Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Team Id : PNT2022TMID09656

Team Members

1. RAHUL C
2. VIGNESH P
3. SURAVARAPU SAI CHARAN REDDY
4. MYTHIRESH G

CONTENTS

LITERATURE SURVEY

Existing problem
References
Problem Statement
Definition

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas
Ideation & Brainstorming
Proposed Solution
Problem Solution fit

REQUIREMENT ANALYSIS

Functional requirement
Non-Functional requirements

PROJECT DESIGN

Data Flow Diagrams
Solution & Technical Architecture
User Stories

PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation Sprint Delivery Schedule
Reports from JIRA

CODING & SOLUTIONING (Explain the features added in the project along with code)

Feature 1
Feature 2
Database Schema (if Applicable)

TESTING

Test Cases
User Acceptance Testing

RESULTS

Performance Metrics

ADVANTAGES& DISADVANTAGES CONCLUSION FUTURE SCOPE 13. APPENDIX

Source Code

GitHub & Project Demo Link

Hardware Requirements:

Processor : Intel Core i5HDD: 1TB

RAM: Minimum 2GB;

Recommended 4GB

Software Requirements:

Operating system : Windows 10 Dataset: IAM

Dataset(Words, Lines)Programming Language:

Python

Numpy : Core package providing powerful tools to manipulate data arrays, such as our character images.

OpenCV : OpenCV is a large open-source library for image processing, character recognition, and machine learning. Itcanscan handwritten images.

Autocorrect : It is used to correct the spelling. It supports manylanguages.

Tensorflow : Tensorflow is the core open source library to helpyou develop and train Machine Learning models.

Survey Papers:

Paper 1:

Author Name : Zhao Y, Cheng, J
Title : ECG classification using deep CNN improved bywavelet transform
Publication website : https://opus.lib.uts.edu.au/handle/10453/142924
Published Date : 2020-06-30
Objective: Atrial fibrillation is the most common persistentform of arrhythmia. A method based on wavelet transformcombined with deep

convolutional neural network is applied for automatic classification of electrocardiograms. Since the ECG signal is easily inferred, the ECG signal is decomposed into 9 kinds of subsignals with different frequency scales by wavelet function, and then wavelet reconstruction is carried out after segmented filtering to eliminate the influence of noise.

Technology used: Tensor flow - Tensor flow is the core open source library to help you develop and train MachineLearning models.

Paper 2:

1. ECG classification using deep CNN improved bywavelet transform
2. **Publication Year**: 2022-06-30
3. **Author**: . Zhao Y, Cheng, J . It is challenging to visually detect heart disease from the electrocardiographic (ECG) signals. Implementing an automated ECG signal detection system can help diagnosis arrhythmia in order to improve the accuracy of diagnosis. In this paper, we proposed, implemented,and compared an automated system using two different frameworks of the combination of convolutional neural network (CNN) and long-short term memory (LSTM) for classifying normal sinus signals, atrial fibrillation, and other noisy signals. The dataset we used is from the MIT-BIT Arrhythmia Physionet. Our approach demonstrated that the cascade of two deep learning network has higher performance than the concatenation of them, achieving a weighted f1 score of 0.82. The experimental results have successfully validated that the cascade of CNN and LSTM can achieve satisfactory performance on discriminating ECG signals Image. However, even after applying all the said techniques Might not possible to achieve the full accuracy in a Preprocessing system.

PROBLEM STATEMENT

To identify the erythema by the user in the device and to convert images into prediction.

What does this problem focus on?	The generative models can perform recognition driven segmentation. The method involves a relatively small number of parameter and hence training is relatively easy and fast.

When does this occur?	This matter occurs classified the irregular heartbeat from the given input data personal individual .
Why do we need this?	To recognize the irregular heartbeat that can be used to classify the images.
How to do this?	Unlike many other recognition schemes, it does not rely on some form of prenormalization of input images, but can handle arbitrary scalings, translations and a limited degree of image rotation.
Where it is used?	It is used in personal tracker and monitor ECG.

Ideation Phase

Brainstorm & Idea Prioritization Template

Date	19 September 2022
Team ID	PNT2022TMID09656
Project Name	Classification of arrhythmia by using deep learning with 2-d ecg spectral image representation
Maximum Marks	4 Marks

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
A Novel Method for Handwritten Digit Recognition

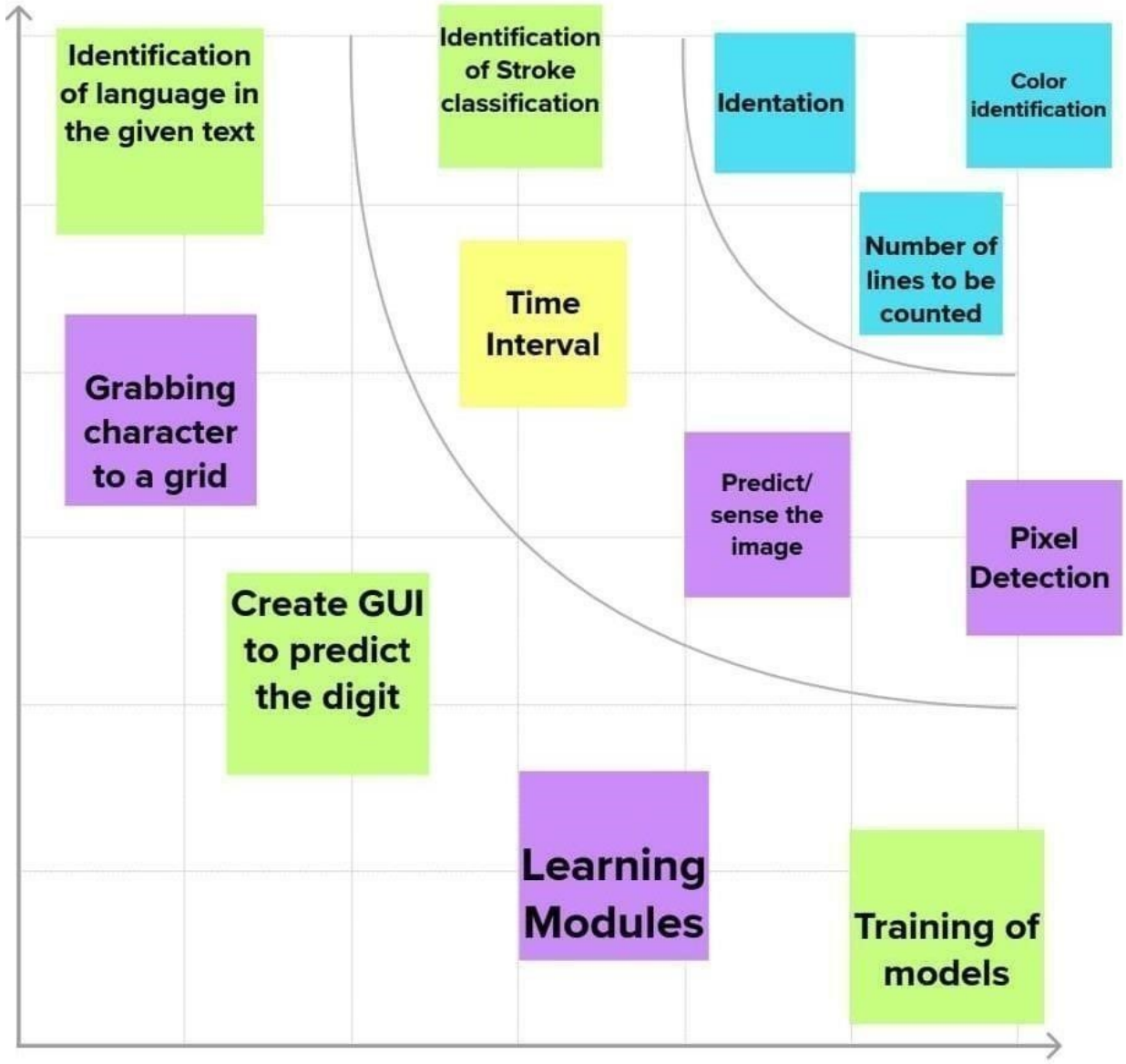
Handwriting recognition is a challenging task because every person in this world has their own style of writing. It is the capability of the computer to automatically identify and understand the handwritten digits. Due to the technological advancements, everything is being digitalized to reduce human effort. Hence, handwritten digit recognition is a need-of-the-hour task in many real-time applications. MNIST data set, which has 70000 handwritten digit samples, is widely used for this recognition process.

