NUTRITION ASSISTANT APPLICATION

PROJECT REPORT

DATE:

19/11/2022

TEAM MEMBERS:

VARSHINI S

SWETHA P

SATHYAPRIYA S

SNEHAYAAZHINI S

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
6.3 Reports from JIRA

7. CODING & SOLUTIONING

7.1 Feature 1
7.2 Feature 2
7.3 Database Schema(if Applicable)

8. TESTING

8.1 Test Cases
8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrices

1.INTRODUCTION

1.1 Project Overview

A balanced diet is necessary for both proper nutrition and health. It protects you against many chronic noncommunicable diseases, such as cancer, diabetes, and heart disease. A balanced diet that limits salt, sugar, saturated fats, and trans fats from industrial production is necessary for good health.

Obesity rates are rising alarmingly quickly as a result of people's lack of knowledge about appropriate eating practice, which reflects the hazards to their health. It's still not very convenient for people to use app-based nutrient dashboard systems, even though food packaging includes nutrition (and calorie) labels. These systems can analyze real-time images of a meal and analyze it for nutrition content, which can be very handy and improve dietary habits and subsequently help with maintaining a healthy lifestyle.

1.2 Purpose

By identifying the supplied food image, this project attempts to create a web application that automatically calculates food qualities like components and nutritional value. Our approach uses Food APIs to provide the nutritional information of the recognized food and Clarifa's AI-Driven Food Detection Model for precise food recognition.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

- Obesity increases the risk of several debilitating and deadly diseases including diabetes,heart disease,and some cancers.
- One of the most basic functions of this project is to guide its users towards a healthy diet and assist them to achieve their health goals.

2.2 References

RESEARCH PAPER BASED ON OUR PROJECT :

- https://www.researchgate.net/publication/322152435_Enhancing_Cloud_and_Big_Data_Systems_for_healthy_Food_and_Information_Systems_Practice_A_Conceptual_Study
- **https://www.researchgate.net/publication/346411010_DEVELOPMENT_OF_A_CLOUD_BASED_SOLUTION_FOR_EFFECTIVE_NUTRITION_INTERVENTION_IN_THE_MANAGEMENT_OF_LIFESTYLE_DISEASES**

2.3 Problem Statement Definition

Problem Statement (PS)	I am (Customer)	I am trying to	But	Because	Which makes me feel
PS-1	Fitness freak	Finding a perfect pre workout plan for maintaining fitness	I cannot choose a correct plan	It is Confusing	A perfect daily pre workout plan suggestion
PS-2	Student	Find a balanced nutrition diet to lose weight	There is no balanced diet available without workout	I have no time to do workout	A best nutritional based diet plan with less workout
PS-3	Body Builder	Choose a best plan for whole body workout.	It is hard to select a best workout plan	A wrong workout plan will lead to a change in the shape of my body	Perfect diet and workout plan for bodybuilding

3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Empathy map

SAY

- Cannot Identify ingredients.
- Too much unrecognizable ingredients name.
- Calorific value or nutrients of the ingredients not known

DOES

- Every time surfs the internet for the ingredients and nutrition details.
- Asks to people who may know.

USER

FEEL

- Gets confused.
- Worried about calorie intake.
- Lose motivation to eat healthy and continues unhealthy eating habits.

THINK

- Cognitive load increases.
- Irritated.
- Confused.
- Not knowing what they want to need.

PAINS

- Extensive searching to know about each ingredient
- Not knowing about the calorific value of the food they eat

GAINS

- Easy one step destination to know abbot all ingredients, nutrition and calorific values
- Cost and time saving

