AI-powered Nutrition Analyzer for Fitness Enthusiasts

APROJECTREPORT

Submittedby

KAVIYA K **[710119205006]**

AGNES METILDA R **[710119205001]**

AKASH M **[710119205003]**

MAHENDIRAN M **[710119205007]**

TEAMID - PNT2022TMID42325

BACHELOR OF TECHNOLOGY

IN

INFORMATIONTECHNOLOGY

ANNAUNIVERSITY-600025

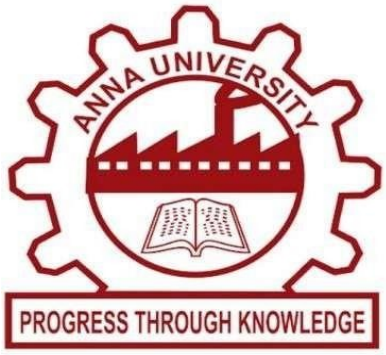


Table of Content

INTRODUCTION

1.1 Project Overview

In today's culture, many people suffer from a range of ailments and illnesses. It's not always simple to recommend a diet right away. The majority of individuals are frantically trying to reduce weight, gain weight, or keep their health in check. Time has also become a potential stumbling block. The study relies on a database that has the exact amounts of a variety of nutrients. As a result of the circumstance, a program that would encourage individuals to eat healthier has been created. Only three sorts of goods are recommended: weight loss, weight gain, and staying healthy. The Diet Recommendation System leverages user inputs such as medical data and the option of vegetarian or non-vegetarian meals from the two categories above to predict food items. We'll discuss about food classification, parameters, and machine learning in this post. This study also includes a comparative review of the advantages and disadvantages of machine learning methods. Finally, we'll discuss future research directions for the diet guidance system.

1.2 Purpose

Nowadays, a human being is suffering from various health problems such as fitness problem, inappropriate diet, mental problems etc. Various studies depict that inappropriate and inadequate intake of diet is the major reasons of various health issues and diseases. A study by WHO reports that inadequate and imbalanced intake of food causes around 9% of heart attack deaths, about 11% of ischemic heart disease deaths, and 14% of gastrointestinal cancer deaths worldwide. Moreover, around 0.25 billion children are suffering from Vitamin-A deficiency, 0.2 billion people are suffering from iron deficiency (anaemia), and 0.7 billion people are suffering from iodine deficiency. The main objective of this work to recommend a diet to different individual.

LITERATURE SURVEY

2.1 Existing Problem

The World Health Organization identifies the overall increasing of non-communicable diseases as a major issue, such as premature heart diseases, diabetes, and cancer. Unhealthy diets have been identified as the important causing factor of such diseases. In this context, personalized nutrition emerges as a new research field for providing tailored food intake advices to individuals according to their physical, physiological data, and further personal information. Specifically, in the last few years, several types of research have proposed computational models for personalized food recommendation using nutritional knowledge and user data.

2.2 References

Title: A Food Recommender System Considering Nutritional Information and User Preferences

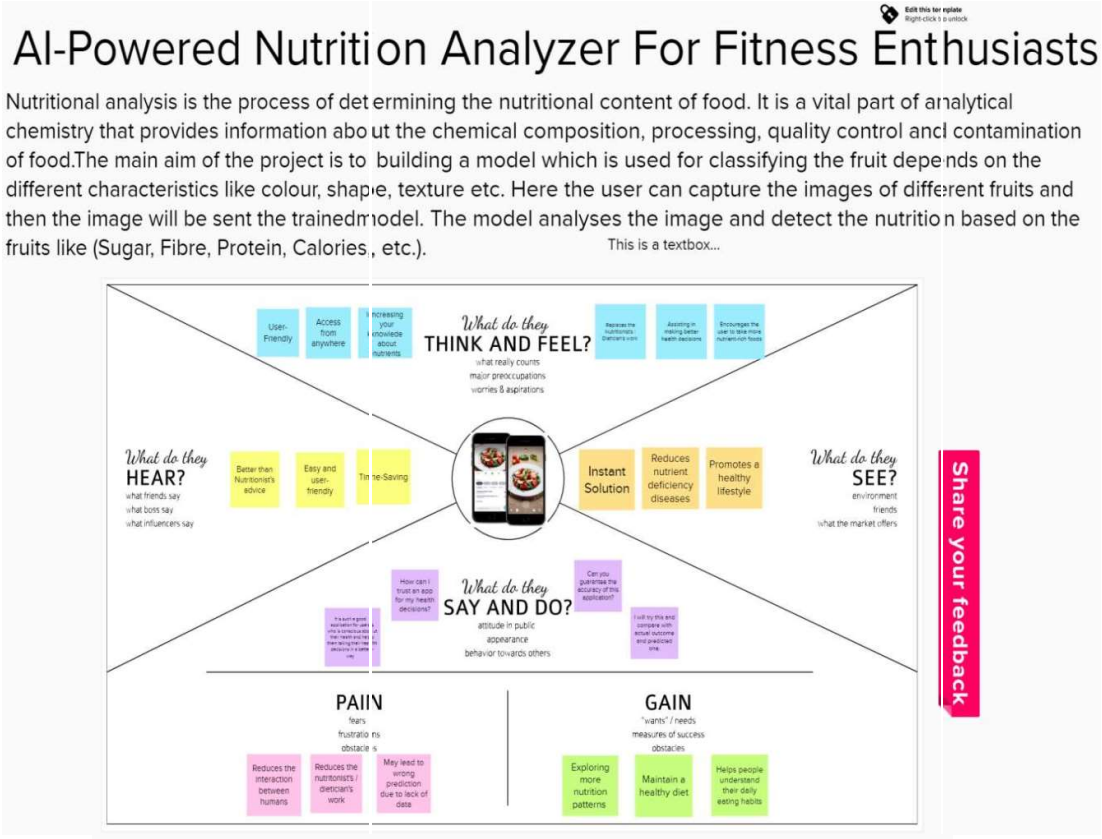
Author: RACIEL YERA TOLEDO1 AHMAD A. ALZAHRANI, AND LUIS MARTíNEZ

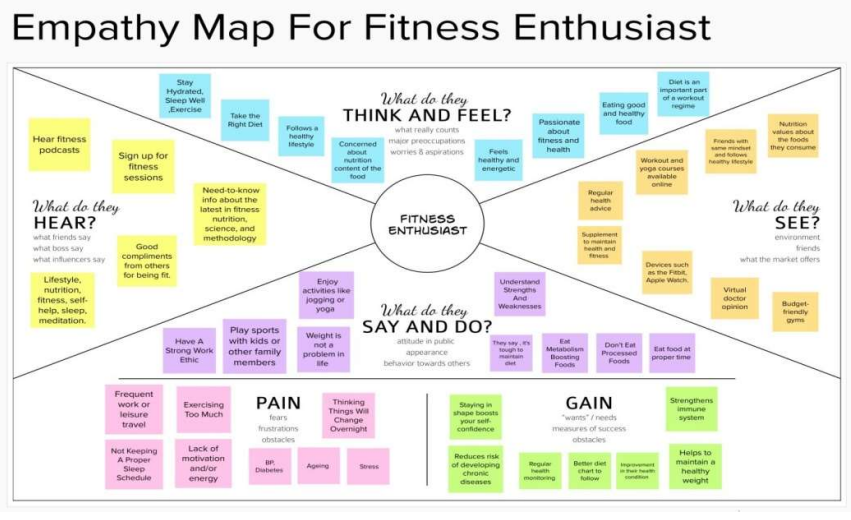
2.3 Problem Statement Definition

This paper presents a general framework for daily meal plan recommendations, incorporating as main feature the simultaneous management of nutritional-aware and preference-aware information, in contrast to the previous works which lack this global viewpoint. The proposal incorporates a pre-filtering stage that uses AHPSort as multi-criteria decision analysis tool for filtering out foods which are not appropriate to the current user characteristics. Furthermore, it incorporates an optimization-based stage for generating a daily meal plan whose goal is the recommendation of food highly preferred by the user, not consumed recently, and satisfying his/her daily nutritional requirements. A case study is developed for testing the performance of the recommender system.This project has been developed using Machine Learning algorithms.cluster the food according to calories and then Random Forest Classifier is used to classify the fooditems and predict the food items based on input given. Numpy was used to convert features into numpy and then perform the further operation. Tkinter was used to create interface. K-Means was used to perform clustering. Train_test_split was used to divide the dataset into train and test portions to train and test themodel. Random Forest Classifier used to predict the food items base

