SOURCE CODE :

Parkinson_predict.ibynb

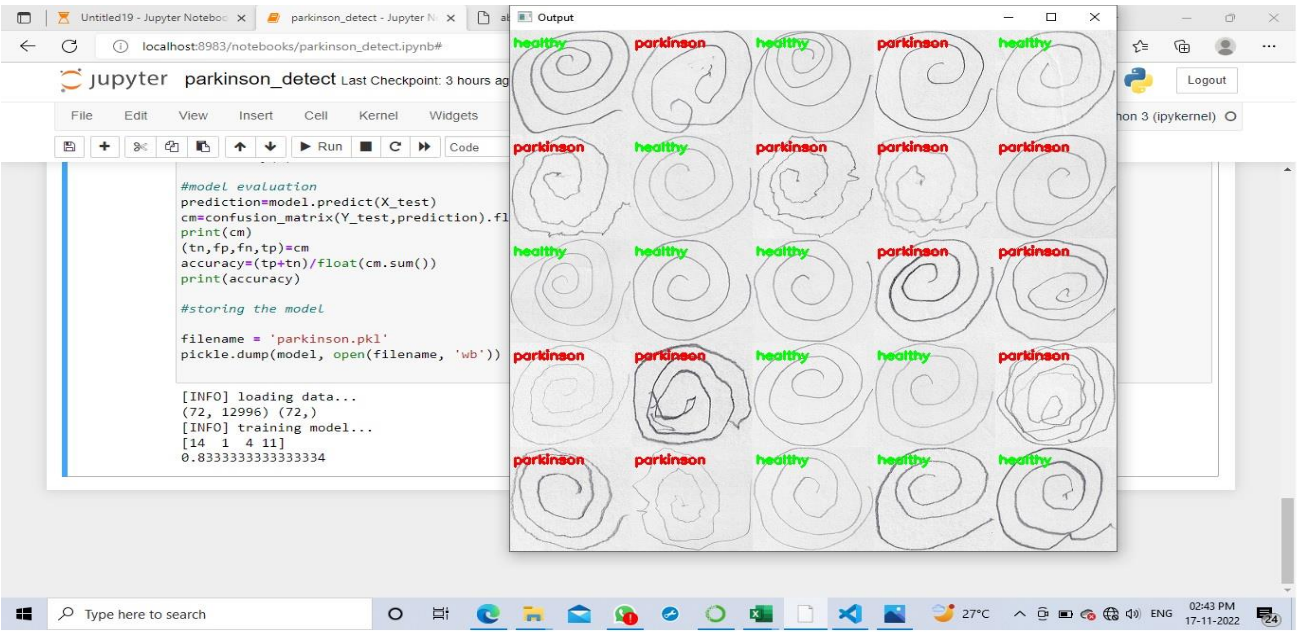
```
#import the pakages

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from skimage import feature
from imutils import build_montages
from imutils import paths
import numpy as np
import cv2
import os
import pickle

#Quatifying image
def quantify_image(image):
    #compute histogram of oriented gradients feature vector for the
input image
    features=feature.hog(image,orientations=9,pix-
els_per_cell=(10,10),cells_per_block=(2,2),trans-
form_sqrt=True,block_norm="L1")
    return features

def load_split(path):
    #grab list of images in the input dir,then initialize the list
of data and class labels

    imagepaths=list(paths.list_images(path))
    data,labels=[],[]

    #loop over the image path
    for imagepath in imagepaths:
        #extract the class label from the filename
        label=imagepath.split(os.path.sep)[-2]

        #load the input image
        image=cv2.imread(imagepath)
        image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        image=cv2.resize(image,(200,200))
        image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]

        #quantify the image
        features=quantify_image(image)
        #update the data and labels
        data.append(features)
        labels.append(label)

    return (np.array(data),np.array(labels))

# define path to train and test dir

trainingpath= r"Desktop/dataset/spiral/training"
testingpath=r"Desktop/dataset/spiral/testing"

#loading train and test data

print("[INFO] loading data...")
(X_train,Y_train)=load_split(trainingpath)
(X_test,Y_test)=load_split(testingpath)



#Label Encoding
le=LabelEncoder()
Y_train=le.fit_transform(Y_train)
Y_test=le.transform(Y_test)
print(X_train.shape,Y_train.shape)

#Training The Model

print("[INFO] training model...")
model=RandomForestClassifier(n_estimators=100)
model.fit(X_train,Y_train)

#testing the model
testingpath=list(paths.list_images(testingpath))
idxs=np.arange(0,len(testingpath))
idxs=np.random.choice(idxs,size=(25,),replace=False)
images=[]

#loop over the testing samples
for i in idxs:
    image=cv2.imread(testingpath[i])
    output=image.copy()

    # load the input image,convert to grayscale and resize

    output=cv2.resize(output,(128,128))
    image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    image=cv2.resize(image,(200,200))
    image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]
```

```
    #quantify the image and make predictions based on the  extracted feature using last trained random forest
    features=quantify_image(image)
    preds=model.predict([features])
    label=le.inverse_transform(preds)[0]
    #the set of output images
    if label=="healthy":
        color=(0,255,0)
    else:
        color=(0,0,255)

    cv2.putText(output,label,(3,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
    images.append(output)

#creating a montage
montage=build_montages(images,(128,128),(5,5))[0]
cv2.imshow("Output",montage)
cv2.waitKey(0)

#model evaluation
prediction=model.predict(X_test)
cm=confusion_matrix(Y_test,prediction).flatten()
print(cm)
(tn,fp,fn,tp)=cm
accuracy=(tp+tn)/float(cm.sum())
print(accuracy)

#storing the model

filename = 'parkinson.pkl'
pickle.dump(model, open(filename, 'wb'))
```

