

# **1.INTRODUCTION**

## **1.1 PROJECT REVIEW**

The electrocardiogram (ECG) is one of the most extensively employed signals used in the diagnosis and prediction of cardiovascular diseases (CVDs). The ECG signals can capture the heart's rhythmic irregularities, commonly known as arrhythmias. A careful study of ECG signals is crucial for precise diagnoses of patients' acute and chronic heart conditions. In this study, we propose a two-dimensional (2-D) convolutional neural network (CNN) model for the classification of ECG signals into eight classes; namely, normal beat, premature ventricular contraction beat, paced beat, right bundle branch block beat, left bundle branch block beat, atrial premature contraction beat, ventricular flutter wave beat, and ventricular escape beat. The one-dimensional ECG time series signals are transformed into 2-D spectrograms through short-time Fourier transform. The 2-D CNN model consisting of four convolutional layers and four pooling layers is designed for extracting robust features from the input spectrograms.

## **1.2 Purpose**

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle-income countries. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. 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.

## 2. LITERATURE SURVEY

### 2.1 Existing problem

The electrocardiogram (ECG) is one of the most extensively employed signals used in the diagnosis and prediction of cardiovascular diseases (CVDs). The ECG signals can capture the heart's rhythmic irregularities, commonly known as arrhythmias. A careful study of ECG signals is crucial for precise diagnoses of patients' acute and chronic heart conditions. In this study, we propose a two-dimensional (2-D) convolutional neural network (CNN) model for the classification of ECG signals into eight classes; namely, normal beat, premature ventricular contraction beat, paced beat, right bundle branch block beat, left bundle branch block beat, atrial premature contraction beat, ventricular flutter wave beat, and ventricular escape beat. The one-dimensional ECG time series signals are transformed into 2-D spectrograms through short-time Fourier transform. The 2-D CNN model consisting of four convolutional layers and four pooling layers is designed for extracting robust features from the input spectrograms. We achieved a state-of-the-art average classification accuracy of 99.11%, which is better than those of recently reported results in classifying similar arrhythmias.

### 2.2 References

[1] **Po-Ya Hsu** Department of Computer Science & Engineering, University of California

San Diego, The USA

**Chung-Kuan Cheng** Department of Computer Science & Engineering, University of

California, San Diego, The USA

<https://ieeexplore.ieee.org/document/9176679>

[2] **A. Rajkumar** Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, India

**M. Ganesan** Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, India

**R. Lavanya** Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, India

<https://ieeexplore.ieee.org/abstract/document/8728362>

[3] **EunKwang Jeon** Dept. of Computer Science & Engineering, Soonchunhyang University, Asan, South Korea

**MinSu Chae** Dept. of Computer Science & Engineering, Soonchunhyang University, Asan, South Korea

**Sangwook Han** Dept. of Computer Science & Engineering, Soonchunhyang University, Asan, South Korea

<https://ieeexplore.ieee.org/document/8989066>

[4] **Amin Ullah** University of Engineering and Technology

**Taxila Syed Anwar** University of Engineering and Technology

**Taxila Muhammad Bilal Hankuk** University of Foreign Studies

**Raja Majid Mehmood** Xiamen University Malaysia

## 2.3 Problem Statement Definition

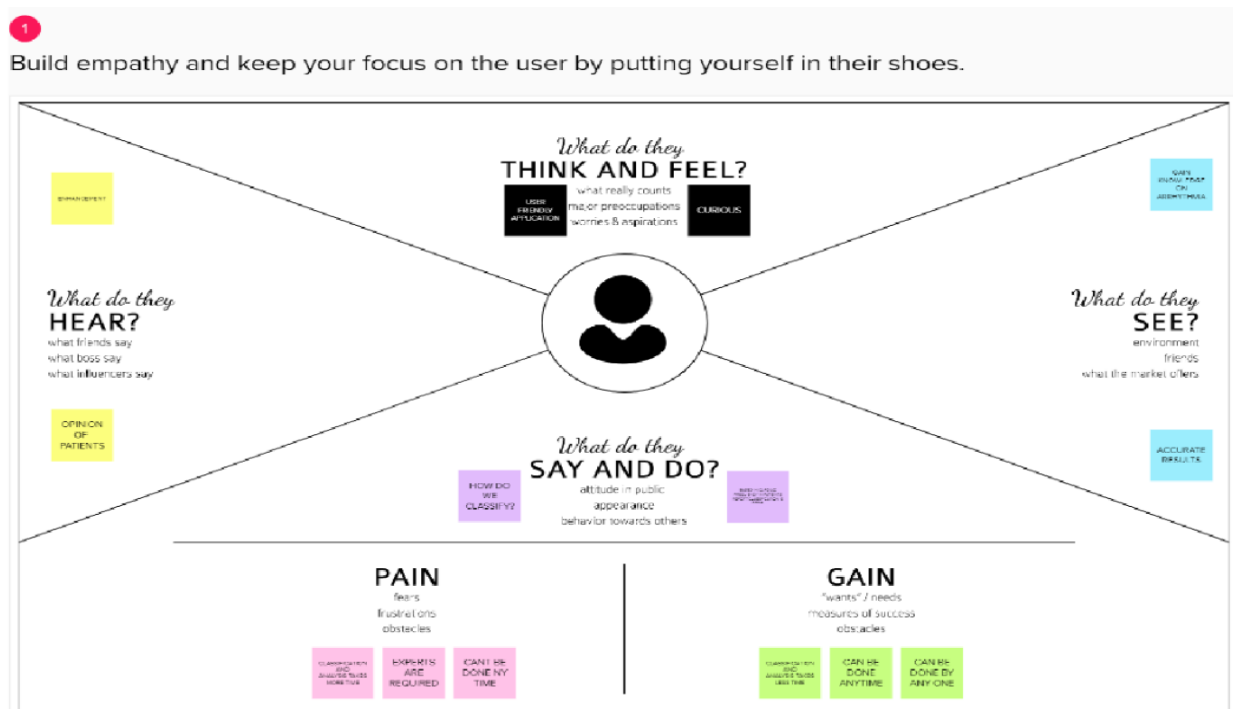
Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. 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 side 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.


### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



## 3.2 Ideation & Brainstorming



### 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

##### A Team gathering

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

##### B Set the goal

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

##### C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1

#### Define your problem statement

Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. 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.