IDEATION & PROPOSED SOLUTION

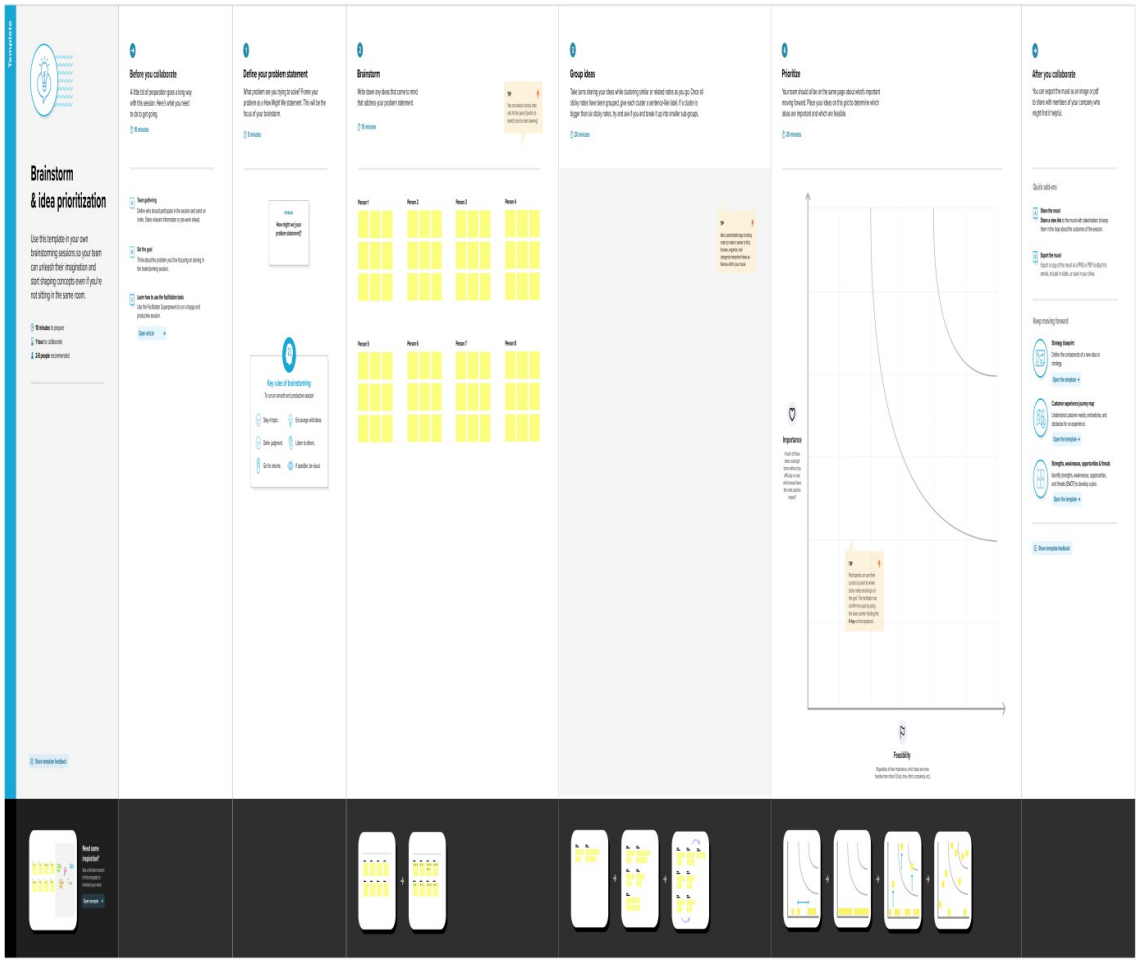
3.1 Empathy Map Canvas





3.2 Ideation& Brainstorming

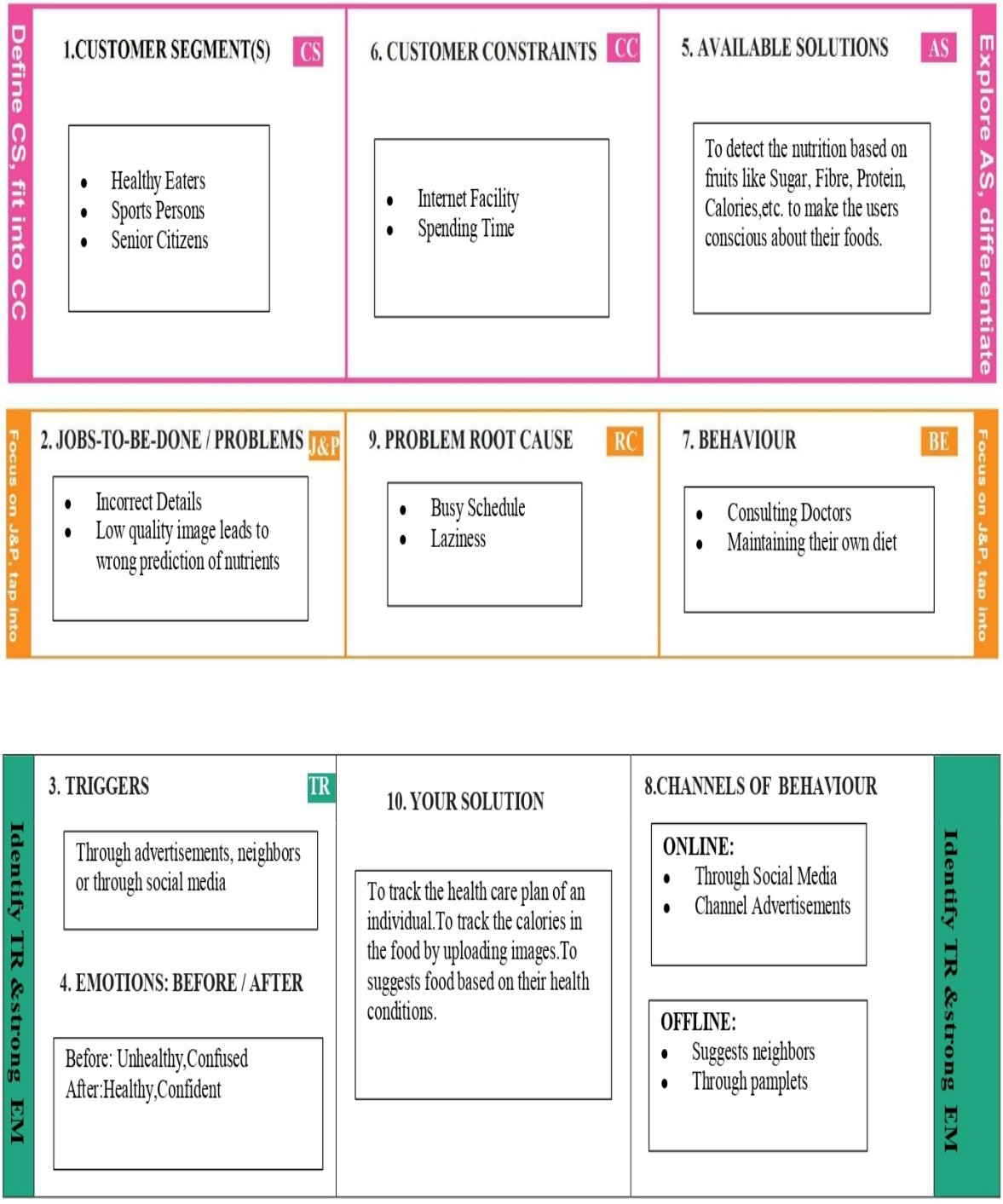
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amountof creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.



3.3 Proposed Solution

S. No	Parameter	Description
1.	Problem Statement (Problem to be solved)	This project has been developed using Machine Learning algorithms. K-Means clustering was used to cluster the food according to calories and then Random Forest Classifier is used to classify the food items and predict the food items based on input given. • Numpy was used to convert features into numpy and then perform the further operations. • Tkinter was used to create interface. KMeanswas used to perform clustering. • Train_test_split was used to divide the dataset into train and test portions to train and test the model.
2.	Idea / Solution description	Items that have been previously rated by the user before play a pertinent part in looking for neighbours that shares appreciation with him When a neighbour of a person is found, various algorithms could be utilized combining the tasks of friends to produce recommendations
3.	Novelty / Uniqueness	First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need 9 to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset.

3.4 Problem Solution Fit



REQUIRMENT ANALYSIS

4.1 Functional Requirement

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	Data Input	Getting the person information.
FR-4	Data processing	Upload the data and classify it.
FR-5	Data Classification	Identify the person's information.
FR-6	Data description	Suggesting the diet plans

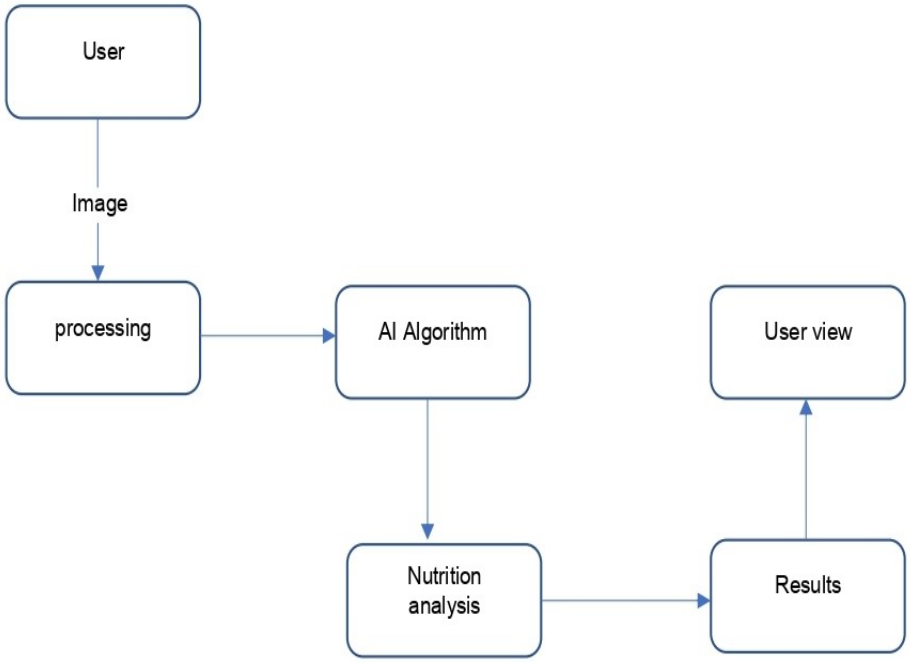
4.2 Non-Functional Requirement

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Datasets of all the food and nutrition used to detection the diet plans.
NFR-2	Security	User and Data
NFR-3	Reliability	The food quality and predicting the nutrition
NFR-4	Performance	The performance is based on the quality of the food used for nutrition.
NFR -5	Availability	It is available for all user to predict the diet plan.
NFR-6	Scalability	Increasing the diet plan and nutrition

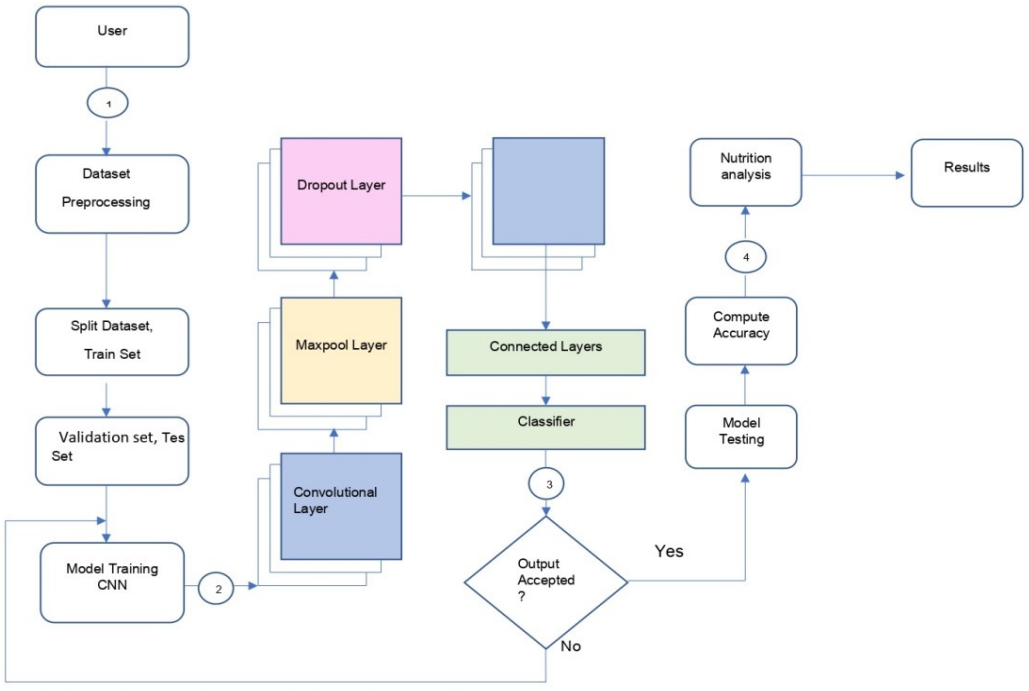
PROJECT DESIGN

5.1 Data Flow Diagram

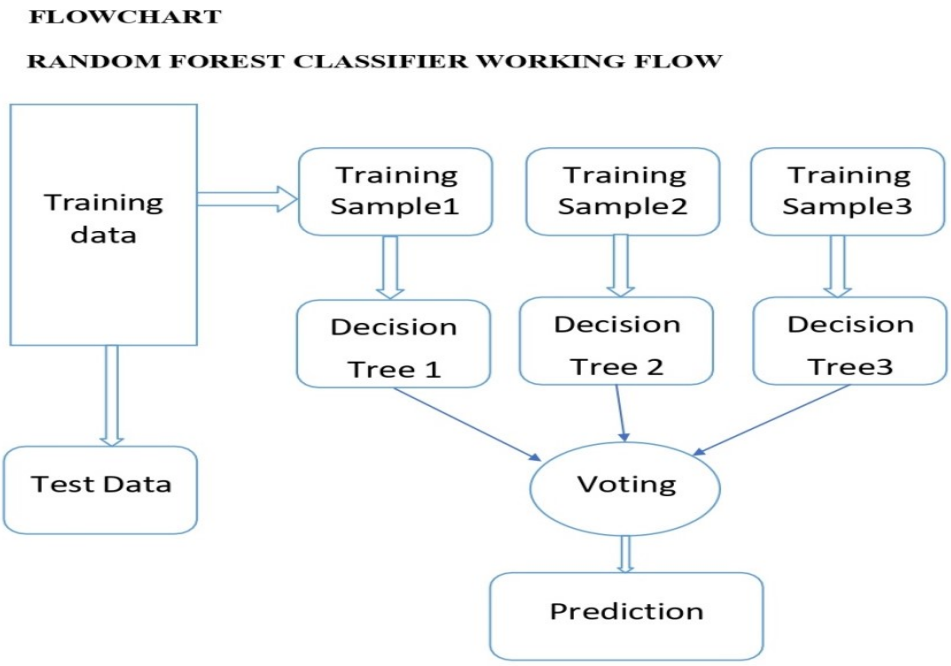
Simplified Data Flow Diagram:



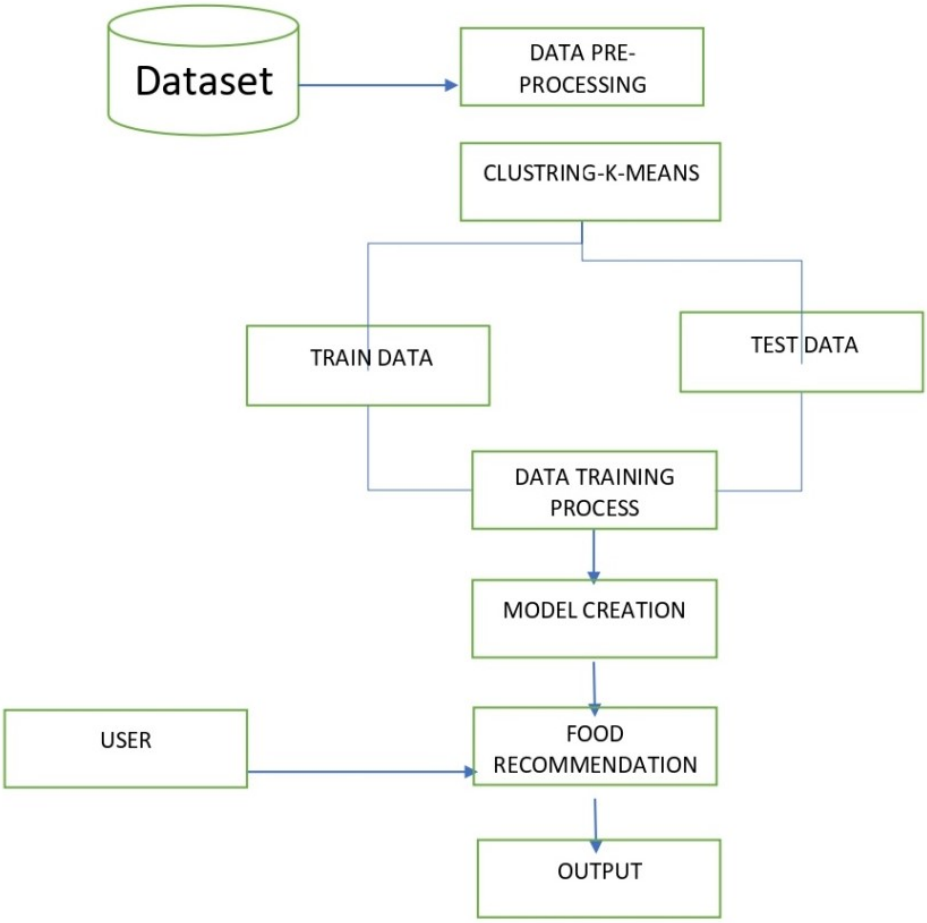
Data Flow Diagram:



5.2 Solution &Technical Architecture



System Architecture



5.3 User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Upload	USN-1	As a user, I can upload the image by gallery	I can uploaded the image	High	Sprit-1
Customer (Web User)	Upload	USN-2	As a user, I can upload the image by take image using camera	I can upload the image	Low	Sprit-2
Customer (Web User)	Registration	USN-3	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account	High	Sprit-1
Customer (Web User)	Login	USN-4	As a user, I can log into the application by entering email & password	I can access my account	High	Sprit-1
Customer Care Executive	Enquiry/Customer services	USN-1	As a customer care executive, I can get the feedback and make report	I can interact with user	Medium	Sprit-1
Administrator	update	USN-1	As a administrator, I can update the performance	I can update and give more functionality	Medium	Sprit-1
Administrator	Add information	USN-2	As a administrator, I can add some extra information about the services	I can improve the access	Low	Sprit-2
Maintenance Team	Maintenance	USN-1	As a member, maintain the any technical problems or the any other issues in the system	I can maintaining the services	High	Sprit-1

PROJECT PLANNING & SCHEDULING

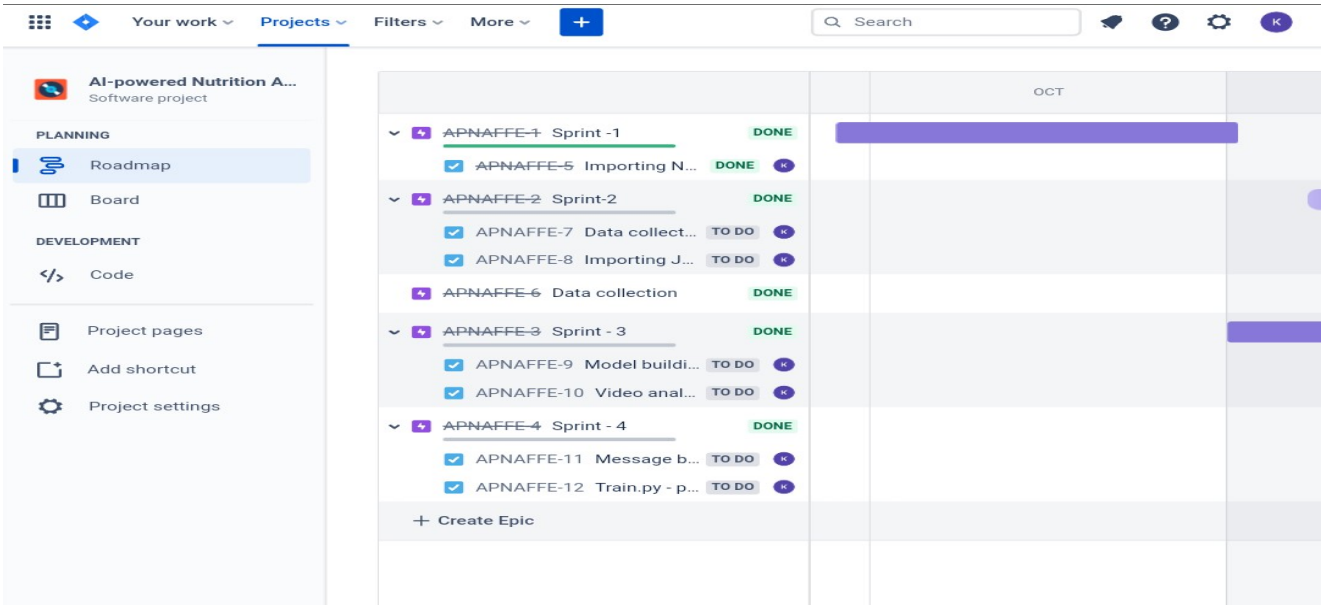
6.1 Sprint Planning & Estimation

Sprint	Functional Requiremnt (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Login	USN-1	As a user, I can enter the personal data	2	High	KAVIYA K AGNES METILDA RAKASH M MAHENDIRAN M
Sprint-1		USN-2	As a user, Its based on individual personal data	1	High	KAVIYA K AGNES METILDA RAKASH M
Sprint-2	Input	USN-3	As the input dataset is given whenever the personal data	2	High	AGNES METILDA RAKASH M MAHENDIRAN M
Sprint-2		USN-4	The system is executed when the given data is detected using the datasets	2	Medium	AGNES METILDA RAKASH M MAHENDIRAN M
Sprint-3	Login	USN-5	As a user, I can chat bot into the application by entering food suggestions based on diseases	2	High	KAVIYA K AGNES METILDA RAKASH M
Sprint-3	Dashboard	USN-6	The final dashboard is made accessible to the user.	2	Medium	AGNES METILDA RAKASH M KAVIYA K
Sprint-4	Output	USN-7	Is recommend a diet to different individual for weight gain and weight loss And also various health problem	2	High	AGNES METILDA RAKASH M KAVIYA K

6.2 Sprint Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story PointsCompleted (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	7 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	4 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA



CODING & SOLUTIONING

FEATURE 1:

```
import numpy as n
import numpy
numpy.__version__
import pandas as pd
pd.read_csv('/content/food.csv')
import pandas as pd
pd.read_csv('/content/nutrition_distriution.csv')
pip install tk
from tkinter import *
import tkinter as tk
from PIL import ImageFilter,Image
import numpy as np
from collections import defaultdict
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('intents2.json').read())
words = pickle.load(open('words.pkl','rb'))
labels = pickle.load(open('labels.pkl','rb'))
data=pd.read_csv('food.csv')
Breakfastdata=data['Breakfast']
BreakfastdataNumpy=Breakfastdata.to_numpy()
Lunchdata=data['Lunch']
LunchdataNumpy=Lunchdata.to_numpy()
Dinnerdata=data['Dinner']
DinnerdataNumpy=Dinnerdata.to_numpy()
Food_itemsdata=data['Food_items']
def show_entry_fields():
print("\n Age: %s\n Veg-NonVeg: %s\n Weight: %s kg\n Hight: %s
cm\n"(e1.get(),e2.get(),e3.get(),e4.get()))
def Weight_Loss():
show_entry_fields()
breakfastfoodseparated=[]
```

```
Lunchfoodseparated=[]
Dinnerfoodseparated=[]
breakfastfoodseparatedID=[]
LunchfoodseparatedID=[]
DinnerfoodseparatedID=[]
for i in range(len(Breakfastdata)):
  if BreakfastdataNumpy[i]==1:
breakfastfoodseparated.append( Food_itemsdata[i] )
breakfastfoodseparatedID.append(i)
  if LunchdataNumpy[i]==1:
Lunchfoodseparated.append(Food_itemsdata[i])
LunchfoodseparatedID.append(i)
  if DinnerdataNumpy[i]==1:
Dinnerfoodseparated.append(Food_itemsdata[i])
DinnerfoodseparatedID.append(i)
LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
Valapnd=[0]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
#print(LunchfoodseparatedIDdata)
# retrievingBreafast data rows by loc method
breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
```

```
import
os
        import time
        import pandas as pd
        import numpy as np
        from tkinter import *
        from PIL import ImageFilter,Image
        from tkinter import filedialog, messagebox
        from sklearn.cluster import KMeans
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        import nltk
        from nltk.stem.lancaster import LancasterStemmer
        stemmer = LancasterStemmer()
        import pickle
        import numpy as np
        from keras.models import load_model
        model = load_model('model.h5')
        import json
        import random
        intents = json.loads(open('intents2.json').read())
        words = pickle.load(open('words.pkl','rb'))
        labels = pickle.load(open('labels.pkl','rb'))
        data=pd.read_csv('food.csv')
        Breakfastdata=data['Breakfast']
        BreakfastdataNumpy=Breakfastdata.to_numpy()
        Lunchdata=data['Lunch']
        LunchdataNumpy=Lunchdata.to_numpy()
        Dinnerdata=data['Dinner']
        DinnerdataNumpy=Dinnerdata.to_numpy()
        Food_itemsdata=data['Food_items']
        def show_entry_fields():
        print("\n Age: %s\n Veg-NonVeg: %s\n Weight: %s kg\n Hight: %s cm\n" %
        (e1.get(), e2.get(),e3.get(), e4.get()))
        def Weight_Loss():
        show_entry_fields()
        breakfastfoodseparated=[]
        Lunchfoodseparated=[]
        Dinnerfoodseparated=[]
        breakfastfoodseparatedID=[]
        LunchfoodseparatedID=[]
        DinnerfoodseparatedID=[]
            for i in range(len(Breakfastdata)):
                if BreakfastdataNumpy[i]==1:
        breakfastfoodseparated.append( Food_itemsdata[i] )
        breakfastfoodseparatedID.append(i)
```

```
        if LunchdataNumpy[i]==1:
Lunchfoodseparated.append(Food_itemsdata[i])
LunchfoodseparatedID.append(i)
        if DinnerdataNumpy[i]==1:
Dinnerfoodseparated.append(Food_itemsdata[i])
DinnerfoodseparatedID.append(i)
    # retrieving Lunch data rows by loc method |
LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
    #print(LunchfoodseparatedIDdata)
val=list(np.arange(5,15))
Valapnd=[0]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
    #print(LunchfoodseparatedIDdata)
    # retrievingBreafast data rows by loc method
breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
    # retrieving Dinner Data rows by loc method
DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
    #calculating BMI
    age=int(e1.get())
    veg=float(e2.get())
    weight=float(e3.get())
    height=float(e4.get())
bmi = weight/((height/100)**2)
agewiseinp=0
    for lp in range (0,80,20):
test_list=np.arange(lp,lp+20)
        for i in test_list:
if(i == age):
                tr=round(lp/20)
agecl=round(lp/20)
    #conditions
print("Your body mass index is: ", bmi)
    if ( bmi< 16):
print("Acoording to your BMI, you are Severely Underweight")
```

```
clbmi=4
elif( bmi>= 16 and bmi< 18.5):
print("Acoording to your BMI, you are Underweight")
clbmi=3
elif( bmi>= 18.5 and bmi< 25):
print("Acoording to your BMI, you are Healthy")
clbmi=2
elif( bmi>= 25 and bmi< 30):
print("Acoording to your BMI, you are Overweight")
clbmi=1
elif( bmi>=30):
print("Acoording to your BMI, you are Severely Overweight")
clbmi=0
   #converting into numpy array
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
   breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
ti=(clbmi+agecl)/2
   ## K-Means Based  Dinner Food
   Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # retrieving the labels for dinner food
dnrlbl=kmeans.labels_
   ## K-Means Based  lunch Food
   Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # retrieving the labels for lunch food
lnchlbl=kmeans.labels_
   ## K-Means Based  lunch Food

Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # retrieving the labels for breakfast food
brklbl=kmeans.labels_
inp=[]
   ## Reading of the Dataet
datafin=pd.read_csv('nutrition_distriution.csv')
   ## train set
dataTog=datafin.T
bmicls=[0,1,2,3,4]
```

```
agecls=[0,1,2,3,4]
weightlosscat = dataTog.iloc[[1,2,7,8]]
weightlosscat=weightlosscat.T
weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
weightgaincat=weightgaincat.T
healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]
healthycat=healthycat.T
weightlosscatDdata=weightlosscat.to_numpy()
weightgaincatDdata=weightgaincat.to_numpy()
healthycatDdata=healthycat.to_numpy()
weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]
weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]
healthycat=healthycatDdata[1:,0:len(healthycatDdata)]
   weightlossfin=np.zeros((len(weightlosscat)*5,6),dtype=np.float32)
   weightgainfin=np.zeros((len(weightgaincat)*5,10),dtype=np.float32)
healthycatfin=np.zeros((len(healthycat)*5,9),dtype=np.float32)
   t=0
   r=0
   s=0
yt=[]
yr=[]
ys=[]
   for zz in range(5):
      for jj in range(len(weightlosscat)):
valloc=list(weightlosscat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightlossfin[t]=np.array(valloc)
yt.append(brklbl[jj])
         t+=1
      for jj in range(len(weightgaincat)):
valloc=list(weightgaincat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightgainfin[r]=np.array(valloc)
yr.append(lnchlbl[jj])
         r+=1
      for jj in range(len(healthycat)):
valloc=list(healthycat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
healthycatfin[s]=np.array(valloc)
ys.append(dnrlbl[jj])
         s+=1
X_test=np.zeros((len(weightlosscat),6),dtype=np.float32)
print('####################')
```

```
    #randomforest
    for jj in range(len(weightlosscat)):
valloc=list(weightlosscat[jj])
valloc.append(agecl)
valloc.append(clbmi)
X_test[jj]=np.array(valloc)*ti
X_train=weightlossfin# Features
y_train=yt # Labels
    #Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
    #Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
    #print (X_test[1])
    X_test2=X_test
y_pred=clf.predict(X_test)
    print ('SUGGESTED FOOD ITEMS ::')
    for ii in range(len(y_pred)):
        if y_pred[ii]==2:     #weightloss
            print (Food_itemsdata[ii])
findata=Food_itemsdata[ii]
            if int(veg)==1:
datanv=['Chicken Burger']
                for it in range(len(datanv)):
                    if findata==datanv[it]:
                        print('VegNovVeg')
print('\n Thank You for taking our recommendations. :)')
def Weight_Gain():
show_entry_fields()
breakfastfoodseparated=[]
Lunchfoodseparated=[]
Dinnerfoodseparated=[]
breakfastfoodseparatedID=[]
LunchfoodseparatedID=[]
DinnerfoodseparatedID=[]
    for i in range(len(Breakfastdata)):
        if BreakfastdataNumpy[i]==1:
breakfastfoodseparated.append( Food_itemsdata[i] )
breakfastfoodseparatedID.append(i)
        if LunchdataNumpy[i]==1:
Lunchfoodseparated.append(Food_itemsdata[i])
LunchfoodseparatedID.append(i)
        if DinnerdataNumpy[i]==1:
Dinnerfoodseparated.append(Food_itemsdata[i])
DinnerfoodseparatedID.append(i)
    # retrieving rows by loc method |
LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]
```

```
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
   # retrieving rows by loc method
breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
   breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
   # retrieving rows by loc method
DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
   DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
   #claculating BMI
   age=int(e1.get())
   veg=float(e2.get())
   weight=float(e3.get())
   height=float(e4.get())
bmi = weight/((height/100)**2)
   for lp in range (0,80,20):
test_list=np.arange(lp,lp+20)
      for i in test_list:
if(i == age):
            tr=round(lp/20)
agecl=round(lp/20)
print("Your body mass index is: ", bmi)
   if ( bmi< 16):
print("Acoording to your BMI, you are Severely Underweight")
clbmi=4
elif( bmi>= 16 and bmi< 18.5):
print("Acoording to your BMI, you are Underweight")
clbmi=3
elif( bmi>= 18.5 and bmi< 25):
print("Acoording to your BMI, you are Healthy")
clbmi=2
elif( bmi>= 25 and bmi< 30):
print("Acoording to your BMI, you are Overweight")
clbmi=1
elif( bmi>=30):
print("Acoording to your BMI, you are Severely Overweight")
```

```
clbmi=0
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
   breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
ti=(bmi+agecl)/2
   ## K-Means Based  Dinner Food
   Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # plt.bar(XValu,kmeans.labels_)
dnrlbl=kmeans.labels_
   # plt.title("Predicted Low-High Weigted Calorie Foods")
   ## K-Means Based  lunch Food
   Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # fig,axs=plt.subplots(1,1,figsize=(15,5))
   # plt.bar(XValu,kmeans.labels_)
lnchlbl=kmeans.labels_
   # plt.title("Predicted Low-High Weigted Calorie Foods")
   ## K-Means Based  lunch Food

Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]
   X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
   # fig,axs=plt.subplots(1,1,figsize=(15,5))
   # plt.bar(XValu,kmeans.labels_)
brklbl=kmeans.labels_
   # plt.title("Predicted Low-High Weigted Calorie Foods")
inp=[]
   ## Reading of the Dataet
datafin=pd.read_csv('nutrition_distriution.csv')
datafin.head(5)
dataTog=datafin.T
bmicls=[0,1,2,3,4]
agecls=[0,1,2,3,4]
weightlosscat = dataTog.iloc[[1,2,7,8]]
weightlosscat=weightlosscat.T
weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
weightgaincat=weightgaincat.T
healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]
healthycat=healthycat.T
weightlosscatDdata=weightlosscat.to_numpy()
```

```
weightgaincatDdata=weightgaincat.to_numpy()
healthycatDdata=healthycat.to_numpy()
weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]
weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]
healthycat=healthycatDdata[1:,0:len(healthycatDdata)]
   weightlossfin=np.zeros((len(weightlosscat)*5,6),dtype=np.float32)
   weightgainfin=np.zeros((len(weightgaincat)*5,10),dtype=np.float32)
healthycatfin=np.zeros((len(healthycat)*5,9),dtype=np.float32)
   t=0
   r=0
   s=0
yt=[]
yr=[]
ys=[]
   for zz in range(5):
      for jj in range(len(weightlosscat)):
valloc=list(weightlosscat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightlossfin[t]=np.array(valloc)
yt.append(brklbl[jj])
         t+=1
      for jj in range(len(weightgaincat)):
valloc=list(weightgaincat[jj])
         #print (valloc)
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightgainfin[r]=np.array(valloc)
yr.append(lnchlbl[jj])
         r+=1
      for jj in range(len(healthycat)):
valloc=list(healthycat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
healthycatfin[s]=np.array(valloc)
ys.append(dnrlbl[jj])
         s+=1
X_test=np.zeros((len(weightgaincat),10),dtype=np.float32)
print('####################')
   # In[287]:
   for jj in range(len(weightgaincat)):
valloc=list(weightgaincat[jj])
valloc.append(agecl)
valloc.append(clbmi)
X_test[jj]=np.array(valloc)*ti
X_train=weightgainfin# Features
```

```
y_train=yr # Labels
   #Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
   #Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
   X_test2=X_test
y_pred=clf.predict(X_test)
   print ('SUGGESTED FOOD ITEMS ::')
   for ii in range(len(y_pred)):
      if y_pred[ii]==1:
         print (Food_itemsdata[ii])
findata=Food_itemsdata[ii]
         if int(veg)==2:
datanv=['Chicken Burger']
            for it in range(len(datanv)):
               if findata==datanv[it]:
                  print('VegNovVeg')
print('\n Thank You for taking our recommendations. :)')
def Healthy():
show_entry_fields()
breakfastfoodseparated=[]
Lunchfoodseparated=[]
Dinnerfoodseparated=[]
breakfastfoodseparatedID=[]
LunchfoodseparatedID=[]
DinnerfoodseparatedID=[]
   for i in range(len(Breakfastdata)):
      if BreakfastdataNumpy[i]==1:
breakfastfoodseparated.append( Food_itemsdata[i] )
breakfastfoodseparatedID.append(i)
      if LunchdataNumpy[i]==1:
Lunchfoodseparated.append(Food_itemsdata[i])
LunchfoodseparatedID.append(i)
      if DinnerdataNumpy[i]==1:
Dinnerfoodseparated.append(Food_itemsdata[i])
DinnerfoodseparatedID.append(i)
   # retrieving Lunch data rows by loc method |
LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
   #print(LunchfoodseparatedIDdata)
val=list(np.arange(5,15))
Valapnd=[0]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T
   #print(LunchfoodseparatedIDdata)
   # retrievingBreafast data rows by loc method
```

```
breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
   breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
   # retrieving Dinner Data rows by loc method
DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
   DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
   #calculating BMI
   age=int(e1.get())
   veg=float(e2.get())
   weight=float(e3.get())
   height=float(e4.get())
bmi = weight/((height/100)**2)
agewiseinp=0
   for lp in range (0,80,20):
test_list=np.arange(lp,lp+20)
       for i in test_list:
if(i == age):
              tr=round(lp/20)
agecl=round(lp/20)
   #conditions
print("Your body mass index is: ", bmi)
   if ( bmi< 16):
print("Acoording to your BMI, you are Severely Underweight")
clbmi=4
elif( bmi>= 16 and bmi< 18.5):
print("Acoording to your BMI, you are Underweight")
clbmi=3
elif( bmi>= 18.5 and bmi< 25):
print("Acoording to your BMI, you are Healthy")
clbmi=2
elif( bmi>= 25 and bmi< 30):
print("Acoording to your BMI, you are Overweight")
clbmi=1
elif( bmi>=30):
print("Acoording to your BMI, you are Severely Overweight")
clbmi=0
   #converting into numpy array
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
```

```
    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
ti=(clbmi+agecl)/2
    ## K-Means Based  Dinner Food
    Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]
    X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
    # retrieving the labels for dinner food
dnrlbl=kmeans.labels_
    ## K-Means Based  lunch Food
    Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]
    X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
    # retrieving the labels for lunch food
lnchlbl=kmeans.labels_
    ## K-Means Based  lunch Food

Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]
    X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
XValu=np.arange(0,len(kmeans.labels_))
    # retrieving the labels for breakfast food
brklbl=kmeans.labels_
inp=[]
    ## Reading of the Dataet
datafin=pd.read_csv('nutrition_distriution.csv')
    ## train set
dataTog=datafin.T
bmicls=[0,1,2,3,4]
agecls=[0,1,2,3,4]
weightlosscat = dataTog.iloc[[1,2,7,8]]
weightlosscat=weightlosscat.T
weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
weightgaincat=weightgaincat.T
healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]
healthycat=healthycat.T
weightlosscatDdata=weightlosscat.to_numpy()
weightgaincatDdata=weightgaincat.to_numpy()
healthycatDdata=healthycat.to_numpy()
weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]
weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]
healthycat=healthycatDdata[1:,0:len(healthycatDdata)]
    weightlossfin=np.zeros((len(weightlosscat)*5,6),dtype=np.float32)
    weightgainfin=np.zeros((len(weightgaincat)*5,10),dtype=np.float32)
healthycatfin=np.zeros((len(healthycat)*5,9),dtype=np.float32)
```

```
    t=0
    r=0
    s=0
yt=[]
yr=[]
ys=[]
    for zz in range(5):
        for jj in range(len(weightlosscat)):
valloc=list(weightlosscat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightlossfin[t]=np.array(valloc)
yt.append(brklbl[jj])
            t+=1
        for jj in range(len(weightgaincat)):
valloc=list(weightgaincat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
weightgainfin[r]=np.array(valloc)
yr.append(lnchlbl[jj])
            r+=1
        for jj in range(len(healthycat)):
valloc=list(healthycat[jj])
valloc.append(bmicls[zz])
valloc.append(agecls[zz])
healthycatfin[s]=np.array(valloc)
ys.append(dnrlbl[jj])
            s+=1
X_test=np.zeros((len(weightlosscat),6),dtype=np.float32)
print('####################')
    #randomforest
    for jj in range(len(weightlosscat)):
valloc=list(weightlosscat[jj])
valloc.append(agecl)
valloc.append(clbmi)
X_test[jj]=np.array(valloc)*ti
X_train=weightlossfin# Features
y_train=ys # Labels
    #Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
    #Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
    #print (X_test[1])
    X_test2=X_test
y_pred=clf.predict(X_test)
    print ('SUGGESTED FOOD ITEMS ::')
```

```
    for ii in range(len(y_pred)):
        if y_pred[ii]==2:
            print (Food_itemsdata[ii])
findata=Food_itemsdata[ii]
            if int(veg)==1:
datanv=['Chicken Burger']
                for it in range(len(datanv)):
                    if findata==datanv[it]:
                        ('VegNovVeg')
print('\n Thank You for taking our recommendations. :)')
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
sentence_words = [stemmer.stem(word.lower()) for word in sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the
sentence
def bow(sentence, words, show_details=True):
    # tokenize the pattern
sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))
def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
results.sort(key=lambda x: x[1], reverse=True)
return_list = []
    for r in results:
return_list.append({"intent": labels[r[0]], "probability": str(r[1])})
    return return_list
#getting chatbot response
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
```

```
list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result
def chatbot_response(text):
ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
if __name__ == '__main__':
    import tkinter as tk
    IMAGE_PATH = 'image.jpg'
    WIDTH=700
    HEIGTH = 500
    root = tk.Tk()
root.title("Diet Recommendation System")
    canvas = tk.Canvas(root, width=WIDTH, height=HEIGTH)
canvas.pack()
    from PIL import Image, ImageTk
img = ImageTk.PhotoImage(Image.open(IMAGE_PATH).resize((WIDTH,
HEIGTH), Image.ANTIALIAS))
canvas.background = img  # Keep a reference in case this code is put in a function.
bg = canvas.create_image(0, 0, anchor=tk.NW, image=img)
    label1 = tk.Label(root, text='Age ',font=("Times New Roman",10),bg='#389396')
canvas.create_window( 200, 100, window=label1)
    label2= tk.Label(root, text='veg/Non veg (1/0) ',font=("Times New
Roman",10),bg='#389396')
canvas.create_window( 200, 150, window=label2)
    label3= tk.Label(root, text=' Weight (in kg) ',font=("Times New
Roman",10),bg='#389396')
canvas.create_window( 200, 200, window=label3)
    label4= tk.Label(root, text=' Height (in cm) ',font=("Times New
Roman",10),bg='#389396')
canvas.create_window( 200, 250, window=label4)
    e1 = tk.Entry (root, width=40)
canvas.create_window(500, 100, window=e1)
    e2 = tk.Entry (root, width=40)
canvas.create_window(500, 150, window=e2)
    e3 = tk.Entry (root, width=40)
canvas.create_window(500, 200, window=e3)
    e4 = tk.Entry (root, width=40)
canvas.create_window(500, 250, window=e4)
    button2 = tk.Button (root, text='Quit',command=root.quit, bg='#fcba03')
canvas.create_window(200, 300, window=button2)
    button3 = tk.Button (root, text='Weight Loss',command=Weight_Loss,
```

```
bg='#fcba03')
canvas.create_window(270, 300, window=button3)
   button4 = tk.Button (root, text='Weight Gain',command=Weight_Gain,
bg='#fcba03')
canvas.create_window(380, 300, window=button4)
   button5 = tk.Button(root,text='Healthy',command=Healthy,bg='#fcba03')
canvas.create_window(450, 300, window=button5)
   def disease ():
      def send():
msg = EntryBox.get("1.0",'end-1c').strip()
EntryBox.delete("0.0",END)
         if msg != '':
ChatLog.config(state=NORMAL)
ChatLog.insert(END, "user: " + msg + '\n\n')
ChatLog.config(foreground="#442265", font=("Verdana", 8 ))
            res = chatbot_response(msg)
ChatLog.insert(END, "Food: " + res + '\n\n')
ChatLog.config(state=DISABLED)
ChatLog.yview(END)
      base = tk.Toplevel(root)
base.title("Food suggestions based diseases")
base.geometry("500x500")
base.resizable(width=TRUE, height=TRUE)
      #Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)
ChatLog.config(state=DISABLED)
      #Bind scrollbar to Chat window
      scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set
      #Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12",
height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )
      #Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
      #EntryBox.bind("<Return>", send)
      #Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
base.mainloop()
button3 = tk.Button (root, text='Food suggestions based diseases',command=disease
, bg='#fcba03') # button to call the 'values' command above
canvas.create_window(600, 300, window=button3)
```

