A PROJECT REPORT ON

TYPE 1 DIABETES INSULIN PREDICTION

A project report submitted in fulfillment of the 2 month Internship

Under

National Institue Of Technology, Warangal

Project Submitted By

Rohit Kumar Pali

Under The Guidance Of

Mentors: Priyanka Chawla

Declaration of Authorship

We hereby declare that this thesis titled" Type 1 Diabetes Insulin Prediction" and the work presented by the undersigned candidate, as part of 2 Month Internship. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Rohit Kumar Pali
Thesis Title: Type 1 Diabetes Insulin Prediction

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled

“ Type 1 Diabetes Insulin Prediction”

Prepared under my supervision and guidance by Rohit Kumar Pali be accepted in fulfilment ofthe requirement for awarding the 2 Month Internship Under the National Institute Of Technology Warangal. The project, in our opinion, is worthy for its acceptance.

Mentor : Priyanka Chawla

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. I, Rohit Kumar Pali student of Bhilai Institute Of Technology (C.G), I am extremely grateful to my mentor from the National Institute Of Technology Warangal during my 2 months internship for the confidence bestowed in me and entrusting my project entitled
“Type 1 Diabetes Insulin Prediction” with specialreference.

I express my sincere thanks to Mentor: Priyanka Chawla of NIT Warangal for making the resources available at right time and providing valuable insightsleading to the successful completion of our project who even assisted me in completing the project.

Name: Rohit Kumar Pali

Contents

ABSTRACT

Insulin pumps are devices used by people with diabetes to administer insulin. Theyoften come with a calculator that suggests the amount of insulin to take based on various factors like the amount of carbohydrates eaten. After the first insulin dose, there is still some insulin left in the body that needs to be accounted for to avoid taking too much insulin and causing low blood sugar levels. This leftover insulin is called Bolus Insulin on Board (BOB). To calculate the right amount of insulin to takeafter the first dose, the calculator needs to know how long insulin lasts in the body.

This is called Duration of Insulin Action (DIA). However, there is no agreed-upon DIA setting, and many people are using a setting that is too short. This can lead to takingtoo much insulin and experiencing low blood sugar levels. This article explains the problems caused by using a too-short DIA setting and suggests an appropriate DIA time for insulin pump users

Introduction

Insulin pumps are used by people with diabetes to maintain their blood sugar levels. Basal insulin is used to keep blood sugar levels steady during periods of fasting, while bolus insulin is used to counteract rises in blood sugar after meals. A pump's bolus calculator helps users determine the correct amount of bolus insulin needed based on their carbohydrate intake and other factors.

However, if the duration of insulin action is not accurately set in the calculator, it can result in insulin stacking, where too much insulin is administered, leading to hypoglycemia. Many users and healthcare providers do not understand the importance of setting the correct duration of insulin action, and the recommended time range of 3 to 5 hours may not be appropriate for everyone.

Insulin stacking is common, with many users taking multiple boluses within a short time frame, and this can have negative impacts on blood sugar control. Different pump manufacturers have their own unique bolus calculators, and there is a need formore research on how these calculators differ and how they can be improved. Bolus calculators are also being used in blood glucose systems and apps for people who use multiple daily injections.

Problem Statement

1. What is the problem all about ?

The problem revolves around the accurate setting of the duration of insulin action in insulin pumps and bolus calculators used by people with diabetes. Basal insulin maintains stable blood sugar levels during fasting, while bolus insulin is used to counteract post-meal blood sugar rises. The bolus calculator helps users determine the correct amount of bolus insulin based on factors such as carbohydrate intake. However, if the duration of insulin action is not set correctly, it can lead to insulin stacking, where multiple boluses are taken within a short time frame. This can result in excessive insulin administration, causing hypoglycemia (low blood sugar). Many users and healthcare providers may not fully understand the importance of setting the correct duration of insulin action, and the recommended time range of 3 to 5 hours may not be suitable for all individuals. Different pump manufacturers have their own unique bolus calculators, and more research is needed to understand the differences and improve their accuracy.

2. Why is this an important problem to solve?

Setting the correct duration of insulin action is crucial for people with diabetes using insulin pumps or bolus calculators to avoid insulin stacking and hypoglycemia. Hypoglycemia can lead to serious health complications and adversely affect a person's quality of life. Understanding the appropriate duration of insulin action is vital for healthcare providers to accurately dose insulin and prevent dangerous low blood sugar levels. Insulin stacking is a common problem and can have negative impacts on blood sugar control, which can increase the risk of complications and undermine diabetes management. Solving this problem can significantly improve blood sugar management and overall health for people with diabetes.

3.Business/Real-world impact of solving this problem ?

Solving the problem of accurately setting the duration of insulin action can have substantial real-world impact on diabetes management. By avoiding insulin stacking and preventing hypoglycemia, individuals with diabetes can achieve better blood sugar control, leading to improved health outcomes and a reduced risk of diabetes-related complications. This can enhance the quality of life for people with diabetes, allowing them to better manage their condition and reduce the burden of frequent blood sugar fluctuations. Improved bolus calculators and insulin pump settings can empower individuals with diabetes to take control of their insulin therapy and achieve more stable blood sugar levels, leading to better overall health and well-being. Additionally, better understanding and research in this area can lead to advancements in diabetes technology, making insulin pumps and bolus calculators more accurate and user-friendly.

Data Description:

1. *Source of the dataset*

   So, the source of the dataset is Kaggle..com

2. *Explanation of each feature and datapoint available*

   In the given dataset it has Age, Blood pressure , BMI , Blood glucose level , Heart rate, Genetics & Meal Carbohydrates

Machine Learning Model & Feature Engineering

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from google.colab import files
import io

data = files.upload()

<IPython.core.display.HTML object>

df=pd.read_csv('Type1Diabetes.csv')
df.head()

   Age  BloodPressure(120/80)   BMI  Blood Glucose Level(BGL)  Heart Rate  \
0   21                     66  28.1                        70          96
1   21                     50  23.0                        77          95
2   21                      0   0.0                        71          96
3   21                     55  19.1                        82          91
4   21                     82  24.7                        74         100

   Genetics (Y/N)  Meal Carbohydrates
0               1                   9
1               1                  10
2               1                  16
3               0                  20
4               1                  30

df.tail()

     Age  BloodPressure(120/80)   BMI  Blood Glucose Level(BGL)  Heart Rate  \
705   69                     80  26.8                        72          97
706   69                     82   0.0                        60         117
707   70                     82  32.5                        60         119
708   72                      0  19.6                        51         102
709   81                     74  25.9                        51         101

     Genetics (Y/N)  Meal Carbohydrates
705               1                  21
706               0                  54
707               0                  10
708               0                  10
709               0                  11

df.shape

(710, 7)

df.describe()
```

```
              Age  BloodPressure(120/80)         BMI  \
count  710.000000             710.000000  710.000000   
mean    33.140845              68.735211   31.879296   
std     11.770964              19.790305    8.002313   
min     21.000000               0.000000    0.000000   
25%     24.000000              62.500000   27.100000   
50%     29.000000              72.000000   32.000000   
75%     40.000000              80.000000   36.500000   
max     81.000000             122.000000   67.100000   

       Blood Glucose Level(BGL)  Heart Rate  Genetics (Y/N)  \
count                710.000000  710.000000      710.000000   
mean                  66.387324   99.454930        0.246479   
std                   16.159487   13.273198        0.431264   
min                   50.000000   78.000000        0.000000   
25%                   51.000000   89.000000        0.000000   
50%                   60.000000   98.000000        0.000000   
75%                   78.000000  108.000000        0.000000   
max                  100.000000  130.000000        1.000000   

       Meal Carbohydrates  
count          710.000000  
mean            16.771831  
std             13.484831  
min              2.000000  
25%              7.250000  
50%             12.000000  
75%             21.000000  
max             74.000000  
```

Now In X we have Patient features & In Y we have 'age'

```
x=df.drop(columns='Age')
```

```
x
```

```
     BloodPressure(120/80)   BMI  Blood Glucose Level(BGL)  Heart Rate  \
0                       66  28.1                        70          96   
1                       50  23.0                        77          95   
2                        0   0.0                        71          96   
3                       55  19.1                        82          91   
4                       82  24.7                        74         100   
..                     ...   ...                       ...         ...   
705                     80  26.8                        72          97   
706                     82   0.0                        60         117   
707                     82  32.5                        60         119   
708                      0  19.6                        51         102   
709                     74  25.9                        51         101   

     Genetics (Y/N)  Meal Carbohydrates  
0                 1                   9  
1                 1                  10  
2                 1                  16  
3                 0                  20  
4                 1                  30  
..              ...                 ...  
705               1                  21  
706               0                  54  
```

```
707                  0                    10
708                  0                    10
709                  0                    11

[710 rows x 6 columns]
```

```
y=df['Age']
```

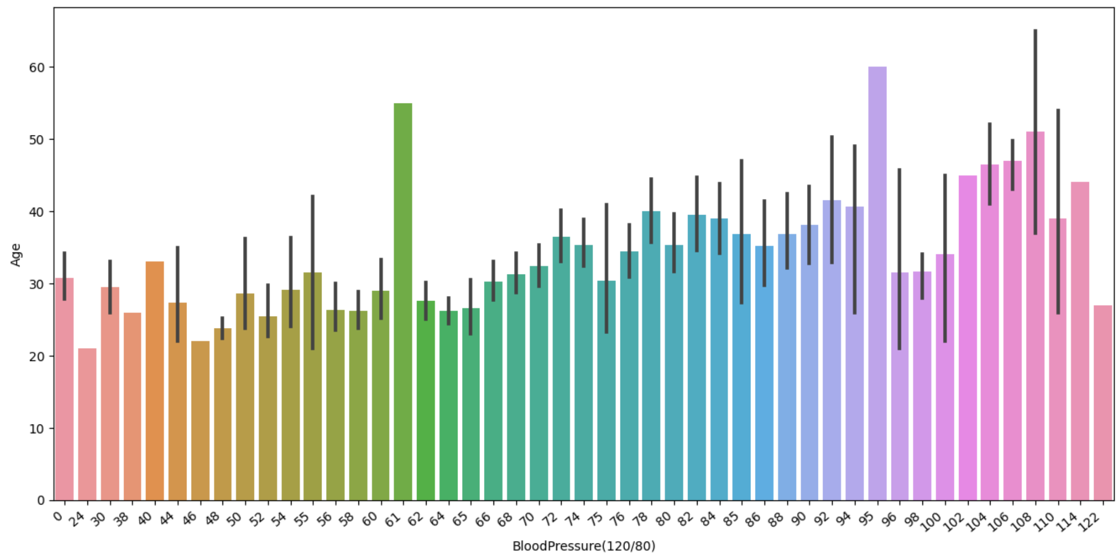
```
y
```

```
0      21
1      21
2      21
3      21
4      21
       ..
705    69
706    69
707    70
708    72
709    81
Name: Age, Length: 710, dtype: int64
```

Now we use seaborn to represent the given data in a graphical manner

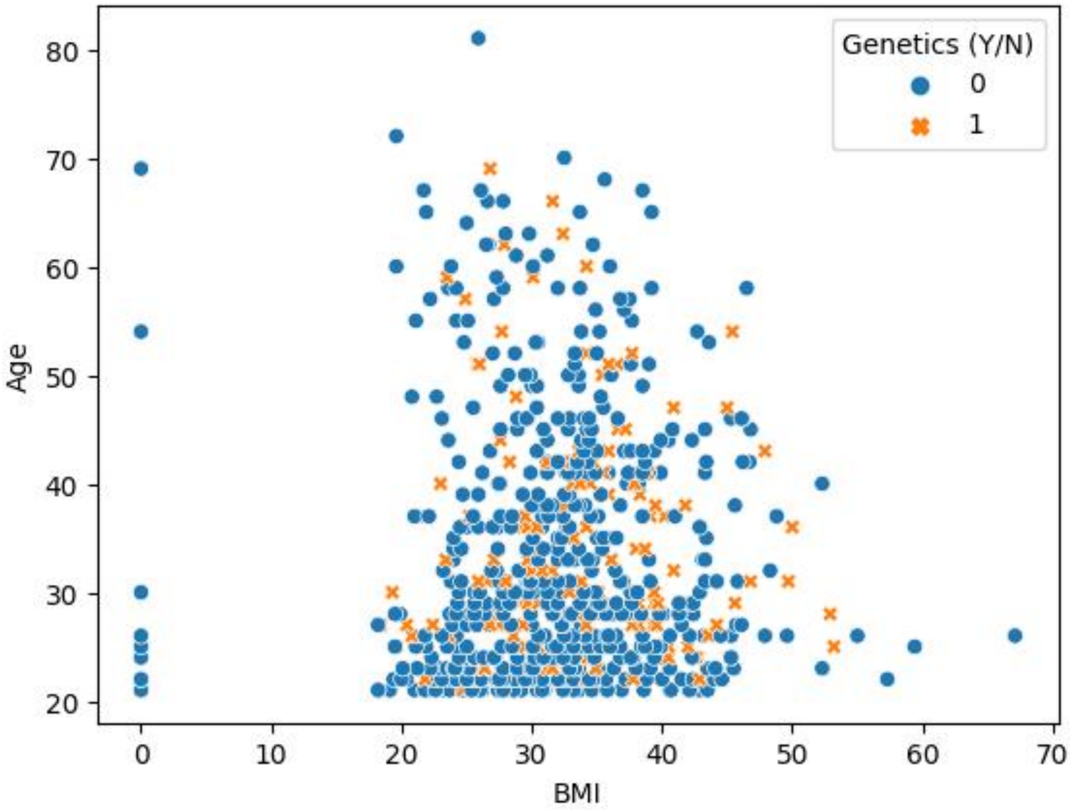
Relationship Between Of Blood Pressure & Age

