



# **Advanced Database Systems**

## **Lab Assignment 2**

### **Medical Recommendation System**

**Group No. 7**

**Members:**

**Aashay Kaurav (2022H1030100)**

**Bhagirath Parmar (2022H1030076)**

**Gourav (2022H1030071)**

**Parth Goswami (2022H1030066)**

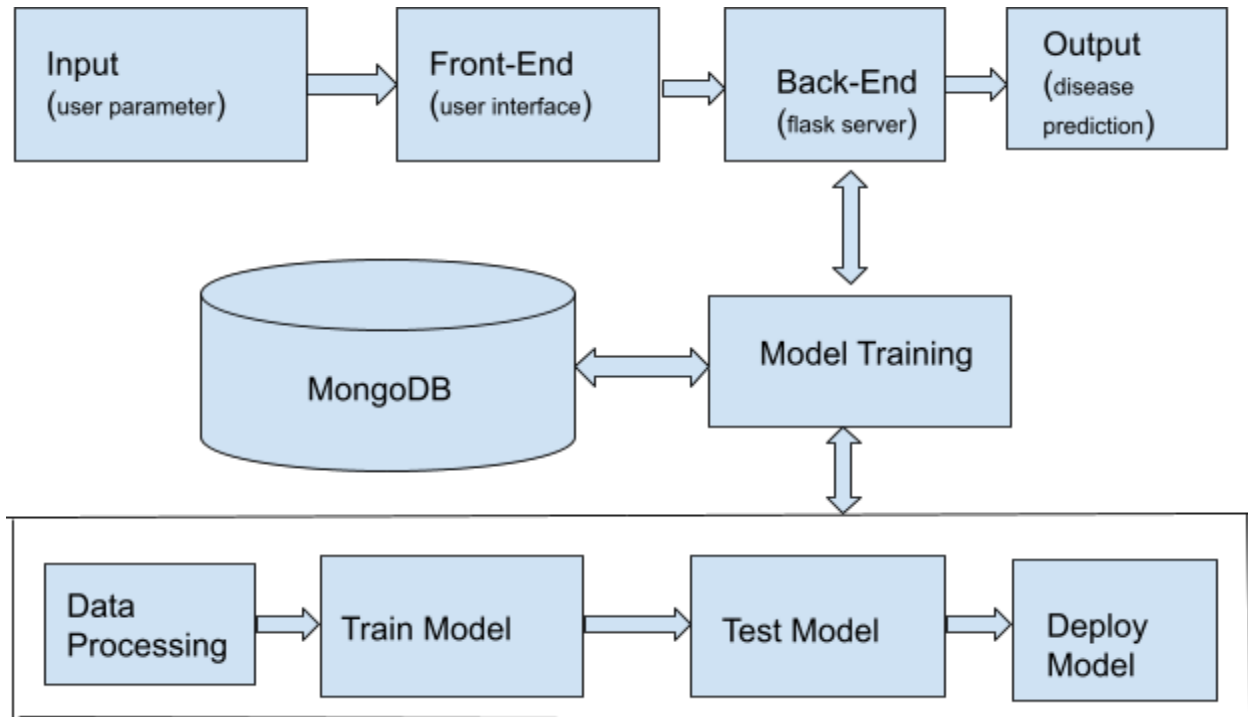


**Birla Institute of Technology & Science, Pilani**  
Hyderabad Campus

## **CONTENTS**

Brief working flow.....	
Setting up the stage .....	
UI interaction guide.....	
Explaining the codes.....	

## Brief working flow



## Setting up the stage: installing environment in machine

### Prerequisites:

- OS: Ubuntu 22.04 LTS
- MONGODB version-6.0.5
- Python version- 3.10.6
- Pymongo version 4.3.3

1) The MongoDB public GPG Key has to be imported from the official MongoDB repository. Write the following code in the terminal:

```
bhagirath@bhagirath-X510UQR: ~  
bhagirath@bhagirath-X510UQR:~$ curl -fsSL https://pgp.mongodb.com/server-6.0.pub | \  
sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg
```

2) Then we have to create the list file for our MongoDB.

```
bhagirath@bhagirath-X510UQR: ~  
bhagirath@bhagirath-X510UQR:~$ echo "deb [ arch=amd64,arm64 signed=/usr/share/keyrings/mongodb-server-6.0.gpg ] https://repo.mongodb.org/apt/ubuntu jammy/mong  
odb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
```