MODEL BUILDING :

```
import
nltk
        nltk.download('punkt')
        from nltk.stem.lancaster import LancasterStemmer
        stemmer = LancasterStemmer()
        import numpy
        import tensorflow
        import random
        import json
        import pickle
        from keras.models import Sequential
        from keras.layers import Dense, Activation, Dropout
        from tensorflow.keras.optimizers import SGD
        import random
        import numpy as np
        with open("intents2.json") as file:
           data = json.load(file)
        words = []
        labels = []
        docs_x = []
        docs_y = []
        #tokenizing each pattern in our dataset
        #adding to list of tokenize patterns or wrds and their labels or tag
        for intent in data["intents"]:
           for pattern in intent["patterns"]:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])
           if intent["tag"] not in labels:
        labels.append(intent["tag"])
        #reducing each words to their root word and sorting them
```

```
words = [stemmer.stem(w.lower()) for w in words if w != "?"]
words = sorted(list(set(words)))
labels = sorted(labels)
#inserting list of words and labels into pickle file
#performing serialization
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(labels,open('labels.pkl','wb'))
training = []
output = []
out_empty = [0 for _ in range(len(labels))]
#encoding our input to bag of words
for x, doc in enumerate(docs_x):
   bag = []
wrds = [stemmer.stem(w.lower()) for w in doc]
   for w in words:
      if w in wrds:
bag.append(1)
      else:
bag.append(0)
output_row = out_empty[:]
output_row[labels.index(docs_y[x])] = 1
training.append(bag)
output.append(output_row)
training = numpy.array(training)
output = numpy.array(output)
#creating bag of words encoding of dataset
def bag_of_words(s, words):
   bag = [0 for _ in range(len(words))]
s_words = nltk.word_tokenize(s)
s_words = [stemmer.stem(word.lower()) for word in s_words]
   for se in s_words:
      for i, w in enumerate(words):
         if w == se:
            bag[i] = 1
   return numpy.array(bag)
#building deep neural networks
model = Sequential()
model.add(Dense(128, input_shape=(len(training[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(output[0]), activation='softmax'))
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient
gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',                optimizer=sgd,
```

```
metrics=['accuracy'])
#fitting and saving the model
hist = model.fit(np.array(training), np.array(output), epochs=1000, batch_size=8,
verbose=1)
model.save('model.h5', hist)
```

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

You can successfully age by avoiding medical conditions like heart disease and diabetes. Eating healthy food is the best way to prevent the inflammation and other consequences of these medical conditions, according to a 2017 report in Molecules. The authors note that healthy food can also play a positive role in fighting cancer.Processed foods like pre-cut vegetables and meat are quality convenience foods for busy people and for those who can't visit a faraway market to buy vegetables and meat.Eating healthy food regularly will keep the body functioning optimally, and therefore, people will not usually feel sluggish, weak, or have trouble concentrating on anything. According to Sawhney (2021) organic food

makes people happy and hopeful, while consuming food high in protein improves motivation and concentration.