OUTPUT :

```
[INFO] loading data...
(72, 12996) (72,)
[INFO] training model...
[14  1  4 11]
0.8333333333333334
```

IMAGE PREPROCESSING OUTPUT:



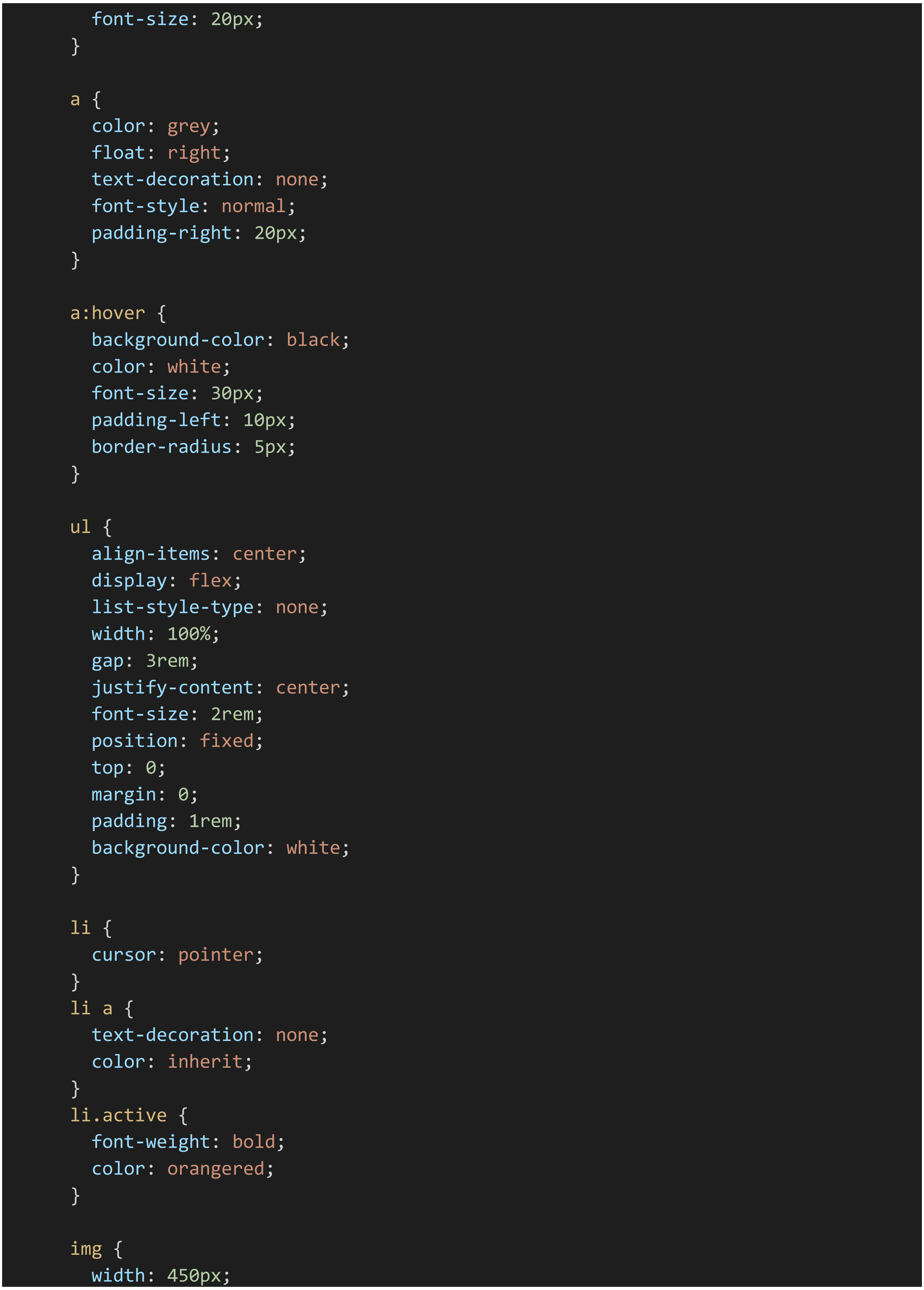
```
#model evaluation
prediction=model.predict(X_test)
cm=confusion_matrix(Y_test,prediction).fl
print(cm)
(tn,fp,fn,tp)=cm
accuracy=(tp+tn)/float(cm.sum())
print(accuracy)

#storing the model

filename = 'parkinson.pkl'
pickle.dump(model, open(filename, 'wb'))
```

```
[INFO] loading data...
(72, 12996) (72,)
[INFO] training model...
[14  1  4 11]
0.8333333333333334
```