File | C:/Users/SWETHA%20.P/Downloads/Proposed_Solution%20(3)%20(1).pdf

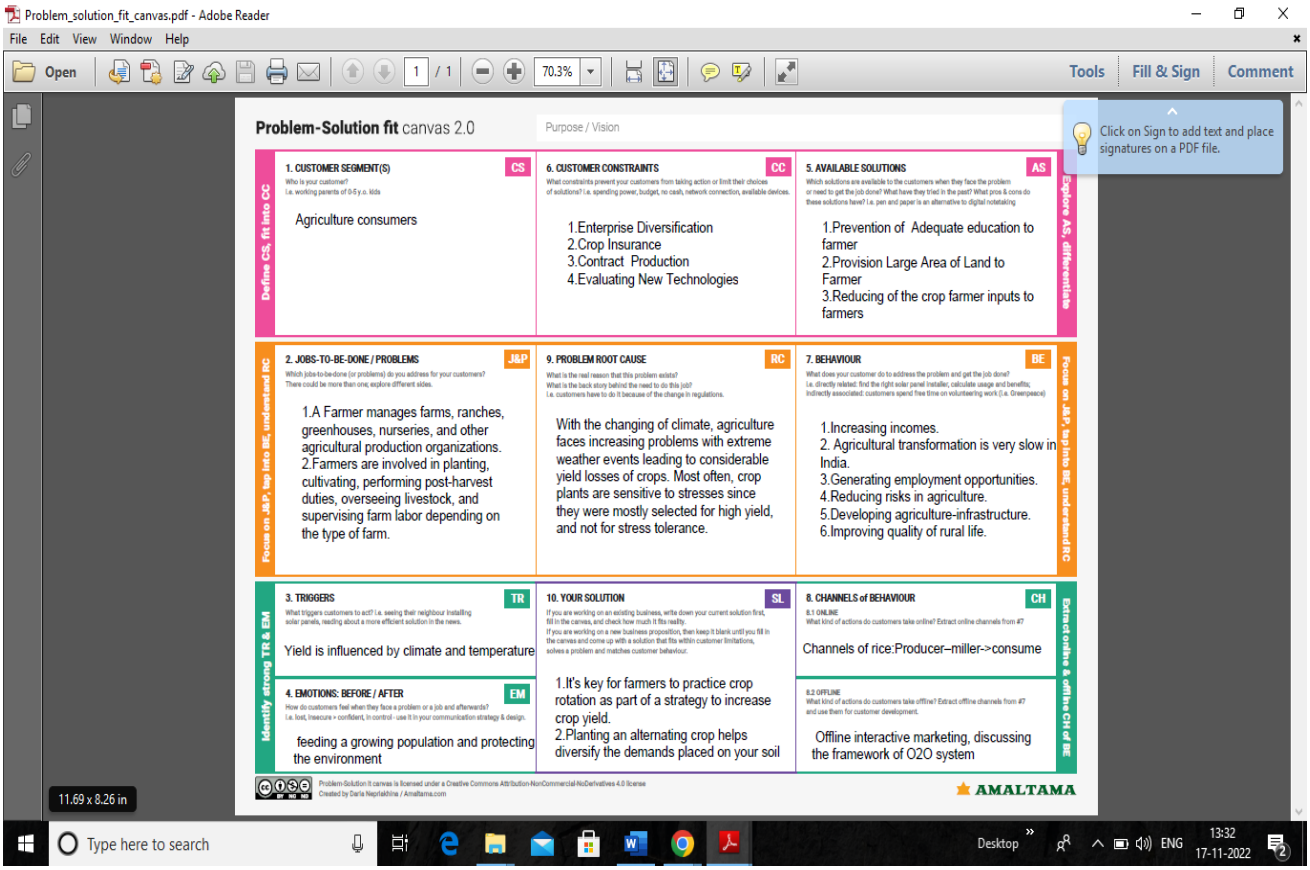
Gmail YouTube Maps New Tab Learning Outcomes... WhatsApp Classes Top Engineering Co... boAt Warranty My Bag » Other bookmarks

Proposed_Solution (3) (1).pdf 1 / 1 71%

Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• The obesity rate keeps rising because people don't know how to eat healthily which causes a number of health problems • This system will analyze the food and evaluate its nutritional content in order to prevent this problem
2.	Idea / Solution description	• By creating a web application that classifies input food images to estimate food attributes like nutrition value automatically
3.	Novelty / Uniqueness	• Clustering the people based on their BMI value
4.	Social Impact / Customer Satisfaction	• This app helps users keep track of their daily caloric intake and prevent obesity by choosing healthier foods.
5.	Business Model (Revenue Model)	• This app would benefit everyone from a health and financial standpoint.
6.	Scalability of the Solution	• Many people these days are concerned about their health. Consequently, this app has an impact on the next generation.

3.4 Problem Solution



4.REQUIREMENT ANALYSIS

4.1 Functional requirement

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 Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User profile completion	Collecting users information like height , weight etc
FR-4	Meal image	Uploading live image of food
FR-5	Display calorie information	Integrate the Clarifai API to obtain the food's name. Utilize Nutrition API (rapid API) integration to gather calorie data.