#### Key rules of brainstorming

To run a smooth and productive session

- 🗨️ Stay in topic.
- 💡 Encourage wild ideas.
- ⏸️ Defer judgment.
- 👂 Listen to others.
- 🗣️ Go for volume.
- 👁️ If possible, be visual.

### Step-2: Brainstorm, Idea Listing and Grouping

2

#### Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

Diya			Dhruv			Nivedha			Preethi		
Identifying the problem	Brainstorming ideas	Get feedback from others	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas
Identifying the problem	Brainstorming ideas	Get feedback from others	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas
Identifying the problem	Brainstorming ideas	Get feedback from others	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas	Brainstorming ideas

3

#### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

#### Features



#### Applications



#### Results



### 3.3 Proposed Solution

Classification of arrhythmia using deep learning with 2d ECG spectral image representation. To create an application that is used to classify the arrhythmia and provide more detailed information about it. Here we use deep learning techniques and with the help of 2D ECG spectral image to classify the arrhythmia. Provides accurate results and detailed information required by the users or patients. Users or customers can easily use the app because of its user friendly interface and simplicity. Can be used by anyone at any time. As this application can be very useful for the earlier and fast classification of arrhythmia it we be used by many patients suffering by it. Experts guidance is not required when we have a app that can be used by anyone. Data of the patient will be securely stored and maintained for future purposes

### 3.4 Problem Solution fit

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Who is your customer? i.e. working parents of 0-5 y.o. kids  <b>Patients are customers here</b>	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.  <ul style="list-style-type: none"> <li>Need of experts</li> <li>Budget problem</li> </ul>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking  The algorithms used for arrhythmia classification incorporate preprocessing, feature extraction, and classification. Classification becomes complicated when class overlap and class imbalance problems occur together
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.  <b>The problem here is classification of arrhythmia takes more time and requires experts. It can't be done anytime by anyone</b>	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in neighborhoods.  <b>Arrhythmia means heart is not beating properly. This can cause anything to form cardiac arrest to death.</b>	<b>7. BEHAVIOUR</b> <span>BE</span> What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)  <b>The problem of arrhythmia is directly connected to patient. When he/she feels irregular heartbeat or any breathing issues he can address the issue.</b>
Identify strong TR & EM	<b>3. TRIGGERS</b> <span>TR</span> What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.  <b>The point that triggers the customers to use this is that it doesn't require anyone's assistance.</b>	<b>10. YOUR SOLUTION</b> <span>SL</span> If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.  The ECG signals can capture the heart's rhythmic irregularities, commonly known as arrhythmias. we propose a two-dimensional (2-D) convolutional neural network (CNN) model for the classification of ECG signals into eight classes; namely, normal beat, paced beat, right bundle branch block beat, left bundle branch block beat, atrial premature contraction beat, ventricular flutter wave beat, and ventricular escape beat.	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE:</b> What kind of actions do customers take online? Extract online channels from #7  <b>8.2 OFFLINE:</b> What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.  <ul style="list-style-type: none"> <li>Users need to upload of image of the ECG.</li> <li>Patients need to undergo scan to get images of the heartbeat.</li> </ul>

## **4.REQUIREMENT ANALYSIS**

### **4.1 Functional requirement**

**FR-1** Registration through Form Registration through mail.

**FR-2** Confirmation via Email Confirmation via OTP

**FR-3** Get User Input Upload image as jpeg Upload image as png

**FR-4** Save Image Images are saved in the uploads folder

**FR-5** Chat with Doctor Consult with Doctor

**FR-6** Report Generation Get complete Report

### **4.2 Non-Functional requirements**

**NFR-1** Usability Classification of Arrhythmia with the help of AI.

**NFR-2** Security User's data cannot be accessed by unauthorised people.

**NFR-3** Reliability The system performs without failure.

**NFR-4** Performance High accuracy.

**NFR-5** Availability Anyone who is authorised.

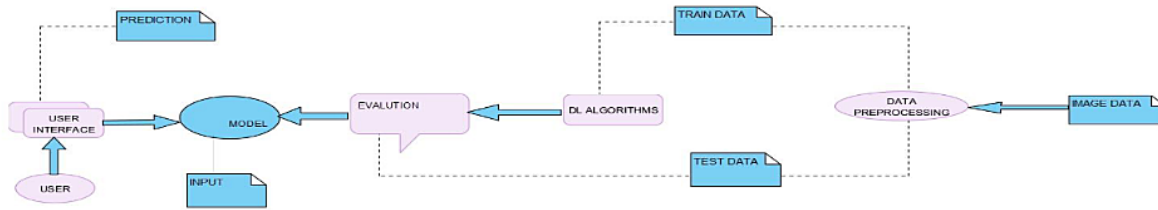
**NFR-6** Scalability Does not affect the performance even though used by many users.

## **5.PROJECT DESIGN**

### **5.1 Data Flow Diagrams**

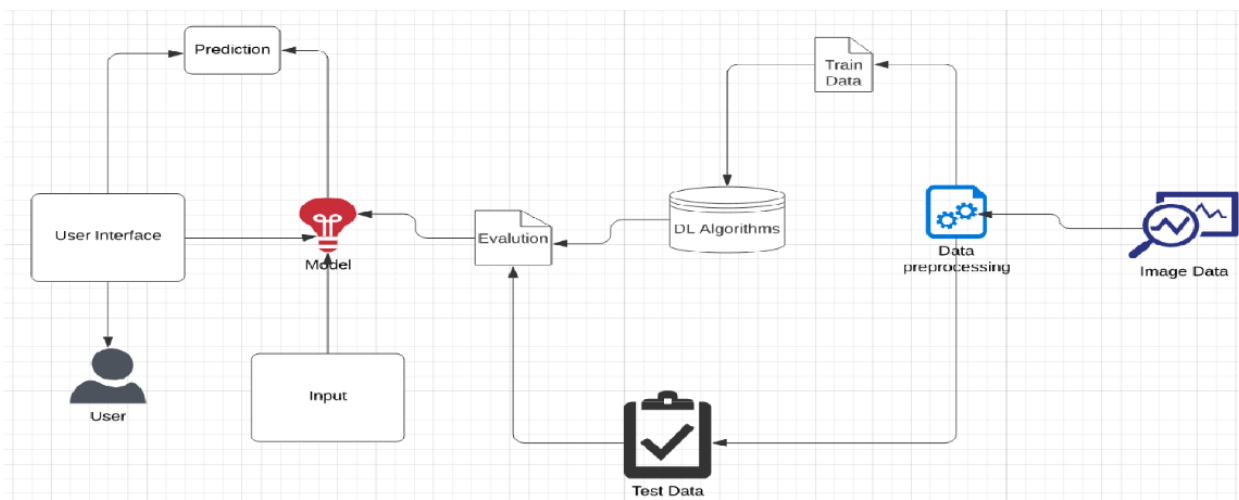
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

.



