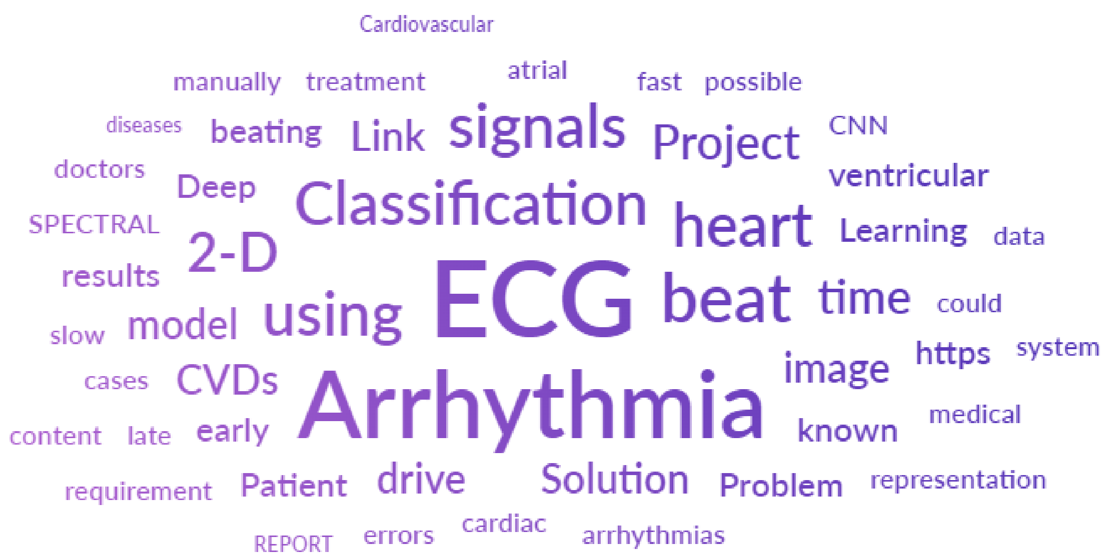
REPORT ON CLASSIFICATION OF ARRHYTHMIA USING DEEP LEARNING WITH 2-D ECG SPECTRAL IMAGE

- ARTIFICIAL INTELLIGENCE



BATCH OF STUDENTS
PREPARED BY :

1. BHUVANA K
2. KALAISELVI S
3. KANISHYA A
4. KIRUTHIKA V

1. INTRODUCTION:

Project overview:

- Cardiovascular diseases (CVDs) are the leading cause of human death, with over 17 million people known to lose their lives annually due to CVDs .

- A classification model to identify CVDs at their early stage could effectively reduce the mortality rate by providing a timely treatment.

- One of the common sources of CVDs is cardiac arrhythmia, where heartbeats are known to deviate from their regular beating pattern. A normal heartbeat varies with age, body size, activity, and emotions. In cases where the heartbeat feels too fast or slow, the condition is known as palpitations.

- An arrhythmia does not necessarily mean that the heart is beating too fast or slow, it indicates that the heart is following an irregular beating pattern. It could mean that the heart is beating too fast—tachycardia (more than 100 beats per minute (bpm)), or slow—bradycardia (less than 60 bpm), skipping a beat, or in extreme cases, cardiac arrest. Some other common types of abnormal heart rhythms include atrial fibrillation, atrial flutter, and ventricular fibrillation.

- 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.

Purpose:

The major purpose of this model is to get accurate and reliable test results (Arrhythmia ECG report) within the few instance of time. This may reduces the burden for waiting in the hospitals, time.

2.LITERATURE SURVEY:

Existing Problem:

Problem Statement (Problem to be solved)

1. Patient suffering from **Arrhythmia** has to be treated as early as possible and each arrhythmia has its own treatment so it might be risky to the patient**, if detected late or the possibility of medical errors by doctors as ECG are reviewed manually**

Idea / Solution description

1. **Late detection and medical errors has to be avoided**
2. **It could be done so by creating an app for classifying the arrhythmia accurately and as early as possible**

Novelty / Uniqueness

1. **The Solution provided is quite unique as 2-D ECG is fed to the model and the image given as input is classified by using CNN to classify the Arrhythmia**

References:

1.Classification on Arrhythmia by using deep learning with 2-D ECG spectral representation---Amin Ullah, Mohammad Anwar, Muhammad Bilal, Raja Majid Mehmood. in 2020.
LINK:
https://www.researchgate.net/publication/341623436_Classification_of_Arrhythmia_by_Using_Deep_Learning_with_2-D_ECG_Spectral_Image_Representation

2.Automatic Classification of Cardiac Arrhythmias based on ECG Signals Using Transferred Deep Learning Convolution Neural Network---P.Giriprasad Gaddam, A. Sanjeeva Reddy and r.V. Sreehari---2021.

Link:https://iopscience.iop.org/article/10.1088/1742-6596/2089/1/012058/pdf

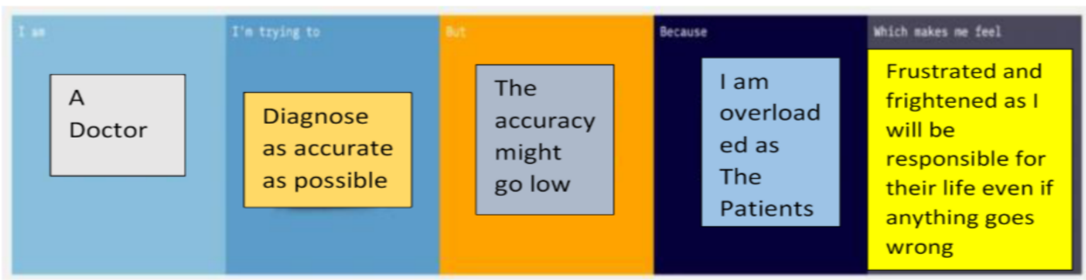
3. Computer-Aided Diagnostics of Heart Disease Risk Prediction Using Boosting Support Vector Machine---Ebeneezer Owusu, prince Boakye-Sekyerehene,Just ice Kwame Appati and Julius Yaw Ludu--Dec 2021
Link:https://downloads.hindawi.com/journals/cin/2021/3152618.pdf

