Cairo University
Faculty of Computers and Artificial intelligenceDepartment of Computer Sciences

Detecting Covid-19 with ECG and deep learning

Supervised by

Dr. Ahmed Farouk
Dr. Eman Hossny
Eng. Hassan Murad

Implemented by

20198036	*Rofida Gamal Muhammed*
20198119	*Muhammed Mosaad Abdeltawab*
20198039	*Rewan Hisham Sultan*
20198028	*Dalia Hosny Ahmed*

Graduation Project
Academic Year 2022-2023
Documentation

Table of content:

List of Figures:

List of tables:

List of abbreviation:

- **ECG**: Electrocardiogram.
- **CNN**: Convolutional Neural Networks.
- **ResNet:** Residual Network
- **VGG:** Visual Geometry Group
- **ML**: Machine Learning.
- **DL**: Deep Learning.
- **MI:** Myocardial Infarction.
- **AHB:** Abnormal Heart Beats.
- **RMI:** Recovered Myocardial infarction.
- **HOG:** Histogram Of Oriented Gradients.
- **HTML:** Hyper Text Markup Language.
- **CSS:** Cascading Style Sheet.
- **ERD**: entity relationship diagram.
- **GUI:** Graphical user interface.
- **MVC:** Model–View–Controller architectural pattern.
- **PHP:** is a general-purpose scripting language geared towards web development.

Chapter one: Introduction

Abstract:

The COVID-19 pandemic has witnessed the rapid global transmission of Coronavirus Disease 2019 (COVID-19), leading to various health complications, including cardiovascular issues among affected patients. Consequently, a key inquiry arises: how can COVID-19 and heart diseases be detected through the analysis of electrocardiogram (ECG) trace images? Recent scientific endeavors have proposed diverse methodologies and techniques to identify the SARS-CoV-2 virus by employing distinct images and data. Nonetheless, this study represents the first attempt to explore the potential utilization of deep convolutional neural network (CNN) models for the purpose of detecting COVID-19 from ECG trace images. Within this document, deep learning techniques were employed to identify COVID-19 as well as other cardiovascular diseases (CVDs). A publicly accessible dataset of 1937 ECG images, classified into five distinct categories, namely Normal, COVID-19, myocardial infarction (MI), abnormal heartbeat (AHB), and recovered myocardial infarction (RMI), was utilized. we used 7 pre-trained models: EfficientNetB0, EfficientNetB3, DenseNet201, InciptionNetV3, MobileNetV2, ResNet50 and VGG19 with Five-fold cross validation and therefore, 80% of data were used for training and 20 % for testing. Out of the training dataset subset, 10% were utilized for validation to avoiding overfitting issues. Finally, the results were a weighted average of five folds, but the best model was EfficientNetB3 with accuracy 98.065%.

Key words: Electrocardiogram (ECG); COVID-19; Deep learning; Convolutional neural networks, cardiovascular diseases (CVDs).

Background:

introduction:

The emergence of the novel coronavirus in December 2019, originating in Wuhan, China, marked the onset of a global pandemic. Referred to as COVID-19 (Coronavirus Disease-2019), this illness caused by the SARS-CoV-2 virus rapidly spread across all corners of the globe. COVID-19 primarily affects the lungs and has the potential to be fatal. It has been observed that this new strain of coronavirus exhibits heightened transmissibility, posing greater challenges in terms of containment and carrying a higher pandemic risk.

While most individuals infected with COVID-19 experience mild symptoms and recover, vulnerable populations with pre-existing medical conditions such as diabetes, cancer, chronic respiratory failure, and heart failure face a significantly higher risk of severe illness and mortality. Consequently, there is a pressing need to devise novel strategies for the detection of the SARS-CoV-2 virus.

The field of biomedical applications has seen remarkable advancements in artificial intelligence, facilitating the development of trained networks capable of aiding in reliable computer-aided diagnostic decisions. This progress has alleviated the burden on healthcare facilities, including medical professionals and support staff. Recent studies have proposed various deep learning models to identify abnormalities in medical images, encompassing chest X-ray images and computerized tomography (**CT**) scans.

*1.1.1 What is meant by Machine Learning (**ML**)?*

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

*1.1.2 What is meant by Deep Learning (**Dl**)?*

Deep learning is a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.

*1.1.3 What is meant by Electrocardiogram (**ECG**)?*

An electrocardiogram (ECG) is a simple test that can be used to check your heart's rhythm and electrical activity. An ECG is often used alongside other tests to help diagnose and monitor conditions affecting the heart.

1.1.4 What is meant by Lazy Classifier?

It's an automated machine learning technique, it builds a lot of basic models "e.g., SVC, Logistic Regression, Random Forest Classifier, LGBM Classifier, etc.." without much code and helps understand which models work better without any parameter tuning and finally show testing accuracy to each one of them in form of numeric values or chart.

*1.1.5 What is meant by Convolutional Neural Networks (**CNNs**)?*

A convolutional neural network (CNN or convnet) [1] is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data.

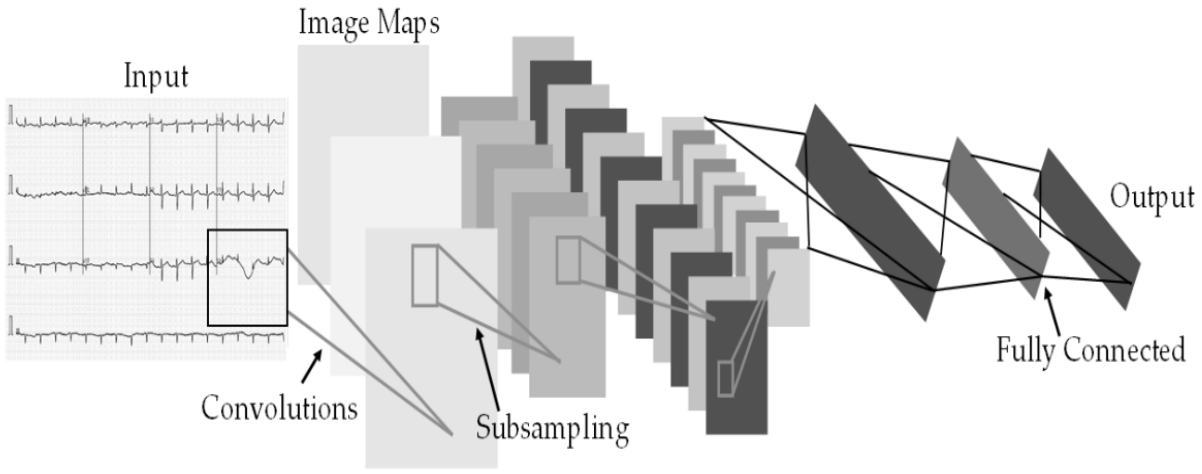


Figure 1: Architecture of a Convolutional Neural Network.

Figure 1 Architecture of CNN

1.1.6 What is meant by preprocessing data?

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

1.1.7 What are pre-trained models?

CNN models, it networks have also a great version that is based on transfer learning and pre-trained models. Transfer learning is a popular method in computer vision because it allows us to build accurate models in a timesaving way (Rawat & Wang 2017). With transfer learning, instead of starting the learning process from scratch, you start from patterns that have been learned when solving a different problem.

A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem like the one that we want to solve. Accordingly, due to the computational cost of training such models, it is common practice to import and use models from published literature (e.g., VGG, Inception, Efficient-Net, Res-Net).

1.1.8 What are Efficient-Net models ?

[2] They are a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/ resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the Efficient-Net scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.

Efficient-Net is a better version of CNN pre-trained models where:

- Reduce computing costs.
- Reduce battery usage.
- Reduce training and inference speed.
- Enable use of deep learning on mobile apps and other edge devices.

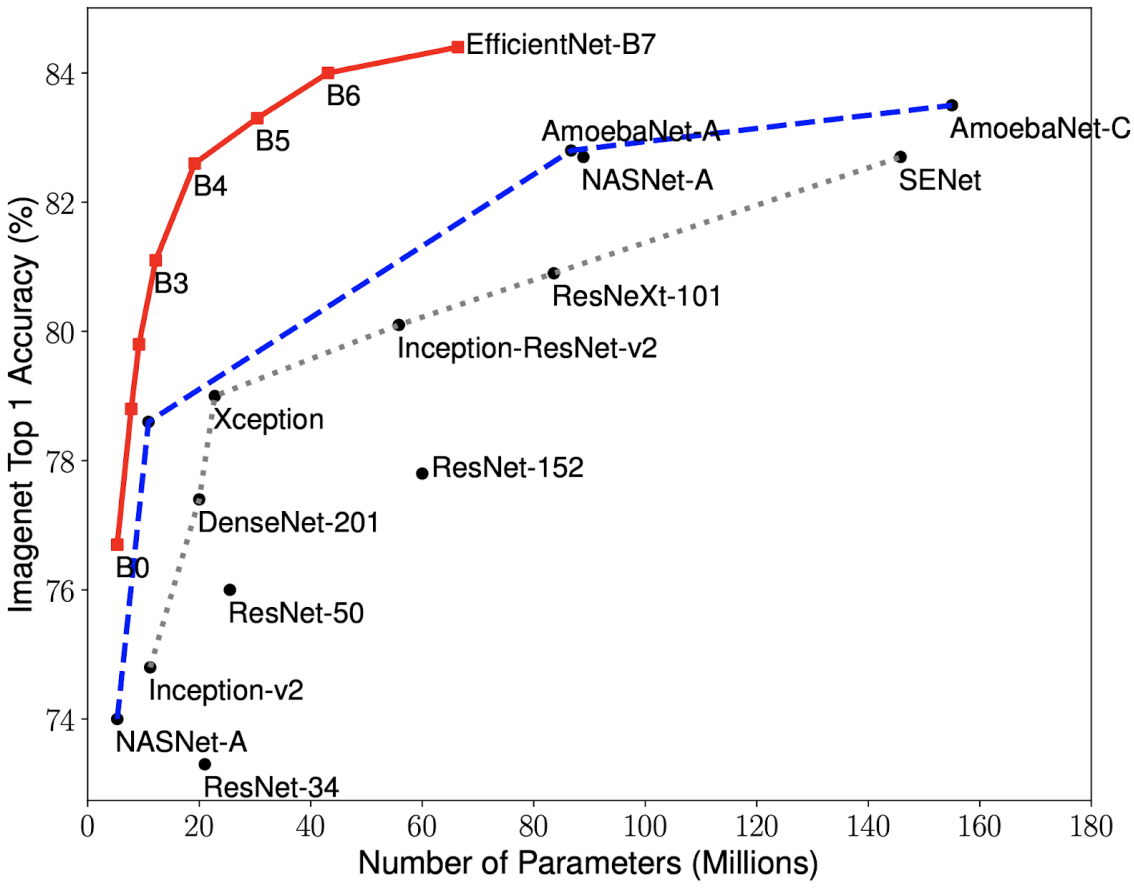


Figure2: Efficient Models VS CNN models Performance

Motivation:

We have made a great effort during the project and in the following lines we will give you a brief description of what we have done. Our project's aim is to detect COVID-19 using ECG trace images by using deep convolutional neural network models.

Our data is classified into 5 main Classes, which are:

- ECG images of COVID-19 patients.
- ECG images of Myocardial Infarction (Heart Attack) patients.
- ECG images of patients that have abnormal heartbeats.
- ECG images of patients that have history of MI.
- Normal person ECG images.

COVID-19 can cause the heart rate to become faster or irregular in response to fever or inflammation, as the heart works faster to pump more blood around the body to fight the infection, it also causes blood clots and heart damage due to lack of oxygen.

That's why we used Electrocardiogram (ECG) because it's one of the simplest and fastest tests used to evaluate the heart where, electrodes (small, plastic patches that stick to the skin) are placed at certain spots on the chest, arms, and legs. The electrodes are connected to an ECG machine by lead wires. The electrical activity of the heart is then measured, interpreted, and printed out. No electricity is sent into the body.

The study was carried using three different deep CNN models (ResNet50, MobileNetv2, inception v3), were used to investigate classification scheme five-class classification (Normal, COVID-19, MI, AHB, and RMI), we are going to use Efficient-Net models (B0 and B3) and VGG19.

Problem Definition:

Currently, most detection of Covid-19 is based on X-Ray for chest and Neural Network Algorithms. This will be the one of the first studies that will explore the possibility of using deep convolutional neural network (CNN) and Efficient-Net models to detect COVID-19 and other cardiac abnormality detection from electrocardiogram (ECG) trace images. The approach of deep learning on ECG trace images has been explored with promising results. And with this it will be easy to take an image from your phone and see if you have Covid-19 or not and can be used by doctors also for rapid results.

Proposed Solution:

- working with ECG (Electrocardiogram) test which is affordable and available.
- Detecting if the patient has Heart problems / Covid-19 or not, so by one test we can identify two serious problems.
- Using machine learning (shallow learning and CNN models)

Gantt chart of project time plan:

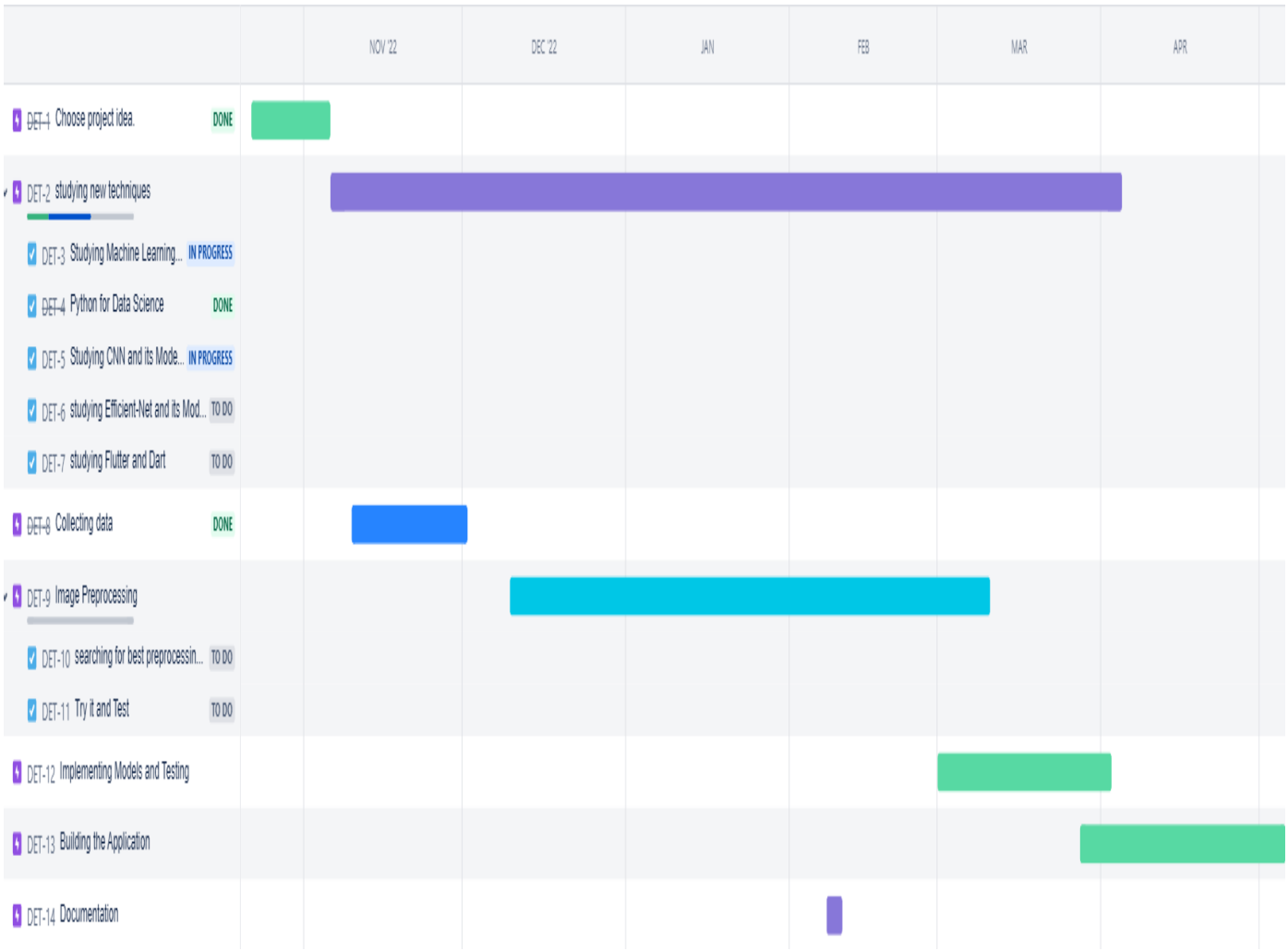


Figure 3 The project time plan

Project development methodology:

In our project, we adopted a waterfall approach while developing the machine learning model, website, and mobile application. This methodology involved a sequential progression through distinct phases, with each phase relying on the completion of the previous one. We carefully planned and documented requirements, design, development, testing, and deployment stages. This structured approach enabled us to focus on one aspect at a time, ensuring thorough completion before moving on to the next. By adhering to a predetermined schedule and predefined milestones, we maintained clear project goals and minimized the need for significant revisions. Although the waterfall approach is characterized by less flexibility compared to agile methodologies, it provided a systematic and organized framework to deliver a robust and fully functional machine learning model, website, and mobile application.

Models:

Our project will help in faster computer-aided diagnosis of COVID-19 and other cardiac abnormalities from ECG images using Machine learning and deep learning techniques. An ECG image is a graphical representation of the electrical activity of the heart over time, as detected by electrodes placed on the body's surface, ECG dataset[ref] consists of images for five distinct categories, such as Normal, COVID-19, myocardial infarction (MI), abnormal heartbeat (AHB), and recovered myocardial infarction (RMI).

The next figures show a sample for each class:

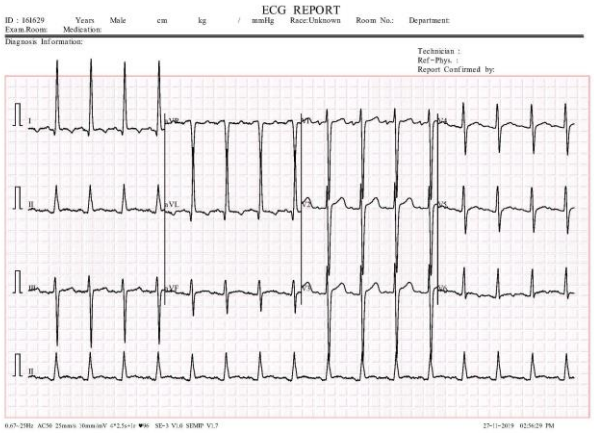


Figure 4 Abnormal heartbeat ECG image.

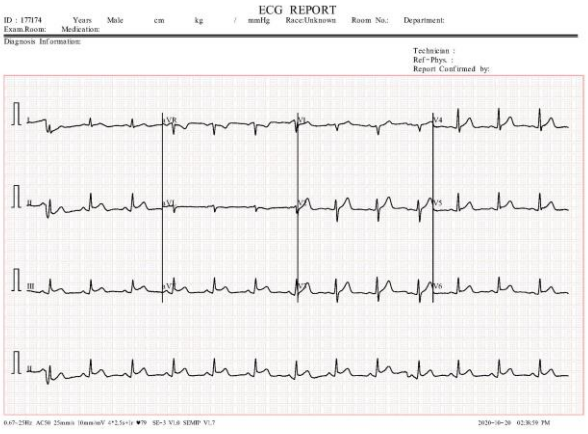


Figure 5 Myocardial Infarction ECG image.