4.2 Non-Functional Requirement

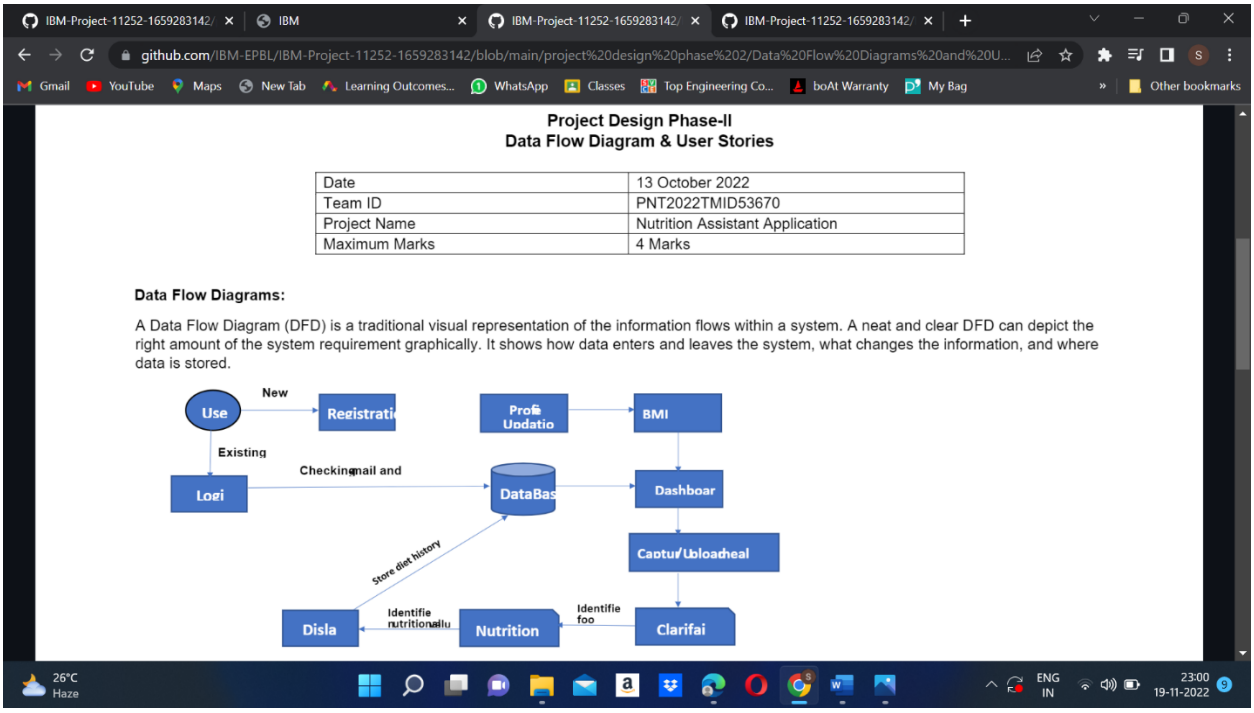
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effortless and clear design User friendly
NFR-2	**Security**	Integrated authentication and authorisation protocols for each system
NFR-3	**Reliability**	95 percent of use cases must result in perfect performance from the system
NFR-4	**Performance**	The landing page supporting multiple users must have a response time of no more than five second
NFR-5	**Availability**	Services must always remain uninterrupted, with the exception of when servers are being updated
NFR-6	**Scalability**	For high workloads, provide either horizontal or vertical scaling.