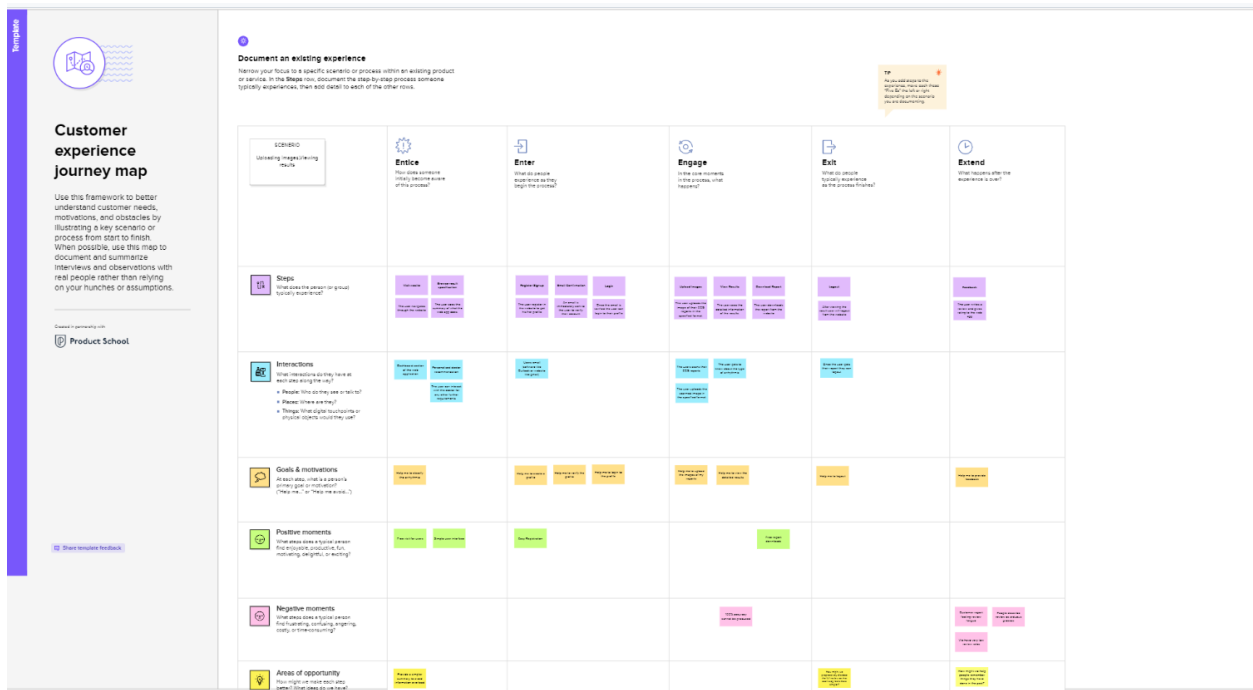
## 5.2 Solution & Technical Architecture

Technical architecture—which is also often referred to as application architecture, IT architecture, business architecture, etc.—refers to creating a structured software solution that will meet the business needs and expectations while providing a strong technical plan for the growth of the software application through its lifetime. IT architecture is equally important to the business team and the information technology team. Technical architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of technical architects is to achieve all the business needs with an application that is optimised for both performance and security.





## 5.3 User Stories



## 6.PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation SPRINT 1

#### ● Download the dataset

Google Gmail YouTube Maps IBM aws educate Arrhythmia (2) - IB...

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

IBM-EPBL / IBM-Project-3155-1658503360 (Public)

Watch 1 Fork 1 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main IBM-Project-3155-1658503360 / Project Development Phase / Sprint 1 / Dataset Collection / train

Go to file Add file ...

sandeepvarmass 100+ files acb2e64 10 days ago History

Left Bundle Branch Block	100+ files	10 days ago
Normal	100+ files	10 days ago
Premature Atrial Contraction	100+ files	10 days ago
Premature Ventricular Contractions	100+ files	10 days ago
Right Bundle Branch Block	100+ files	10 days ago
Ventricular Fibrillation	100+ files	10 days ago

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

## • Image Preprocessing

### Import the ImageGenerator

```
Import the ImageDataGenerator library

[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

### Configure ImageGenerator

```
Configure ImageDataGenerator Class

[ ] train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, vertical_flip=True, horizontal_flip=True)

[ ] test_data=ImageDataGenerator(rescale=1./255)
```

### Apply ImageGenerator functionality and trainset

```
Apply ImageDataGenerator functionality to trainset and testset

[ ] x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/Project Development Phase/data/train", target_size=(64,64), class_mode="categorical", batch_size=128)
Found 15341 images belonging to 6 classes.

[ ] x_test=test_data.flow_from_directory(r"/content/drive/MyDrive/Project Development Phase/data/test", target_size=(64,64), class_mode="categorical", batch_size=128)
Found 6825 images belonging to 6 classes.

