

# Sprint-3

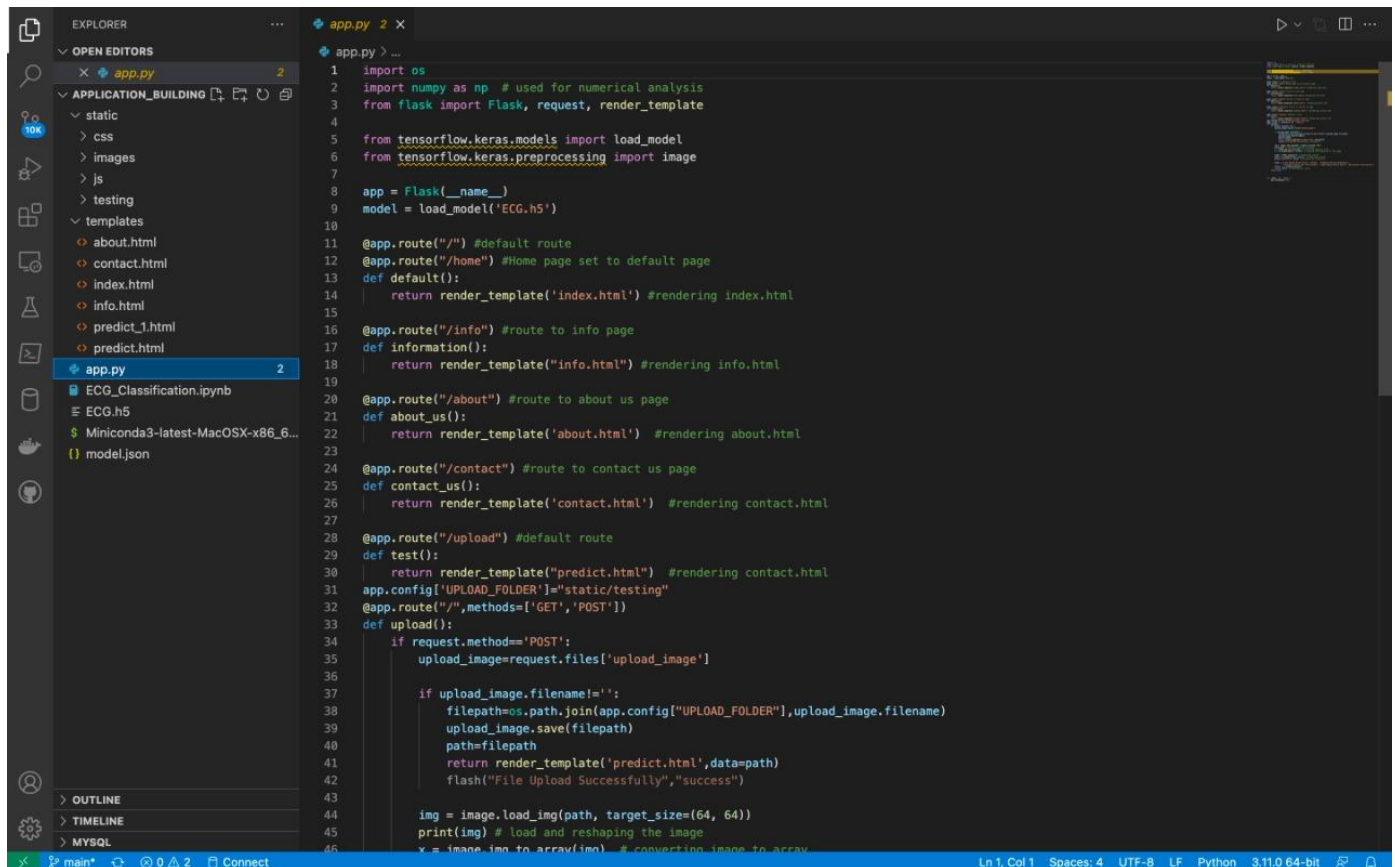
## Application Building

### BUILD THE PYTHON CODE

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

#### TASK:

*Build the python code.*



```
1 import os
2 import numpy as np # used for numerical analysis
3 from flask import Flask, request, render_template
4
5 from tensorflow.keras.models import load_model
6 from tensorflow.keras.preprocessing import image
7
8 app = Flask(__name__)
9 model = load_model('ECG.h5')
10
11 @app.route("/") #default route
12 @app.route("/home") #Home page set to default page
13 def default():
14     return render_template('index.html') #rendering index.html
15
16 @app.route("/info") #route to info page
17 def information():
18     return render_template("info.html") #rendering info.html
19
20 @app.route("/about") #route to about us page
21 def about_us():
22     return render_template('about.html') #rendering about.html
23
24 @app.route("/contact") #route to contact us page
25 def contact_us():
26     return render_template('contact.html') #rendering contact.html
27
28 @app.route("/upload") #default route
29 def test():
30     return render_template("predict.html") #rendering contact.html
31 app.config['UPLOAD_FOLDER'] = "static/testing"
32 @app.route("/", methods=['GET', 'POST'])
33 def upload():
34     if request.method == 'POST':
35         upload_image = request.files['upload_image']
36
37         if upload_image.filename != '':
38             filepath = os.path.join(app.config['UPLOAD_FOLDER'], upload_image.filename)
39             upload_image.save(filepath)
40             path = filepath
41             return render_template('predict.html', data=path)
42             flash("File Upload Successfully", "success")
43
44 img = image.load_img(path, target_size=(64, 64))
45 print(img) # load and reshaping the image
46 x = image.img_to_array(img) # converting image to array
```

#### 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 # restoring 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)
```