5. Project Design

5.1 Data Flow Diagrams



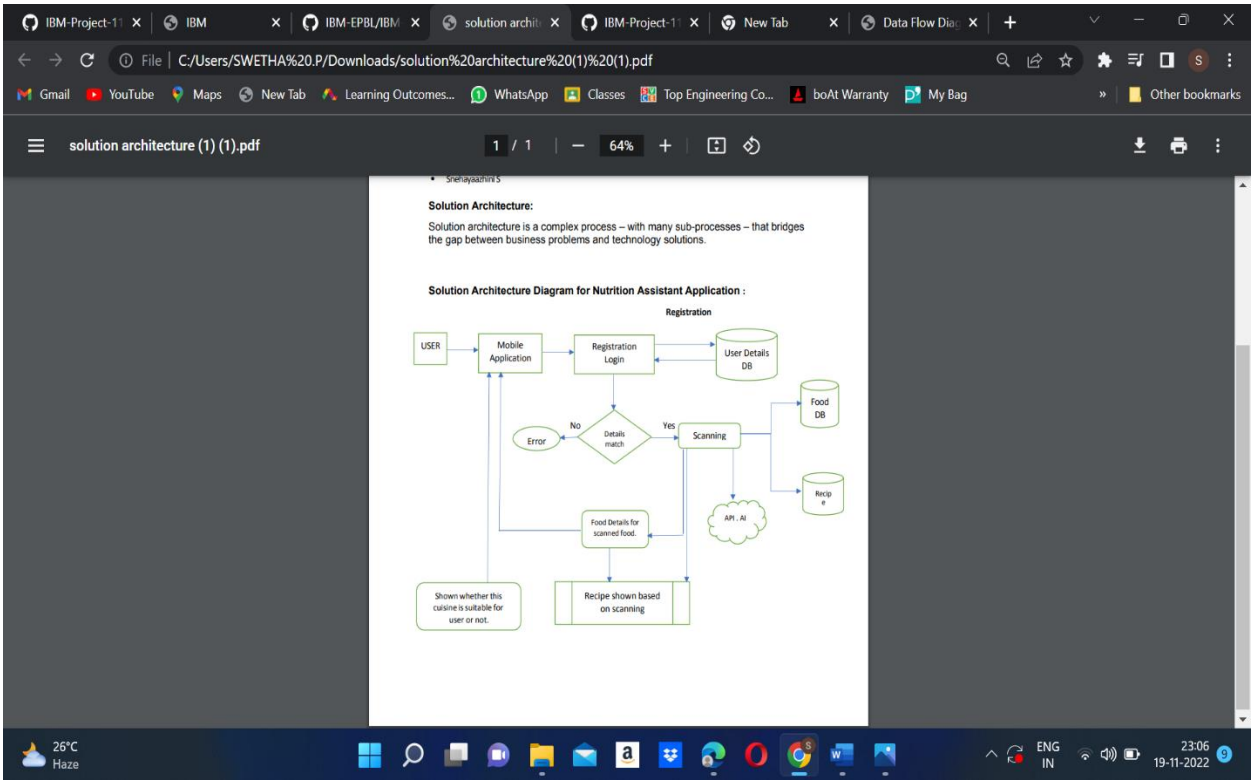
Project Design Phase-II
Data Flow Diagram & User Stories

Date	13 October 2022
Team ID	PNT2022TMID53670
Project Name	Nutrition Assistant Application
Maximum Marks	4 Marks

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.