[ ] x_train.class_indices

{'Left Bundle Branch Block': 0,
 'Normal': 1,
 'Premature Atrial Contraction': 2,
 'Premature Ventricular Contractions': 3,
 'Right Bundle Branch Block': 4,
 'Ventricular Fibrillation': 5}
```

## • Model Building

### Import Libraries

```
MODEL BUILDING

Import the libraries

[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Convolution2D, MaxPooling2D, Flatten
```

### Initialize the model

```
Initialize the model

[ ] model=Sequential()
```

### Adding CNN layers

```
[ ] model.add(Convolution2D(32,(3,3),activation="relu",strides=(1,1),input_shape=(64,64,3)))
```

```
[ ] model.add(MaxPooling2D(pool_size=(2,2)))
```

```
[ ] model.add(Flatten())
```

```
[ ] model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
flatten (Flatten)	(None, 30752)	0
Total params: 896		
Trainable params: 896		

Activate Windows  
Go to Settings to activate Windows

## Adding Dense Layers

### Adding Dense layer

#### Hidden layer

```
[ ] model.add(Dense(500,activation="relu"))
```

```
[ ] model.add(Dense(500,activation="relu"))
```

#### Output layer

```
[ ] model.add(Dense(6,activation="softmax"))
```

## Train the model

```
[ ] Epoch 1/5  
120/120 [=====] - 140s 1s/step - loss: 0.1920 - accuracy: 0.9401 - val_loss: 0.4968 - val_accuracy: 0.8731  
Epoch 2/5  
120/120 [=====] - 147s 1s/step - loss: 0.1607 - accuracy: 0.9512 - val_loss: 0.5703 - val_accuracy: 0.8727  
Epoch 3/5  
120/120 [=====] - 142s 1s/step - loss: 0.1358 - accuracy: 0.9572 - val_loss: 0.4914 - val_accuracy: 0.8831  
Epoch 4/5  
120/120 [=====] - 140s 1s/step - loss: 0.1181 - accuracy: 0.9640 - val_loss: 0.5450 - val_accuracy: 0.8794  
Epoch 5/5  
120/120 [=====] - 133s 1s/step - loss: 0.1109 - accuracy: 0.9666 - val_loss: 0.4703 - val_accuracy: 0.8801  
<keras.callbacks.History at 0x7f527adb750>
```

```
[ ] model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/5  
120/120 [=====] - 127s 1s/step - loss: 0.1022 - accuracy: 0.9692 - val_loss: 0.5888 - val_accuracy: 0.8659  
Epoch 2/5  
120/120 [=====] - 139s 1s/step - loss: 0.0934 - accuracy: 0.9710 - val_loss: 0.5789 - val_accuracy: 0.8689  
Epoch 3/5  
120/120 [=====] - 128s 1s/step - loss: 0.0862 - accuracy: 0.9729 - val_loss: 0.4989 - val_accuracy: 0.8848  
Epoch 4/5  
120/120 [=====] - 128s 1s/step - loss: 0.0778 - accuracy: 0.9765 - val_loss: 0.6542 - val_accuracy: 0.8759  
Epoch 5/5  
120/120 [=====] - 137s 1s/step - loss: 0.0793 - accuracy: 0.9745 - val_loss: 0.5369 - val_accuracy: 0.8844  
<keras.callbacks.History at 0x7f527adc0a50>
```

Activate Windows  
Go to Settings to activate Windows

## Save the model

### Save the model

```
[ ] model.save('arrhythmia.h5')
```

## Testing Model


```
Testing the model

[ ] import numpy as np
    from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing import image

[ ] model=load_model('arrhythmia.h5')

[ ] img=image.load_img("/content/drive/MyDrive/Project Development Phase/data/test/Right Bundle Branch Block/fig_101.png",target_size=(64,64))

[ ] img



[ ] x=image.img_to_array(img)

[ ] x

[[255., 255., 255.]], dtype=float32)

[ ] pred=model.predict(x)
    1/1 [=====] - 0s 43ms/step

[ ] pred
    array([[0., 0., 0., 0., 1., 0.]], dtype=float32)

[ ] index=['Left Bundle Branch Block',
          'Normal',
          'Premature Atrial Contraction',
          'Premature Ventricular Contractions',
          'Right Bundle Branch Block',
          'Ventricular Fibrillation']

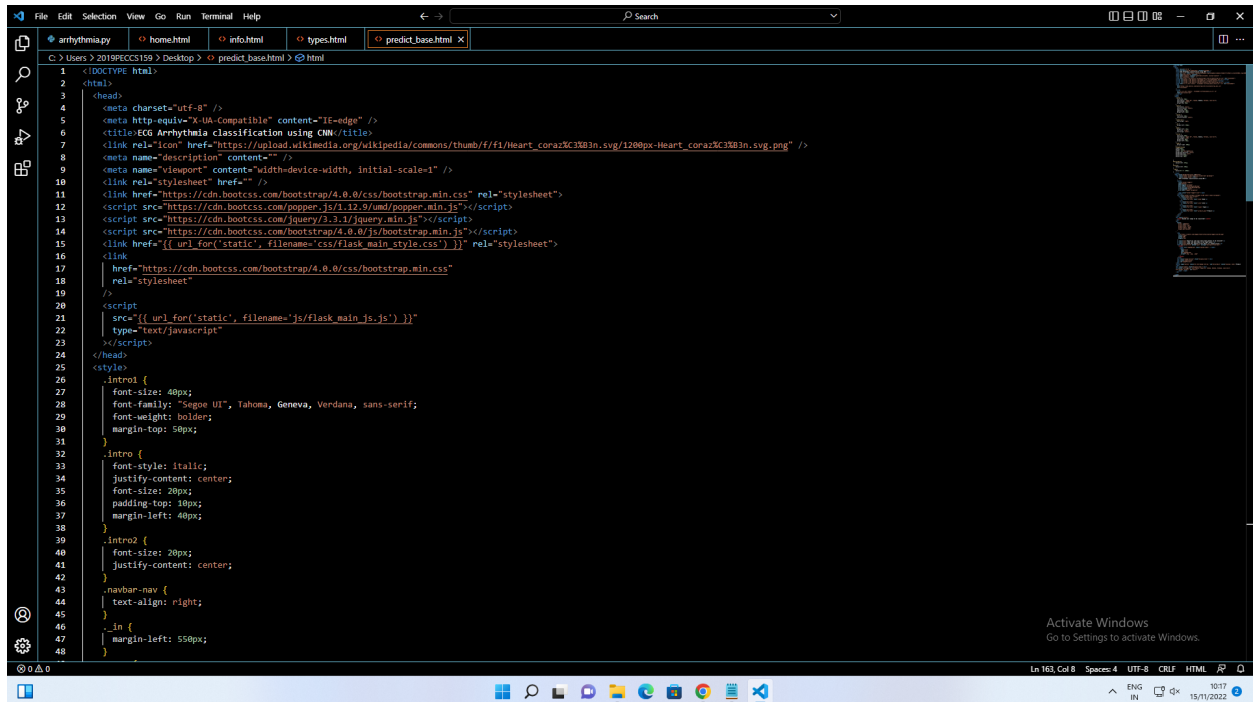
[ ] index[np.argmax(pred)]
    'Right Bundle Branch Block'
```