DISADVANTAGES:

A 2015 paper in Nutrition Reviews gave a detailed analysis of food prices. The article showed that vegetables and fruits had a higher per-calorie cost than sweets. So, it actually costs more to stay healthy and keep a healthy body weight through a diet that includes nutritious fruits and vegetables. Frequent intake of processed foods can make people become angry and irritable. Consumption of natural whole foods can help level out your mood, sustain energy levels and leave you feeling content and relaxed. Some processed foods are genetically modified, which may cause a negative impact on your health.One of the biggest disadvantages of eating healthy food is that it can be expensive. According to the Food Foundation, cited in (Goncalves, 2021), healthier foods are almost three times as expensive as their less healthy counterparts. It is very difficult for people with less available money to buy fresh fruits and vegetables regularly.Eating healthy is important for human health, energy, and appearance. It can help people reduce the risk for chronic health challenges. However, eating healthy food does not have to be expensive or difficult. It can be easier to maintain if people have the right meal choices and are consistent with their eating habits. There are many healthy food options to choose from and people need to find what works best for them. Having said that, it should be mentioned that market and regulatory authorities need to ensure that buying healthy foods is within the buying capacity of all the citizens.Eating healthy can be difficult to maintain over a long period of time.

11.CONCLUSION

Diet Food Suggestions Based Clustering MethodIn this project, we proposed a predictive approach using machine learning algorithm, where K-Means clustering and Random Forest was more accurate. Based on the obtained results we recommended food for different level for diet food.As the world grows more fitness-conscious with passing time, the demand for technological solutions to cater to this burgeoning demand is diversifying. Lately, a number of startups in India and worldwide are using predictive analytics artificial intelligence and natural language processing to help scores of fitness enthusiasts to track and monitor their nutrition and calorie intake.

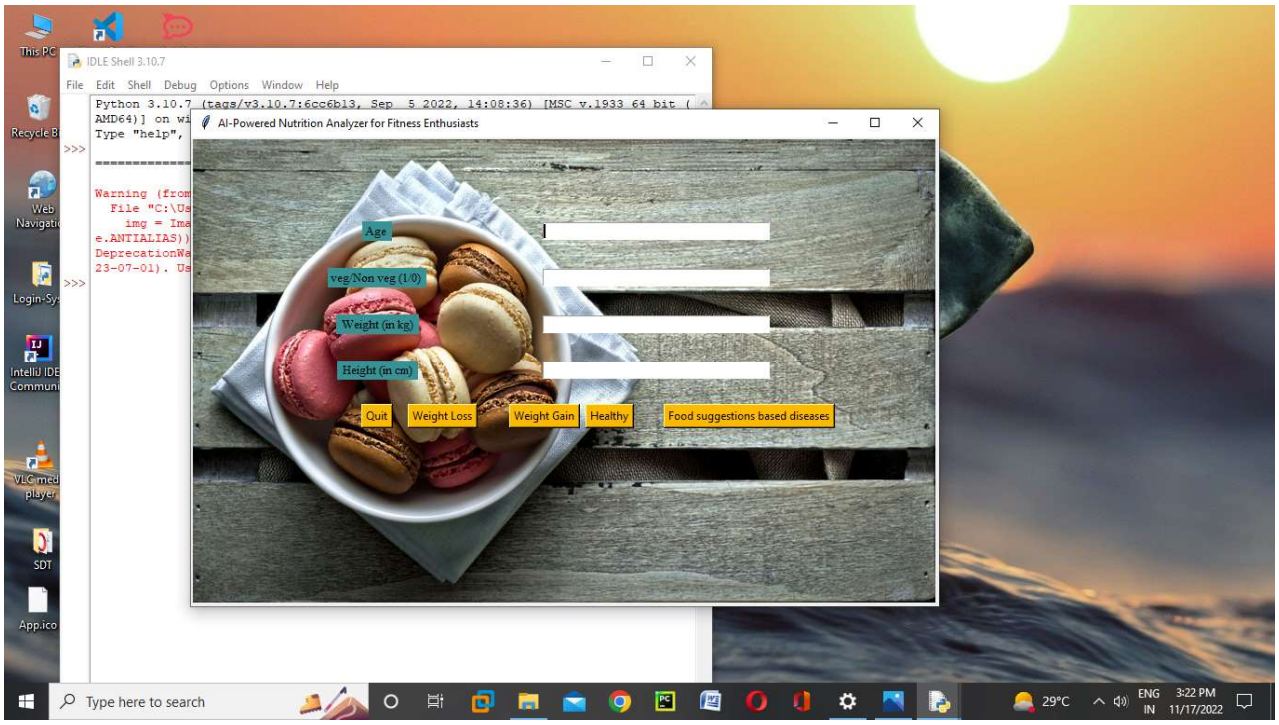
In this global trend has had a positive impact on scores of startups and websites catering to this segment. AI and its various subsets have been leveraged by these platforms to identify the calorie intake and also to make food recommendations for a healthy diet. In most cases, what we see is that these platforms act as a data repository where while providing real-time information to its users, it also makes available to numerous clients who work in this field for a determined rate.