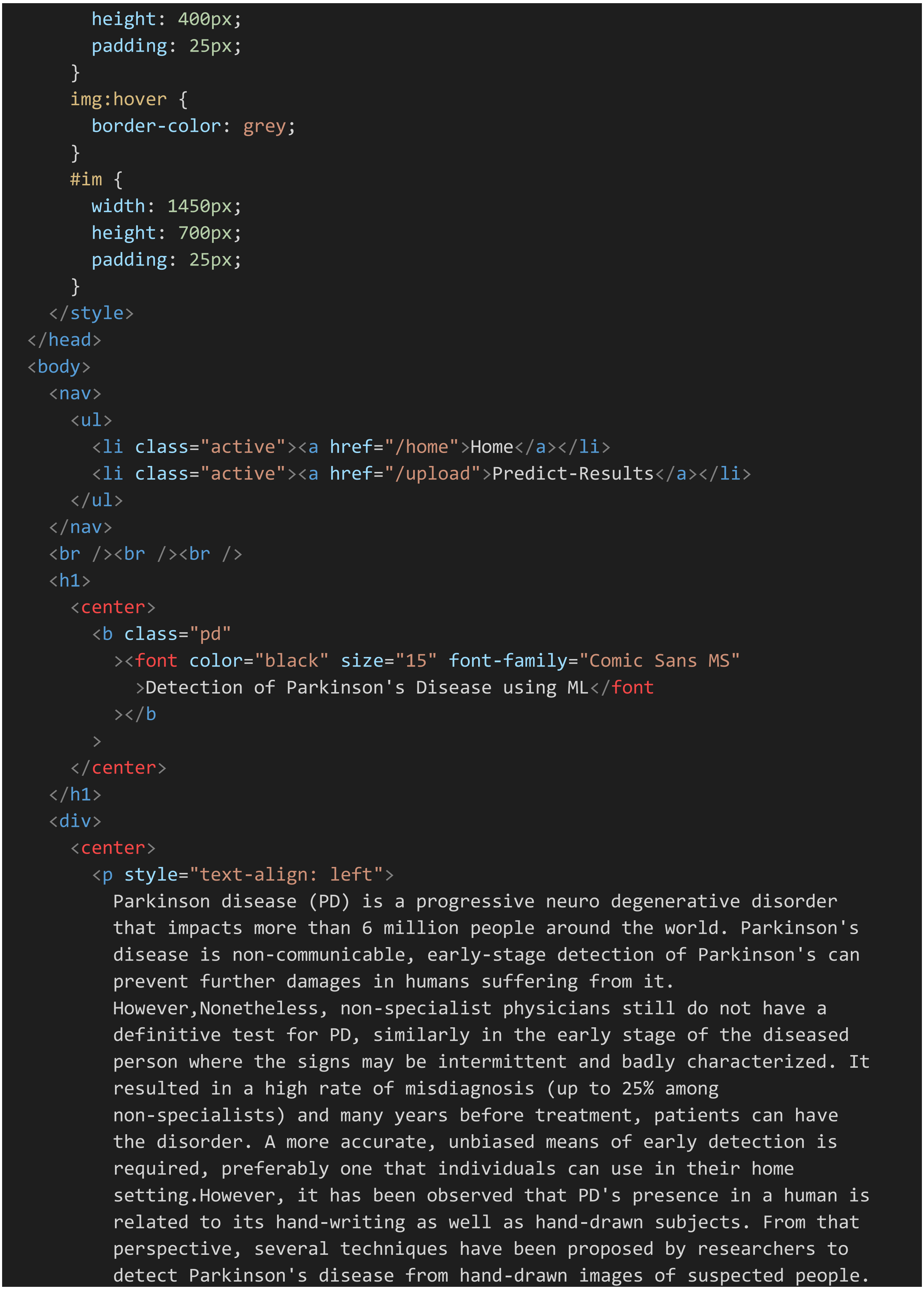
HTML CODES :

base.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>HomePage</title>
    <style>
      body {
        background: linear-gradient(to right, #33ccff 0%, #99ffcc 100%);
        background-size: cover;
        background-position: relative;
        background-repeat: no-repeat;
        height: 100%;
        width: 100%;
      }
      h3 {
        text-align: center;
        color: white;
      }
      .main {
        margin-top: 100px;
      }
      p {
        color: black;
        text-indent: 10px;
        margin: 10px;
```