Figure 6 Patient that have History of Myocardial Infarction ECG image.

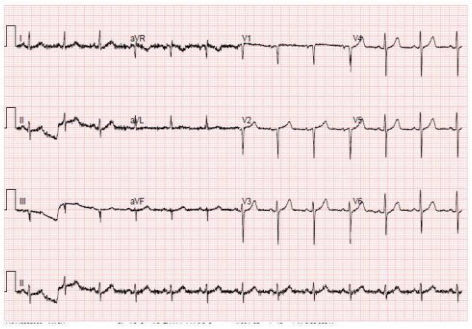


Figure 7 COVID-19 ECG image.

Figure 8 Normal Person ECG image.

1.5.1 Pre-processing and Machine learning techniques:

Firstly: in preprocessing we used the Histogram of Oriented Gradients (HOG) [ref] is a popular pre-processing technique used in computer vision and image processing applications, particularly for object detection and recognition tasks. The basic idea behind HOG is to extract and quantify the gradient orientation information from an image.

Here are the steps involved in applying the HOG pre-processing technique for ECG images:

1. Image Preprocessing: The first step is to apply some pre-processing techniques to the input image to prepare it for the HOG feature extraction. This preprocessing: The ECG signal is first preprocessed to remove baseline wander and noise. This can be done using filtering techniques such as wavelet denoising or median filtering.
2. Image Conversion: The preprocessed ECG signal is then converted into a grayscale image by plotting the signal amplitude over time. The x-axis represents time in seconds, and the y-axis represents the amplitude in millivolts.
3. Cell Formation: The image is divided into small cells, and the gradient information within each cell is accumulated into a histogram of gradient orientations.

4. Block Normalization: The histogram values are then normalized across a group of cells, called a block. This normalization step helps to reduce the effect of illumination variations and contrast changes in the ECG signal.
5. HOG Feature Vector: The normalized histogram values from all the blocks are concatenated to form a HOG feature vector that represents the ECG signal.

The resulting HOG feature vector can be used as input to various machine learning algorithms for object detection and recognition tasks. The HOG technique is widely used in applications such as face detection, pedestrian detection, and vehicle detection in images and videos. By using HOG pre-processing, we can extract and quantify important features from the image that can be used to train machine learning models for improved accuracy and performance. In our project after extracting important features from image and save the result as List of pixels then convert it to data frame and save this data frame as CSV file ("type of excel sheets files") to use in modeling process.

This figure shows the result after applying HOG technique:

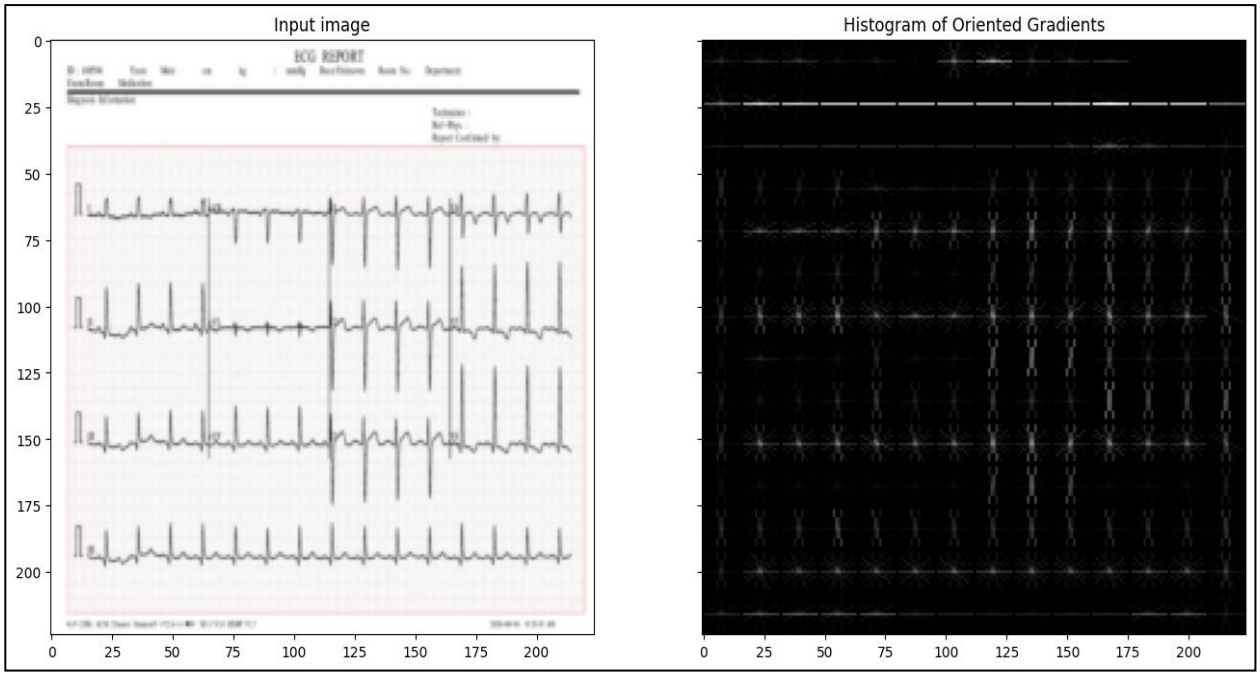


Figure 9 HOG technique on ECG of patient that have abnormal heartbeats.

Secondly: in the modeling step we use Lazy classifier, it's one of the best python libraries that helps you to semi-automate your Machine Learning Task. It builds a lot of basic models without much code and helps understand which models work better without any parameter tuning.

Here's a brief overview of how a lazy classifier works:

1-Data Preprocessing: in this step we use HOG technique for as we explain before.

2-Data Splitting: The dataset is already splitted into training and testing Sets.

3-Classifier Initialization: A lazy classifier is initialized with the training data.

Examples of lazy classifiers include k-Nearest Neighbors (k-NN), Local Outlier Factor (LOF), and Local Correlation Integral (LCI).

4-Testing: The lazy classifier is then used to classify the testing data. For each new instance in the testing set, the lazy classifier searches the training set for the k-nearest neighbors (in the case of k-NN) or the nearest neighbors (in the case of LOF and LCI) and assigns the class label based on the majority vote of the neighbors.

5- Performance Evaluation: The performance of the lazy classifier is then evaluated using various metrics such as accuracy, precision, recall, and F1-score. The confusion matrix is also used to evaluate the performance of the classifier.

This figure shows the result of lazy classifier after applying it on our dataset:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LGBMClassifier	0.96	0.94	None	0.96	1339.63
ExtraTreesClassifier	0.96	0.94	None	0.96	11.54
XGBClassifier	0.96	0.94	None	0.96	382.12
RandomForestClassifier	0.96	0.94	None	0.96	22.77
LogisticRegression	0.96	0.94	None	0.96	39.29
RidgeClassifierCV	0.96	0.94	None	0.96	14.47
RidgeClassifier	0.96	0.94	None	0.96	13.50
QuadraticDiscriminantAnalysis	0.93	0.94	None	0.94	58.19
LabelPropagation	0.93	0.94	None	0.94	11.56
LabelSpreading	0.93	0.94	None	0.94	11.76
SGDClassifier	0.96	0.94	None	0.96	35.55
BaggingClassifier	0.95	0.94	None	0.95	287.38
LinearSVC	0.96	0.94	None	0.96	884.44
PassiveAggressiveClassifier	0.95	0.94	None	0.95	58.07
LinearDiscriminantAnalysis	0.95	0.93	None	0.95	96.79
NuSVC	0.95	0.93	None	0.95	437.28
SVC	0.95	0.93	None	0.95	286.77
DecisionTreeClassifier	0.94	0.93	None	0.94	47.99
Perceptron	0.95	0.93	None	0.95	28.93
CalibratedClassifierCV	0.95	0.93	None	0.95	1811.40
ExtraTreeClassifier	0.94	0.92	None	0.93	6.39
KNeighborsClassifier	0.86	0.85	None	0.86	7.07
NearestCentroid	0.69	0.69	None	0.68	10.82
AdaBoostClassifier	0.67	0.68	None	0.68	370.24
BernoulliNB	0.65	0.66	None	0.64	11.03
GaussianNB	0.52	0.54	None	0.47	10.50
DummyClassifier	0.21	0.20	None	0.08	5.93

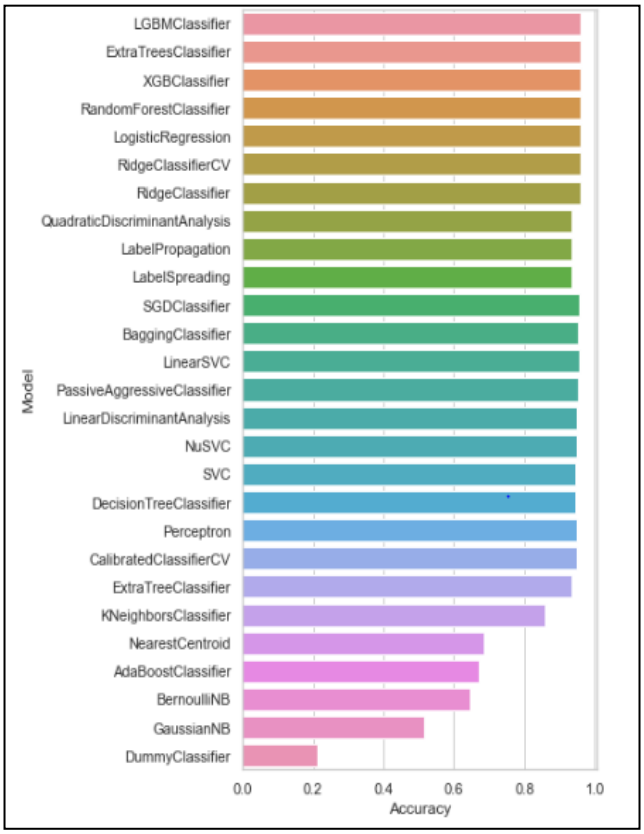


Figure 10 Lazy classifier accuracy

1.5.2 Pre-processing and Deep learning techniques:

Firstly: pre-processing step our dataset contains 2 different problems

1. it has unnecessary information that make model biased like: ID, Gender, line after this info that little bit look like signals that is in images, etc.
2. image quality is too low and this will lead to miss detecting.

So, to solve the first problem we found that signals ("our interest part") inside rectangle as we see in the next figure:

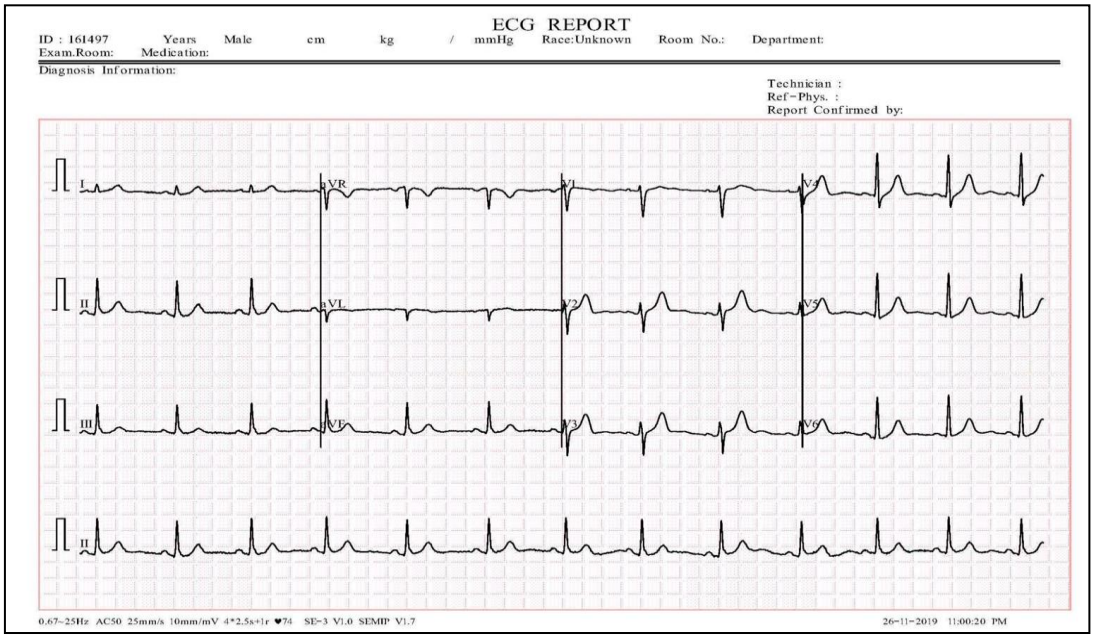


Figure 11 ECG image for normal person

- To solve this we decided to crop images depending on detecting the biggest rectangle using a special function we built called biggest **"biggestContour"** that It is used to detect the edges of a rectangle and another function called processing that gets the image ready to pass to the biggest contour function and calls it inside the processing function. you can find its implementation in Appendix section.

This figure shows the image after detecting the **biggestContour** function:

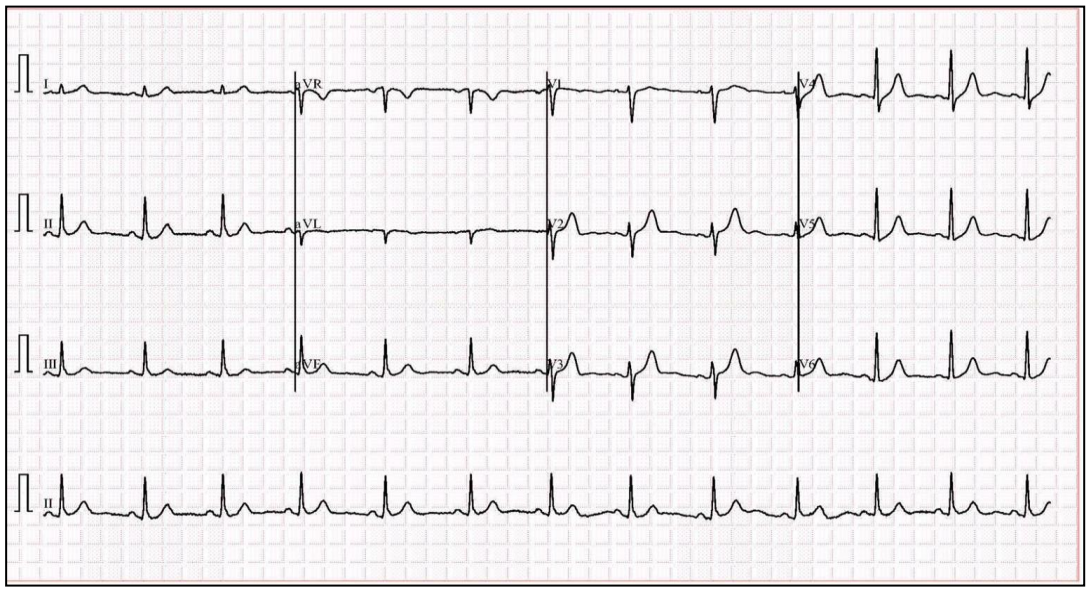


Figure 12 ECG image for normal person after being cropped

Now, let's solve the second problem: to improve image quality we used gamma correction technique. **Gamma correction** is a technique used in image processing to adjust the brightness and contrast of an image by modifying the gamma value. The gamma value is a parameter that controls the relationship between the pixel value and the brightness that is perceived by the human eye. In ECG images, gamma correction can be used to enhance the visibility of the signal and improve the accuracy of feature extraction algorithms. Here's a brief overview of how to use gamma correction with ECG images:

1. **Preprocessing:** The ECG signal is first preprocessed to remove baseline wander and noise. This can be done using filtering techniques such as wavelet denoising or median filtering.

2. **Image Conversion:** The preprocessed ECG signal is then converted into a grayscale image by plotting the signal amplitude over time. The x-axis represents time in seconds, and the y-axis represents the amplitude in millivolts.

3. **Gamma Correction** [3]**:** The gamma value is adjusted to enhance the visibility of the ECG signal. A higher gamma value will increase the contrast of the image, while a lower gamma value will decrease the contrast.

The main advantage of using gamma correction with ECG images is that it can improve the accuracy of feature extraction algorithms by enhancing the visibility of the signal. However, the choice of gamma value may affect the performance of the feature extraction and classification algorithms. A higher gamma value may result in over-enhancement of the ECG signal, while a lower gamma value may result in under-enhancement, so after many experiments we found that the best gamma factor is between range [1.2:1.4]. Here is equation [1] represent how gamma correction work on images:

$$s(X) = 255\left(\frac{X}{255}\right)^{1/\gamma(X)}$$

Figure 13 Gamma correction equation.

Here is a figure shows the effect of gamma correction:

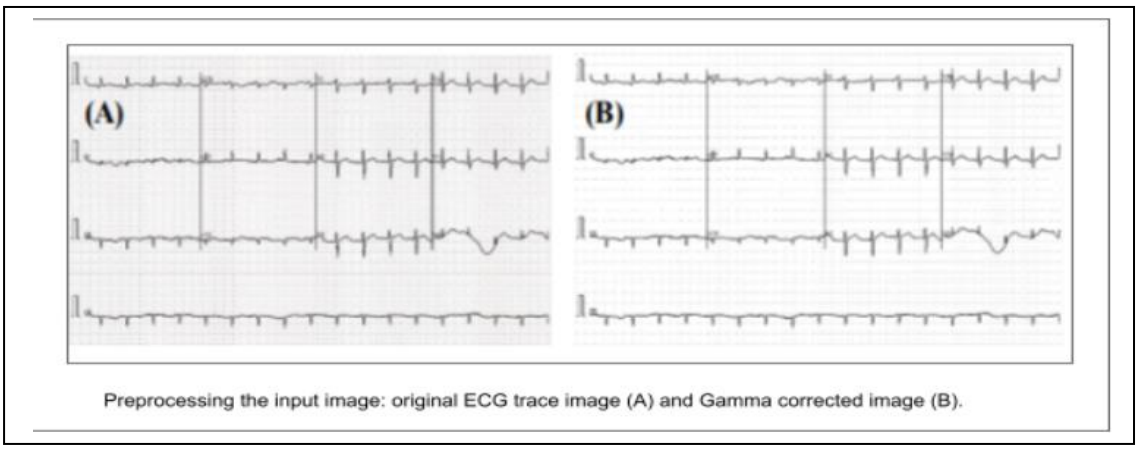


Figure 14. Preprocessing input image

Secondly: in the modeling step we used deep learning techniques specific to Convolutional Neural Networks (CNNs), transfer learning and CNN pre-trained models.

CNNs are a type of deep neural network that are commonly used in computer vision tasks such as image classification, object detection, and segmentation. CNNs are designed to automatically learn hierarchical representations of the input data by applying a series of convolutional and pooling layers.

Transfer learning is a technique used in deep learning where a pre-trained model is used as a starting point for a new task. The pre-trained model is typically trained on a large dataset, such as ImageNet, and has learned to extract features that are useful for various computer vision tasks. By using transfer learning, the new model can leverage the learned features from the pre-trained model and fine-tune them for a specific task with a smaller dataset.

CNN pre-trained models are pre-trained CNN architectures that have been trained on large-scale datasets such as ImageNet, and are available for use in various computer vision tasks. These pre-trained models can be used for transfer learning, where the pre-trained model is fine-tuned for a specific task using a smaller dataset. Some popular pre-trained CNN models include:

1. **VGG** (Visual Geometry Group) - a CNN architecture with up to 19 layers that achieved high accuracy in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014, and also give us a high accuracy with medical data.

