Palestine Technical University – Kadoorie

College of Engineering and Technology

Department of Computer Systems Engineering

Project title:

DEVELOPING AN ECG ANALYZER SYSTEM USING AI IN PALESTINE

Prepared By:

Arwa Abdelal – 202010554

Heba Hamdan – 202010678

Jana Qabajah – 202010125

Supervised by:

Dr. Thaer Sammar

A graduation project submitted in partial fulfilment of the requirements for the bachelor's degree in computer systems engineering.

Tulkarm, Palestine

30 June 2024

ACKNOWLEDGEMENT

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

الحَمْدُ لِلَّهِ الَّذِي مَا سَلَكْنَا البِدَايَاتِ إِلَّا بِتَيْسِيرِهِ، وَمَا بَلَغْنَا النِّهَايَاتِ إِلَّا بِتَوْفِيقِهِ، وَمَا حَقَّقْنَا الغَايَاتِ إِلَّا بِفَضْلِهِ، وَإِلَيْهِ يُنْسَبُ الفَضْلُ وَالكَمَالُ وَالإِكْمَال.

وَسَطَ أَحْدَاثٍ أَلِيمَةٍ وَقَعَتْ بِأَبْنَاءِ شَعْبِنَا وَمِنْ قَلْبِ اقْتِحَامَاتٍ مُتَكَرِّرَةٍ، لَمْ يَكُنْ سَهْلًا عَلَيْنَا أَنْ نُكْمِلَ مَسِيرَتَنَا وَأَنْ نَصِلَ مُبْتَغَانَا، فَهَا نَحْنُ اليَوْمَ نَقِفُ رَافِعِينَ الرَّأْسَ رَحْمَةً وَإِجْلَالًا لِشُهَدَائِنَا الأَبْرَارِ وَفَخْرًا لِأَسْرَانَا البَوَاسِلِ، وَمَا نَحْنُ اليَوْمَ إِلَّا رَغْبَةً مِنَّا لِإِثْبَاتِ فِلَسْطِينِيَّتَنَا وَتَجَذُّرَنَا.

نُهْدِي تَخَرُّجَنَا وَثَمَرَةَ جُهْدِنَا إِلَى مَنْ مَهَّدَتْ لَنَا طَرِيقَ العِلْمِ وَاحْتَوَتْ أَحْلَامَنَا وَكَانَتْ مَصْدَرَ قُوَّتِنَا بِدَعَوَاتِهَا... إِلَى أُمَّهَاتِنَا الغَالِيَات.

إِلَى مُلْهِمِنَا وَدَاعِمِنَا وَرَمْزُ العَطَاءِ الأَوَّلِ وَمَصْدَرُ فَخْرِنَا، لِمَنْ أَضَاءَ دُرُوبَنَا... إِلَى آبَائِنَا.

إِلَى سَنَدِنَا وَضِلْعِنَا الثَّابِتِ، لِمَنْ رَسَمُوا مَعَنَا المُسْتَقْبَلَ وَآمَنُوا بِقُدُرَاتِنَا وَانْتَظَرُوا هَذِهِ اللَّحْظَةَ لِيَفْتَخِرُوا بِنَا كَمَا نَفْتَخِرُ بِهِمْ... إِلَى إِخْوَانِنَا وَأَخَوَاتِنَا.

نَتَقَدَّمُ بِالشُّكْرِ الجَزِيلِ لِلَّذِينَ وَقَفُوا مَعَنَا بِمَسِيرَتِنَا الأَكَادِيمِيَّةِ وَلَمْ يَكِلُّوا مِنْ إِرْشَادِنَا وَإِعْطَاءِ تَوْجِيهَاتِهِمُ القَيِّمَةِ... إِلَى مُعَلِّمِينَا وَأَسَاتِذَتِنَا الكِرَامِ وَنَخُصُّ بِالذِّكْرِ الدُّكْتُور المُشْرِف ثَائِر سَمَّار، جَزَاكُمُ اللَّهُ خَيْرَ الجَزَاء.

لِمَنْ كَانُوا عَوْنًا لَنَا فِي رِحْلَتِنَا الأَكَادِيمِيَّةِ وَغَمَرُونَا بِالحُبِّ، إِلَى رِفَاقِ الخُطْوَةِ الأُولَى وَالخُطْوَةِ مَا قَبْلَ الأَخِيرَةِ... إِلَى صَدِيقَاتِنَا.

ABSTRACT

Heart disease rates are escalating around the world, especially in Palestine (Husseini et al., 2009). These diseases, which include heart attacks, strokes, and various forms of heart conditions, present a growing health challenge. In this context, the analysis of electrocardiograms (ECG) assumes a critical role, especially in diagnosing arrhythmias that can have serious repercussions on a patient's overall health.

Traditional diagnostic methods for assessing ECGs heavily depend on the expertise and subjective interpretation of medical professionals. This reliance can unfortunately lead to diagnostic inaccuracies. The primary types of errors involved either the failure to detect certain cardiac conditions or the incorrect diagnosis of these conditions.

Given these challenges, there is a pressing need for more reliable and precise diagnostic tools. This project aims to address this gap by exploring the application of artificial intelligence in the realm of cardiology. Specifically, it investigates the feasibility of utilizing resNet50 algorithm for the automatic detection of heart problems through the analysis of ECG images. The goal is to reduce the reliance on subjective interpretation, decrease error rates, and enhance the accuracy and efficiency of cardiac diagnostics. This could represent a significant advancement in medical technology, offering a more robust and dependable tool for healthcare professionals in diagnosing heart-related ailments.

The project is also directed to medical students who have difficulty interpreting ECG images, providing them with an accessible and reliable means to understand and analyze ECG data more accurately.

Table of Contents

<u>List of Figures</u>

List of Tables

CHAPTER 1
INTRODUCTION

1.1 Overview

Globally, heart disease rates are on the rise. Central to diagnosing cardiac health issues, particularly irregular heartbeats (Dattani et al., 2023), is the analysis of electrocardiograms (ECG). An ECG is a test that records the timing and strength of the electrical signals that produce heartbeats. Physicians review these readings to gather information about heart rhythms and to detect any disturbances in their patterns. Research highlights the occurrence of misdiagnoses and failures to detect conditions in ECG readings by human doctors (Cook et al., 2020; Amini et al., 2022). In response to this challenge, this project proposes the use of artificial intelligence-deep learning to detect heart issues, aiming to increase the precision and reliability of cardiac diagnostics.

1.2 Problem Statement

As mentioned briefly in the introduction, in the face of an escalating tide of heart disease, particularly in Palestine, accurate and timely diagnosis plays a crucial role in securing a patient's well-being. Yet, traditional methods often rely on subjective interpretations of electrocardiograms (ECGs) by medical professionals, a process susceptible to human error and potentially leading to misdiagnoses and delayed interventions. This vulnerability in the existing system poses a significant threat to patient health and outcomes. Biases and inconsistencies can creep in, jeopardizing diagnostic accuracy. Furthermore, resource limitations in certain regions restrict accessibility to specialized cardiologists and sophisticated diagnostic tools, thereby exacerbating the challenges faced by patients in underserved communities (Knudtson et al., 2006).

Adding to these challenges, medical students often face significant difficulties in learning to interpret ECG images accurately. The complexity of ECG patterns and the necessity for detailed understanding make it a formidable task, often leading to inconsistencies in interpretation among novice practitioners. This learning curve can further contribute to diagnostic delays and errors, impacting patient care quality.

By building upon the promising findings of studies which demonstrated the notable accuracy of AI in ECG analysis for heart disease diagnosis (John, 2023), our project aims to develop and implement a robust AI-driven system for improved cardiac care. This system will not only enhance diagnostic accuracy and consistency but also serve as a valuable educational tool for medical students, ensuring earlier detection, better treatment outcomes, and ultimately, saving lives.

1.3 Objectives

1.3.1 General Objective

The general objective of this project is to design and implement a website using AI, called "ECG analyzer" to assist medical students in improving their skills to easily interpret ECG images. Additionally, it aims to aid healthcare professionals by providing accurate ECG analysis efficiently. This project employs deep learning techniques, specifically ResNet50 architecture, to achieve high accuracy in ECG image analysis.

1.3.2 Specific Objectives

- Reduce misdiagnosis of heart disease and avoid the consequences resulting from it.
- Saving time for doctors by giving accurate results with a suitable diagnosis.
- Providing an accurate and quick scientific reference supported by AI for medical specialization students and improve the learning curve.
- Providing an interactive learning website for medical image diagnosis.
- Providing a user-friendly and flexible interface that everyone can deal with.
- Providing a system that medical students and healthcare professionals can easily access from everywhere.

1.4 Dataset

In this section, we discuss the two options of datasets considered for our project and the rationale behind our selection.

1.4.1 ECG Image Dataset 1

MIT-BIH Arrhythmia Dataset and PTB Diagnostic ECG Dataset (Kachuee et al., 2018).

The first dataset option comprises two collections of heartbeat signals sourced from the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. This dataset has been extensively used in previous studies for exploring heartbeat classification using deep neural network architectures and investigating the effectiveness of transfer learning techniques. The dataset consists of electrocardiogram (ECG) shapes representing both normal heartbeats and those affected by various arrhythmias and myocardial infarction. Each signal is preprocessed and segmented, with segments corresponding to individual heartbeats.

The dataset split into 80% for training and 20% for testing.

It is characterized by it is Size, The dataset consists of a large number of images, totaling 123,998, categorized into five classes: Normal beat, Supraventricular premature beat, Premature ventricular contraction, Fusion of ventricular and normal beat, and Unclassifiable beat, including myocardial infarction cases.

The disadvantage of this dataset that despite its extensive size and suitability for training deep neural networks, this dataset was not chosen due to the short-term nature of its images, which could lead to unrealistic model training and potentially limit its generalization capabilities.

1.4.2 ECG Image Dataset 2

Ch. Pervaiz Elahi Institute of Cardiology Multan Dataset (Khan& Hussain, 2021).
Which we have adopted in training our model.

The second dataset option comprises ECG images of cardiac patients collected under the auspices of the Ch. Pervaiz Elahi Institute of Cardiology Multan, Pakistan. This dataset aims to facilitate research on cardiovascular diseases within the scientific community.

The disadvantage of this dataset that it is size is small , it’s contains a total of 928 images categorized into four classes: ECG Images of Myocardial Infarction Patients, ECG Images of Patients with Abnormal Heartbeat, ECG Images of Patients with a History of MI (Myocardial Infarction), and Normal Person ECG Images.

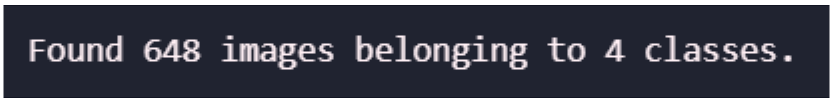
Table 1: Number of images of each classification in ECG Image Dataset 2

Name of classification	Number of classification
ECG Images of Myocardial Infarction	239 images
ECG Images of Patient that have abnormal heartbeat	233 images
ECG Images of Patient that have History of MI	172 images
Normal Person ECG Images	284 images

Although smaller in size compared to the first option, this dataset is characterized by its long-term pulse images, reflecting realistic ECG patterns akin to those found in clinical reports.

Prior to training the AI models, the ECG images underwent preprocessing steps to enhance their suitability for machine learning algorithms. We divided it into

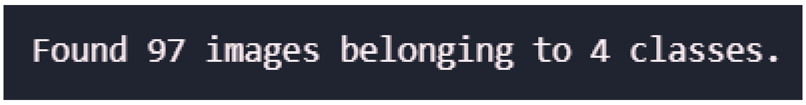
70% for training:



```
Found 648 images belonging to 4 classes.
```

Figure 1: The total number of images in the training set

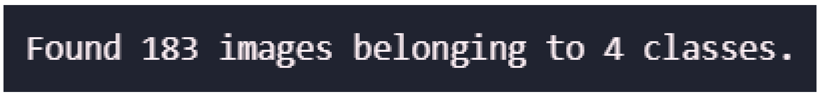
10% for validation



```
Found 97 images belonging to 4 classes.
```

Figure 2: The total number of images in the validation set

20% for testing



```
Found 183 images belonging to 4 classes.
```

Figure 3: The total number of images in the testing set

And we did Common preprocessing techniques such as normalization, resizing, and augmentation were applied to standardize the input data and augment the dataset to mitigate overfitting.

Given its realism and alignment with real-world ECG patterns, we opted for this dataset. However, it's important to note that utilizing a long-term dataset may pose challenges in recognizing short-term image patterns, potentially necessitating strategies to enhance **model generalization**.

CHAPTER 2
RELATED WORKS

2.1 Related projects

2.1.1 PMcardio-ECG Analysis

PMcardio represents a significant advancement in medical technology, offering healthcare professionals the ability to swiftly and accurately interpret ECGs, make informed decisions regarding patient management, and function effectively as a virtual doctor. However, its usability poses a significant challenge. The application's interface may be perceived as complex or difficult to navigate by some users, potentially hindering its widespread adoption and effectiveness in real-world clinical settings. Addressing this issue through comprehensive user interface improvements and enhanced usability features is crucial to improve cardiovascular care and maximize the utility of PMcardio as an indispensable tool for healthcare professionals. [9]

2.1.2 ECG+ | Analyzer for QTc & HRV

ECG+ is an innovative application for Apple Watch, transcends basic ECG readings and empowers researchers with advanced analysis of critical intervals and heart rate variability (HRV) metrics. Offering in-depth corrected QT -measurement of the time it takes for the ventricles of the heart to contract and relax- (QTc) analysis with customizable formula selection, ECG+ facilitates precise assessment of ventricular repolarization, a crucial marker for potentially life-threatening arrhythmias.

The intuitive interface coupled with seamless integration into Apple Health empowers researchers to efficiently gather and analyze large datasets, paving the way for groundbreaking discoveries in the field of cardiology.

Despite its innovative features and capabilities, one notable limitation of ECG+ is its platform exclusivity to Apple Watch. This restriction may hinder researchers who do not have access to Apple devices or who prefer alternative platforms. Building a website for classifying ECG images could help address this limitation by providing a more universally accessible platform for researchers to analyze and classify ECG data, irrespective of their device preferences. By offering a web-based solution, researchers from diverse backgrounds can contribute to and benefit from the classification of ECG images, ultimately fostering collaboration and advancing our understanding of cardiovascular health. [10]

2.1.3 ECG for Doctors

App simplifies electrocardiogram reading by providing a comprehensive collection of over 5 million fully interpreted ECG cases, fostering essential medical knowledge and honing ECG reading skills. Catering to medical professionals, students, and paramedics, the app seamlessly integrates a diverse range of real 12-lead ECG cases, each identified with a concise description. Users can choose between Play mode for real-time telemetry display and Read mode for printed ECG tracings, enhancing their interpretative capabilities. With an intuitive design, ECG Pro has become a valuable reference for over 5 million users, including medical students, teachers, healthcare providers, physicians, practitioners, and nurses.

The ECG Pro app lacks interactive features for users to actively practice ECG interpretation, potentially limiting skill development compared to traditional methods. [11]

2.2 Deep learning models

In our project, we utilized two powerful deep learning models: a Convolutional Neural Network (CNN) and ResNet50, to classify ECG (electrocardiogram) images. These models were integral to achieving our objectives, providing robust capabilities for feature extraction and classification of ECG patterns.

2.2.1 Custom CNN model

A Convolutional Neural Network (CNN) is a powerful type of Deep Learning architecture frequently applied in Computer Vision, a branch of Artificial Intelligence focused on interpreting visual data like images. Convolutional Neural Networks are particularly effective for image classification tasks (Alzubaidi et al., 2021).

CNN architecture

A typical Convolutional Neural Network architecture is made up of three main components: the input layer, the hidden layers, and the output layer.

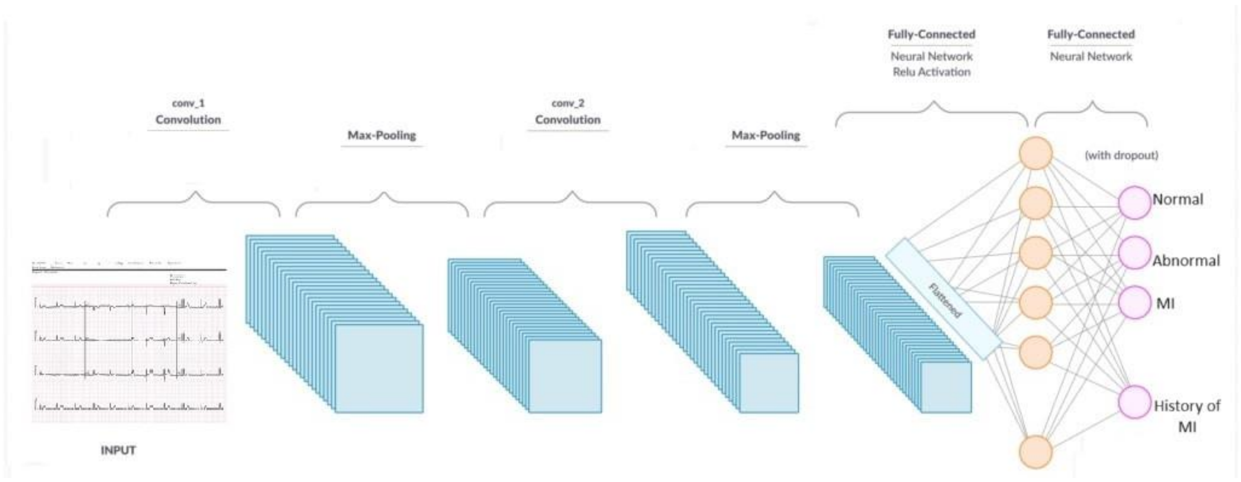


Figure 4: Convolutional Neural Network model architecture

Input Layer: It's the layer in which we give input to passes it to the hidden layers. In CNN, Generally, the input will be an image or a sequence of images.