3) We will update the local package database .

```
bhagirath@bhagirath-X510UQR: ~  
bhagirath@bhagirath-X510UQR:~$ sudo apt-get update
```

4) Now install the MongoDB packages using the given command:

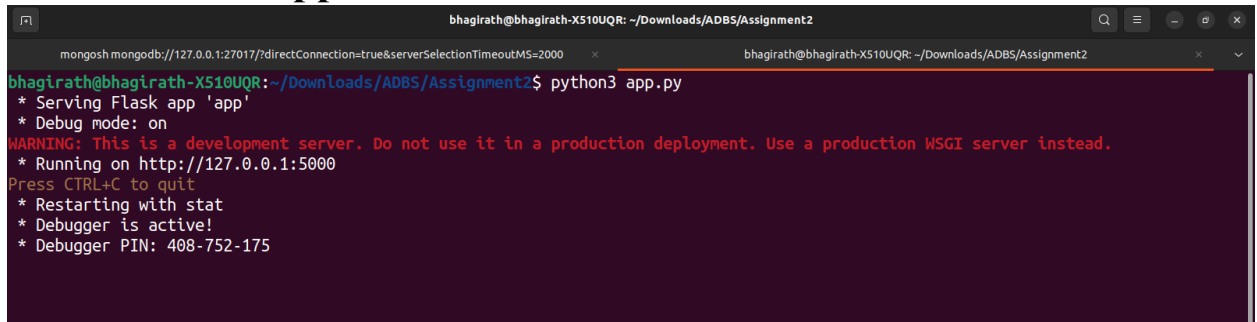
```
bhagirath@bhagirath-X510UQR: ~  
bhagirath@bhagirath-X510UQR:~$ sudo apt-get install -y mongodb-org
```

5) We are using the *pymongo* library to use MongoDB into python.

```
bhagirath@bhagirath-X510UQR: ~  
bhagirath@bhagirath-X510UQR:~$ pip install pymongo
```

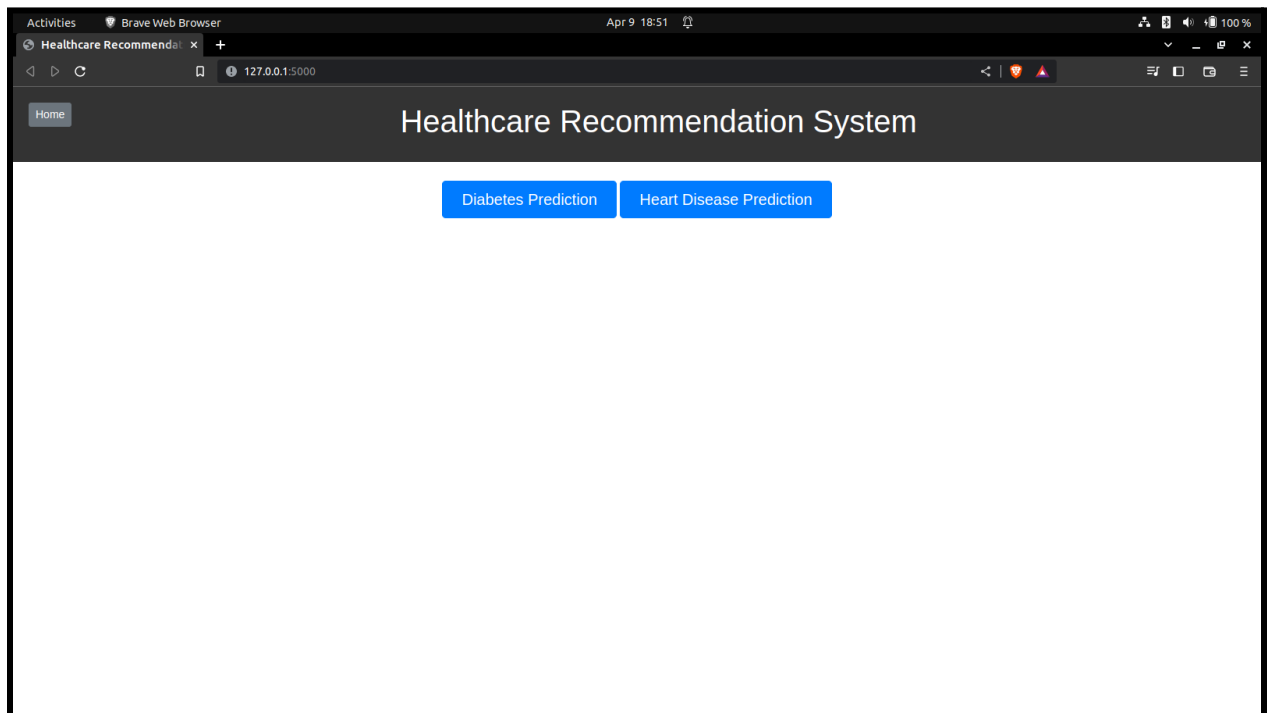
## UI Interaction guide: Steps to run and use the application:

### 1) Start the flask application.

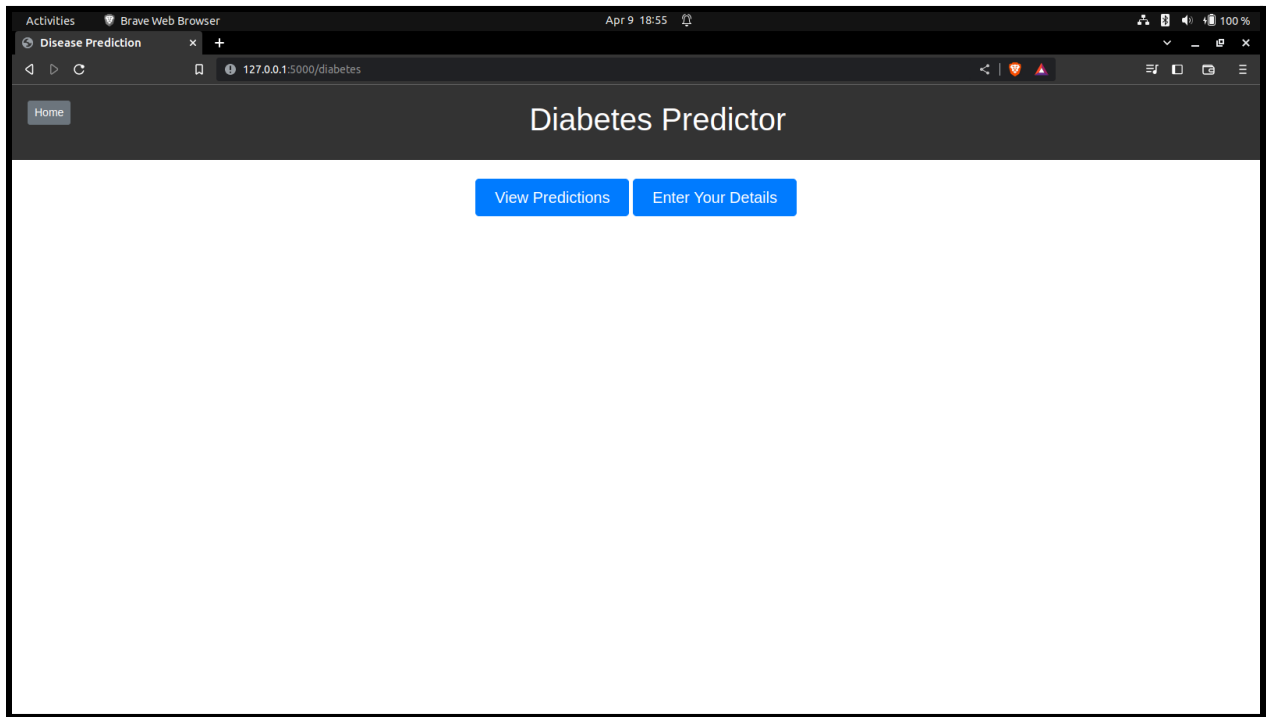


```
bhagirath@bhagirath-X510UQR: ~/Downloads/ADBS/Assignment2
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
bhagirath@bhagirath-X510UQR: ~/Downloads/ADBS/Assignment2$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 408-752-175
```