2. **ResNet** (Residual Network) - a CNN architecture with a deep network structure that achieved high accuracy in the ILSVRC 2015.

3. **Inception** - a family of CNN architectures that use multi-level feature extraction modules called inception modules, which were designed to capture multi-scale and multi-level features.

4. **Efficient-Net** is a family of CNN models that were introduced in 2019 by Google AI researchers. Efficient-Net models are designed to achieve state-of-the-art performance on various computer vision tasks while minimizing the number of parameters and computational resources required for training and inference. The Efficient-Net architecture is based on a compound scaling method that scales the network depth, width, and resolution in a systematic way. This allows Efficient-Net models to achieve high accuracy with fewer parameters and computational resources compared to other CNN models.

In our project we used 7 pre-trained models: EfficientNetB0, EfficientNetB3, DenseNet201, InciptionNetV3, MobileNetV2, ResNet50 and VGG19 with Five-fold cross validation was used and therefore, 80% of data were used for training and 20 % for testing. Out of the training dataset subset, 10% were utilized for validation to avoid overfitting issues. Finally, the results were a weighted average of five folds.

Here is a table of the dataset [4] information:

Types	Total No. of images/ class	Train set count/fold	Validation set count/fold	Test set count/ fold
Normal	390	312	31	78
COVID-19	250	200	20	50
Abnormal HB	390	312	31	78
Myocardial Infarction	382	305	30	77
Recovered MI	382	305	30	77

Table 1 shows the details of the number of training, validation, and test ECG images used.

Models and their accuracy:

Model	Average accuracy
EfficientNetB0	98.062%
EfficientNetB3	98.065%
DenseNet201	97.793%
InciptionNetV3	97.922%
ResNet50	95.569%
VGG19	96.665%
MobileNetV2	10.2%

Table 2 Comparison of the performances of the different CNN models

And here a figure shows the classification report of last fold for each model:

DenseNet201 classification report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	39
1	1.00	1.00	1.00	82
2	0.97	1.00	0.99	68
3	1.00	0.99	0.99	87
4	0.99	0.99	0.99	83
accuracy			0.99	359
macro avg	0.99	0.99	0.99	359
weighted avg	0.99	0.99	0.99	359

EfficientNetB0 classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	39
1	1.00	1.00	1.00	82
2	1.00	1.00	1.00	68
3	1.00	1.00	1.00	87
4	1.00	1.00	1.00	83
accuracy			1.00	359
macro avg	1.00	1.00	1.00	359
weighted avg	1.00	1.00	1.00	359

EfficientNetB3 classification report

	precision	recall	f1-score	support
0	1.00	0.97	0.99	39
1	1.00	1.00	1.00	82
2	0.99	1.00	0.99	68
3	1.00	1.00	1.00	87
4	0.99	0.99	0.99	83
accuracy			0.99	359
macro avg	0.99	0.99	0.99	359
weighted avg	0.99	0.99	0.99	359

InceptionNetV3 classification report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	39
1	0.99	1.00	0.99	82
2	1.00	1.00	1.00	68
3	0.99	1.00	0.99	87
4	1.00	0.98	0.99	83
accuracy			0.99	359
macro avg	1.00	1.00	1.00	359
weighted avg	0.99	0.99	0.99	359

ResNetV50 classification report

	precision	recall	f1-score	support
0	1.00	0.92	0.96	39
1	1.00	0.99	0.99	82
2	0.99	1.00	0.99	68
3	0.91	0.99	0.95	87
4	0.96	0.92	0.94	83
accuracy			0.97	359
macro avg	0.97	0.96	0.97	359
weighted avg	0.97	0.97	0.97	359

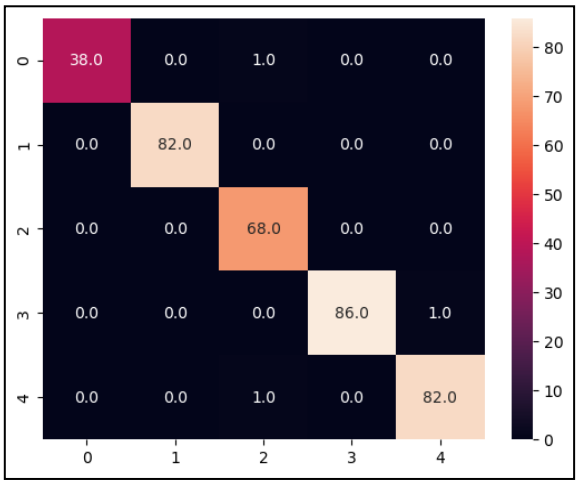
VGG19 classification report

	precision	recall	f1-score	support
0	1.00	0.95	0.97	39
1	1.00	0.98	0.99	82
2	0.94	0.96	0.95	68
3	0.93	0.98	0.96	87
4	0.98	0.96	0.97	83
accuracy			0.97	359
macro avg	0.97	0.96	0.97	359
weighted avg	0.97	0.97	0.97	359

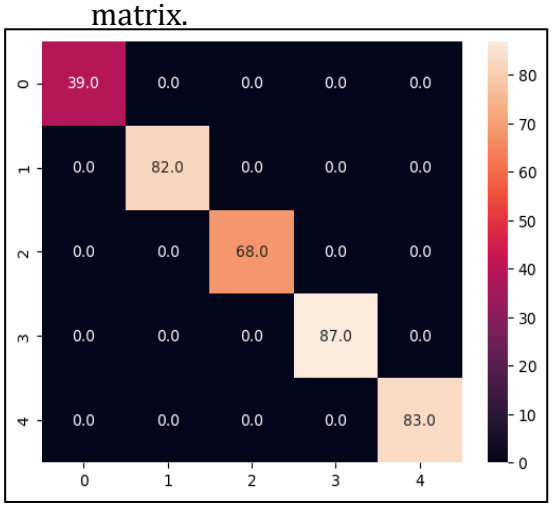
Figure 15 Classification reports of models

Figure 16 Shows the confusion matrix of last fold for each model

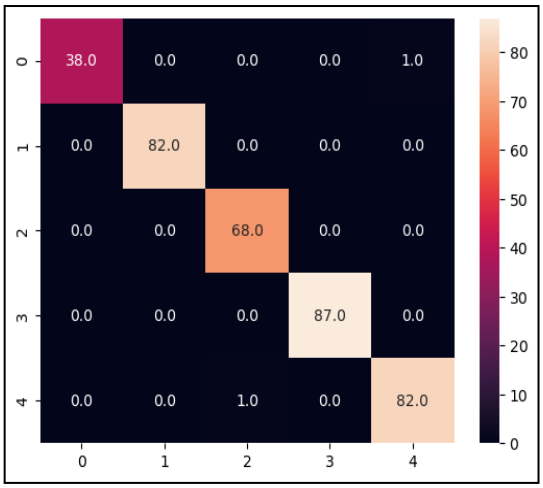
DenseNet201 confusion matrix.



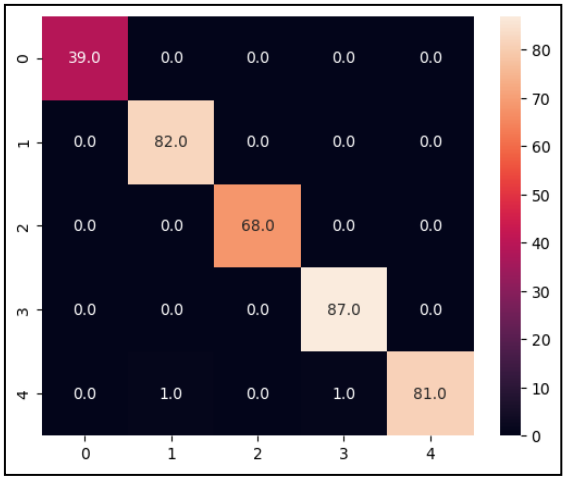
EfficientNetB0 confusion matrix.



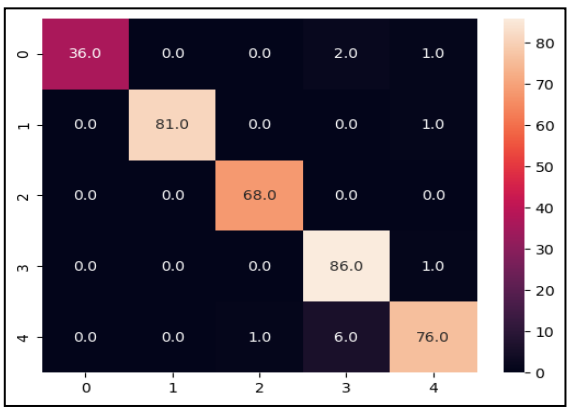
EfficientNetB3 confusion matrix.



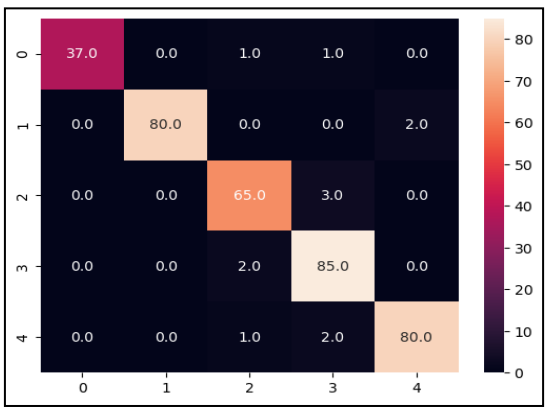
InciptionNetV3 confusion matrix.



ResNet50 confusion matrix.



VGG19 confusion matrix.



Website Application:

For the **front-end methodologies,** we employed a professional approach to achieve our desired results. We utilized the reputable online marketplace called Codecanyon, specializing in the sale of pre-built software components, plugins, and scripts. This allowed us to acquire unique and high-quality GUI templates tailored to our requirements.

Additionally, we harnessed the power of the Bootstrap framework, which greatly facilitated the development of responsive and mobile-friendly websites and web applications. Bootstrap provided us with a comprehensive collection of pre-built HTML, CSS, and JavaScript components and utilities that we seamlessly integrated into our projects.

For **backend methodologies**, we adopted a professional approach centered around the utilization of the Laravel framework[5]. Recognized for its elegant syntax, robust features, and developer-friendly nature, Laravel served as the cornerstone for constructing scalable and maintainable web applications, catering to a spectrum ranging from small websites to enterprise-level systems.

One of the distinguishing features of Laravel is its adherence to the Model-View-Controller (MVC) architectural pattern. This pattern entails the deliberate separation of the application's logic, presentation, and data layers. By embracing this division, we achieved an elevated level of code organization, modularity, and testability. The MVC architecture facilitated a clear separation of concerns, enabling us to efficiently manage the different aspects of our application.

This figure shows the MVC pattern:

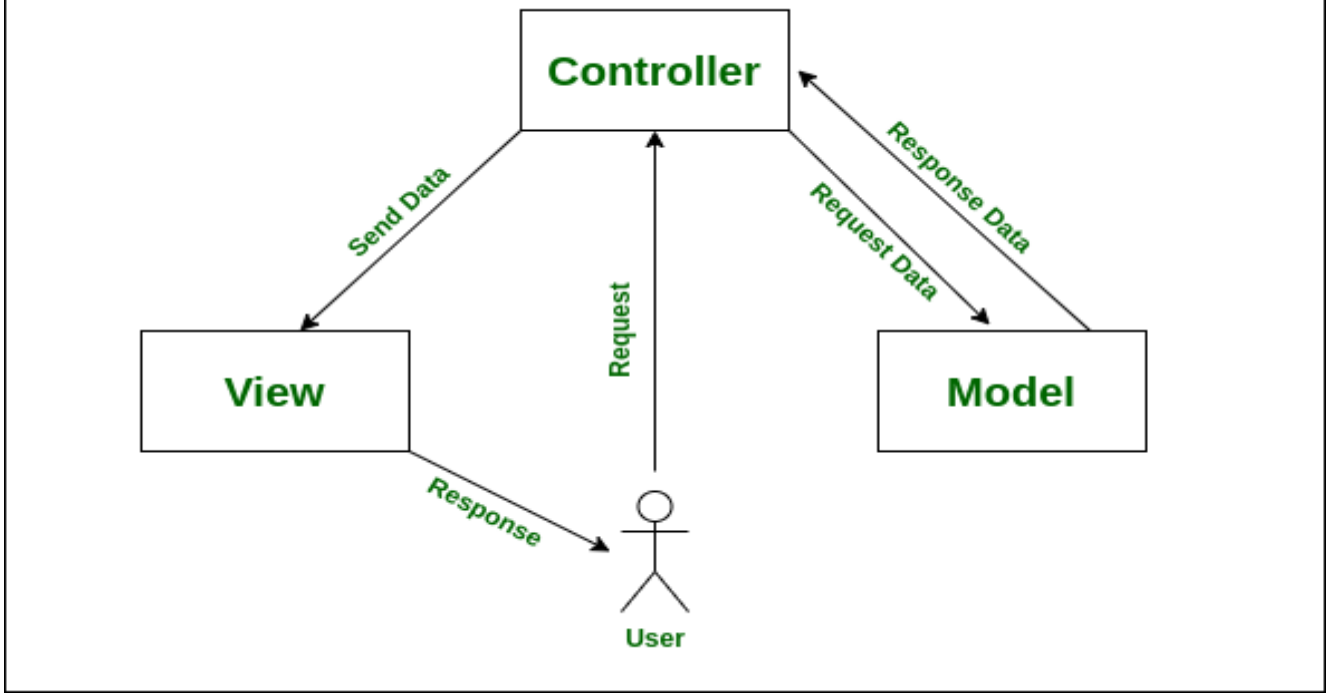


Figure 17 MVC Pattern

Through the implementation of our Registration and Login functionality, users gain access to our Model page, where they can upload an electrocardiogram (ECG) image and receive a comprehensive report. This report classifies the individual as either "Normal," "Covid-19," "RMI," "MI," or "AHB," based on the analysis of the uploaded image.

To facilitate this crucial phase, we leveraged Flask, a middleware (API) framework, to establish seamless communication between the website page and the Efficient-Net b3 Model. This integration ensured a smooth and efficient workflow for the analysis of ECG images.

During this process, users are prompted to select an ECG image from the Gallery as a file. However, we encountered a logical error when attempting preprocessing. As the PIL library primarily operates on images rather than files, we encountered incorrect results due to improper preprocessing. To rectify this issue, we implemented a solution whereby we convert the file into an image extension. This conversion allowed us to effectively perform the necessary preprocessing steps accurately.

The preprocessing process encompasses several essential steps:

1. Converting the file into an image format.
2. Detecting the rectangular boundaries of the ECG image within the file.
3. Applying the Gamma correction technique to enhance the visual quality and optimize subsequent analysis.

You can find the preprocessing function and validation of login and registration at Appendix section.

Flutter Mobile Application:

Introducing a Flutter application that enables users to classify electrocardiogram (ECG) images with a high degree of accuracy and precision. The application allows users to select an ECG image from their gallery and upload it to a server for classification. The server utilizes an advanced machine learning model to analyze the ECG image and classify it into one of five distinct categories: normal person, COVID patient, myocardial infarction patient, abnormal heartbeats, and patients with history of MI.

The application's classification algorithm is designed to provide users with reliable and actionable insights into ECG data, allowing them to make informed decisions regarding patient care, research, and analysis. The application's user-friendly interface and technology make it an essential tool for medical professionals, researchers, and individuals with a keen interest in ECG analysis.

The application is connected to the machine learning model using Render server, we specifically chose Render because Render is a cloud platform that provides developers with a powerful and flexible infrastructure for building and deploying web applications and services. Render's serverless architecture enables developers to focus on writing code rather than managing servers, allowing for faster development cycles and reduced operational overhead, It also supports a wide range of programming languages, frameworks and libraries, it’s platform can handle thousands of requests per second and allows free access.

App Authentication:

For authentication we used *Shared Preferences* library. *Shared preferences* are a powerful and convenient tool for managing local data storage in Flutter applications. It provides a simple and efficient way to store and retrieve data that can be easily accessed throughout the application. One of the main benefits of using shared preferences is that it allows developers to store small amounts of data, such as user preferences, authentication tokens, and app settings, without the need for a complex database setup. This makes it a great choice for applications that require simple data storage that can be accessed quickly and easily. Another advantage is that Shared Preferences is well-documented and can be integrated in any Flutter project easily, making it easy for developers to add, modify and retrieve data as needed. it's also a local data storage tool so this will help the user to keep access to the application even if there was a problem with the server connection, where it depends on the idea of caching frequently used data so that, the user doesn't have to enter his/her data again every time he/she uses the app.

The used tools in the project:

Throughout the course of this study, we have employed various essential resources, encompassing both software and hardware components, to fulfill our research requirements.

- **Hardware:**
  This Project does not need any specific hardware except one's personal computer or smart phone.

- **Software:**
  - **Programming Languages:**

Python	Is a popular programming language for machine learning because it has many powerful libraries and frameworks that make it easy to implement machine learning algorithms.
Flask	Is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
JavaScript	Is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.
PHP	Is a general-purpose scripting language and interpreter that is freely available and widely used for web development.
HTML	is the standard markup language for documents designed to be displayed in a web browser. It is often assisted by technologies such as CSS or scripting languages such as JS.

CSS	Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTM.
Laravel	is a free and open-source PHP web framework intended for the development of web applications following the model–view–controller (MVC) architectural pattern.
Flutter	Flutter is a free and open-source UI framework for creating native mobile applications from Google.
Dart	Dart is a programming language that developed by Google. Can be used to develop web and mobile apps as well as server and desktop applications.
SQL	Stands for Structured Query Language which is basically a language used by databases.

Table 3 Programming languages

- **IDEs and programs:**

PyCharm	PyCharm is an integrated development environment used for programming in Python.
Visual studio code	commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Android studio	Is the official integrated development environment (IDE) for Android application development.
Google Colab	Is a cloud based Jupyter notebook environment. It runs in your web browser (you can even run it on your favorite Chromebook) and lets anyone with internet access experiment with machine learning and coding for artificial intelligence.
XAMPP	Is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

Table 4 IDEs and Programs.

- **Libraries that must be installed:**

SKLearn	Which is probably the most useful library for machine learning in Python with efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction-which needed for (CNN, SVM, and RNN) models
TensorFlow	Is an end-to-end platform that makes it offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API which makes it easy for us to build and deploy ML models.

Keras	Is an API designed for human beings which follows best practices for reducing cognitive load and offers consistent & simple APIs.
OpenCV (Cv2)	is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking.

Table 5 Libraries that must be installed.

Report Organization:

Chapter Two:

- **Related Work:** In chapter two, we will establish other work associated with our research. There, we show different ways to achieve our goals. We will declare the authors of these methods, the classification method they use in their project, and of course the accuracy.

Chapter Three:

- **System Analysis:** In chapter three, we will talk about project specification. It will contain descriptions of how the program will be used from a user perspective and performance details such as usability, reliability, and stability. In addition, illustrate a use case diagram that emphases our program.

Chapter Four:

- **System Design:** The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces using some diagrams. System Component Diagram, System Class Diagrams, Sequence Diagrams, Project ERD, System GUI Design.

Chapter Five:

- **Implementation and Testing:** This phase includes screen shots of training and testing of models, the website, and the mobile application.

Chapter two: Related Work

Various studies have recently revealed that ECG can be applied for the diagnosis of COVID-19 due to the changes caused in signals, there are many studies that try to show it and find a vital treatment to get rid of it.

1- The authors Published this study [6] in August 2022

This paper [6] worked on dataset that consists of 3915 images.