The hidden layers are the most important part of a CNN architecture, consists of multiple layers, they are:

- Convolutional Layer: it is responsible for extracting features from the input image. It performs a convolution operation on the input image, where a filter or kernel is applied to the image to identify and extract specific features.

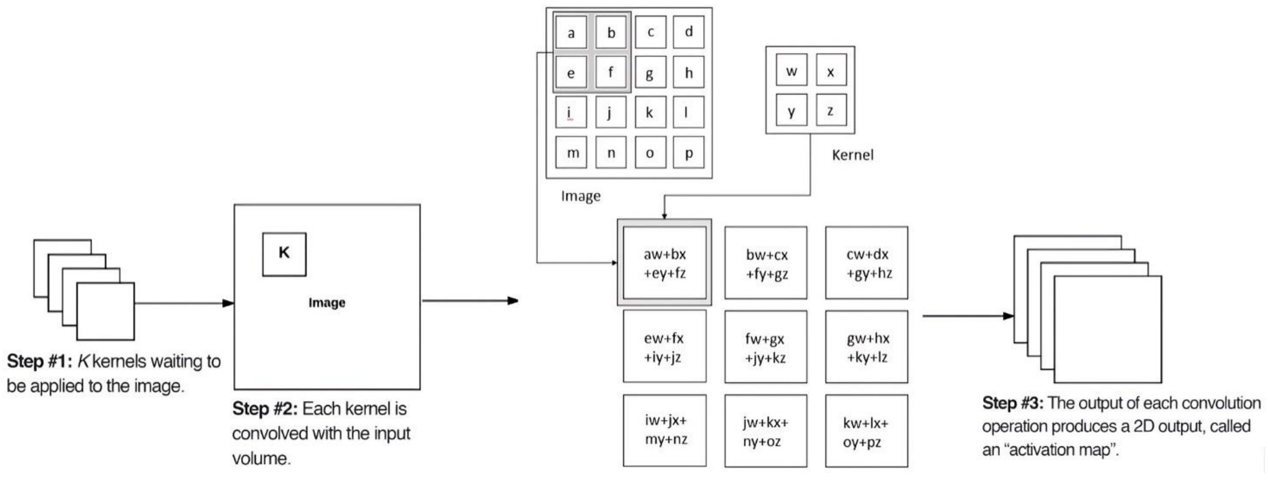

a b c d
e f g h
i j k l
m n o p
Image
w x
y z
Kernel
K
Image
aw+bx +ey+fz
bw+cx +fy+gz
cw+dx +gy+hz
ew+fx +iy+jz
fw+gx +jy+kz
gw+hx +ky+lz
iw+jx+ my+nz
jw+kx+ ny+oz
kw+lx+ oy+pz
Step #1: K kernels waiting to be applied to the image.
Step #2: Each kernel is convolved with the input volume.
Step #3: The output of each convolution operation produces a 2D output, called an "activation map".

Figure 5: Convolutional Layer

- Pooling Layer: it is responsible for reducing the spatial dimensions of the feature maps produced by the convolutional layer. It performs a down-sampling operation to reduce the size of the feature maps and reduce computational complexity.

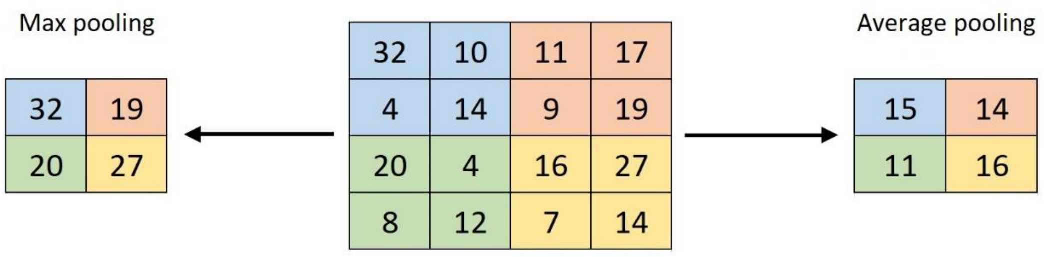


Figure 6: Pooling Layer

- Activation Layer: it is applies a non-linear activation function, such as the ReLU (Rectified Linear Unit) function, to the output of the pooling layer. This function helps to introduce non-linearity into the model, allowing it to learn more complex representations of the input data.

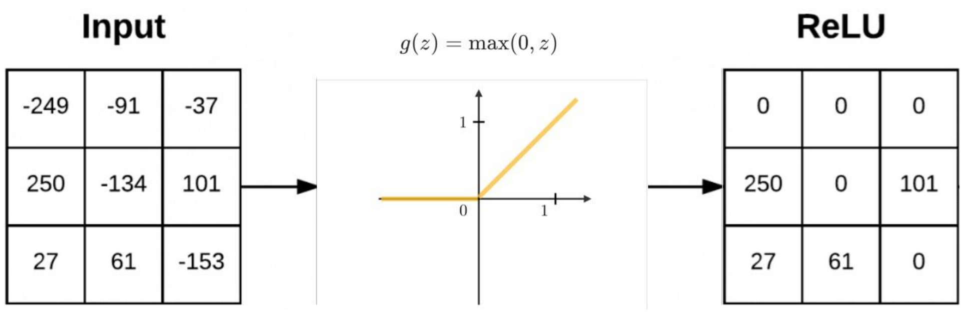


Figure 7: Activation Layer

- Normalization Layer: it is performs normalization operations, such as batch normalization or layer normalization, to ensure that the activations of each layer are well-conditioned and prevent overfitting.

- Dropout Layer: The dropout layer is used to prevent overfitting by randomly dropping out neurons during training. This helps to ensure that the model does not memorize the training data but instead generalizes to new, unseen data.

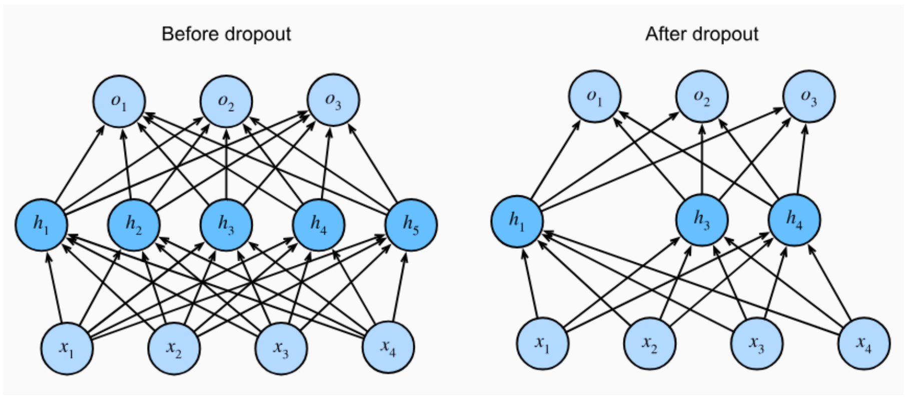


Figure 8: Dropout Layer

- Fully connected layer: also known as Dense Layer, after the convolutional and pooling layers have extracted features from the input image, the fully connected layer can then be used to combine those features and make a final prediction. In a CNN, the fully connected layer is usually the final layer and is used to produce the output predictions. The activations from the previous layers are flattened and passed as inputs to the dense layer, which performs a weighted sum of the inputs and applies an activation function to produce the final output.

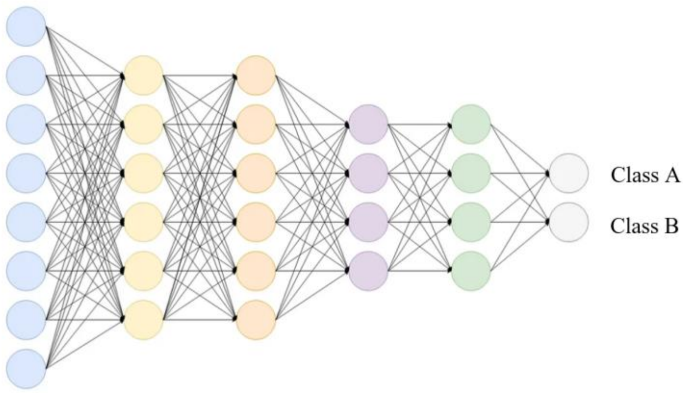


Figure 9: Fully connected layer

The number of hidden layers and the number of filters in each layer can be adjusted to optimize the network’s performance.

The output layer provides the predicted class label or probability scores for each class.

2.2.2 Pre-trained ResNet50 model

Recognizing the potential benefits of leveraging pre-trained models, we also investigated the use of Residual Network ResNet50.

ResNet refers to a specific architecture for a CNN model known for its ability to train deeper models effectively. The 50 indicates the depth of the network, meaning it has 50 layers that has been pre-trained on a large dataset, typically ImageNet for a specific task such as image classification before being made available for further use. During this training process, the model learns to extract features from input images and make predictions about their classes.

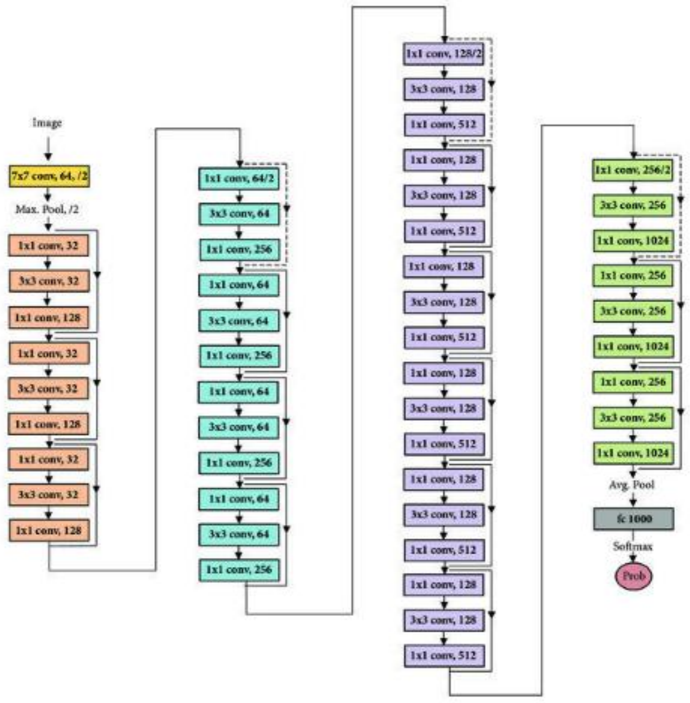


Figure 10: Pre-trained ResNet50 model architecture

We chose the resNet50 model for these reasons:

1. Transfer Learning Capability

ResNet50's architecture is highly compatible with transfer learning, a technique where a model pre-trained on a large dataset (such as ImageNet) is fine-tuned on a smaller, task-specific dataset. Transfer learning leverages the pre-trained weights from large-scale datasets, allowing the model to extract rich feature representations from images. This is particularly advantageous for medical image analysis, where annotated data is often scarce and expensive to obtain (Lavian, 2019).

2. Proven Track Record in Medical Imaging

ResNet50 has demonstrated remarkable success in various medical imaging tasks, including disease detection in radiology and pathology (Yang et al., 2021). Its deep architecture captures intricate patterns and abnormalities in medical images, which are often subtle and complex. This proven efficacy in similar applications underscores its suitability for ECG image classification.

3. Superior Performance on Small Datasets

ResNet50, a 50-layer deep residual network, is particularly well-suited for small datasets. Its residual connections help in training deep networks effectively. This is crucial for small datasets where overfitting is a significant risk. The ability to train a deeper network while avoiding overfitting leads to better generalization on unseen data (Changchong& Weihai, 2018).

4. Balance between Complexity and Performance

ResNet50's architecture strikes an optimal balance between complexity and performance. With 50 layers, it is deep enough to capture intricate patterns in ECG images but not so complex as to require prohibitive computational resources. This balance makes it suitable for deployment in a web-based application where computational efficiency is critical.

CHAPTER 3
SYSTEM REQUIREMENT

3.1 Functional requirements

1. **User registration(Signup)**

    1. Registration Form:
    - The user must fill the registration form which contains username, email, password, and confirm password.
    - The registration form must allow users to register as either a "Medical Student" or a "Healthcare Professional".

    2. Client-Side Validation:
    - Username must be a string with a minimum length of 3 characters and a maximum length of 20 characters.
    - Email must be a valid email format and at least eight characters long.
    - Password must be a string with a minimum length of 8 characters and at least one uppercase letter, one lowercase letter, one digit and one special character.
    - Confirm Password must match the password field.
    - Error Messages:
      -- If the username is empty, return the message "Username is required".
      -- If the username is less than 3 characters, return the message "Username must be at least 3 characters long".
      -- If the username is more than 20 characters, return the message "Username must be at most 20 characters long".
      -- If the email is empty, return the message "Email is required".
      -- If the email is not a valid string, return the message "Please enter a valid email".
      -- If the password is empty, return the message "Password is required".
      -- If the password doesn’t match the specified pattern, return the message "Password must contain at least 8 characters, one uppercase letter, one lowercase letter, one number, and one special character".
      -- If confirm password is empty, return the message "Confirm password is required".
      -- If confirm password mismatch password field, return the message "Confirm password must match the password".

    If all client-side validations pass, a POST HTTP request is sent to the server to perform additional validation checks.

    3. Server-Side Validation:
    - Same as client-side validation for username, email, password, and confirm Password.
    - Error Messages same as client-side.

4. Account Creation:

- If all server-side validations pass, a new account is created as an inactive account.
- The user receives an email at the provided email address.
- The email contains a link to activate the account.

5. Account Activation:

- The user must click the activation link in the email to activate their account. The activation link is valid for 24 hours only.

2. User login

The log in is a gateway for users to access their accounts and interact with the system's features and functionalities.

1. Login Form contains two fields, they are email and password.
2. Client-Side Validation:
   - Ensure both email and password fields are filled.
   - Validate email format.
   - Error Messages:
     -- If the email is empty, return the message "Email is required".
     -- If confirm password is empty, return the message "Confirm password is required".
3. Server-Side Validation:
   - Verify that the provided email exists in the database. If it does not exist, return an error message "data invalid".
   - Check if the provided password matches the corresponding email. If not, return an error message "invalid password".
   - If the user does not confirm their email within 24 hours of registering on the website, return the message "plz confirm your email".
4. Successful Login:
   - Redirect the user to the upload image page.
5. Failed Login process:
   - In case of incorrect information, appropriate error messages are displayed to the user.
   - Provide a way for users to reset their password if they forget it.
6. The system maintains user login sessions even after the browser tab is closed. To achieve this, essential user information are stored in local storage, enabling automatic login when the user returns to the system after closing the browser tab.

3. Upload ECG image

The system enables users to upload one image at a time, supporting file formats such as JPG, JPEG, and PNG. Before sending the image to the model for analysis, users

have the option to preview it and then choose to either confirm the upload or cancel it. During the uploading process, progress bar appear to signify that the system is processing the image. Utilizing AI-deep learning algorithms, the uploaded image undergoes comprehensive analysis.
Upon completion of the analysis, the system displays the medical diagnosis result to the user through a dedicated "Show Result" icon. Clicking on this icon reveals the result in a message box, presenting medical diagnosis.

4. **View medical explanation**
   It is just for students, offering access to four distinct heart diagnoses for educational purposes. These diagnoses are consolidated onto a single page, facilitating effortless navigation and learning. Students can readily access detailed explanations associated with each diagnosis, enriching their understanding of cardiovascular health and providing a centralized platform for studying and referencing medical information.

5. **User Logout**
   - Users have the option to log out at their discretion.
   - The system automatically log out users when their session, which stores their information, expires after 24 hours.

6. **Reset password**
   Users have the ability to reset their password in case they forget it.
   1. The system provides a "Forgot Password" option on the login page.
   2. Upon selecting this option, users are directed to a password reset page where they can enter their email address.
   3. After submitting their email address, users receive a verification code via email. This code is unique and is valid for a single use only. The system ensures that only one active verification code is valid at any given time. Requesting a new code invalidates the previous one.
   4. Users are directed to a page where they can enter the verification code and create a new password.
   5. Once the new password is entered and confirmed, the user's password is successfully reset, granting them access to their account.

3.2 Non-functional requirements

1. **Performance:**
   - Image Analysis Time: It takes at most 18 seconds to analyze the image and show the result accurately and effectively.

- Button Response: There is no noticeable delay when pressing buttons, due to the system's speed.
- System Efficiency: The system works effectively and classifies medical diagnoses with high accuracy.
- Model Accuracy: The accuracy of the AI model in heart diagnosis is exceptionally high, ensuring reliable and precise results.

2. **Reliability:**

- The system is designed for high availability, ensuring that it remains operational and accessible to users at all times.
- Both client-side and server-side validation are implemented to ensure data integrity and reliability.
- Asynchronous handler techniques are used to prevent server failures and improve system robustness.
- Failover mechanisms are implemented to prevent data loss and ensure continuity.

3. **Usability**

- All pages feature responsive user interfaces, allowing users to interact easily and efficiently across various devices. The system is built with media queries that ensure the website fits all devices, regardless of size, providing a seamless experience on smartphones, tablets, laptops, and desktops.
- The website’s color scheme is designed to be comfortable for the eyes, enhancing prolonged usability.
- The signup and login forms are straightforward and user-friendly, facilitating quick and easy access.
- The explanation page for students consolidates all explanations into a single page, making it easy to learn and navigate.
- Icons and buttons throughout the website are intuitive and clearly indicate their functions, ensuring users can understand and use them without confusion.

