Internship project on

AILO: DISEASE PREDICTION

Submitted to

The Department of Computer Science and Engineering

Bachelor of Technology In Computer Science and Engineering

K.SANAJAY	23R11A05B1
K.KEERTHANA	23R11A05B3
L.GABRIEL	23R11A05C1

Under the Guidance of

Adgaonker Shashank

Department of Computer Science and Engineering

Geethanjali College of Engineering and Technology (UGC Autonomous)

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi, Accredited by NBA)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

June-2025

TABLE OF CONTENTS

ABSTRACT:

In recent years, the integration of Artificial Intelligence (AI) into healthcare has opened new possibilities for early disease detection and personalized medical care. This project, titled AILO (Artificial Intelligence for Logical Observation), presents a machine learning-based system that predicts diseases based on user-input symptoms, offering an accessible and efficient diagnostic aid.

The system allows users to select symptoms from a predefined list through an intuitive web interface. Once the symptoms are submitted, the backend processes them using multiple machine learning algorithms including Decision Tree, Random Forest, and Naive Bayes. Each algorithm independently analyzes the input data and generates its prediction. The system then displays the output from all three models, helping to cross-verify and increase the confidence in the final diagnosis. In the test case shown, all algorithms consistently predicted the disease as Gastroenteritis, demonstrating a strong consensus among the models.

Key features of AILO include:

1.User-Friendly Interface: A clean and simple UI for symptom selection and result visualization.
2.Multi-Model Diagnosis: Utilizes multiple ML models to improve reliability and reduce the chances of misdiagnosis.
3.Scalable Design: Can be extended to support more diseases and incorporate advanced models like deep learning in the future.
4.Educational and Assistive Tool: Useful for students, educators, and general users who seek preliminary insights into potential health issues.

The project was developed under the guidance of Adgaonker Shashank Sir, focusing on the practical implementation of data science in the healthcare domain. With the rising need for quick and remote healthcare solutions, AILO showcases how technology can assist in democratizing health diagnostics, especially in areas with limited access to medical professionals.

INTRODUCTION:

In today’s fast-paced world, timely and accurate disease detection has become essential to ensure better healthcare outcomes and reduce the burden on medical infrastructure. Our project, Disease Prediction System, aims to bridge the gap between symptoms and diagnosis using technology. It is a smart and efficient tool that helps users predict possible diseases based on the symptoms they input.

This system uses machine learning algorithms trained on large datasets of medical records and symptom patterns. By analyzing the symptoms provided by a user, the model predicts the most probable disease(s), helping individuals take early precautions or seek timely medical consultation.

The project not only supports awareness but also empowers users to become more proactive about their health. With a user-friendly interface, quick response, and reliable accuracy, this system can serve as an initial screening tool—especially useful in remote or under-resourced areas.

PROBLEM STATEMENT:

In many parts of the world, especially in remote or under-resourced areas, individuals often lack immediate access to professional medical advice. Early symptoms of diseases are frequently ignored or misinterpreted, leading to delayed diagnoses, worsened health outcomes, and increased pressure on healthcare systems.

Moreover, with the overwhelming amount of health information available online, users may feel confused or misled when trying to understand their symptoms. There is a pressing need for a reliable, intelligent, and accessible solution that can assist individuals in understanding potential health issues at an early stage. Our project addresses this gap by creating a **Disease Prediction System** that uses **machine learning algorithms** to predict the most probable disease(s) based on user-input symptoms. This solution aims to assist users in taking timely action, improving health awareness, and reducing diagnostic delays.

SOLUTION

To address the challenges of delayed medical diagnosis and limited access to healthcare professionals, especially in remote or underserved areas, we propose a smart and scalable **Disease Prediction System** that leverages the power of **machine learning** and **data analysis**.

🔍 1. User Symptom Input Module

- A user-friendly interface allows users to select or type in their current symptoms.
- A well-organized symptom list ensures easy navigation and comprehensive coverage of common and critical signs.
- Input is sanitized and pre-processed to match symptom patterns in the training dataset.

▯ 2. Machine Learning-Based Prediction Engine

- The system uses supervised learning algorithms (e.g., **Decision Tree**, **Random Forest**, or **Naive Bayes**) trained on medical datasets that map symptoms to diseases.
- Feature extraction is performed from symptom inputs, and the model classifies the most likely disease(s).
- The model is trained on a dataset consisting of thousands of entries with symptoms and corresponding diagnoses.
- Probabilistic confidence scores are shown, indicating how certain the system is about each prediction.