Research object: this paper proposes an automated scheme for detection and classification COVID-19 using deep learning approaches that consists of 4 steps:

1.1- Image Preprocessing: which is performed in four stages, cropping the ECG images, masking, median filter, and sharpening based on unsharp masking then finally we get a filtered COVID-19 ECG image report using the four proposed stages.

1.2- Data augmentation: they are used two types of data augmentation, basic image manipulations and augmentation based on deep learning approaches.

1.3- Feature Extraction: they are used five pre-trained models to extract features, Vgg16, Vgg19, Resnet101, Xception, Grouped Convolutional layer (The Proposed ECGConvnet Model) in addition, a novel model is proposed known as ECGConvnet.

1.4- Classification: The last process, and it is the phase that tests the performance of all it, they are used four different classifiers: SoftMax, RF, MLP, and SVM.

2- This study [7] was proposed by the authors in November 2022.

In paper [7] they worked on dataset that consists of 1937 images.

Research object: These paper shows that, they are used pre-trained deep learning models are used for COVID-19 detection in ECG images.
The optimized weighted average ensemble method is used to improve the performance for multi-class classification.
The obtained results shows that the deep models can be used in early detection of COVID-19 in ECG images, in this paper, they are using highly imbalanced dataset just do a cropping and filtration to it in the preprocessing stage.

The article proposes the deep learning models that are used for predicting the COVID-19 in binary and multi-class classification using ECG images.

The proposed method has four steps:

2.1- Transfer learning technique: In medical image analysis, there is no large dataset available due to privacy issues to train the deep learning models. Therefore, the researchers use the transfer learning technique to train their models. Different research groups have released pre-trained models for reuse it.

2.2- Optimized weighted average ensemble method: A single model cannot extract all the features of the dataset. Therefore, researchers use an ensemble of different models to enhance the performance. In this paper, the optimized weighted average ensemble method is used for multi-class classification. This ensemble technique is an enhancement of the average ensemble method in which all the models contribute equally to generating predictions.

2.3- Model architecture: The CNN model can be divided into the feature extraction block and the fully-connected block. In this study, the feature extraction blocks of pre-trained CNN models such as VGG-19, DenseNet-121, and EfficientNet-B4 are used for the classification. The fully-connected layers of these pre-trained models are removed to reduce the trainable parameters.

2.4- Classification: The last process, and it is the phase that tests the performance of all it, they are used VGG-19, DenseNet-121, and EfficientNet-B4.

3- This study [8] was proposed by the authors in January 2023.

In paper [8] they worked on dataset that consists of 1937 images.

Research object: This paper provides a deep Convolutional Neural Networks-based transfer learning strategy for the automated diagnosis of COVID-19 and other cardiac disorders using ECG trace images. The performance of the six different CNN models was evaluated for the classification of three different schemes: two-class classification (Normal and COVID-19), three-class classification (Normal, COVID-19, and Cardiac abnormality) and five-class classification (Normal, COVID-19, myocardial infarction (MI), Abnormal heartbeat (HB), and recovered MI).

The proposed method has three steps:

3.1- gamma correction enhancement technique: non-linear procedure that is applied to the pixels of a source image. To improve the image, gamma correction employs the projection relationship between the pixel value and the gamma value according to the internal map.

3.2- Augmentation: dataset does not have a similar number of images for the different categories, training with an imbalanced dataset can produce a biased model. Thus, data augmentation for the training set can help in having a similar number of images in the various classes, which can provide reliable results as stated in many recent publications, three augmentation strategies (rotation, scaling, and translation) are used.

3.3- Classification: The last process, and it is the phase that tests the performance of all it, they are ResNet18, ResNet50, ResNet101, InceptionV3, DenseNet201, and MobileNetv2.

The main differences between their studies and our study:

- we use different models of CNN that more efficient which are Efficient-Net models (B0 and B3) and used 7 pre-trained models in our project: EfficientNetB0, EfficientNetB3, DenseNet201, InciptionNetV3, MobileNetV2, ResNet50 and VGG19.

- We use Shallow Learning (Lazy Classifier).

- we use more techniques for data preprocessing (Cell Formation, Rectangle detection, Gamma correction, Augmentation).

- We have added more images from different resource to make dataset larger.

- We will make use of our model in a friendly application programming interface API to make it easier.

	Classifier	Accuracy	Limitations
Study [6] Published in August 2022.	**SoftMax Random Forest (RF). Multi-Layer Perception (MLP). Support Vector Machine (SVM).**	**96.4% 91.8% 97.5% 98%**	**Models that applied for training isn`t the best models, there is a model can give a good accuracy than it. The diagnosis system that they used taking a long time to run. Accuracy isn`t the best of all models except SVM.**
Study [7] Published in November 2022.	**Convolutional neural network (CNN). VGG-19. DenseNet-121. EfficientNet-B4.**	**99.09% 92.35% 92.95% 95.29%**	**The proposed model is trained and tested on a limited dataset. The proposed model needs to be validated on large ECG datasets. Data amount is small. CNN Which gives high accuracy but take a large time to processed.**
study [8] Published January 2023.	**Convolutional neural network (CNN). ResNet18. ResNet50. ResNet101. InceptionV3. DenseNet201. MobileNetv2.**	**99.1% 99.11% 99.1% 97.36% 97.4% 97.36%**	**Preprocessing methods should be updated and use additional methods behind Gamma correction. The proposed models are trained and tested on a limited dataset. Data amount is very small. CNN Which gives high accuracy take a large time to processed.**

Our study	**EfficientNetB0** **EfficientNetB3** **DenseNet201** **InciptionNetV3** **ResNet50** **VGG19**	**98.062%** **98.065%** **97.793%** **97.922%** **95.569%** **96.665%**	**The proposed models are trained and tested on a limited dataset.** **Models are developed at Limited hardware resources.**

Table 6 Related Work

Chapter 3: System Analysis

Project specification:

Functional Requirements:

- **Registration:** If this is the first time for the user to use the application, he/she will register.

- **Login:** If the user already registers before, the user will login to the app with his user's name and password.

- **Image input:** The user will input the image that he/she wants to predict if it has a disease (Covid19/heart) or it's normal. The image will be processed and transferred to the model that will give the result and print a report to the user.

Non-Functional Requirements:

- **Performance:** For example, Response Time, Throughput, Utilization, Static Volumetric.

- **Usability:** Prioritize the important functions of the system based on usage patterns. Frequently used functions should be tested for usability, as should complex and critical functions.

- **Reliability:** Users must trust the system, even after using it for a long time. Create a requirement that data created in the system will be retained for several years without the data being changed by the system. Make it easier to monitor system performance.

- **Security:** Security is the degree of resistance to, or protection from, harm. It's important to have this feature as a requirement.

- **Availability:** refers to a property of software that is there and ready to carry out its task when you need it to be.

Use Case Diagram:

The use case represents the functionality of the COVID-19 prediction model, allowing users to register and then login into the system then can use in, so user can input relevant data (images) and obtain a prediction of this image.

This figure shows the use case diagram.

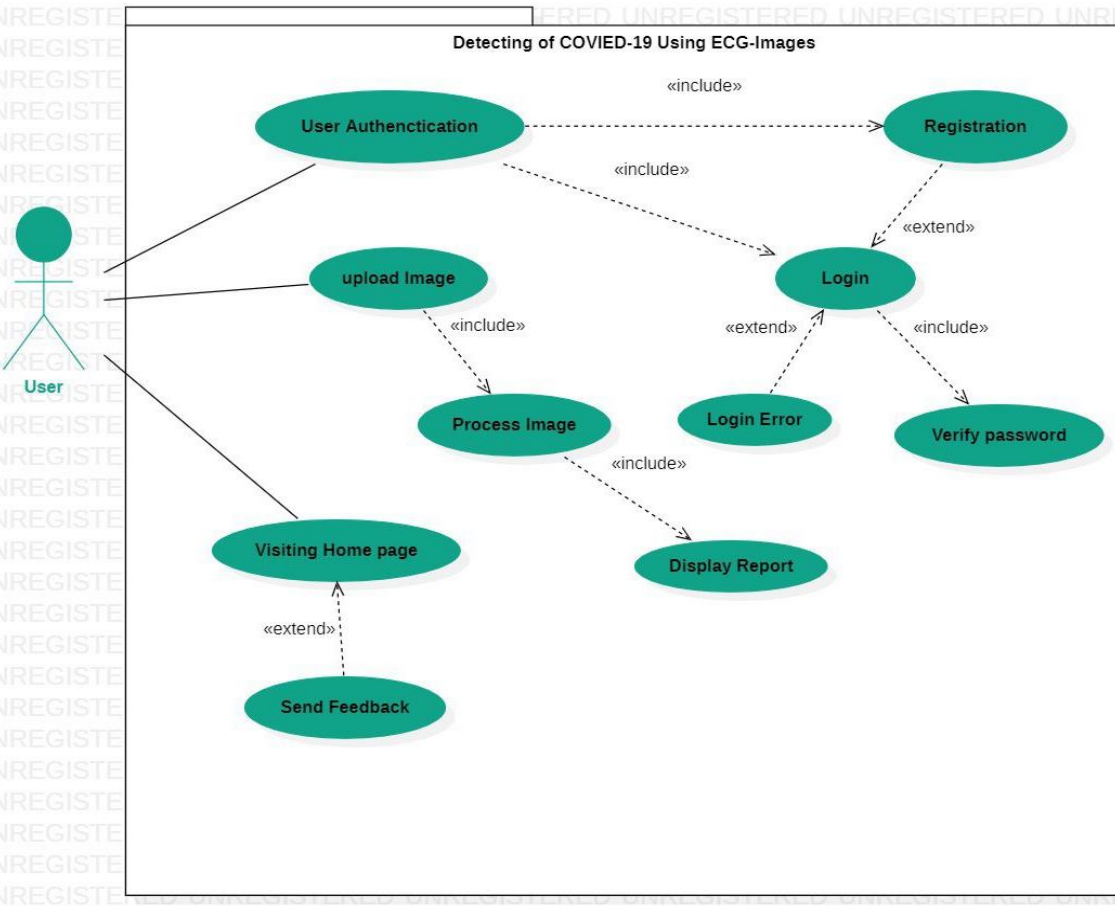


Figure 18 Use case diagram.

Chapter 4: System Design

System Component Diagram:

This figure shows the system component diagram.

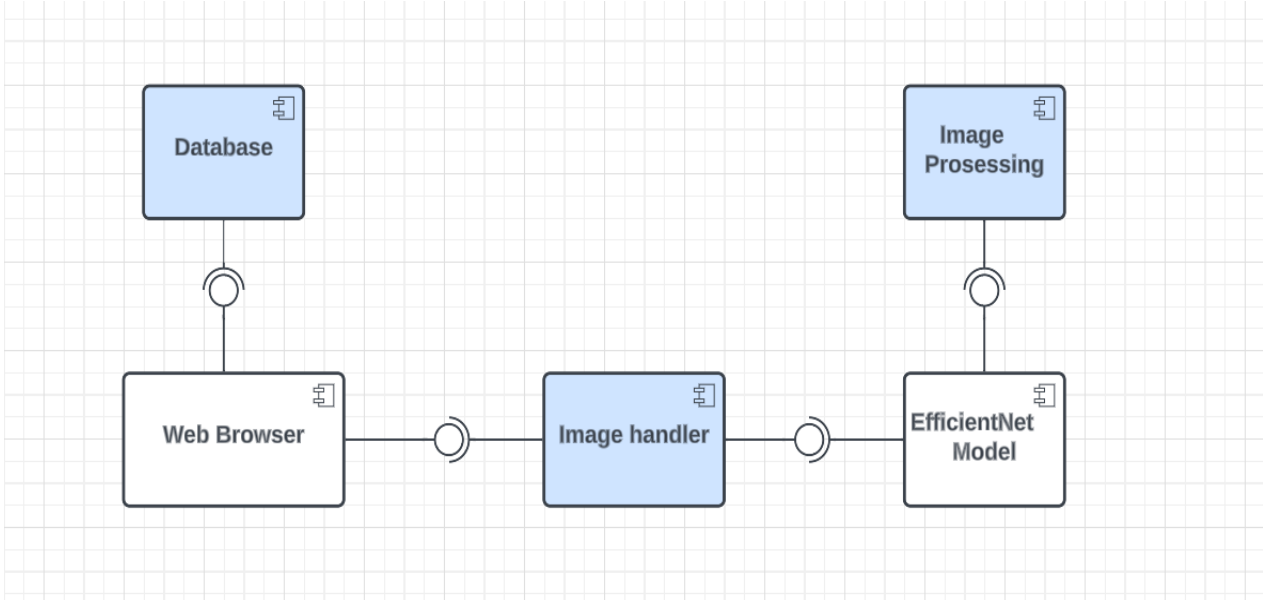


Figure 19 System component

System Class Diagrams:

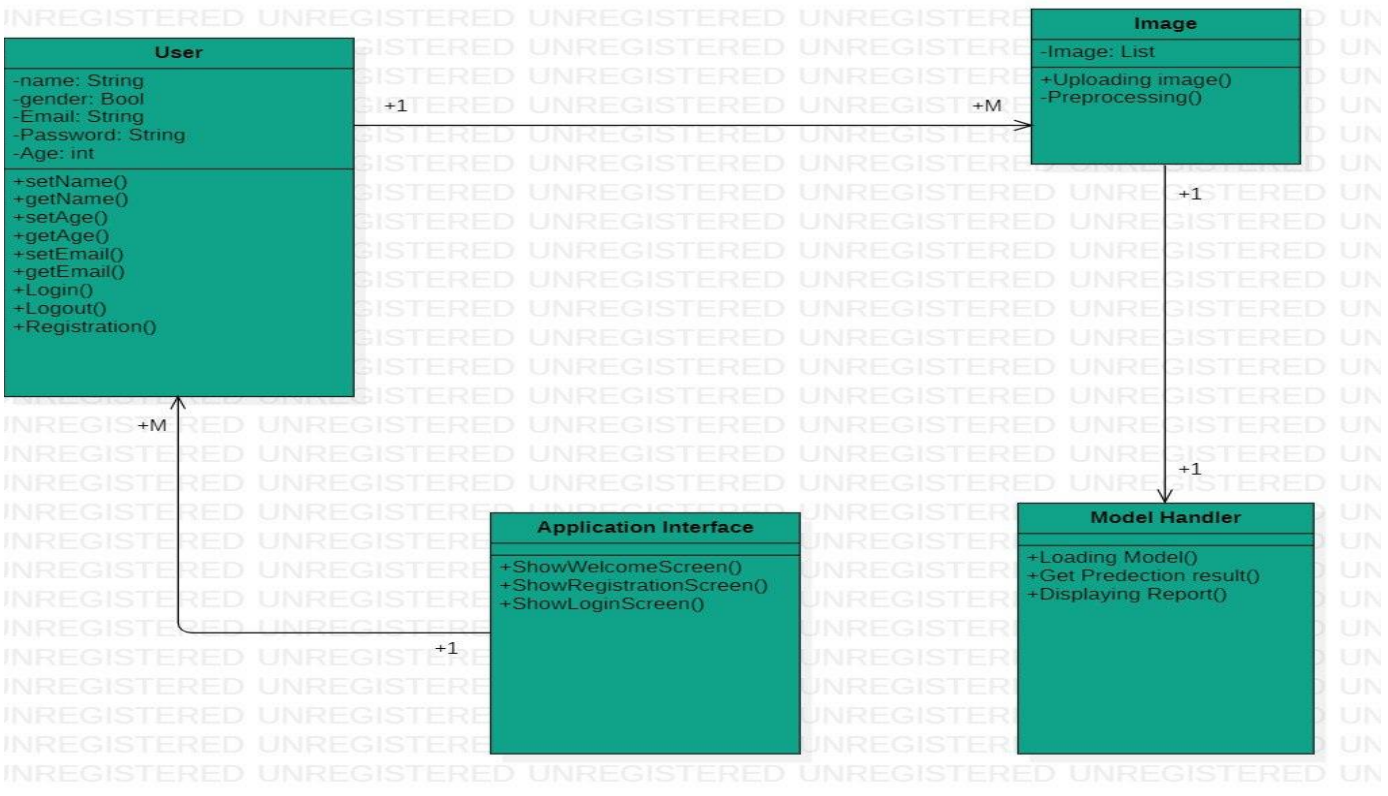


Figure 20 Class diagram

Sequence Diagrams:

This figure shows sequence diagram:

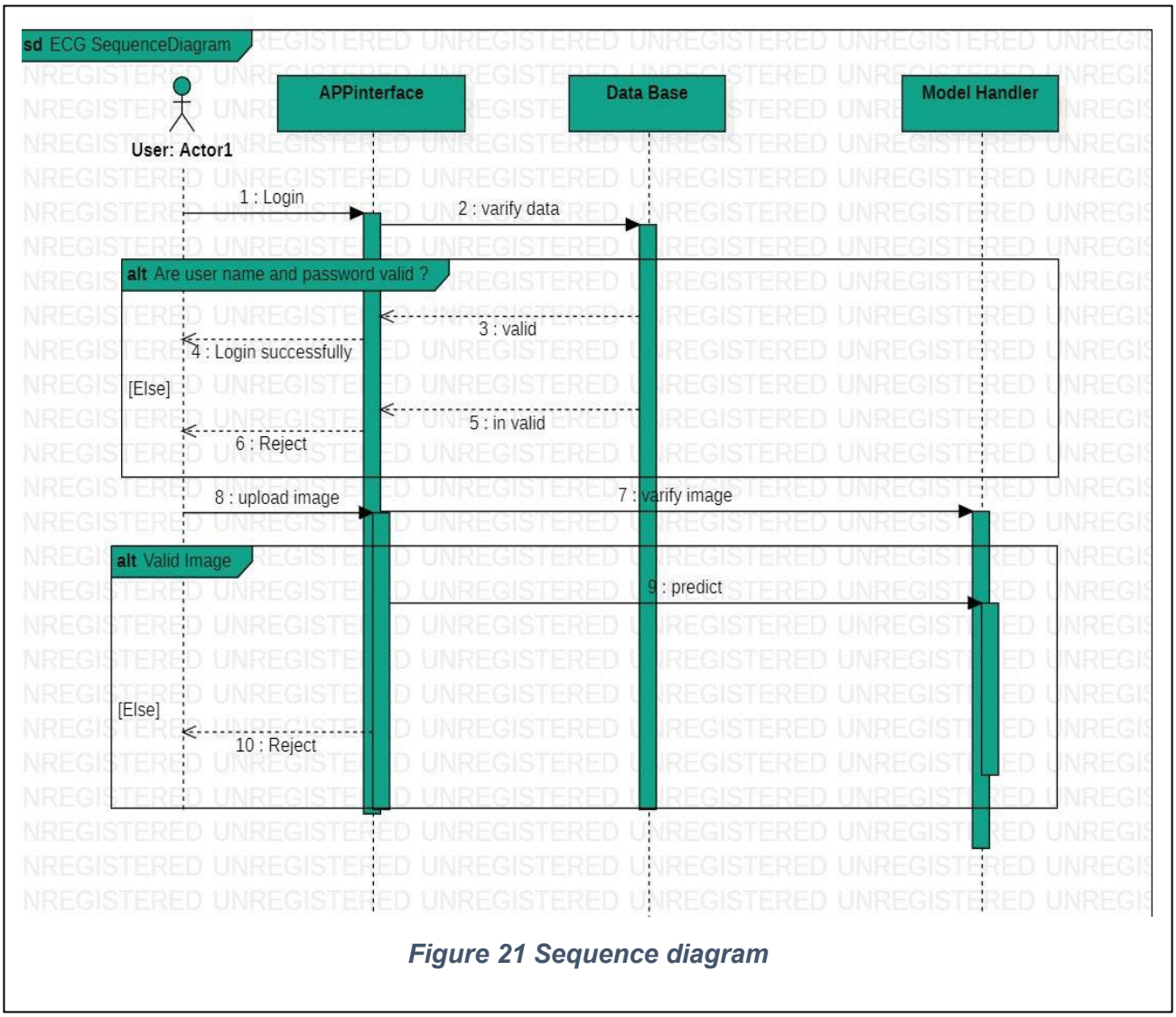


Figure 21 Sequence diagram

Project ERD:

This figure shows ERD:

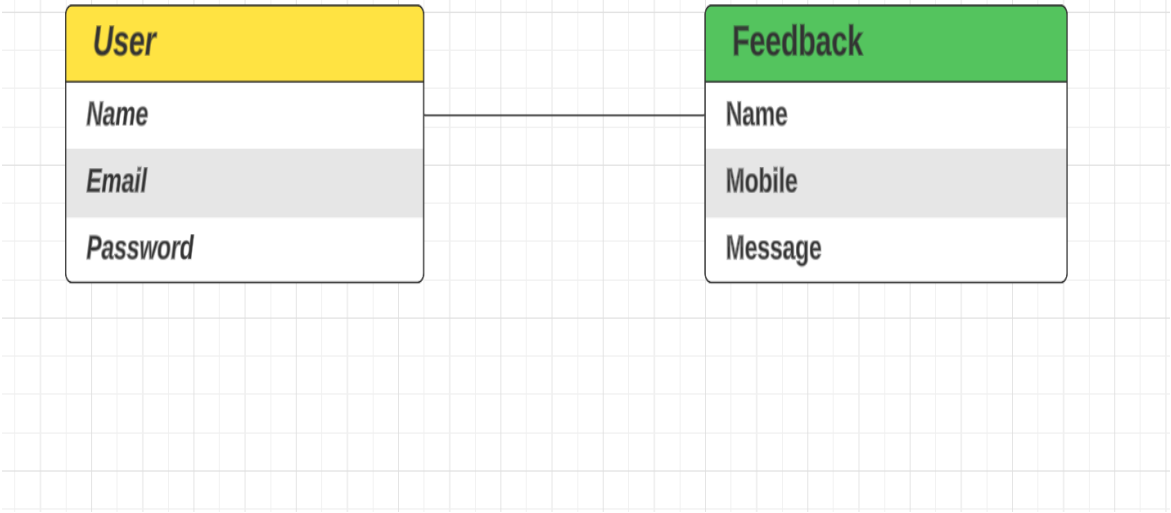


Figure 22 ERD

System GUI Design:

This figure shows the Graphical user interface:

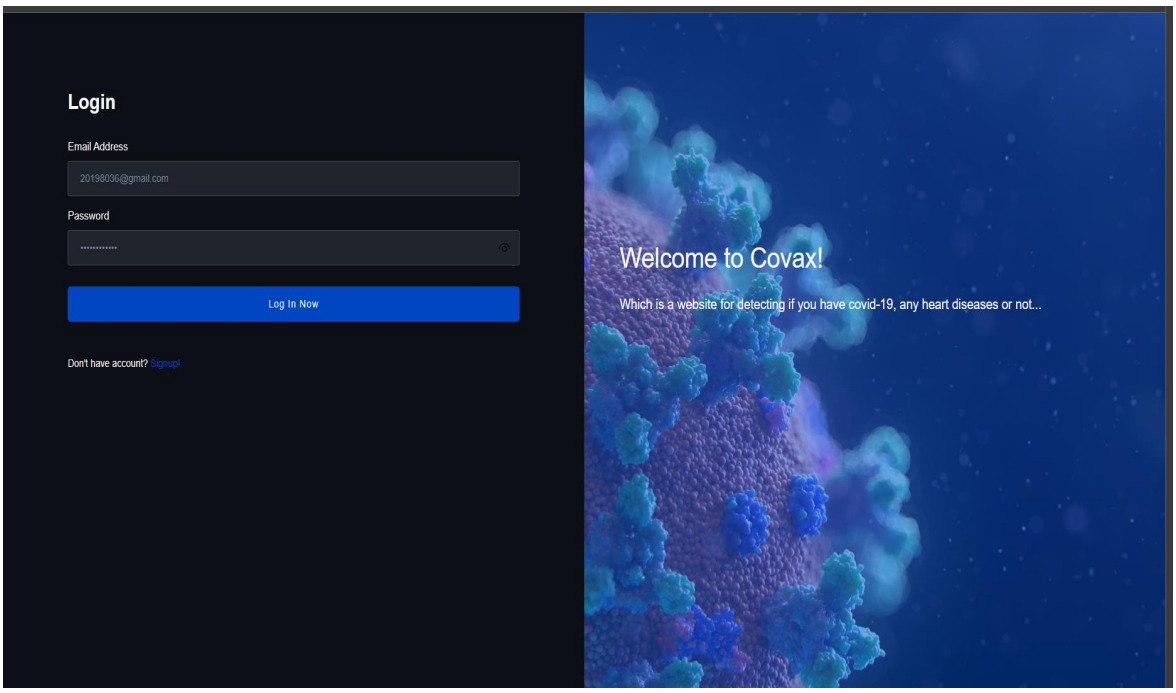


Figure 23 GUI

Chapter 5: Implementation and Testing

COVAX Website

Home Page

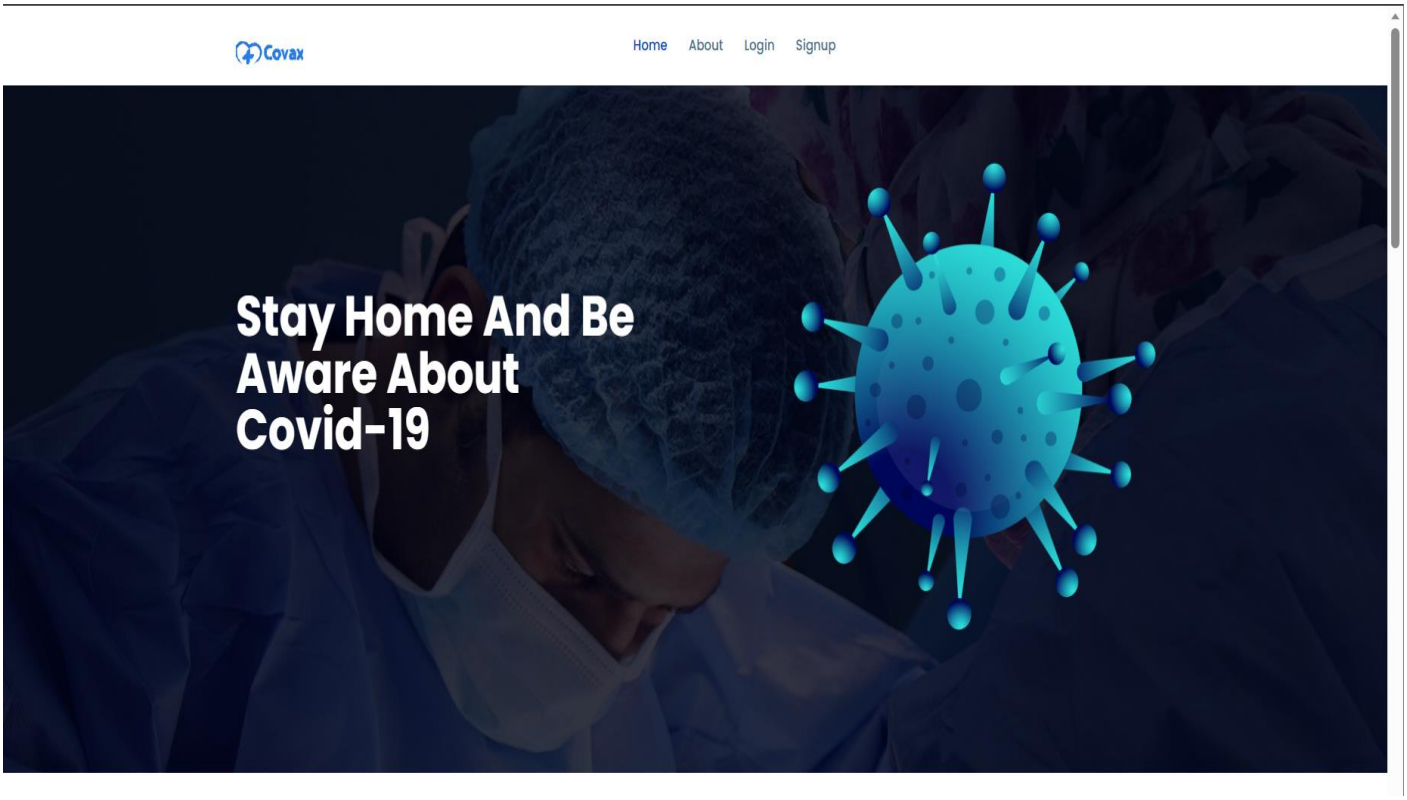


Figure 24 Home page screen

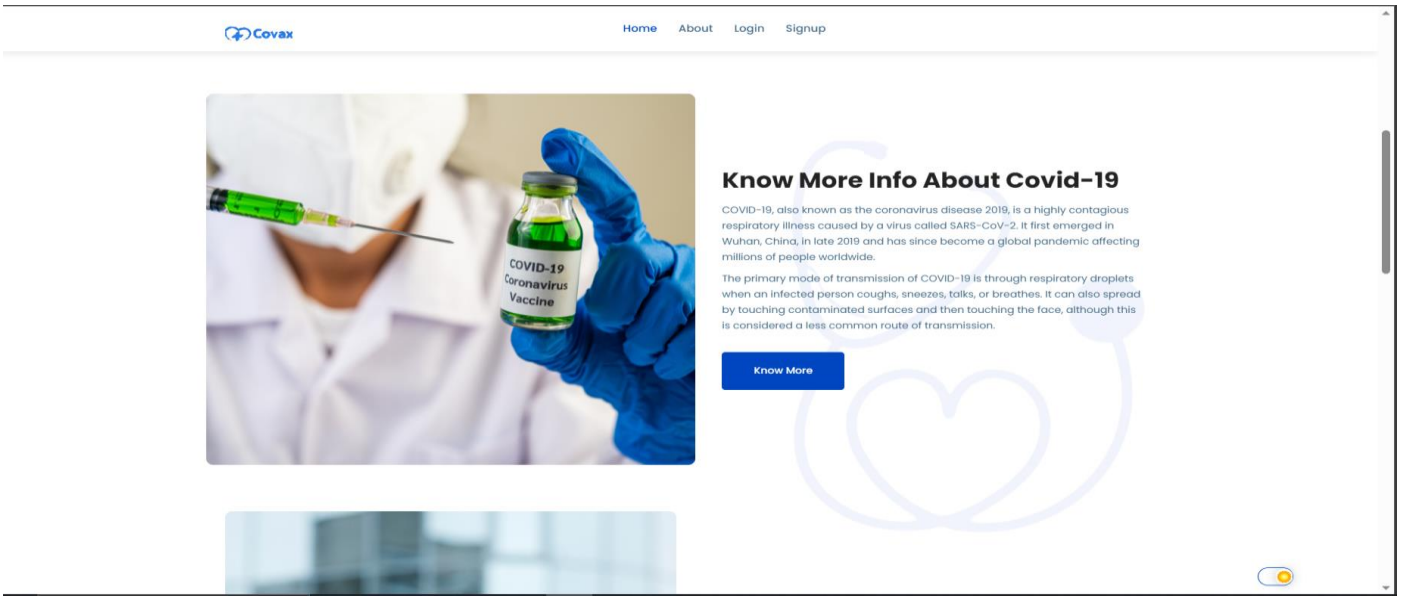


Figure 24.1 Home page screen

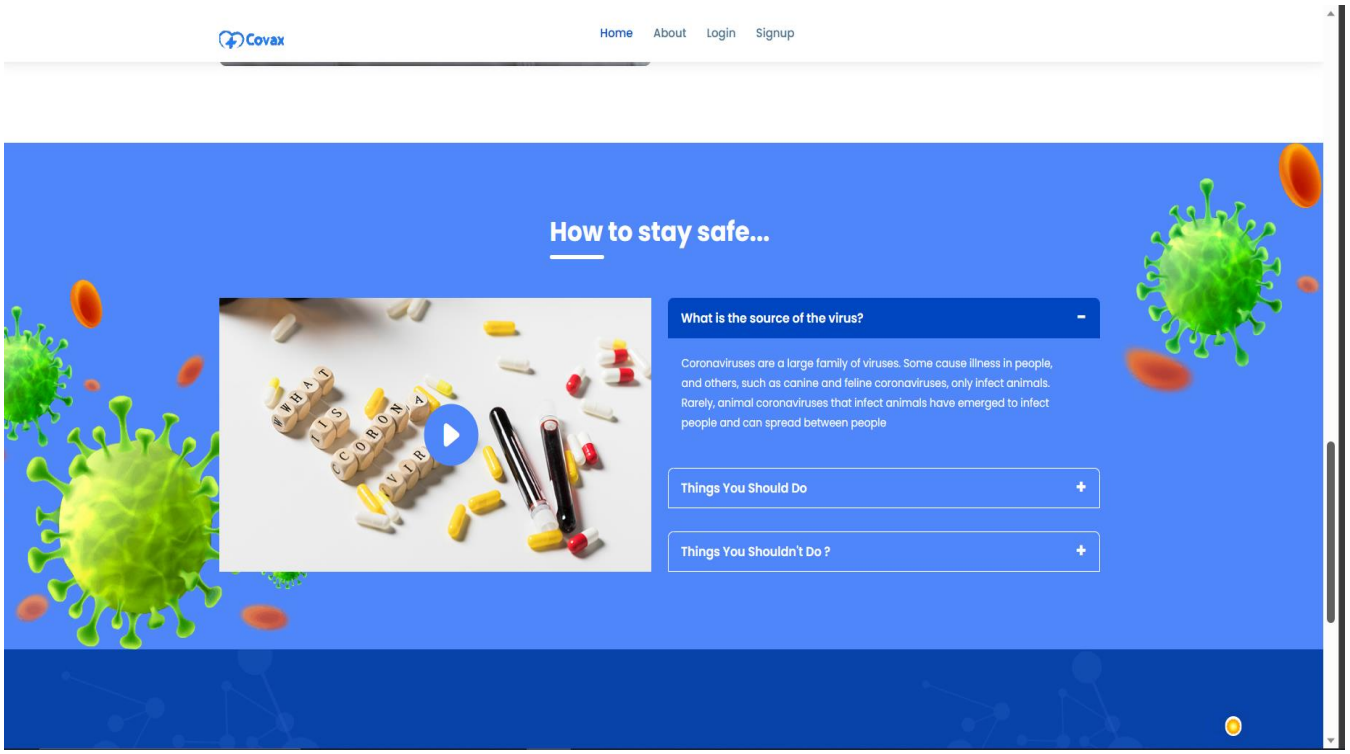


Figure 24.2 Home page screen

Registration page:

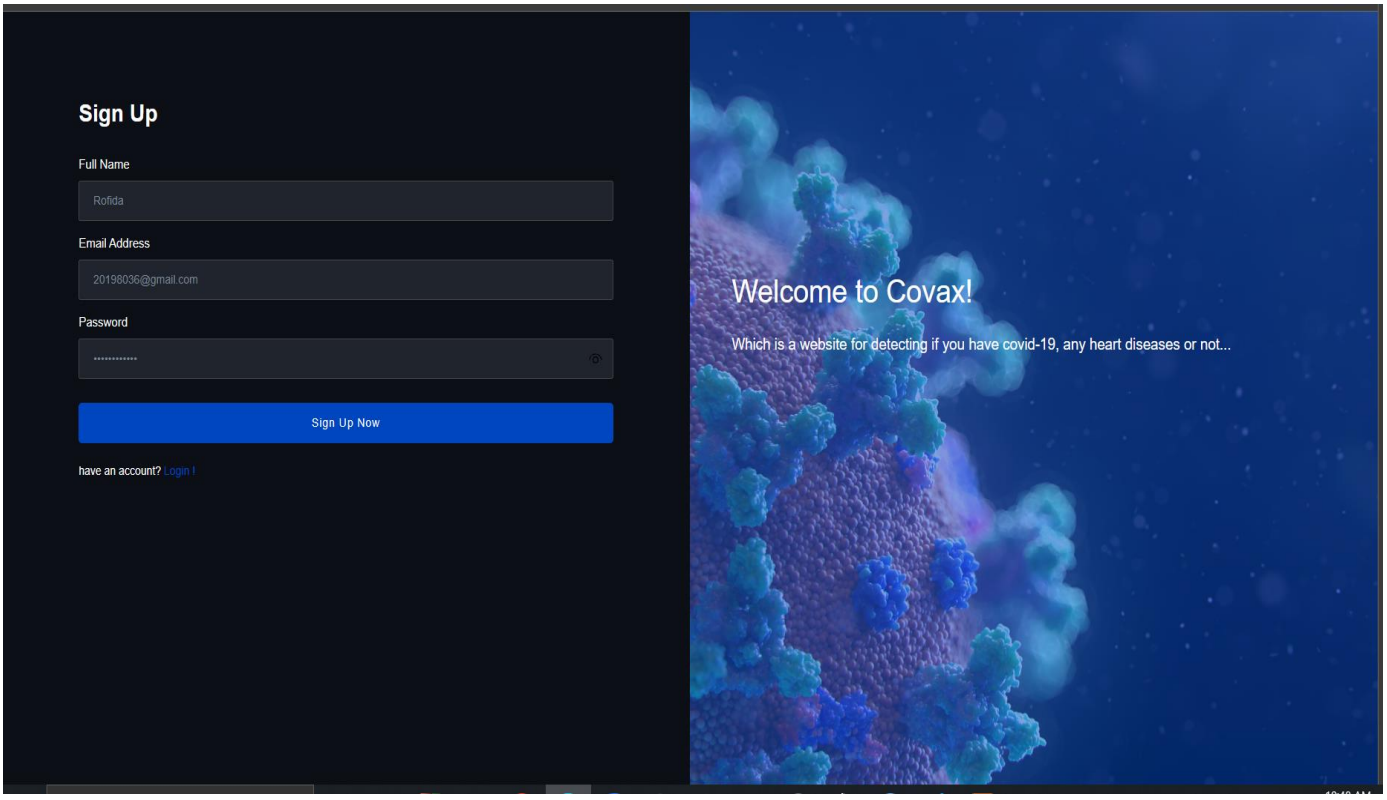


Figure 25 Registration screen

Login Page:

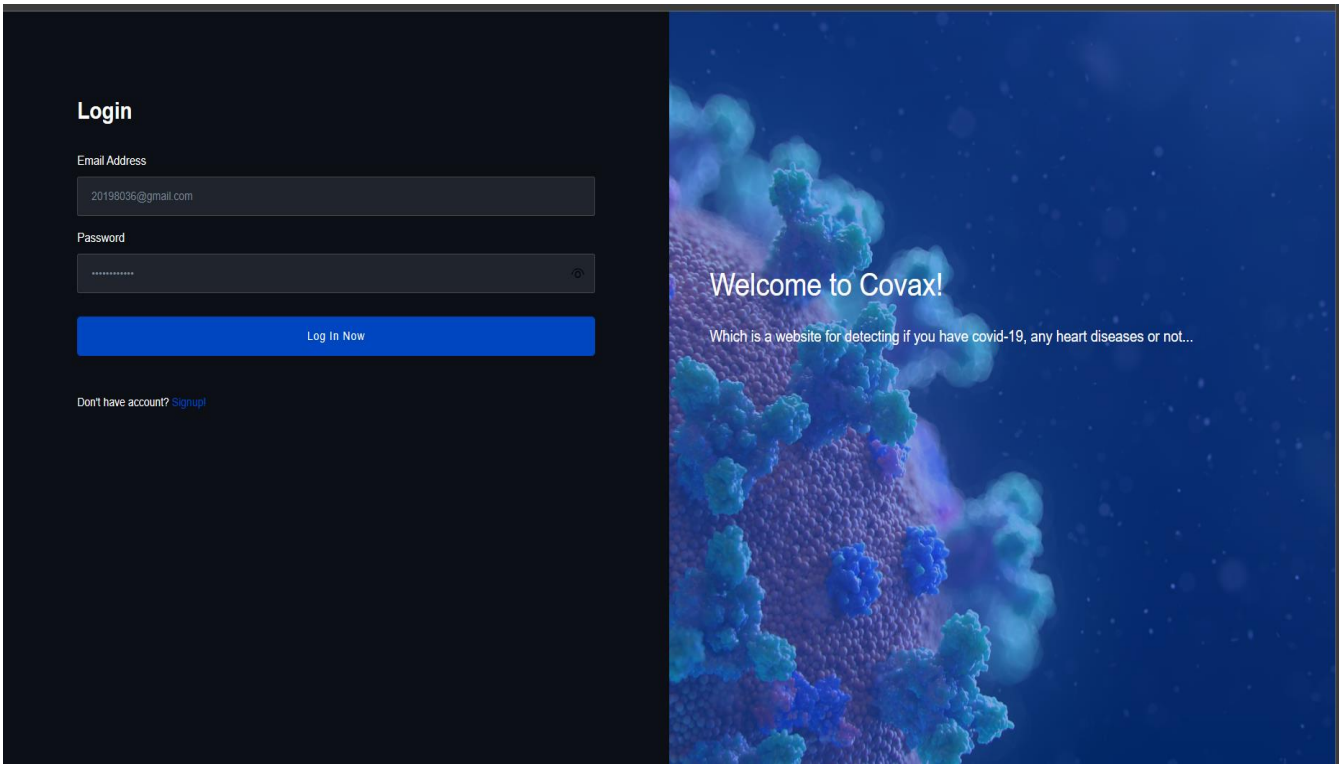


Figure 26 Login page screen

About page:

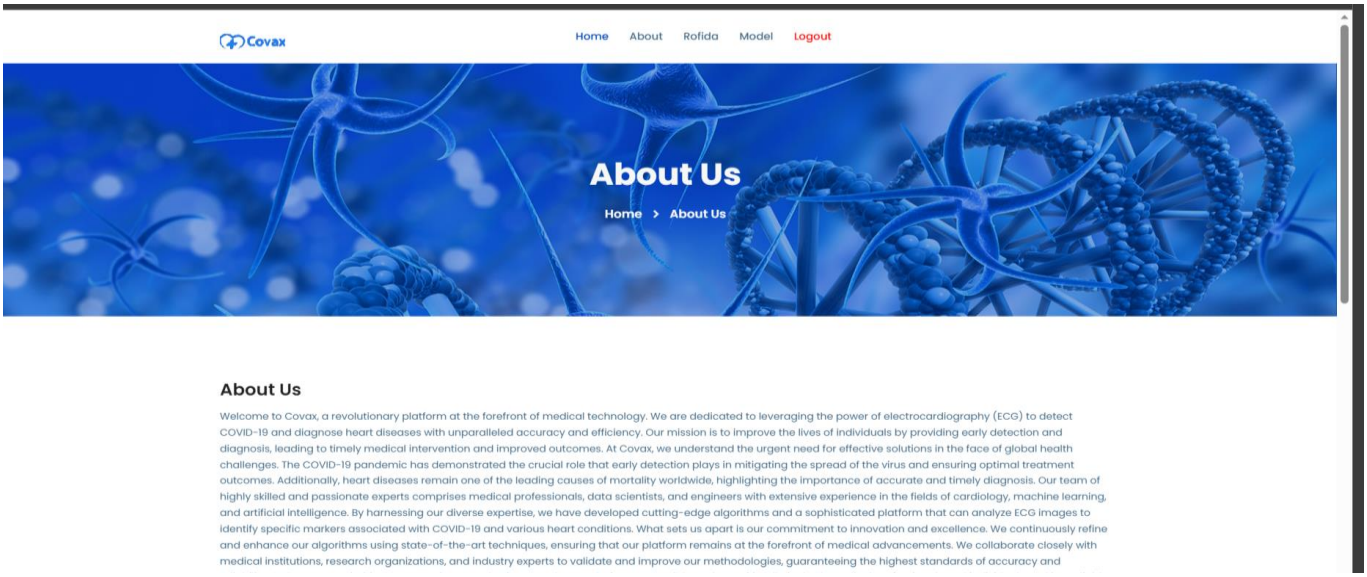


Figure 27 About page

Feedback Form at footer of home and about pages:

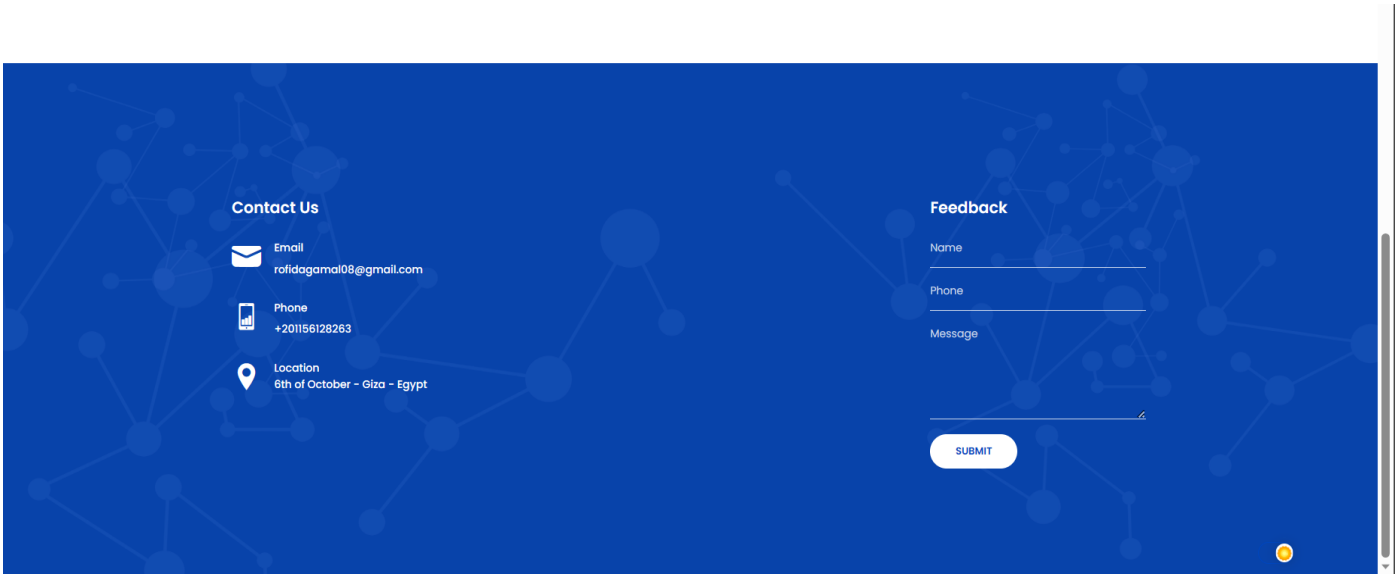


Figure 28 Feedback page

Model Page before selecting image:

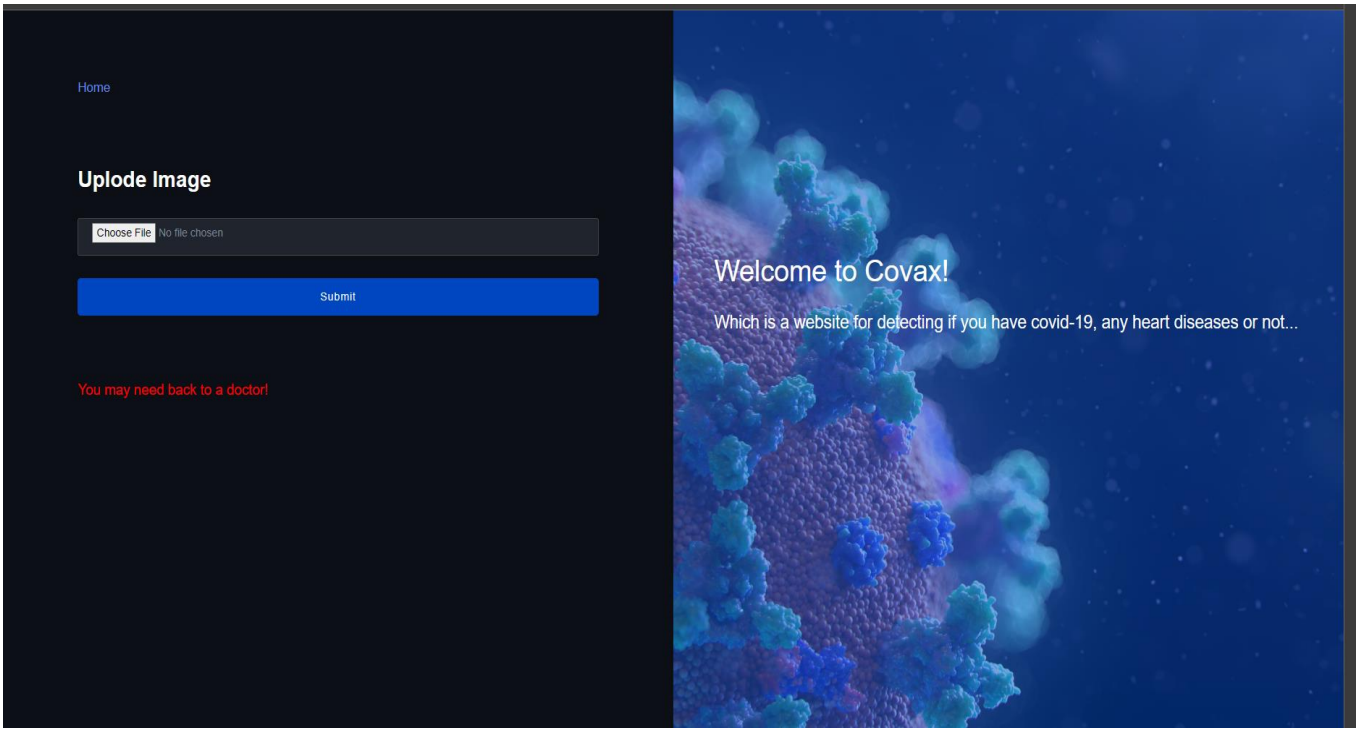
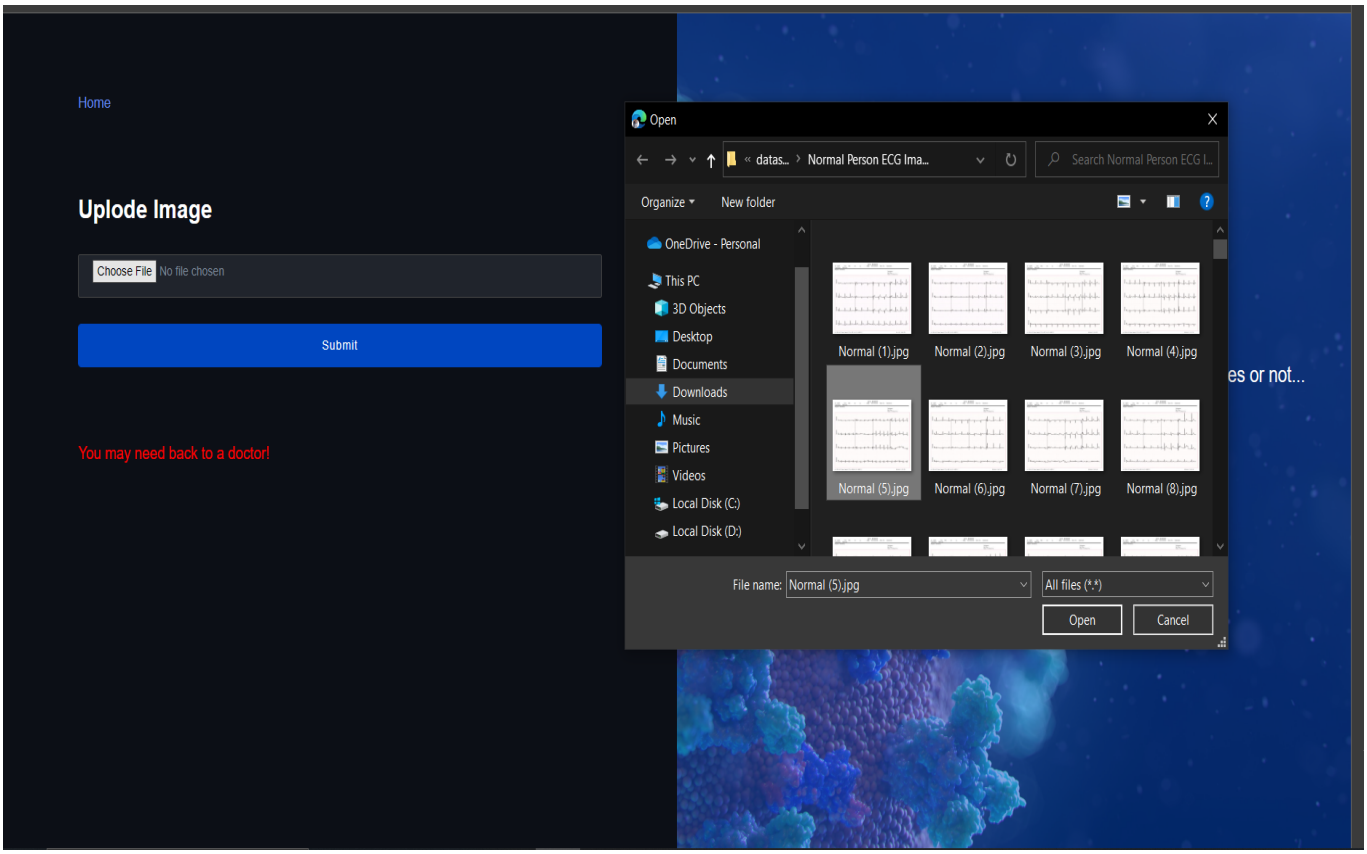
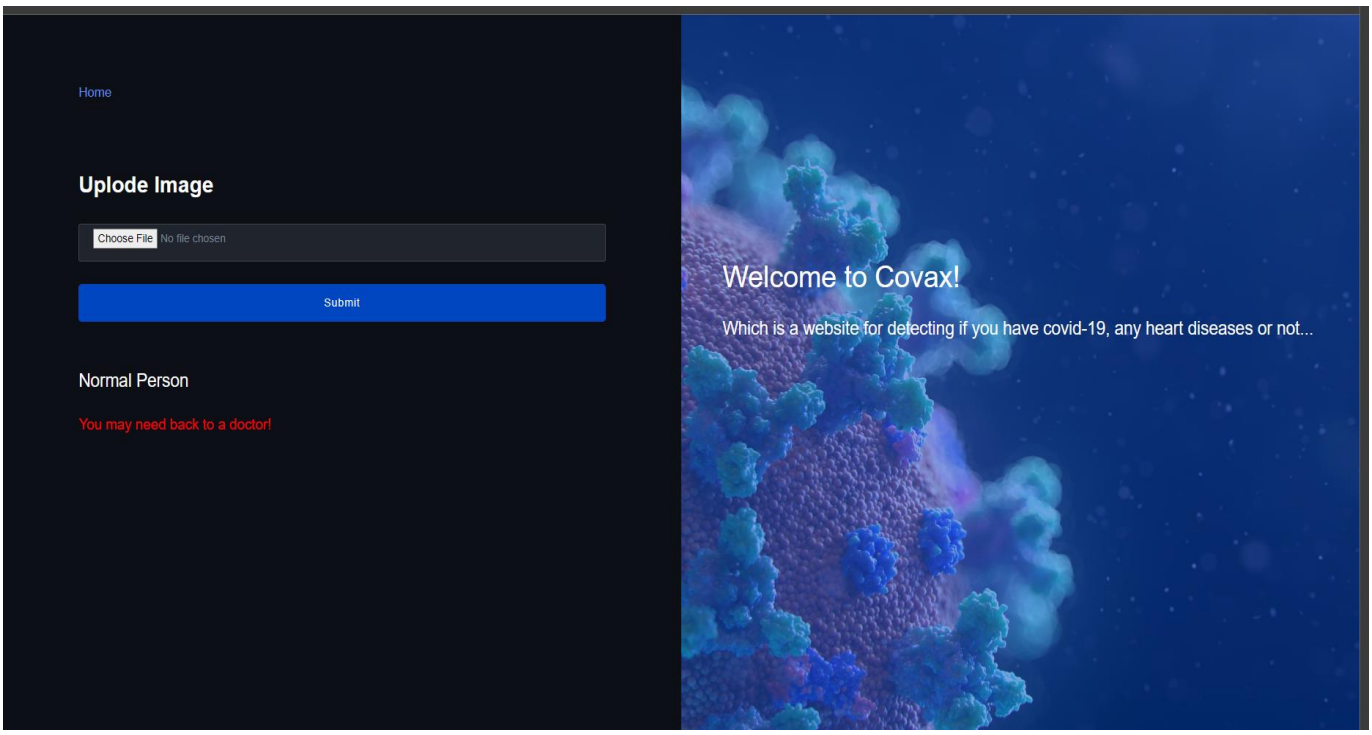


Figure 29 Model page for uploading image

Model page after selecting image and upload it to the model:





Database after inserting user at Users table:

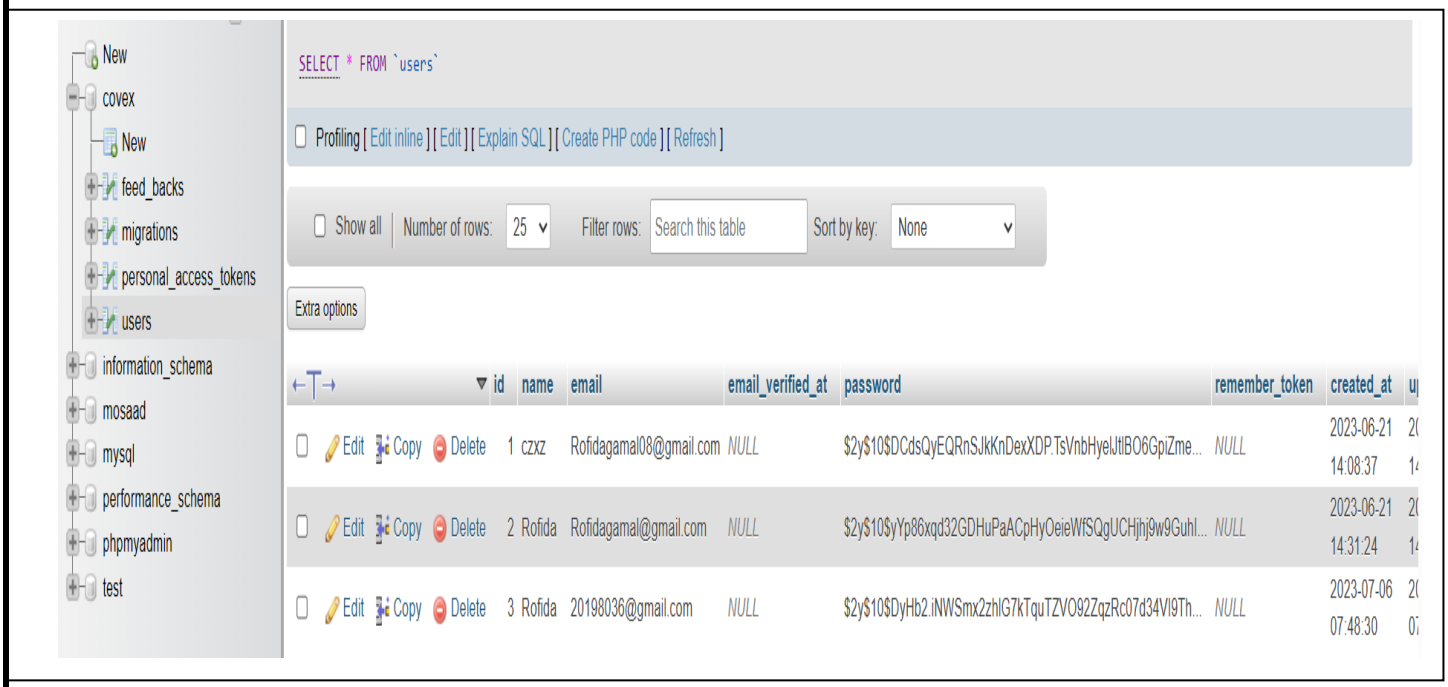


Figure 30 Database after inserting user data

Database after inserting feedback:

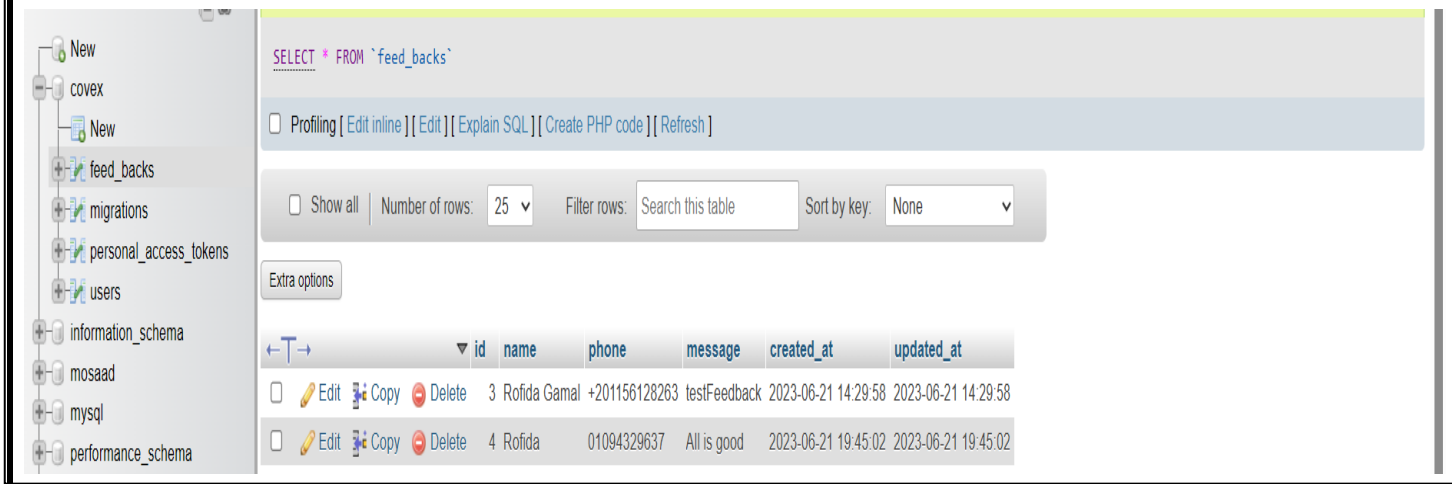


Figure 31 Database after inserting feedback

COVAX mobile App

Sign up:

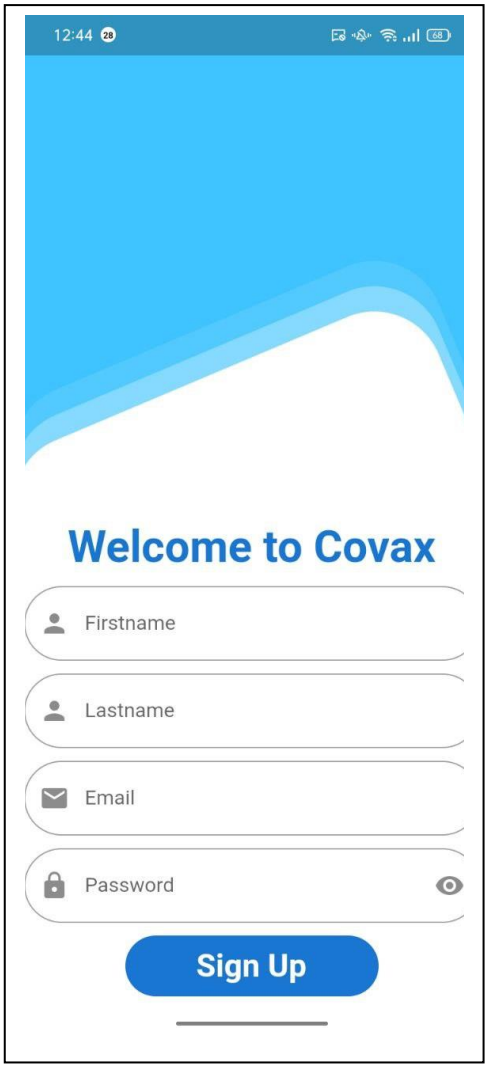


Figure 32 Sign up

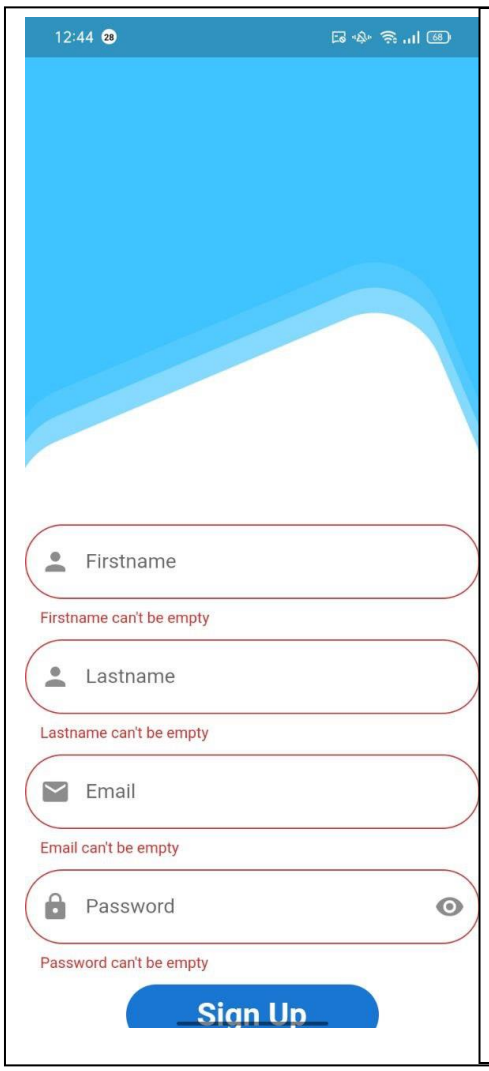


Figure 33 Sign up validation

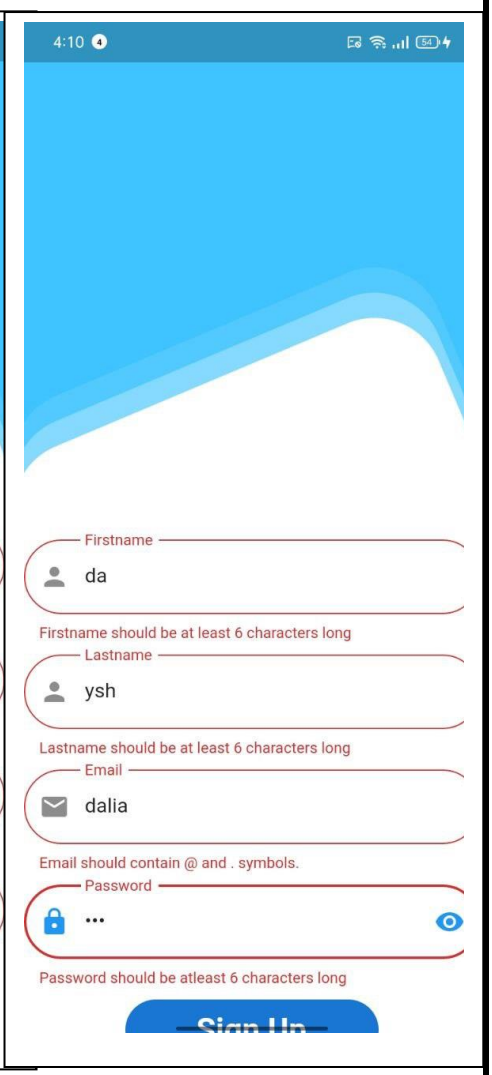
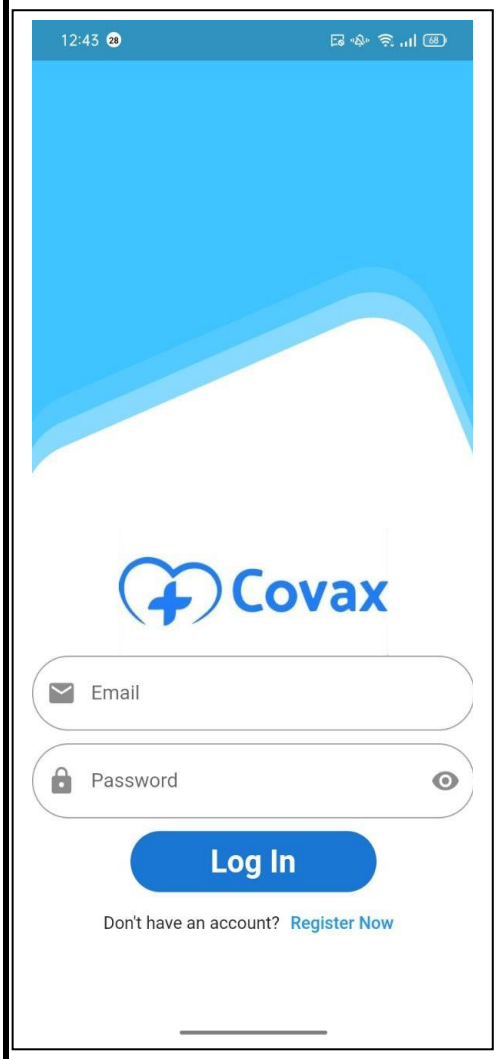
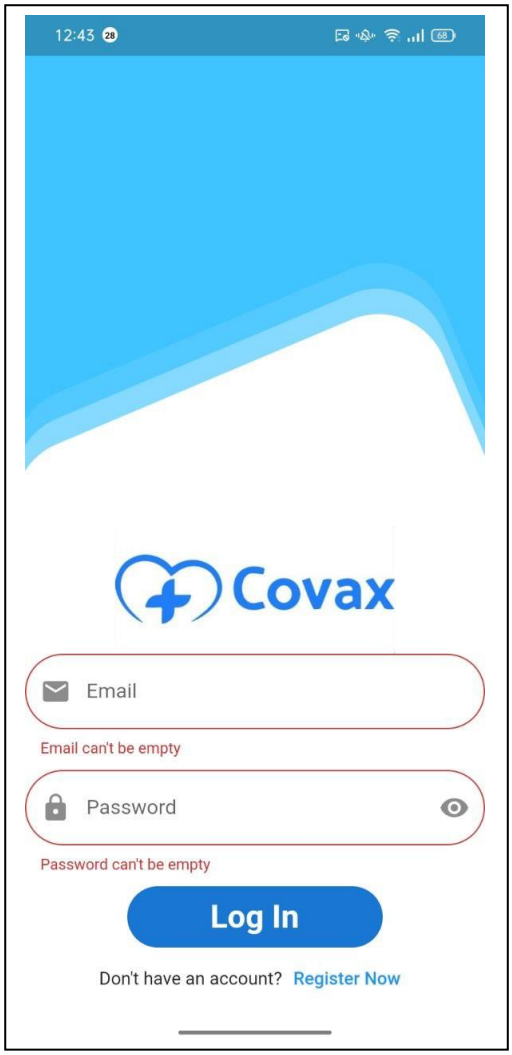


Figure 33.1 Validation

Login:





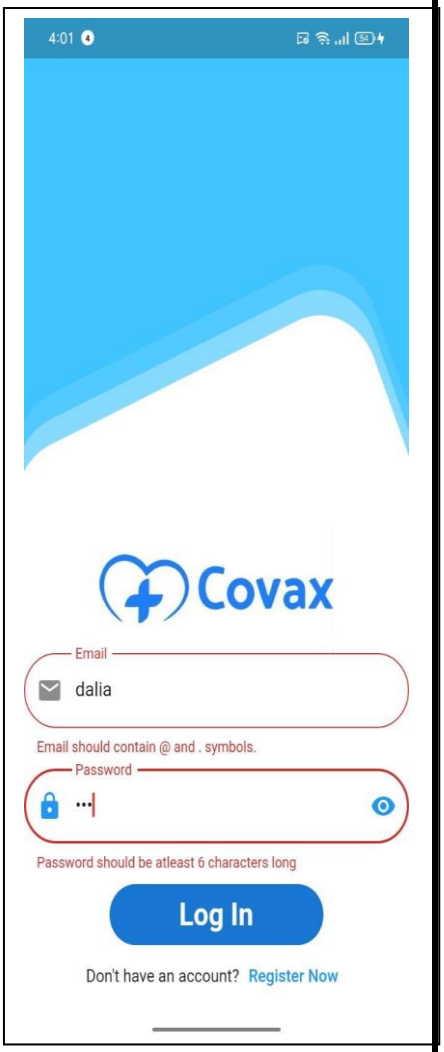


Figure 34 Login Page.

Figure 35 Login Validation

Figure 35.1 Validation

Model Page:

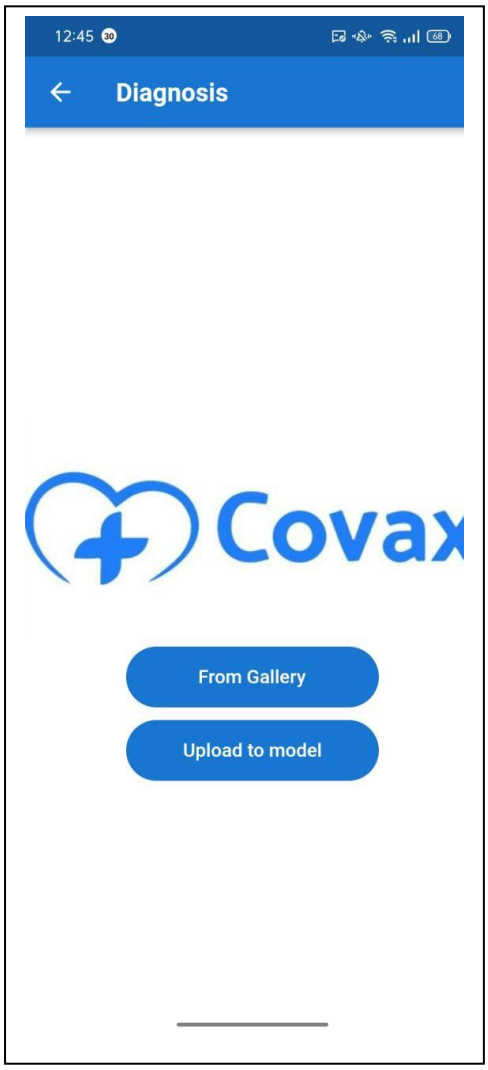


Figure 36 Model Page

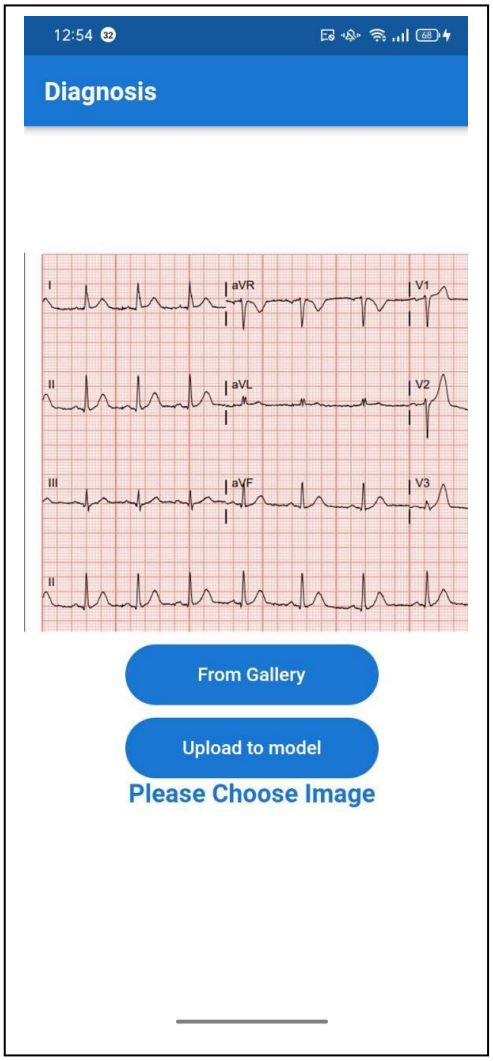


Figure 37 Example of uploading image

Test Cases:

The figures below show the result of the model when it was tested on different cases from the 5 classes we mentioned before.

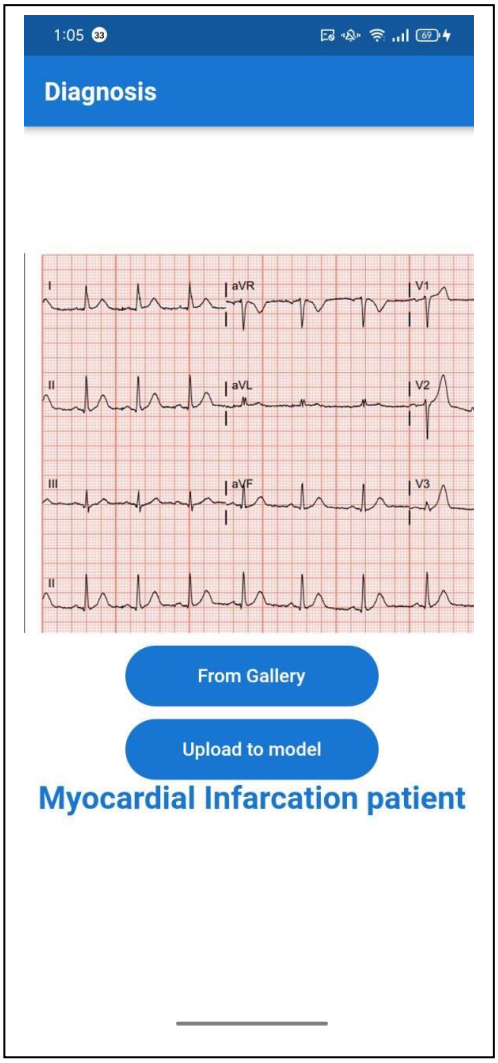


Figure 38 Myocardial Infarction case.