4. **Accessibility**

In our ECG analyzer website, we leveraged React Helmet to enhance accessibility by dynamically setting page titles and Meta descriptions based on patient data, specifying the website's language, and strategically using ARIA attributes for interactive elements.

5. **Security:**

- Password Security: All passwords are hashed before being stored in the database.
- JWT technology:
  - JWT (JSON Web Token) technology is used for authentication and authorization.
  - JWT tokens are utilized to verify user login sessions and prevent unauthorized data manipulation.
  - Bearer tokens are employed for secure communication between the client and server, ensuring only authorized users can access resources.
- Secure Communication: Secure HTTP (HTTPS) is used for communication between the client and server.
- Data Interchange Format: JSON format is used for data interchange between the client and server.
- Session Expiration: Automatic session expiration after 24 hours and secure password reset mechanisms are in place to protect user accounts.
- Confirmation Email Expiry: Confirmation email links are valid for 24 hours after being sent.
- Security configurations and sensitive information are managed using the ‘dotenv’ package to securely store environment variables.
- Reset Code Usage: Reset codes are valid for single use only. If a new code is requested, the previous code becomes invalid. Once a reset code is used, it cannot be reused.
- Protected Components**:** The system uses protected components to check if the user is authorized to access a page. If the user is not authorized, they are redirected to the login page.

CHAPTER 4
METHODOLOGY AND TECHNOLOGIES

In this chapter, we explore the software development methodology and deep learning models used in our project to analyze ECG images through a website. We adopted an agile development approach to ensure flexibility and iterative progress. Central to our approach are deep learning models, specifically Convolutional Neural Networks (CNN) and ResNet50, comparing their performance and implementation. Detailed insights into how each model was utilized and the code implementation strategies will be provided.

4.1 Software development Methodology

Our project went through a lot of changes in both functionality and design throughout the development time. To ensure an efficient and effective development process, we have chosen to adopt the **Agile methodology**.

Agile is a project management framework that focuses on iterative and incremental development, collaboration, flexibility, and responsiveness to change, With Agile, we intend to break the project into smaller, manageable tasks and regularly evaluate and adapt the approach based on emerging feedback and insights

This iterative nature of Agile allow us to continually improve the deep learning model.

Agile's focus on collaboration also enable us to work closely with stakeholders, such as healthcare professionals and domain experts, to gather their insights and ensure model accuracy and relevance to real-world scenarios. In general, by adopting the Agile methodology, we got a more dynamic and efficient development process that enhanced the success of our graduation project.

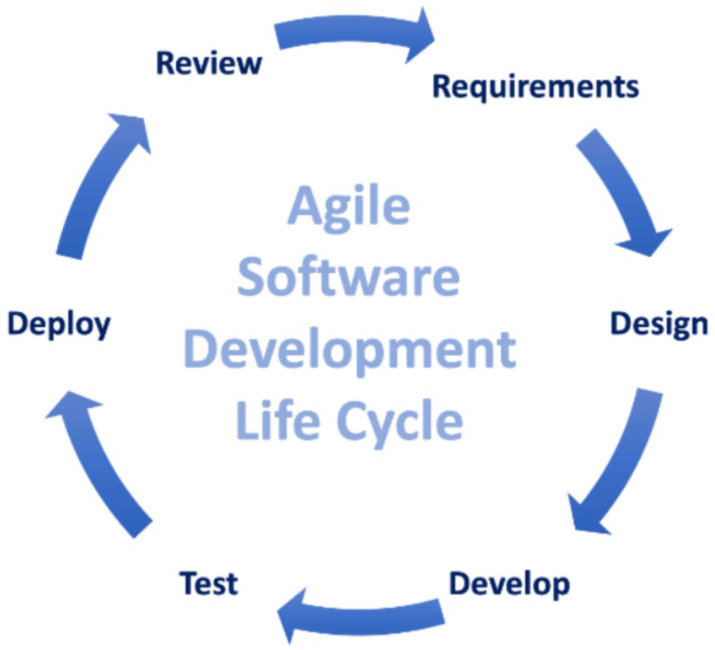


Figure 11: Agile methodology

4.2 Implementation

This section details the development of the AI-deep learning models for ECG image classification within the website designed for medical students and doctors. We utilized Python as the programming language for writing the code. Our primary objective was to create a model with high accuracy and low loss value for effective ECG image classification.

We explored two approaches:

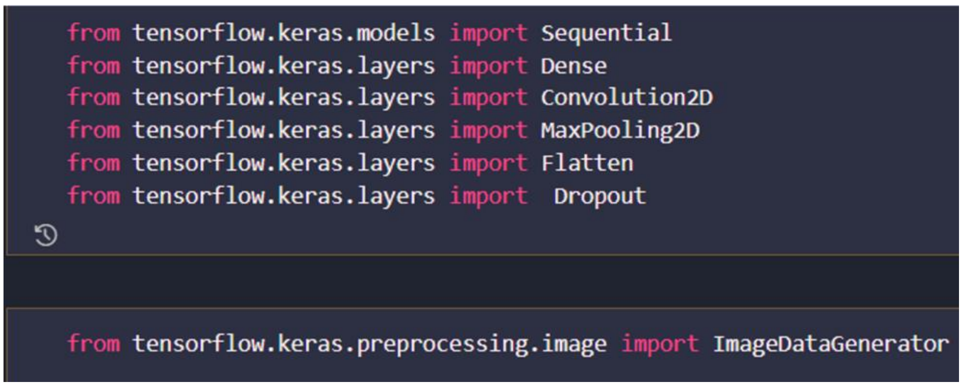
4.2.1 Custom CNN model

Import python libraries and Submodule

- From tensorflow.keras.models which is a module that provides classes for building and manipulating neural network models in TensorFlow. We import the **Sequential model** which is a linear stack of layers, allowing us to easily build a neural network by adding layers sequentially.

- From tensorflow.keras.layers which is a module that provides various types of layers to build a CNN model. We import:
  Dense, it is a layer that represents a densely connected layer, where each neuron is connected to every neuron in the previous layer.
  Convolution2D, it is a layer that performs 2D spatial convolution over images, by applying a specified number of filters to extract features from input images.
  MaxPooling2D, it is a layer that performs max pooling operation for spatial data, reducing the dimensionality of feature maps and extracting dominant features.
  Flatten, it is a layer that flattens the input, converting it into a 1D array, which is necessary before feeding it into a fully connected layer.
  Dropout, it is a layer that applies dropout regularization, randomly setting a fraction of input units to zero during training to prevent overfitting.

- From tensorflow.keras.preprocessing.image which is a module that provides tools for image preprocessing. We import **ImageDataGenerator** which generates batches of augmented/normalized data from image data.


```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import  Dropout
```

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

Figure 12: Import python libraries and Submodule in CNN implementation

Image preprocessing

Creates a data generator for training images and test images.

Within the ImageDataGenerator, we specify different data augmentation techniques to be applied to the images. We rescale the image pixel values from a range of 0 to 255 to a range of 0 to 1. This is a common pre-processing step for CNNs as it improves the training process.

We put shear_range value equal to 0.2, this argument is a random shearing transformations to the images during training. Shearing is a distortion that tilts the image slightly along a diagonal axis. The value 0.2 specifies the maximum shearing factor, meaning the image can be sheared by up to 20% in either direction.

The zoom_range equal to 0.2, this argument is a random zoom to the images during training. It randomly zooms in or out of the image by up to 20%.

The horizontal_flip equal to True, this argument randomly flips half of the training images horizontally (mirroring the image). This helps the model learn features that are independent of the image's original orientation.

The train_datagen applies augmentations (rescaling, shearing, zooming, and flipping) that mentioned before to the training images to increase the dataset size virtually and improve the model's ability to generalize to unseen data.

The test_datagen only applies rescaling to the testing images. This ensures the model makes predictions on images with the same scale it was trained on.

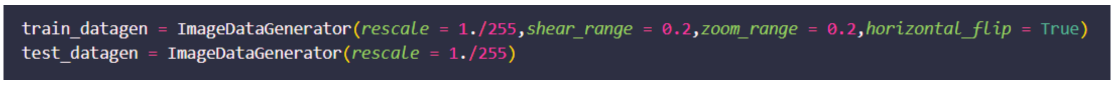

```
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

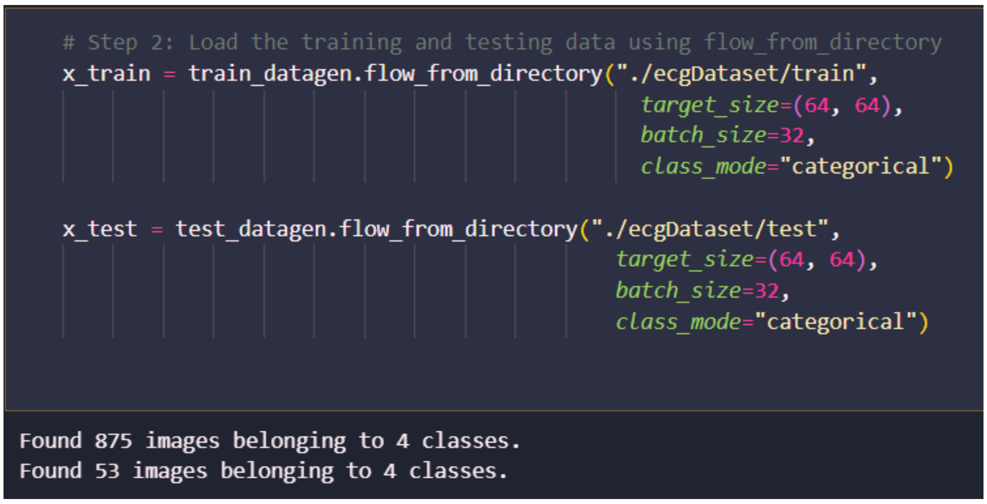
Figure 13: Image preprocessing in CNN implementation

Load images from a directory

The function below is used to load images from ecgDataset directory structure and prepare them for training and testing a CNN model.

Images are resized to a uniform size 64x64 pixels, batched into groups of 32 for training the CNN model. The batch_size is the number of images processed by the model at a time during training.

Finally, we applied Categorical encoding to the training data labels.



```
# Step 2: Load the training and testing data using flow_from_directory
x_train = train_datagen.flow_from_directory("./ecgDataset/train",
                                            target_size=(64, 64),
                                            batch_size=32,
                                            class_mode="categorical")

x_test = test_datagen.flow_from_directory("./ecgDataset/test",
                                          target_size=(64, 64),
                                          batch_size=32,
                                          class_mode="categorical")
```

```
Found 875 images belonging to 4 classes.
Found 53 images belonging to 4 classes.
```

Figure 14: Load images from a directory in CNN implementation

Define a CNN architecture

We created a sequential model, which is a linear stack of layers where the output of each layer feeds into the next. Then we Defined Layers within the Model, at first we defined convolutional layer that extract features using 16 filters each and a 3x3 kernel size. We applied ReLU function which introduce non-linearity to the model.

The argument input_shape define the shape of the input images of size 64x64 pixels with 3 channels (RGB color channels).

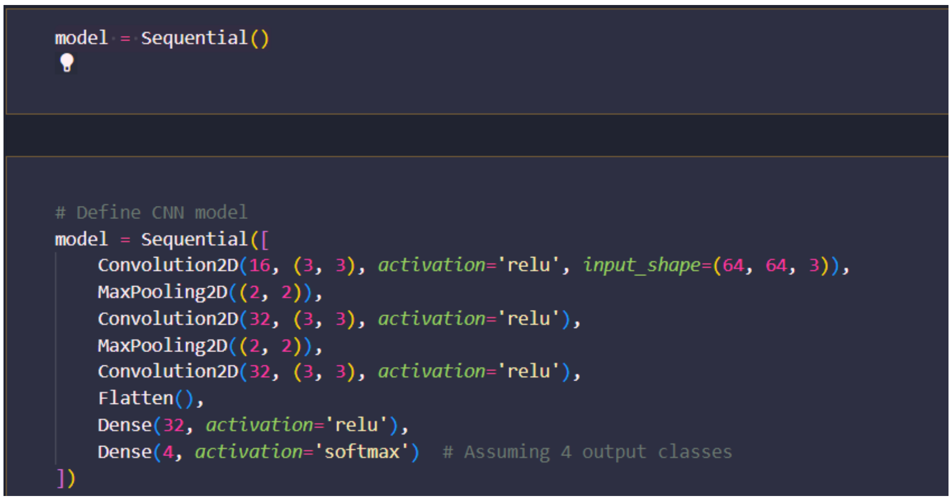
We defined a max pooling layer inserted between convolutional layers, it is reduces the dimensionality of the data by taking the maximum value from a 2x2 region in the input.

The second convolutional layer extract higher-level features from the previous layer's outputs using 32 filters each and a 3x3 kernel size. They also use the ReLU activation function.

We added flatten layer that flattens the multi-dimensional output of the convolutional layers into a single one-dimensional vector. This is necessary before feeding the data into fully-connected layers.

At last, we added two of dense Layers (Fully-Connected), first one with 32 neurons. Here, the ReLU activation is used again.

The second one defines the final output layer of the model, it has 4 neurons because we want to classify the ECG data into 4 different categories. The softmax activation function ensures the output values sum to 1 and represent probabilities for each class.



```
model = Sequential()
```

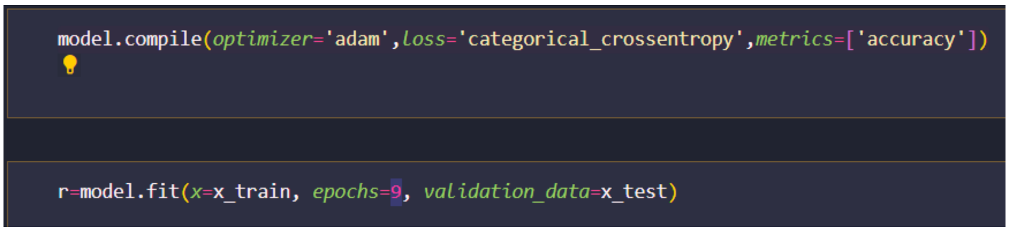
```
# Define CNN model
model = Sequential([
    Convolution2D(16, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D((2, 2)),
    Convolution2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Convolution2D(32, (3, 3), activation='relu'),
    Flatten(),
    Dense(32, activation='relu'),
    Dense(4, activation='softmax')  # Assuming 4 output classes
])
```

Figure 15: Define our CNN architecture

Training the Model

The compilation configures how the model learns. It defines an optimizer (Adam) that adjusts the model's internal parameters and a loss function (categorical crossentropy) that measures how well the model's predictions match the actual labels. Additionally, it tracks accuracy to monitor learning progress.

The model is trained for a nine epochs over the training data using the training images (x_train). While training, the model's performance is also evaluated on the testing images (x_test) to prevent overfitting.



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

