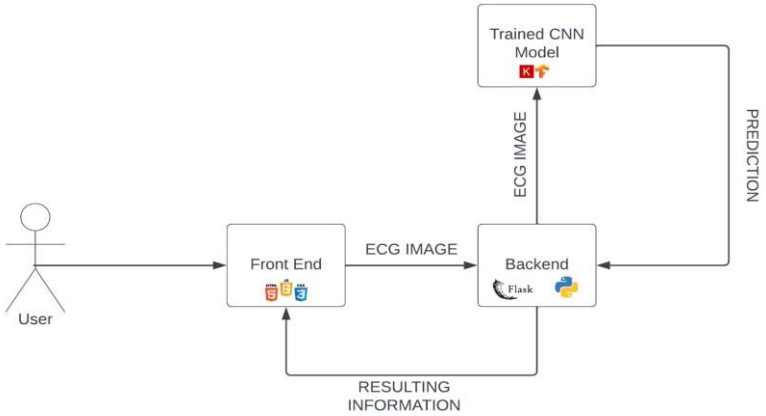
Technical Architecture:

Date	22 October 2022
Team ID	PNT2022TMID45689
Project Name	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



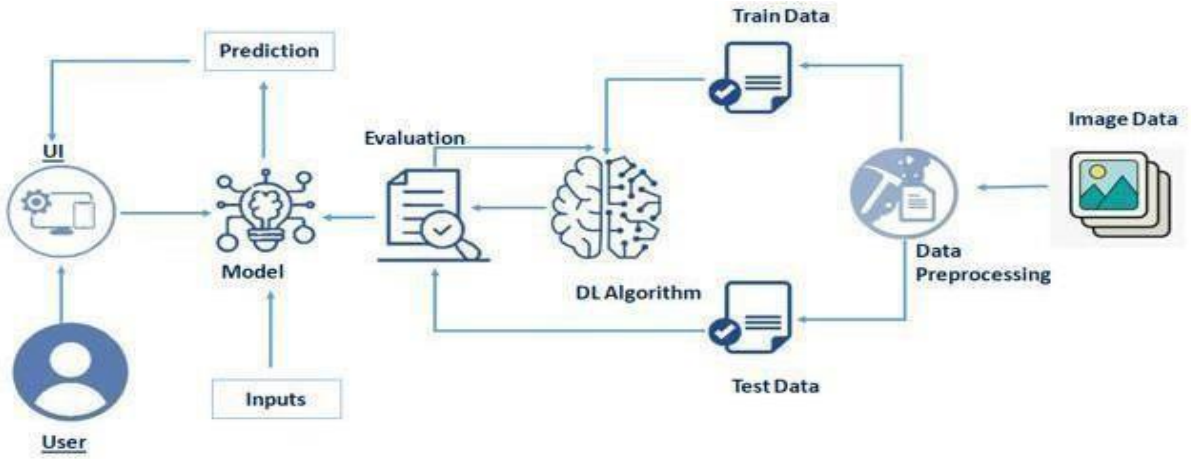


Table-1: Components & Technologies:

S No.	Component	Description	Technology
1.	User Interface	How user interacts with User interface to upload image	Anaconda, Jupyter notebooks, Spyder, Python.
2.	Model analyses	Once model analyses the uploaded image, the prediction is showcased on the UI	Python, OpenCV
3.	Data collection	Create the dataset	Kaggle.com, data.gov, UCI

4.	Data Preprocessing-1	Import the ImageDataGenerator library	Python, Keras, Numpy
5.	Data Preprocessing-2	Configure ImageDataGenerator class	Python, Numpy, Keras
6.	Data Preprocessing-3	Apply ImageDataGenerator functionality to Train set and Test set.	Python, Numpy, Keras
7.	Model Building-1	Import the model building libraries and Initializing The model	Python, Numpy, Keras, Tensorflow
8.	Model Building-2	Adding layers and configure.	Python, Numpy, Keras, Tensorflow
9.	Model Building-3	Training and testing the model, Optimize and save the model.	Python, Numpy, Keras, Tensorflow
10.	Application Building	Purpose of create an HTML file and Building Python code.	HTML, Python, CSS, JS
11.	Train the model on IBM	CNN Development and integrate it with the flask Application.	IBM Watson
12.	Deployment	Deploy the application and make it available for the public to use.	IBM Cloud

Table-2: Application Characteristics:

S No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-source software is that by which the source code or the base code is usually available for modification or enhancement.	Flask (Python)
2.	Security Implementations	By placing a filtration barrier between the targeted server and the attacker using authentication.	E.g. SHA-256, Encryptions, HTTPS etc.
3.	Scalable Architecture	Does not affect the performance even though used by many users.	IBM Cloud

4.	Availability	Justify the availability of application (E.g. use of load balancers, distributed servers etc.)	IBM Cloud
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc.	IBM Cloud, Flask, Numpy