```
import seaborn as sns
plt.subplots(figsize=(15,7))
ax = sns.barplot(x='BloodPressure(120/80)', y='Age',data=df)
ax.set_xticklabels(ax.get_xticklabels(),rotation=40,ha='right')
plt.show()
```



Relationship Of Fueltype & Price BMI & AGE

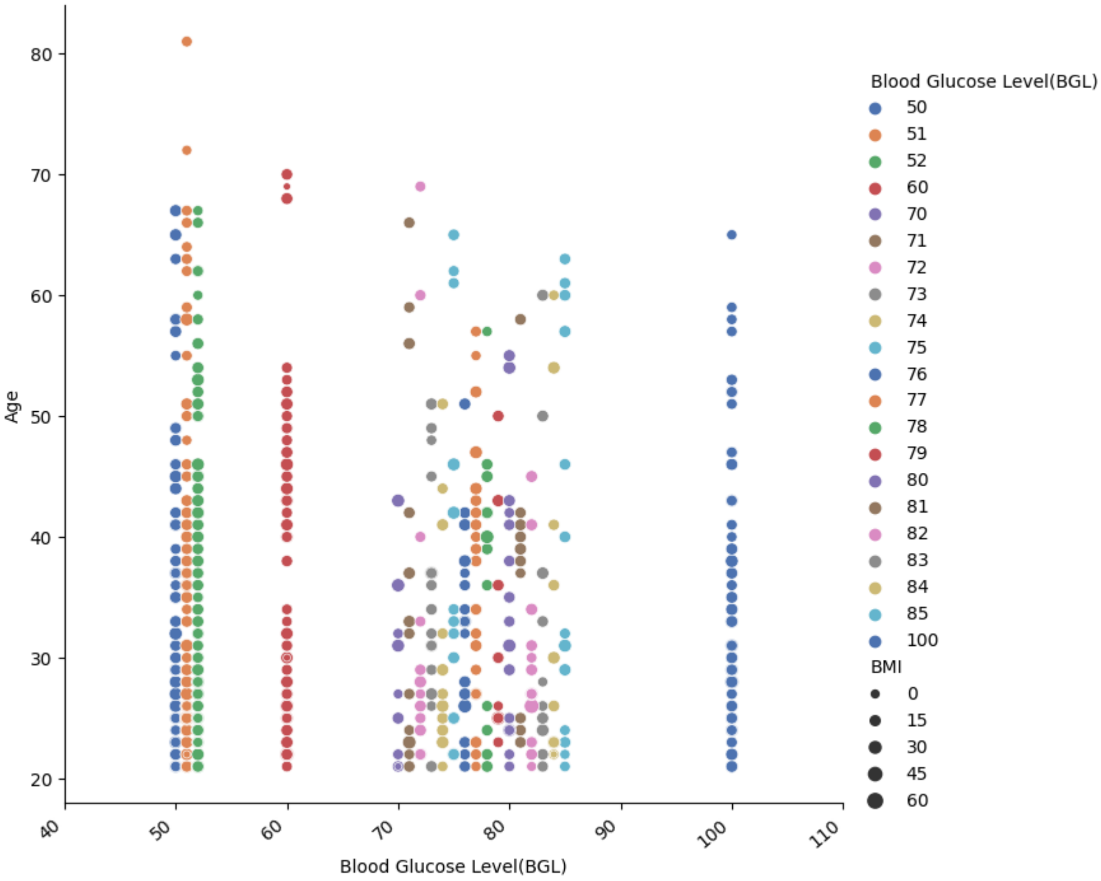
```
sns.scatterplot(x="BMI",y="Age",hue="Genetics (Y/N)",style="Genetics
(Y/N)",data=df)
plt.show()
```



Now checking relationship between Blood Glucose Level(BGL) , BMI & Age

```
ax=sns.relplot(x='Blood Glucose Level(BGL)',y='Age',data=df,hue='Blood Glucose
Level(BGL)',size='BMI',height=7,aspect=1 , palette="deep")
ax.set_xticklabels(rotation=40,ha='right')
```

```
<seaborn.axisgrid.FacetGrid at 0x7efc8c782ad0>
```



```
x

     BloodPressure(120/80)   BMI  Blood Glucose Level(BGL)  Heart Rate  \
0                       66  28.1                        70          96
1                       50  23.0                        77          95
2                        0   0.0                        71          96
3                       55  19.1                        82          91
4                       82  24.7                        74         100
..                     ...   ...                       ...         ...
705                     80  26.8                        72          97
706                     82   0.0                        60         117
707                     82  32.5                        60         119
708                      0  19.6                        51         102
709                     74  25.9                        51         101

     Genetics (Y/N)  Meal Carbohydrates
0                 1                   9
1                 1                  10
2                 1                  16
3                 0                  20
4                 1                  30
..              ...                 ...
705               1                  21
706               0                  54
707               0                  10
708               0                  10
709               0                  11

[710 rows x 6 columns]

y

0      21
1      21
2      21
3      21
4      21
       ..
705    69
706    69
707    70
708    72
709    81
Name: Age, Length: 710, dtype: int64
```

```python
# Additional
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import pickle
import numpy as np

# Splitting the data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

# Training the Linear Regression model
lr = LinearRegression()
lr.fit(x_train, y_train)
```

```
# Accessing the coefficients and intercept
weights = lr.coef_
intercept = lr.intercept_

# Storing the model
my_model = {'weights': weights, 'intercept': intercept}
pickle.dump(my_model, open('Type11final.pkl', 'wb'))

# Loading the model
loaded_model = pickle.load(open('Type11final.pkl', 'rb'))

# Making predictions on the test set
y_pred = lr.predict(x_test)

# Evaluating model performance
r2 = r2_score(y_test, y_pred)

# Printing the R-squared score
print("R-squared score:", r2)
```

```
R-squared score: 0.036795127911008003
```

Now we have to use the train_test_split & linear regression

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)
```

Here r2_score will measures the linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
```

Now we have to train linear regression model

```
lr=LinearRegression()
```

Now the input data is given to the pipeline first it goes to the column transformer then OHE will perform then it transform the given column.Now all the column & transform column will be given to the linear regresion

*lr.coef_ , lr.intercept_*

```
pipe=make_pipeline(lr)
```

```
pipe.fit(x_train,y_train)
```

```
Pipeline(steps=[('linearregression', LinearRegression())])
```

```
lr.coef_
```

```
array([ 1.40906216e-01, -4.20519286e-02, -2.97459665e-03,  7.11597576e-04,
       -1.33392430e+00, -4.17764320e-02])
```

```
lr.intercept_
```

```
26.143401387964428
```

```
my_model={'weight':lr.coef_,'intercept':lr.intercept_}
```

```
my_model
```

```
{'weight': array([ 1.40906216e-01, -4.20519286e-02, -2.97459665e-03,
7.11597576e-04,
        -1.33392430e+00, -4.17764320e-02]),
 'intercept': 26.143401387964428}
```

```
import pickle
```

```
pickle.dump(my_model,open('Type1final.pkl','wb'))
```

```
pickle.load(open('Type1final.pkl','rb'))
```

```
{'weight': array([ 1.40906216e-01, -4.20519286e-02, -2.97459665e-03,
7.11597576e-04,
        -1.33392430e+00, -4.17764320e-02]),
 'intercept': 26.143401387964428}
```

y = w1x1+w2x2+w3x3+…+intercept

Now we have to predict & store in y

```
y_pred=pipe.predict(x_test)
```

```
y_pred
```

```
array([31.78471974, 33.33002739, 35.14363645, 36.33669446, 33.15698045,
       32.54313615, 32.91346448, 35.55264754, 34.68977771, 33.37406002,
       35.44871226, 34.75736582, 36.1245575 , 34.37525371, 33.02794522,
       33.37546723, 34.53272889, 34.79913144, 34.17239596, 33.76941709,
       33.15950852, 32.7841539 , 33.6672002 , 30.36534812, 32.46318595,
       32.81841093, 36.83707763, 32.71294389, 33.38639063, 32.55745801,
       31.78848079, 32.12057111, 35.21608367, 37.97731549, 36.20523673,
       34.80604594, 37.41313233, 32.57540963, 36.07695003, 33.09029518,
       33.13217321, 31.86519708, 34.9395846 , 35.23845355, 32.43211341,
       34.08201944, 30.50457122, 31.64319519, 32.61609832, 31.32946279,
       33.81350318, 27.69022313, 33.30886019, 30.18550103, 33.44283811,
       34.77148268, 32.33501564, 32.77578514, 34.00536926, 33.49646158,
       34.33311038, 33.69170501, 33.15583375, 36.07176132, 35.29469377,
       35.90457534, 34.22338161, 33.61012194, 29.65545854, 34.43880721,
       35.89788439, 34.43054633, 33.79571801, 34.17810874, 34.38925307,
       36.82637789, 33.22256404, 32.75117149, 32.84317416, 33.36040658,
       31.77603459, 29.66596837, 33.78348612, 34.59315272, 34.67984617,
       34.2012118 , 35.03125039, 30.73695796, 35.51223982, 35.18616728,
       33.22918067, 31.3790839 , 31.24135674, 32.83386258, 35.61278067,
       33.26151163, 34.03393378, 33.62158735, 36.08116229, 31.8606716 ,
       33.78308324, 35.41298518, 37.50525939, 35.14828226, 31.03653383,
       35.17020348, 33.71553481, 34.04869404, 31.10834492, 35.74954524,
       35.33499377, 33.2247914 , 30.10769291, 31.23933245, 33.08633485,
       34.72665193, 36.14814497, 23.49300966, 33.36916315, 32.35485063,
       35.59235068, 33.20809292, 32.84006254, 35.35242243, 31.96683181,
       35.19516871, 37.00915598, 23.58061217, 34.16996258, 32.55474374,
       34.18096306, 34.47974848, 31.83287256, 35.05152803, 32.5907088 ,
       31.76814619, 34.23168076, 34.5424672 , 33.08135099, 35.78266567,
       33.66076657, 33.94578909])
```

Now we have to see the r2_score //r2_score = measure linear regression

```
r2_score(y_test,y_pred)
```

```
0.022434484958555156
```

In Different train and test split data we get different r2 score Then we perform random r2 score in train and test split data & choose the data which has maximum score.

```
for i in range(10):
  x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2)
  lr=LinearRegression()
  pipe=make_pipeline(lr)
  pipe.fit(x_train,y_train)
  y_pred=pipe.predict(x_test)
  print(r2_score(y_test,y_pred), i)
```

```
0.03481327936377243 0
0.04845739259447712 1
0.012767368759200926 2
0.03168852729425842 3
0.037671220974441755 4
0.03756565580526283 5
0.064859294352827 6
0.07535193312507249 7
0.02421636368169411 8
0.05638514017702401 9
```

Now We use random_state=i to getting the highest value. After that we have to store the value to storing the value we use score & then append the value

```
score=[]
for i in range(1000):
  x_train,x_test,y_train,y_test=train_test_split(x,y,
test_size=0.2,random_state=i)
  lr=LinearRegression()
  pipe=make_pipeline(lr)
  pipe.fit(x_train,y_train)
  y_pred=pipe.predict(x_test)
  score.append(r2_score(y_test,y_pred))
```

Now we have to check the greatest r2 score

```
np.argmax(score)
```

```
48
```

Checking greatest Score

```
score[np.argmax(score)]
```

```
0.93701575849099512
```

Checking random r2 score

```
x_train,x_test,y_train,y_test=train_test_split(x,y,
test_size=0.2,random_state=np.argmax(score))
lr=LinearRegression()
pipe=make_pipeline(lr)
pipe.fit(x_train,y_train)
y_pred=pipe.predict(x_test)
r2_score(y_test,y_pred)
```

```
0.93701575849099512
```

Now we have to import pickle

```python
import pickle
```

```python
pickle.dump(pipe,open('Type1.pkl','wb'))
```

```python
pipe
```

```
Pipeline(steps=[('linearregression', LinearRegression())])
```

Now We Have To Predict The Machine Learning Model

```python
x
```

```
     BloodPressure(120/80)   BMI  Blood Glucose Level(BGL)  Heart Rate  \
0                       66  28.1                        70          96   
1                       50  23.0                        77          95   
2                        0   0.0                        71          96   
3                       55  19.1                        82          91   
4                       82  24.7                        74         100   
..                     ...   ...                       ...         ...   
705                     80  26.8                        72          97   
706                     82   0.0                        60         117   
707                     82  32.5                        60         119   
708                      0  19.6                        51         102   
709                     74  25.9                        51         101   

     Genetics (Y/N)  Meal Carbohydrates  
0                 1                   9  
1                 1                  10  
2                 1                  16  
3                 0                  20  
4                 1                  30  
..              ...                 ...  
705               1                  21  
706               0                  54  
707               0                  10  
708               0                  10  
709               0                  11  

[710 rows x 6 columns]
```

```python
pipe.predict(pd.DataFrame([[82,32.7,74,100,1,30]]
                          ,columns=['BloodPressure(120/80)','BMI','Blood Glucose Level(BGL)','Heart Rate','Genetics (Y/N)','Meal Carbohydrates']))
```