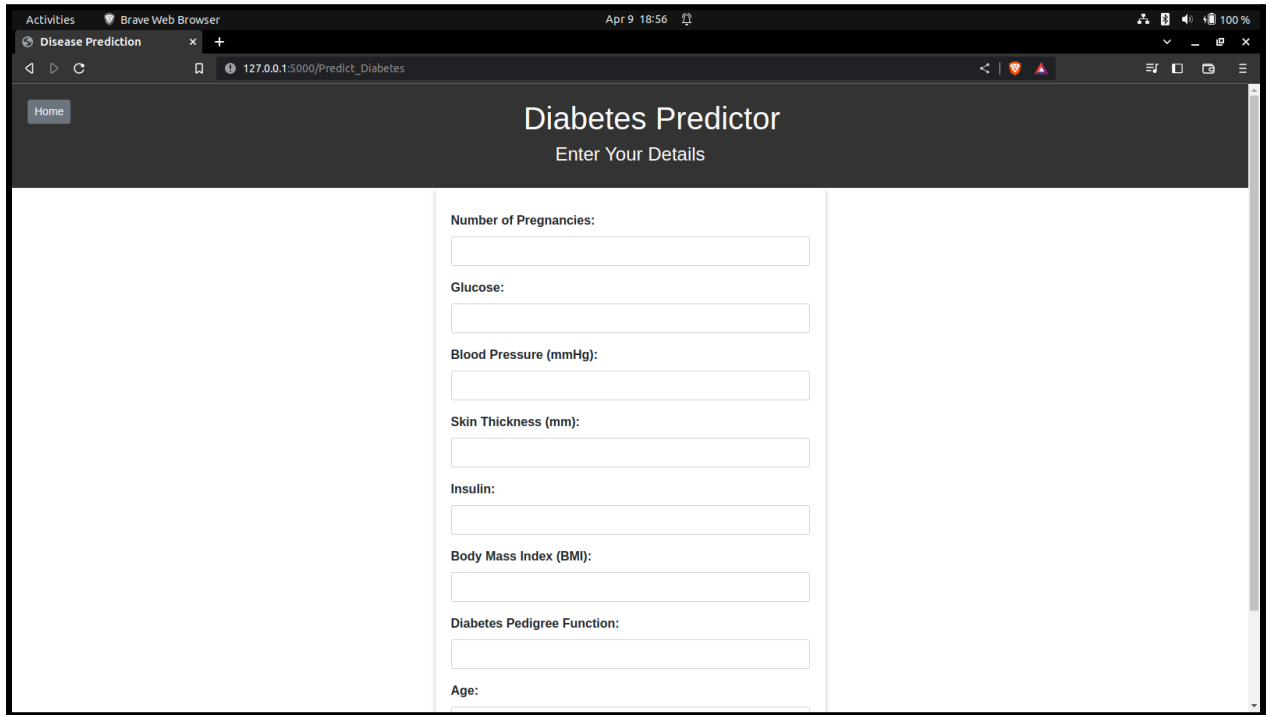
### 2) Open the URL below in the browser to get to the home page: you have two diseases you can see in the home page, clicking on either will take you to the corresponding system designed for it.



### 3) Choose a disease from the user interface :



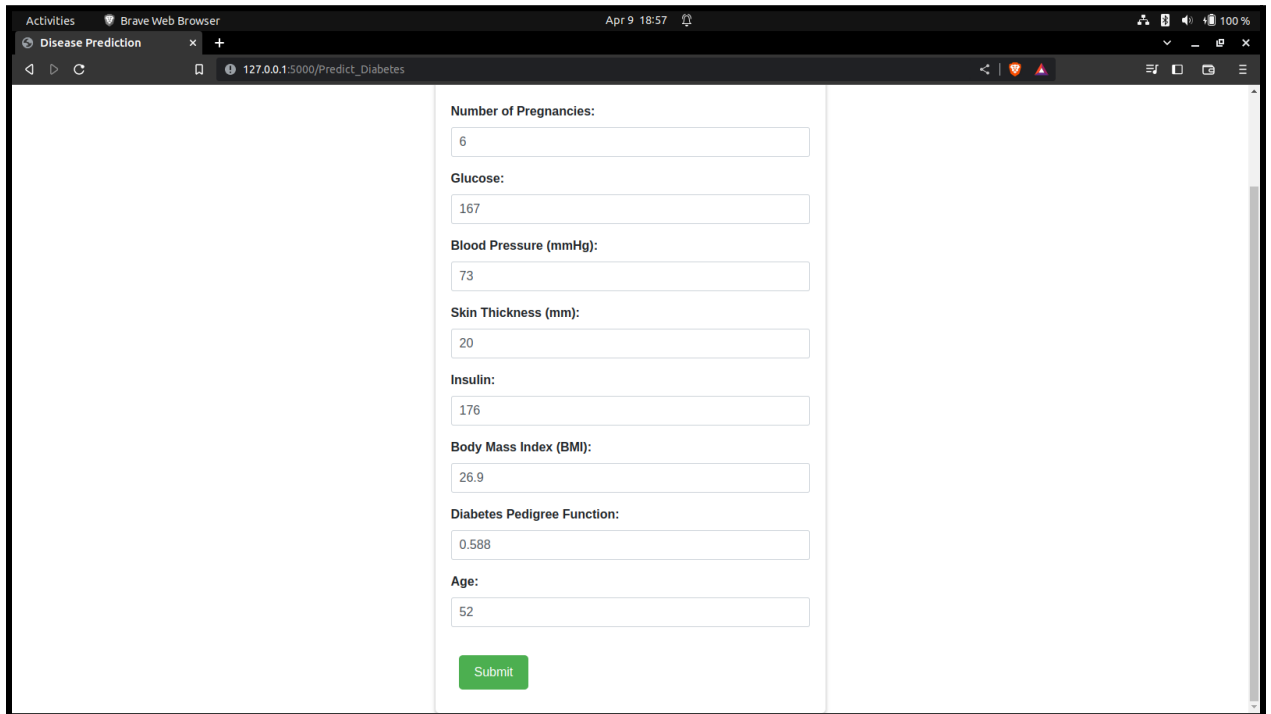
4) Here, we have chosen the “Enter Your Details” tab. In this prompt, you will be asked to enter your details as required in the form given.



