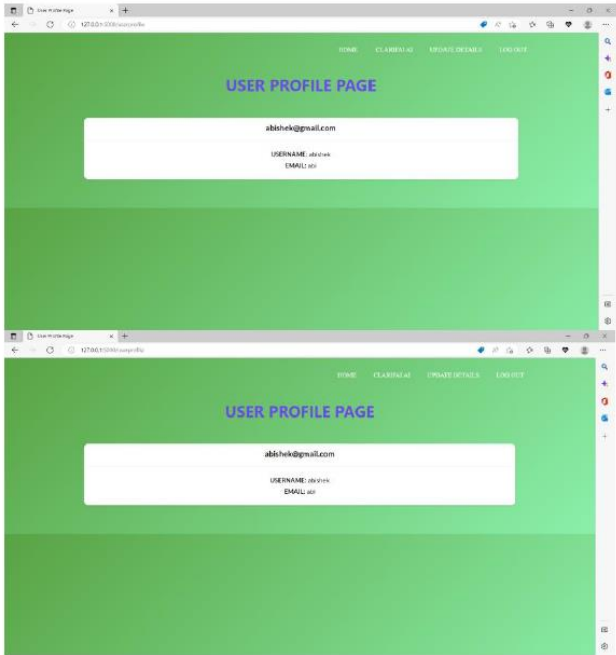
Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

8.2 User Acceptance Testing

Acceptance testing can be defined in many ways, but a simple definition is that it succeeds when the software functions in a manner that can be reasonably expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to find whether the inputs are accepted by the database or other validations. For example, accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.3 Testing Results

Performance Testing

S. No	Parameter	Screenshot
1.	User Login	

9 ADVANTAGES & DISADVANTAGES

Advantages

- User friendly
- Get immediate results
- Anyone can use
- simple user interface
- Saves time

Disadvantages

- Cannot upload photos of large size
- Network Traffic Problems

10 CONCLUSION

In this project we finally developed Nutritional Assistant Application, with the help of nutrition API and cloud technologies. Our goal is to help people to live a healthy and balanced life. We hope that people maintain their calories and live a heal

13 CODING

Source Code

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import ibm_db
import re
import sendgrid
from sendgrid.helpers.mail import Mail, Email, To, Content
import requests
from random import *
from clarifai_grpc.grpc.api import service_pb2, resources_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
from flask_mail import Mail, Message
import os
from flask_mail import Mail, Message
app = Flask(__name__)

mail = Mail(app) # instantiate the mail class
# configuration of mail
app.config['MAIL_SERVER']='smtp.sendgrid.net'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =
'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'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
ctp = randint(000000,999999)

app.secret_key = 'a'

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;SSLServerCerti
ficate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=bjg03480;PWD=Nw0nFgeEM0S3MdSN;",'', '')

picsfolder = os.path.join('static','pics')
app.config['UPLOAD_FOLDER']=picsfolder

@app.route('/')

```

```

@app.route('/homepage')
def homepage():
    icon = os.path.join(app.config['UPLOAD_FOLDER'],'icon.gif')
    return render_template('homepage.html',user_image=icon)

@app.route('/about')
def about():
    icon = os.path.join(app.config['UPLOAD_FOLDER'],'icon.gif')
    return render_template('about.html',user_image=icon)

@app.route('/login', methods =['GET', 'POST'])
def login():
    msg=''
    if request.method=='POST' and 'username' in request.form and 'passwords' in request.form:
        username = request.form['username']
        passwords = request.form['passwords']
        stmt = ibm_db.prepare(conn,'SELECT * FROM appuser WHERE username = ? AND passwords = ?')
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,passwords)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        if account:
            session['loggedin'] = True
            session['username'] = account['USERNAME']
            msg='Login successful'
            return redirect(url_for('userprofile'))
        else:
            msg='Incorrect username/password'
    return render_template('login.html',msg=msg)

@app.route('/logout')
def logout():
    if 'id' in session:
        session.pop('id',None)
        session.pop('username',None)
        session.pop('passwords',None)
    return redirect(url_for('homepage'))

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        username = request.form['username']

```

```

fullname = request.form['fullname']
email = request.form['email']
passwords = request.form['passwords']
cpassword = request.form['cpassword']
stmt = ibm_db.prepare(conn, 'SELECT * FROM appuser WHERE username = ?')
ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
if account:
    msg = 'Account already exists !'
elif not re.match(r'^@+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'Username must contain only characters and numbers !'
elif not username or not passwords or not email:
    msg = 'Please fill out the form !'
else:
    prep_stmt = ibm_db.prepare(conn, "INSERT INTO appuser(username, fullname, email,
passwords, cpassword) VALUES(?, ?, ?, ?, ?)")
    ibm_db.bind_param(prepare_stmt, 1, username)
    ibm_db.bind_param(prepare_stmt, 2, fullname)
    ibm_db.bind_param(prepare_stmt, 3, email)
    ibm_db.bind_param(prepare_stmt, 4, passwords)
    ibm_db.bind_param(prepare_stmt, 5, cpassword)
    ibm_db.execute(prepare_stmt)
    msg = 'You have successfully registered !'
    return render_template('email.html')
elif request.method == 'POST':
    msg = 'Please fill out the form !'
return render_template('registration.html', msg = msg)