```
array([34.47463796])
```

Thank You

Now model is found

Deployment and productionization

The final phase of this project.is Deployment, here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.

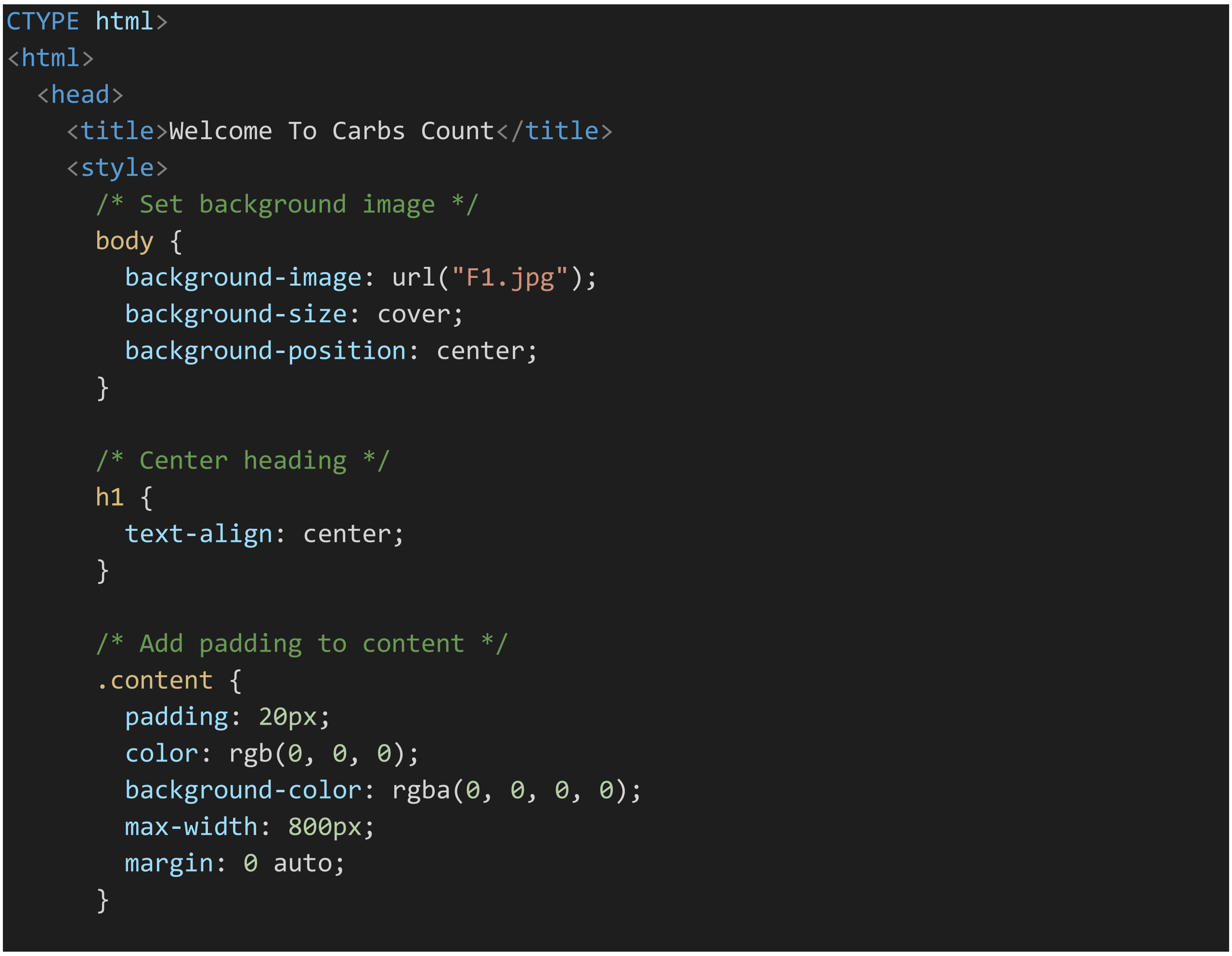
In this final stage, we need to deploy this machine learning pipeline to put of researchavailable to end users for use. The model will be deployed in real-world scenario where it takes continuous raw input from the real world and predict the output.

Prediction Page Building:

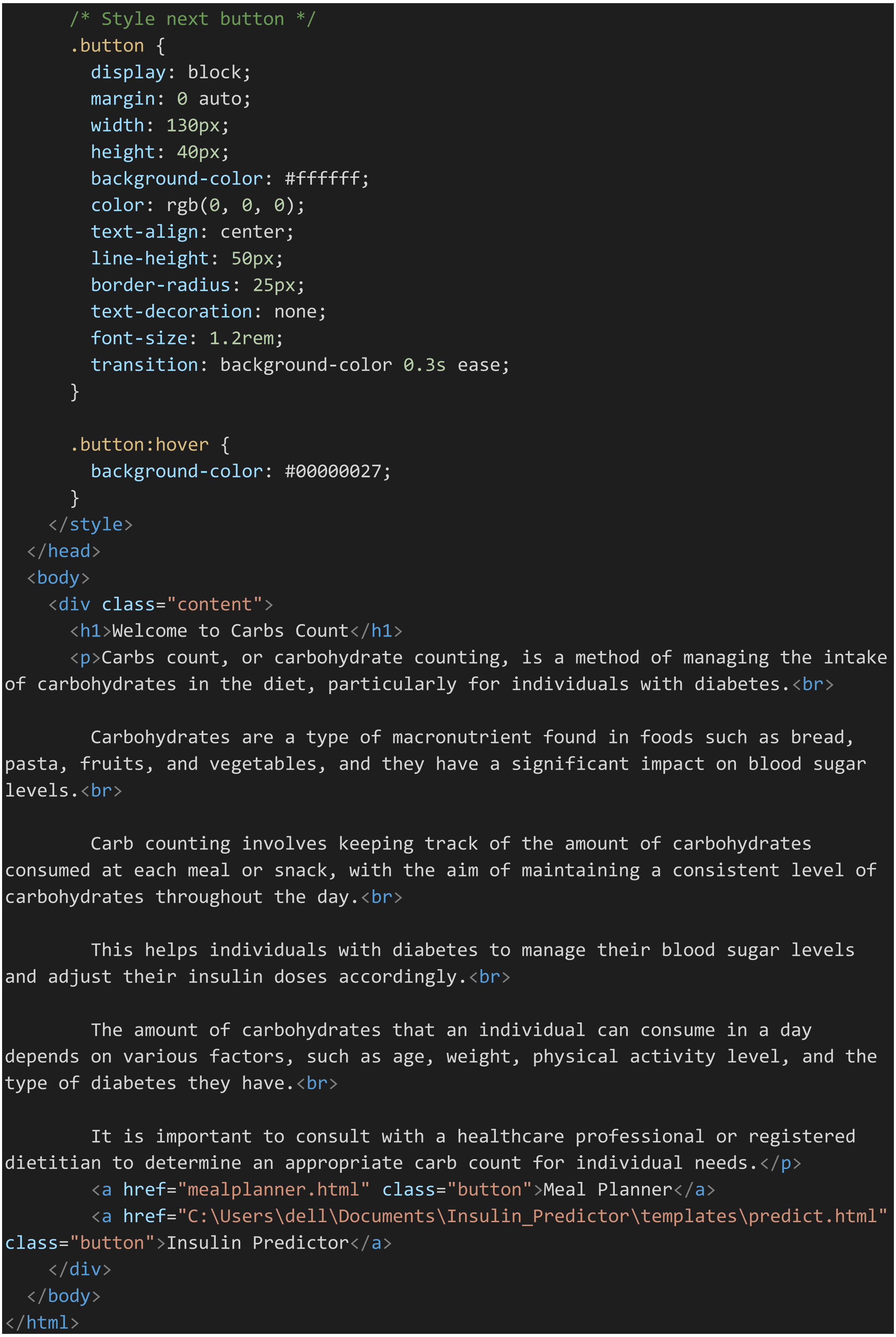
In this step the tool we are using to build our prediction page & other pages.

CODE :

Welcome Page :

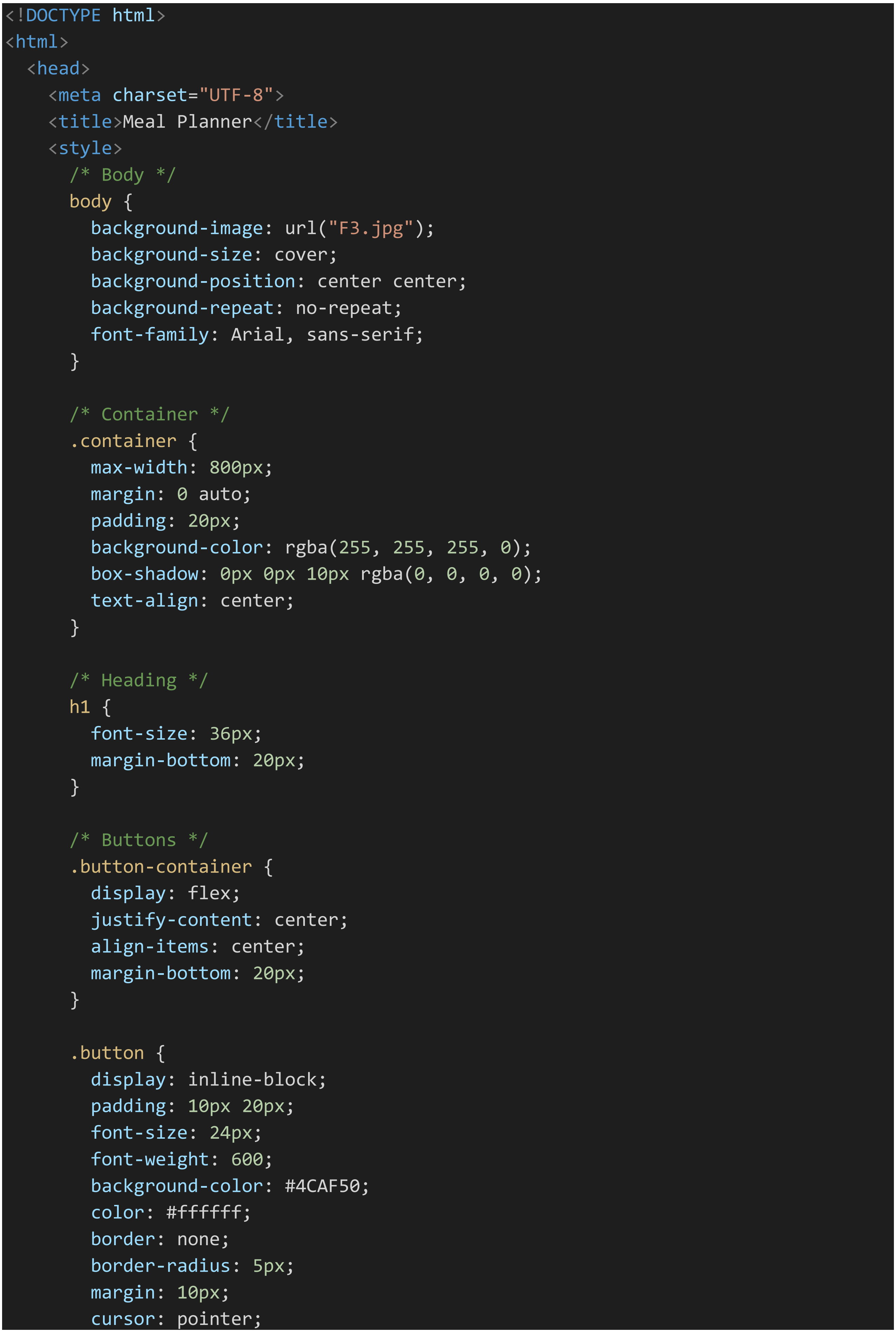


```
CTYPE html>
<html>
  <head>
    <title>Welcome To Carbs Count</title>
    <style>
      /* Set background image */
      body {
        background-image: url("F1.jpg");
        background-size: cover;
        background-position: center;
      }

      /* Center heading */
      h1 {
        text-align: center;
      }

      /* Add padding to content */
      .content {
        padding: 20px;
        color: rgb(0, 0, 0);
        background-color: rgba(0, 0, 0, 0);
        max-width: 800px;
        margin: 0 auto;
      }
```

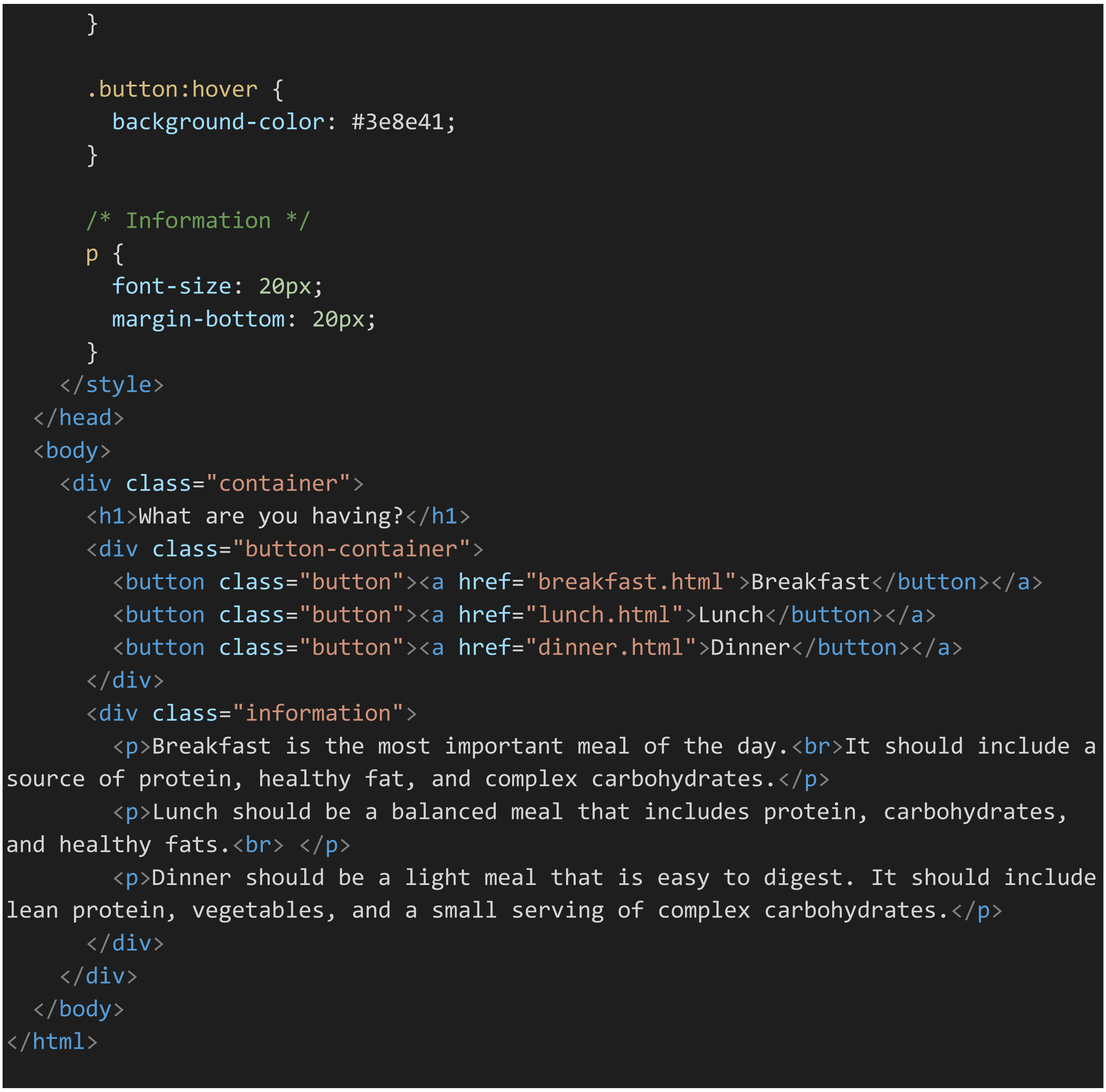


```
      /* Style next button */
      .button {
        display: block;
        margin: 0 auto;
        width: 130px;
        height: 40px;
        background-color: #ffffff;
        color: rgb(0, 0, 0);
        text-align: center;
        line-height: 50px;
        border-radius: 25px;
        text-decoration: none;
        font-size: 1.2rem;
        transition: background-color 0.3s ease;
      }

      .button:hover {
        background-color: #00000027;
      }
    </style>
  </head>
  <body>
    <div class="content">
      <h1>Welcome to Carbs Count</h1>
      <p>Carbs count, or carbohydrate counting, is a method of managing the intake
of carbohydrates in the diet, particularly for individuals with diabetes.<br>

        Carbohydrates are a type of macronutrient found in foods such as bread,
pasta, fruits, and vegetables, and they have a significant impact on blood sugar
levels.<br>

        Carb counting involves keeping track of the amount of carbohydrates
consumed at each meal or snack, with the aim of maintaining a consistent level of
carbohydrates throughout the day.<br>

        This helps individuals with diabetes to manage their blood sugar levels
and adjust their insulin doses accordingly.<br>

        The amount of carbohydrates that an individual can consume in a day
depends on various factors, such as age, weight, physical activity level, and the
type of diabetes they have.<br>