🌐 3. Web-Based Deployment

- The system is accessible via a web application built using frameworks like **Flask (Python)** for backend integration and **HTML/CSS/JavaScript** for the front end.
- Users can access the system on mobile phones or computers without the need for installation.
- All computations happen server-side, ensuring lightweight front-end performance.

📊 4. Prediction Output and Guidance

- The user receives a list of predicted diseases ranked by likelihood.
- For each disease, the system provides:
    - A brief description
    - Common symptoms

    - Recommended next steps (e.g., "Consult a physician", "Get tested for XYZ", etc.)
- Optional: The system can include links to medical resources or connect to teleconsultation services.

🔐 5. Data Privacy and Ethical Considerations

- No personal data is stored unless the user explicitly consents.
- Predictions are not meant to replace professional medical advice but serve as an **initial self-assessment**.
- The model is continuously improved using anonymized and aggregated usage data (if permitted by the user).

✅ 6. Benefits of the System

- **Time-Efficient**: Gives quick and accurate predictions, reducing the time between symptom onset and seeking help.
- **Accessible**: Designed for both urban and rural populations with low system requirements.
- **Scalable**: Can be adapted to include more diseases, languages, or region-specific symptom variations.
- **Educational**: Encourages health awareness and literacy among users.
- **Cost-Effective**: Reduces unnecessary doctor visits for minor conditions and prioritizes medical attention for severe issues.

PROGRAM:

HOME.CSS:

```
        *{
   padding:0;
   margin: 0;
   box-sizing: border-box;
}

body{
   font-family: 'Poppins', sans-serif;
   overflow: hidden;
   background-image: url("b1.webp");
   background-size: cover;
}

.wave{
   position: fixed;
   bottom: 0;
   left: 0;
   height: 100%;
   z-index: -1;
}

.container{
   width: 100vw;
   height: 100vh;
   display: grid;
   grid-template-columns: repeat(2, 1fr);
   grid-gap :7rem;
   padding: 0 2rem;
}

.img{
   align-items: center;
   padding-left: 30%;
   padding-top: 50%;
   font-family: 'Times New Roman', serif;
}
form{
   width: 360px;
}
```

```
.login-content
{
    align-items: center;
    padding-top: 40%;
    padding-left: 20%;
}



.card {
    /* Add shadows to create the "card" effect */
    box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
    transition: 0.3s;
    background-color: white;
    width: fit-content;
    border-radius: 2%
  }

  /* On mouse-over, add a deeper shadow */
  .card:hover {
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
  }

  /* Add some padding inside the card container */
  .container2 {
    padding: 2px 16px;
    font-family: 'Brush Script MT', cursive;
    text-decoration: underline;

  }

@media screen and (max-width: 1050px){
    .container{
        grid-gap: 5rem;
    }
}

@media screen and (max-width: 1000px){
    form{
        width: 290px;
    }

    .login-content h2{
        font-size: 2.4rem;
        margin: 8px 0;
    }

    .img img{
        width: 400px;
    }
}

@media screen and (max-width: 900px){
    .container{
        grid-template-columns: 1fr;
    }

    .img{
        display: none;
    }

    .wave{
        display: none;
    }

    .login-content{
        justify-content: center;
    }
}
```