## SPRINT 2

### home.html

```
File Edit Selection View Go Run Terminal Help
arrhythmia.py home.html x
C:\Users\2019PCCS159\Desktop> home.html > html > body > nav.navbar.navbar-expand-lg.navbar-dark.bg-danger > div.navbar-collapse.collapse.w-100.order-3.dual-collapse2 > ul.navbar-nav.ml-auto > li.nav-item
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8" />
5 <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6 <title>ECG Arrhythmia classification using CNN</title>
7 <link rel="icon" href="https://upload.wikimedia.org/wikipedia/commons/thumb/4/43/Heart_coronary3D03n.svg/1200px-Heart_coronary3D03n.svg.png" />
8 <meta name="description" content="" />
9 <meta name="viewport" content="width=device-width, initial-scale=1" />
10 <link rel="stylesheet" href="" />
11 <link
12 href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/css/bootstrap.min.css"
13 rel="stylesheet" />
14 </head>
15 <body>
16 <style>
17 .intro1 {
18 font-size: 40px;
19 font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
20 font-weight: bold;
21 }
22 .intro {
23 font-style: italic;
24 justify-content: center;
25 font-size: 20px;
26 padding-top: 50px;
27 margin-left: 200px;
28 margin-right: 200px;
29 }
30 .intro2 {
31 font-size: 20px;
32 justify-content: center;
33 }
34 .navbar-nav {
35 text-align: right;
36 }
37 .in {
38 margin-left: 50px;
39 }
40 </style>
41 <body style="background-color: #f4a222">
42 <nav class="navbar navbar-expand-lg navbar-dark bg-danger">
43 <a class="navbar-brand" href="#">
44 ECG Arrhythmia classification using CNN</a>
45 </nav>
46 <button
47 class="navbar-toggler">
```