4.Classification of Arrhythmia from ECG Signals using MATLAB [International Journal of Engineering and Management Rese---Priyanka Mayapur--2019
Link:
https://www.researchgate.net/publication/330185548_Classification_of_Arrhythmia_from_ECG_Signals_

using_MATLAB_International_Journal_of_Engineering_and_Management_Research

Problem Statement Definition:

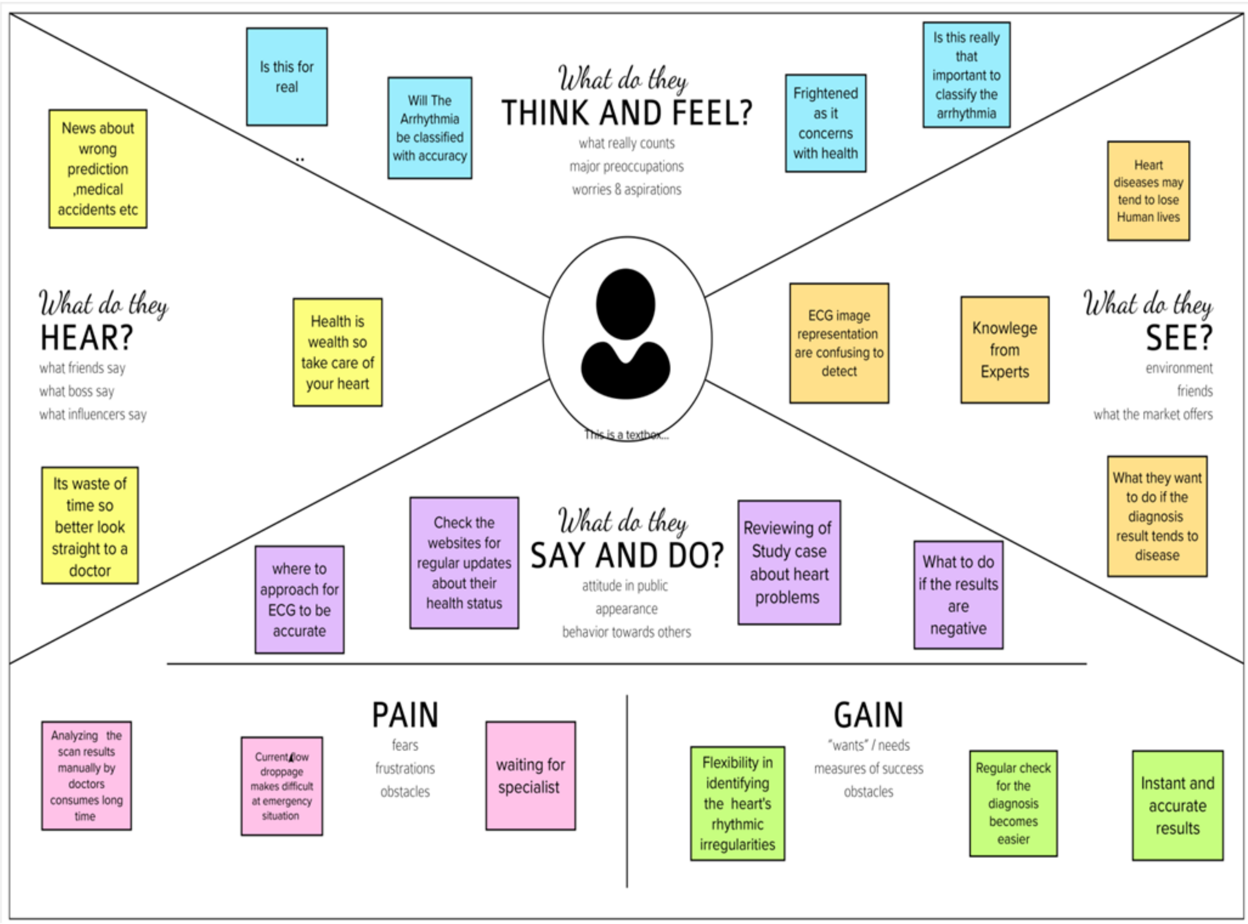
Patient suffering from Arrhythmia has to be treated as early as possible and each arrhythmias has its own treatment so it might be risk to the patient, if detected late or the possibility of medical errors by doctors as ECG are reviewed manually.