PREDSTYL.CSS:

```
body{
   background-color: powderblue;
}
#customers {
   font-family: Arial, Helvetica, sans-serif;
   border-collapse: collapse;
   width: 100%;
  }

  #customers td, #customers th {
   border: 1px solid #ddd;
   padding: 8px;
  }

  #customers tr:nth-child(even){background-color: #f2f2f2;}

  #customers tr:hover {background-color: #ddd;}

  #customers th {
   padding-top: 12px;
   padding-bottom: 12px;
   text-align: left;
   background-color: #04AA6D;
   color: white;
  }
  /* Style the header with a grey background and some padding */
.header {
   overflow: hidden;
   background-color: #f1f1f1;
   padding: 20px 10px;
  }
  h1{
   font-size: 30px;
   font-weight: 600;
   background-image: linear-gradient(to left, rgba(0, 0, 0, 0),grey);
   color: transparent;
   background-clip: text;
   -webkit-background-clip: text;
   padding-left: 2px;
  }
  /* Style the header links */
  .header a {
    float: left;
    color: black;
    text-align: center;
    padding: 12px;
    text-decoration: none;
    font-size: 18px;
    line-height: 25px;
    border-radius: 4px;
  }

  /* Style the logo link (notice that we set the same value of line-height and font-size to
prevent the header to increase when the font gets bigger */
  .header a.logo {
    font-size: 25px;
    font-weight: bold;
  }

  /* Change the background color on mouse-over */
  .header a:hover {
    background-color: #ddd;
    color: black;
  }

  /* Style the active/current link*/
  .header a.active {
    background-color: dodgerblue;
    color: white;
  }

  /* Float the link section to the right */
  .header-right {
    float: right;
  }

  /* Add media queries for responsiveness - when the screen is 500px wide or less, stack the
links on top of each other */
  @media screen and (max-width: 500px) {
    .header a {
      float: none;
      display: block;
      text-align: left;
    }
    .header-right {
      float: none;
    }
  }
```

STYLES.CSS:

```
html{
   height: 100%;
   margin: 0;
}

body{
   font-family: Arial, Helvetica,sans-serif;
   text-align: center;
   margin: 0;
   padding: 0;
   width: 100%;
   height: 100%;
   display: flex;
   flex-direction: column;
   background-color: powderblue;
}

.ml-container{
   align-items: center;
   padding-top: 10%;
   padding-left: 20%;

}
select {
   margin-bottom: 10px;
   margin-top: 10px;
   font-family: cursive, sans-serif;
   outline: 0;
   background: white;
   color: black;
   border: 1px solid crimson;
   padding: 4px;
   border-radius: 9px;
  }
  .button {
   background-color: #4CAF50; /* Green */
   border: none;
   color: white;
   padding: 16px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  transition-duration: 0.4s;
  cursor: pointer;
  background-color: #4CAF50;
  color: black;
  border: 2px solid white;
  border-radius: 4px;
}
.button:hover {
  background-color: white;
  color: #4CAF50;
}
```