The screenshot shows a web browser window with the title 'Disease Prediction' and the address bar displaying '127.0.0.1:5000/Predict\_Diabetes'. The page has a dark header with a 'Home' button and the title 'Diabetes Predictor' followed by the subtitle 'Enter Your Details'. The main content area is a form with the following fields:

- Number of Pregnancies:
- Glucose:
- Blood Pressure (mmHg):
- Skin Thickness (mm):
- Insulin:
- Body Mass Index (BMI):
- Diabetes Pedigree Function:
- Age:

## 5) Enter each detail accurately.



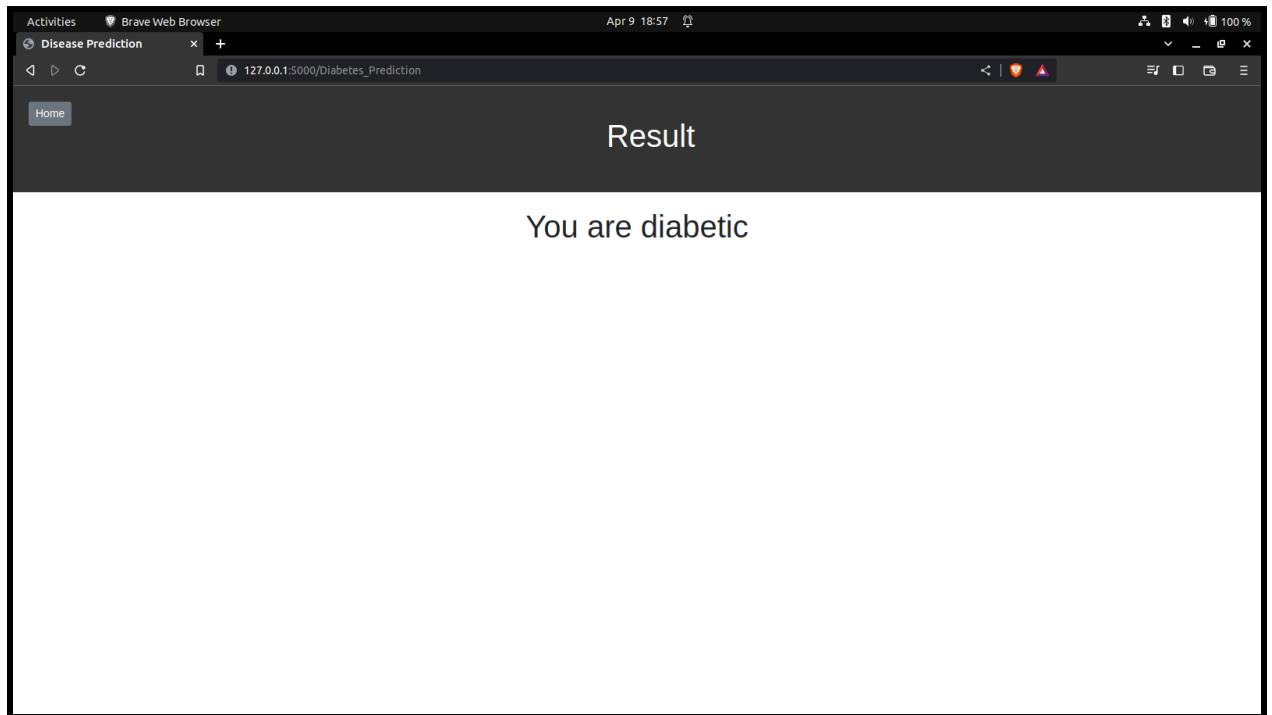
The screenshot shows a Brave Web Browser window with a single tab titled 'Disease Prediction'. The address bar displays the URL '127.0.0.1:5000/Predict\_Diabetes'. The form is a vertical stack of input fields, each preceded by a label. The labels and their corresponding values are: 'Number of Pregnancies:' with '6', 'Glucose:' with '167', 'Blood Pressure (mmHg):' with '73', 'Skin Thickness (mm):' with '20', 'Insulin:' with '176', 'Body Mass Index (BMI):' with '26.9', 'Diabetes Pedigree Function:' with '0.588', and 'Age:' with '52'. At the bottom of the form is a green 'Submit' button. The browser's status bar at the top shows the time as 'Apr 9 18:57' and a battery level of '100%'.