```
      font-size: 20px;
    }

    a {
      color: grey;
      float: right;
      text-decoration: none;
      font-style: normal;
      padding-right: 20px;
    }

    a:hover {
      background-color: black;
      color: white;
      font-size: 30px;
      padding-left: 10px;
      border-radius: 5px;
    }

    ul {
      align-items: center;
      display: flex;
      list-style-type: none;
      width: 100%;
      gap: 3rem;
      justify-content: center;
      font-size: 2rem;
      position: fixed;
      top: 0;
      margin: 0;
      padding: 1rem;
      background-color: white;
    }

    li {
      cursor: pointer;
    }
    li a {
      text-decoration: none;
      color: inherit;
    }
    li.active {
      font-weight: bold;
      color: orangered;
    }

    img {
      width: 450px;
```



```
      height: 400px;
      padding: 25px;
    }
    img:hover {
      border-color: grey;
    }
    #im {
      width: 1450px;
      height: 700px;
      padding: 25px;
    }
  </style>
</head>
<body>
  <nav>
    <ul>
      <li class="active"><a href="/home">Home</a></li>
      <li class="active"><a href="/upload">Predict-Results</a></li>
    </ul>
  </nav>
  <br /><br /><br />
  <h1>
    <center>
      <b class="pd"
        ><font color="black" size="15" font-family="Comic Sans MS"
          >Detection of Parkinson's Disease using ML</font
        ></b
      >
    </center>
  </h1>
  <div>
    <center>
      <p style="text-align: left">
        Parkinson disease (PD) is a progressive neuro degenerative disorder
        that impacts more than 6 million people around the world. Parkinson's
        disease is non-communicable, early-stage detection of Parkinson's can
        prevent further damages in humans suffering from it.
        However,Nonetheless, non-specialist physicians still do not have a
        definitive test for PD, similarly in the early stage of the diseased
        person where the signs may be intermittent and badly characterized. It