APP.PY:

```
from symtable import symtable
from flask import Flask
import string
import numpy as np
import pandas as pd
from collections import Counter
from pandas import DataFrame, read_csv;
import sklearn
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)
def DecisionTree(s1,s2,s3,s4,s5):
    result=''
    from sklearn import tree
    clf3 = tree.DecisionTreeClassifier()
    clf3 = clf3.fit(X.values,y)
    from sklearn.metrics import accuracy_score
    y_pred=clf3.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    from sklearn.metrics import classification_report
    print(classification_report(y_test, y_pred))
    psymptoms = [s1,s2,s3,s4,s5]

    for k in range(0,len(symtoms)):
        for z in psymptoms:
            if(z==symtoms[k]):
                l2[k]=1

    inputtest = [l2]
    predict = clf3.predict(inputtest)
    predicted=predict[0]
    p2=clf3.predict_proba(inputtest)
    print(p2)

    h='no'
    for a in range(0,len(disease)):
        if(predicted == a):
            h='yes'
            break
    if (h=='yes'):
        result=disease[a]
    else:
        result="Not Found"
    print(result)
    return result
def randomforest(s1,s2,s3,s4,s5):
    result=''
    from sklearn.ensemble import RandomForestClassifier
    clf4 = RandomForestClassifier()
    clf4 = clf4.fit(X.values,np.ravel(y))
    from sklearn.metrics import accuracy_score
    y_pred=clf4.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    from sklearn.metrics import classification_report
    print(classification_report(y_test, y_pred))

    psymptoms = []
    psymptoms.append(s1)
    psymptoms.append(s2)
    psymptoms.append(s3)
    psymptoms.append(s4)
    psymptoms.append(s5)

    for k in range(0,len(symtoms)):
        for z in psymptoms:
            if(z==symtoms[k]):
                l2[k]=1

    inputtest = [l2]
    predict = clf4.predict(inputtest)
    predicted=predict[0]
    p2=clf4.predict_proba(inputtest)
    print(p2)

    h='no'
    for a in range(0,len(disease)):
        if(predicted == a):
            h='yes'
            break

    if (h=='yes'):
        result=disease[a]
    else:
        result="Not Found"
    print(result)
    return result
def NaiveBayes(s1,s2,s3,s4,s5):
    result=''
    from sklearn.naive_bayes import GaussianNB
    gnb = GaussianNB()
    gnb=gnb.fit(X.values,np.ravel(y))
    from sklearn.metrics import accuracy_score
    y_pred=gnb.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    from sklearn.metrics import classification_report
    print(classification_report(y_test, y_pred))
    psymptoms = []
    psymptoms.append(s1)
    psymptoms.append(s2)
    psymptoms.append(s3)
    psymptoms.append(s4)
    psymptoms.append(s5)
    for k in range(0,len(symtoms)):
        for z in psymptoms:
            if(z==symtoms[k]):
                l2[k]=1
    inputtest = [l2]
    predict = gnb.predict(inputtest)
    predicted=predict[0]

    h='no'
    for a in range(0,len(disease)):
        if(predicted == a):
            h='yes'
            break

    if (h=='yes'):
        result=disease[a]
    else:
        result="Not Found"
    print(result)
    return result
def Logistic(s1,s2,s3,s4,s5):
    result=''
    from sklearn.linear_model import LogisticRegression
    clf5 = LogisticRegression()
    clf5 = clf5.fit(X.values,y)
    from sklearn.metrics import accuracy_score
    y_pred=clf5.predict(X_test)
    print(accuracy_score(y_test, y_pred))
    from sklearn.metrics import classification_report
    print(classification_report(y_test, y_pred))

    psymptoms = [s1,s2,s3,s4,s5]

    for k in range(0,len(symtoms)):
        for z in psymptoms:
            if(z==symtoms[k]):
                l2[k]=1

    inputtest = [l2]
    predict = clf5.predict(inputtest)
    predicted=predict[0]
    p2=clf5.predict_proba(inputtest)
    print(p2)

    h='no'
    for a in range(0,len(disease)):
        if(predicted == a):
            h='yes'
            break

    if (h=='yes'):
        result=disease[a]
    else:
        result="Not Found"
    print(result)
    return result
disease=['Fungal infection','Allergy','GERD','Chronic cholestasis','Drug Reaction',
'Peptic ulcer diseae','AIDS','Diabetes','Gastroenteritis','Bronchial Asthma','Hypertension',
' Migraine','Cervical spondylosis',
'Paralysis (brain hemorrhage)','Jaundice','Malaria','Chicken pox','Dengue','Typhoid','hepatitis
A',
'Hepatitis B','Hepatitis C','Hepatitis D','Hepatitis E','Alcoholic hepatitis','Tuberculosis',
'Common Cold','Pneumonia','Dimorphic hemmorhoids(piles)',
'Heartattack','Varicoseveins','Hypothyroidism','Hyperthyroidism','Hypoglycemia','Osteoarthri
stis',
'Arthritis','(vertigo) Paroymsal  Positional Vertigo','Acne','Urinary tract infection','Psoriasis',
'Impetigo']
symtoms=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',
'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_in_limbs',
'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','bloody_stool',
'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesity','swollen_legs',
'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittle_nails',
'swollen_extremeties','excessive_hunger','extra_marital_contacts','drying_and_tingling_lips',
'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_neck','swelling_joints',
'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness',
'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_smell_of urine',
'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_look_(typhos)',
'depression','irritability','muscle_pain','altered_sensorium','red_spots_over_body','belly_pain',
'abnormal_menstruation','dischromic
_patches','watering_from_eyes','increased_appetite','polyuria','family_history','mucoid_sputu
m',
'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',
'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_c
alf',
'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',
'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_no
se',
'yellow_crust_ooze']
l2=[]
for x in range(0,len(symtoms)):
    l2.append(0)
df=pd.read_csv("Training.csv")
print(df)
df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic
cholestasis':3,'Drug Reaction':4,
'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial
Asthma':9,'Hypertension ':10,
'Migraine':11,'Cervical spondylosis':12,
'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken
pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic
hepatitis':24,'Tuberculosis':25,
'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
attack':29,'Varicose veins':30,'Hypothyroidism':31,
'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
'(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract
infection':38,'Psoriasis':39,
'Impetigo':40}},inplace=True)
X= df[symtoms]
y = df[["prognosis"]]
np.ravel(y)
tr=pd.read_csv("Testing.csv")
tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic
cholestasis':3,'Drug Reaction':4,
'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial
Asthma':9,'Hypertension ':10,
'Migraine':11,'Cervical spondylosis':12,
'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken
pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic
hepatitis':24,'Tuberculosis':25,
'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
attack':29,'Varicose veins':30,'Hypothyroidism':31,
'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
'(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract
infection':38,'Psoriasis':39,
'Impetigo':40}},inplace=True)
X_test= tr[symtoms]
y_test = tr[["prognosis"]]
np.ravel(y_test)
# result=[]
#
result.append(DecisionTree("mild_fever","neck_pain","runny_nose","cramps","abdominal_
pain"))
#
result.append(randomforest("mild_fever","neck_pain","runny_nose","cramps","abdominal_
pain"))
#
result.append(NaiveBayes("mild_fever","neck_pain","runny_nose","cramps","abdominal_pa
in"))
#
result.append(Logistic("mild_fever","neck_pain","runny_nose","cramps","abdominal_pain")
)
# print(result)
@app.route("/")
def main():
   return render_template("home.html")
@app.route("/select")
def select():
   return render_template("index.html",d=symtoms)
@app.route("/predict",methods=['POST'])
def result():
   result=[]
   s1=request.form['s1']
   print(s1)
   s2=request.form['s2']
   print(s2)
   s3=request.form['s3']
   print(s3)
   s4=request.form['s4']
   print(s4)
   s5=request.form['s5']
   result.append(DecisionTree(s1,s2,s3,s4,s5))
   result.append(randomforest(s1,s2,s3,s4,s5))
   result.append(NaiveBayes(s1,s2,s3,s4,s5))
   # result.append(Logistic(s1,s2,s3,s4,s5))
   print(result)
   # p={}
   # s=set()
   # for i in result:
   #     if i not in s:
   #         p[i]=result.count(i)
   #         s.add(i)
   # w={}
   # print(p)
   # for i in p:
   #     w[i]=(p[i]/3)*100
   # print(w)
   s=set(result)
   return render_template("result.html",prediction=result,k=s)
if __name__ == "__main__":
   app.run(debug=True)
```