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

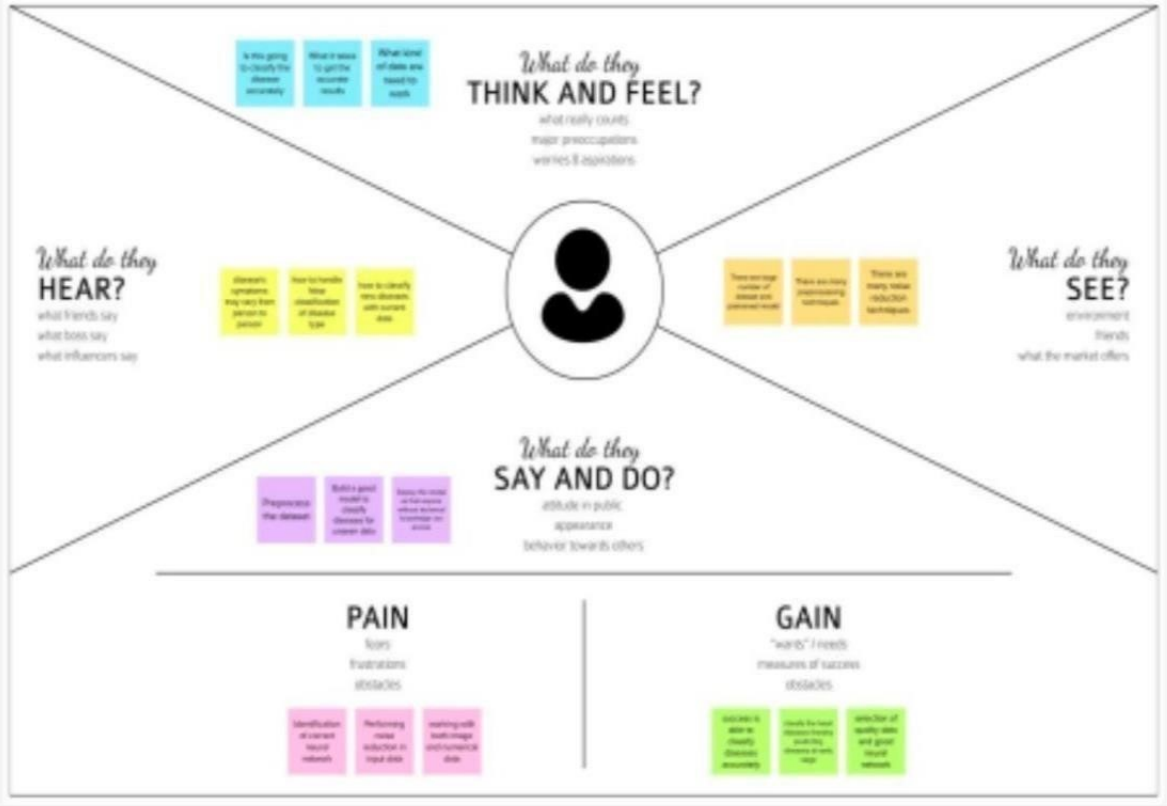
Step-3: Idea Prioritization


Identification of language in the given text
Identification of Stroke classification
Identation
Color identification
Number of lines to be counted
Time Interval
Grabbing character to a grid
Predict/ sense the image
Pixel Detection
Create GUI to predict the digit
Learning Modules
Training of models

Step 4

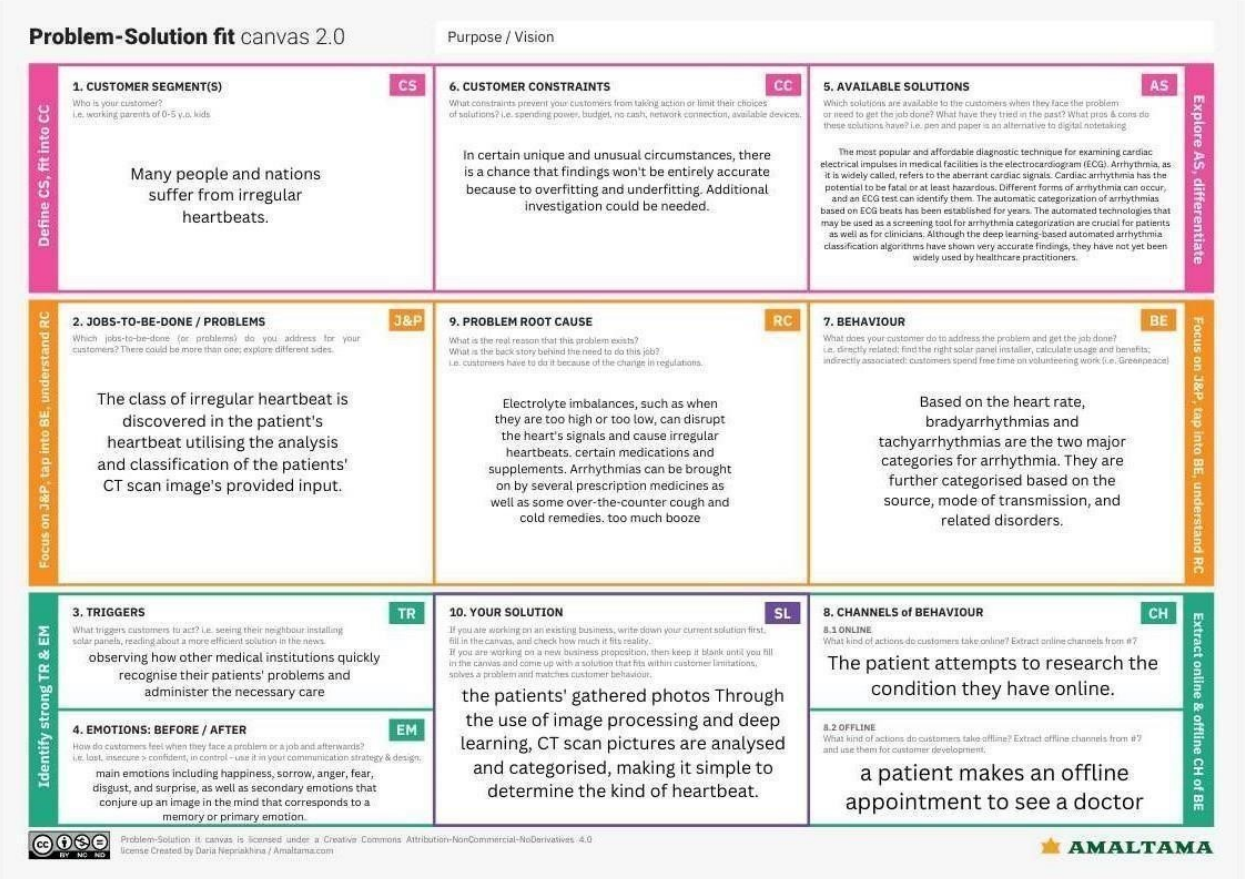
Customer journey map

EMPATHY MAP



Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Project Design Phase-I Problem – Solution Fit **Template**



Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	14 October 2022
Team ID	PNT2022TMID09656
Project Name	Classification of arrhythmia by using deep learning with 2-d ECG spectral image representation

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR- 3	User interface	Check your profile Choose your file Sign Out your account account and change your password
FR- 4	Data processing	Evaluating the model using test data Training DL algorithm for a accuracy result Trained CNN model using Tensorflow,Kearas
FR-5	Predict ECG image	User ECG images in our web application Collection of datasets Database read ECG images

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functiona l Requirement	Description
NFR-1	Usability	Wireless ECG body sensor Savvy is a feasible solution for reliable and accurate long-term heart rhythm monitoring**.** However, there were no studies dealing with usability of this sensor in field testing.
NFR-2	Security	The work presented in this paper is applicable for encrypting and decrypting personalized Electrocardiograph ECG signals for secure transmission.
NFR-3	Reliability	The extent to the consistently performs the specified functions without failure
NFR-4	Performance	It essentially specifies how the system should behave and that it constrains the ECG wavelength of accurate disease information gathering.
NFR-5	Availability	Availability describes how likely the system is accessible to a user at a given point in time and the periodically for a solutions.
NFR-6	Scalability	The ability of the user problem in arrhythmia disease to handle an increase in workload without performance degradation, or its ability to quickly enlarge.