5.2 Solution & Technical Architecture

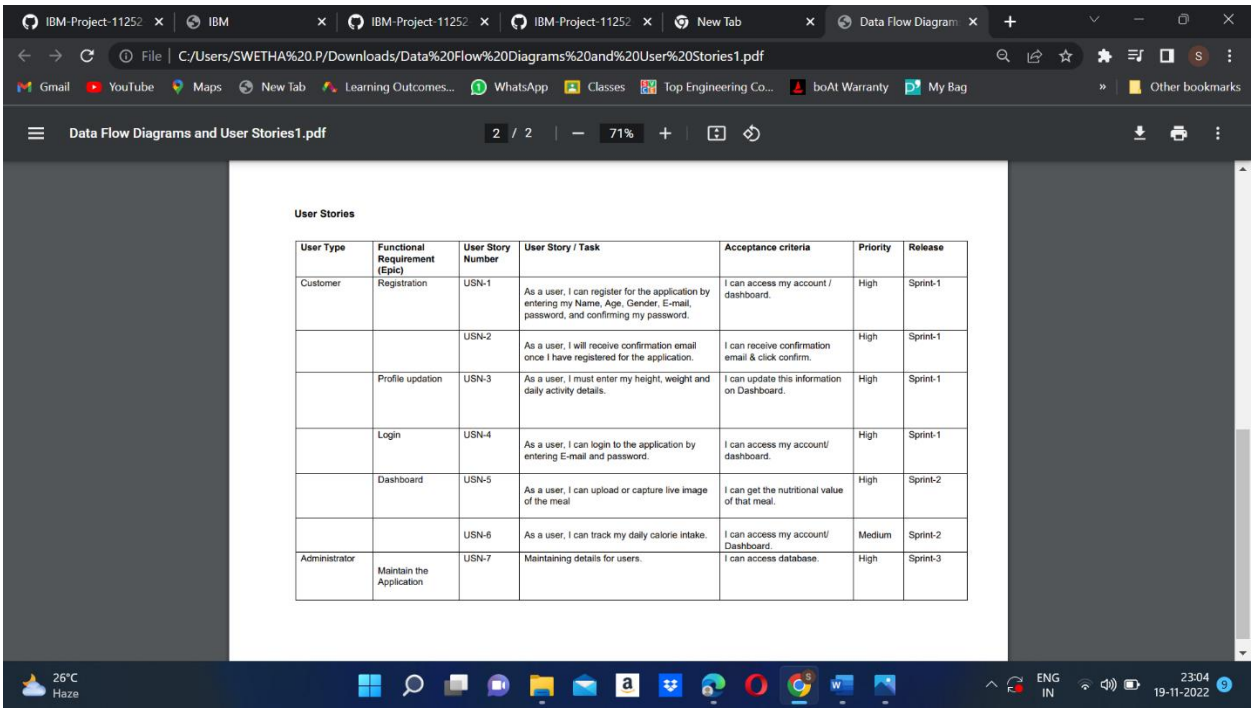


Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Solution Architecture Diagram for Nutrition Assistant Application :

5.3 User Stories



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my Name, Age, Gender, E-mail, 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
	Profile updation	USN-3	As a user, I must enter my height, weight and daily activity details.	I can update this information on Dashboard.	High	Sprint-1
	Login	USN-4	As a user, I can login to the application by entering E-mail and password.	I can access my account/ dashboard.	High	Sprint-1
	Dashboard	USN-5	As a user, I can upload or capture live image of the meal	I can get the nutritional value of that meal.	High	Sprint-2
		USN-6	As a user, I can track my daily calorie intake.	I can access my account/ Dashboard.	Medium	Sprint-2
Administrator	Maintain the Application	USN-7	Maintaining details for users.	I can access database.	High	Sprint-3

6.PROJECT PLANNING AND SCHEDULING

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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

Velocity:

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

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

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 and go through the products available on the website.	20	High	Varshini S Swetha P Sathya Priya S Snehayaazhini S
Sprint-2	Admin Panel	USN-2	The role of the admin is to check out the database about the stock and have a truck of all the things that the users are purchasing.	20	High	Varshini S Swetha P Sathya Priya S Snehayaazhini S
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided by the user	20	High	Varshini S Swetha P Sathya Priya S Snehayaazhini S
Sprint-4	Final Delivery	USN-4	Container of applications using docker Kubernetes and development the application. Create the documentation and final submit the application	20	High	Varshini S Swetha P Sathya Priya S Snehayaazhini S

github.com/IBM-EPBL/IBM-Project-11252-1659283142/blob/main/project%20planning%20phase/Sprint%20Delivery%20Plan1.pdf

Burndown Chart

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

PROJECT PLANNI....pdf
PROJECT PLANNI....pdf
Show all
26°C Partly cloudy
ENG IN
23:26 19-11-2022

