Application Building:

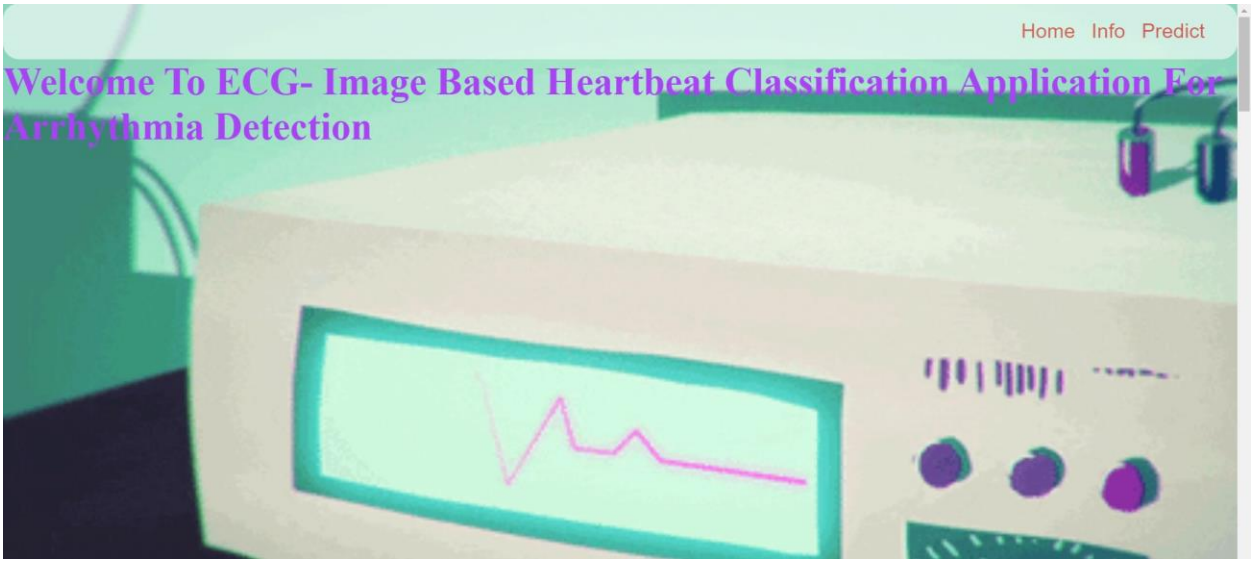
In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI. This section has the following tasks

- **Building HTML Pages:**

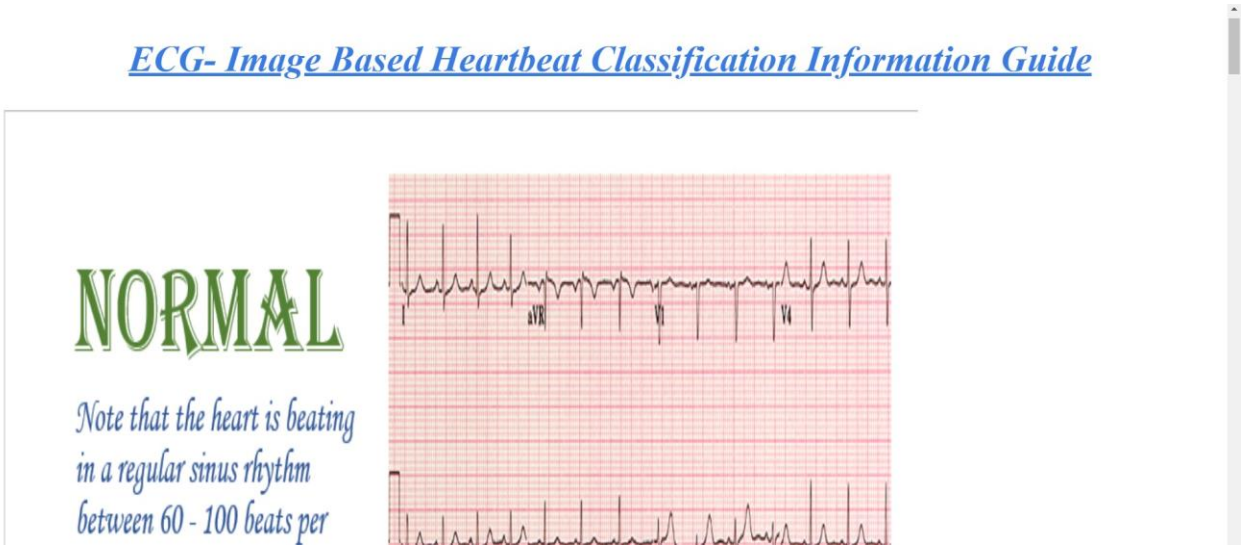
■ We use HTML to create the front end part of the web page.