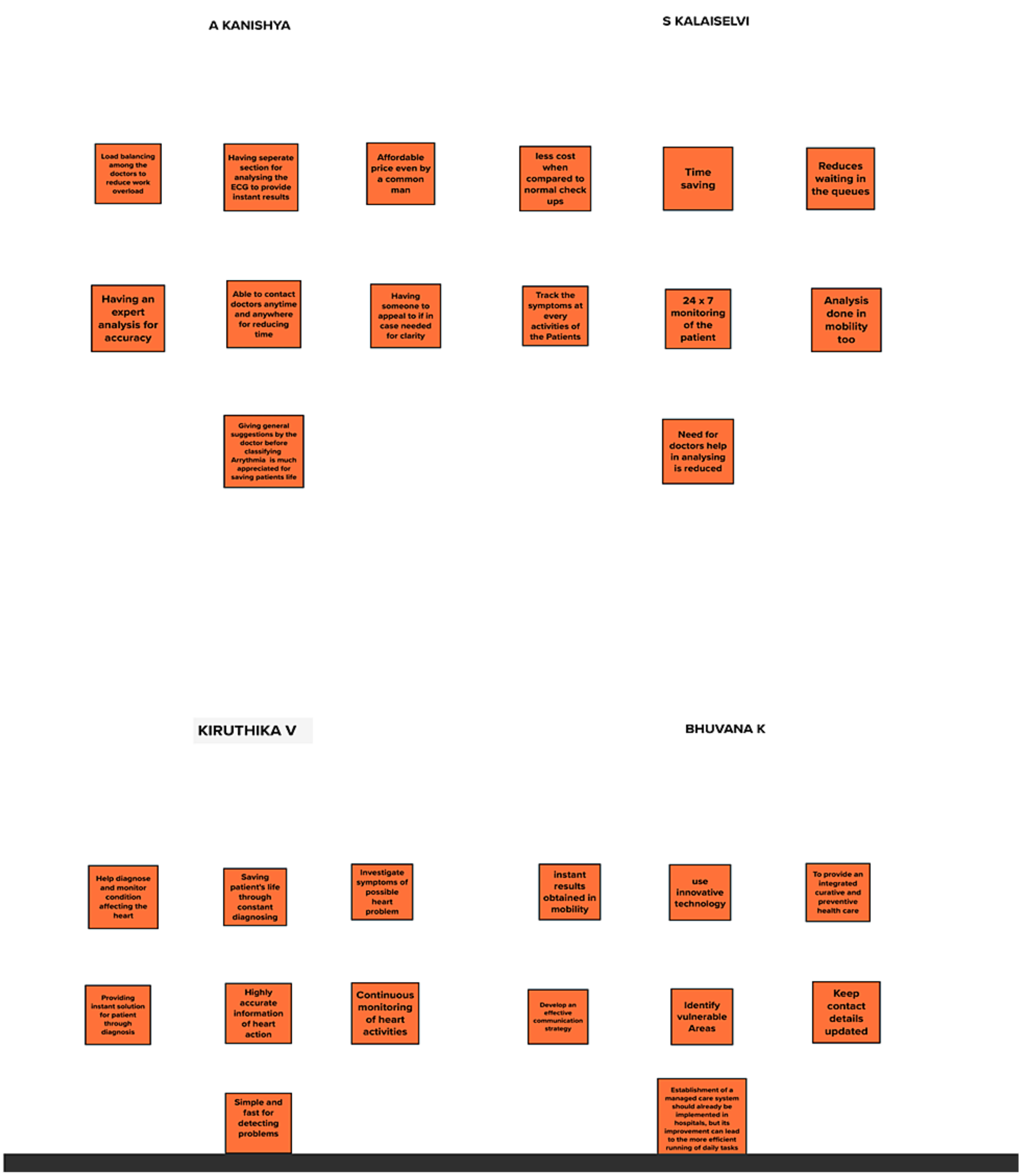
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A Patient	Contact doctors for diagnosing the heart	It takes longer time	I have to wait for other patients to get treated	Frightened as my life could be in danger and has to be treated as soon as possible
PS-2	A Doctor	Diagnose as accurate as possible	The accuracy might go low	I am overloaded as The Patients out numbers the doctors	Frustrated and frightened as I will be responsible for their life even if anything goes wrong

3.IDEATION PHASE:

Empathy map:



Ideation & Brainstroming:



Proposed Solution fit:

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Patient suffering from **Arrhythmia** has to be treated as early as possible and each arrhythmia has its own treatment so it might be risky to the patient**, if detected late or the possibility of medical errors by doctors as ECG are reviewed manually**
2.	Idea / Solution description	• Late detection and medical errors has to be avoided • It could be done so by creating an app for classifying the arrhythmia accurately and as early as possible
3.	Novelty / Uniqueness	The Solution provided is quite unique as 2-D ECG is fed to the model and the image given as input is classified by using CNN to classify the Arrhythmia
4.	Social Impact / Customer Satisfaction	• It doesn't have any impact on the environment • The customer will also gets satisfied as the above mentioned problem is solved and the same app can be used any number of times for multiple person to detect and classify Arrhythmia
5.	Business Model (Revenue Model)	• Since one app can be used to classify the Arrhythmia any number of times for various person so it is a one time investment by the customer and is quite affordable as well • At the same time it also brings revenue to the Organisation
6.	Scalability of the Solution	It is also possible to scale the app[model] further by increasing the images fed to the app[model] thereby making some small changes so that it could be combined with other apps to build integrated app which serves well for customer needs

4.REQUIREMENT ANALYSIS:

Functional requirement:

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	App Installation	• Installation through link • Installation through play store
FR-4	User Interface	It should be user friendly and also should have some virtual memory by default to store recent information
FR-5	Dataset	• Collecting it personally • Through online websites like Kaggle • Through google forms
FR-6	Dash Board	Python related environments like Jupyter notebook ,Google colab etc for creating the application
FR-7	Database	• Stored in cloud for seamless connectivity • The result been classified is stored in cloud so it could be accessed later if required
FR-8	Server	It stores, sends and receives data. It is mainly required for connecting the application with the user in the backend
FR-9	API	It is used for enabling two software components to communicate with each other
FR-10	User Requirements	User should be able to provide or pass their 2D ECG image to the system without any difficulties
FR-11	React JS	We are using • react as the front-end for creating interactive user interface and • Node JS as our backend for server side programming and for back-end API services