```
r=model.fit(x=x_train, epochs=9, validation_data=x_test)
```

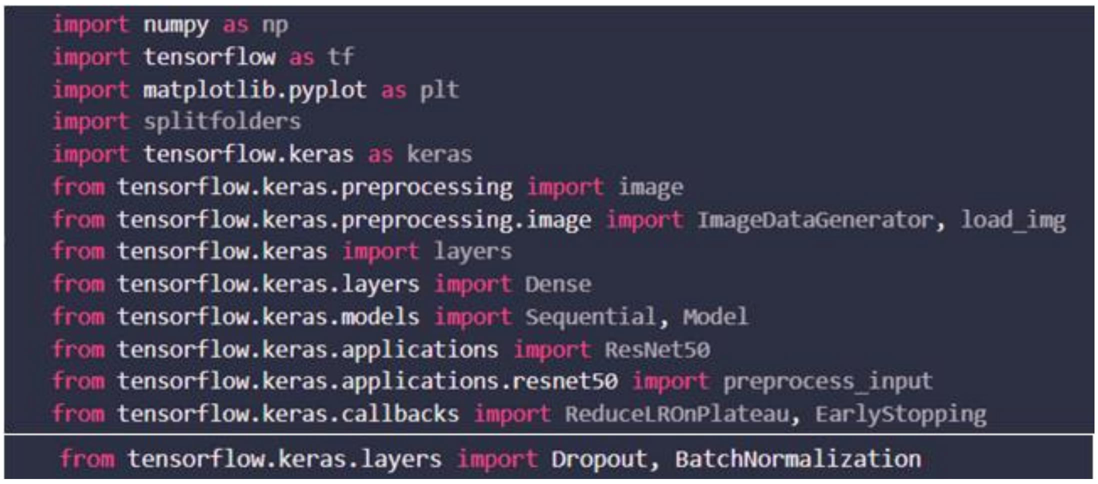
Figure 16: Training CNN Model

4.2.2 Pre-trained ResNet50 Model

Import python libraries and Submodule

We imported a variety of libraries and submodules, they are:

- **numpy** library that provides powerful tools for numerical computations with arrays. It's used for data manipulation.
- **tensorflow** library for building and training machine learning models, including CNNs.
- **matplotlib.pyplot** for creating visualizations like plots and charts, potentially to visualize the training process.
- **splitfolders**, it's a custom library for splitting data into training, validation, and test sets.
- **tensorflow.keras**, it's a submodule of TensorFlow that provides tools specifically for building neural networks.
- From tensorflow.keras.preprocessing.image which is a module that provides tools for image preprocessing. We imported **ImageDataGenerator** which generates batches of augmented/normalized data from image data, and **Image** submodule which helps with image loading and pre-processing.
- **Layers**, it's a submodule provides the building blocks for neural network layers.
- **Dense**, that create a densely connected layer, commonly used in the final stages of a CNN for classification.
- **Sequential, Model**, these for help define the architecture of the neural network model.
- **Applications**, it's a submodule provides pre-trained models like ResNet50.
- **ResNet50**: This is a pre-trained image classification model based on the ResNet architecture.
- **preprocess_input** function it is specific to ResNet50 and helps prepare images for the model's input format.
- **Callbacks**, a submodule that provides functions to monitor and potentially adjust the training process during training.
- **ReduceLROnPlateau**, this callback can reduce the learning rate if the model's performance plateaus, helping to prevent overfitting.
- **EarlyStopping**, this callback can stop training early if the validation performance doesn't improve for a certain number of epochs, preventing further training on a potentially overfitting model.
- From tensorflow.keras.layers which is a module that provides various types of layers to build a CNN model. We imported **Dropout**, which is a layer that applies dropout regularization, randomly setting a fraction of input units to zero during training to prevent overfitting, and **BatchNormalization** which it's a layer that helps normalize the activations (outputs) of the previous layer within each mini-batch during training. This can improve the training speed and stability of the model by ensuring the data distribution is more consistent across mini-batches.



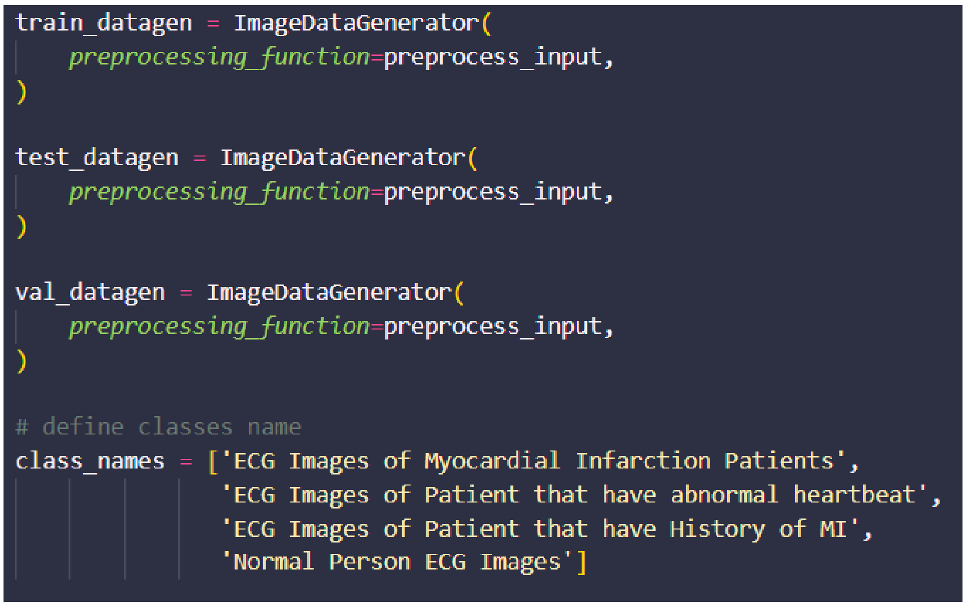
```python
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import splitfolders
import tensorflow.keras as keras
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.layers import Dropout, BatchNormalization
```

Figure 17: Import python libraries in resNet50 implementation

Image preprocessing

We defined three ImageDataGenerator objects, they are train_datagen, test_datagen and val_datagen. Then we set a common pre-processing function for all of them. The preprocess_input function specifically prepares images for the ResNet50 pre-trained model by resizing the image to a specific size expected by ResNet50, Converting the color format (e.g., RGB to BGR) if needed by ResNet50, and Normalizing the pixel values (usually between 0 and 1 or -1 and 1) as expected by the model.

Additionally, we defined class names for our image dataset.



```python
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
)

val_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
)

# define classes name
class_names = ['ECG Images of Myocardial Infarction Patients',
               'ECG Images of Patient that have abnormal heartbeat',
               'ECG Images of Patient that have History of MI',
               'Normal Person ECG Images']
```

Figure 18: Image preprocessing in resNet50 implementation

We utilized the splitfolders.ratio function from the splitfolders library to split our data into training, validation, and test sets. The function has many argument, first one is "./data" which is the path to the directory containing our original dataset.

The path "./data-split" is the output directory where the split data will be saved. The splitfolders.ratio function will create subfolders within this directory named "train", "val", and "test".

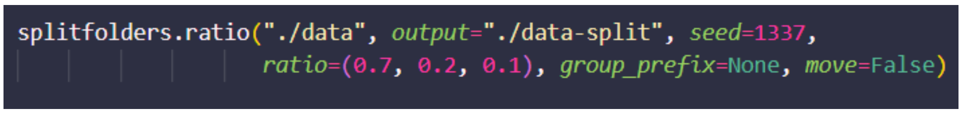
We sets a random seed for splitting the data. Using the same seed ensures reproducible splits if you run the code again.

The tuple ratio= (0.7, 0.2, 0.1) defines the proportion of data to be allocated to each split. The breakdown of our dataset was as follows:

- 70% of the data will go into the "train" folder. This subset was utilized to train the AI models. Training on a significant portion of the dataset allowed the models to learn the underlying patterns and features associated with different cardiac conditions.
- 10% of the data will go into the "val" folder. This subset was utilized to fine-tune the model and monitor its performance during training. The validation set helped prevent overfitting and ensured that the model generalized well to unseen data.
- 20% of the data will go into the "test" folder. The test set was used to evaluate the final performance of the trained models, providing an unbiased assessment of their efficacy in real-world scenarios.

The group_prefix is used for splitting files based on a prefix in the filename. Since it's set to None, it won't be used.

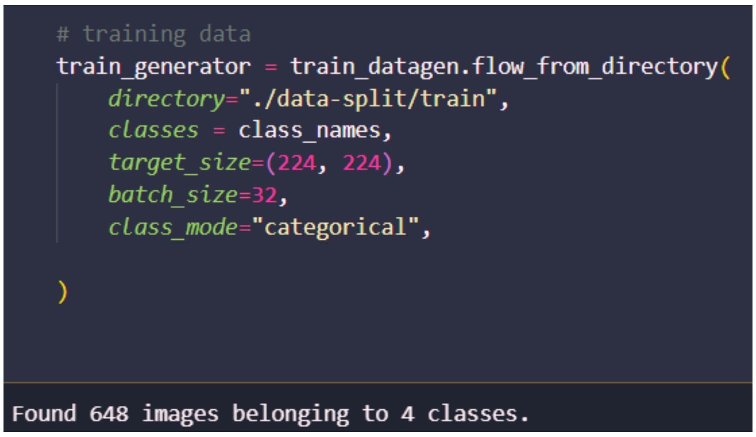
Finally we set move argument to False, the splitfolders.ratio function will copy the files to the new folders. Setting it to True would move the files instead, but it's generally safer to use False value to avoid accidentally deleting our original data.



```
splitfolders.ratio("./data", output="./data-split", seed=1337,
                   ratio=(0.7, 0.2, 0.1), group_prefix=None, move=False)
```

Figure 19: The proportion of data of each split in resNet50 implementation

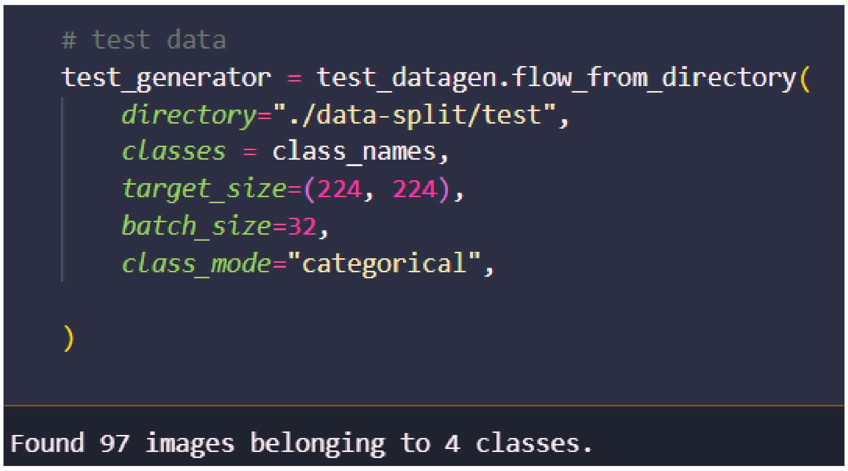
We create a generator named train_generator specifically for our training data. It utilizes the pre-configured train_datagen object for pre-processing. We specified the training data directory, class names, target image size 224x224 pixels likely for ResNet50, batch size equal to 32 images delivered at once, and we set the class labels to a categorical format, which is suitable for multi-class classification.


```
# training data
train_generator = train_datagen.flow_from_directory(
    directory="./data-split/train",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="categorical",

)
```

```
Found 648 images belonging to 4 classes.
```

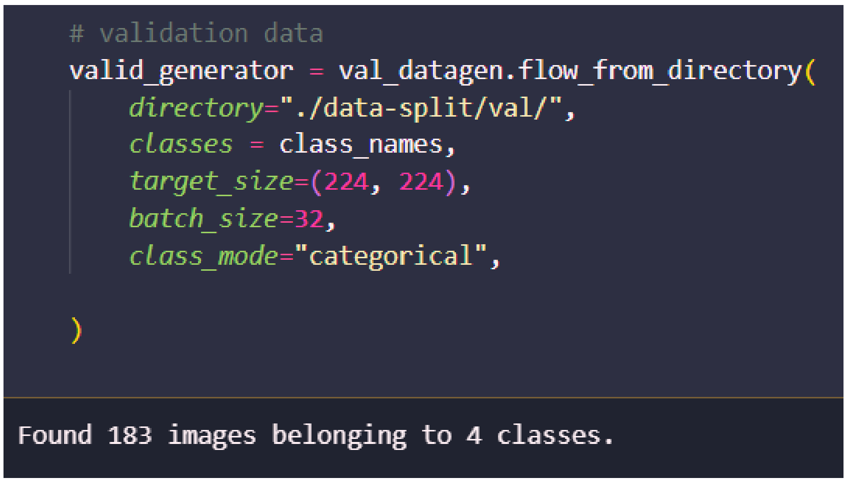
Figure 20: Training data generator

Following the same approach used for the training data generator (train_generator), we create separate generators for test and validation data (test_generator and val_generator) using their respective ImageDataGenerator objects (test_datagen and val_datagen).


```
# test data
test_generator = test_datagen.flow_from_directory(
    directory="./data-split/test",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="categorical",

)
```

```
Found 97 images belonging to 4 classes.
```

Figure 21: Testing data generator

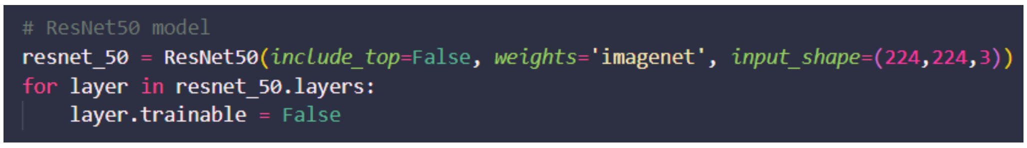

```
# validation data
valid_generator = val_datagen.flow_from_directory(
    directory="./data-split/val/",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="categorical",

)
```

```
Found 183 images belonging to 4 classes.
```

Figure 22: Validation data generator

Loading and configuring a pre-trained ResNet50 model

We loaded a ResNet50 model instance from the TensorFlow Keras applications library, leveraging its pre-trained weights on the ImageNet dataset. It excludes the final classification layers as we will add our own and freezes all the weights in the pre-trained model using transfer learning. This prevents the pre-trained layers from being re-trained from scratch, allowing them to focus on feature extraction while you train your custom layers on top for ECG image classification.



```
# ResNet50 model
resnet_50 = ResNet50(include_top=False, weights='imagenet', input_shape=(224,224,3))
for layer in resnet_50.layers:
    layer.trainable = False
```

Figure 23: Loading and configuring a pre-trained ResNet50 model-part1

We build upon the pre-trained ResNet50 model

We took the output of the pre-trained ResNet50 model and assigns it to the variable x by this line (x = resnet_50.output). This essentially captures the features learned by the pre-trained model.

The global average pooling layer it averages the activations across the entire spatial extent (width and height) of the feature maps in x. This reduces the dimensionality of the data from a 3D tensor to a 1D vector, suitable for feeding into a fully-connected layer.

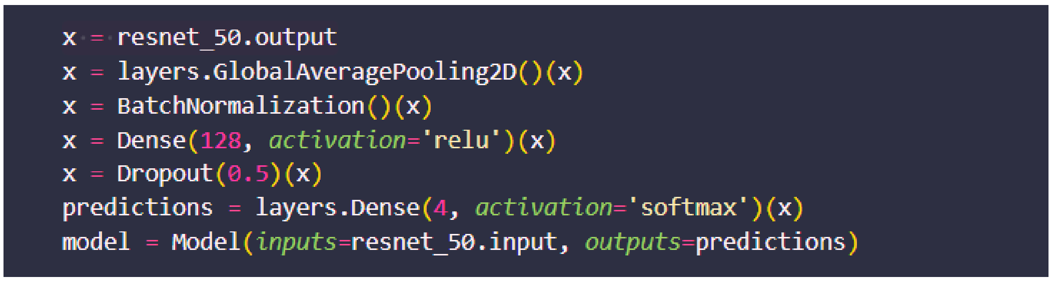
The layer applies batch normalization to the output of the global average pooling layer. As mentioned earlier, this helps improve training stability.

We created a densely connected layer with 128 neurons. The activation='relu' argument specifies the ReLU activation function, which introduces non-linearity into the network.

The dropout layer with a rate of 0.5. During training, it will randomly drop 50% of the activations from the previous layer, helping prevent overfitting.

The final output layer of the model. It has 4 neurons that corresponding to our 4 ECG image classes and uses the softmax activation function. Softmax ensures the output values sum to 1 and represent probabilities for each class.

We defined the overall Keras model. It has specified the input layer (resnet_50.input) where the pre-trained model receives data, and the output layer (predictions) containing our custom classification head.

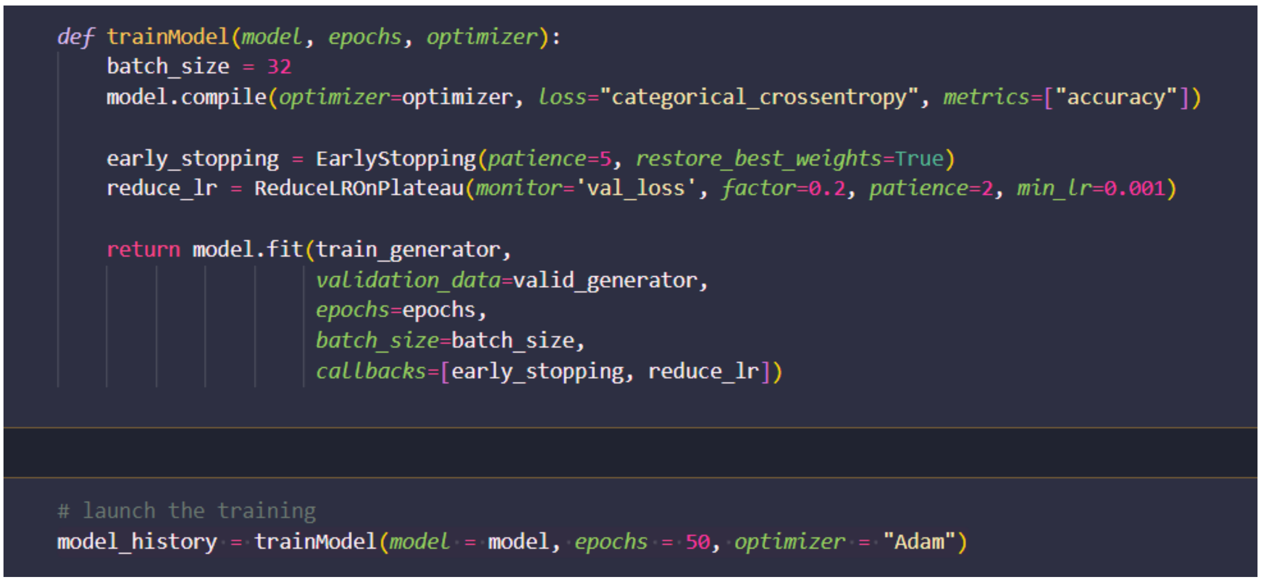


```
x = resnet_50.output
x = layers.GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = layers.Dense(4, activation='softmax')(x)
model = Model(inputs=resnet_50.input, outputs=predictions)
```

Figure 24: Loading and configuring a pre-trained ResNet50 model-part2

Training the Model

We trained our ECG image classifier by calling the trainModel function, passing our built model, the desired number of training epochs which equal to 50, and the chosen optimizer ("Adam"). The trainModel function compiles the model, sets up callbacks to prevent overfitting and adjust learning rate, and then trains the model for the specified epochs using the training and validation data generators. Finally, it returns the training history containing metrics like accuracy and loss for each epoch, which we store in the model_history variable for later evaluation of the training process.