Field Label	Value
Number of Pregnancies:	6
Glucose:	167
Blood Pressure (mmHg):	73
Skin Thickness (mm):	20
Insulin:	176
Body Mass Index (BMI):	26.9
Diabetes Pedigree Function:	0.588
Age:	52

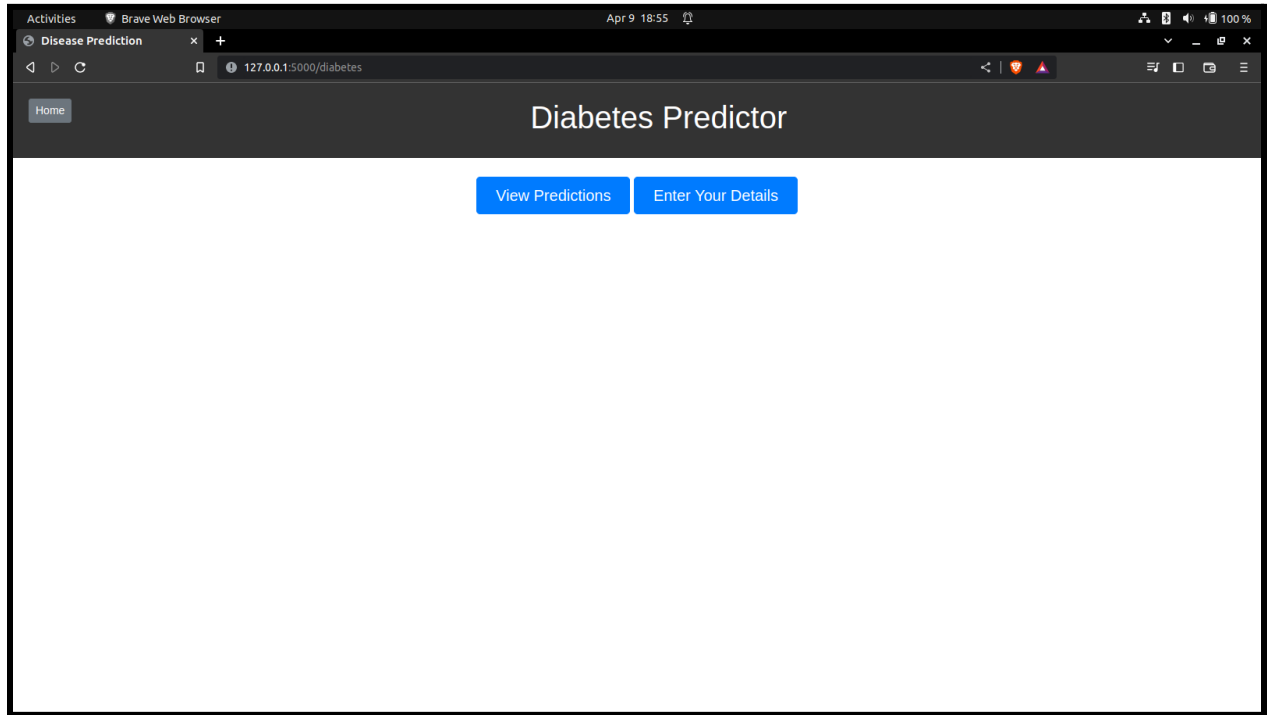
Submit



**6) Upon clicking the “Submit” button, your details are sent from the front-end to the trained model, which will conclude whether you are having the disease or not, as a binary classification problem.**