Figure 39 COVID-19 case.

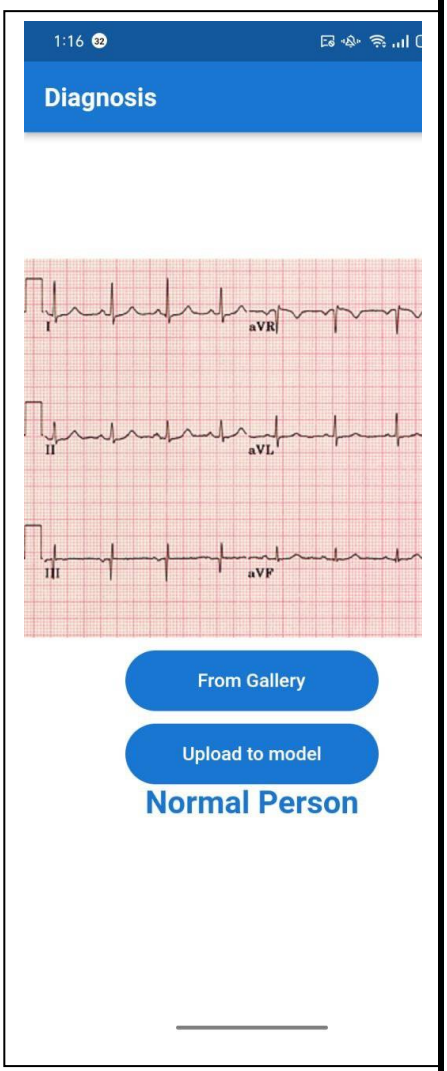
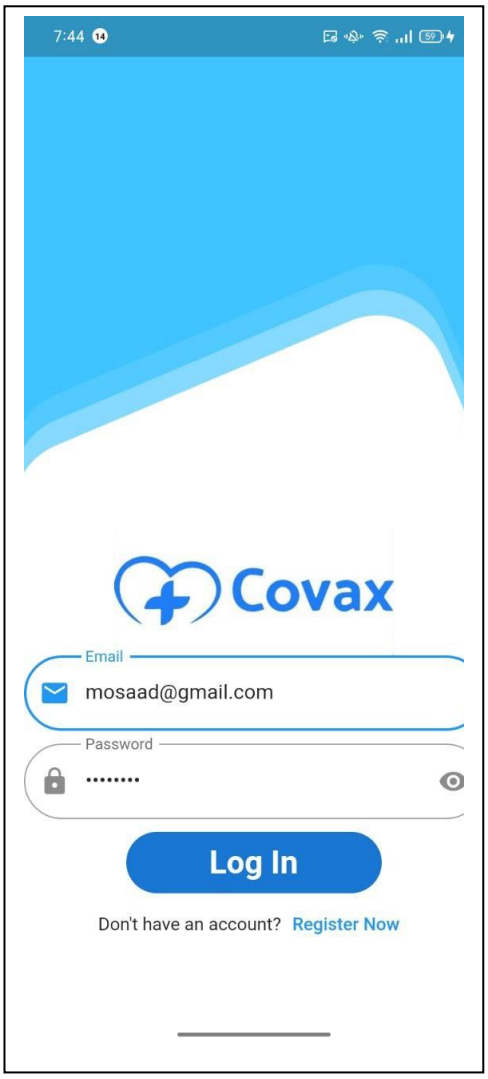


Figure 40 Normal.

The user tries to login while he/she doesn't even have an account



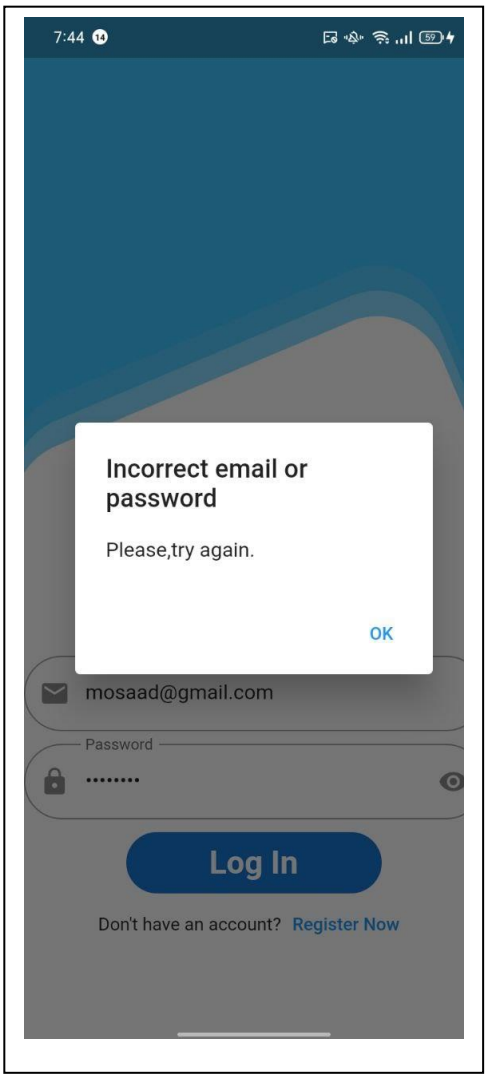


Figure 41 Login without having an account before. *Figure 42 Authentication Error.*

The user enters w wrong password during login

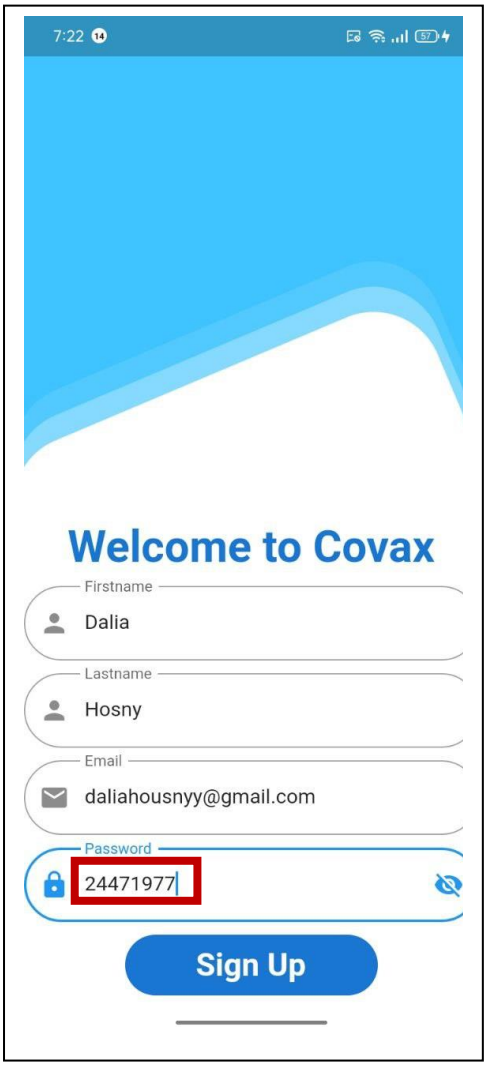


Figure 43 Sign up

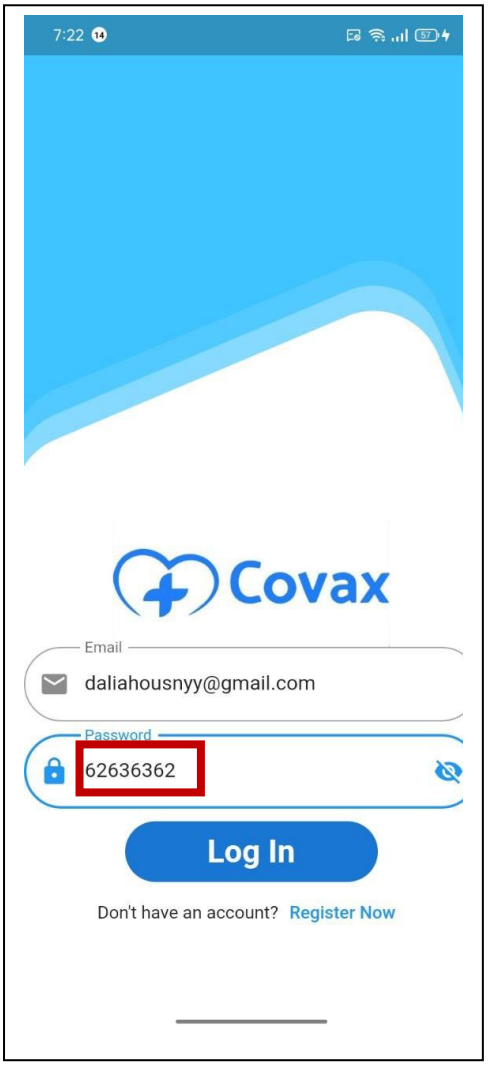


Figure 44 Login with wrong password

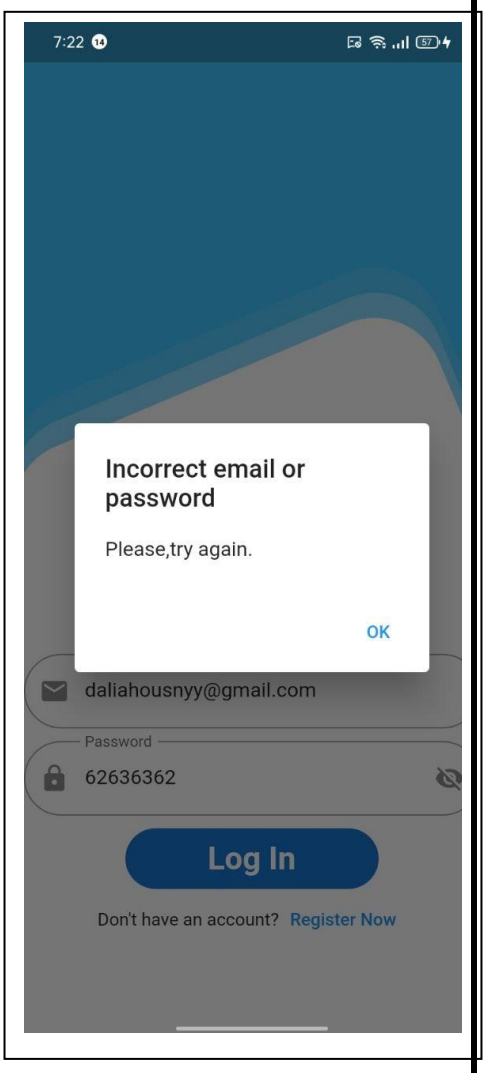
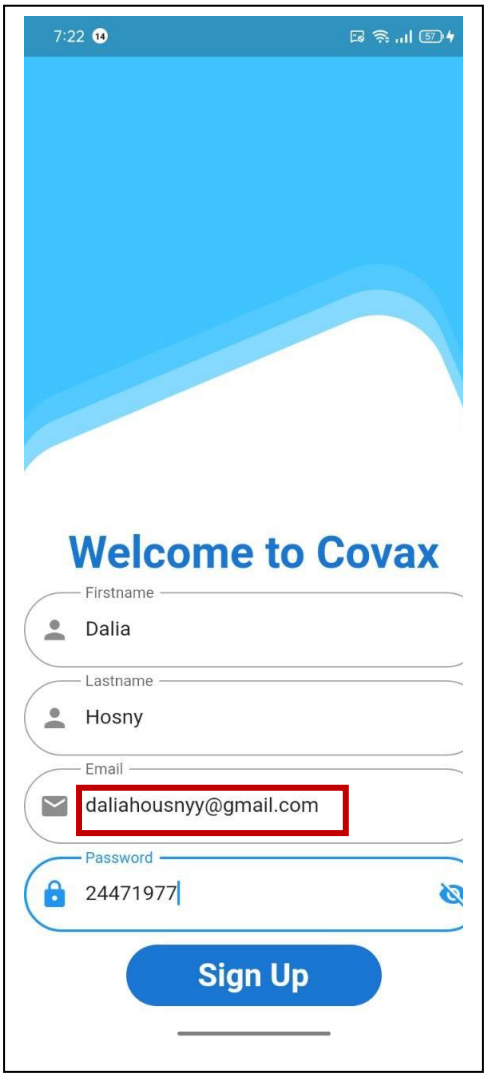
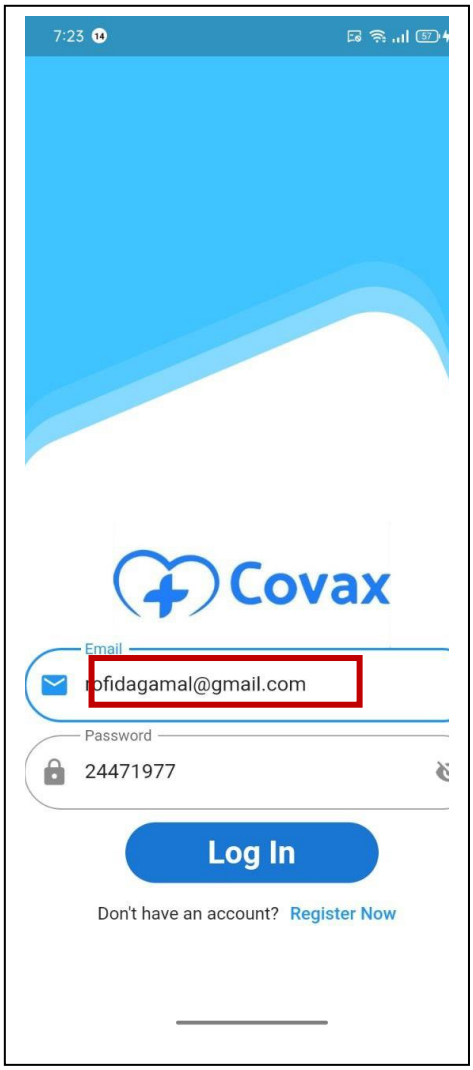


Figure 45 Authentication Error

The user enters an incorrect email.





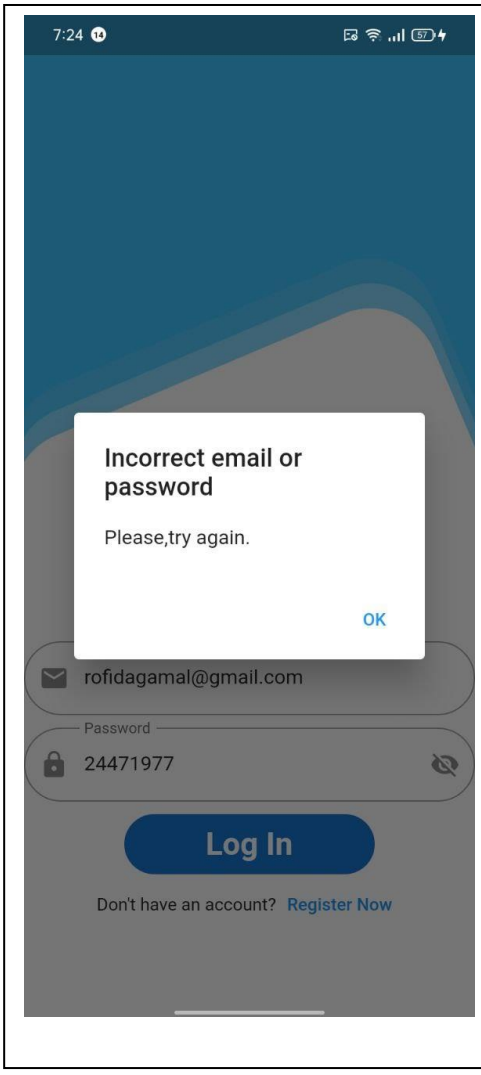


Figure 46 Sign up.

Figure 47 Login with wrong email

Figure 48 Authentication Error

Appendix:

- **Rectangle Detection function:**

```
def biggestContour(contours):
    biggest = np.array([])
    max_area = 0
    for i in contours:
        area = cv2.contourArea(i)
        if area > 5000:
            peri = cv2.arcLength(i, True)
            approx = cv2.approxPolyDP(i, 0.02 * peri, True)
            if area > max_area and len(approx) == 4:
                biggest = approx
                max_area = area
    return biggest, max_area
```

Figure 49 Biggest Contour function

- **Gamma correction:**

```
def adjust_gamma(image, gamma=1.0):
    invGamma = 1.0 / gamma
    table = np.array([((i / 255.0) ** invGamma) * 255
    for i in np.arange(0, 256)]).astype("uint8")
    return cv2.LUT(image, table)
```

Figure 50 Gamma correction function

- **Register Handling function:**

```php
public function handleRegister(Request $request)
{
    $request->validate([
        'name'=>'required|string|max:100',
        'email'=>'required|email|max:100 |unique:users',
        'password'=>'required|string|max:50|min:8'
                    ]);


    $user = User::create([
        'name'=> $request->name,
        'email'=> $request->email,
        'password'=>Hash::make ($request->password),
                    ]);

    Auth::login($user);
    return redirect (route('index'));

}
```

Figure 51 Registeration Validation code

- **Login Handling function:**

```php
public function handleLogin(Request $request)
{
    $request->validate([
        'email' => 'required|email|max:100',
        'password' => 'required|string|max:50|min:8'
    ]);
    $is_login = Auth::attempt(['email' => $request->email, 'password' => $request->password]);
    if (!$is_login) {
        return back();
    }
    return redirect(url('/'));
}
```

Figure 52 Login validation and authentication code.

- **Preprocessing function:**

```python
def preprocessing(image):
    img=Image.open(image)
    img=np.array(img)
    img = Image.fromarray(img)
    img.save('image_gray.png')
    img=cv2.imread('image_gray.png')
    img=procssing(img)
    img=adjust_gamma(img,1.3)
    img.shape=(1,224,224,1)
    img_arr=np.array(img)
    return img_arr
```

Figure 53 image preprocessing before sending it to efficientnet model

References

[1] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[2] Tan, M., & Le, Q. V. (2019). Efficient-Net: Rethinking model scaling for convolutional neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (pp. 6105-6114).

[3] Jobson, D. J., Rahman, Z. U., & Woodell, G. A. (1997). A multiscale retinex for bridging the gap between color images and the human observation of scenes. IEEE Transactions on Image Processing, 6(7), 965-976.

[4] [dataset] A. Haider Khan, M. Hussain, ECG Images dataset of Cardiac and COVID-19 Patients, Mendeley Data, v1, 2020. http://dx.doi.org/10.17632/gwbz3fsgp8.1

[5] Taylor, O. (2011). Laravel Documentation. Retrieved fromhttps://laravel.com/docs.

[6] Mahmoud M. Bassiouni, Islam Hegazy, Nouhad Rizk, El-Sayed A. El-Dahshan & Abdelbadeeh M. Salem. Automated Detection of COVID-19 Using Deep Learning Approaches with Paper-Based ECG Reports.

[7] Author links open overlay panelKunwar Prashant , Prakash Choudhary , Tarun Agrawal , Evam Kaushik. OWAE-Net: COVID-19 detection from ECG images using deep learning and optimized weighted average ensemble technique.

[8] COV-ECGNET: COVID-19 detection using ECG trace images with deep convolutional neural network. Tawsifur Rahman1 , Alex Akinbi2 , Muhammad E. H. Chowdhury1* , Tarik A. Rashid3 , Abdulkadir Şengür4 , Amith Khandakar1 , Khandaker Reajul Islam1 , Aras M. Ismael5*