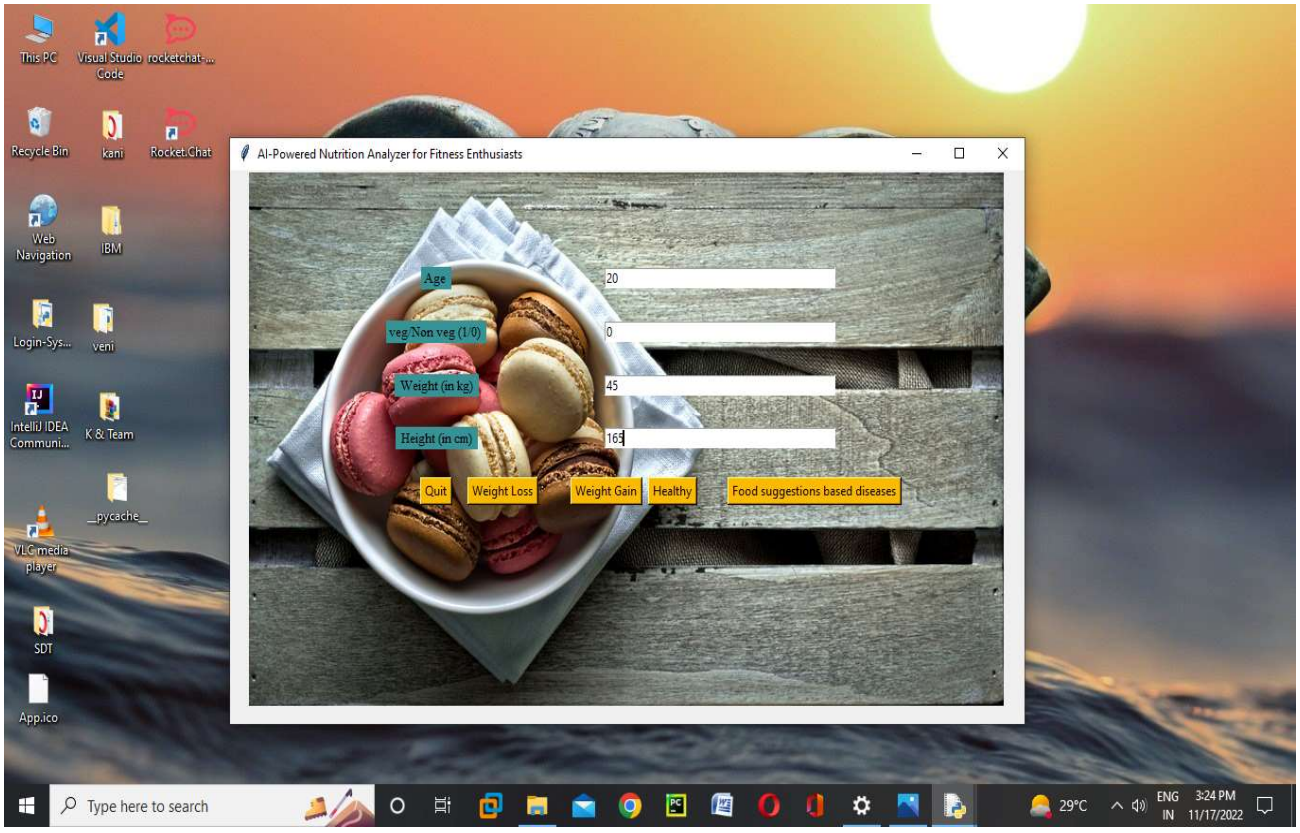
12. FUTURE SCOPE

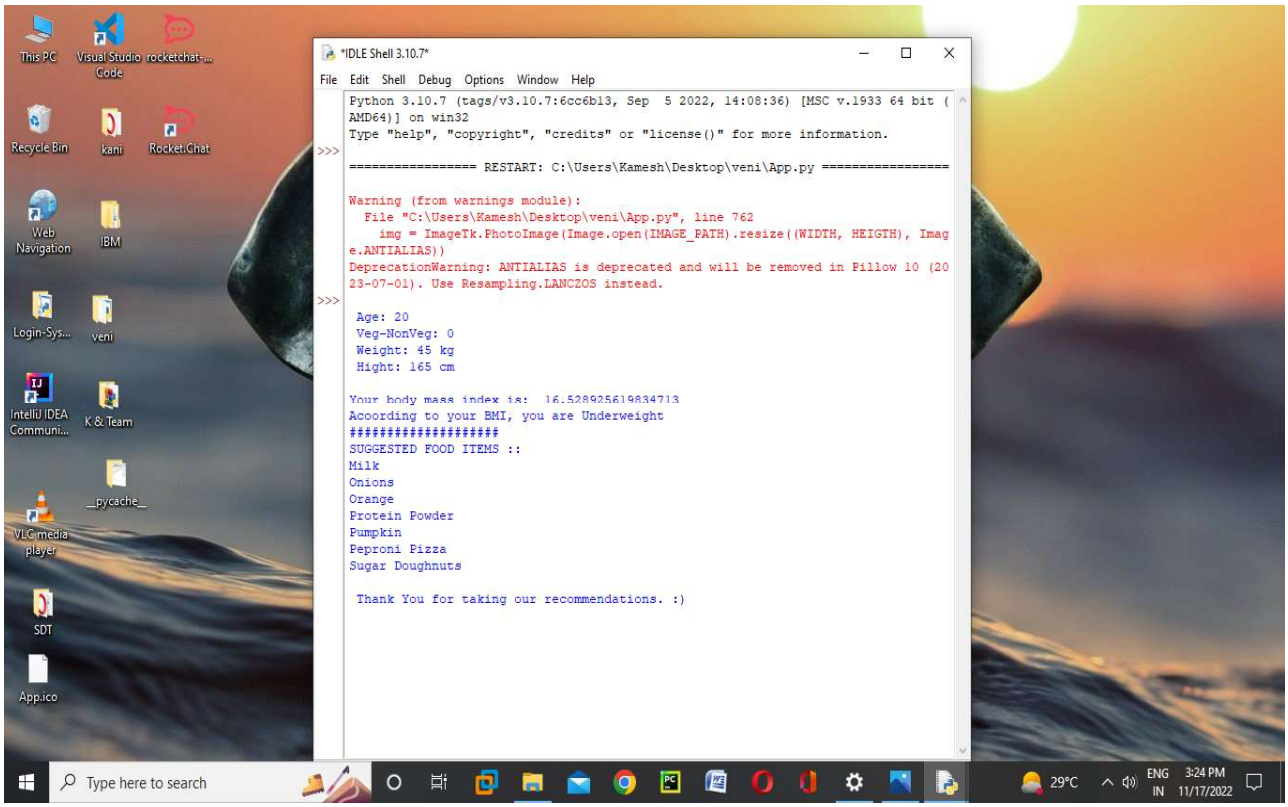
• The module can be implemented as a cloud-based application.

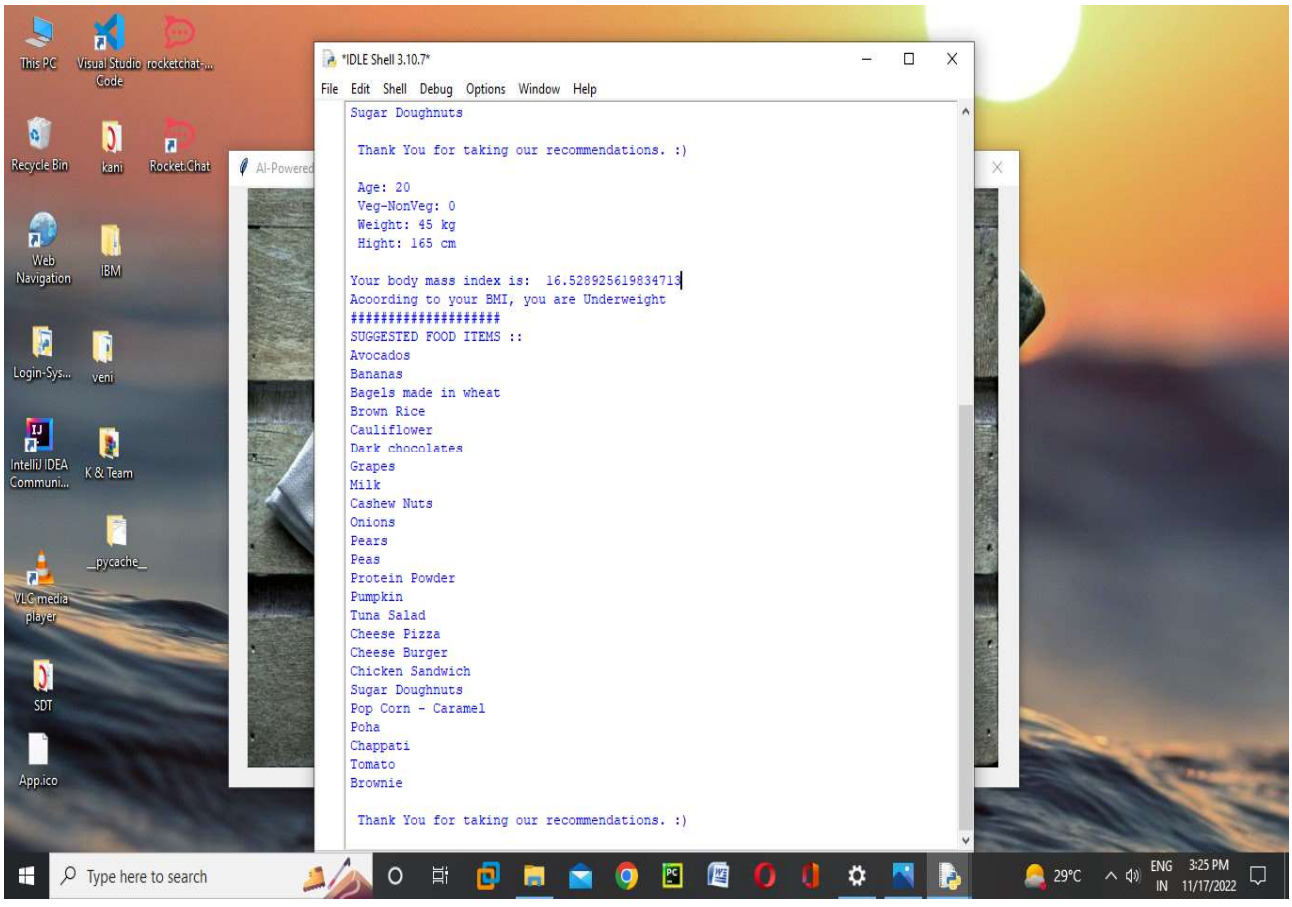
• Packaged as a single entity, ready for productionenvironment deployment