HOME.HTML:

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Form</title>
   <link rel="stylesheet" href="{{url_for('static',filename='home.css') }}">
   <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
   <script src="https://kit.fontawesome.com/a81368914c.js"></script>
   <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
   <img class="wave" src="">
   <div class="container">
      <div class="img">
         <br>
         <h2>DISEASE PREDICTION SYSTEM</h2>
         <br>
         <k>This website where one can select the symptoms he/she is facing and accordingly
the system shall
         predict a certain disease or condition he/she might be suffering from.</k>
      </div>
      <div class="login-content">
         <div class="card">
            <div class="container2">
               <a href="/select">Select Symptoms</a>
              </div>
              <br>
            <img src="static/download.png" alt="Avatar" style="width:100% ; align-content:
center;padding-left: 10%;padding-right: 10%;">
            </div>
         <!-- <form action="\login">
            <h2 class="title">Welcome to sr-jr Portal</h2>
            <img src="static/img/google.png"><br>
            <input type="submit" class="btn" value="Login with Google">
         </form>
      </div> -->
   </div>
   <script type="text/javascript" src="static/js/main.js"></script>
</body>
</html>
```

INDEX.HTML:

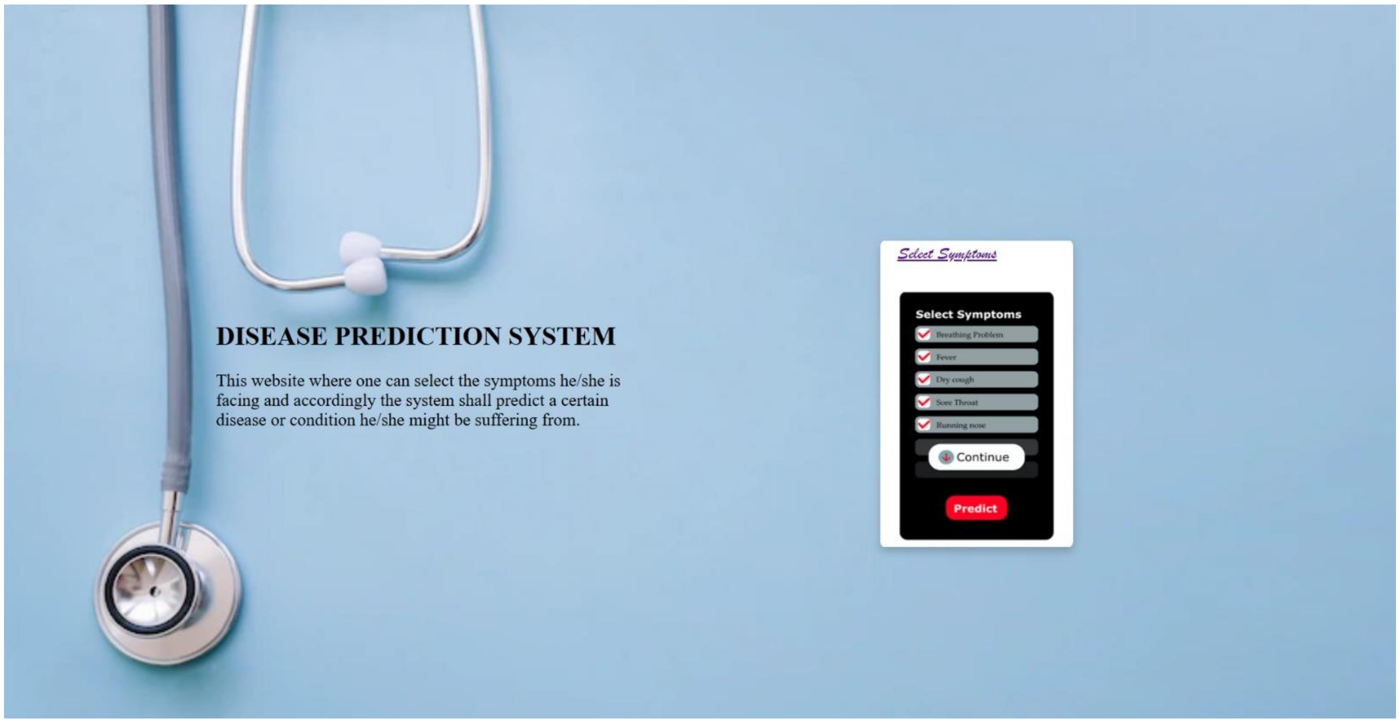
```
<!DOCTYPE html>