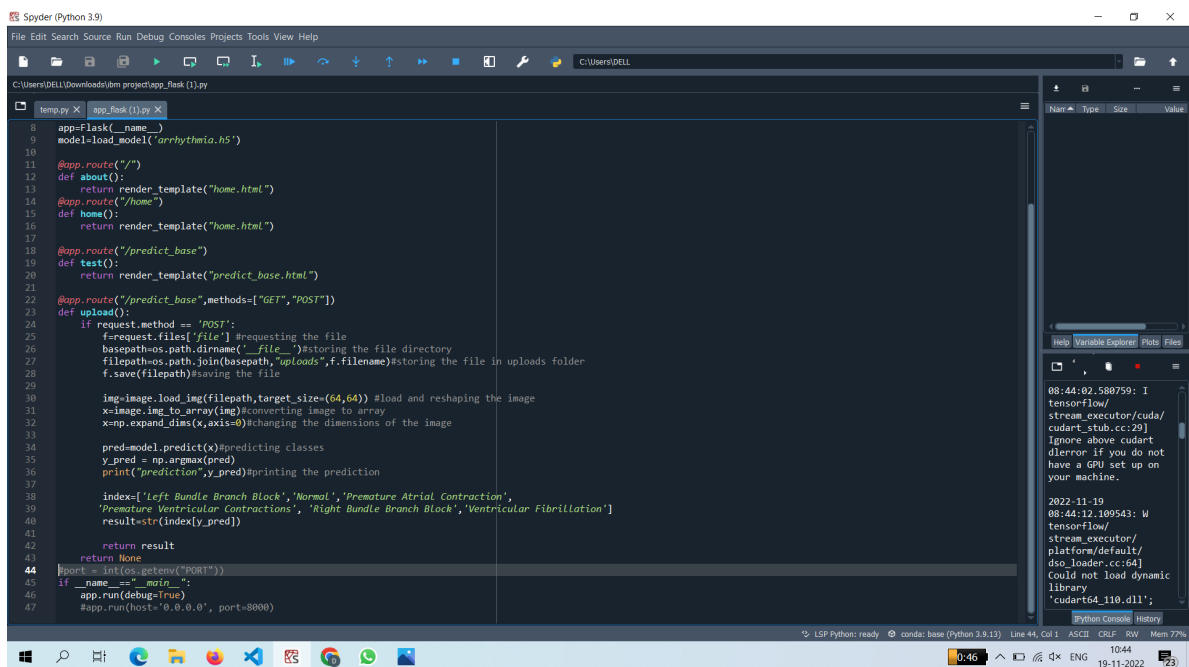
# predict.html



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6   <title>ECG Arrhythmia Classification using CNN</title>
7   <link rel="icon" href="https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Heart_coraz%C3%83n.svg/1200px-Heart_coraz%C3%83n.svg.png" />
8   <meta name="description" content="" />
9   <meta name="viewport" content="width=device-width, initial-scale=1" />
10  <link rel="stylesheet" href="" />
11  <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
12  <script src="https://cdn.bootcss.com/popper.js/1.12.0/umd/popper.min.js"></script>
13  <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
14  <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
15  <link href="{{ url_for('static', filename='css/flask_main_style.css') }}" rel="stylesheet">
16  <link
17    href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
18    rel="stylesheet"
19  />
20  <script>
21    <script src="{{ url_for('static', filename='js/flask_main.js.js') }}"
22    type="text/javascript">
23  </script>
24 </head>
25 <style>
26   .intro1 {
27     font-size: 40px;
28     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
29     font-weight: bolder;
30     margin-top: 50px;
31   }
32   .intro {
33     font-style: italic;
34     justify-content: center;
35     font-size: 20px;
36     padding-top: 10px;
37     margin-left: 40px;
38   }
39   .intro2 {
40     font-size: 20px;
41     justify-content: center;
42   }
43   .navbar-nav {
44     text-align: right;
45   }
46   .in {
47     margin-left: 50px;
48   }
49 </style>
```

## SPRINT 3

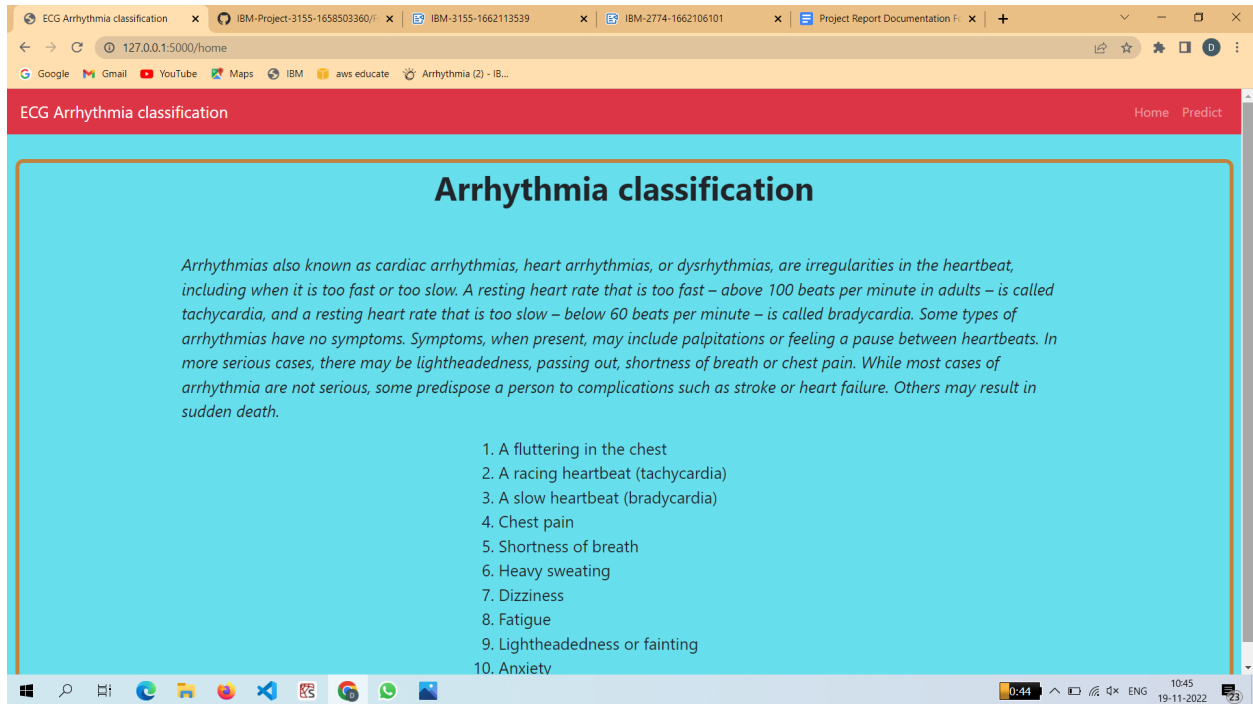
# app.flask.py



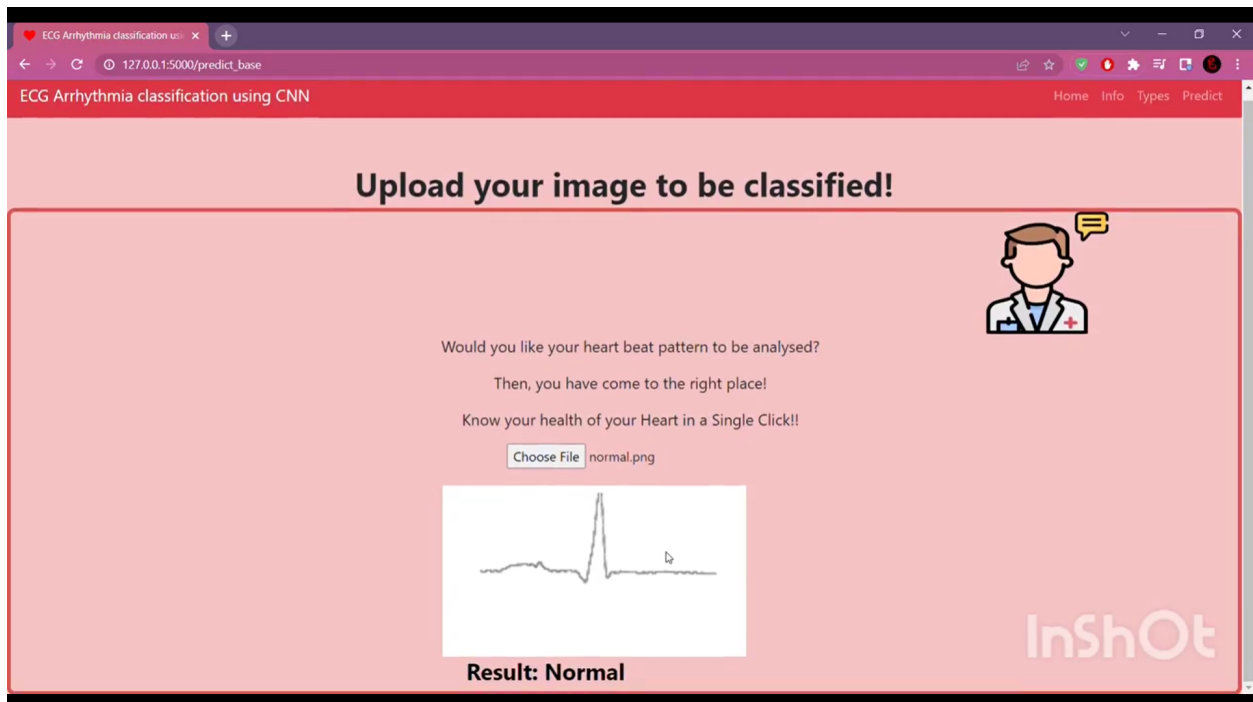
```
8 app=Flask(__name__)
9 model=load_model('arrhythmia.h5')
10
11 @app.route("/")
12 def about():
13     return render_template("home.html")
14 @app.route("/home")
15 def home():
16     return render_template("home.html")
17
18 @app.route("/predict_base")
19 def test():
20     return render_template("predict_base.html")
21
22 @app.route("/predict_base", methods=['GET', 'POST'])
23 def upload():
24     if request.method == 'POST':
25         f=request.files['file'] #requesting the file
26         basepath=os.path.dirname(__file__)#storing the file directory
27         filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
28         f.save(filepath)#saving the file
29
30         img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
31         x=image.img_to_array(img)#converting image to array
32         x=np.expand_dims(x,axis=0)#changing the dimensions of the image
33
34         pred=model.predict(x)#predicting classes
35         y_pred = np.argmax(pred)
36         print('prediction',y_pred)#printing the prediction
37
38         index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
39               'Premature Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
40         result=str(index[y_pred])
41
42         return result
43     return None
44
45 port = int(os.getenv("PORT"))
46 if __name__ == '__main__':
47     app.run(debug=True)
48     app.run(host='0.0.0.0', port=8000)
```

## SPRINT4

### home page



### predict page



## 6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint Start Date	Story Points Completed	Sprint Release Date
Sprint-1	20	2 days	10-nov-2022	10-nov-2022	20	10-nov-2022
Sprint-2	20	2 days	12-nov-2022	12-nov-2022	40	12-nov-2022
Sprint-3	20	2 days	14-nov-2022	14-nov-2022	60	14-nov-2022
Sprint-4	20	2 days	18-nov-2022	18-nov-2022	80	18-nov-2022

## 6.3 Reports from JIRA

The screenshot displays a JIRA report titled "Document an existing experience" which is a Customer Experience Journey Map. The report is structured as a grid with five main stages: Discover, Enter, Engage, Exit, and Extend. Each stage contains a list of customer journey steps, interactions, goals, and motivations, as well as positive and negative moments and areas of opportunity. The report is presented in a web browser window with multiple tabs open, including "ECG Arrhythmia classification", "IBM-Project-3155-1658503360", "IBM-3155-1662113539", "IBM-2774-1662106101", and "Project Report Documentation F...". The browser address bar shows the GitHub repository URL: "github.com/IBM-EPBL/IBM-Project-3155-1658503360/blob/main/Project%20design%20%26%20Planning/project%20design%20phase%20Customer%20Journey.pdf". The JIRA report is 77.6 KB and has a "Download" button. The sidebar on the left contains a "Customer experience journey map" section with a "Product School" link. The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 1:37 on 19-11-2022.

## 7. CODING & SOLUTIONING

### 7.1 Feature 1

#### **Classifies the type of arrhythmia with one click**