```
def trainModel(model, epochs, optimizer):
    batch_size = 32
    model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"])

    early_stopping = EarlyStopping(patience=5, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=0.001)

    return model.fit(train_generator,
                     validation_data=valid_generator,
                     epochs=epochs,
                     batch_size=batch_size,
                     callbacks=[early_stopping, reduce_lr])

# launch the training
model_history = trainModel(model = model, epochs = 50, optimizer = "Adam")
```

Figure 25: Training the resNet50 model

4.3 Programming Language

4.3.1 Python

Known for its versatility, simplicity, and readability, is favored for rapid development, scripting, and seamless integration. Its clear syntax reduces maintenance efforts, it is characterized by large community and libraries.

For these reasons, we used the Python language in our project to write the code for the model (building the model, training it on Dataset, and predicting the diagnosis of the ECG image).

Figure 26: Python logo

4.3.2 JavaScript

JavaScript (JS) is a dynamic, lightweight language known for its versatility and first-class functions. With dynamic features like runtime object construction and function variables, JS supports various programming styles. Governed by ECMAScript standards.

We have used it in web development, both front-end and back-end, to build our website.

Figure 27: JavaScript logo

4.4 Web development technologies

4.4.1 Integrated Development Environment

4.4.1.1 Visual Studio Code (VS Code)

For the development of this project, we used it as the Integrated Development Environment (IDE). VSCode provided a versatile and robust platform for coding each section of the deep learning models and the web application. Its extensive library of extensions, support for multiple programming languages, features, and integrated debugging significantly enhanced the efficiency and quality of the development process.

Figure 28: VS code logo

4.4.2 Front-End Technologies

4.4.2.1 HTML5

HyperText Markup Language, serves to structure and present web content, including text, images, links, and multimedia elements. Its goals include providing a framework for content structuring, presentation, document linking, media embedding, interactive forms, and semantic meaning. HTML forms the backbone of web pages, facilitating their display and functionality across various browsers and devices.

Figure 29: HTML logo

4.4.2.2 CSS

CSS stands for Cascading Style Sheets. It is a style sheet language used for describing the presentation and formatting of a document written in HTML or XML. The primary purpose of CSS is to separate the structure of a web page (defined by HTML or XML) from its presentation aspects. By using CSS, web developers can control the layout, colors, fonts, and other visual elements of a webpage, making it more flexible, consistent, and easier to maintain.

Figure 30: CSS logo

4.4.2.3 Bootstrap

Bootstrap is a front-end framework that offers pre-designed templates, a responsive grid system, and a variety of reusable components and CSS classes, making it easy to create visually appealing and responsive websites with consistent designs. By leveraging Bootstrap's components and styles, developers can rapidly build web interfaces without starting from scratch, while still allowing for customization to suit project requirements.

Figure 31: Bootstrap logo

4.4.2.4 React.js

React.js is a powerful JavaScript library for building user interfaces, known for its component-based architecture and virtual DOM, which enables efficient rendering of dynamic and interactive UIs. With its declarative approach and reusable components, React simplifies front-end development, making it easier to create complex web applications with better performance and maintainability.

Figure 32: React.js logo

4.4.3 Back-End Technologies

4.4.3.1 Node.js

Node.js is a runtime environment that allows developers to run JavaScript code on the server-side, using Google's V8 engine. It's known for its event-driven, non-blocking I/O model, making it lightweight and efficient for building scalable network applications. Node.js is commonly used for web servers, APIs, and real-time applications, leveraging its rich ecosystem of packages from npm (Node Package Manager) for rapid development.

Figure 33: Node.js logo

4.4.4 Database

4.4.4.1 MongoDB

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, offering high scalability and performance. It's known for its ease of use, flexibility, and ability to handle large volumes of unstructured data. MongoDB's document-oriented model allows for dynamic schema changes and powerful querying capabilities.

Figure 34: MongoDB logo

4.5 Packages and Libraries

4.5.1 jsonwebtoken (JWT)

In our website, we employed the JWT (JSON Web Token) npm package to implement robust authentication and authorization mechanisms. JWTs were utilized to securely manage user sessions, providing encrypted tokens that authenticated clients could use to access protected resources. Integration with our Node.js backend facilitated seamless token generation, verification, and expiration handling, ensuring stringent security standards were met.

4.5.2 Child process

We utilised the child_process npm package was instrumental in integrating the backend functionalities of a Node.js website with Python code responsible for machine learning model predictions. This setup allowed seamless communication between the Node.js server, which handled web functionalities and database connections, and the Python scripts executing machine learning algorithms. By utilizing child_process, we ensured efficient coordination and data exchange between the Node.js backend and Python models, enabling real-time predictions and enhancing overall system performance and flexibility. This approach effectively combined the strengths of both technologies to deliver a robust and integrated web application capable of dynamic data processing and predictive analytics.

4.5.3 Dotenv

We implemented dotenv to securely manage environment variables across the Node.js backend. By using dotenv, sensitive configuration details such as database credentials and API keys were stored in an .env file, ensuring they were kept out of version control and accessed only in the runtime environment. This approach enhanced security by preventing inadvertent exposure of sensitive information in the source code.

4.5.4 Tenserflow

TensorFlow was crucial for developing and deploying complex neural network models. Its scalable architecture and integration with Keras provided efficient tools for building and training deep learning architectures. TensorFlow's extensive library and GPU acceleration supported high-performance computations, vital for processing large datasets and achieving accurate predictions. Its flexibility and comprehensive documentation greatly facilitated research across diverse machine learning applications.

4.5.5 keras

In our project, Keras served as a powerful neural networks API, seamlessly integrated with TensorFlow backend. It facilitated rapid development of deep learning models with its user-friendly interface, enabling efficient experimentation and optimization. Keras' compatibility with GPU acceleration enhanced performance for handling datasets and complex architectures, pivotal in achieving robust results in predictive analytics and machine learning tasks.

4.5.6 Nodemailer