@app.route('/userprofile', methods = ['GET', 'POST'])
def userprofile():
    if 'username' in session:
        username = session['username']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM appuser WHERE username = ?')
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_tuple(stmt)
        return render_template('userprofile.html', username = acc[1], fullname = acc[2], email =
acc[3],)
    return render_template('userprofile.html')

@app.route('/updateprofile', methods = ['GET', 'POST'])
def updateprofile():
    msg = ''
    if request.method == 'POST':

```



```

        username=request.form["username"]
        height = request.form['height']
        weight = request.form['weight']
        gender = request.form['gender']
        blood = request.form['blood']
        prep_stmt = ibm_db.prepare(conn,"INSERT INTO userdetail(username, height, weight,
gender, blood) VALUES(?, ?, ?, ?, ?)")
        ibm_db.bind_param(prepare_stmt, 1, username)
        ibm_db.bind_param(prepare_stmt, 2, height)
        ibm_db.bind_param(prepare_stmt, 3, weight)
        ibm_db.bind_param(prepare_stmt, 4, gender)
        ibm_db.bind_param(prepare_stmt, 5, blood)
        ibm_db.execute(prepare_stmt)
        return redirect(url_for('detail'))
    return render_template('updateprofile.html')

@app.route('/detail', methods=['GET', 'POST'])
def detail():
    if 'username' in session:
        username = session['username']
        stmt = ibm_db.prepare(conn, 'SELECT * FROM userdetail WHERE username = ?')
        ibm_db.bind_param(stmt, 1,username)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_tuple(stmt)
        return render_template('detail.html',username = acc[0], height = acc[1], weight = acc[2],
gender = acc[3], blood = acc[4])
    return render_template('detail.html')

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

    # Calorie Ninja
    url = "https://calorieninjas.p.rapidapi.com/v1/nutrition"

    headers = {
        "X-RapidAPI-Key": "8112050c4emsh723a2f0f2077ec0p1be092jsn91abf0528072",
        "X-RapidAPI-Host": "calorieninjas.p.rapidapi.com"
    }

    if request.method == 'POST':
        foodname = request.form['foodname']

        querystring = {"query": foodname}
        response = requests.request(

```

```

        "GET", url, headers=headers, params=querystring)

    return response.text

return render_template('window.html')

@app.route('/window', methods=['POST', 'GET'])
def clarifai():
    if request.files.get('image'):
        image = request.files['image'].stream.read()
        stub = service_pb2_grpc.V2Stub(ClarifaiChannel.get_grpc_channel())

        CLARIFAI_API_KEY = "a4d1d4d3de3a4b6da5777636c754921e"
        APPLICATION_ID = "my-first-application"

        metadata = (("authorization", f"Key {CLARIFAI_API_KEY}"),)

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

        request = service_pb2.PostModelOutputsRequest(
            model_id='1d5fd481e0cf4826aa72ec3ff049e044',
            inputs=[
                resources_pb2.Input(
                    data=resources_pb2.Data(
                        image=resources_pb2.Image(
                            base64=file_bytes
                        )
                    )
                )
            ])
        response = stub.PostModelOutputs(request, 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))

    return render_template('window.html')

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

```

```

if request.method == 'POST':
    email1 = request.form['email1']
    sql = "SELECT * FROM email WHERE email1 = ? "
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,email1)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_tuple(stmt)
    print(account)
    if account:
        msg = 'Account already exists !'
    else:
        insert_sql = "INSERT INTO email(email1) VALUES(?)"
        stmt = ibm_db.prepare(conn,insert_sql)
        ibm_db.bind_param(stmt, 1, email1)
        ibm_db.execute(stmt)
        msg = Message('NUTRITION ASSISTANT',sender ='abishek12082001@gmail.com',recipients =
[email1])
        msg.body = 'Hello user,THIS IS YOUR ONE TIME PASSWORD'
        msg.body = str(otp)
        mail.send(msg)
        return render_template('verify.html')
    return render_template('email.html')

@app.route('/validate',methods=['GET', 'POST'])
def validate():
    user_otp = request.form['otp']
    if otp == int(user_otp):
        return render_template('login.html')
    return render_template('verify.html')

@app.route('/services')
def services():
    icon = os.path.join(app.config['UPLOAD_FOLDER'],'icon.gif')
    return render_template('services.html',user_image=icon)

if __name__ == '__main__':
    app.debug = True
    app.run(host='0.0.0.0',port=8080)

```