        resulted in a high rate of misdiagnosis (up to 25% among
        non-specialists) and many years before treatment, patients can have
        the disorder. A more accurate, unbiased means of early detection is
        required, preferably one that individuals can use in their home
        setting.However, it has been observed that PD's presence in a human is
        related to its hand-writing as well as hand-drawn subjects. From that
        perspective, several techniques have been proposed by researchers to
        detect Parkinson's disease from hand-drawn images of suspected people.
```

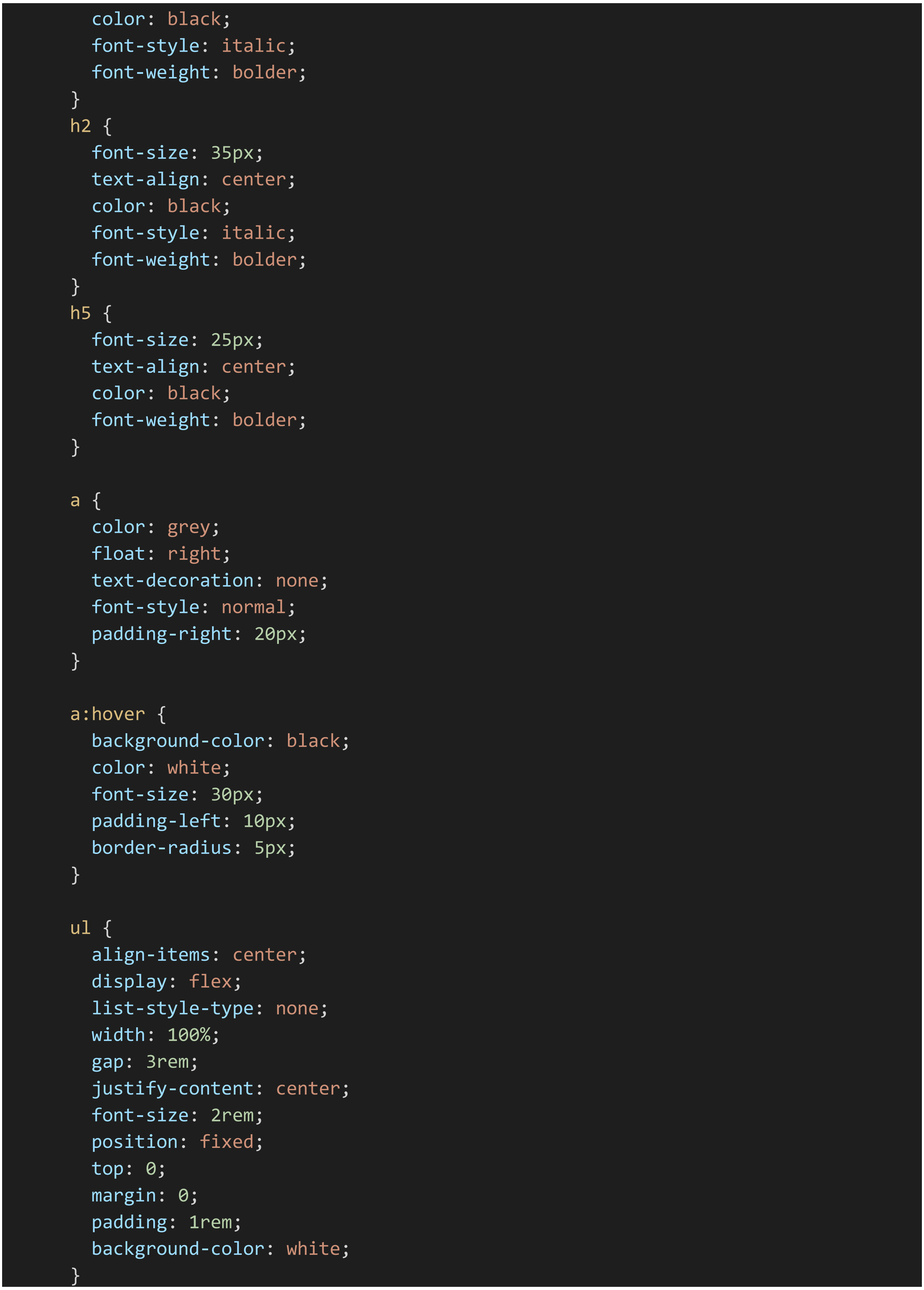
```
          But the previous methods have their constraints.
        </p>
      </center>
      <h4>
        <center>
          <b class="pd"
            ><font color="black" size="12" font-family="Comic Sans MS"
              >Causes and Symptoms of Parkinson's Disease</font
            >
          </b>
        </center>
      </h4>
      <span>
        <img
          src="https://www.narayanahealth.org/blog/wp-content/uploads/2015/04/parkinson.png"
          title="Disease"
        />
      </span>
      <span>
        <img
          src="https://stanfordmedicine25.stanford.edu/the25/parkinsondisease/_jcr_content/main/panel_builder_0/panel_0/panel_builder_0/panel_0/panel_builder/panel_0/image.img.476.high.png/1.png"
          title="Symptoms"
      /></span>
      <span
        ><img
          src="https://www.verywellhealth.com/thmb/Aaqo8oM3QDHSNHCt_DlKCNeWoUk=/1500x0/filters:no_upscale():max_bytes(150000):strip_icc()/zhansen-5200700_Finaledit2-3e7eb00f1bdb4806adb3f67ca4404894.jpg"
          title="Stages"
      /></span>
      <span
        ><img
          src="https://www.gutmicrobiotaforhealth.com/wp-content/uploads/2016/12/parkinson.jpg"
          title="Effect"
      /></span>
      <span
        ><img
          src="https://i.pinimg.com/originals/02/16/e4/0216e4b8a5db4d6e2a3f7043eaf7dc32.jpg"
          title="Cause"
      /></span>
      <span
        ><img
```