Project Design Phase-II

(Data Flow Diagram & User Stories)

Date	17 October 2022
Team ID	PNT2022TMID09656
Project Name	A Novel Method for Handwritten Digit Recognition System
Maximum Marks	4 Marks

Data Flow Diagrams:

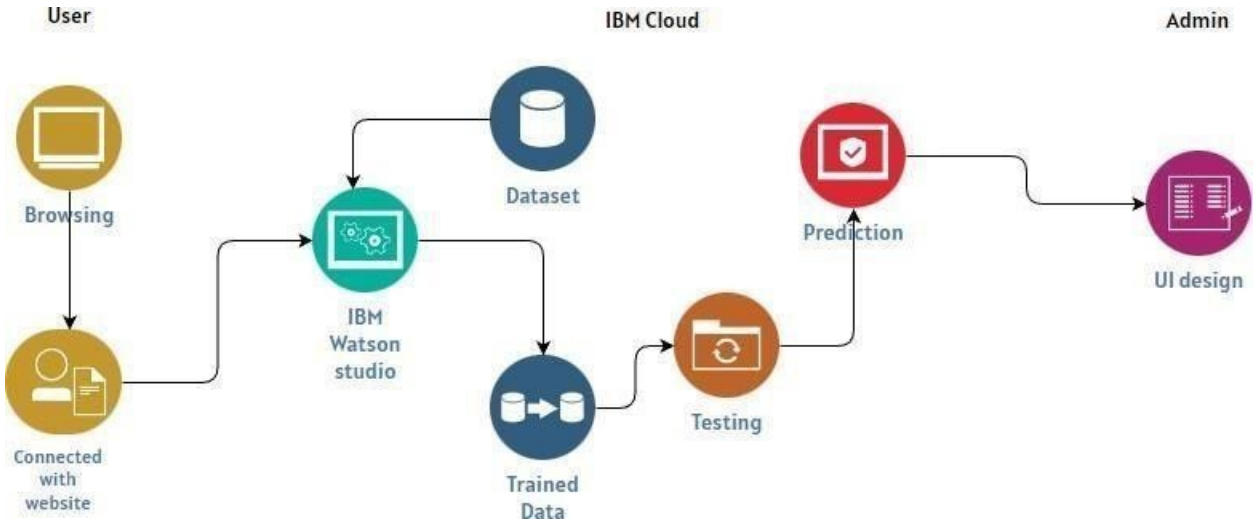
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

User Type	User Story Number	User Story / Task	Release
Customer	USN-1	As a user, I can application by opening it easily.	Sprint-1

	USN-2	As a user, I can upload images	Sprint-1
	USN-3	As a user, I can change the colour of the pen ink.	Sprint-2

Project Design Phase-II

Technology Stack (Architecture & Stack)



Date	14 October 2022
Team ID	PNT2022TMID09656
Project Name	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2 **Table-1 : Components &Technologies:**

S. No	Component	Description	Technology
1.	User Interface	Web UI, Mobile UI.	HTML, CSS, JavaScript / React

			Js.
2.	Application Logic-1	Python is used for backend	Python
3.	Application Logic-2	It's a symbolic math toolkit that performs a variety of tasks including deep neural network training and inference using dataflow and differentiable programming	Tensorflow
4.	Cloud Database	A global technology company that provides hardware, software, cloud-based services and cognitive computing.	IBM Cloud
5.	File Storage	Breaks up data into blocks and then stores those blocks as separate pieces, each with a unique identifier.	IBM Block
6.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
7.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
8.	Machine Learning Model	Object recognition is a subfield of computer vision, artificial intelligence, and machine learning	Object Recognition Model

9.	Deep learning Model	The images from the created dataset are fed into a neural network algorithm.	Image Recognition Model

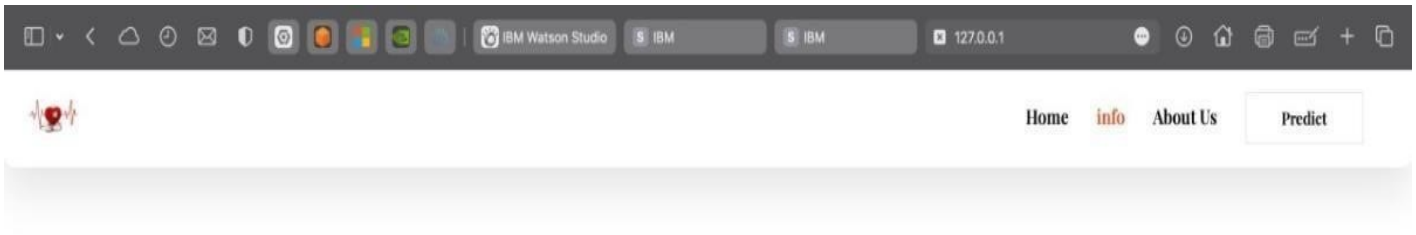
Table-2: Application Characteristics:

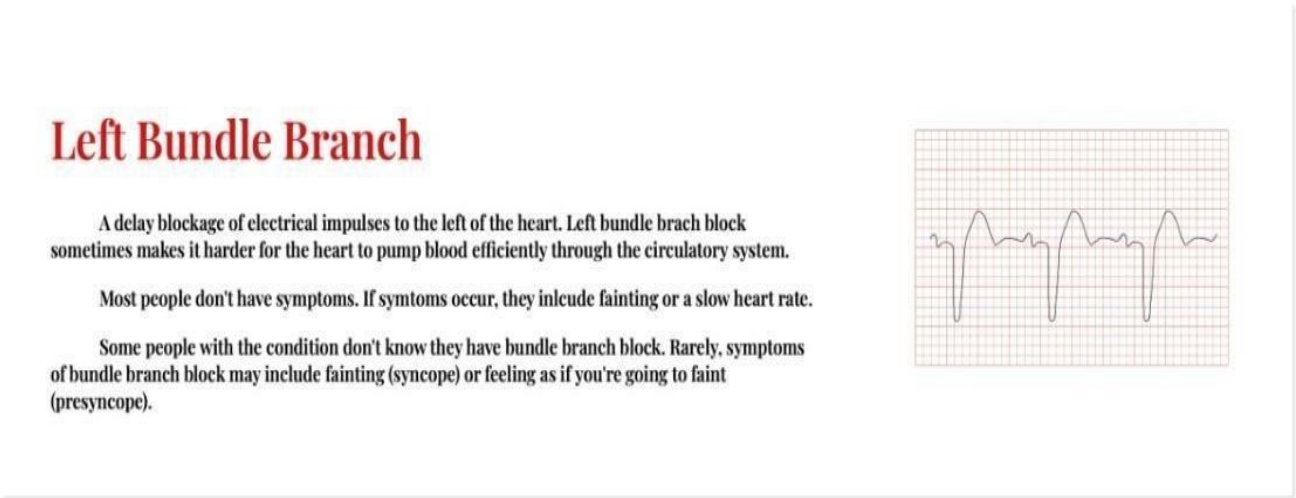
S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Building user interfaces based on UI components.	React Js
2.	Security Implementations	OWASP is a nonprofit foundation that works to improve the security of software.	OWASP
3.	Scalable Architecture	a modular client-server architecture that consists of a presentation tier, an application tier and a data tier	3-tier architecture
4.	Availability	The data on each server can be simultaneously accessed and modified via a network.	Distributed Server

5.	Performance	Increasing data retrieval performance by reducing the need to access the underlying slower storage layer.	Cache