<html lang="en" dir="ltr">
   <head>
      <meta charset="utf-8">
      <title>Disease Predictor</title>
      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css')
}}">
      <script src="https://kit.fontawesome.com/5f3f547070.js"
crossorigin="anonymous"></script>
      <link href="https://fonts.googleapis.com/css2?family=Yatra+One&display=swap"
rel="stylesheet">
   </head>

   <body>

      <div class="home">
      <div class="img" >
         <img src="static\doctor.jpg" alt="'gif" style="width:700px;height:700px;
float:left;transform:scale(-1, 1);"></img>
      </div>
      <div class="ml-container">
         <h1>Select Symptoms</h1>
         <form method="post" action="/predict">
            <select id="s1" name="s1">
               {%for i in d%}
               <option>{{i}}</option>
               {%endfor%}
            </select>
            <br>
            <select id="s2" name="s2">
               {%for i in d%}
               <option>{{i}}</option>
               {%endfor%}
            </select>
            <br>
            <select id="s3" name="s3">
               {%for i in d%}
               <option>{{i}}</option>
```

```
                {%endfor%}
            </select>
            <br>
            <select id="s4" name="s4">
                {%for i in d%}
                <option>{{i}}</option>
                {%endfor%}
            </select>
            <br>
            <select id="s5" name="s5">
                {%for i in d%}
                <option>{{i}}</option>
                {%endfor%}
            </select>
            <br>
            <br>
            <input class="button" type="submit" value="Predict">
        </form>
    </div>
</div>