```
          src="https://jnnp.bmj.com/content/jnnp/91/8/795/F4.large.jpg"
          title="diagnosis"
      /></span>
      <h3>
        <center>
          <font color="black" size="12" font-family="Comic Sans MS"
            >Treatment for parkinson disease</font
          >
        </center>
      </h3>
      <span
        ><img
          src="https://www.mdpi.com/biomolecules/biomolecules-11-00612/article_deploy/html/images/biomolecules-11-00612-g001.png"
          title="diagnosis"
      /></span>
      <span
        ><img
          src="https://media.springernature.com/m685/springer-static/image/art%3A10.1038%2Fs41401-020-0365-y/MediaObjects/41401_2020_365_Fig1_HTML.png"
          title="diagnosis"
      /></span>
      <span
        ><img
          src="https://www.verywellhealth.com/thmb/BgjmOKb2W-7z0gqLZryKBd4FFHs=/1500x0/filters:no_upscale():max_bytes(150000):strip_icc()/advanced-parkinsons-disease-5200544_color_text_v1-3bc74418259340ceaf5f6d407daeff73.jpg"
          title="diagnosis"
      /></span>

      <h3>
        <center>
          <font color="black" size="12" font-family="Comic Sans MS"
            >How brains looks during PD?</font
          >
        </center>
      </h3>