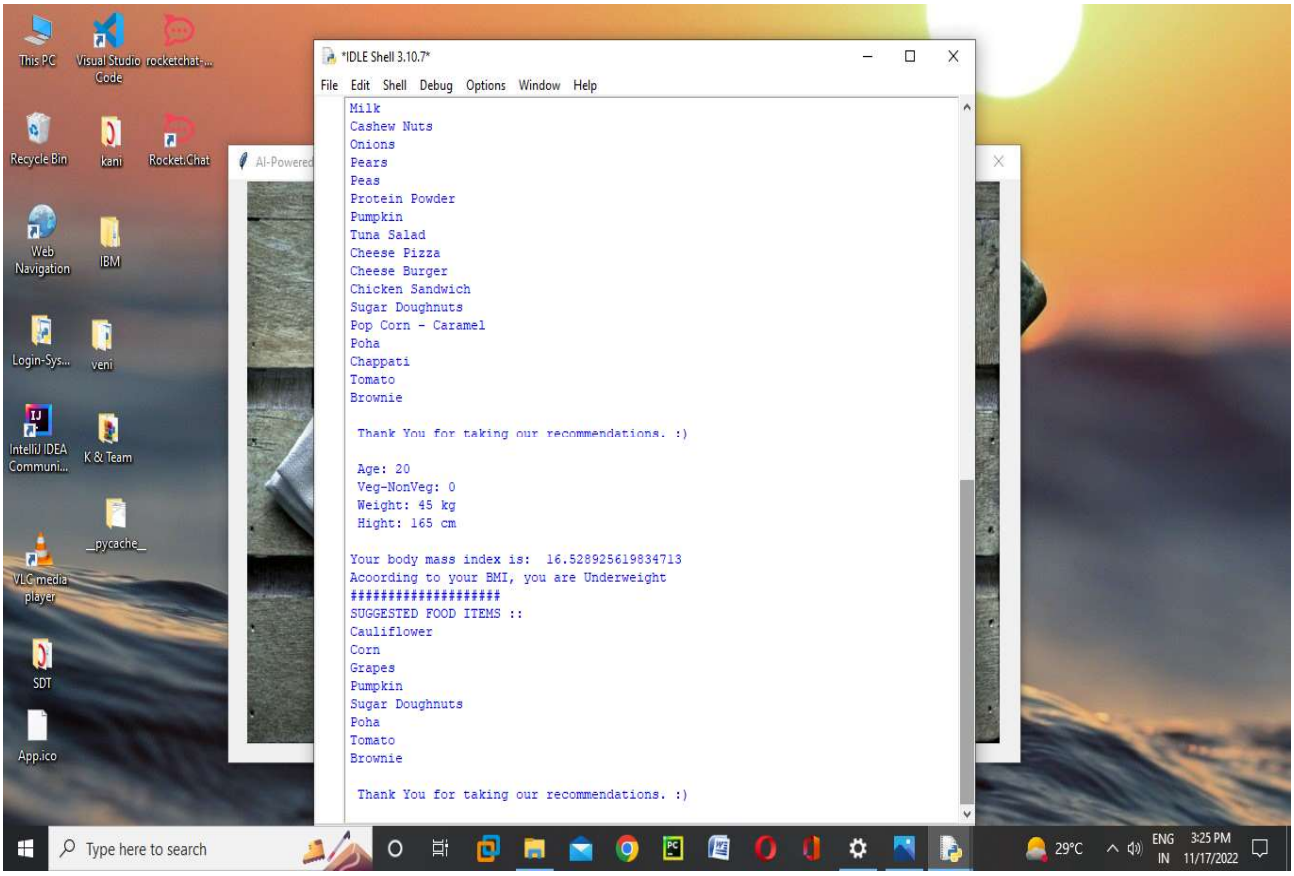
12. OUTPUT

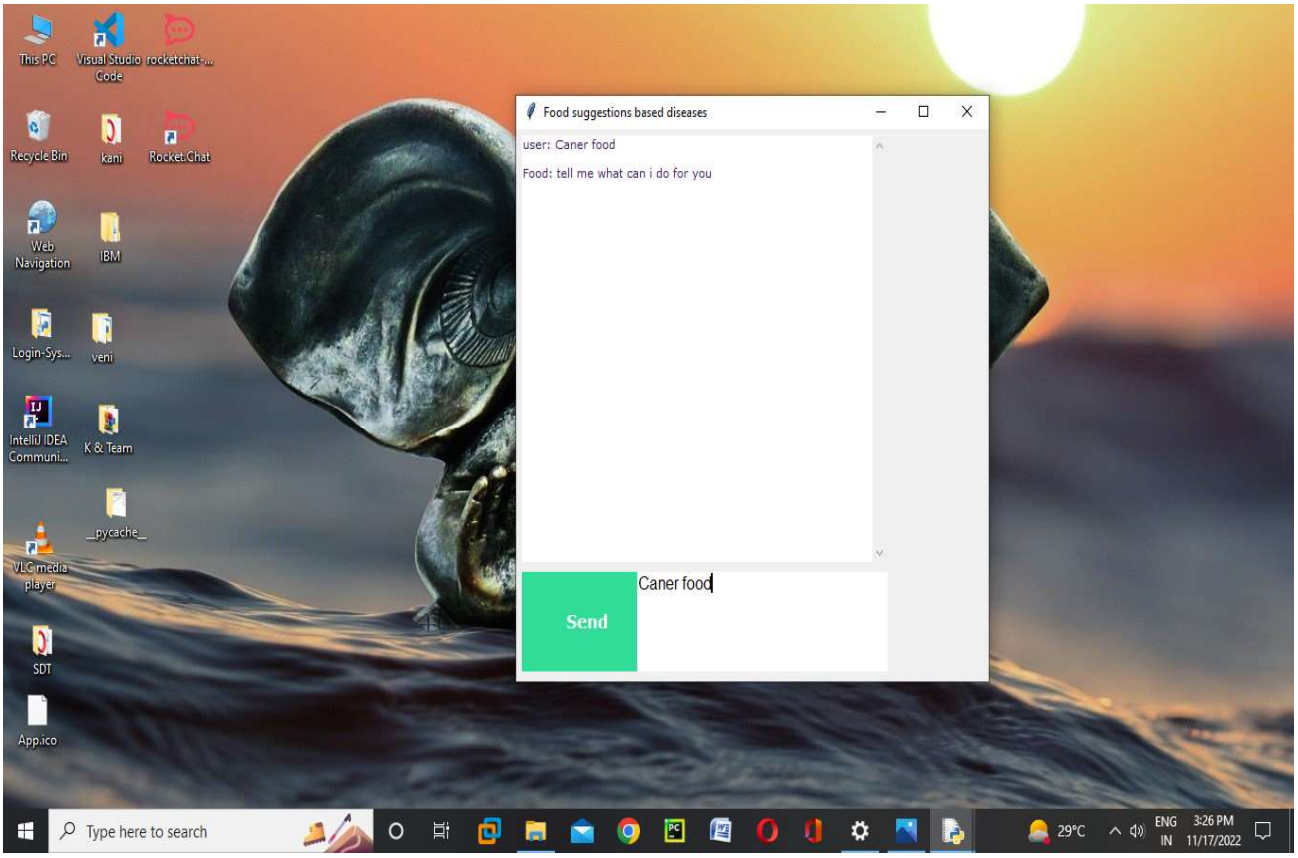

IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (
AMD64)] on wi
Type "help",
>>>
Warning (from
File "C:\Us
img = Ima
e.ANTIALIAS))
DeprecationWa
23-07-01). Us
>>>
AI-Powered Nutrition Analyzer for Fitness Enthusiasts
Age
veg/Non veg (1/0)
Weight (in kg)
Height (in cm)
Quit
Weight Loss
Weight Gain
Healthy
Food suggestions based diseases
Type here to search
29°C
ENG
IN
3:22 PM
11/17/2022

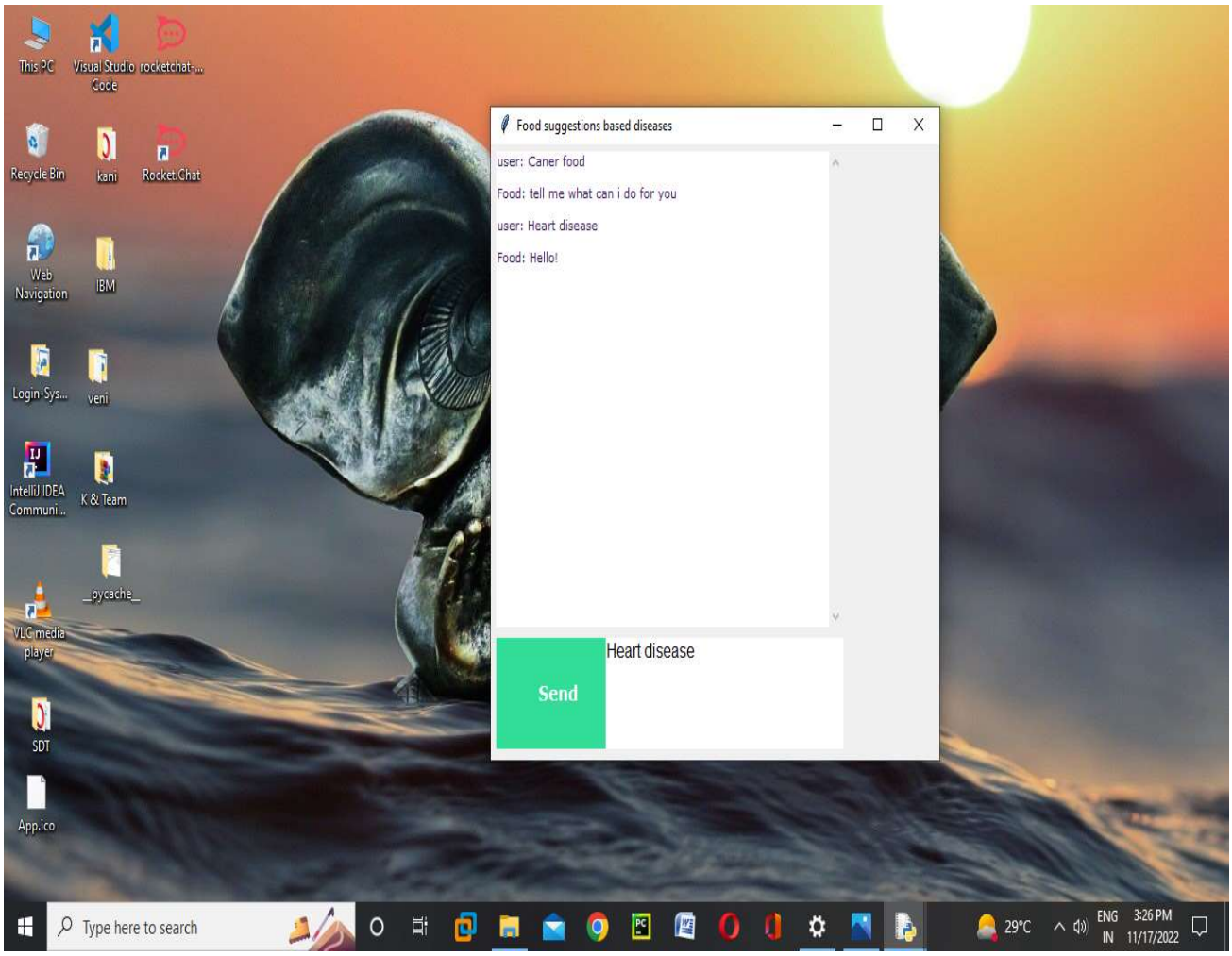

AI-Powered Nutrition Analyzer for Fitness Enthusiasts
Age
20
veg/Non veg (1/0)
0
Weight (in kg)
45
Height (in cm)
165
Quit
Weight Loss
Weight Gain
Healthy
Food suggestions based diseases
Type here to search
29°C
ENG
IN
3:24 PM
11/17/2022


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\Kamesh\Desktop\veni\App.py ==========
Warning (from warnings module):
File "C:\Users\Kamesh\Desktop\veni\App.py", line 762
img = ImageTk.PhotoImage(Image.open(IMAGE_PATH).resize((WIDTH, HEIGTH), Image.ANTIALIAS))
DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use Resampling.LANCZOS instead.
>>>
Age: 20
Veg-NonVeg: 0
Weight: 45 kg
Hight: 165 cm
Your body mass index is: 16.528925619834713
Acoording to your BMI, you are Underweight
####################
SUGGESTED FOOD ITEMS ::
Milk
Onions
Orange
Protein Powder
Pumpkin
Peproni Pizza
Sugar Doughnuts
Thank You for taking our recommendations. :)
This PC
Visual Studio Code
rocketchat-...
Recycle Bin
kani
Rocket.Chat
Web Navigation
IBM
Login-Sys...
veni
IntelliJ IDEA Communi...
K & Team
__pycache__
VLC media player
SDT
App.ico
Type here to search
29°C
ENG IN
3:24 PM
11/17/2022


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Sugar Doughnuts
Thank You for taking our recommendations. :)
Age: 20
Veg-NonVeg: 0
Weight: 45 kg
Hight: 165 cm
Your body mass index is: 16.528925619834713
Acoording to your BMI, you are Underweight
####################
SUGGESTED FOOD ITEMS ::
Avocados
Bananas
Bagels made in wheat
Brown Rice
Cauliflower
Dark chocolates
Grapes
Milk
Cashew Nuts
Onions
Pears
Peas
Protein Powder
Pumpkin
Tuna Salad
Cheese Pizza
Cheese Burger
Chicken Sandwich
Sugar Doughnuts
Pop Corn - Caramel
Poha
Chappati
Tomato
Brownie
Thank You for taking our recommendations. :)
This PC
Visual Studio Code
rocketchat-...
Recycle Bin
kani
Rocket.Chat
Web Navigation
IBM
Login-Sys...
veni
IntelliJ IDEA Communi...
K & Team
__pycache__
VLC media player
SDT
App.ico
Type here to search
29°C
ENG IN
3:25 PM
11/17/2022


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Milk
Cashew Nuts
Onions
Pears
Peas
Protein Powder
Pumpkin
Tuna Salad
Cheese Pizza
Cheese Burger
Chicken Sandwich
Sugar Doughnuts
Pop Corn - Caramel
Poha
Chappati
Tomato
Brownie
Thank You for taking our recommendations. :)
Age: 20
Veg-NonVeg: 0
Weight: 45 kg
Hight: 165 cm
Your body mass index is: 16.528925619834713
Acoording to your BMI, you are Underweight
####################
SUGGESTED FOOD ITEMS ::
Cauliflower
Corn
Grapes
Pumpkin
Sugar Doughnuts
Poha
Tomato
Brownie
Thank You for taking our recommendations. :)
Type here to search
29°C
ENG IN
3:25 PM
11/17/2022


Food suggestions based diseases
user: Caner food
Food: tell me what can i do for you
Send
Caner food
Type here to search
29°C
ENG IN
3:26 PM
11/17/2022


This PC
Visual Studio Code
rocketchat-...
Recycle Bin
kani
Rocket.Chat
Web Navigation
IBM
Login-Sys...
veni
IntelliJ IDEA Communi...
K & Team
pycache
VLC media player
SDT
App.ico
Food suggestions based diseases
user: Caner food
Food: tell me what can i do for you
user: Heart disease
Food: Hello!
Send
Heart disease
Type here to search
29°C
ENG
IN
3:26 PM
11/17/2022

13 . APPENDEX

SOURCE CODE :

GitHub & PROJECT DEMO LINK:

https://github.com/IBM-EPBL/IBM-Project-31971-1660207125

PROJECT VIDEO LINK :

https://youtu.be/Nr4tQjDGJPw