        It is important to consult with a healthcare professional or registered
dietitian to determine an appropriate carb count for individual needs.</p>
        <a href="mealplanner.html" class="button">Meal Planner</a>
        <a href="C:\Users\dell\Documents\Insulin_Predictor\templates\predict.html"
class="button">Insulin Predictor</a>
    </div>
  </body>
</html>
```

Meal Planner :



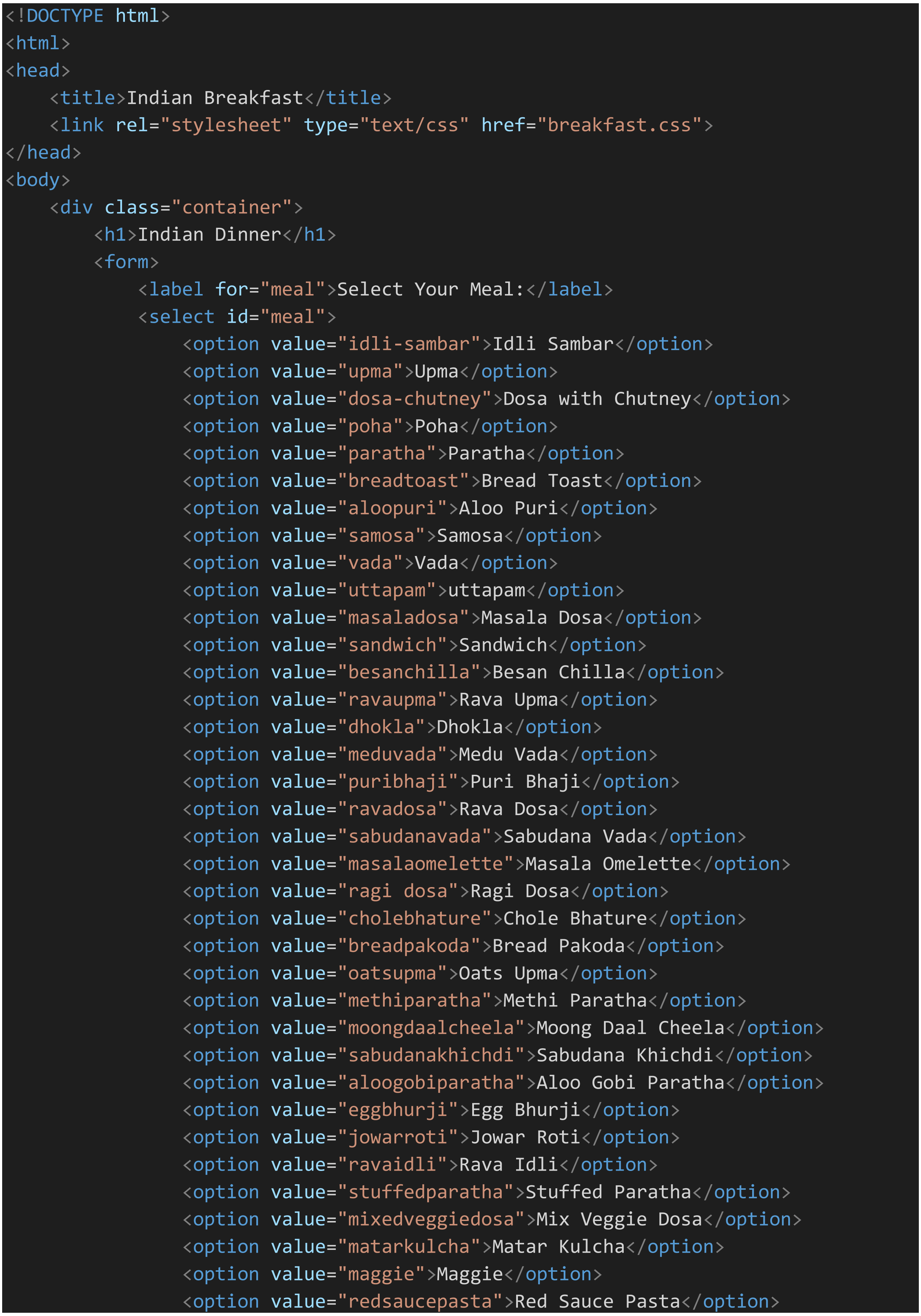
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Meal Planner</title>
    <style>
      /* Body */
      body {
        background-image: url("F3.jpg");
        background-size: cover;
        background-position: center center;
        background-repeat: no-repeat;
        font-family: Arial, sans-serif;
      }

      /* Container */
      .container {
        max-width: 800px;
        margin: 0 auto;
        padding: 20px;
        background-color: rgba(255, 255, 255, 0);
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0);
        text-align: center;
      }

      /* Heading */
      h1 {
        font-size: 36px;
        margin-bottom: 20px;
      }

      /* Buttons */
      .button-container {
        display: flex;
        justify-content: center;
        align-items: center;
        margin-bottom: 20px;
      }

      .button {
        display: inline-block;
        padding: 10px 20px;
        font-size: 24px;
        font-weight: 600;
        background-color: #4CAF50;
        color: #ffffff;
        border: none;
        border-radius: 5px;
        margin: 10px;
        cursor: pointer;
```



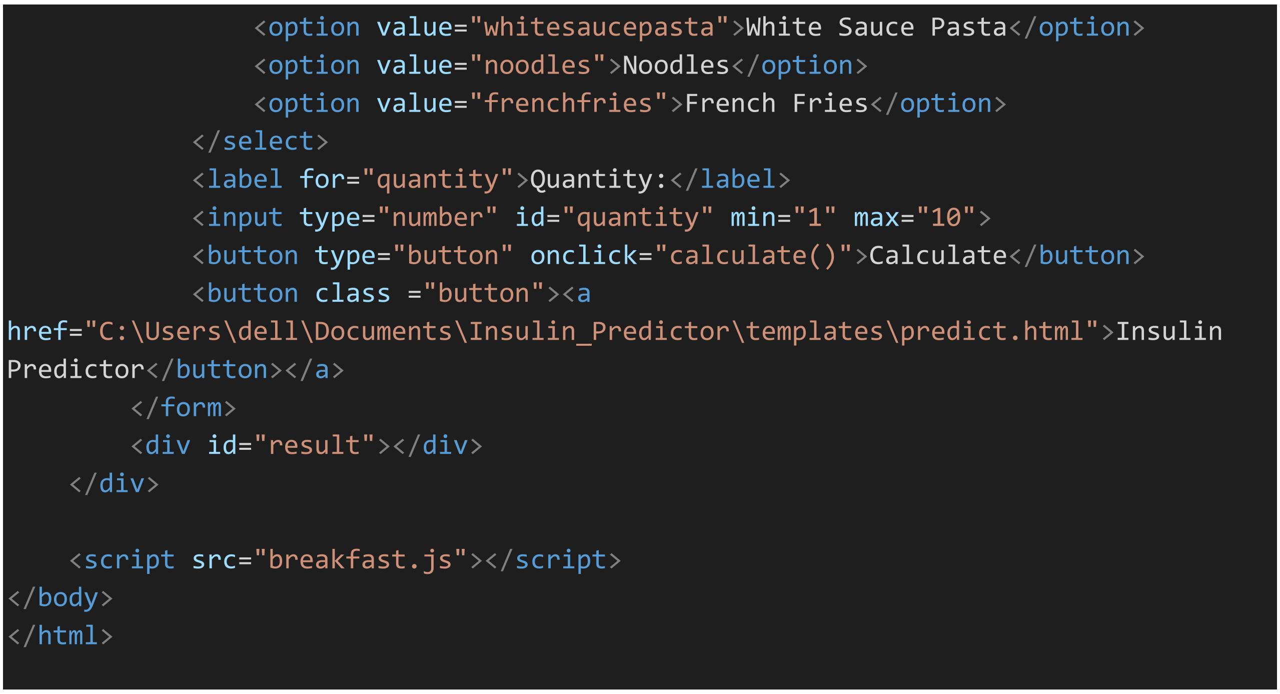
```
      }

      .button:hover {
        background-color: #3e8e41;
      }

      /* Information */
      p {
        font-size: 20px;
        margin-bottom: 20px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h1>What are you having?</h1>
      <div class="button-container">
        <button class="button"><a href="breakfast.html">Breakfast</button></a>
        <button class="button"><a href="lunch.html">Lunch</button></a>
        <button class="button"><a href="dinner.html">Dinner</button></a>
      </div>
      <div class="information">
        <p>Breakfast is the most important meal of the day.<br>It should include a
source of protein, healthy fat, and complex carbohydrates.</p>
        <p>Lunch should be a balanced meal that includes protein, carbohydrates,
and healthy fats.<br> </p>
        <p>Dinner should be a light meal that is easy to digest. It should include
lean protein, vegetables, and a small serving of complex carbohydrates.</p>
      </div>
    </div>
  </body>
</html>
```

Indian Breakfast :

HTML

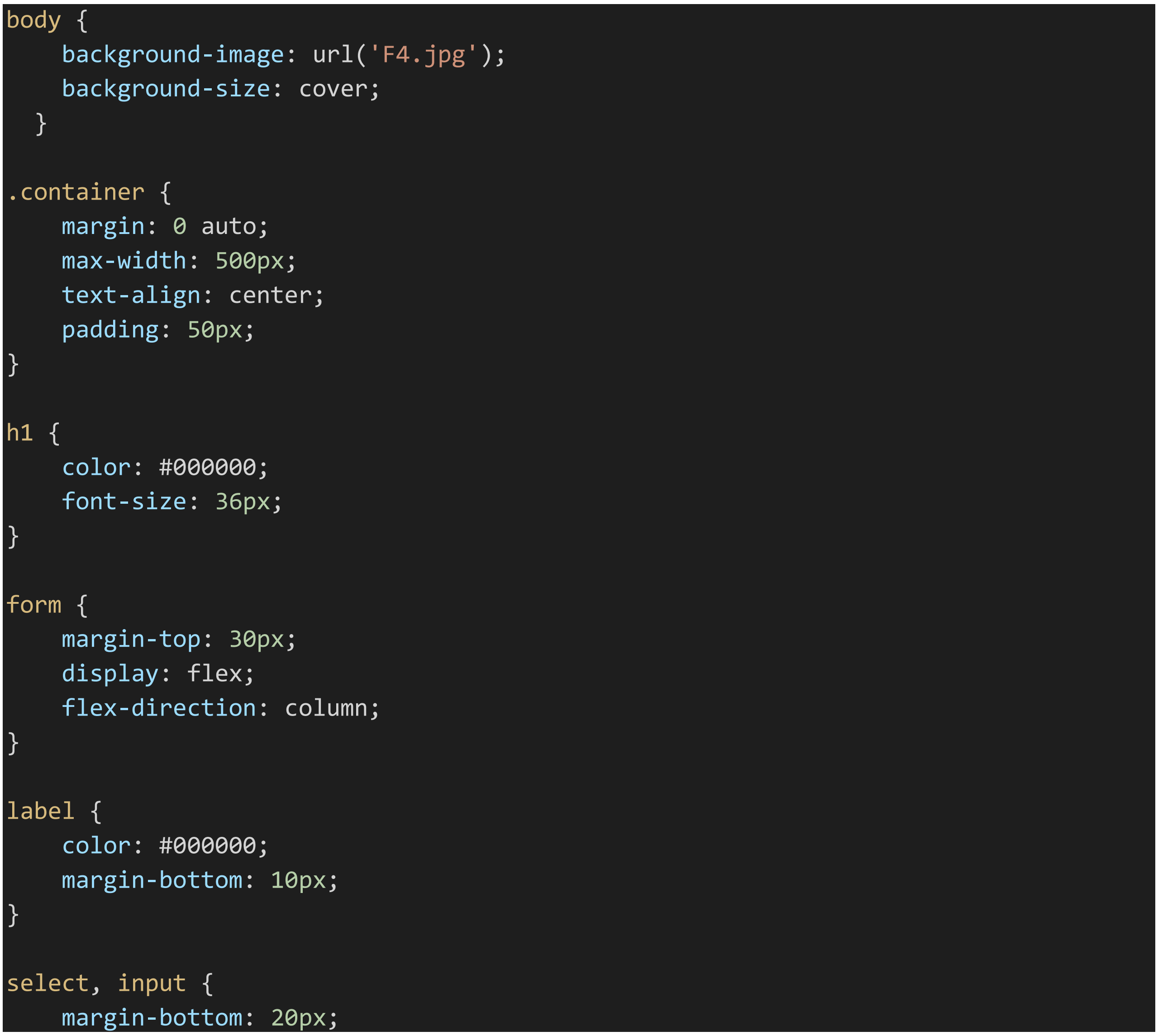


```
<!DOCTYPE html>
<html>
<head>
    <title>Indian Breakfast</title>
    <link rel="stylesheet" type="text/css" href="breakfast.css">
</head>
<body>
    <div class="container">
        <h1>Indian Dinner</h1>
        <form>
            <label for="meal">Select Your Meal:</label>
            <select id="meal">
                <option value="idli-sambar">Idli Sambar</option>
                <option value="upma">Upma</option>
                <option value="dosa-chutney">Dosa with Chutney</option>
                <option value="poha">Poha</option>
                <option value="paratha">Paratha</option>
                <option value="breadtoast">Bread Toast</option>
                <option value="aloopuri">Aloo Puri</option>
                <option value="samosa">Samosa</option>
                <option value="vada">Vada</option>
                <option value="uttapam">uttapam</option>
                <option value="masaladosa">Masala Dosa</option>
                <option value="sandwich">Sandwich</option>
                <option value="besanchilla">Besan Chilla</option>
                <option value="ravaupma">Rava Upma</option>
                <option value="dhokla">Dhokla</option>
                <option value="meduvada">Medu Vada</option>
                <option value="puribhaji">Puri Bhaji</option>
                <option value="ravadosa">Rava Dosa</option>
                <option value="sabudanavada">Sabudana Vada</option>
                <option value="masalaomelette">Masala Omelette</option>
                <option value="ragi dosa">Ragi Dosa</option>
                <option value="cholebhature">Chole Bhature</option>
                <option value="breadpakoda">Bread Pakoda</option>
                <option value="oatsupma">Oats Upma</option>
                <option value="methiparatha">Methi Paratha</option>
                <option value="moongdaalcheela">Moong Daal Cheela</option>
                <option value="sabudanakhichdi">Sabudana Khichdi</option>
                <option value="aloogobiparatha">Aloo Gobi Paratha</option>
                <option value="eggbhurji">Egg Bhurji</option>
                <option value="jowarroti">Jowar Roti</option>
                <option value="ravaidli">Rava Idli</option>
                <option value="stuffedparatha">Stuffed Paratha</option>
                <option value="mixedveggiedosa">Mix Veggie Dosa</option>
                <option value="matarkulcha">Matar Kulcha</option>
                <option value="maggie">Maggie</option>
                <option value="redsaucepasta">Red Sauce Pasta</option>
```



```html
                <option value="whitesaucepasta">White Sauce Pasta</option>
                <option value="noodles">Noodles</option>
                <option value="frenchfries">French Fries</option>
            </select>
            <label for="quantity">Quantity:</label>
            <input type="number" id="quantity" min="1" max="10">
            <button type="button" onclick="calculate()">Calculate</button>
            <button class ="button"><a
href="C:\Users\dell\Documents\Insulin_Predictor\templates\predict.html">Insulin
Predictor</button></a>
        </form>
        <div id="result"></div>
    </div>

    <script src="breakfast.js"></script>
</body>
</html>
```

CSS



```css
body {
    background-image: url('F4.jpg');
    background-size: cover;
  }

.container {
    margin: 0 auto;
    max-width: 500px;
    text-align: center;
    padding: 50px;
}

h1 {
    color: #000000;
    font-size: 36px;
}

form {
    margin-top: 30px;
    display: flex;
    flex-direction: column;
}

label {
    color: #000000;
    margin-bottom: 10px;
}

select, input {
    margin-bottom: 20px;
    padding: 5px;
}

button {
    background-color: #ffffff;
    color: #000;
    border: none;
    padding: 10px;
    margin-top: 20px;
    cursor: pointer;
}
a{
    text-decoration: none;
    color: #000;
}

#result {
    margin-top: 30px;
    color: #000000;
    font-size: 24px;
}
```