      <span
        ><img
          id="im"
          src="https://ichef.bbci.co.uk/news/976/cpsprodpb/16161/production/_107456409_parkinsons.jpg"
          title="Stage"
      /></span>
      <span
        ><img
```

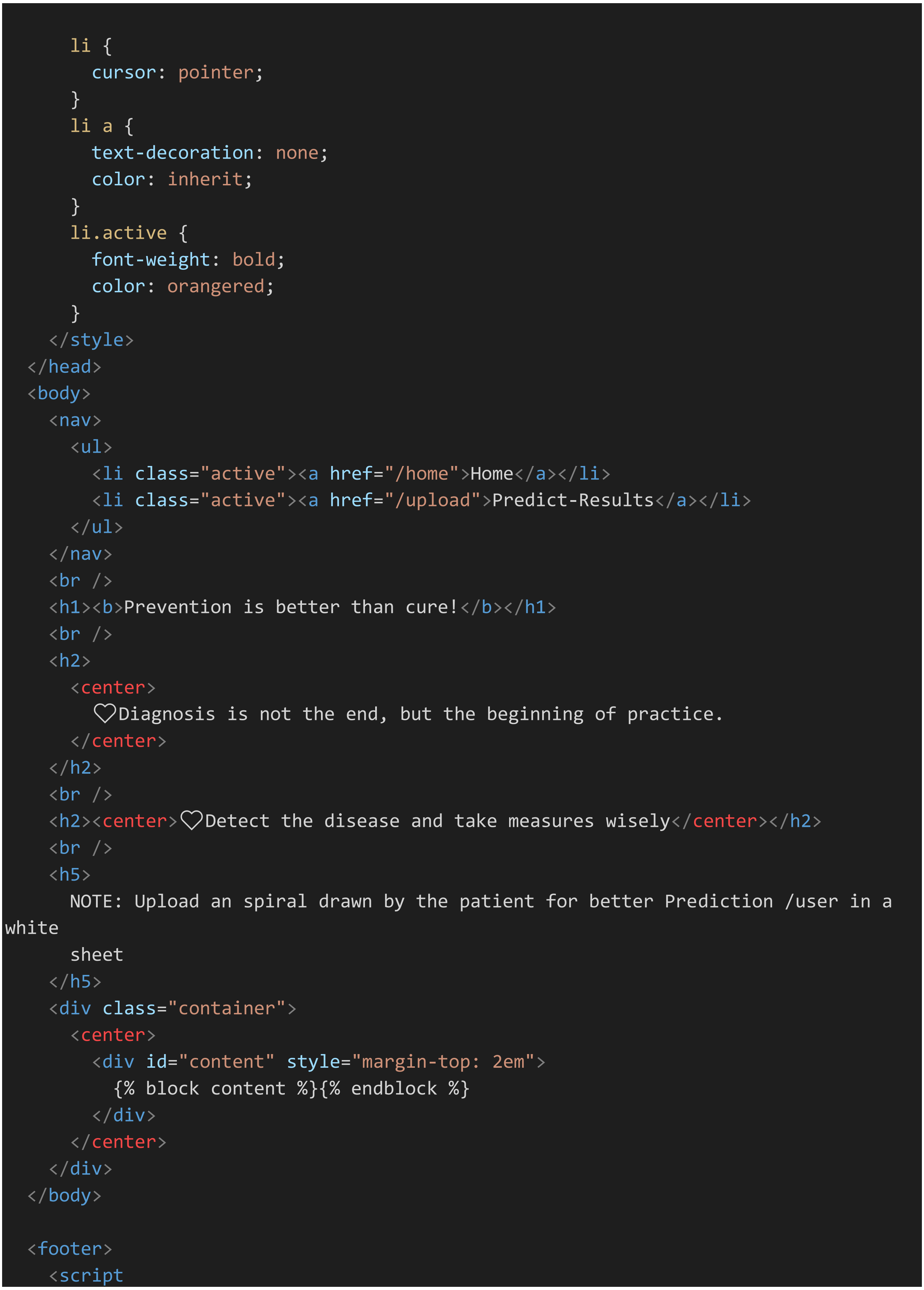
```
          id="im"
          src="https://img.parkinsonsinfoclub.com/wp-content/uploads/back-conditions-neck-conditions-london-back-pain-clinic-scaled.jpeg"
          title="Stage"
      /></span>
      <br /><br />
    </div>
  </body>
</html>
```

Base.html

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Predict</title>
    <link
      href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link
      href="{{ url_for('static', filename='css/main.css') }}"
      rel="stylesheet"
    />
    <style>
      body {
        background-image: url("https://img.freepik.com/free-vector/clean-medical-patterned-background-vector_53876-140867.jpg?w=1060&t=st=1667911964~exp=1667912564~hmac=4298568f384f42cfc60423d63ac6a8c806e4fe025c1bed2f32ae68b3f15b2139");
        background-position: center;
        background-repeat: no-repeat;
        background-size: cover;
        height: 100%;
        width: 100%;
      }
      h1 {
        font-size: 40px;
        text-align: center;
```



```
        color: black;
        font-style: italic;
        font-weight: bolder;
      }
      h2 {
        font-size: 35px;
        text-align: center;
        color: black;
        font-style: italic;
        font-weight: bolder;
      }
      h5 {
        font-size: 25px;
        text-align: center;
        color: black;
        font-weight: bolder;
      }

      a {
        color: grey;
        float: right;
        text-decoration: none;
        font-style: normal;
        padding-right: 20px;
      }

      a:hover {
        background-color: black;
        color: white;
        font-size: 30px;
        padding-left: 10px;
        border-radius: 5px;
      }

      ul {
        align-items: center;
        display: flex;
        list-style-type: none;
        width: 100%;
        gap: 3rem;
        justify-content: center;
        font-size: 2rem;
        position: fixed;
        top: 0;
        margin: 0;
        padding: 1rem;
        background-color: white;
      }
```



```
      li {
        cursor: pointer;
      }
      li a {
        text-decoration: none;
        color: inherit;
      }
      li.active {
        font-weight: bold;
        color: orangered;
      }
    </style>
  </head>
  <body>
    <nav>
      <ul>
        <li class="active"><a href="/home">Home</a></li>
        <li class="active"><a href="/upload">Predict-Results</a></li>
      </ul>
    </nav>
    <br />
    <h1><b>Prevention is better than cure!</b></h1>
    <br />
    <h2>
      <center>
        ♡Diagnosis is not the end, but the beginning of practice.
      </center>
    </h2>
    <br />
    <h2><center>♡Detect the disease and take measures wisely</center></h2>
    <br />
    <h5>
      NOTE: Upload an spiral drawn by the patient for better Prediction /user in a
white
      sheet
    </h5>
    <div class="container">
      <center>
        <div id="content" style="margin-top: 2em">
          {% block content %}{% endblock %}
        </div>
      </center>
    </div>
  </body>

  <footer>
    <script
```

```
      src="{{ url_for('static', filename='js/main.js') }}"
      type="text/javascript"
    ></script>
  </footer>
</html>
```