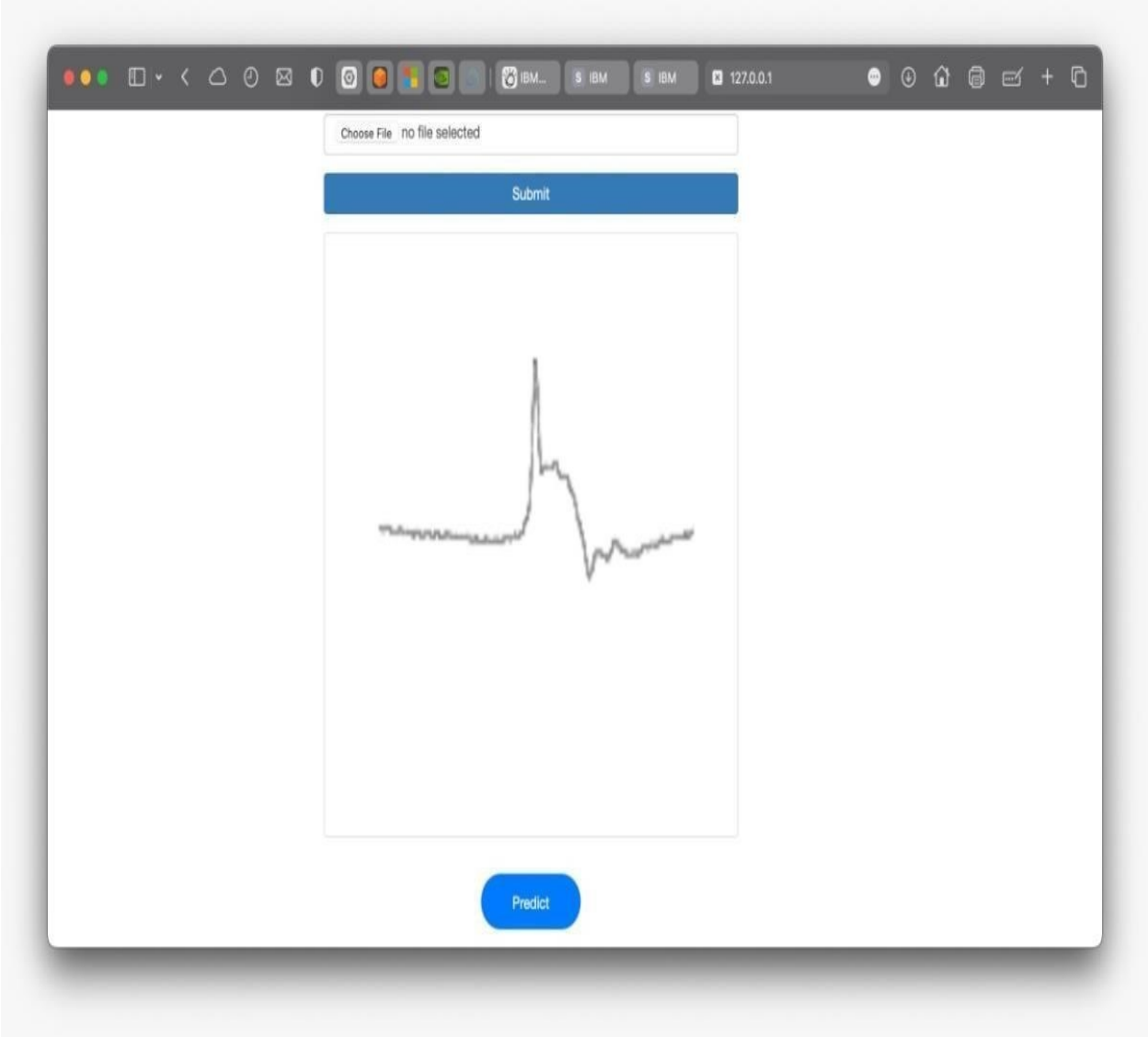
Webpages

Info.html

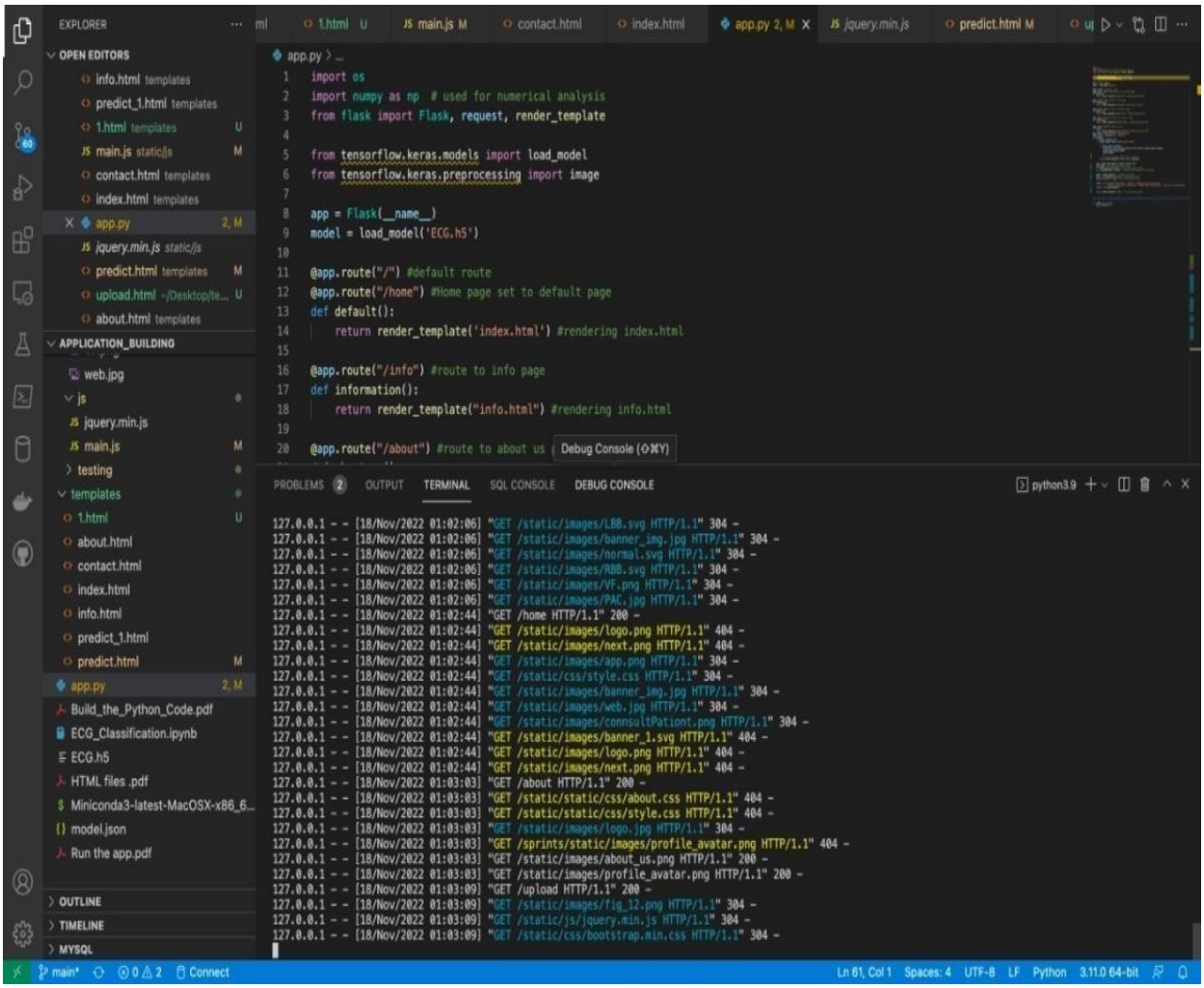




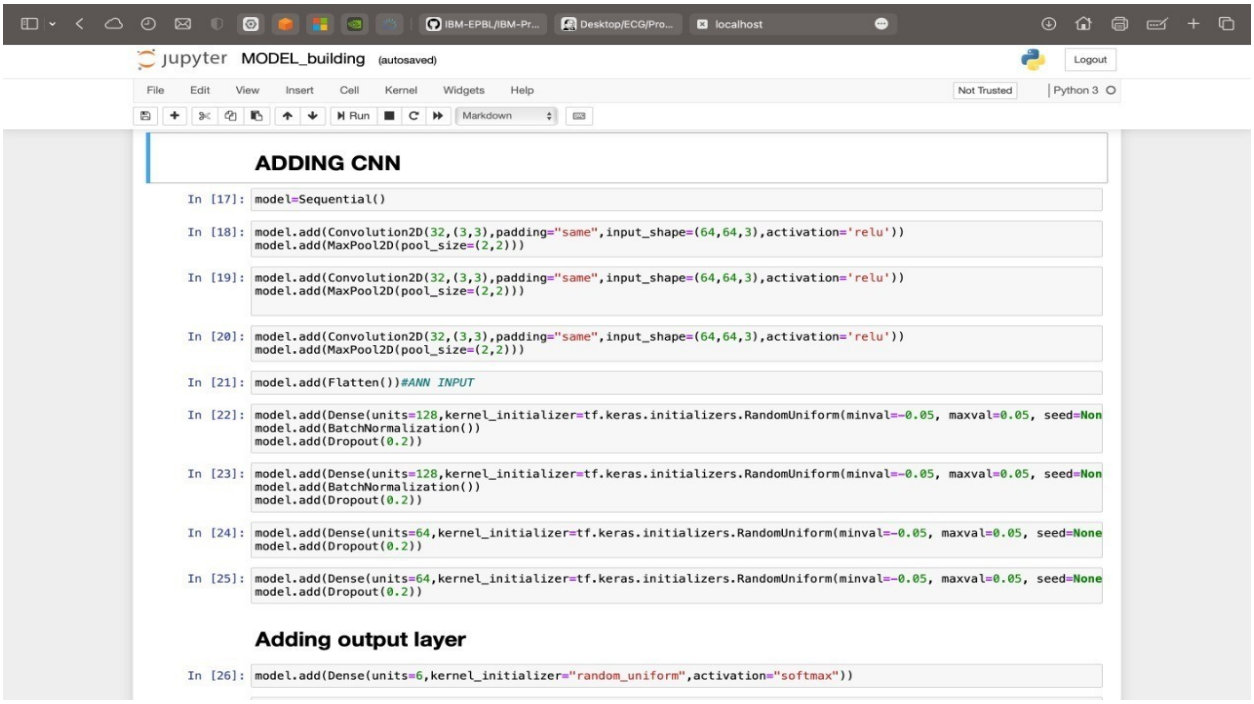
Predict.html



Run the application



MODEL BUILDING



jupyter MODEL_building (autosaved)