■ Here, we created 4 html pages- home.html, predict_base.html, predict.html, information.html.

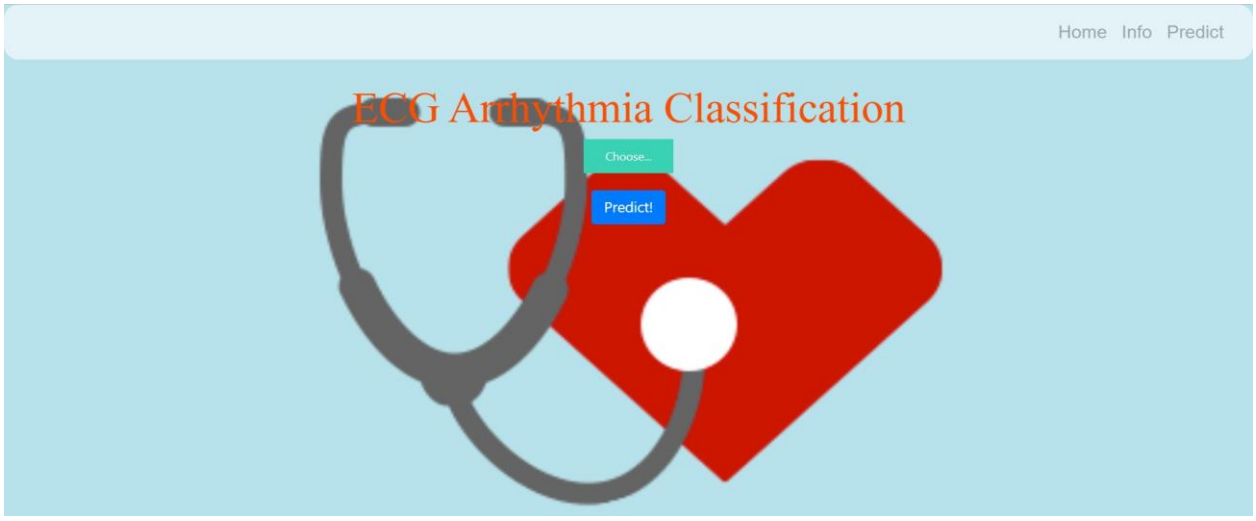
■ home.html displays the home page.