Pred.html

```

{% extends "base.html" %} {% block content %}

<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <center>
            <label for="imageUpload" class="upload-label">
                Choose...
            </label>
            <input type="file" name="file" id="imageUpload" accept=".png, .jpg,
.jpeg">
        </center>
    </form>

   <center> <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div></center>
        </div>
        <center>
            <div>
                <button type="button" class="btn btn-primary btn-lg " id="btn-pre-
dict">Predict!</button>
            </div>
        </center>
    </div>

    <div class="loader" style="display:none;"></div>

    <h3 id="result">
        <span> </span>
    </h3>

</div>

{% endblock %}
```

main.css

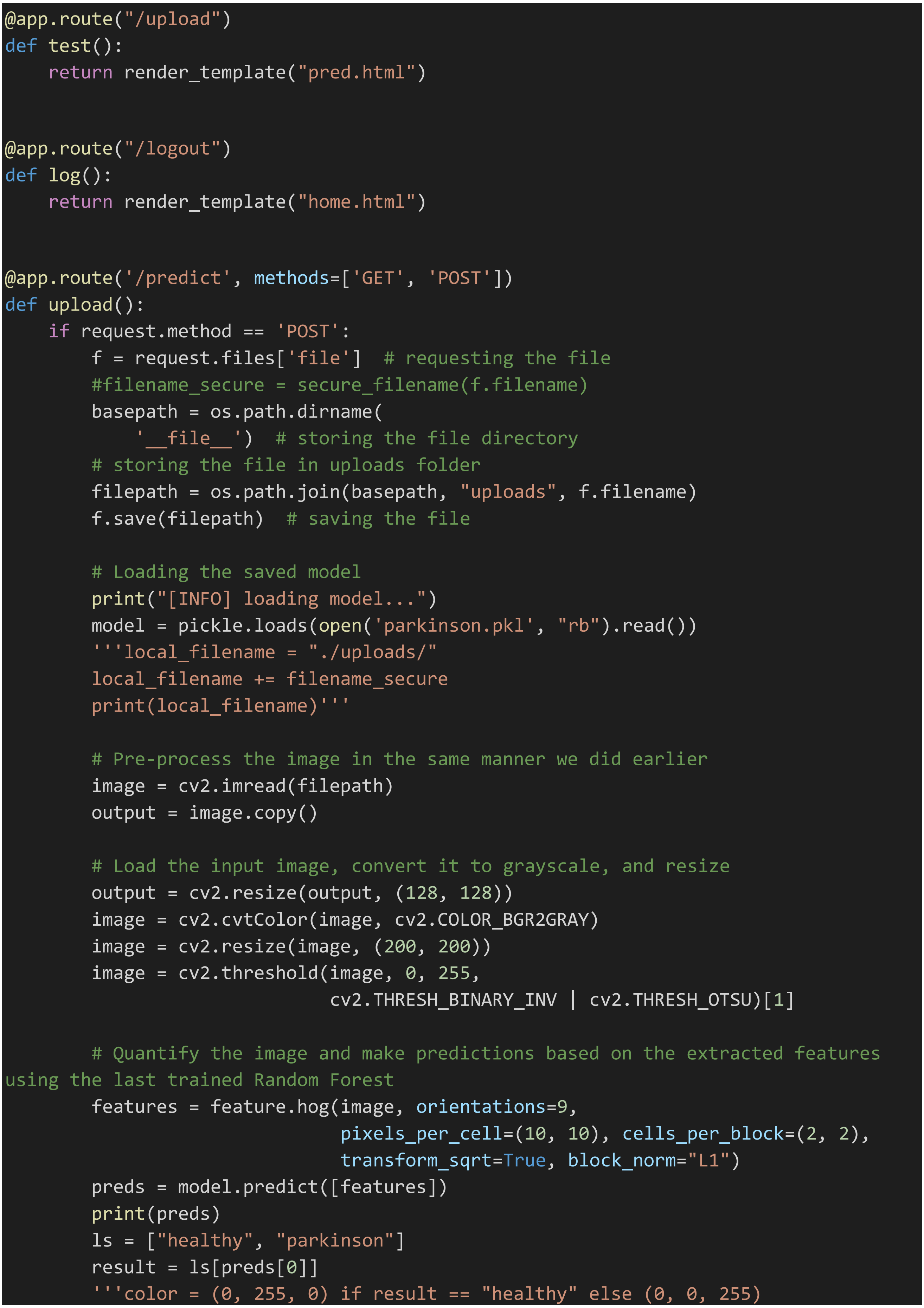
```
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label {
    display: inline-block;
    padding: 12px 30px;
    background: #fe2727;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}

.upload-label:hover {
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3;
    /* Light grey */
    border-top: 8px solid #3498db;
    /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}
```

Main.js

```
$(document).ready(function() {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function(e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result
+ ')');
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            };
            reader.readAsDataURL(input.files[0]);
        }
    }
    $("#imageUpload").change(function() {
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });

    // Predict
    $('#btn-predict').click(function() {
        var form_data = new FormData($('#upload-file')[0]);

        // Show loading animation
        $(this).hide();
        $('.loader').show();
        // Make prediction by calling api /predict
        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function(data) {
                // Get and display the result
                $('.loader').hide();
                $('#result').fadeIn(600);
                $('#result').text('Prediction : ' + data);
                console.log('Success!');
            },
        });
    });

});
```

app.py

```
from flask import Flask, request, render_template
import pickle
import cv2
from skimage import feature
import os.path
#from werkzeug.utils import secure_filename

#from model import model


app = Flask(__name__)


@app.route("/")
def about():
    return render_template("home.html")


@app.route("/home")
def home():
    return render_template("home.html")
```



```
@app.route("/upload")
def test():
    return render_template("pred.html")


@app.route("/logout")
def log():
    return render_template("home.html")


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']  # requesting the file
        #filename_secure = secure_filename(f.filename)
        basepath = os.path.dirname(
            '__file__')  # storing the file directory
        # storing the file in uploads folder
        filepath = os.path.join(basepath, "uploads", f.filename)
        f.save(filepath)  # saving the file

        # Loading the saved model
        print("[INFO] loading model...")
        model = pickle.loads(open('parkinson.pkl', "rb").read())
        '''local_filename = "./uploads/"
        local_filename += filename_secure
        print(local_filename)'''

        # Pre-process the image in the same manner we did earlier
        image = cv2.imread(filepath)
        output = image.copy()

        # Load the input image, convert it to grayscale, and resize
        output = cv2.resize(output, (128, 128))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (200, 200))
        image = cv2.threshold(image, 0, 255,
                              cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

        # Quantify the image and make predictions based on the extracted features
using the last trained Random Forest
        features = feature.hog(image, orientations=9,
                               pixels_per_cell=(10, 10), cells_per_block=(2, 2),
                               transform_sqrt=True, block_norm="L1")
        preds = model.predict([features])
        print(preds)
        ls = ["healthy", "parkinson"]
        result = ls[preds[0]]
        '''color = (0, 255, 0) if result == "healthy" else (0, 0, 255)
        cv2.putText(output, result, (3, 20),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
        cv2.imshow("Output", output)
        cv2.waitKey(0)'''
        return result
    return None


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