Java Script

```
function calculate() {
    // Get the selected meal from the dropdown list
    var meal = document.getElementById("meal").value;

    // Get the quantity entered by the user
    var quantity = document.getElementById("quantity").value;

    // Define a dictionary of meal names and their corresponding calorie counts
    var calorieDict = {
        "idli-sambar": 30,
        "upma": 30,
        "dosa-chutney": 36,
        "poha": 26,
        "paratha": 35,
        "breadtoast": 24,
        "aloopuri": 50,
        "samosa": 20,
        "vada": 23,
        "uttapam": 22,
        "masaladosa": 56,
        "sandwich": 30,
        "besanchilla": 20,
        "ravaupma": 30,
        "dhokla": 20,
        "meduvada": 16,
        "puribhaji": 50,
        "ravadosa": 40,
        "sabudanavada": 20,
        "masalaomelette": 5,
        "ragi dosa" : 20 ,
        "cholebhature" : 50,
        "breadpakoda" : 15,
        "oatsupma" : 25,
        "methiparatha" : 30,
        "moongdaalcheela" : 20 ,
        "sabudanakhichdi" : 54 ,
        "aloogobiparatha" : 30 ,
        "eggbhurji" : 5 ,
        "jowarroti" : 20 ,
        "ravaidli" : 25 ,
        "stuffedparatha" : 30 ,
        "mixedveggiedosa" : 45 ,
        "matarkulcha" : 45 ,
        "maggie" : 15 ,
        "redsaucepasta" : 15 ,
        "whitesaucepasta" : 15 ,
        "noodles" : 15,
        "frenchfries" : 40 ,
    };

    // Define the insulin ratio for 1 unit of insulin per 15 grams of
carbohydrates
    var insulinRatio = 15;

    // Calculate the total calorie count for the selected meal and quantity
    var totalCalories = calorieDict[meal] * quantity;

    // Calculate the total amount of carbohydrates in the selected meal and
quantity
    var totalCarbs = totalCalories / 4; // 1 gram of carbs = 4 calories

    // Calculate the amount of insulin required to cover the total amount of
carbohydrates
    //var insulinDose = totalCarbs / insulinRatio;

    // Round the insulin dose to 2 decimal places
    //insulinDose = insulinDose.toFixed(2);

    // Display the total calorie count and insulin dose to the user
    var resultElement = document.getElementById("result");
    resultElement.innerHTML = "Total Calories: " + totalCalories + " Grams" ;//+
"<br>Insulin Dose: " + insulinDose + " units";
  }
```

Indian Lunch :

HTML

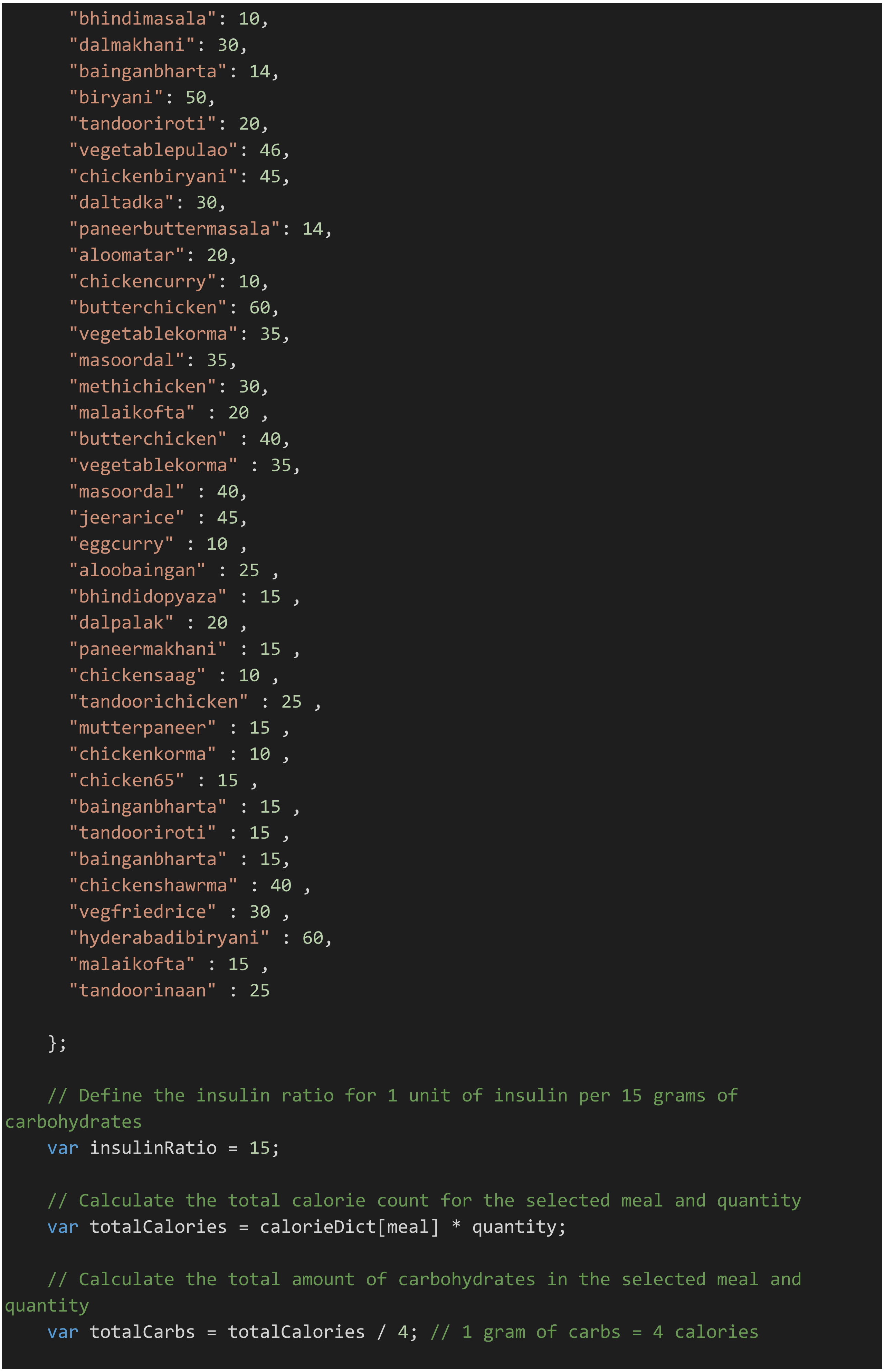
```
<!DOCTYPE html>
<html>
<head>
    <title>Indian Lunch</title>
    <link rel="stylesheet" type="text/css" href="lunch.css">
</head>
<body>
    <div class="container">
        <h1>Indian Lunch</h1>
        <form>
            <label for="meal">Select Your Meal:</label>
            <select id="meal">
                <option value="chanamasala">Chana Masala</option>
                <option value="rajmachawal">Rajma Chawal</option>
                <option value="aloogobi">Aloo Gobi</option>
                <option value="chickentikkamasala">Chicken Tikka Masala</option>
                <option value="palakpaneer">Palak Paneer</option>
                <option value="bhindimasala">Bhindi Masala</option>
                <option value="dalmakhani">Dal Makhani</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="biryani">Biryani</option>
                <option value="tandooriroti">Tandoori Roti</option>
                <option value="vegetablepulao">Vegetable Pulao</option>
                <option value="chickenbiryani">Chicken Biryani</option>
                <option value="daltadka">Dal Tadka</option>
                <option value="paneerbuttermasala">Paneer Butter Masala</option>
                <option value="aloomatar">Aloo Matar</option>
                <option value="chickencurry">Chicken Curry</option>
                <option value="butterchicken">Butter Chicken</option>
                <option value="vegetablekorma">Vegetable Korma</option>
                <option value="masoordal">Masoor Dal</option>
                <option value="methichicken">Methi Chicken</option>
                <option value="ragi dosa">Ragi Dosa</option>
                <option value="cholebhature">Chole Bhature</option>
                <option value="malaikofta">Malai Kofta</option>
                <option value="butterchicken">Butter Chicken</option>
                <option value="vegetablekorma">Vegetable Korma</option>
                <option value="masoordal">Masoor Dal</option>
                <option value="methichicken">Methi Chicken</option>
                <option value="jeerarice">Jeera Rice</option>
                <option value="eggcurry">Egg Curry</option>
                <option value="aloobaingan">Aloo Baingan</option>
                <option value="bhindidopyaza">Bhindi Do Pyaza</option>
                <option value="dalpalak">Dal Palak</option>
                <option value="paneermakhani">Paneer Makhani</option>
                <option value="chickensaag">Chicken Saag</option>
                <option value="tandoorichicken">Tandoori Chicken</option>
                <option value="mutterpaneer">Mutter Paneer</option>
                <option value="chickenkorma">Chicken Korma</option>
                <option value="chicken65">Chicken 65</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="tandooriroti">Tandoori Roti</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="chickenshawrma">Chicken Shawrma</option>
                <option value="vegfriedrice">Veg Fried Rice</option>
                <option value="hyderabadibiryani">Hyderabadi Biryani</option>
                <option value="malaikofta">Malai Kofta</option>
                <option value="tandoorinaan">Tandoori Naan</option>

            </select>
            <label for="quantity">Quantity:</label>
            <input type="number" id="quantity" min="1" max="10">
            <button type="button" onclick="calculate()">Calculate</button>
            <button class ="button"><a
href="C:\Users\dell\Documents\Insulin_Predictor\templates\predict.html">Insulin
Predictor</button></a>
        </form>
        <div id="result"></div>
    </div>

    <script src="lunch.js"></script>
</body>
</html>
```

Css

```
body {
    background-image: url('F5.jpg');
    background-size: cover;
  }

.container {
    margin: 0 auto;
    max-width: 500px;
    text-align: center;
    padding: 50px;
}

h1 {
    color: #000000;
    font-size: 36px;
}

form {
    margin-top: 30px;
    display: flex;
    flex-direction: column;
}

label {
    color: #000000;
    margin-bottom: 10px;
}

select, input {
    margin-bottom: 20px;
    padding: 5px;
}

button {
    background-color: #ffffff;
    color: #000;
    border: none;
    padding: 10px;
    margin-top: 20px;
    cursor: pointer;
}
a{
    text-decoration: none;
    color: #000;
}
#result {
    margin-top: 30px;
    color: #000000;
    font-size: 24px;
}
```

Java Script

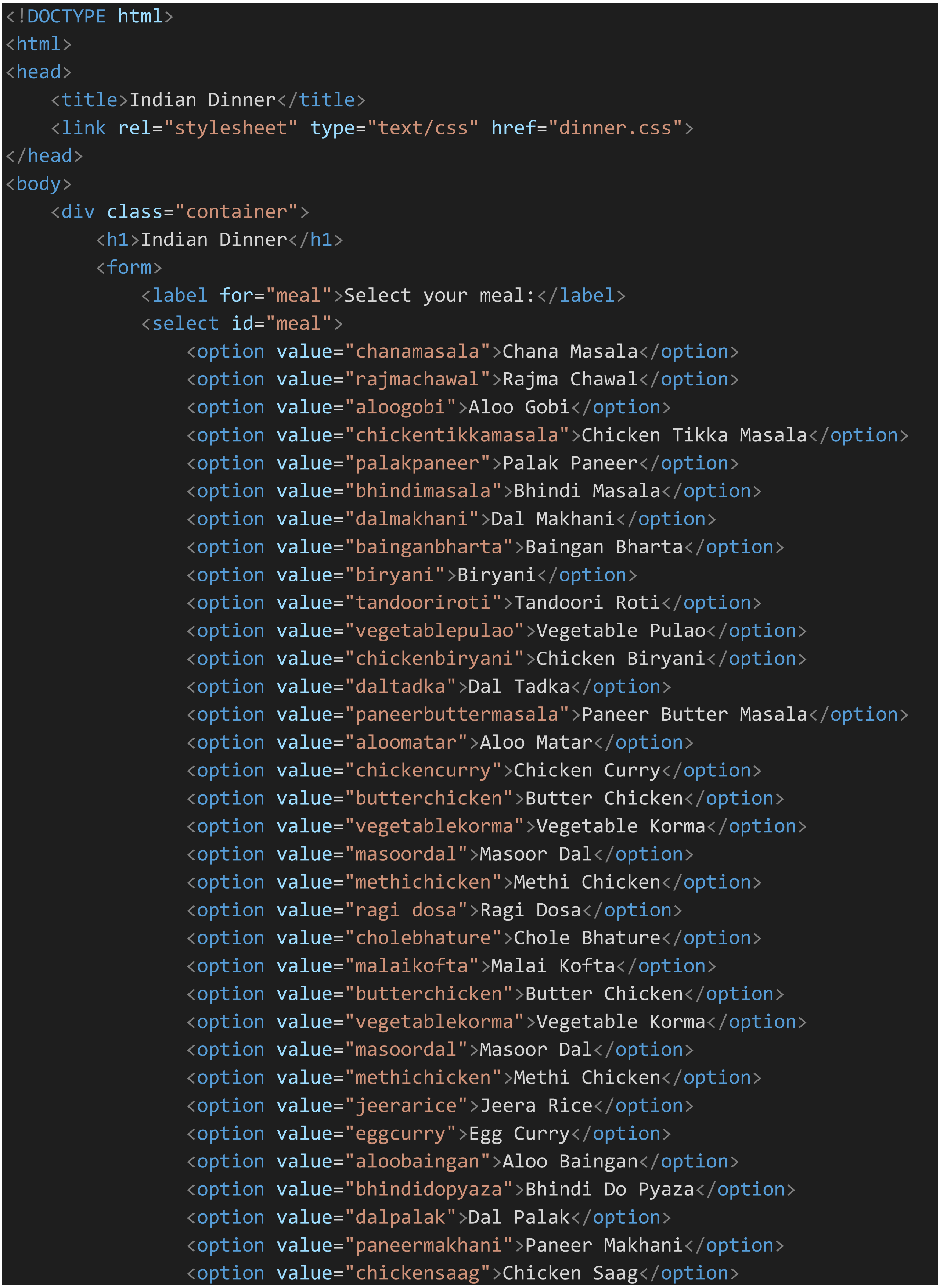
```
function calculate() {
    // Get the selected meal from the dropdown list
    var meal = document.getElementById("meal").value;

    // Get the quantity entered by the user
    var quantity = document.getElementById("quantity").value;

    // Define a dictionary of meal names and their corresponding calorie counts
    var calorieDict = {
      "chanamasala": 35,
      "rajmachawal": 60,
      "aloogobi": 16,
      "chickentikkamasala": 14,
      "palakpaneer": 10,
```



```
      "bhindimasala": 10,
      "dalmakhani": 30,
      "bainganbharta": 14,
      "biryani": 50,
      "tandooriroti": 20,
      "vegetablepulao": 46,
      "chickenbiryani": 45,
      "daltadka": 30,
      "paneerbuttermasala": 14,
      "aloomatar": 20,
      "chickencurry": 10,
      "butterchicken": 60,
      "vegetablekorma": 35,
      "masoordal": 35,
      "methichicken": 30,
      "malaikofta" : 20 ,
      "butterchicken" : 40,
      "vegetablekorma" : 35,
      "masoordal" : 40,
      "jeerarice" : 45,
      "eggcurry" : 10 ,
      "aloobaingan" : 25 ,
      "bhindidopyaza" : 15 ,
      "dalpalak" : 20 ,
      "paneermakhani" : 15 ,
      "chickensaag" : 10 ,
      "tandoorichicken" : 25 ,
      "mutterpaneer" : 15 ,
      "chickenkorma" : 10 ,
      "chicken65" : 15 ,
      "bainganbharta" : 15 ,
      "tandooriroti" : 15 ,
      "bainganbharta" : 15,
      "chickenshawrma" : 40 ,
      "vegfriedrice" : 30 ,
      "hyderabadibiryani" : 60,
      "malaikofta" : 15 ,
      "tandoorinaan" : 25

    };

    // Define the insulin ratio for 1 unit of insulin per 15 grams of
carbohydrates
    var insulinRatio = 15;

    // Calculate the total calorie count for the selected meal and quantity
    var totalCalories = calorieDict[meal] * quantity;

    // Calculate the total amount of carbohydrates in the selected meal and
quantity
    var totalCarbs = totalCalories / 4; // 1 gram of carbs = 4 calories
    // Calculate the amount of insulin required to cover the total amount of
carbohydrates
   // var insulinDose = totalCarbs / insulinRatio;

    // Round the insulin dose to 2 decimal places
    //insulinDose = insulinDose.toFixed(2);

    // Display the total calorie count and insulin dose to the user
    var resultElement = document.getElementById("result");
    resultElement.innerHTML = "Total Calories: " + totalCalories +" Grams";//+
"<br>Insulin Dose: " + insulinDose + " units";
  }
```