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

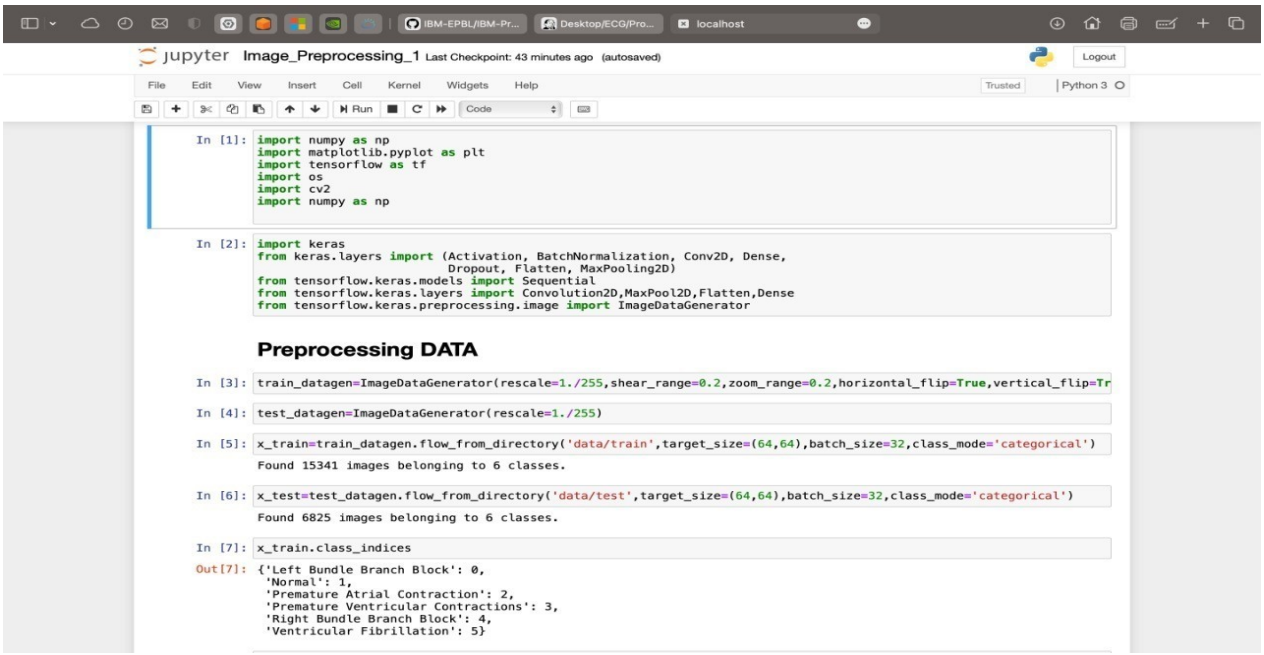
Run — Markdown

ADDING CNN

```
In [17]: model=Sequential()

In [18]: model.add(Convolution2D(32,(3,3),padding="same",input_shape=(64,64,3),activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2)))

In [19]: model.add(Convolution2D(32,(3,3),padding="same",input_shape=(64,64,3),activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2)))

In [20]: model.add(Convolution2D(32,(3,3),padding="same",input_shape=(64,64,3),activation='relu'))
         model.add(MaxPool2D(pool_size=(2,2)))

In [21]: model.add(Flatten())#ANN INPUT

In [22]: model.add(Dense(units=128,kernel_initializer=tf.keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=Non
         model.add(BatchNormalization())
         model.add(Dropout(0.2))

In [23]: model.add(Dense(units=128,kernel_initializer=tf.keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=Non
         model.add(BatchNormalization())
         model.add(Dropout(0.2))

In [24]: model.add(Dense(units=64,kernel_initializer=tf.keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None
         model.add(Dropout(0.2))

In [25]: model.add(Dense(units=64,kernel_initializer=tf.keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None
         model.add(Dropout(0.2))
```

Adding output layer

```
In [26]: model.add(Dense(units=6,kernel_initializer="random_uniform",activation="softmax"))
```



jupyter Image_Preprocessing_1 Last Checkpoint: 43 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

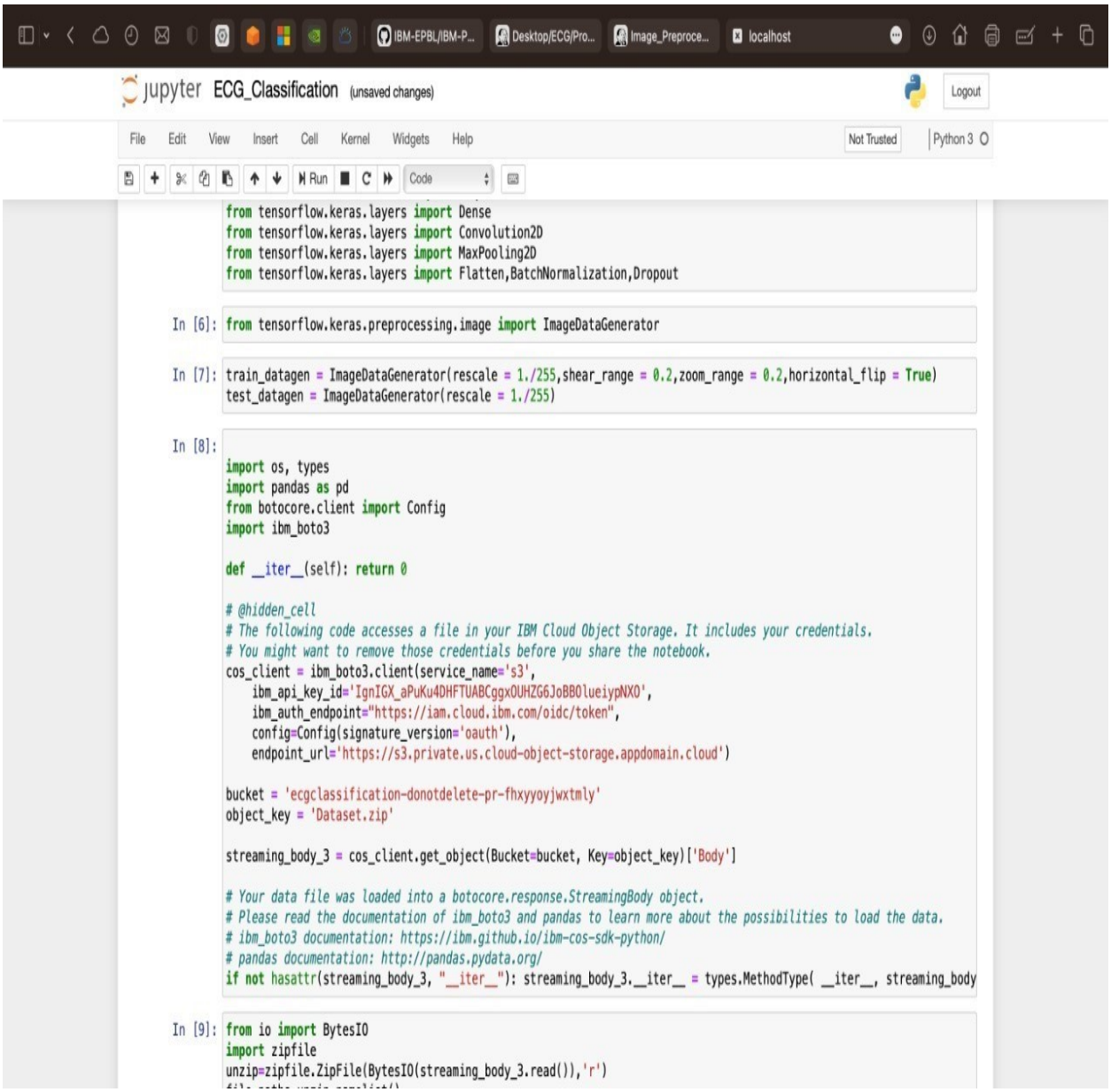
Run — Code

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import tensorflow as tf
        import os
        import cv2
        import numpy as np

In [2]: import keras
        from keras.layers import (Activation, BatchNormalization, Conv2D, Dense,
                                  Dropout, Flatten, MaxPooling2D)
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Convolution2D,MaxPool2D,Flatten,Dense
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