7. CODING & SOLUTIONING

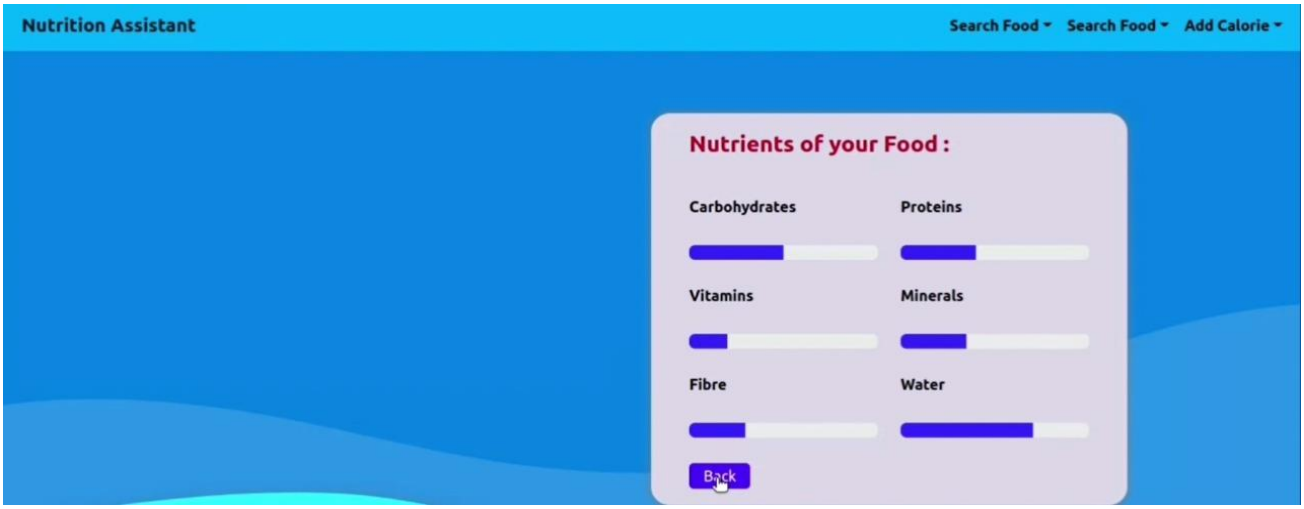
```
# Google Auth Configuration
os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"

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"
)

# 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

# Creates user table if not exists
create_table = "CREATE TABLE IF NOT EXISTS user(email varchar(30), username varchar(30), password varchar(30))"
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)
```

```
88          print(response.body)
89          print(response.headers)
90      except Exception as e:
91          print(e)
92
93  # Sign up page
94  @app.route(SIGN_UP_PAGE_URL, methods=['GET', 'POST'])
95  def signup():
96      msg = ''
97
98      if session.get('user'):
99          return redirect(HOME_PAGE_URL)
100
101     if request.method == 'POST':
102         user = request.form['user']
103         email = request.form['email']
104         password = request.form['password']
105
106         duplicate_check = "SELECT * FROM user WHERE username=?"
107         account = execute_sql(statement=duplicate_check, user=user)
108
109         if account:
110             msg = "There is already an account with this username!"
111         else:
112             insert_query = "INSERT INTO user values(?, ?, ?)"
113             execute_sql(statement=insert_query, email=email, user=user, password=password)
114
115             send_confirmation_mail(user, email)
116             return redirect(LOG_IN_PAGE_URL)
117     return render_template('signup.html', msg=msg)
118
119 # Login page
120 @app.route(LOG_IN_PAGE_URL, methods=['GET', 'POST'])
121 def login():
122     msg = ''
123
124     if session.get('user'):
125         return redirect(HOME_PAGE_URL)
126
127     if request.method == "POST":
128
129         user = request.form['user']
130         password = request.form['password']
131
132         duplicate_check = "SELECT * FROM user WHERE username=?"
133         account = execute_sql(statement=duplicate_check, user=user)
134
135         print(account)
136         if account and account['PASSWORD'] == password:
137             session['user'] = user
138             return redirect(HOME_PAGE_URL)
```