```
import os
import numpy as np
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('arrhythmia.h5')

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

@app.route("/predict_base")
def test():
    return render_template("predict_base.html")

@app.route("/predict_base",methods=["GET","POST"])
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
return None
#port = int(os.getenv("PORT"))
if __name__=="__main__":
    app.run(debug=True)
    #app.run(host='0.0.0.0', port=8000)

```

## 7.2 Feature 2

### User friendly interface

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>ECG Arrhythmia classification</title>

    <meta name="description" content="" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="" />
    <link
        href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
        rel="stylesheet"
    />
</head>

```

```
<style>
.intro1 {
  font-size: 40px;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  font-weight: bolder;
}
.intro {
  font-style: italic;
  justify-content: center;
  font-size: 20px;
  padding-top: 50px;
  margin-left: 200px;
  margin-right: 200px;
}
.intro2 {
  font-size: 20px;
  justify-content: center;
}

.navbar-nav {
  text-align: right;
}
._in {
  margin-left: 550px;
}
</style>
<body style="background-color: #67deeb">
  <nav class="navbar navbar-expand-lg navbar-dark bg-danger">
    <a class="navbar-brand" href="#">
```

```
>ECG Arrhythmia classification</a>
>
<button
  class="navbar-toggler"
  type="button"
  data-toggle="collapse"
  data-target="#navbarNavAltMarkup"
  aria-controls="navbarNavAltMarkup"
  aria-expanded="false"
  aria-label="Toggle navigation"
>
  <span class="navbar-toggler-icon"></span>
</button>
<div class="navbar-collapse collapse w-100 order-3 dual-collapse2">
  <ul class="navbar-nav ml-auto">
    <li class="nav-item">
      <a class="nav-link" href="/home">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/predict_base">Predict</a>
    </li>
  </ul>
</div>
</nav>
<div
  style="
    border: #c0833e;
    border-width: 5px;
    border-style: solid;
```

border-radius: 10px;

rate that is too slow – below 60 beats per minute – is called bradycardia.

Some types of arrhythmias have no symptoms.

Symptoms, when present, may include palpitations or feeling a pause between heartbeats.

In more serious cases, there may be lightheadedness, passing out, shortness of breath or chest pain.

While most cases of arrhythmia are not serious, some predispose a person to complications such as stroke or heart failure.

Others may result in sudden

height: max-content;

margin-top: 30px;

margin-left: 10px;

margin-right: 10px;

"

>

<div class="intro1">

<center>Arrhythmia classification</center>

</div>

<div>

<p class="intro">

Arrhythmias also known as cardiac arrhythmias, heart arrhythmias, or dysrhythmias, are irregularities in the heartbeat,

including when it is too fast or too slow.

A resting heart rate that is too fast – above 100 beats per minute in adults – is called tachycardia,

and a resting heart death.

</p>

</div>

```
<div class="intro2">
```

```
<ol class="_in">
```

```
<li>A fluttering in the chest</li>
```

```
<li>A racing heartbeat (tachycardia)</li>
```

```
<li>A slow heartbeat (bradycardia)</li>
```

```
<li>Chest pain</li>
```

```
<li>Shortness of breath</li>
```

```
<li>Heavy sweating</li>
```

```
<li>Dizziness</li>
```

```
<li>Fatigue</li>
```

```
<li>Lightheadedness or fainting</li>
```

```
<li>Anxiety</li>
```

```
</ol>
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

## 8. Testing

### 8.1 User Acceptance Testing

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	19
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	10	26
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	4	1	1	6
Totals	22	15	10	30	77

## 9. Performance Metrics

```
[ ] model.fit(x_train,epochs=1,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))

120/120 [=====] - 182s 2s/step - loss: 0.1781 - accuracy: 0.9447 - val_loss: 0.5165 - val_accuracy: 0.8362
<keras.callbacks.History at 0x7efd403a8150>