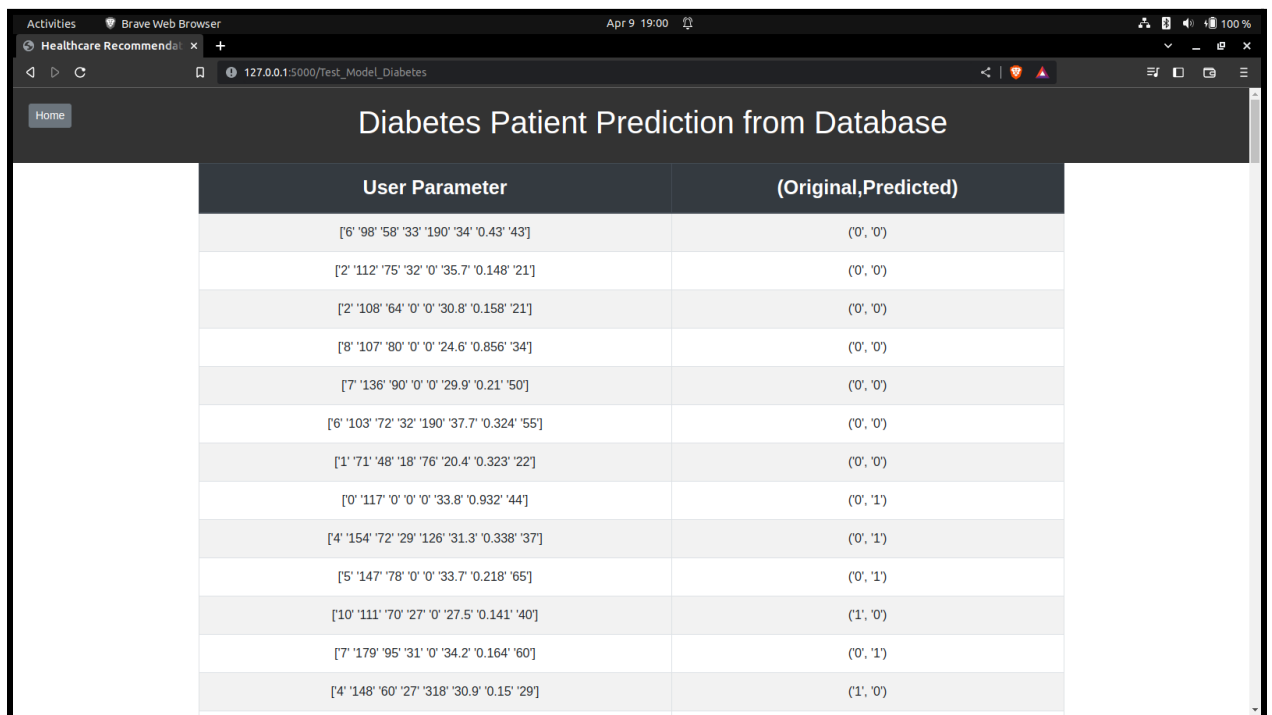
7) Now we will have a look at the “View Predictions” tab. On clicking the button, the 20% part of our data, which was our testing data, will be output on the screen.



8) The data of our predictions is displayed on the screen in two columns. The left column depicts the user parameters that were available to us for the training, the right column depicts a tuple consisting of a pair of boolean variables. The first value of the tuple is the original value, and the second value is the predicted value. '0' stands for "not having disease" and '1' stands for "having disease".

For example, if the tuple is ('0','0'), then the original data contained that the person does not have the illness, and our model predicts that he will not have the illness.

If the tuple is ('0','1'), then the original data stated that the person does not have the illness, but our model predicts with ~86% accuracy that the person will have illness.



User Parameter	(Original,Predicted)
[6' '98' '58' '33' '190' '34' '0.43' '43]	('0', '0')
[2' '112' '75' '32' '0' '35.7' '0.148' '21]	('0', '0')
[2' '108' '64' '0' '0' '30.8' '0.158' '21]	('0', '0')
[8' '107' '80' '0' '0' '24.6' '0.856' '34]	('0', '0')
[7' '136' '90' '0' '0' '29.9' '0.21' '50]	('0', '0')
[6' '103' '72' '32' '190' '37.7' '0.324' '55]	('0', '0')
[1' '71' '48' '18' '76' '20.4' '0.323' '22]	('0', '0')
[0' '117' '0' '0' '0' '33.8' '0.932' '44]	('0', '1')
[4' '154' '72' '29' '126' '31.3' '0.338' '37]	('0', '1')
[5' '147' '78' '0' '0' '33.7' '0.218' '65]	('0', '1')
[10' '111' '70' '27' '0' '27.5' '0.141' '40]	('1', '0')
[7' '179' '95' '31' '0' '34.2' '0.164' '60]	('0', '1')
[4' '148' '60' '27' '318' '30.9' '0.15' '29]	('1', '0')