</body>
</html>
```

RESULT.HTML:

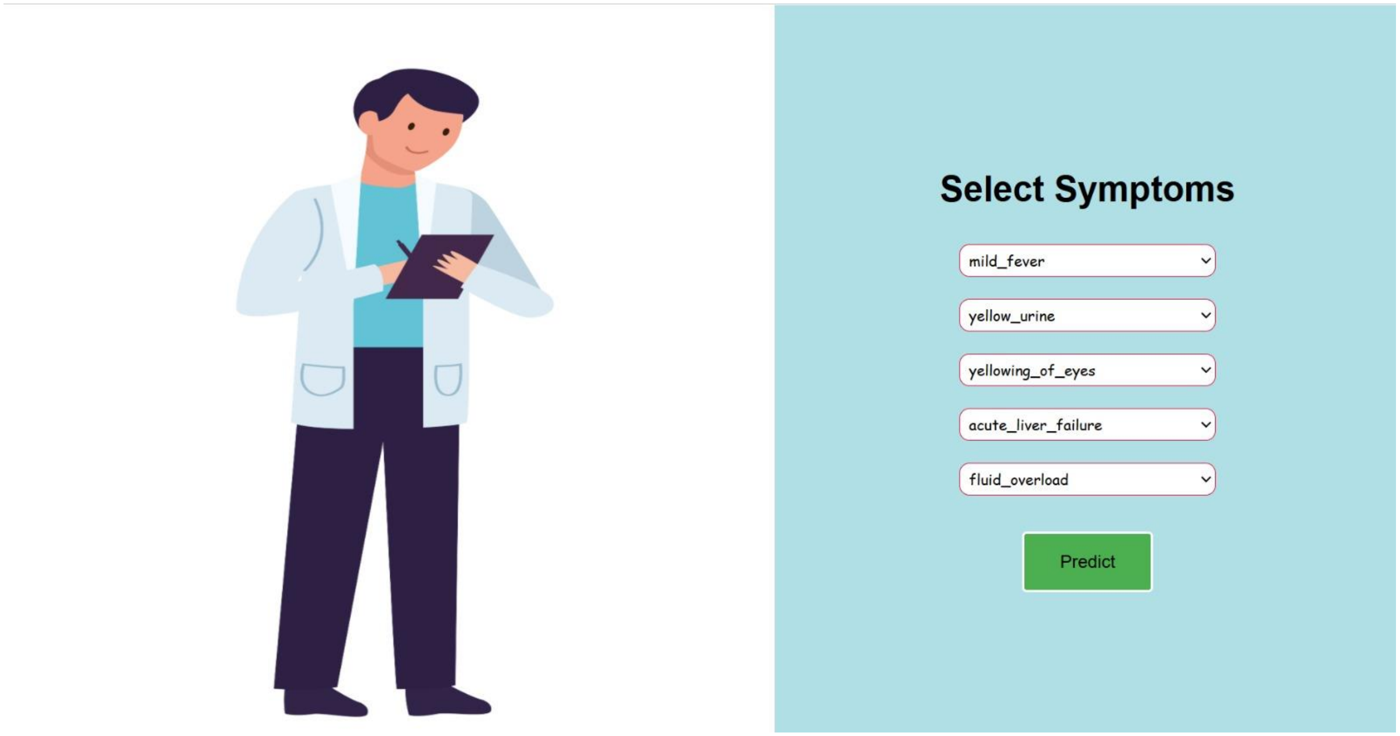
```
<!DOCTYPE html>

<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Disease Prediction</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='predstyle.css')
}}">
    <script src="https://kit.fontawesome.com/5f3f547070.js"
crossorigin="anonymous"></script>
    <link href="https://fonts.googleapis.com/css2?family=Yatra+One&display=swap"
rel="stylesheet">
  </head>

  <body>
    <div class="header">
      <a class="logo">Disease Predictor</a>
      <div class="header-right">
        <a class="active" href="/">Home</a>
      </div>
      </div>