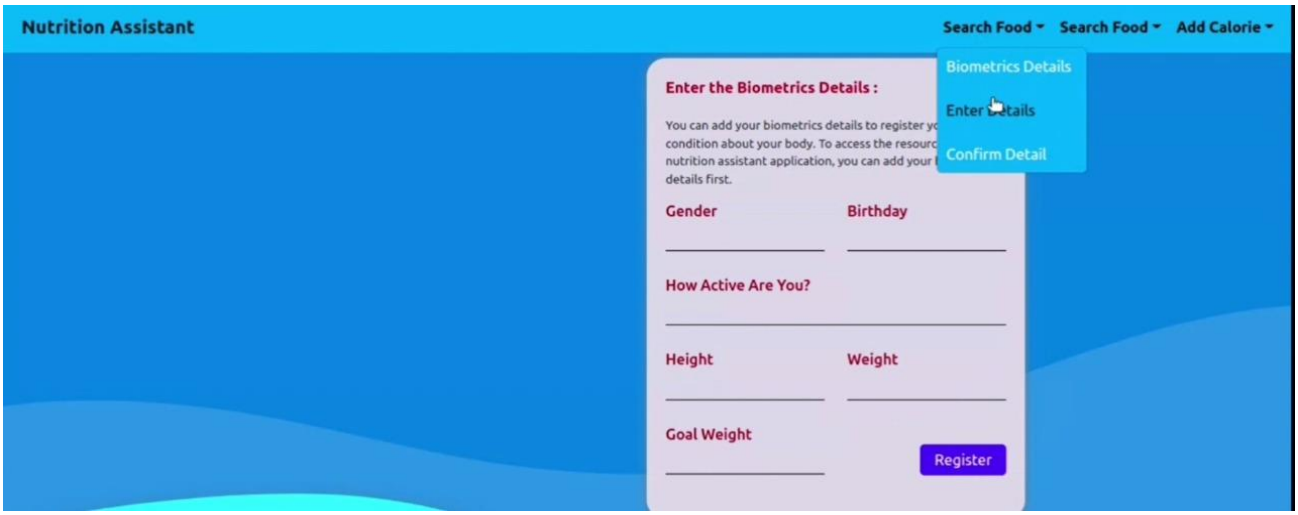
```python
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    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 user WHERE username=?"
    execute_sql(statement=delete_query, user=user)

    session.clear()
    return redirect(SIGN_UP_PAGE_URL)

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

8. TESTING

8.1 Test Cases

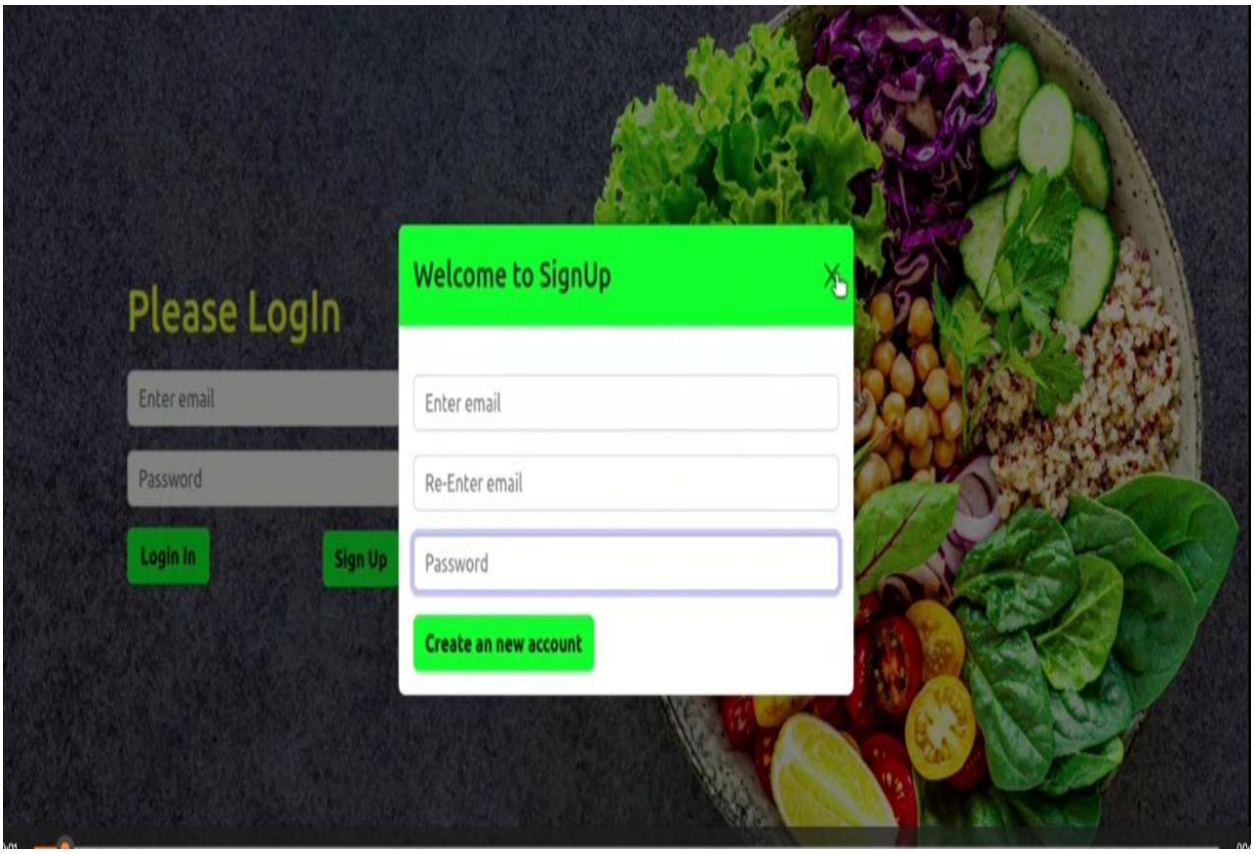


8.2 User Acceptance Testing



9. RESULTS

9.1Performance Metrics



10. ADVANTAGES & DISADVANTAGES

Advantages:

- It serves as an electronic medical and dietetic record, and personalized nutrition consultation approach can be client can converse to his/ her personal dietitian at their own convenient setting.

- It helps to make life easier for individuals who need to track their food intake for health reasons.

- These nutritionapplication can also help people find restaurants suitable for their dietary needs

- It allows you to analyze your own food choices to assess and tweak your eating plan and patterns.

- All popular food trackers allow you to assess much more than calories. They show protein, carbs, fat, sugar, saturated fat, fiber and more. Many programs also display micronutrients like potassium and iron. This allows you to see how balanced your meals are each day and over the course of a week. It also allows you to target certain nutrients for health conditions you are trying to address. For example, you can keep track of fiber if you are using lifestyle factors to address cholesterol. A pregnant woman who is anemic might want to keep track of protein and iron.

Disadvantages:

- We can become hyper-focused on numbers (calories, carbs, fiber, sugar etc) over eating a wide variety of healthy, whole foods.

- Even if chicken, brown rice and broccoli is a nutrient dense, ‘balanced’ meal, eating it every day is boring and causes us to miss out on a variety of other delicious foods and beneficial nutrients

- It’s not sustainable long term

- It can actually remove a level of mindfulness because the goal is to hit target numbers NOT listen to your body.

11. Conclusion

The cloudbased system would have the ability to calculate the nutritional requirements and to guide first line nutritional management to patients and clients automatically .

Thus this application helps you understand your eating habits and patterns, and help you identify the foods — good and not-so-good — you eat on a regular basis. Research shows that for people interested in losing weight, keeping a journal can be a very effective tool to help change behavior.

12. FUTURE SCOPE

The younger generation is increasingly dependent on junk food and has no knowledge what ingredients are added to make it taste good. These foods are unhealthy. Therefore, as a result of utilising this application, people will be more careful about what they consume going forward in order for them to live a balanced, healthy existence.

13 APPENDIX

SOURCE CODE

