Sprint-3

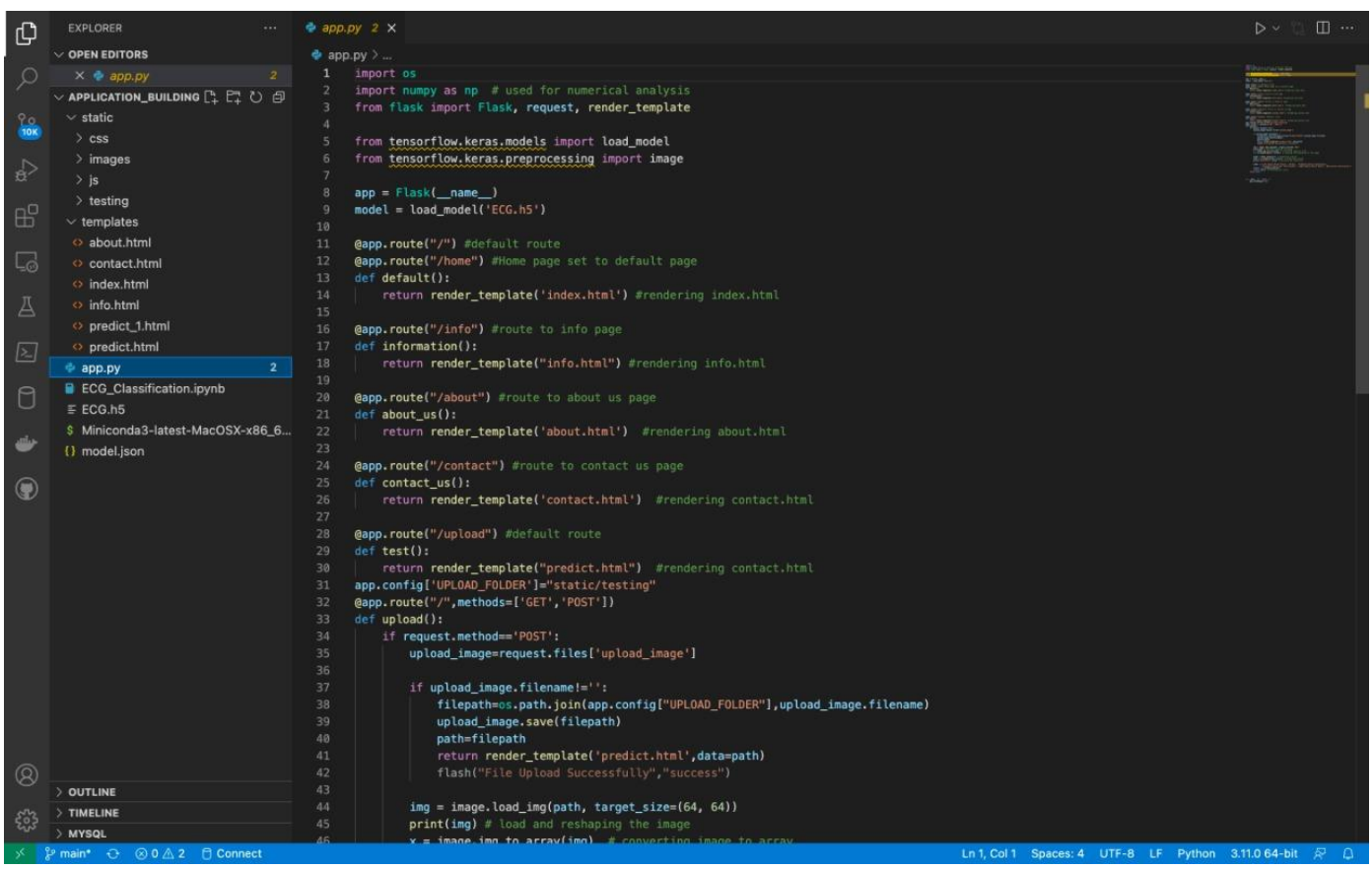
Application Building

BUILD THE PYTHON CODE

Date	11Nov 2022
TeamID	PNT2022TMID15718
ProjectName	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

TASK:

Build the python code.



APP.PY:

```
import os
import numpy as np  # used for numerical analysis
```

```
from flask import Flask, request, render_template
# Flask-It is our framework which we are going to use to run/serve our application.
# request-for accessing file which was uploaded by the user on our application.
# render_template- used for rendering the html pages
from tensorflow.keras.models import load_model  # to load our trained model
from tensorflow.keras.preprocessing import image

app = Flask(__name__)  # our flask app
model = load_model('ECG.h5')  # loading the model

@app.route("/") #default route
@app.route("/home") #Home page set to default page
def default():
    return render_template('index.html') #rendering index.html

@app.route("/info") #route to info page
def information():
    return render_template("info.html") #rendering info.html

@app.route("/about") #route to about us page
def about_us():
    return render_template('about.html')  #rendering about.html

@app.route("/contact") #route to contact us page
```

```
def contact_us():
    return render_template('contact.html')  #rendering contact.html

@app.route("/upload") #default route
def test():
    return render_template("predict.html")  #rendering contact.html

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

        img = image.load_img(filepath, target_size=(64, 64))  # load and
reshaping the image
        x = image.img_to_array(img)  # converting image to array
        x = np.expand_dims(x, axis=0)  # changing the dimensions of the
image

        preds = model.predict(x)  # predicting classes
        pred = np.argmax(preds, axis=1)  # predicting classes
        print("prediction", pred)  # printing the prediction
```

```
        index = ['Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction',
                 'Premature Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation']
        result = str(index[pred[0]])
        return result  # resturing the result
    return None


# port = int(os.getenv("PORT"))
    if __name__ == "__main__":
        app.run(debug=False)  # running our app
# app.run(host='0.0.0.0', port=8000)
```