A powerful Node.js module, to manage essential email communications, enhancing both user experience and security. We implemented Nodemailer to send confirmation emails upon user registration, ensuring the validity of user identities, and to dispatch verification codes for password resets, protecting against unauthorized access. Nodemailer's straightforward API, and secure transmission capabilities made it an indispensable tool, streamlining our development process and ensuring reliable email delivery.

4.5.7 Mongoose

Mongoose is an elegant Node.js ORM (Object-Relational Mapping) library for MongoDB, providing a straightforward way to model application data and interact with MongoDB databases using JavaScript. It simplifies database operations, offering features like schema validation, middleware, and query building, making MongoDB interactions more organized and efficient.

CHAPTER 5
SOFTWARE ANALYSIS AND DESIGN

5.1 Use case diagram

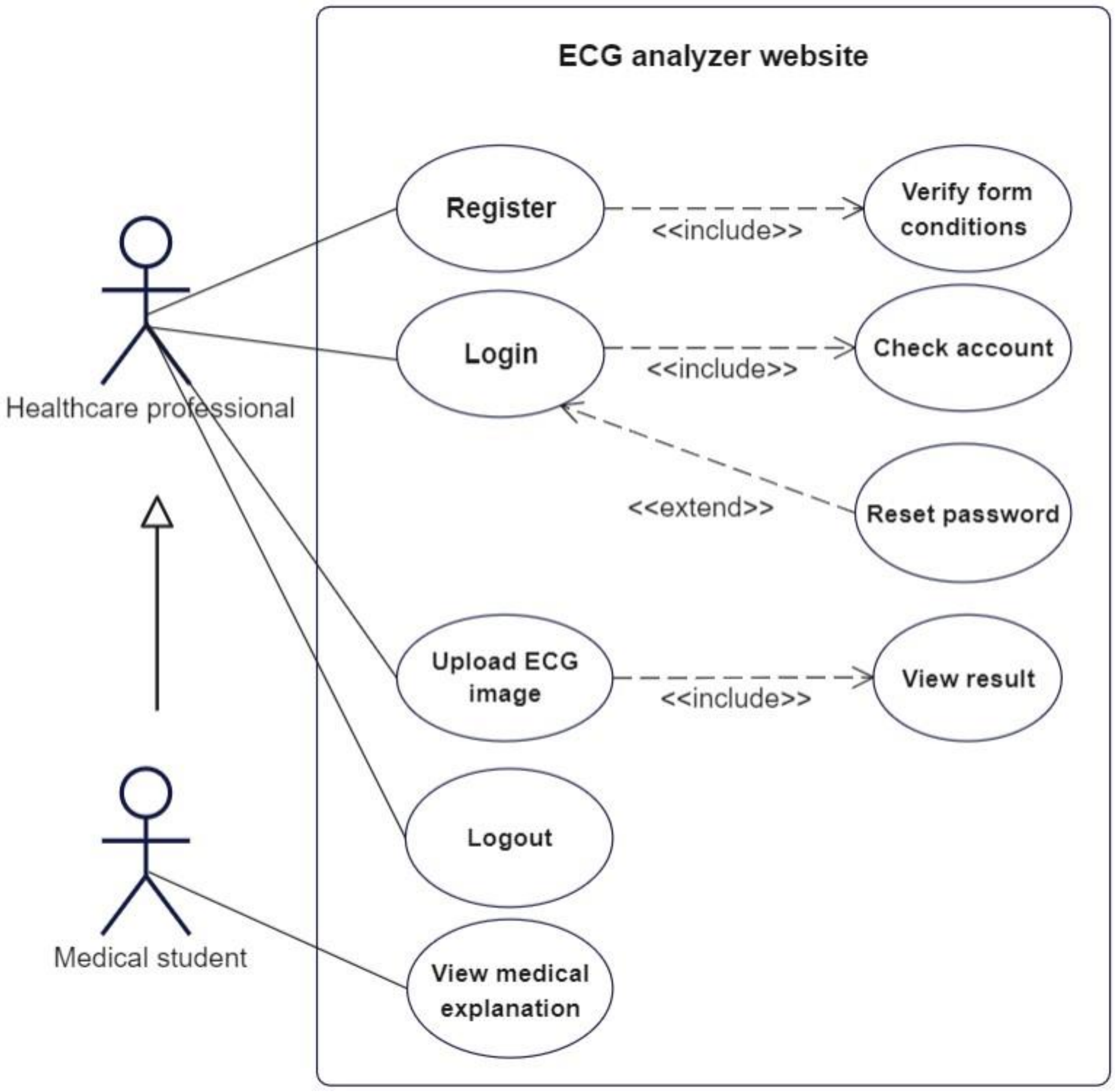


Figure 35: Use case diagram

Table 2: Register use case description

Use case name:	Register	
Scenario:	Create online user account.	
Triggering event:	User want create online account by clicking register button.	
Brief description:	Online user creates user account by entering basic information (username, Email, Password, and user role).	
Actors:	HealthCare professional, Medical student.	
Related use cases:	Verify form conditions use case.	
Stakeholders:		
Preconditions:	The user should be either medical student or healthCare professional. Email should be valid and unique. Username must be a string with a minimum length of 3 characters and a maximum length of 20 characters. Email must be a valid email format and at least five characters long. Password must be a string with a minimum length of 8 characters and at least one uppercase letter, one lowercase letter, one digit and one special character. Confirm Password must match the password field. The user must activate their account.	
Postconditions:	User account must be created and saved.	
Flow of activities:	Actor	System
	1. User enter name, email, user role and password	1.1 System check if entered data is valid
Exception conditions:	1.1 Username is empty. 1.2 Username less than 3 characters long. 1.3 Username more than 20 characters. 1.4 Email is empty. 1.5 Email is not valid. 1.6 Password is empty. 1.7 Password doesn’t match the specified pattern. 1.8 Confirm password is empty 1.9 Confirm password mismatch password field.	

Table 3: Login use case description

Use case name:	Login	
Scenario:	Login to user account.	
Triggering event:	User want enter online account by clicking login button.	
Brief description:	User enter account by fill log in information (Email and Password).	
Actors:	HealthCare professional, Medical student	
Related use cases:	Check account use case, Reset password use case.	
Stakeholders:		
Preconditions:	The user should has account. Ensure both email and password fields are filled. The user must confirm their email within 24 hours of registering on the website.	
Postconditions:	User must be able to access upload page. User as a medical student can access to view medical explanation page also.	
Flow of activities:	Actor	System
	1. User enter email and password	1.1System check if entered data is match.
Exception conditions:	1.1 Email is empty. 1.2 Password is empty. 1.3 Data invalid. 1.4 Invalid password. 1.5 Email is not confirm.	

Table 4: Upload ECG image use case description

Use case name:	Upload ECG image.	
Scenario:	User upload ECG image.	
Triggering event:	User click on upload button.	
Brief description:	User upload ECG image to analyze it.	
Actors:	Health care professional, Medical student	
Related use cases:	View result use case.	
Stakeholders:		
Preconditions:	The user must be already logged in.	
Postconditions:	ECG image must be uploaded. ECG image must be analyzed. Result must be display.	
Flow of activities:	Actor	System
	1. User enter ECG image.	1.1System analyze the entered image and display Medical diagnosis.
Exception conditions:	1.1 Medical diagnosis not available for the ECG image. 1.2 image file extension is not allowed.	

Table 5: Logout use case description

Use case name:	Logout.	
Scenario:	Logout from user account.	
Triggering event:	User want to logout from their account by clicking logout button, or it has been 24 hours since the user's last login.	
Brief description:	User logout account by click on logout button on Navigation Bar of the website.	
Actors:	HealthCare professional, Medical student	
Related use cases:		
Stakeholders:		
Preconditions:	The user should have already logged in.	
Postconditions:	User must logged out the website.	
Flow of activities:	Actor	System
	1. User click on log out button	1.1System The system logs the user out of the site
Exception conditions:	---	

Table 6: View medical explanation use case description

Use case name:	View medical explanation.
Scenario:	The user get explanation about medical diagnosis of heart diseases and how to read ECG.
Triggering event:	User click view explanation page button.
Brief description:	The user view medical explanation about the heart diseases cases.
Actors:	Medical student.
Related use cases:	
Stakeholders:	
Preconditions:	The user already logged in as a medical student. The user clicked view explanation button.
Postconditions:	The explanation about medical diagnosis of heart diseases must be viewed.
Flow of activities:	**Actor:** 1. Medical student click view explanation page button. **System:** 1.1 System open view page.
Exception conditions:	---

5.2 System sequential diagrams

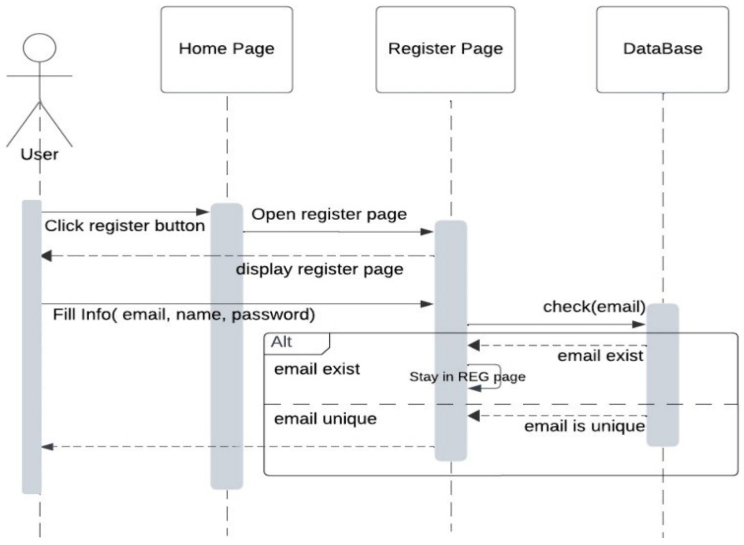


Figure 36: Register sequential diagram

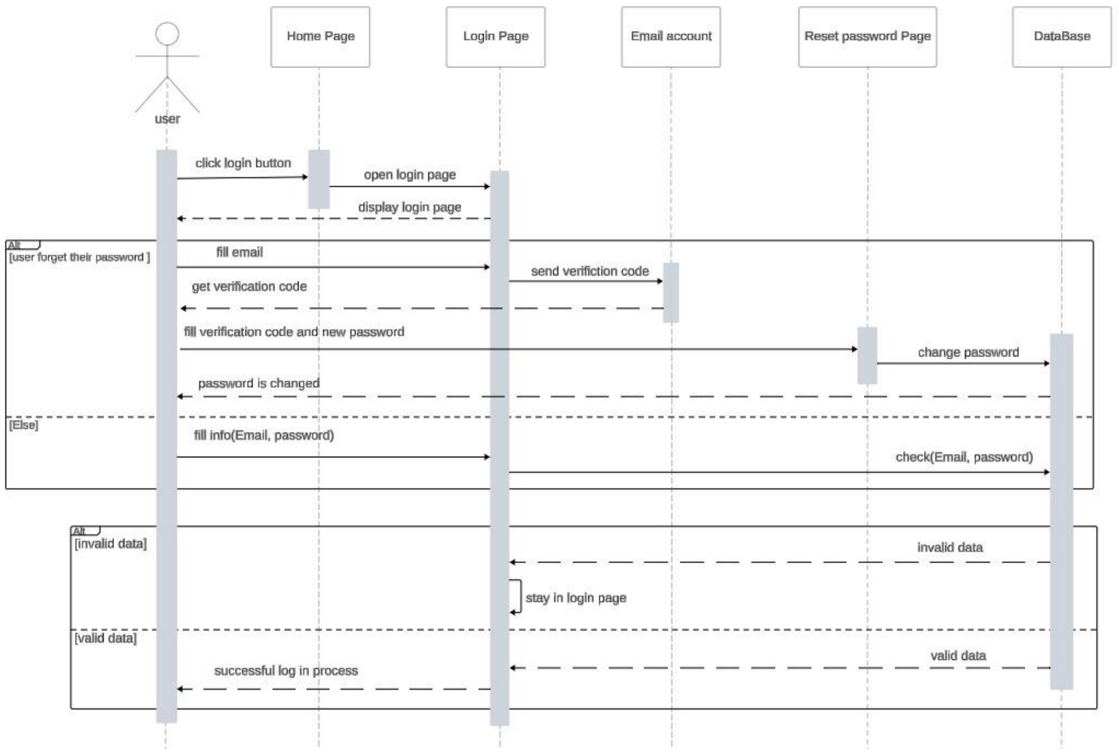


Figure 37: Login sequential diagram

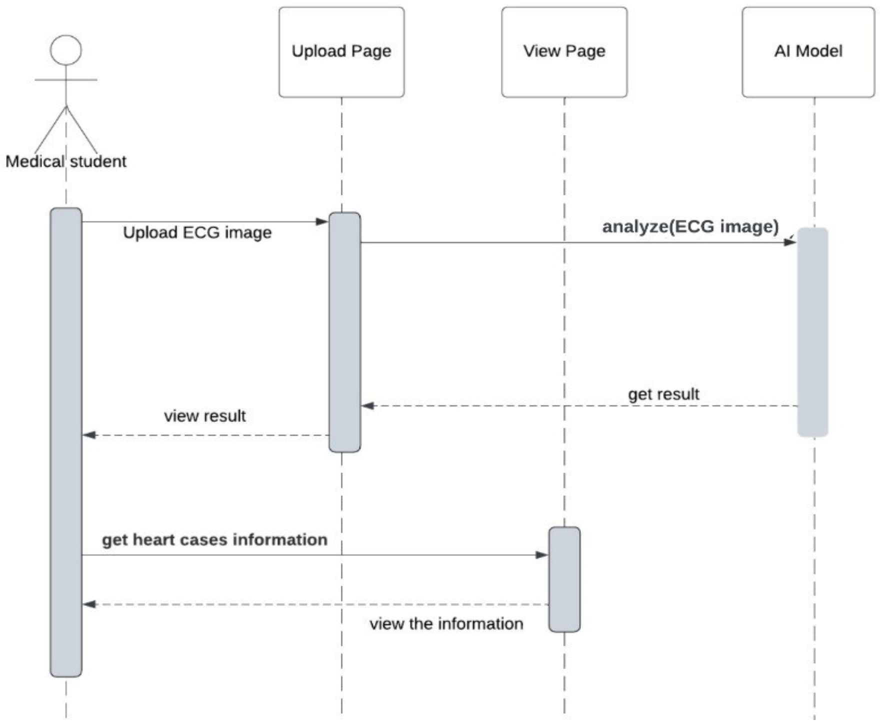


Figure 38: Medical student sequential diagram

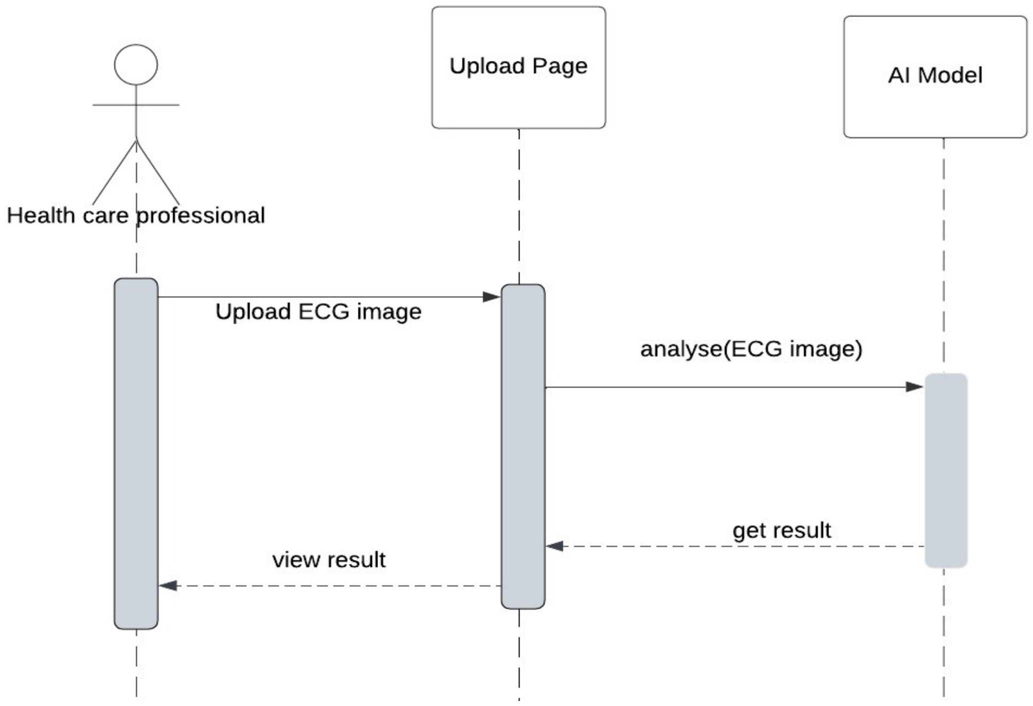
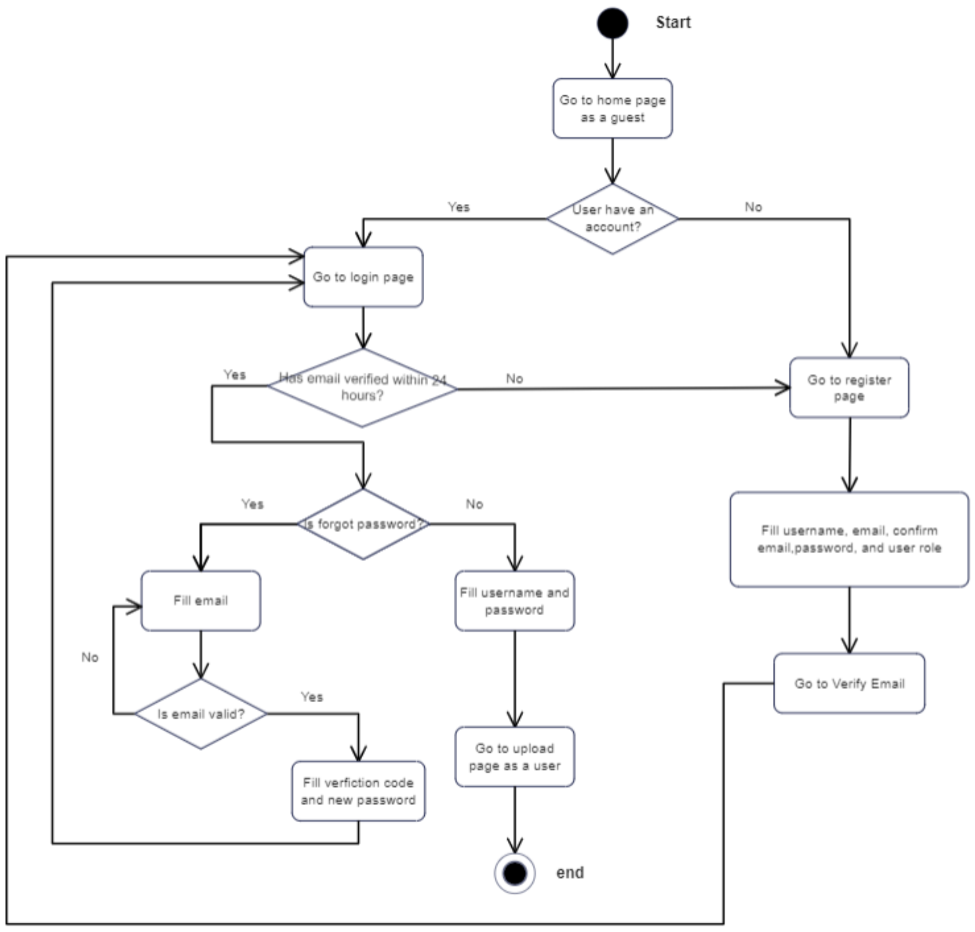


Figure 39: Healthcare professional sequential diagram

5.3 Activity diagrams



Start
Go to home page as a guest
User have an account?
Yes
No
Go to login page
Has email verified within 24 hours?
Yes
No
Go to register page
Is forgot password?
Yes
No
Fill username, email, confirm email,password, and user role
Fill email
Fill username and password
No
Go to Verify Email
Is email valid?
Yes
Go to upload page as a user
Fill verfiction code and new password
end

Figure 40: Register & Login activity diagram

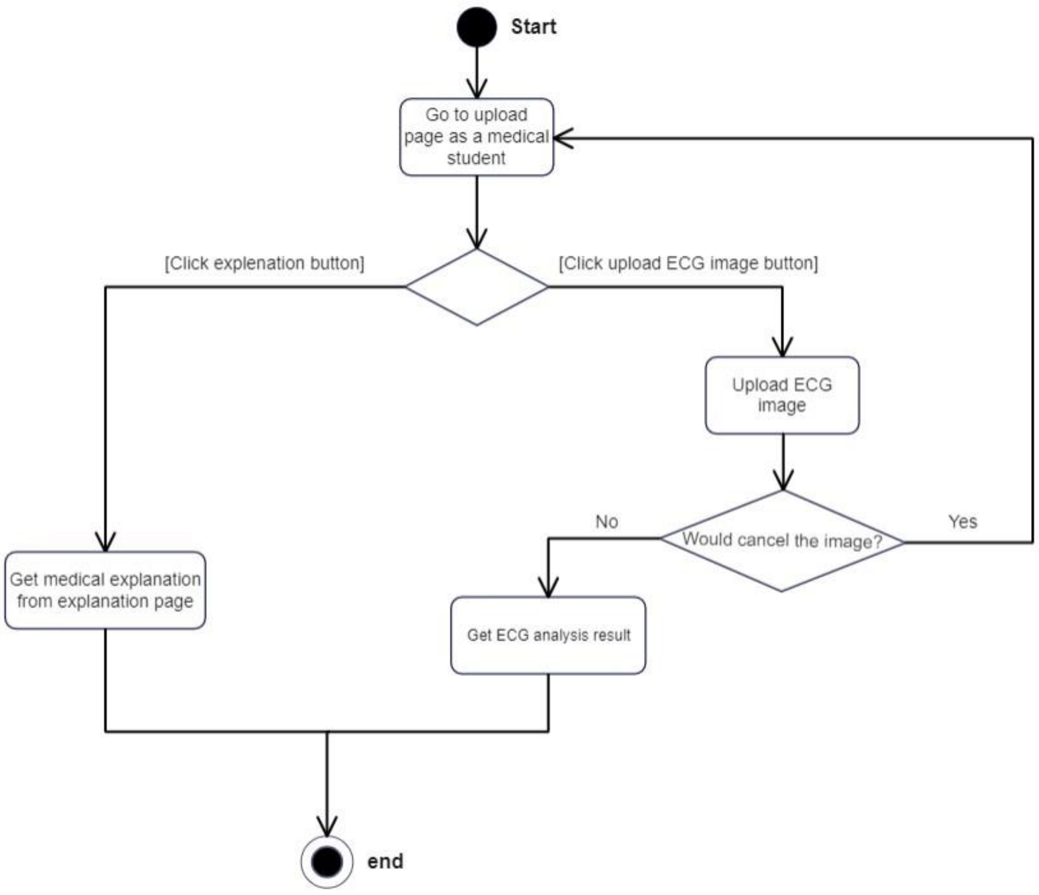

Start
Go to upload page as a medical student
[Click explenation button]
[Click upload ECG image button]
Upload ECG image
No
Would cancel the image?
Yes
Get medical explanation from explanation page
Get ECG analysis result
end

Figure 41: Medical student activity diagram

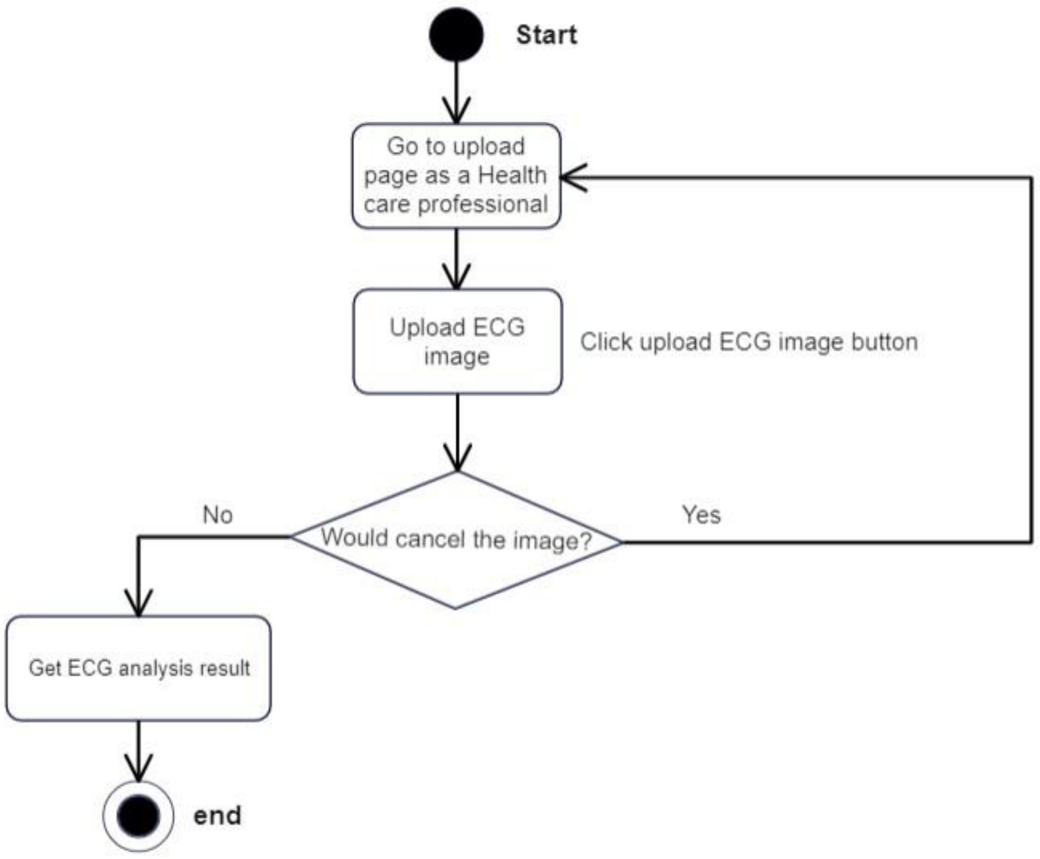

Start
Go to upload page as a Health care professional
Upload ECG image
Click upload ECG image button
Would cancel the image?
No
Yes
Get ECG analysis result
end

Figure 42: Health care professional activity diagram

5.4 Class diagram

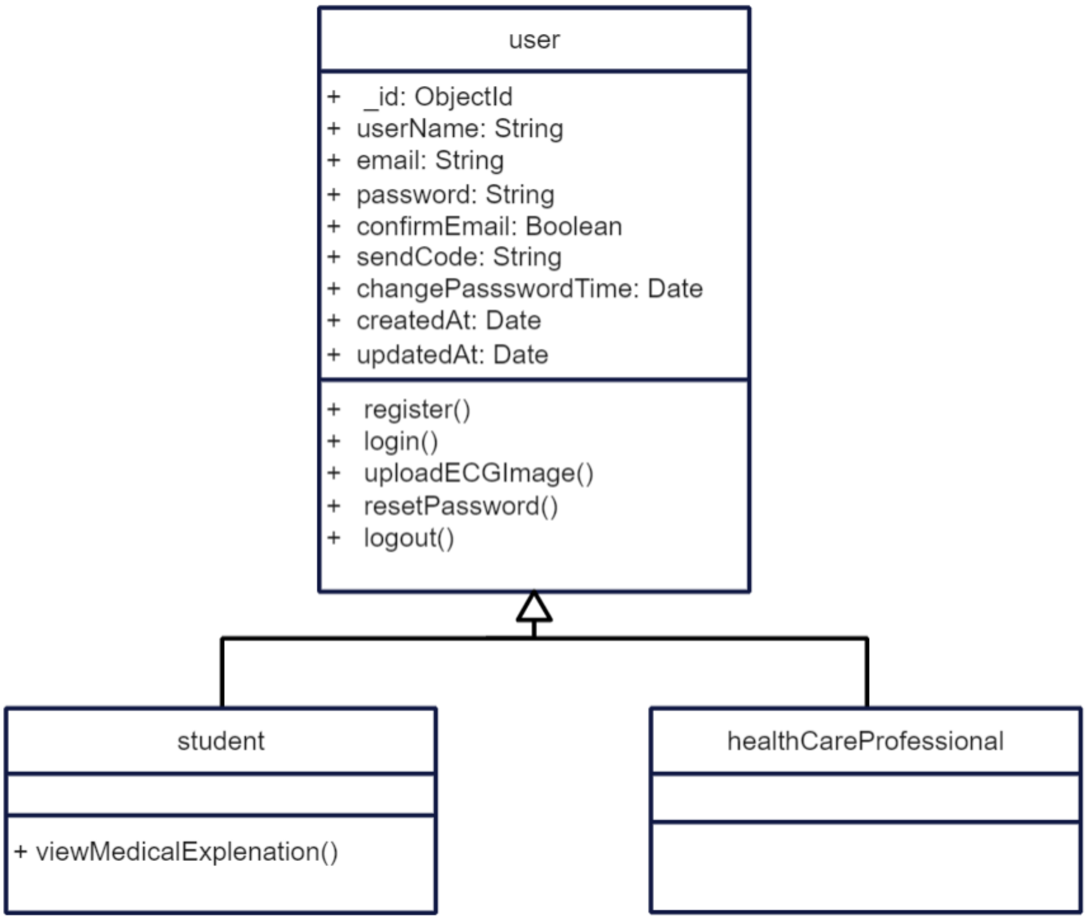


Figure 43: Class diagram

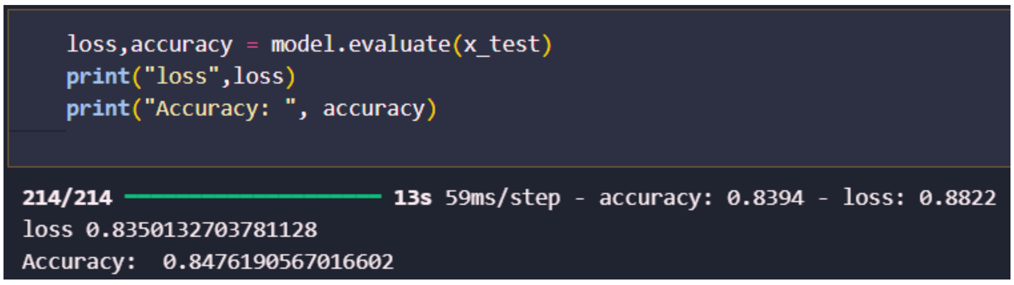
CHAPTER 6
RESULTS AND DISCUSSION

6.1 Model performance Analysis

We conduct a thorough evaluation of the performance metrics of two distinct convolutional neural network (CNN) architectures: a standard CNN and ResNet50. After training both models on our ECG dataset, we meticulously analyze their performance to provide valuable insights into their effectiveness for image classification. By comparing the outcomes, our goal is to decision-making regarding to use the model in our website.

6.1.1 Presenting the results of CNN model

The model showed promising results, achieved an accuracy of 84.76% on the test dataset. Additionally, the loss value of 0.835 on the testing set. Furthermore, the reported test accuracy of 84.76% reaffirms the model's ability to correctly classify images, emphasizing its proficiency in discerning between different classes within the dataset.