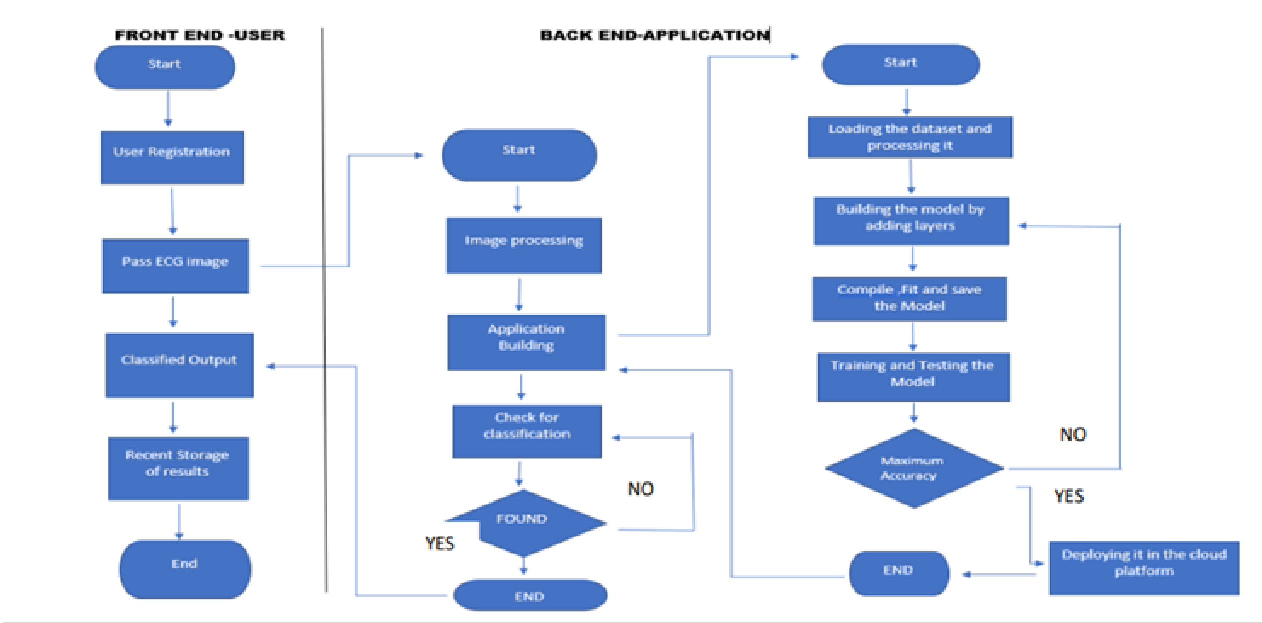
Non-requirement:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• The application should be supported by all Operating system . • It should be able to access even through Mobile phone • It should be user friendly to the customers
NFR-2	**Security**	The Application should be very much secure . There should not be any vulnerabilities or Application layer security problems such as Hacking etc . so to ensure security cryptographic techniques for username and password could be used
NFR-3	**Reliability**	The application should be reliable by means of • Accuracy • Easy to use • Flexibility • Can be accessed multiple number of times
NFR-4	**Performance**	The performance of the application should be enhanced by • using advanced API's • Effective memory utilisation • Easy accessibility of previous records improvising • Accuracy of the system Thereby resulting in the effectiveness of the system
NFR-5	**Availability**	The application is compatible for both mobile and desktop users and should be available to the user 24/7 and can be accessed anytime
NFR-6	**Scalability**	The application must be scalable enough to support more than 20,000 users at the same time ,while maintaining its efficiency and optimal performance

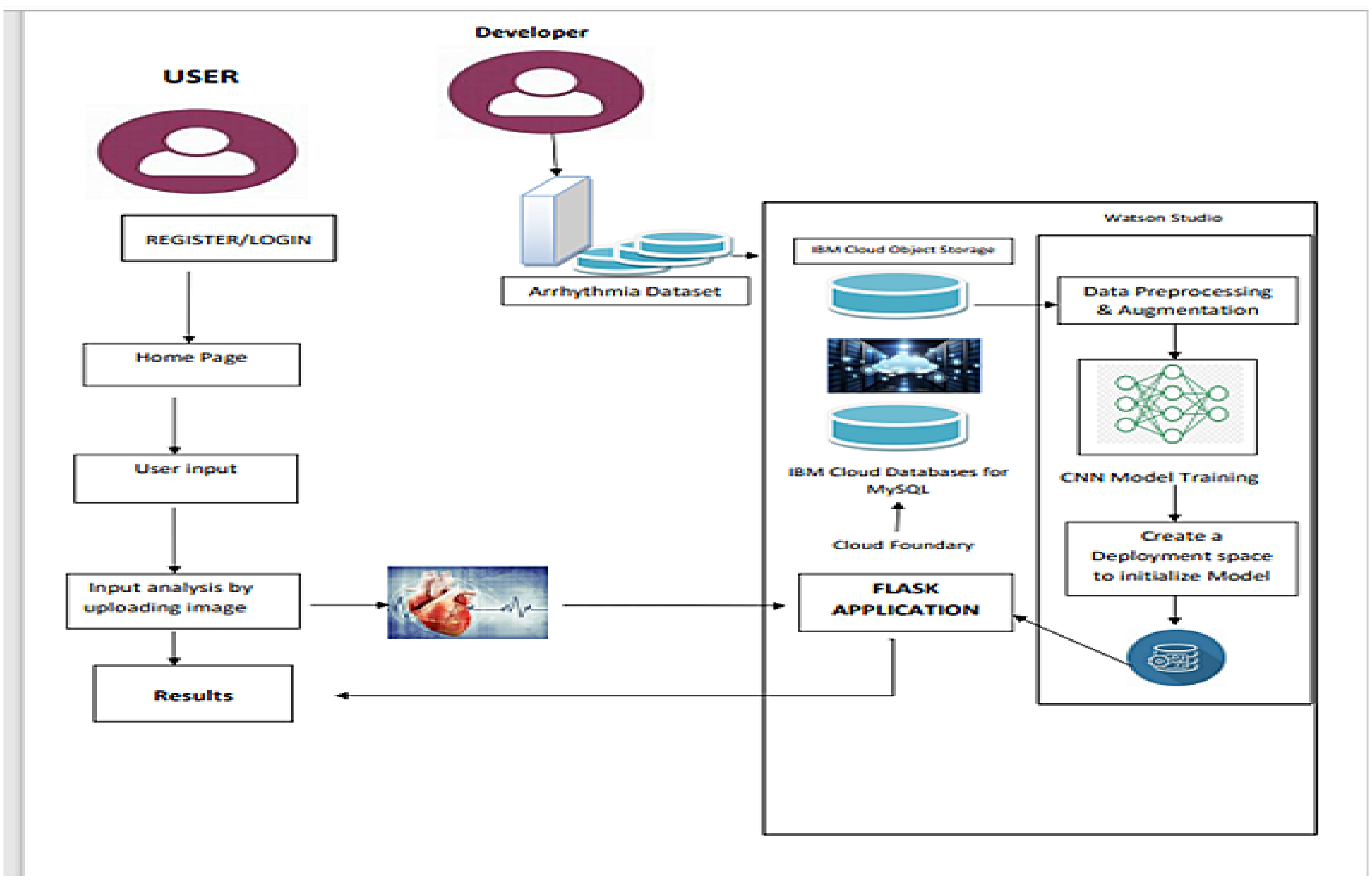
5.PROJECT DESIGN:

Dataflow Diagram:

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.



Solution & Technical Architecture:



User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard		Jupyter notebook or google colab,in general python related environment			
Customer (Web user)	Login		As a User accessing through web, I can see some backend operations ,get some notifications, and also my login details will be stored in cache as well. For frequent users login can be done by entering email and password once			
Customer Care Executive	Login					
Administrator	Login		Ensuring accuracy in the Applications and maintaining confidentiality about the customer details as well			

6.PROJECT PLANNING & SCHEDULING:

Sprint Planning & Estimation:

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect the Dataset from various sources	9	High	Kalaiselvi S
Sprint-1		USN-2	Process the images in the dataset for better accuracy [Image processing and Augmentation]	8	Medium	Kalaiselvi S
Sprint-2	Model Building	USN-3	Import the necessary libraries and then initialize the model	8	Medium	Kanishya A
Sprint-2		USN-3	Necessary libraries are added and then the model is trained using CNN	10	High	Kanishya A
Sprint-3	Training and Testing	USN-4	The model is trained and tested for accuracy and then accurate model is saved	9	High	Bhuvana
Sprint-4	Application Building	USN-5	The HTML files are created ,Python codes are built and then finally the app is made to run	8	Medium	Kiruthika V

Sprint Delivery Scheduling:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	7	12 Nov 2022
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022	7	12 Nov 2022
Sprint-3	10	6 Days	07 Nov 2022	12 Nov 2022	10	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

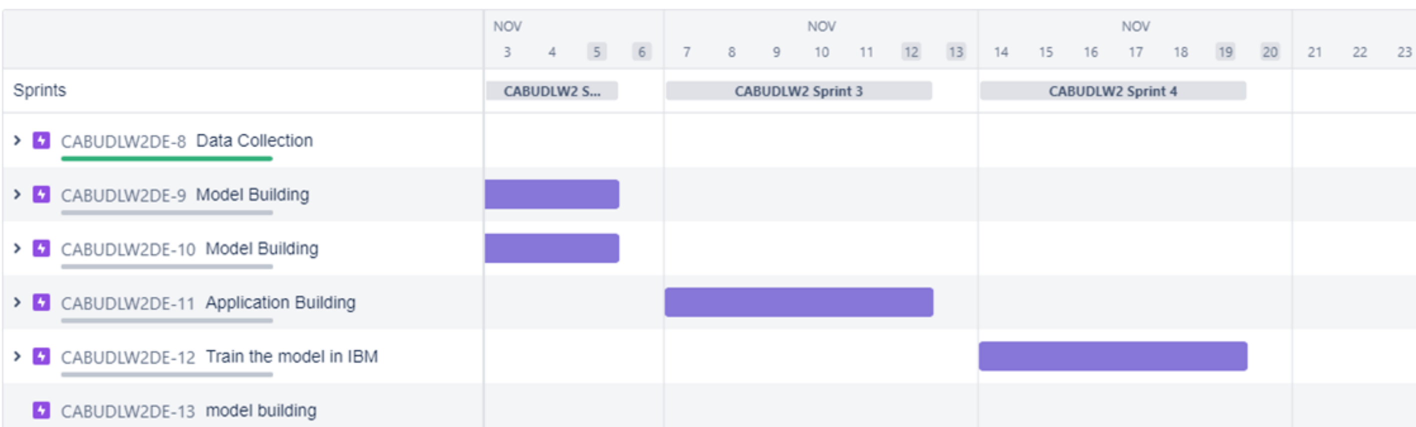
Average Velocity:

AV = Velocity/Sprint Duration

=10/6

=1.666

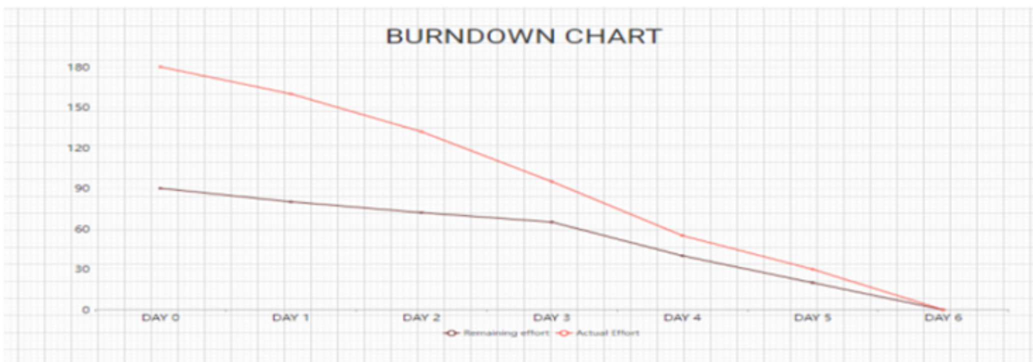
Reports from JIRA:



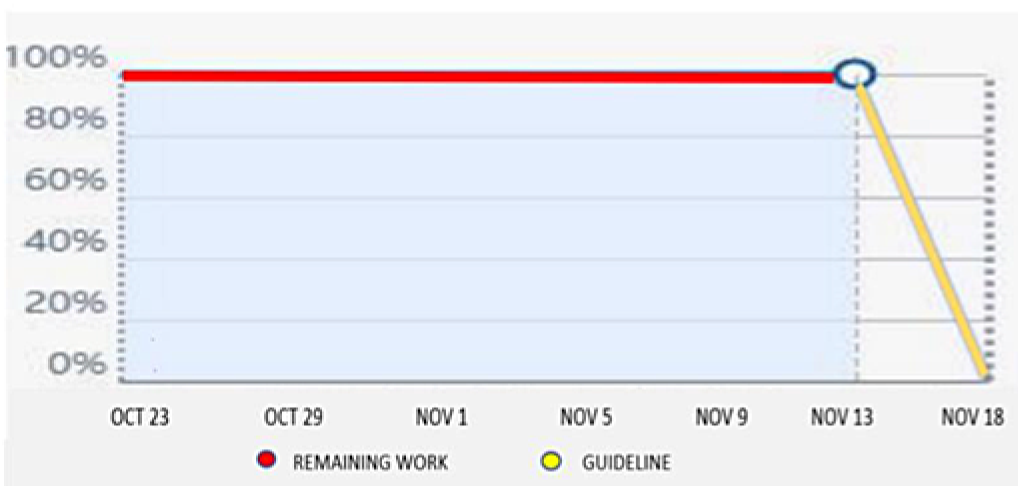
BURNDOWN GRAPH:

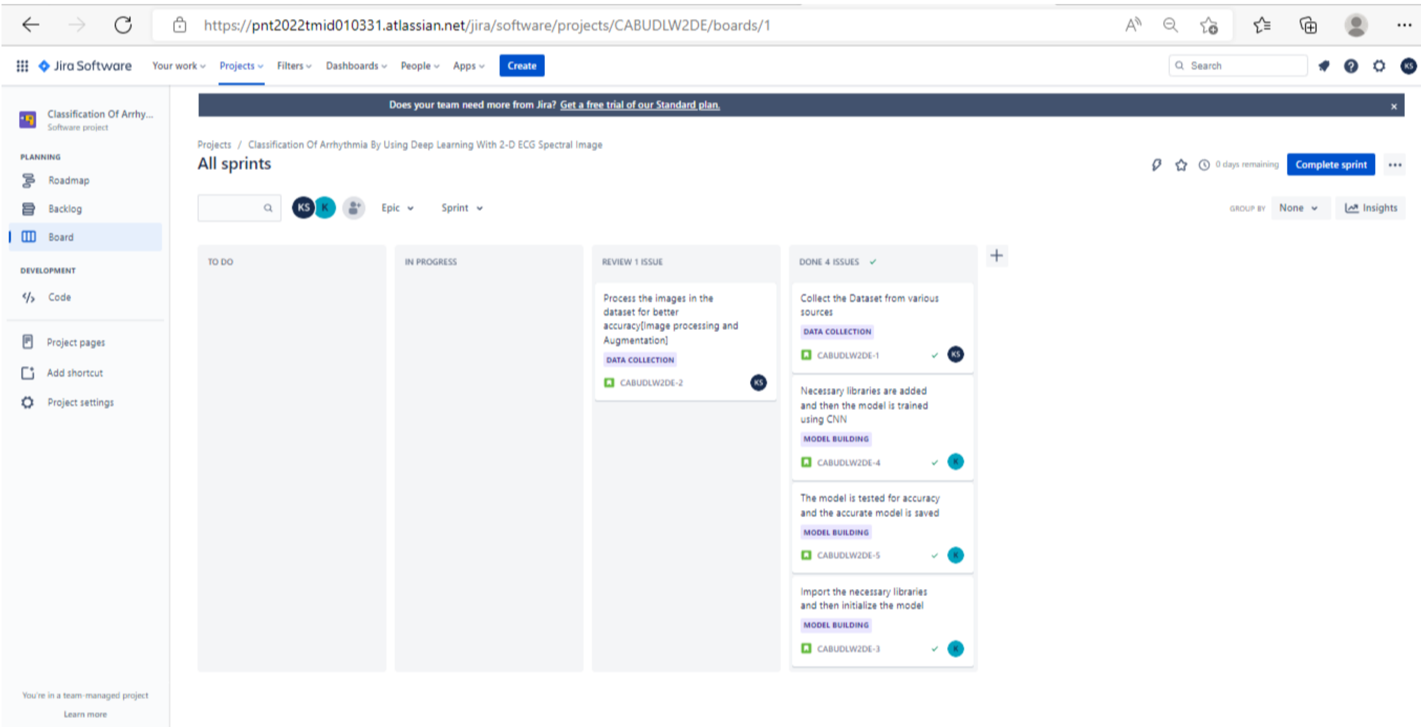
A Burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

Burndown Chart:



SPRINT BURNDOWN CHART



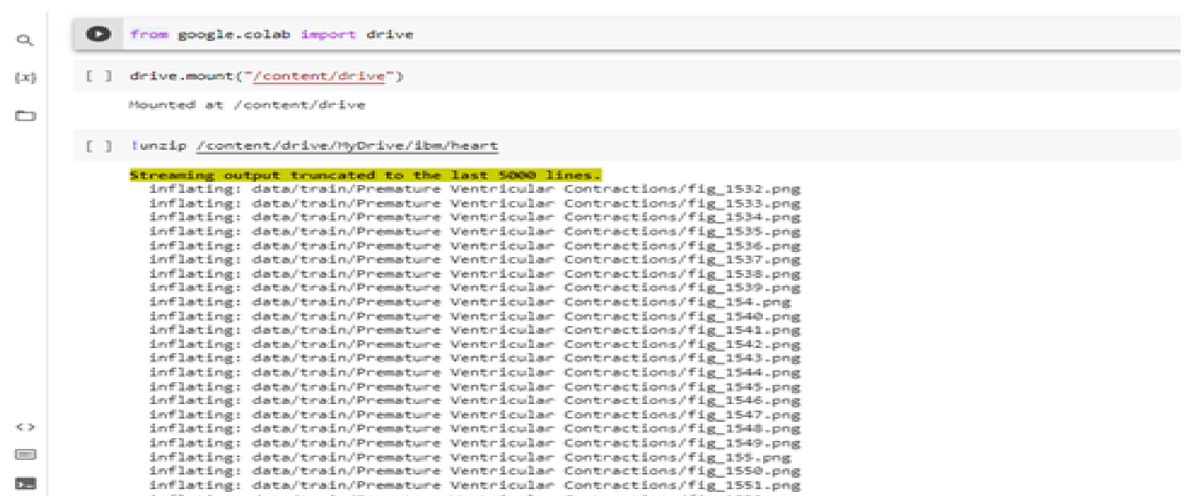


7.CODING &SOLUTIONING

(Explain the features added in the project with code)

Feature 1 Data loading

```
from google.colab import drive
drive.mount("/content/drive")
////Mounted at /content/drive
!unzip /content/drive/MyDrive/ibm/heart
from  tensorflow.keras.preprocessing.image import ImageDataGenerator
```



Feature 2 image processing

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

In [ ]:

```
#image_augmentation
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
x_train=train_ds.flow_from_directory(r'/content/data/train',
                                     target_size=(192,128),
                                     class_mode='categorical',
                                     batch_size=32)
Found 15341 images belonging to 6 classes.
```

In [ ]:

```
x_test=test_ds.flow_from_directory(r'/content/data/test',
                                   target_size=(192,128),
                                   class_mode='categorical',
                                   batch_size=32)
Found 6825 images belonging to 6 classes.
```

In [ ]:

In [ ]:

```
x_train.class_indices
```

Out[ ]:

```
{'Left Bundle Branch Block': 0,
 'Normal': 1,
```

```
'Premature Atrial Contraction': 2,
'Premature Ventricular Contractions': 3,
'Right Bundle Branch Block': 4,
'Ventricular Fibrillation': 5}
```

8.TESTING :

Test cases:

```
from tensorflow.keras.preprocessing import image

model=load_model("ECG.h5")
```

Test case 1

```
#test_case_1
img=image.load_img('fig_35.png',target_size=(192,128))

import numpy as np
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)

x.shape

(1, 192, 128, 3)

y=np.argmax(model.predict(x))

1/1 [==============================] - 0s 25ms/step

y

4

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

index[y]

'Right Bundle Branch Block'
```

Test case 2

```
#test_case_2

img=image.load_img('fig_5749.png',target_size=(192,128))

import numpy as np
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)

y=np.argmax(model.predict(x))

1/1 [==============================] - 0s 23ms/step

index[y]

'Premature Ventricular Contractions'
```

test case 3

```
#test_case_3

img=image.load_img('fig_3497.png',target_size=(192,128))
import numpy as np
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x))

y
index[y]
```

✓ 3s completed at 7:39 PM

9.RESULTS:

Performance Merics

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/tensorflow/python/keras/engine/training.py:1940: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/10
480/480 [==============================] - 466s 968ms/step - loss: 1.2137 - accuracy: 0.5979 - val_loss: 0.8051 - val_accuracy: 0.6967
Epoch 2/10
480/480 [==============================] - 463s 965ms/step - loss: 0.4495 - accuracy: 0.8578 - val_loss: 0.6057 - val_accuracy: 0.7962
Epoch 3/10
480/480 [==============================] - 462s 962ms/step - loss: 0.3797 - accuracy: 0.8824 - val_loss: 0.6401 - val_accuracy: 0.8166
Epoch 4/10
480/480 [==============================] - 462s 963ms/step - loss: 0.3556 - accuracy: 0.8902 - val_loss: 0.6136 - val_accuracy: 0.8252
Epoch 5/10
480/480 [==============================] - 459s 955ms/step - loss: 0.3442 - accuracy: 0.8937 - val_loss: 0.6836 - val_accuracy: 0.8019
Epoch 6/10
480/480 [==============================] - 461s 960ms/step - loss: 0.3178 - accuracy: 0.9053 - val_loss: 0.7337 - val_accuracy: 0.7704
Epoch 7/10
480/480 [==============================] - 458s 955ms/step - loss: 0.3188 - accuracy: 0.9010 - val_loss: 0.6205 - val_accuracy: 0.8421
Epoch 8/10
480/480 [==============================] - 459s 956ms/step - loss: 0.2825 - accuracy: 0.9171 - val_loss: 0.6164 - val_accuracy: 0.8344
Epoch 9/10
480/480 [==============================] - 457s 953ms/step - loss: 0.2486 - accuracy: 0.9271 - val_loss: 0.5976 - val_accuracy: 0.8299
Epoch 10/10
480/480 [==============================] - 455s 948ms/step - loss: 0.2375 - accuracy: 0.9327 - val_loss: 0.5734 - val_accuracy: 0.8533
Out[25]: <tensorflow.python.keras.callbacks.History at 0x7f956ae6fd60>

In [26]: model.save('ECG.h5')