Indian Dinner :

HTML



```html
<!DOCTYPE html>
<html>
<head>
    <title>Indian Dinner</title>
    <link rel="stylesheet" type="text/css" href="dinner.css">
</head>
<body>
    <div class="container">
        <h1>Indian Dinner</h1>
        <form>
            <label for="meal">Select your meal:</label>
            <select id="meal">
                <option value="chanamasala">Chana Masala</option>
                <option value="rajmachawal">Rajma Chawal</option>
                <option value="aloogobi">Aloo Gobi</option>
                <option value="chickentikkamasala">Chicken Tikka Masala</option>
                <option value="palakpaneer">Palak Paneer</option>
                <option value="bhindimasala">Bhindi Masala</option>
                <option value="dalmakhani">Dal Makhani</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="biryani">Biryani</option>
                <option value="tandooriroti">Tandoori Roti</option>
                <option value="vegetablepulao">Vegetable Pulao</option>
                <option value="chickenbiryani">Chicken Biryani</option>
                <option value="daltadka">Dal Tadka</option>
                <option value="paneerbuttermasala">Paneer Butter Masala</option>
                <option value="aloomatar">Aloo Matar</option>
                <option value="chickencurry">Chicken Curry</option>
                <option value="butterchicken">Butter Chicken</option>
                <option value="vegetablekorma">Vegetable Korma</option>
                <option value="masoordal">Masoor Dal</option>
                <option value="methichicken">Methi Chicken</option>
                <option value="ragi dosa">Ragi Dosa</option>
                <option value="cholebhature">Chole Bhature</option>
                <option value="malaikofta">Malai Kofta</option>
                <option value="butterchicken">Butter Chicken</option>
                <option value="vegetablekorma">Vegetable Korma</option>
                <option value="masoordal">Masoor Dal</option>
                <option value="methichicken">Methi Chicken</option>
                <option value="jeerarice">Jeera Rice</option>
                <option value="eggcurry">Egg Curry</option>
                <option value="aloobaingan">Aloo Baingan</option>
                <option value="bhindidopyaza">Bhindi Do Pyaza</option>
                <option value="dalpalak">Dal Palak</option>
                <option value="paneermakhani">Paneer Makhani</option>
                <option value="chickensaag">Chicken Saag</option>
                <option value="tandoorichicken">Tandoori Chicken</option>
                <option value="mutterpaneer">Mutter Paneer</option>
                <option value="chickenkorma">Chicken Korma</option>
                <option value="chicken65">Chicken 65</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="tandooriroti">Tandoori Roti</option>
                <option value="bainganbharta">Baingan Bharta</option>
                <option value="chickenshawrma">Chicken Shawrma</option>
                <option value="vegfriedrice">Veg Fried Rice</option>
                <option value="hyderabadibiryani">Hyderabadi Biryani</option>
                <option value="malaikofta">Malai Kofta</option>
                <option value="tandoorinaan">Tandoori Naan</option>
            </select>
            <label for="quantity">Quantity:</label>
            <input type="number" id="quantity" min="1" max="10">
            <button type="button" onclick="calculate()">Calculate</button>
            <button class ="button"><a
href="C:\Users\dell\Documents\Insulin_Predictor\templates\predict.html">Insulin
Predictor</button></a>
        </form>
        <div id="result"></div>
    </div>

    <script src="dinner.js"></script>
</body>
</html>
```

CSS

```
body {
    background-image: url('F6.jpg');
    background-size: cover;
  }

.container {
    margin: 0 auto;
    max-width: 500px;
    text-align: center;
    padding: 50px;
}

h1 {
    color: #000000;
    font-size: 36px;
}

form {
    margin-top: 30px;
    display: flex;
    flex-direction: column;
}
```

```
label {
    color: #000000;
    margin-bottom: 10px;
}

select, input {
    margin-bottom: 20px;
    padding: 5px;
}

button {
    background-color: #ffffff;
    color: #000;
    border: none;
    padding: 10px;
    margin-top: 20px;
    cursor: pointer;

}
a{
    text-decoration: none;
    color: #000;
}

#result {
    margin-top: 30px;
    color: #000000;
    font-size: 24px;
}
```

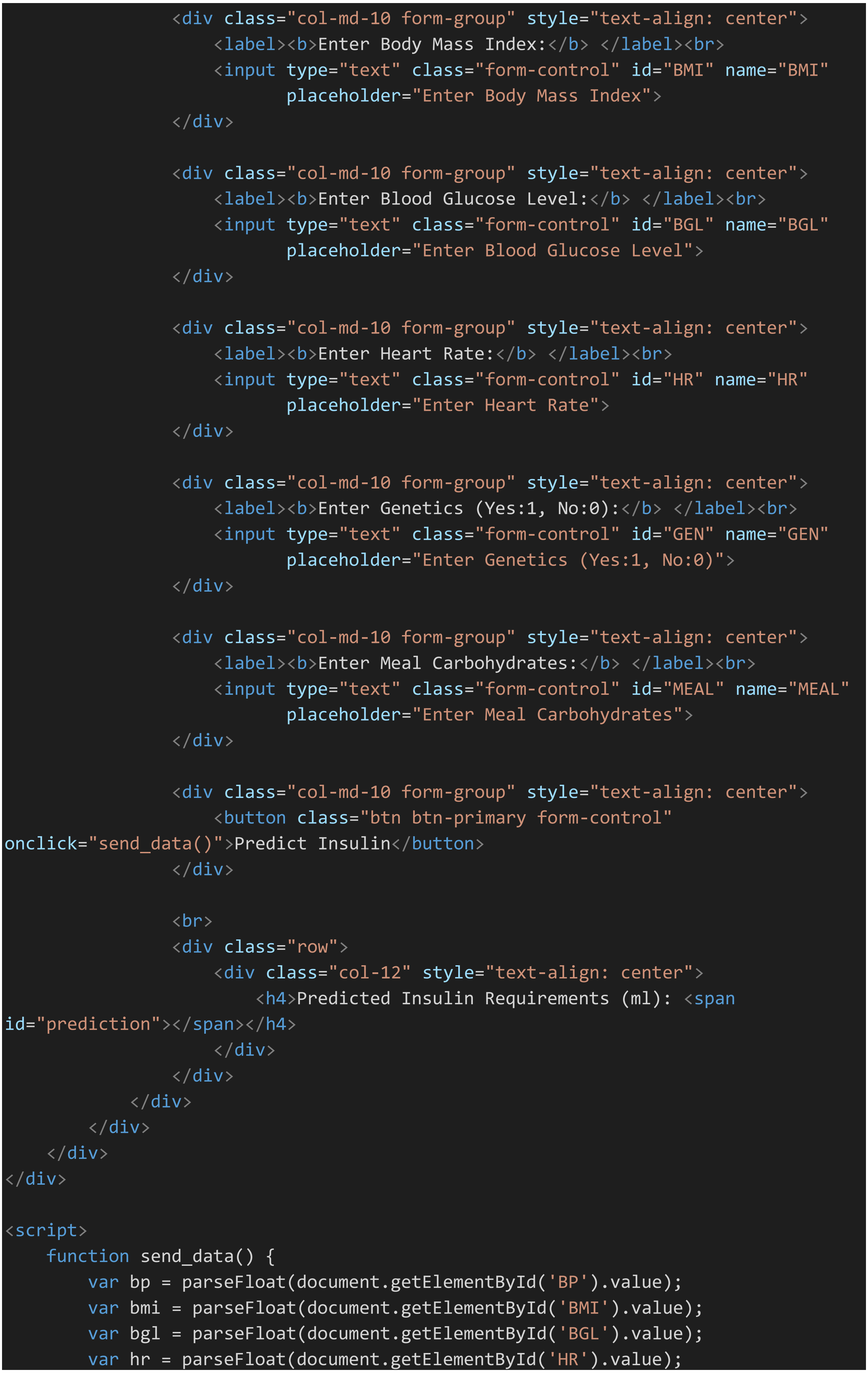
Java Script

```
function calculate() {
    // Get the selected meal from the dropdown list
    var meal = document.getElementById("meal").value;

    // Get the quantity entered by the user
    var quantity = document.getElementById("quantity").value;

    // Define a dictionary of meal names and their corresponding calorie counts
    var calorieDict = {
      "chanamasala": 35,
      "rajmachawal": 60,
      "aloogobi": 16,
      "chickentikkamasala": 14,
      "palakpaneer": 10,
      "bhindimasala": 10,
      "dalmakhani": 30,
      "bainganbharta": 14,
      "biryani": 50,
      "tandooriroti": 20,
      "vegetablepulao": 46,
      "chickenbiryani": 45,
      "daltadka": 30,
      "paneerbuttermasala": 14,
      "aloomatar": 20,
      "chickencurry": 10,
      "butterchicken": 60,
      "vegetablekorma": 35,
      "masoordal": 35,
      "methichicken": 30,
      "malaikofta" : 20 ,
      "butterchicken" : 40,
      "vegetablekorma" : 35,
      "masoordal" : 40,
      "jeerarice" : 45,
      "eggcurry" : 10 ,
      "aloobaingan" : 25 ,
      "bhindidopyaza" : 15 ,
      "dalpalak" : 20 ,
      "paneermakhani" : 15 ,
      "chickensaag" : 10 ,
      "tandoorichicken" : 25 ,
      "mutterpaneer" : 15 ,
      "chickenkorma" : 10 ,
      "chicken65" : 15 ,
      "bainganbharta" : 15 ,
      "tandooriroti" : 15 ,
      "bainganbharta" : 15,
      "chickenshawrma" : 40 ,
      "vegfriedrice" : 30 ,
      "hyderabadibiryani" : 60,
      "malaikofta" : 15 ,
      "tandoorinaan" : 25

    };

    // Define the insulin ratio for 1 unit of insulin per 15 grams of
carbohydrates
    var insulinRatio = 15;

    // Calculate the total calorie count for the selected meal and quantity
    var totalCalories = calorieDict[meal] * quantity;

    // Calculate the total amount of carbohydrates in the selected meal and
quantity
    var totalCarbs = totalCalories / 4; // 1 gram of carbs = 4 calories

    // Calculate the amount of insulin required to cover the total amount of
carbohydrates
    var insulinDose = totalCarbs / insulinRatio;
    // Round the insulin dose to 2 decimal places
    //insulinDose = insulinDose.toFixed(2);

    // Display the total calorie count and insulin dose to the user
    var resultElement = document.getElementById("result");
    resultElement.innerHTML = "Total Calories: " + totalCalories +" Grams";//+
"<br>Insulin Dose: " + insulinDose + " units";
  }
```

Insulin Prediction Page :

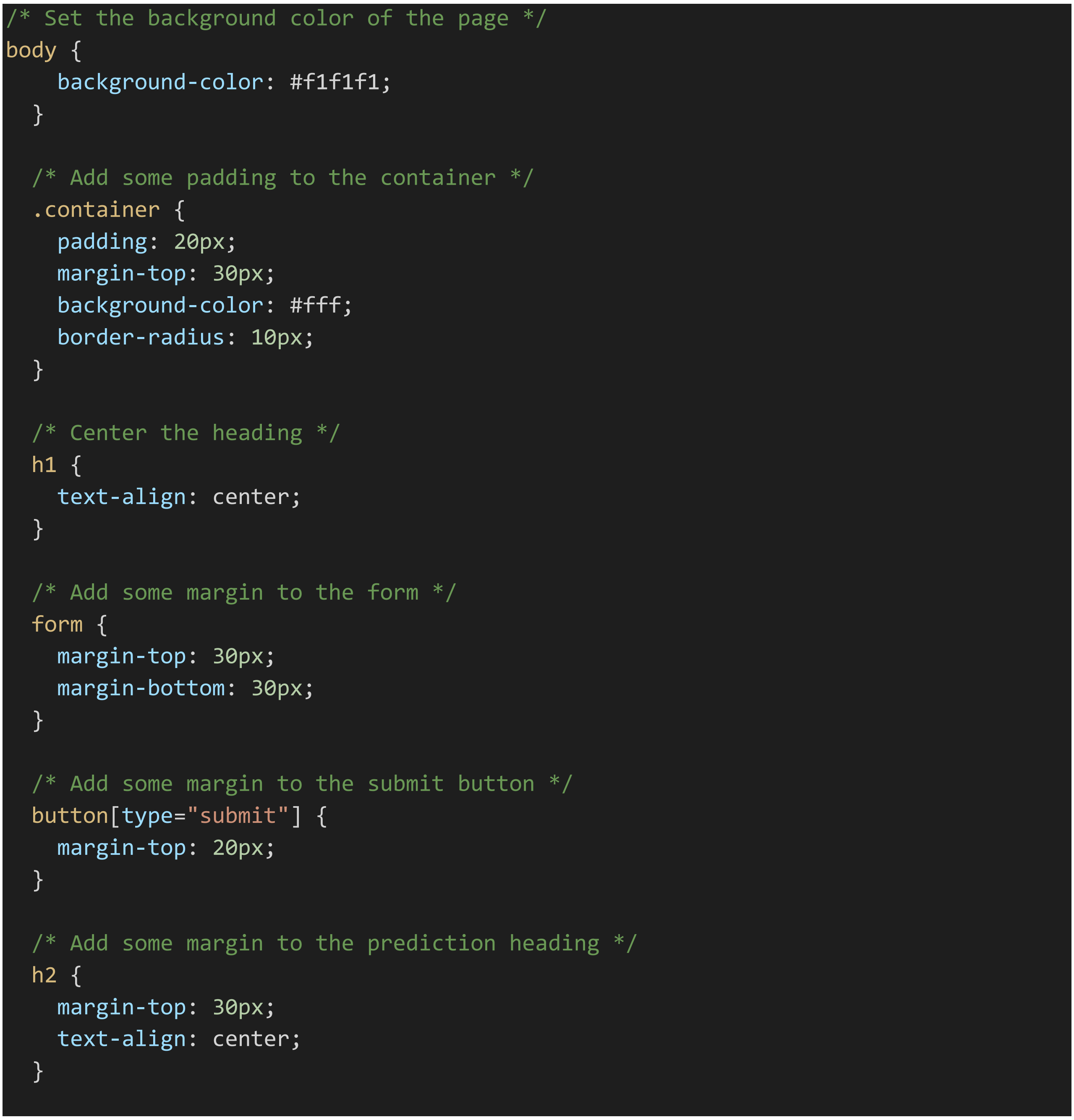
HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>TYPE 1 DIABETES PREDICTION</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css')
}}">
    <link rel="stylesheet" type="text/css"
          href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.11.2/css/all.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
            crossorigin="anonymous"></script>