```
loss,accuracy = model.evaluate(x_test)
print("loss",loss)
print("Accuracy: ", accuracy)
```

```
214/214 ━━━━━━━━━━━━━━━━━━━━ 13s 59ms/step - accuracy: 0.8394 - loss: 0.8822
loss 0.8350132703781128
Accuracy:  0.8476190567016602
```

Figure 44: The results of CNN model (accuracy & loss value)

Here a graph that plots the training loss and validation loss of a model. The x-axis represents the training epochs, the y-axis represents the loss value. The blue line represents the training loss. It generally trends downwards as the model is trained on the training data and begins to learn the patterns. The orange line represents the validation loss. This is the loss measured on a separate dataset (the validation set) not used for training. It helps prevent overfitting. In this graph, the validation loss appears to be increasing after a certain point, which suggests overfitting be occurring.

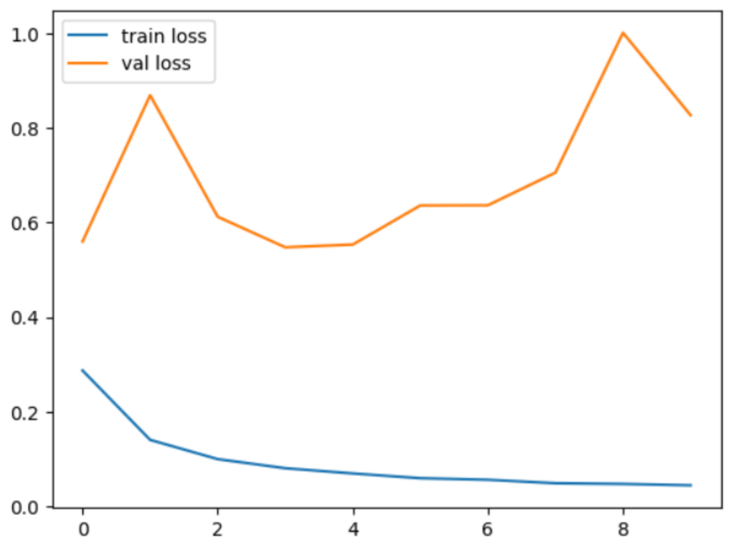


Figure 45: Graph of train & validation loss

Here a graph of the training accuracy and validation accuracy of the CNN model, likely trained on ECG data for classification. The x-axis represents the training epochs, the y-axis represents the accuracy value, the blue line represents the training accuracy, and the orange line represents the validation accuracy. The validation accuracy fluctuates suggests that the model overfitting the training data.

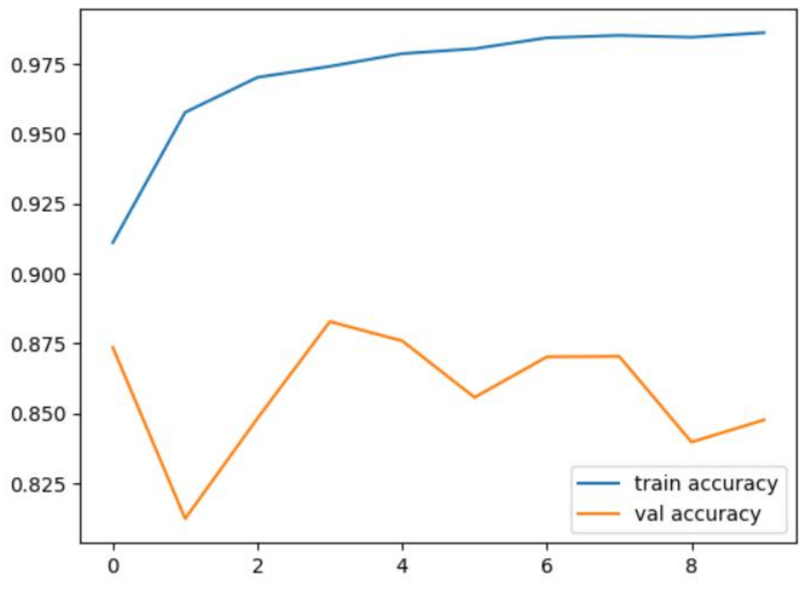


Figure 46: Graph of train & validation accuarcy

In conclusion, while achieving a commendable accuracy of 84.76% on the test data, it's notable that the model's loss value remains relatively high at 0.835. This indicates that there is still room for improvement in minimizing errors during classification. Given this observation, transitioning to a ResNet50 model could be a prudent choice.

6.1.2 Presenting the results of resnet50

Initially, our model achieved an accuracy of 73.2%. Through collaborative efforts and iterative modifications to our code, incorporating techniques such as

- Dataset Splitting:

  We adopted a prudent approach by splitting our dataset into three subsets: 70% for training, 20% for testing, and 10% for validation. This division ensured robust evaluation and prevented overfitting.

- Preprocessing Function:

  Leveraging the preprocess_input function, we standardized and prepared our ECG images, optimizing them for the neural network, thus enhancing model convergence during training.

- Batch Normalization:

  Introducing batch normalization layers enabled smoother and faster convergence by normalizing the activations of each layer, making the optimization process more stable and efficient.

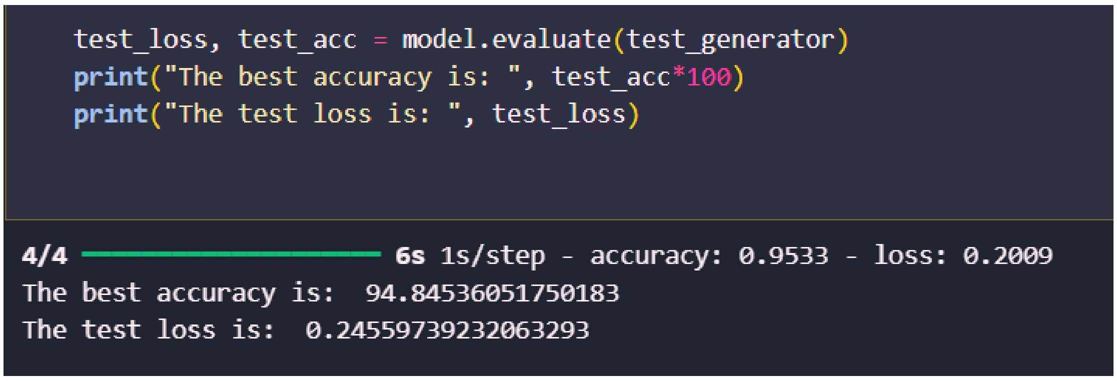
- Early Stopping:

  With EarlyStopping, we dynamically halted training when the model's performance plateaued, preventing overfitting and conserving computational resources.

- ReduceLROnPlateau:

  We incorporated the ReduceLROnPlateau callback, which intelligently adjusted the learning rate when the model's performance stagnated, ensuring steady progress towards convergence.

So significant improvements were made. These enhancements collectively contributed to boosting the accuracy to 94.84% while simultaneously reducing the loss value to 24.55%. This substantial progress underscores the effectiveness of our teamwork and the efficacy of the implemented strategies in refining the model's performance.