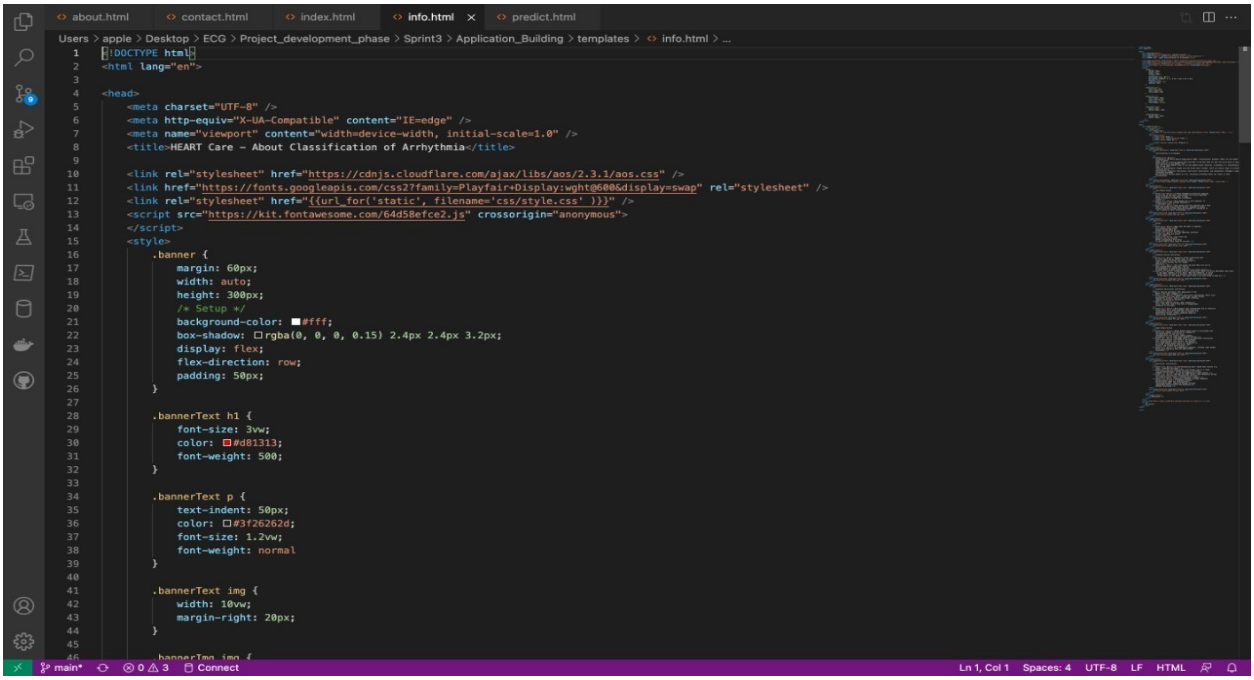
Preprocessing DATA

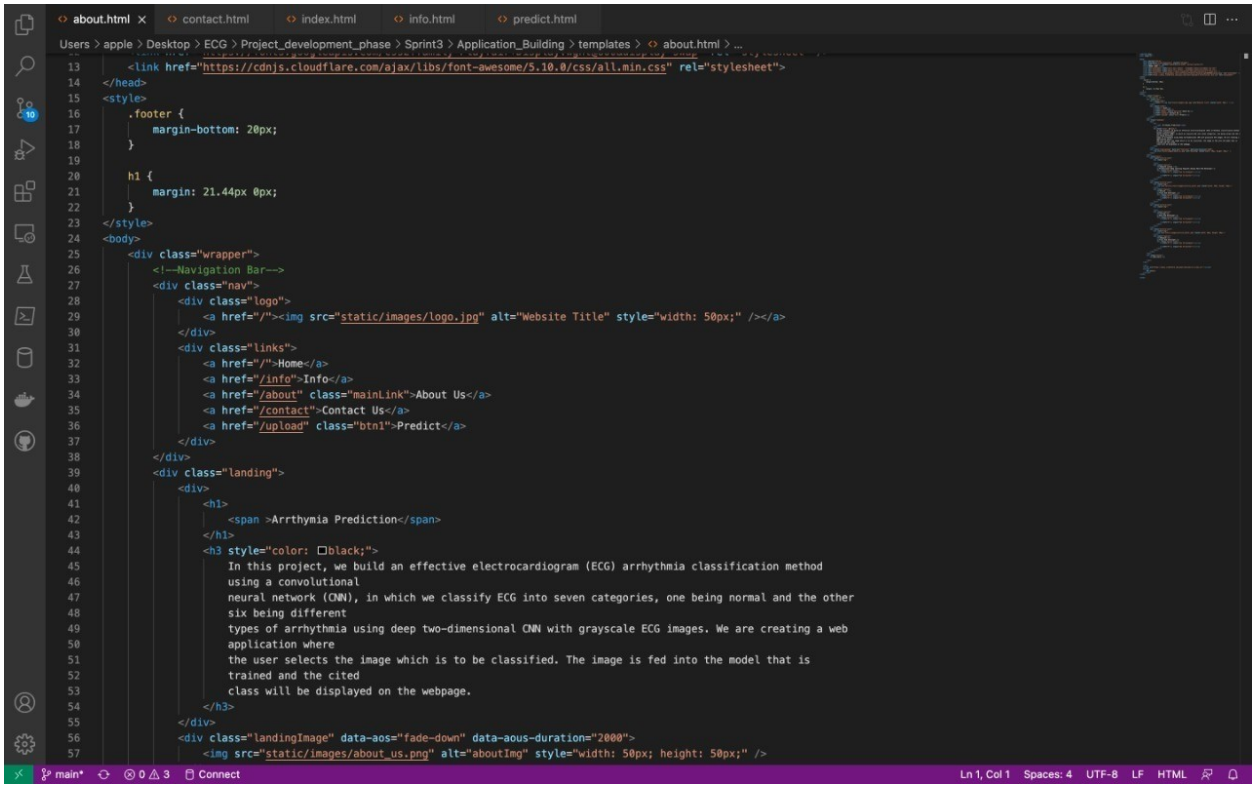
```
In [3]: train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=Tr

In [4]: test_datagen=ImageDataGenerator(rescale=1./255)

In [5]: x_train=train_datagen.flow_from_directory('data/train',target_size=(64,64),batch_size=32,class_mode='categorical')
        Found 15341 images belonging to 6 classes.

In [6]: x_test=test_datagen.flow_from_directory('data/test',target_size=(64,64),batch_size=32,class_mode='categorical')
        Found 6825 images belonging to 6 classes.

In [7]: x_train.class_indices
Out[7]: {'Left Bundle Branch Block': 0,
         'Normal': 1,
         'Premature Atrial Contraction': 2,
         'Premature Ventricular Contractions': 3,
         'Right Bundle Branch Block': 4,
         'Ventricular Fibrillation': 5}
```

ECG Classification



HTML CODING


about.html
contact.html
index.html
info.html
predict.html
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > templates > info.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>HEART Care - About Classification of Arrhythmia</title>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/aos/2.3.1/aos.css" />
<link href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@600&display=swap" rel="stylesheet" />
<link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}" />
<script src="https://kit.fontawesome.com/64d58efce2.js" crossorigin="anonymous">
</script>
<style>
.banner {
margin: 60px;
width: auto;
height: 300px;
/* Setup */
background-color: #fff;
box-shadow: rgba(0, 0, 0, 0.15) 2.4px 2.4px 3.2px;
display: flex;
flex-direction: row;
padding: 50px;
}
.bannerText h1 {
font-size: 3vw;
color: #d81313;
font-weight: 500;
}
.bannerText p {
text-indent: 50px;
color: #3f262626d;
font-size: 1.2vw;
font-weight: normal
}
.bannerText img {
width: 10vw;
margin-right: 20px;
}
main*
Connect
Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML


about.html
contact.html
index.html
info.html
predict.html
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > templates > about.html > ...
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
</head>
<style>
.footer {
margin-bottom: 20px;
}
h1 {
margin: 21.44px 0px;
}
</style>
<body>