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
          integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"></script>
</head>
<body class="bg-dark">
<div class="container">
    <div class="row">
        <div class="card mt-50" style="width: 100%; height: 100%">
            <div class="card-header" style="text-align: center">
                <h1>WELCOME TO TYPE 1 DIABETES INSULIN PREDICTOR</h1>
            </div>
            <div class="card-body">
                <div class="col-12" style="text-align: center">
                    <h5>This Will Predict Insulin Requirements for Type 1 Diabetes
Patients:</h5>
                </div>
                <br>
                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Blood Pressure:</b> </label><br>
                    <input type="text" class="form-control" id="BP" name="BP"
                           placeholder="Enter Blood Pressure">
                </div>
```



```
                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Body Mass Index:</b> </label><br>
                    <input type="text" class="form-control" id="BMI" name="BMI"
                           placeholder="Enter Body Mass Index">
                </div>

                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Blood Glucose Level:</b> </label><br>
                    <input type="text" class="form-control" id="BGL" name="BGL"
                           placeholder="Enter Blood Glucose Level">
                </div>

                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Heart Rate:</b> </label><br>
                    <input type="text" class="form-control" id="HR" name="HR"
                           placeholder="Enter Heart Rate">
                </div>

                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Genetics (Yes:1, No:0):</b> </label><br>
                    <input type="text" class="form-control" id="GEN" name="GEN"
                           placeholder="Enter Genetics (Yes:1, No:0)">
                </div>

                <div class="col-md-10 form-group" style="text-align: center">
                    <label><b>Enter Meal Carbohydrates:</b> </label><br>
                    <input type="text" class="form-control" id="MEAL" name="MEAL"
                           placeholder="Enter Meal Carbohydrates">
                </div>

                <div class="col-md-10 form-group" style="text-align: center">
                    <button class="btn btn-primary form-control"
onclick="send_data()">Predict Insulin</button>
                </div>

                <br>
                <div class="row">
                    <div class="col-12" style="text-align: center">
                        <h4>Predicted Insulin Requirements (ml): <span
id="prediction"></span></h4>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
    function send_data() {
        var bp = parseFloat(document.getElementById('BP').value);
        var bmi = parseFloat(document.getElementById('BMI').value);
        var bgl = parseFloat(document.getElementById('BGL').value);
        var hr = parseFloat(document.getElementById('HR').value);
        var gen = parseInt(document.getElementById('GEN').value);
        var meal = parseFloat(document.getElementById('MEAL').value);

        var insulin_requirements = calculateInsulinRequirements(bp, bmi, bgl, hr, 
gen, meal);
        document.getElementById('prediction').innerHTML = 
insulin_requirements.toFixed(2);
    }

    function calculateInsulinRequirements(bp, bmi, bgl, hr, gen, meal) {
        var baselineInsulin =  10 // Set the baseline insulin value here
        var targetGlucose =  100 // Set the target glucose level here
        var insulinSensitivity =  0.5 // Set the insulin sensitivity factor here
        var insulinCarbRatio = 2 // Set the insulin-to-carbohydrate ratio here
        var geneticFactor =  1 // Set the genetic factor here

        var insulinRequirements = baselineInsulin + (bgl - targetGlucose) * 
insulinSensitivity + meal * insulinCarbRatio + gen * geneticFactor;
        return insulinRequirements;
    }
</script>

<!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
            integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
            crossorigin="anonymous"></script>
    <script 
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
            crossorigin="anonymous"></script>
    <script 
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
            integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3IpuThere were a few errors in the 
HTML code, which I have corrected. Here's the updated HTML code:
</body>
</html>
```

CSS



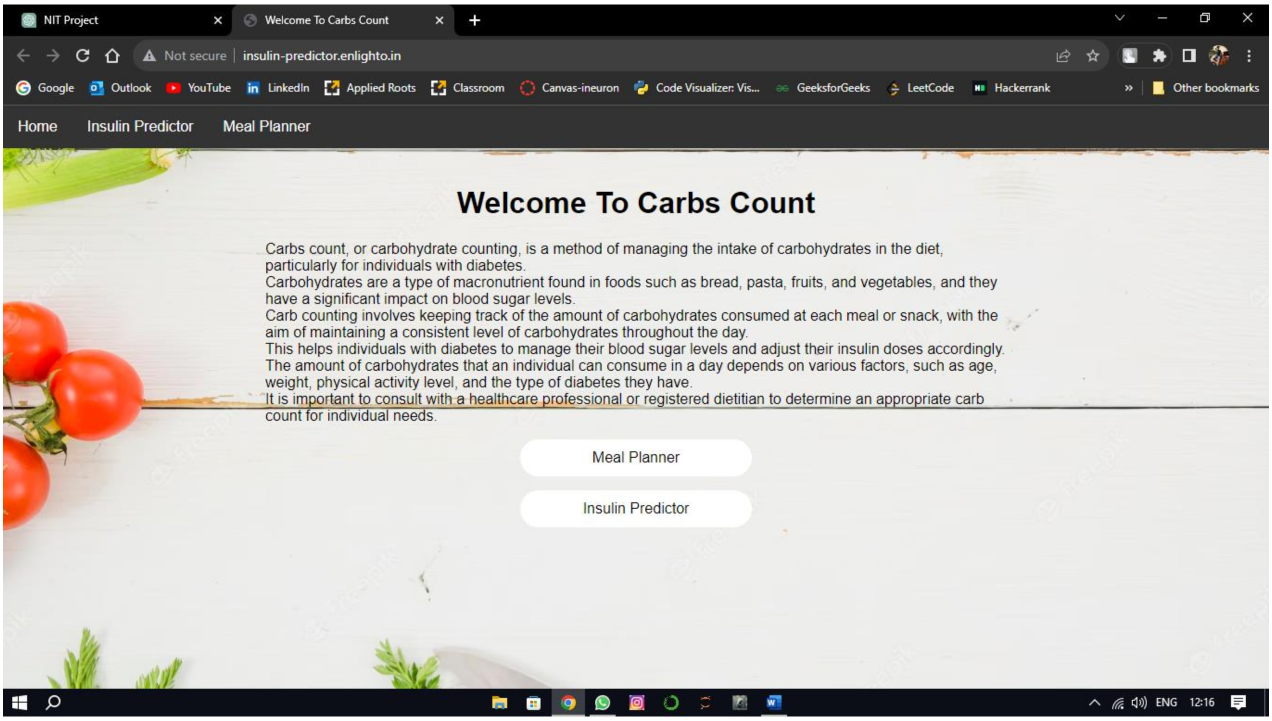
```css
/* Set the background color of the page */
body {
    background-color: #f1f1f1;
  }

  /* Add some padding to the container */
  .container {
    padding: 20px;
    margin-top: 30px;
    background-color: #fff;
    border-radius: 10px;
  }

  /* Center the heading */
  h1 {
    text-align: center;
  }

  /* Add some margin to the form */
  form {
    margin-top: 30px;
    margin-bottom: 30px;
  }

  /* Add some margin to the submit button */
  button[type="submit"] {
    margin-top: 20px;
  }

  /* Add some margin to the prediction heading */
  h2 {
    margin-top: 30px;
    text-align: center;
  }
```

Python File For Connectivity Using Flask

```
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)

# Load the trained model
model=pickle.load(open(r'C:\Users\dell\Documents\Insulin_Predictor\Type1final.pkl'
,'rb'),encoding='utf-8')

# Define the route for the home page
@app.route('/')
def home():
    return render_template('predict.html')

# Define the route for the prediction
@app.route('/predict', methods=['POST'])
def predict():
    # Get the input values from the HTML form
    BP = float(request.form['BP'])
    BMI = float(request.form['BMI'])
    BGL = float(request.form['BGL'])
    HR = float(request.form['HR'])
    GEN = int(request.form['GEN'])
    MEAL = float(request.form['MEAL'])

    # Perform any necessary preprocessing on the input values
    # ...

    # Apply the prediction formula to get the predicted insulin requirements
    baseline_insulin =  10 # Set the baseline insulin value here
    target_glucose =  100 # Set the target glucose level here
    insulin_sensitivity = 0.5 # Set the insulin sensitivity factor here
    insulin_carb_ratio =  2 # Set the insulin-to-carbohydrate ratio here
    genetic_factor =  1 # Set the genetic factor here

    insulin_requirements = baseline_insulin + (BGL - target_glucose) *
insulin_sensitivity + MEAL * insulin_carb_ratio + GEN * genetic_factor

    # Return the predicted insulin requirements to the HTML page
    return render_template('predict.html', prediction=insulin_requirements)