## Explaining the codes

### Establishing connection with MongoDB database

```
from pymongo import MongoClient

client=MongoClient("mongodb://localhost:27017/")
db = client['ADBS_Dataset']
collection = db['diabetes']
```

### Loading CSV file into MongoDB database

```
# this is for diabetes
header= [ "Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "
csvfile = open('/home/bhagirath/Downloads/ADBS/Assignment2/Data/diabetes.csv', 'r'

reader = csv.DictReader( csvfile )

for each in reader:
    row={}
    for field in header:
        row[field]=each[field]

    collection.insert_one(row)
```

### Reading data from MongoDB database

```
# Define the projection to exclude the "_id" field
projection = {"_id": 0}

# Execute a query with the projection
cursor = collection.find({}, projection)

# Create an empty list to store the results
data = []

# Iterate over the cursor and append each document to the list
for document in cursor:
    data.append(document)
```

## Processing data:

We are Extracting data into X axis and Y axis

After that we split data into parts of 80-20,

80% for training purpose and 20% for testing purpose

```
# ***** Processing Data *****
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

# Extract the features and labels from the data
X = []
Y = []

for feature in data:
    X.append([feature['Pregnancies'], feature['Glucose'], feature['BloodPressure'], feature['SkinThickness'], feature['Insulin'], feature['DiabetesPedigree'], feature['Outcome']])
    Y.append(feature['Outcome'])

# print(X[0])
# print(Y[0])

X = np.array(X)

# Split the data into training and test sets
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, random_state=42)
```

Training the model with training data using SVM (support vector machines) technique,

and using *pickle* library to save the trained model to the disk

```
# ***** Training Model *****
import pickle
from sklearn.svm import SVC

svm_model = SVC(kernel='linear', C=1, gamma='scale')
svm_model.fit(X_train, y_train)

pickle.dump(svm_model, open("../SavedModel/Diabetes_classifier", 'wb'))
```

## Testing model accuracy with testing data

```
# ***** Testing Model Accuracy *****  
# Evaluate the performance of the model on test data  
from sklearn.metrics import accuracy_score  
  
y_pred = svm_model.predict(X_test)  
accuracy = accuracy_score(y_test, y_pred)  
  
print(f"Accuracy: {accuracy}")
```

## Flask Backend Prediction:

This is for “View Prediction”

In this we are calling function of other python file which will return the list containing original and predicted result of user data

```
# ***** this is for Diabetes *****  
@app.route("/Test_Model_Diabetes", methods=['GET', 'POST'])  
def Test_Model_Diabetes():  
    result=diabetes_prediction_from_database()  
    return render_template('show_from_database.html',Disease='Diabetes',result=result)
```

```
y_pred = svm_model.predict(X_test)  
result= list(zip(y_test, y_pred))  
result= list(zip(X_test, result))
```

## This is for “Enter your Details”

First we are getting the data from the form presented to the user

```
# ***** this is for Diabetes_Prediction *****
@app.route('/Diabetes_Prediction', methods=['POST'])
def Diabetes_Prediction():
    # Get the form data from the request object
    pregnancies = float(request.form['pregnancies'])
    glucose = float(request.form['glucose'])
    bp = float(request.form['blood-pressure'])
    skin = float(request.form['skin-thickness'])
    Insulin = float(request.form['Insulin'])
    bmi = float(request.form['BMI'])
    dpf = float(request.form['diabetespedigreefunction'])
    age = float(request.form['age'])
```

Loading the model that we have trained previously

Passing that input data to model for prediction and return it to the webpage

```
# Pass the form data to the machine learning model for prediction
input_data = [[pregnancies, glucose, bp, skin, Insulin, bmi, dpf, age]]
# input_data = [[1, 90, 70, 29, 0, 27.5, 0.4, 31]]

Diabetes_classifier = pickle.load(open("../SavedModel/Diabetes_classifier", 'rb'))
prediction = Diabetes_classifier.predict(input_data)
# print(prediction)

# if (prediction[0] == '0'):
#     return ('The person is not diabetic')
# else:
#     return ('The person is diabetic')

return render_template('test_result.html', Disease='Diabetes', Result=prediction[0])
```