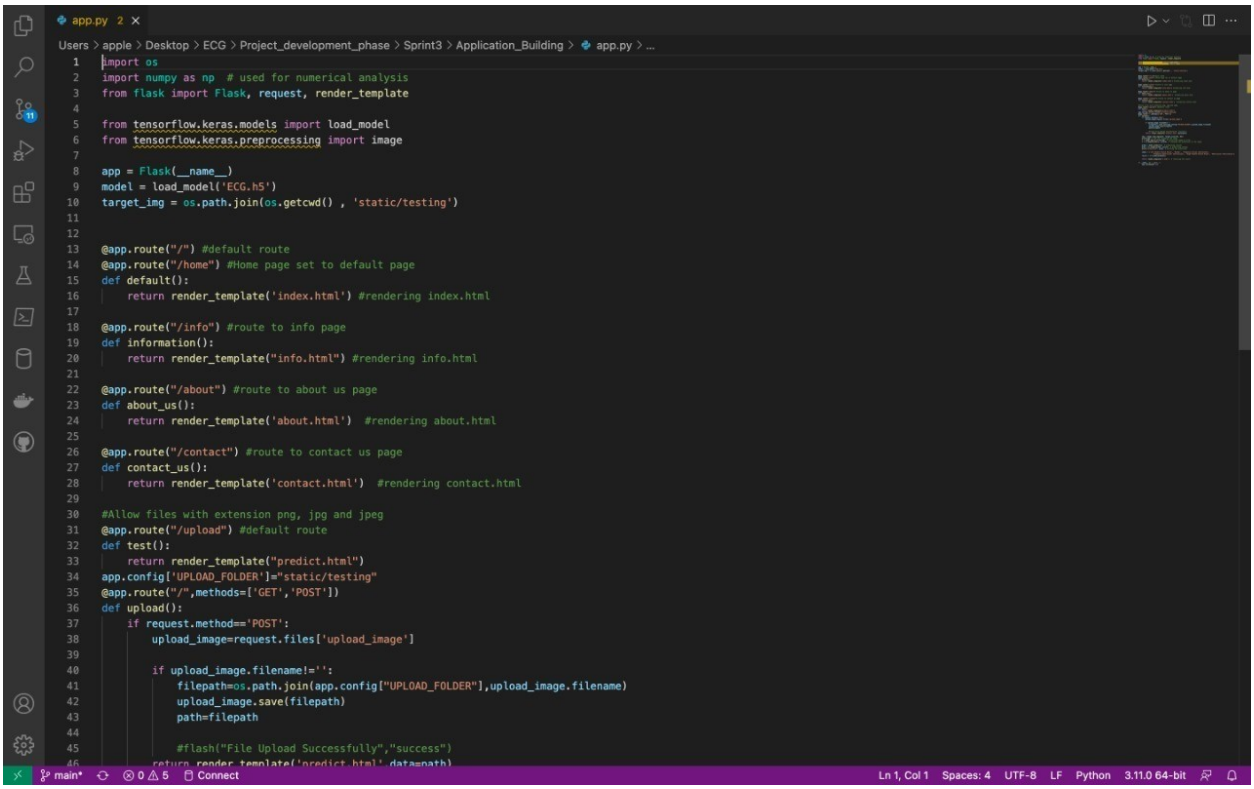
<!--Navigation Bar-->

Home
Info
About Us
Contact Us
Predict

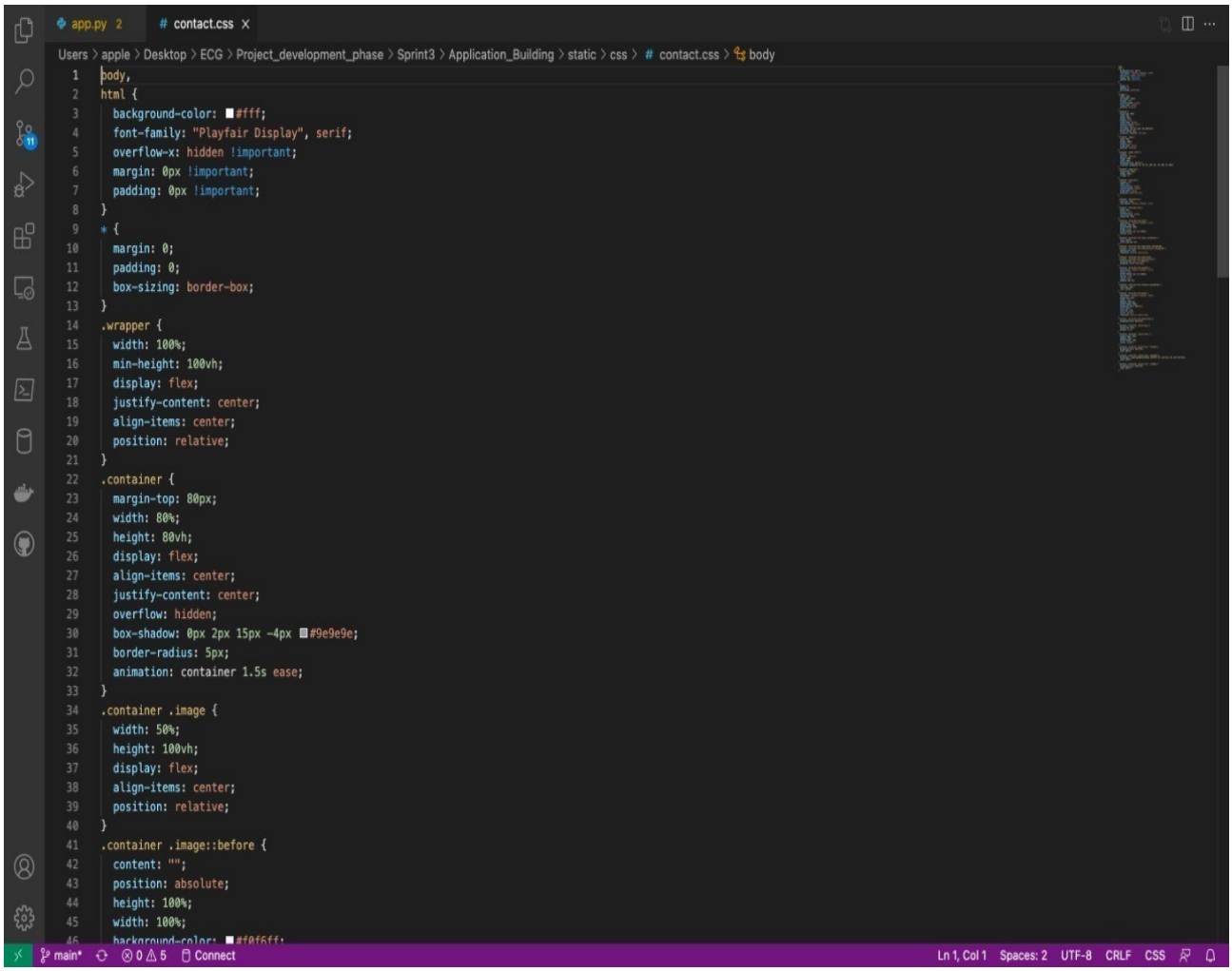
<h1>
Arrthymia Prediction
</h1>
<h3 style="color: black;">
In this project, we build an effective electrocardiogram (ECG) arrhythmia classification method
using a convolutional
neural network (CNN), in which we classify ECG into seven categories, one being normal and the other
six being different
types of arrhythmia using deep two-dimensional CNN with grayscale ECG images. We are creating a web
application where
the user selects the image which is to be classified. The image is fed into the model that is
trained and the cited
class will be displayed on the webpage.
</h3>