if __name__ == '__main__':
    app.run(debug=True)
```

Output

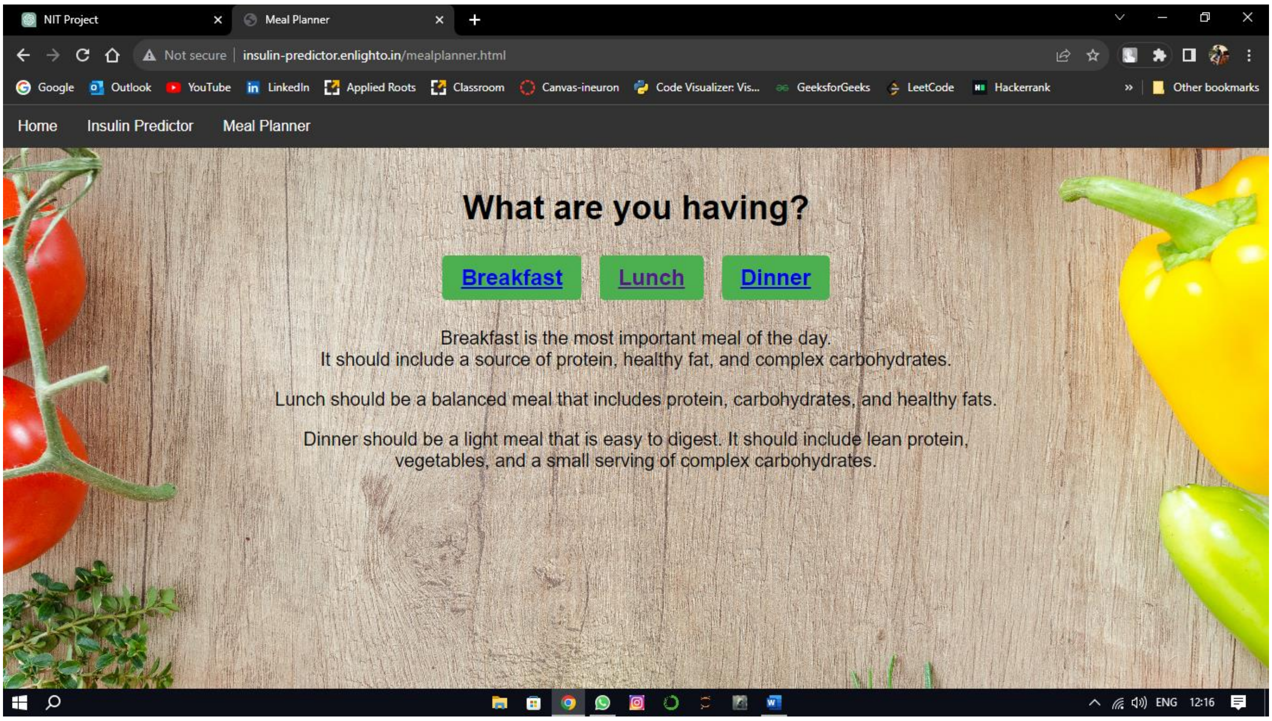
Link : http://insulin-predictor.enlighto.in/


NIT Project
Welcome To Carbs Count
Not secure | insulin-predictor.enlighto.in
Google Outlook YouTube LinkedIn Applied Roots Classroom Canvas-ineuron Code Visualizer: Vis... GeeksforGeeks LeetCode Hackerrank
Other bookmarks
Home Insulin Predictor Meal Planner
ENG 12:16

Welcome To Carbs Count

Carbs count, or carbohydrate counting, is a method of managing the intake of carbohydrates in the diet, particularly for individuals with diabetes.
Carbohydrates are a type of macronutrient found in foods such as bread, pasta, fruits, and vegetables, and they have a significant impact on blood sugar levels.
Carb counting involves keeping track of the amount of carbohydrates consumed at each meal or snack, with the aim of maintaining a consistent level of carbohydrates throughout the day.
This helps individuals with diabetes to manage their blood sugar levels and adjust their insulin doses accordingly.
The amount of carbohydrates that an individual can consume in a day depends on various factors, such as age, weight, physical activity level, and the type of diabetes they have.
It is important to consult with a healthcare professional or registered dietitian to determine an appropriate carb count for individual needs.

Meal Planner
Insulin Predictor

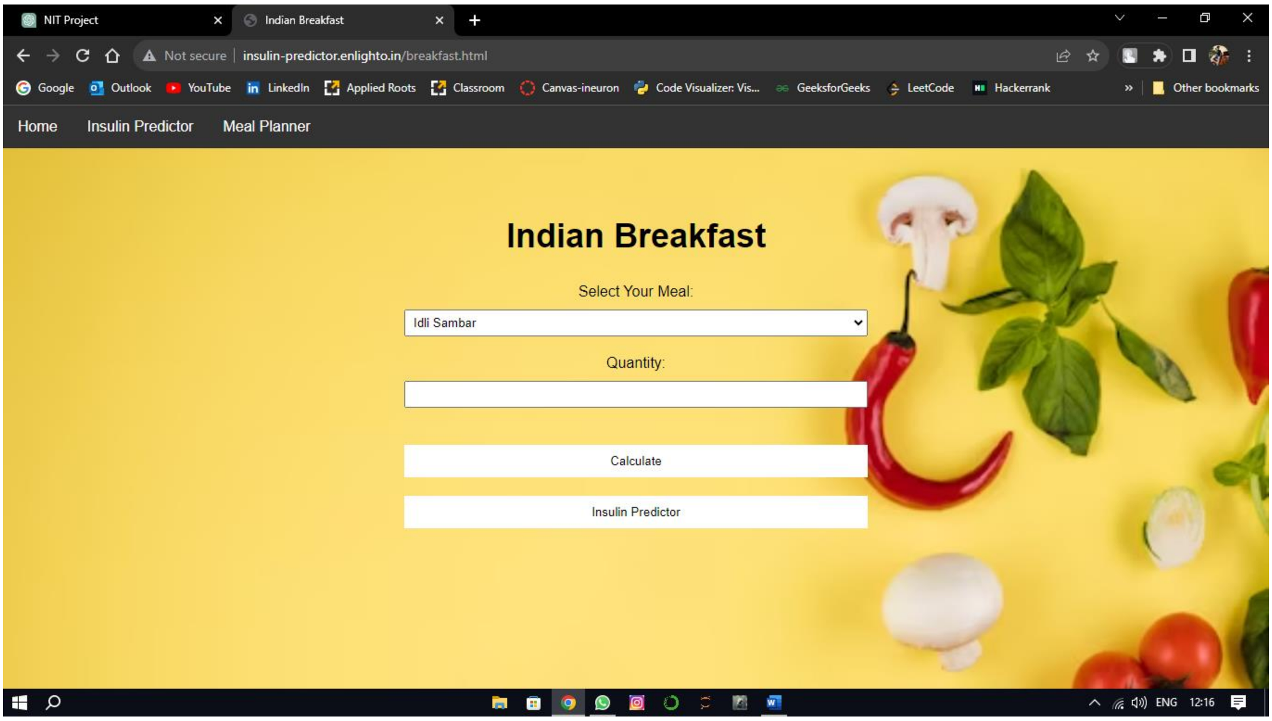

NIT Project
Meal Planner
Not secure | insulin-predictor.enlighto.in/mealplanner.html
Google Outlook YouTube LinkedIn Applied Roots Classroom Canvas-ineuron Code Visualizer: Vis... GeeksforGeeks LeetCode Hackerrank
Other bookmarks
Home Insulin Predictor Meal Planner
ENG 12:16

What are you having?

Breakfast is the most important meal of the day.
It should include a source of protein, healthy fat, and complex carbohydrates.

Lunch should be a balanced meal that includes protein, carbohydrates, and healthy fats.

Dinner should be a light meal that is easy to digest. It should include lean protein, vegetables, and a small serving of complex carbohydrates.

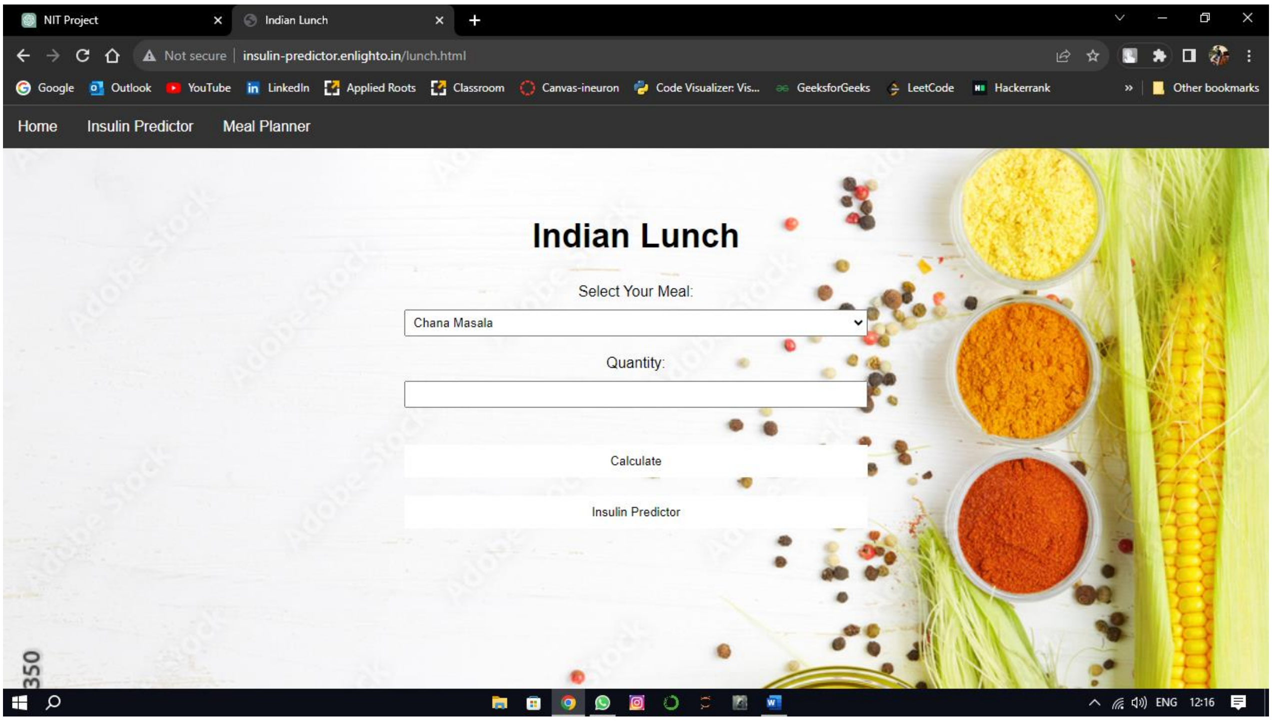


Indian Breakfast

Select Your Meal:

Idli Sambar

Quantity:

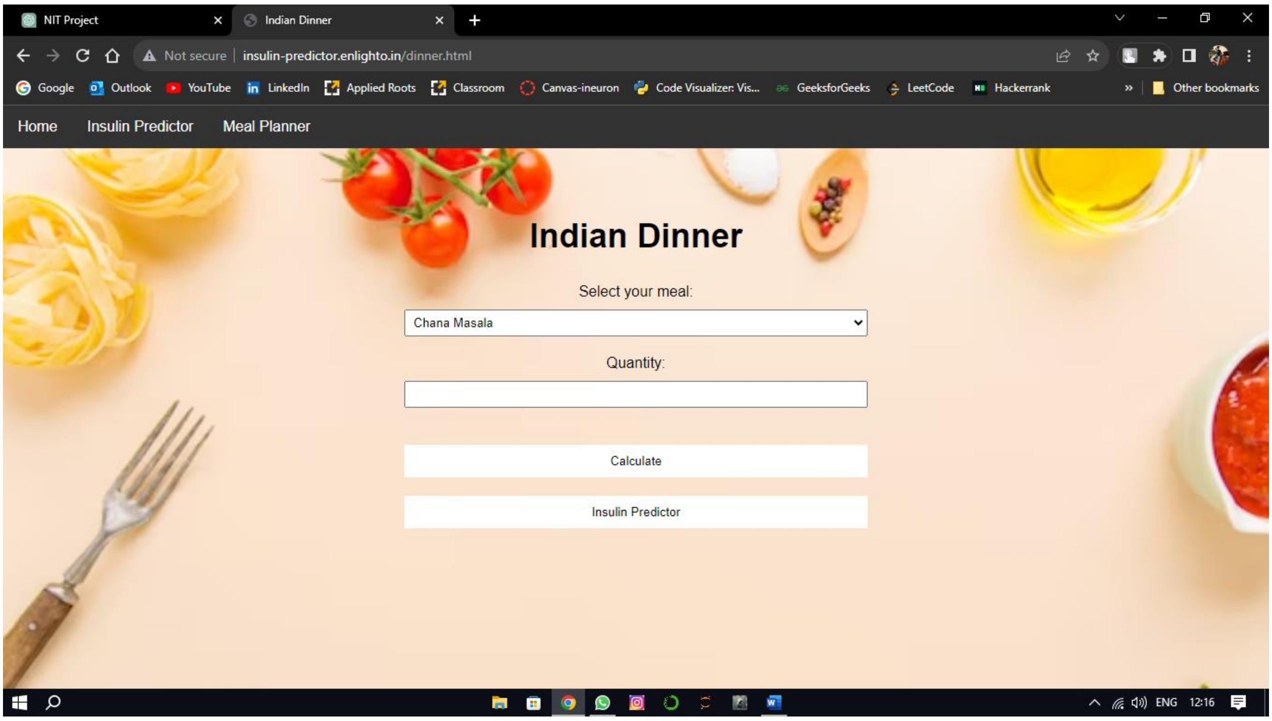


Indian Lunch

Select Your Meal:

Chana Masala

Quantity:



Indian Dinner

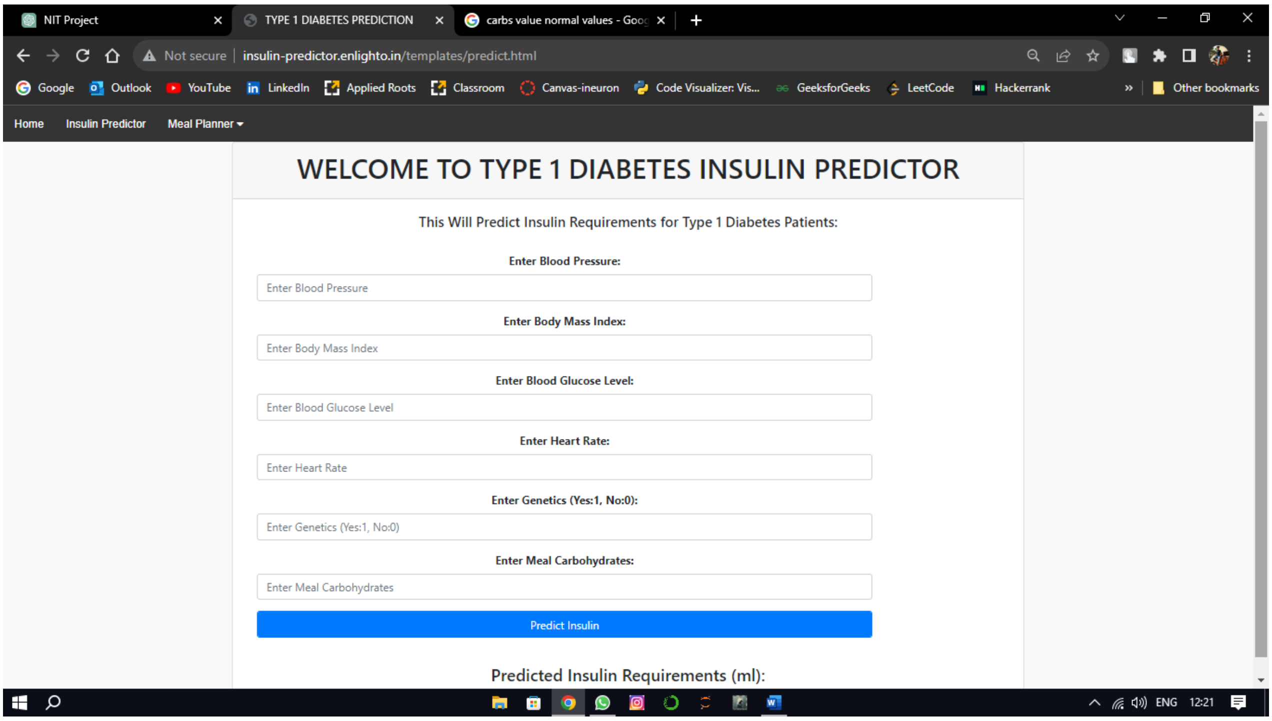
Select your meal:

Chana Masala

Quantity:

Calculate

Insulin Predictor



WELCOME TO TYPE 1 DIABETES INSULIN PREDICTOR

This Will Predict Insulin Requirements for Type 1 Diabetes Patients:

Enter Blood Pressure:

Enter Blood Pressure

Enter Body Mass Index:

Enter Body Mass Index

Enter Blood Glucose Level:

Enter Blood Glucose Level

Enter Heart Rate:

Enter Heart Rate

Enter Genetics (Yes:1, No:0):

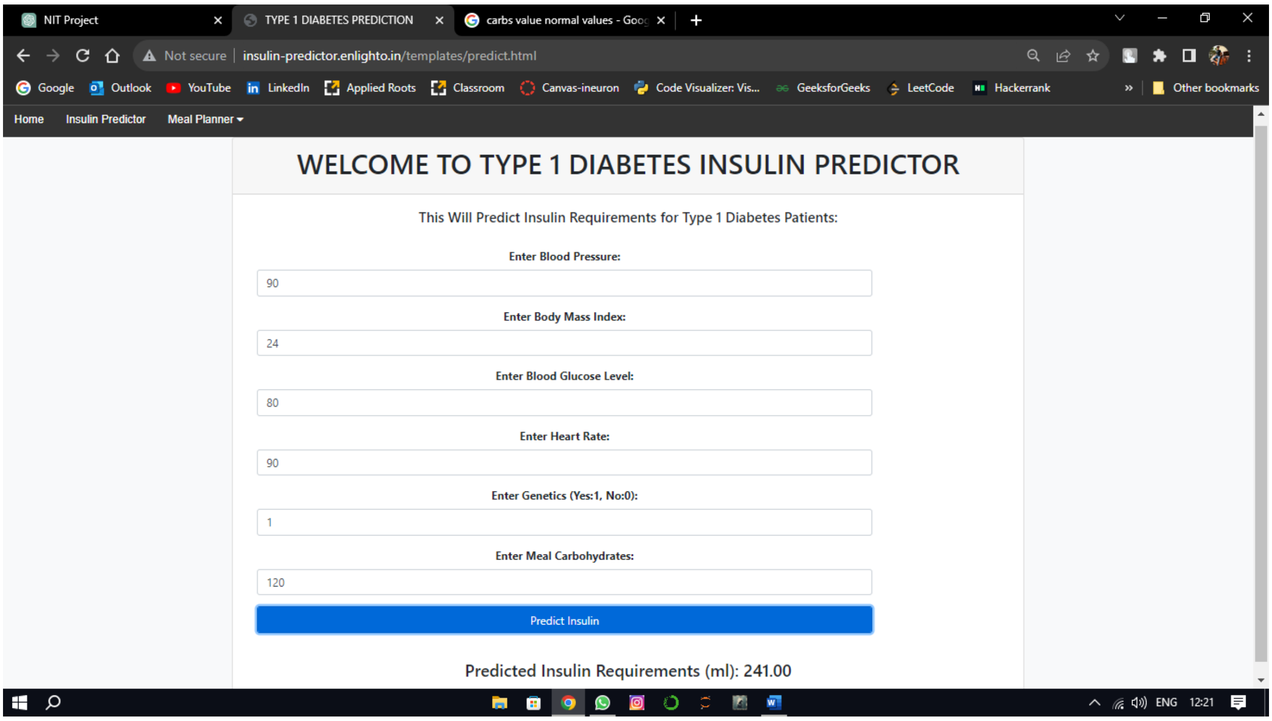
Enter Genetics (Yes:1, No:0)

Enter Meal Carbohydrates:

Enter Meal Carbohydrates

Predict Insulin

Predicted Insulin Requirements (ml):



WELCOME TO TYPE 1 DIABETES INSULIN PREDICTOR

This Will Predict Insulin Requirements for Type 1 Diabetes Patients:

Enter Blood Pressure: 90

Enter Body Mass Index: 24

Enter Blood Glucose Level: 80

Enter Heart Rate: 90

Enter Genetics (Yes:1, No:0): 1

Enter Meal Carbohydrates: 120

Predicted Insulin Requirements (ml): 241.00

THANK YOU