<h1>Final Predictions</h1>

<table id="customers">
  <tr>
    <th>Alogrithm</th>
    <th>Prediction</th>
  </tr>
  <tr>
    <td>DECISION TREE</td>
    <td>{{prediction[0]}}</td>
  </tr>
  <tr>
    <td>RANDOM FOREST</td>
    <td>{{prediction[1]}}</td>
  </tr>
  <tr>
    <td>NAVIE BAYES</td>
    <td>{{prediction[2]}}</td>
  </tr>
</table>
<img src="static/f.svg" style="width :500px; align-items:center; padding-left:
30%;"></img>
  </body>
</html>
```

OUTPUT:



DISEASE PREDICTION SYSTEM

This website where one can select the symptoms he/she is facing and accordingly the system shall predict a certain disease or condition he/she might be suffering from.



Select Symptoms

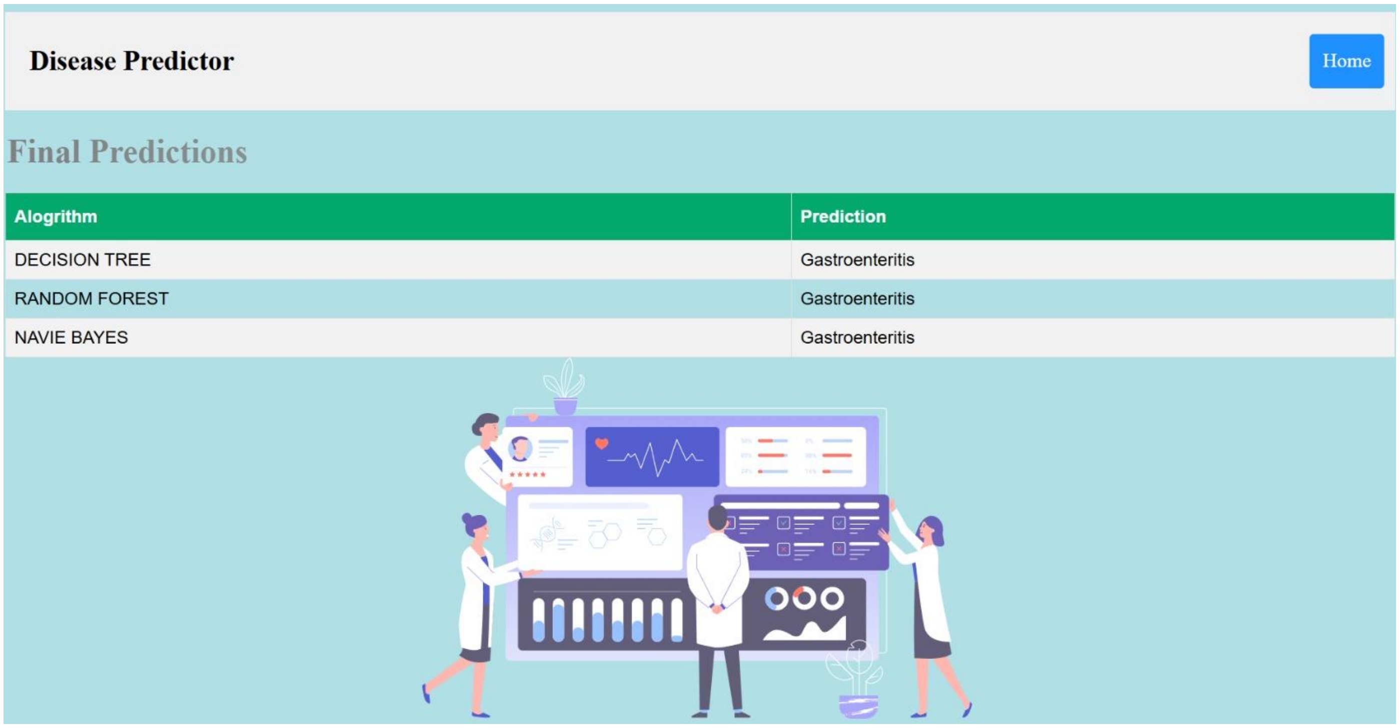
mild_fever

yellow_urine

yellowing_of_eyes

acute_liver_failure

fluid_overload



Disease Predictor

Final Predictions

Alogrithm	Prediction
DECISION TREE	Gastroenteritis
RANDOM FOREST	Gastroenteritis
NAVIE BAYES	Gastroenteritis

Machine Learning Algorithms

Machine learning algorithms are essentially sets of instructions that allow computers to learn from data, make predictions, and improve their performance over time without being explicitly programmed.

1. **Decision Trees**
A decision tree **splits data into branches based on feature values, creating a tree-like structure**.

- Each decision node represents a feature; leaf nodes provide the final prediction.
- The process continues until a final prediction is made at the leaf nodes
- Works for both classification and regression tasks.

2. **Naive Bayes**
Based on Bayes' theorem and assumes all features are independent of each other (hence "naive")

- Calculates probabilities for each class and assigns the most likely class to a data point.
- Assumption of feature independence might not hold in all cases (rarely true in real-world data)
- Works well for high-dimensional data.
- Commonly used in text classification tasks like spam filtering : Naive Bayes

3. **Random Forest**
Random forest is an ensemble method that combines **multiple decision trees.**

- Uses random sampling and feature selection for diversity among trees.
- Final prediction is based on majority voting (classification) or averaging (regression).
- **Advantages** : reduces overfitting compared to individual decision trees.
- Handles large datasets with higher dimensionality.