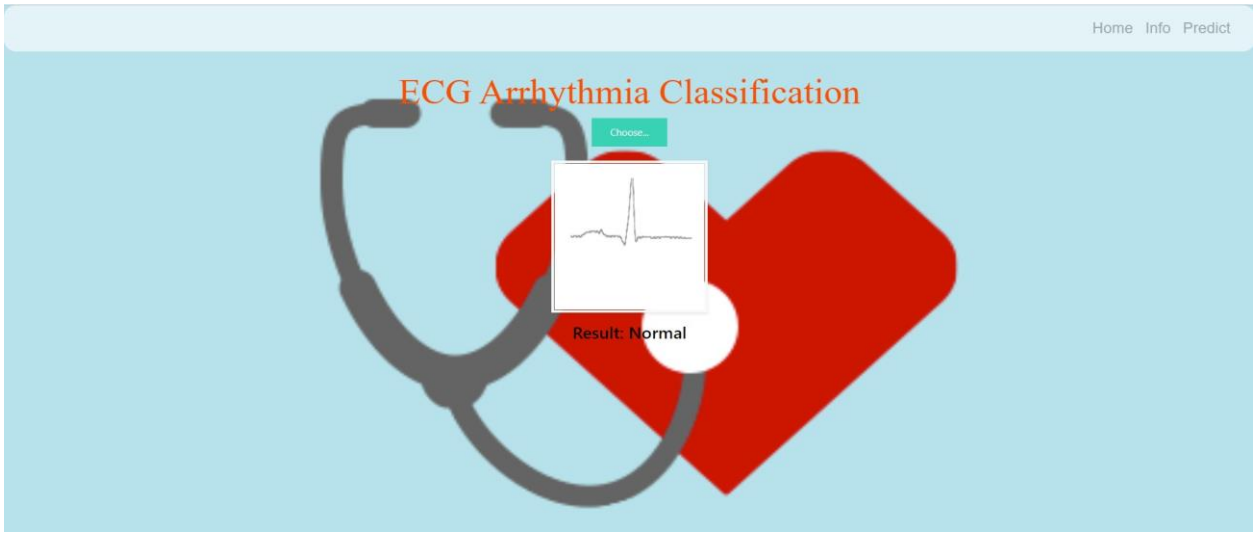


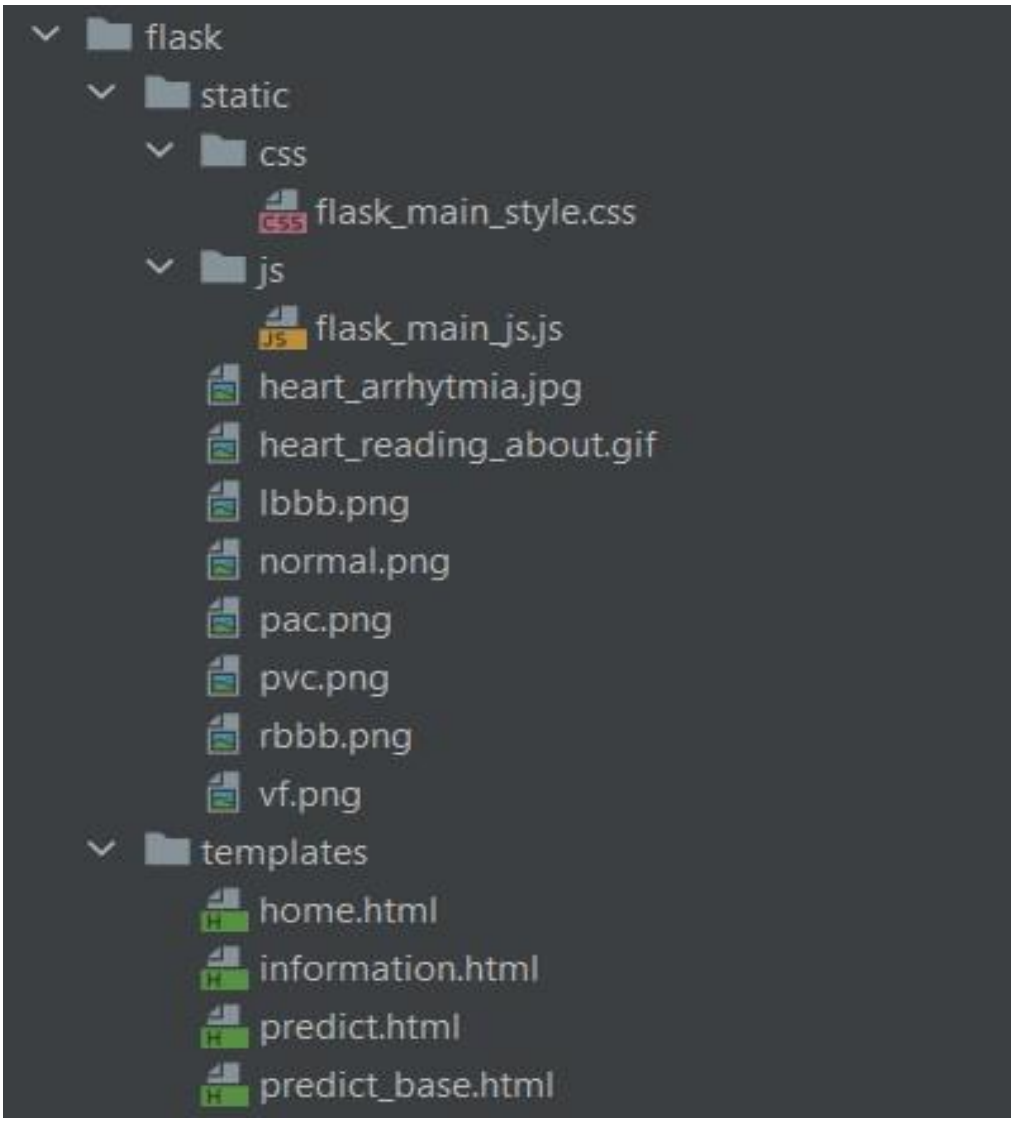
■ information.html displays all important details to be known about ECG.



- predict-base.html and predict.html accept input from the user and predicts the values.







- **Building server-side script:**

We will build the flask file 'app.py' which is a web framework written in python for server-side scripting.

- The app starts running when the "__name__" constructor is called in main.
- render_template is used to return HTML file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user.

```python
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
def about():
    return render_template("home.html")#rendering html page

@app.route("/about") #default route
def home():
    return render_template("home.html")#rendering html page

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

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

```python
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

        pred=model.predict(x)#predicting classes
        y_pred = np.argmax(pred)
        print("prediction",y_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[y_pred])

        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)
```

- **Running The App:**

```
C:\Users\M Sheshikiran Reddy\VIT\20BAI1061\CNN_PROJECT_SMARTINTERNZ\flask>python app_flask.py
```

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app_flask' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.