In [27]: !tar -zcvf image-Classification-model_new.tgz ECG.h5

         ECG.h5

In [28]: ls -1

         data/
         ECG.h5
         image-Classification-model_new.tgz
```

```
In [6]: model=load_model("ECG.h5")

In [15]: #test_case_1
         img=image.load_img('fig_35.png',target_size=(192,128))

In [16]: import numpy as np
         x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)

In [17]: x.shape

Out[17]: (1, 192, 128, 3)

In [18]: y=np.argmax(model.predict(x))

         1/1 [==============================] - 0s 25ms/step

In [19]: y

Out[19]: 4

In [20]: index=['Left Bundle Branch Block','Normal','Premature Atrial Contractions','Premature Ventricular Contractions','Right Bundle Bra

In [21]: index[y]

Out[21]: 'Right Bundle Branch Block'

In [23]: #test_case_2

In [27]: img=image.load_img('fig_5749.png',target_size=(192,128))
```

```
In [23]: #test_case_2

In [27]: img=image.load_img('fig_5749.png',target_size=(192,128))

In [28]: import numpy as np
         x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)

In [29]: y=np.argmax(model.predict(x))

         1/1 [==============================] - 0s 23ms/step

In [30]: index[y]

Out[30]: 'Premature Ventricular Contractions'

In [31]: #test_case_3

In [33]: img=image.load_img('fig_3497.png',target_size=(192,128))
         import numpy as np
         x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)
         y=np.argmax(model.predict(x))

         y
         index[y]

         1/1 [==============================] - 0s 27ms/step

Out[33]: 'Normal'
```

10. ADVANTAGES &DISADVANTAGES:

Advantage:

Flexibility in identifying the hearts rhythmic irregularities.
Regular check for the diagnosis becomes easier.
Instant and accurate results.

Disadvantages:

Analysing the scan results manually by doctors consumes long time
Current flow droppage makes difficult at emergency situation.
Waiting for specialists

11.FUTURE SCOPE:

Advanced technologies such as AI and wearables were seen as the future of ECG, and rightly so, given ongoing developments in regulatory movement, public enthusiasm, and R&D innovation. Further, the heart complications surrounding COVID-19 continue to validate ECG's application and value in diagnostic decision-making.

During isolation, can be integrated with other apps to produce full fledged app. In near future everything will be digitalized so this bevery much useful.

12. APPENDIX:

Source code

```
pwd
!pip install keras==2.2.4
!pip install tensorflow==2.5.0
import os, types
```

```
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes
your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='sZmW7ChAxF_z7fqdh9QjWZaoANyi2onbO3YJsULM0GGe',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

bucket = 'classificationofecg-donotdelete-pr-pvvx2hiz4wniw3'
object_key = 'Classification of Arrhythmia by Using Deep Learning with 2-D ECG
Spectral Image Representation.zip'

streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the
possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_2.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
from  tensorflow.keras.preprocessing.image import ImageDataGenerator
#image_augmentation
train_ds=ImageDataGenerator(rescale=1./255,
                            shear_range=0.2,
                            zoom_range=0.2,
                            horizontal_flip=True,

                            vertical_flip=True)
test_ds=ImageDataGenerator(rescale=1./255)
x_train=train_ds.flow_from_directory(r'data/train',
                                     target_size=(192,128),
                                     class_mode='categorical',
                                     batch_size=32)
#Found 15341 images belonging to 6 classes.
```

```
x_train.class_indices
x_test=test_ds.flow_from_directory(r'data/test',
                                   target_size=(192,128),
                                   class_mode='categorical',
                                   batch_size=32)
x_train.class_indices
#sprint-2
#create model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
model=Sequential()
#add layers
model.add(Convolution2D(32,(3,3),input_shape=(192,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Convolution2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(32))
model.add(Dense(6,activation='softmax'))
model.summary()
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy
'])
model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,valid
ation_data=x_test,validation_steps=len(x_test))
model.save('ECG.h5')
!tar -zcvf image-Classification-model_new.tgz ECG.h5
ls -1
#testing the model
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model=load_model("ECG.h5")
img1=image.load_img(r'data/test/Premature Ventricular Contractions/VEBfig_13.png')
img1
img1=img1.resize((128,192))
x=image.img_to_array(img1)
x
import numpy as np
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x))
y
index=['Left Bundle Branch Block','Normal','Premature Atrial
Contractions','Premature Ventricular Contractions','Right Bundle Branch
Block','Ventricular Fibrillation']
index[y]
client.repository.download(model_id,'my_model.tar.gz')
import tensorflow as tf
tf.__version__
!pip install keras==2.2.4
```

```
#deployment
!pip install watson-machine-learning--Client
from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"jODT-AnyGz3AWuG_kZdrQUOBNM5whihNrQnnLZ-h1x3U"
}
client=APIClient(wml_credentials)
client
def guid_space_name(client,img_class):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if
item['entity']['name']==ecg_deploy)['metadata']['id'])
space_uid=guid_space_name(client,'ecg_deploy')
print("Space UID"+space_uid)
client.set.default_space(space_uid)
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_1.15-
py3.6')
software_space_uid
model_details=client.repository.store_model(model='ECG.h5',meta_props={
    client.repository.ModelMetaNames.NAME:"CNN",
    client.repository.ModelMetaNames.TYPE:'KERAS_2.2.4',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
})
model_id=client.repository.get_model_uid(model_details)
model_id
client.repository.download(model_id,'my_model.tar.gz')
client.repository.download(model_id,'fruit-training.ter.gz')
```

GitHub link:

https://github.com/IBM-EPBL/IBM-Project-3119-1658502518

PROJECT DEMO LINK:

https://drive.google.com/file/d/15wI1uM1iglDnTelofWxlN1l
mw49v8GCf/view?usp=sharing