main*
Connect
Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML

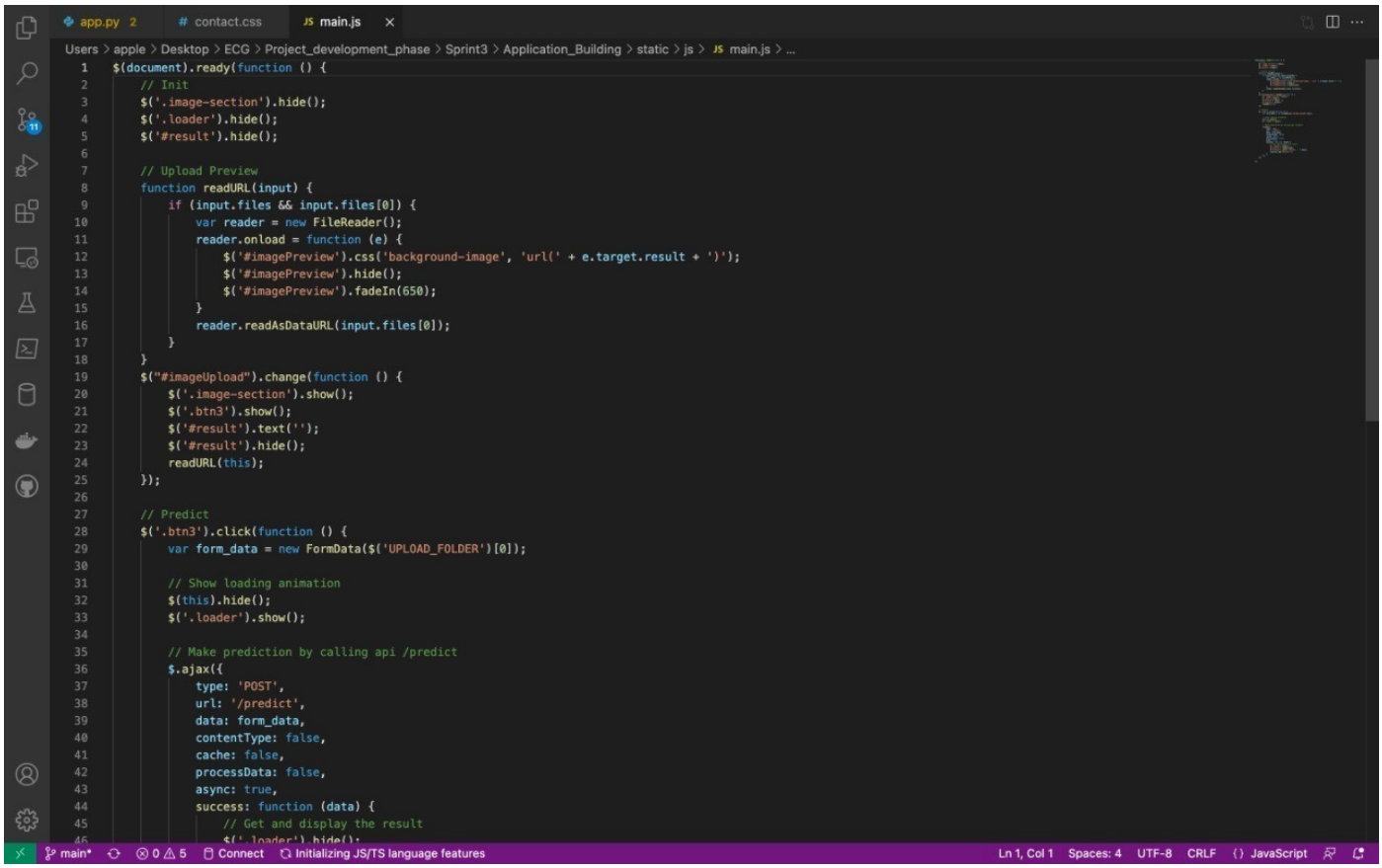
APP.PY

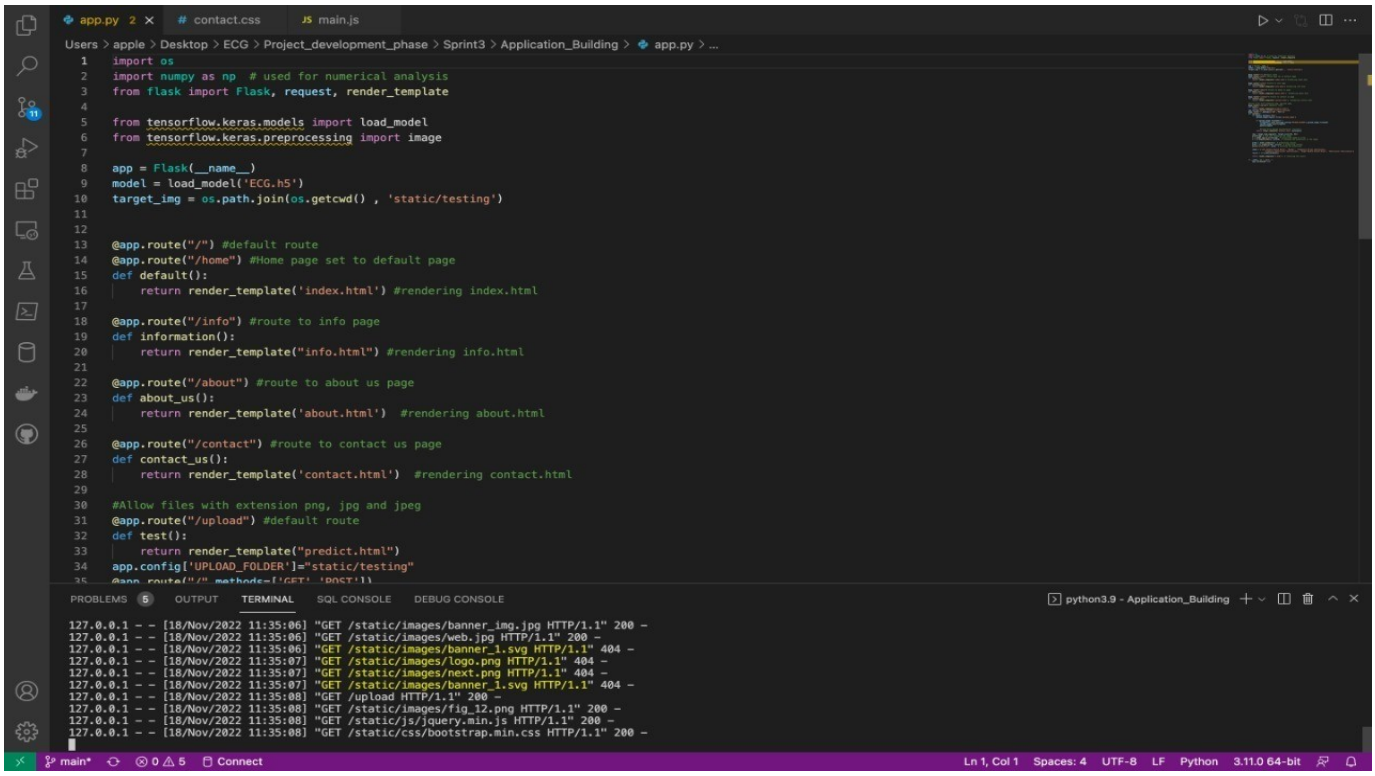

app.py 2
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > app.py > ...
import os
import numpy as np # used for numerical analysis
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
app = Flask(__name__)
model = load_model('ECG.h5')
target_img = os.path.join(os.getcwd() , 'static/testing')
@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
#Allow files with extension png, jpg and jpeg
@app.route("/upload") #default route
def test():
return render_template("predict.html")
app.config['UPLOAD_FOLDER']="static/testing"
@app.route("/",methods=['GET','POST'])
def upload():
if request.method=='POST':
upload_image=request.files['upload_image']
if upload_image.filename!='':
filepath=os.path.join(app.config["UPLOAD_FOLDER"],upload_image.filename)
upload_image.save(filepath)
path=filepath
#flash("File Upload Successfully","success")
return render_template('predict.html',data=path)
main*
0 5
Connect
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.0 64-bit

CSS


app.py 2
contact.css
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > static > css > contact.css > body
body,
html {
background-color: #fff;
font-family: "Playfair Display", serif;
overflow-x: hidden !important;
margin: 0px !important;
padding: 0px !important;
}
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
.wrapper {
width: 100%;
min-height: 100vh;
display: flex;
justify-content: center;
align-items: center;
position: relative;
}
.container {
margin-top: 80px;
width: 80%;
height: 80vh;
display: flex;
align-items: center;
justify-content: center;
overflow: hidden;
box-shadow: 0px 2px 15px -4px #9e9e9e;
border-radius: 5px;
animation: container 1.5s ease;
}
.container .image {
width: 50%;
height: 100vh;
display: flex;
align-items: center;
position: relative;
}
.container .image::before {
content: "";
position: absolute;
height: 100%;
width: 100%;
main*
0 5
Connect
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
CSS

JAVA SCRIPT


app.py 2
contact.css
main.js
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > static > js > 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();
$('.btn3').show();
$('#result').text('');
$('#result').hide();
readURL(this);
});
// Predict
$('.btn3').click(function () {
var form_data = new FormData($('UPLOAD_FOLDER')[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
main*
0 5
Connect
Initializing JS/TS language features
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript


app.py 2
contact.css
main.js
Users > apple > Desktop > ECG > Project_development_phase > Sprint3 > Application_Building > app.py > ...
import os
import numpy as np # used for numerical analysis
from flask import Flask, request, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
app = Flask(__name__)
model = load_model('ECG.h5')
target_img = os.path.join(os.getcwd() , 'static/testing')
@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
#Allow files with extension png, jpg and jpeg
@app.route("/upload") #default route
def test():
return render_template("predict.html")
app.config['UPLOAD_FOLDER']="static/testing"
PROBLEMS 5
OUTPUT
TERMINAL
SQL CONSOLE
DEBUG CONSOLE
python3.9 - Application_Building
127.0.0.1 - - [18/Nov/2022 11:35:06] "GET /static/images/banner_img.jpg HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 11:35:06] "GET /static/images/web.jpg HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 11:35:06] "GET /static/images/banner_1.svg HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 11:35:07] "GET /static/images/logo.png HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 11:35:07] "GET /static/images/next.png HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 11:35:07] "GET /static/images/banner_1.svg HTTP/1.1" 404 -
127.0.0.1 - - [18/Nov/2022 11:35:08] "GET /upload HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 11:35:08] "GET /static/images/fig_12.png HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 11:35:08] "GET /static/js/jquery.min.js HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2022 11:35:08] "GET /static/css/bootstrap.min.css HTTP/1.1" 200 -
main*
0 5
Connect
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.0 64-bit