[ ] history=model.fit(x_train,epochs=1,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))

120/120 [=====] - 138s 1s/step - loss: 0.0979 - accuracy: 0.9687 - val_loss: 0.5897 - val_accuracy: 0.8620

▶ model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))

Epoch 1/5
120/120 [=====] - 132s 1s/step - loss: 0.0981 - accuracy: 0.9686 - val_loss: 0.5342 - val_accuracy: 0.8716
Epoch 2/5
120/120 [=====] - 127s 1s/step - loss: 0.0900 - accuracy: 0.9727 - val_loss: 0.6188 - val_accuracy: 0.8668
Epoch 3/5
120/120 [=====] - 126s 1s/step - loss: 0.0926 - accuracy: 0.9725 - val_loss: 0.6126 - val_accuracy: 0.8637
Epoch 4/5
120/120 [=====] - 127s 1s/step - loss: 0.0836 - accuracy: 0.9737 - val_loss: 0.5963 - val_accuracy: 0.8623
Epoch 5/5
120/120 [=====] - 129s 1s/step - loss: 0.0743 - accuracy: 0.9756 - val_loss: 0.4265 - val_accuracy: 0.8923
<keras.callbacks.History at 0x7f866830d350>
```

## 10.ADVANTAGES & DISADVANTAGES

### ADVANTAGES

- Provides accurate results and detailed information required by the users or patients.
- Users or customers can easily use the app because of its user-friendly interface and simplicity.
- Can be used by anyone at any time.
- As this application can be very useful for the earlier and fast classification of arrhythmia it we be used by many patients suffering from it.
- Experts guidance is not required when we have a app that can be used by anyone.
- Data of the patient will be securely stored and maintained for future purposes.

## 11.Conclusion

Arrhythmia is a severe CVD that can be predicted via ECG segment processing. Arrhythmia must be accurately diagnosed and prevented early to reduce cardiac disease. Our proposed system model met the study's primary goal of assisting doctors in swiftly determining the kind of ECG or verifying their diagnostics in a medical context while maintaining a high level of precision and cost. In this work, a CNN-Bi-LSTM model

is proposed to categorise five categories of ECG fragments to construct an effective and resilient autonomous computer-aided diagnosis system. The developed network achieved maximum accuracies of 100%, 98.0%, and 98.0% of training, validation, and testing using MIT-BIH data set. In comparison, the St-Petersburg data set achieved 98.0%, 95.0%, and 95.0% accuracies of training, validation, and testing in identifying arrhythmia. This research showed many advantages, including its ability to help clinicians reliably make ECG recording-related clinical decisions. Moreover, it was intended to be as simple as possible while delivering the most significant performance. The described method is straightforward for health professionals and does not involve signal modification or feature extraction. Additionally, this research focused only on one kind of CVD, namely, arrhythmia, whereas the manifestations of cardiac disease are often complex and varied. As a result, more types of ECG data will need to be added to broaden the scope of the planned network.

## **12.FUTURE SCOPE**

- To upload dataset and store patient details
- Online doctor consultation

## **13.APPENDIX**

### **Source Code**

#### **1) Model Building Code**

```
from google.colab import drive  
  
drive.mount('/content/drive')  
  
cd /content/drive/MyDrive/Project Development Phase  
  
#Import the ImageDataGenerator library  
  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
#Configure ImageDataGenerator Class
```

```
train_datagen=ImageDataGenerator(rescale=1./255, zoom_range=0.2, vertical_flip=True, horizontal_flip=True)
```

```
test_data=ImageDataGenerator(rescale=1./255)
```

```
#Apply ImageDataGenerator functionality to trainset and testset
```

```
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/ProjectDevelopment  
Phase/data/train", target_size=(64,64), class_mode="categorical", batch_size=128)
```

```
x_test=test_data.flow_from_directory(r"/content/drive/MyDrive/ProjectDevelopment  
Phase/data/test", target_size=(64,64), class_mode="categorical", batch_size=128)
```

```
x_train.class_indices
```

```
#MODEL BUILDING
```

```
#Import the libraries
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Convolution2D, MaxPooling2D, Flatten
```

```
#Initialize the model
```

```
model=Sequential()
```

```
#Adding CNN layers
```

```
model.add(Convolution2D(32,(3,3), activation="relu", strides=(1,1), input_shape=(64,64,3)))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Flatten())
```

```
model.summary()
```

```
#Adding Dense layer
```

```
#Hidden layer
```

```
model.add(Dense(500, activation="relu"))
```



```
model.add(Dense(500,activation="relu"))
```

### **#Output layer**

```
model.add(Dense(6,activation="softmax"))
```

### **#Configure the learning process**

```
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accuracy'])
```

```
len(x_train)
```

### **#Train the model**

```
model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

```
model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

```
model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

```
model.fit(x_train,epochs=5,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

### **#Save the model**

```
model.save('arrhythmia.h5')
```

### **#Testing the model**

```
import numpy as np
```

```
from tensorflow.keras.models import load_model
```

```
from tensorflow.keras.preprocessing import image
```

```
img=image.load_img("/content/drive/MyDrive/Project Development Phase/data/test/Premature Ventricular Contractions/VEBfig_11.png",target_size=(64,64))
```

```
x=image.img_to_array(img)
```

```
x=np.expand_dims(x,axis=0)
```

```
pred=model.predict(x)
```

```
index=['Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature  
Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation']
```

```
index[np.argmax(pred)]
```

### **App.py source code**

```
import os
```

```
import numpy as np
```

```
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('arrhythmia.h5')
```

```
@app.route("/")
```

```
def about():
```

```
    return render_template("home.html")
```

```
@app.route("/home")
```

```
def home():
```

```
    return render_template("home.html")
```

```

@app.route("/predict_base")
def test():
    return render_template("predict_base.html")

@app.route("/predict_base",methods=["GET","POST"])
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
    return None

#port = int(os.getenv("PORT"))
if __name__=="__main__":
    app.run(debug=True)

```

```
#app.run(host='0.0.0.0', port=8000)
```

**GitHub**

<https://github.com/IBM-EPBL/IBM-Project-3155-1658503360>

**Project Demo Link**

<https://github.com/IBM-EPBL/IBM-Project-3155-1658503360/upload/main/project%20Demo>