```
# 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"
)

# 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

# Creates user table if not exists
create_table = "CREATE TABLE IF NOT EXISTS user(email varchar(30), username varchar(30), password varchar(30))"
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)
```

```
88          print(response.body)
89          print(response.headers)
90      except Exception as e:
91          print(e)
92
93  # Sign up page
94  @app.route(SIGN_UP_PAGE_URL, methods=['GET', 'POST'])
95  def signup():
96      msg = ''
97
98      if session.get('user'):
99          return redirect(HOME_PAGE_URL)
100
101     if request.method == 'POST':
102         user = request.form['user']
103         email = request.form['email']
104         password = request.form['password']
105
106         duplicate_check = "SELECT * FROM user WHERE username=?"
107         account = execute_sql(statement=duplicate_check, user=user)
108
109         if account:
110             msg = "There is already an account with this username!"
111         else:
112             insert_query = "INSERT INTO user values(?, ?, ?)"
113             execute_sql(statement=insert_query, email=email, user=user, password=password)
114
115             send_confirmation_mail(user, email)
116             return redirect(LOG_IN_PAGE_URL)
117     return render_template('signup.html', msg=msg)
118
119 # Login page
120 @app.route(LOG_IN_PAGE_URL, methods=['GET', 'POST'])
121 def login():
122     msg = ''
123
124     if session.get('user'):
125         return redirect(HOME_PAGE_URL)
126
127     if request.method == "POST":
128
129         user = request.form['user']
130         password = request.form['password']
131
132         duplicate_check = "SELECT * FROM user WHERE username=?"
133         account = execute_sql(statement=duplicate_check, user=user)
134
135         print(account)
136         if account and account['PASSWORD'] == password:
137             session['user'] = user
138             return redirect(HOME_PAGE_URL)
```

```python
    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 user WHERE username=?"
    execute_sql(statement=delete_query, user=user)

    session.clear()
    return redirect(SIGN_UP_PAGE_URL)

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

GITHUB & PROJECT DEMO LINK

IBM-PROJECT-11252-1659283142

https://drive.google.com/file/d/149mu8vm0tScAkjUcsYSe597AynJVhJJg/view?usp=sharing