```
test_loss, test_acc = model.evaluate(test_generator)
print("The best accuracy is: ", test_acc*100)
print("The test loss is: ", test_loss)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.9533 - loss: 0.2009
The best accuracy is:  94.84536051750183
The test loss is:  0.24559739232063293
```

Figure 47: The results of resNet50 model (accuracy & loss value)

In this graph the y-axis shows loss. The loss starts at around 1.4 and trends downward to about 0.2 over the course of the training process. There appears a gap between the training loss and the validation loss throughout the process, though both trend downward.

In general, loss refers to how well a model is performing on a specific task. Lower loss indicates better performance. The training loss represents the model’s performance on the data it is trained on, while the validation loss represents the model’s performance on unseen data.

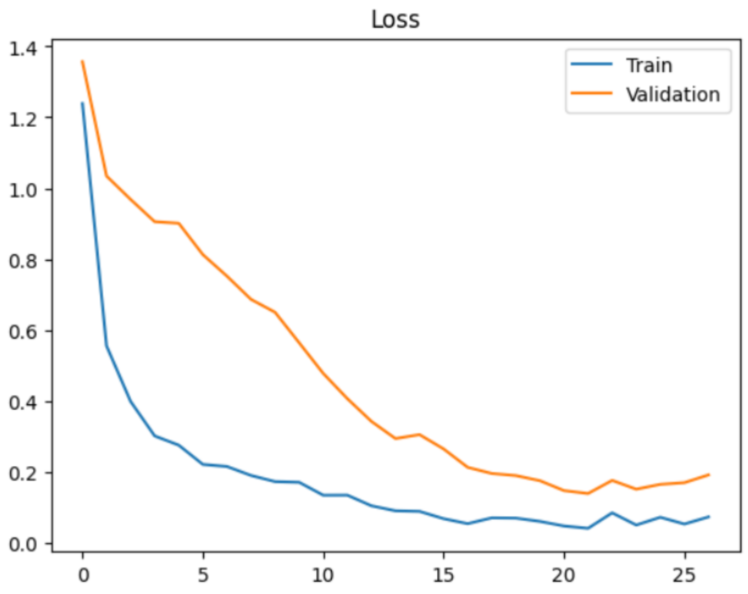


Figure 48: Graph of train & validation loss

Here a graph showing the accuracy of ResNet50 model. The y-axis shows accuracy, and the x-axis shows epochs, which are iterations of training a model on a dataset. There are two lines on the graph:

A blue line labeled "Train" shows the accuracy of the model on the training data. As the number of epochs increases, the training accuracy increases. This suggests that the model is learning the patterns in the training data.

A red line labeled "Validation" shows the accuracy of the model on the validation data. The validation data is a separate dataset from the training data that is used to evaluate how well the model generalizes to unseen data. In the graph, the validation accuracy is lower than the training accuracy. This could be a sign of a little overfitting.

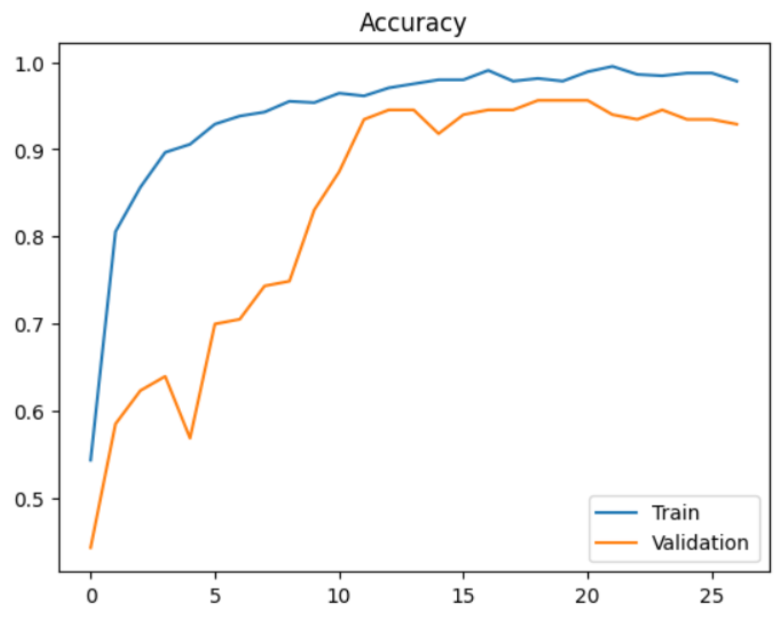


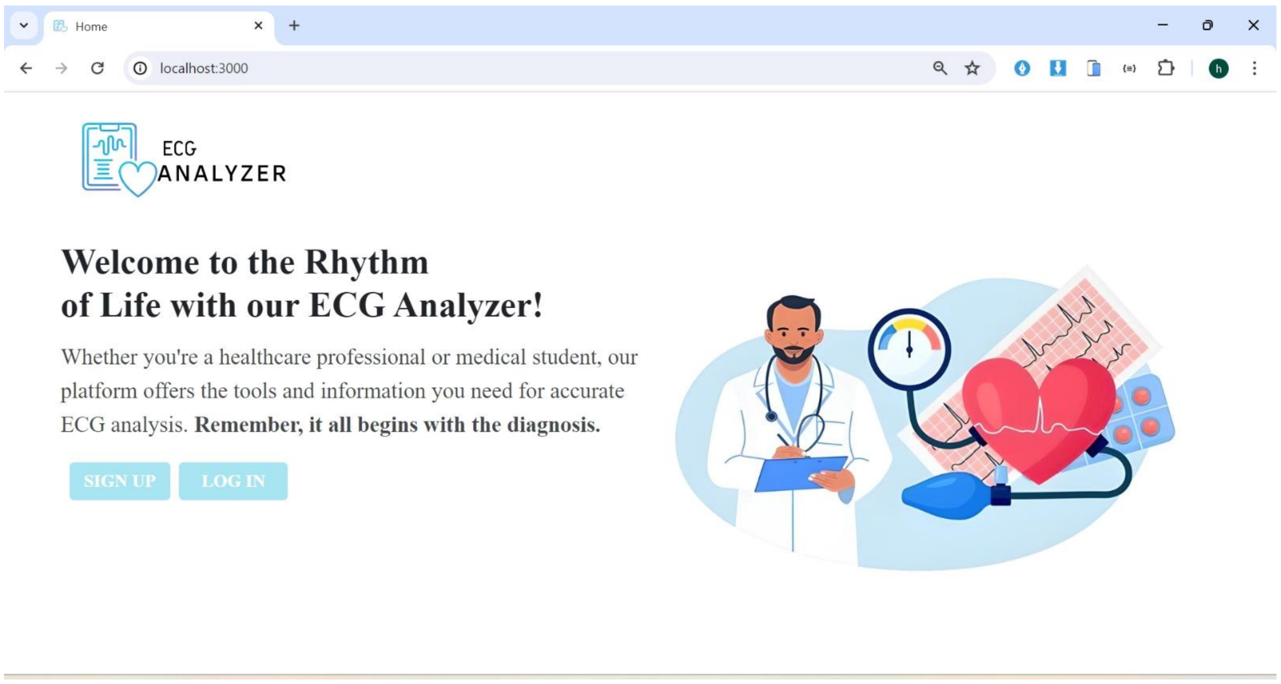
Figure 49: Graph of train & validation accuracy

Due to its remarkable performance enhancements in ResNet50 model, our team unanimously decided to integrate this refined model into our website. The choice of ResNet50 for the ECG analyzer website is driven by its ability to handle small datasets effectively through transfer learning, its robust architecture that prevents vanishing gradient problems, and its proven success in medical imaging tasks. By leveraging ResNet50, the ECG analyzer can provide accurate classification results and serve as an educational tool for medical students and healthcare professionals, ultimately contributing to improved diagnosis and learning outcomes.

6.2 User interfaces

The system interfaces form the cornerstone of user interaction with the web application, ensuring users can effectively engage with its functionalities. This section provides an overview of the interface design. Key elements include responsive design through media queries, user-friendly forms, consolidated information displays, and intuitive icons and buttons. These components are crafted to deliver a consistent, engaging, and accessible experience across all devices.

- The home page that will appear when visitors open ECG analyzer website. It contains signup and login buttons

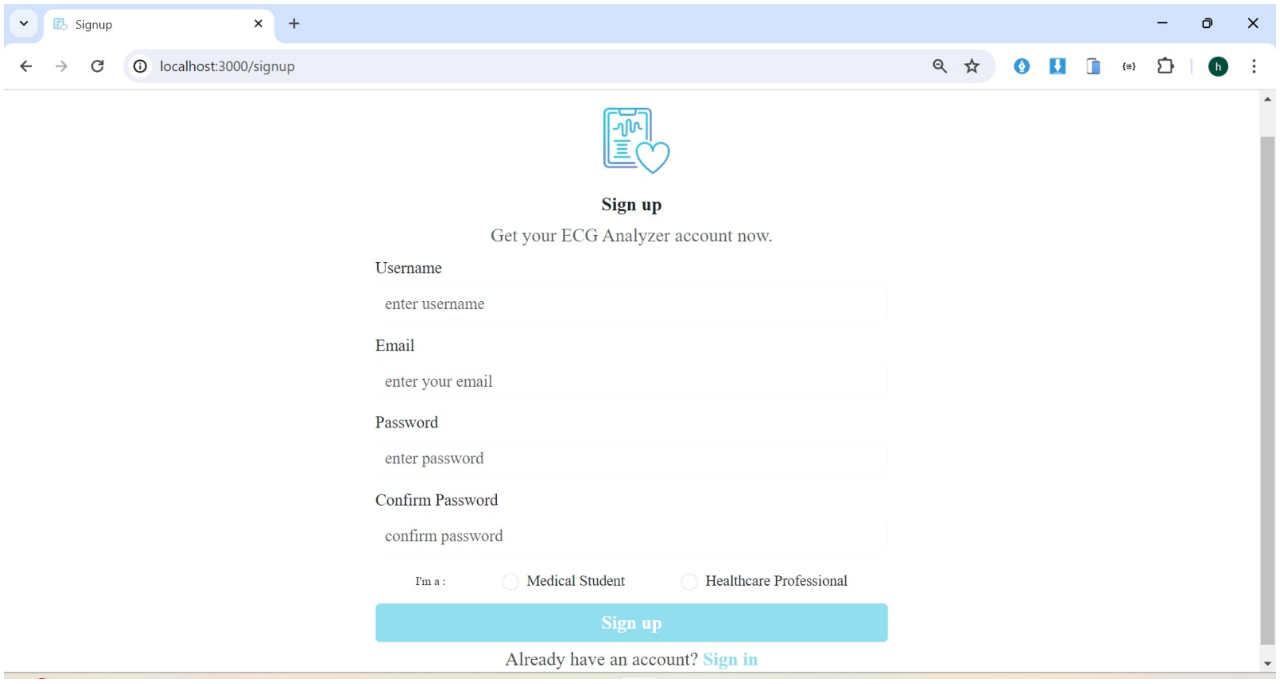


Welcome to the Rhythm of Life with our ECG Analyzer!

Whether you're a healthcare professional or medical student, our platform offers the tools and information you need for accurate ECG analysis. **Remember, it all begins with the diagnosis.**

Figure 50: Home page

- Users should have account to interact with the website's components


Signup

localhost:3000/signup

Sign up

Get your ECG Analyzer account now.

Username

enter username

Email

enter your email

Password

enter password

Confirm Password

confirm password

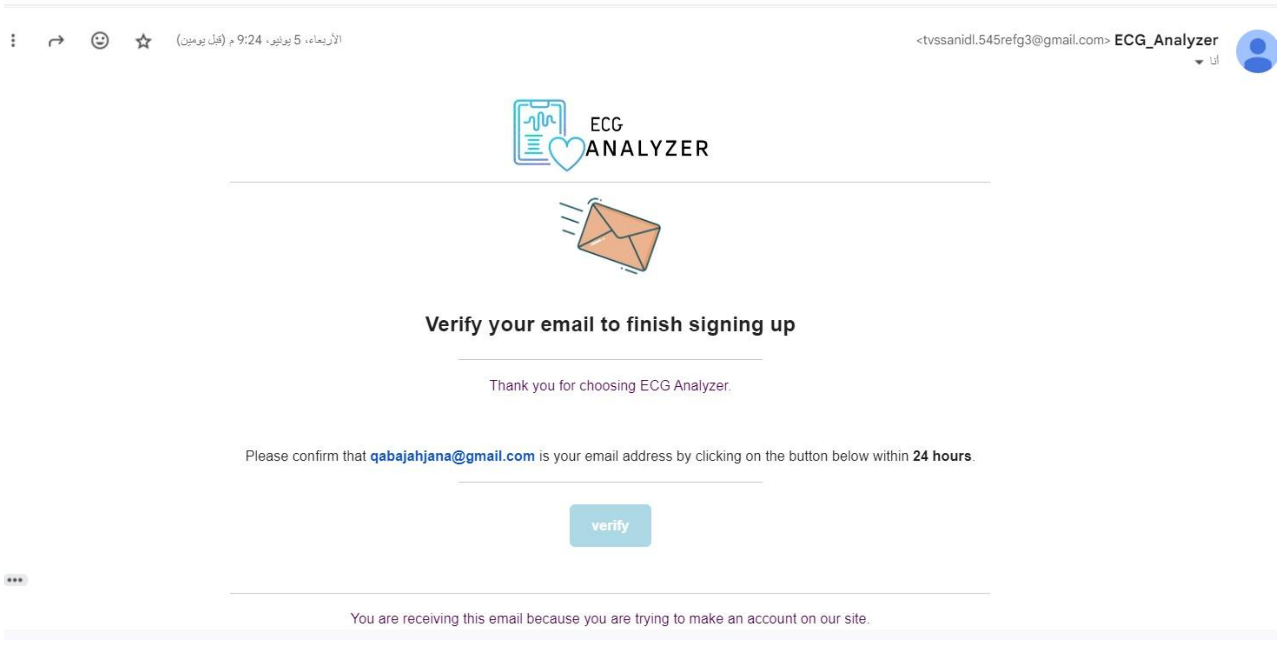
I'm a : Medical Student Healthcare Professional

Sign up

Already have an account? Sign in

Figure 51: Register (Sign up) page

- Users must verify their email via confirm link in the email to activate their account.


الأربعاء، 5 يونيو، 9:24 م (قبل يومين)

ECG_Analyzer <tvssanidl.545refg3@gmail.com>

أنا

ECG ANALYZER

Verify your email to finish signing up

Thank you for choosing ECG Analyzer.

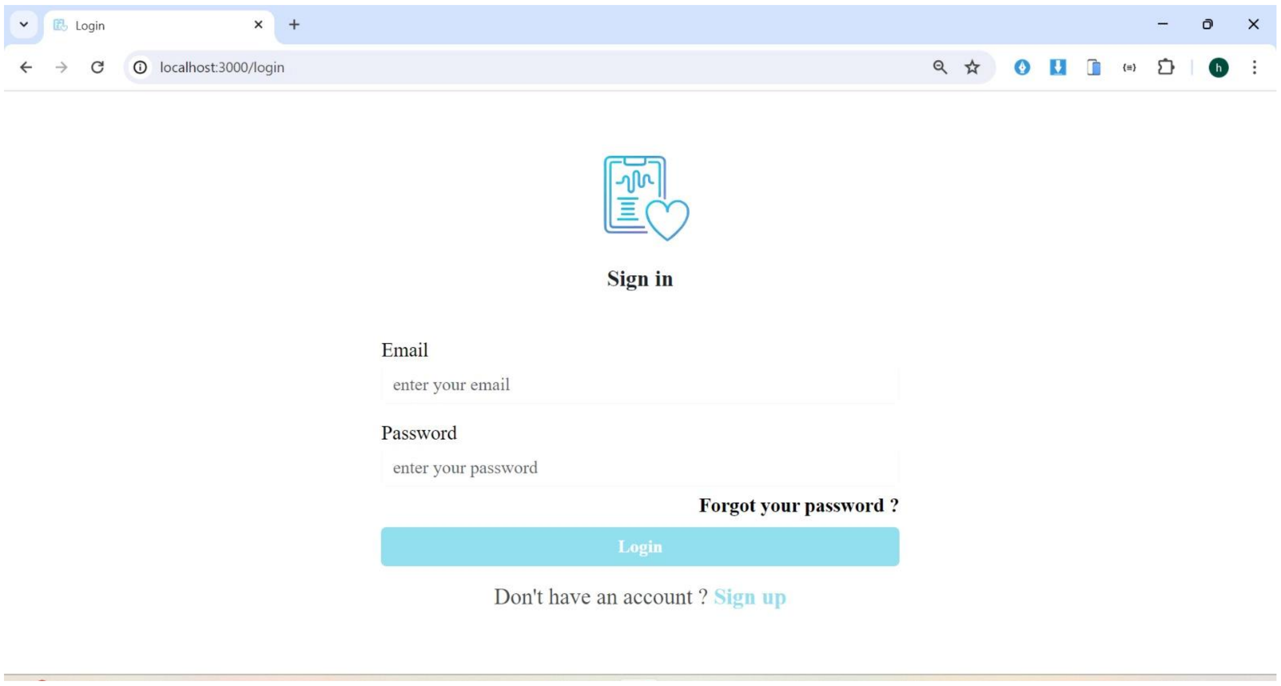
Please confirm that **qabajahjana@gmail.com** is your email address by clicking on the button below within **24 hours**.

verify

You are receiving this email because you are trying to make an account on our site.

Figure 52: Confirm email for signing up

- Sign in page where the user login to website.



Login

localhost:3000/login

Sign in

Email

enter your email

Password

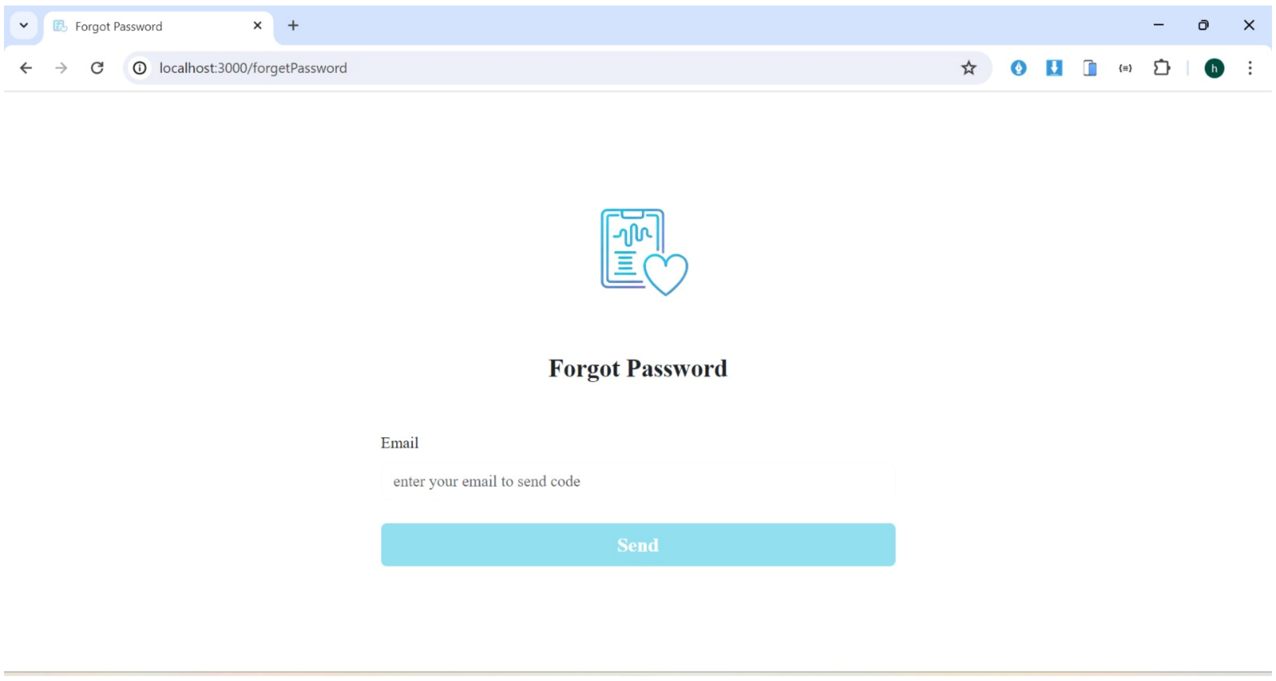
enter your password

Forgot your password ?

Don't have an account ? Sign up

Figure 53: Sign in (login) page

- If a user forgets their password, they can visit the "Forgot Password" page to reset it by providing their email address.



Forgot Password

localhost:3000/forgetPassword

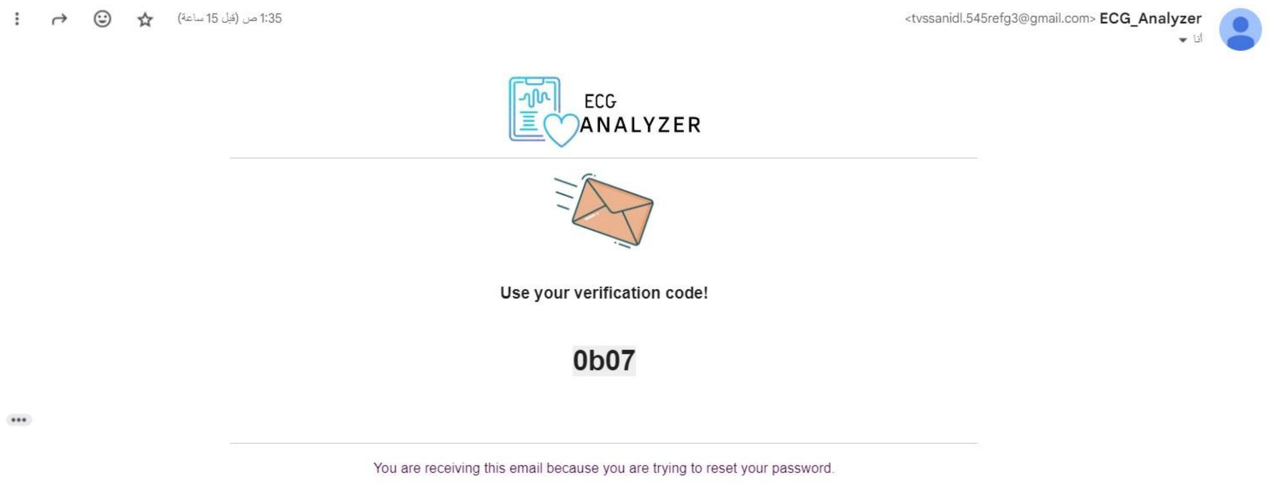
Forgot Password

Email

enter your email to send code

Figure 54: Forget password page

- They will receive a verification code via email.



1:35 ص (قبل 15 ساعة)

ECG_Analyzer <tvssanidl.545refg3@gmail.com>

أنا

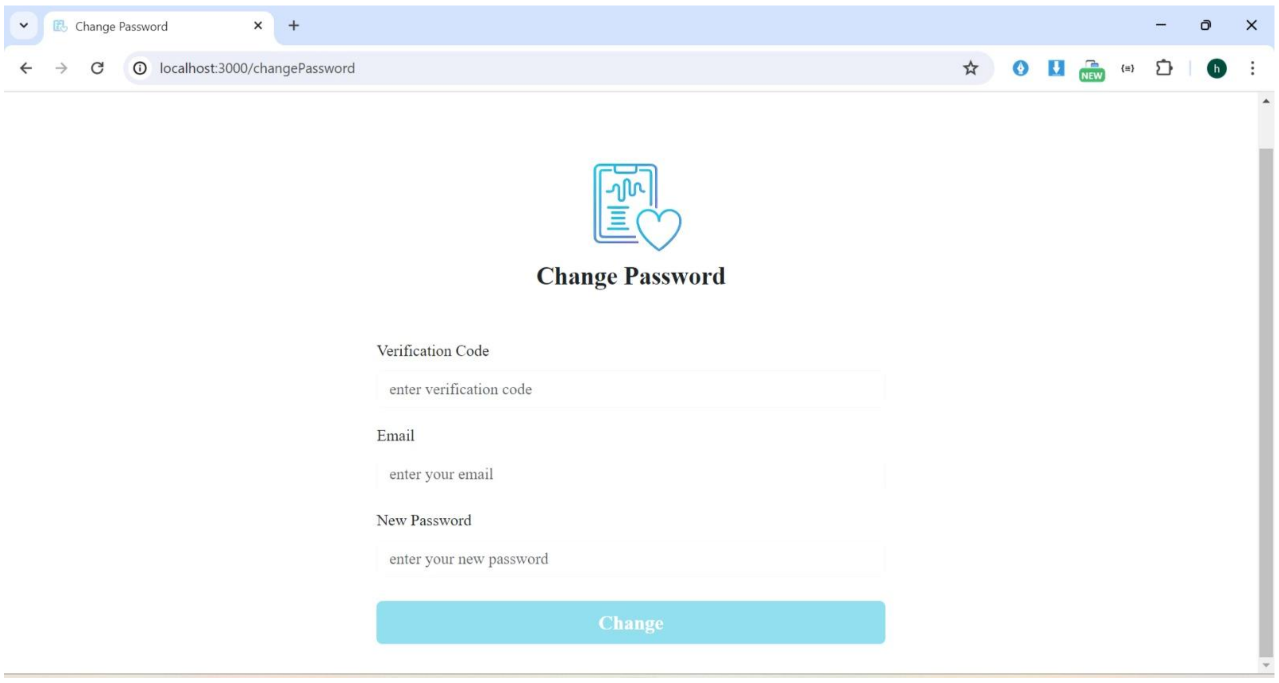
Use your verification code!

0b07

You are receiving this email because you are trying to reset your password.

Figure 55: Verification code email

- Then, they will enter the verification code along with a new password.



Change Password

localhost:3000/changePassword

Change Password

Verification Code

enter verification code

Email

enter your email

New Password

enter your new password

Change

Figure 56: Change password page

- After logging in, users can upload ECG images.
- Medical student have “Explanation” button.

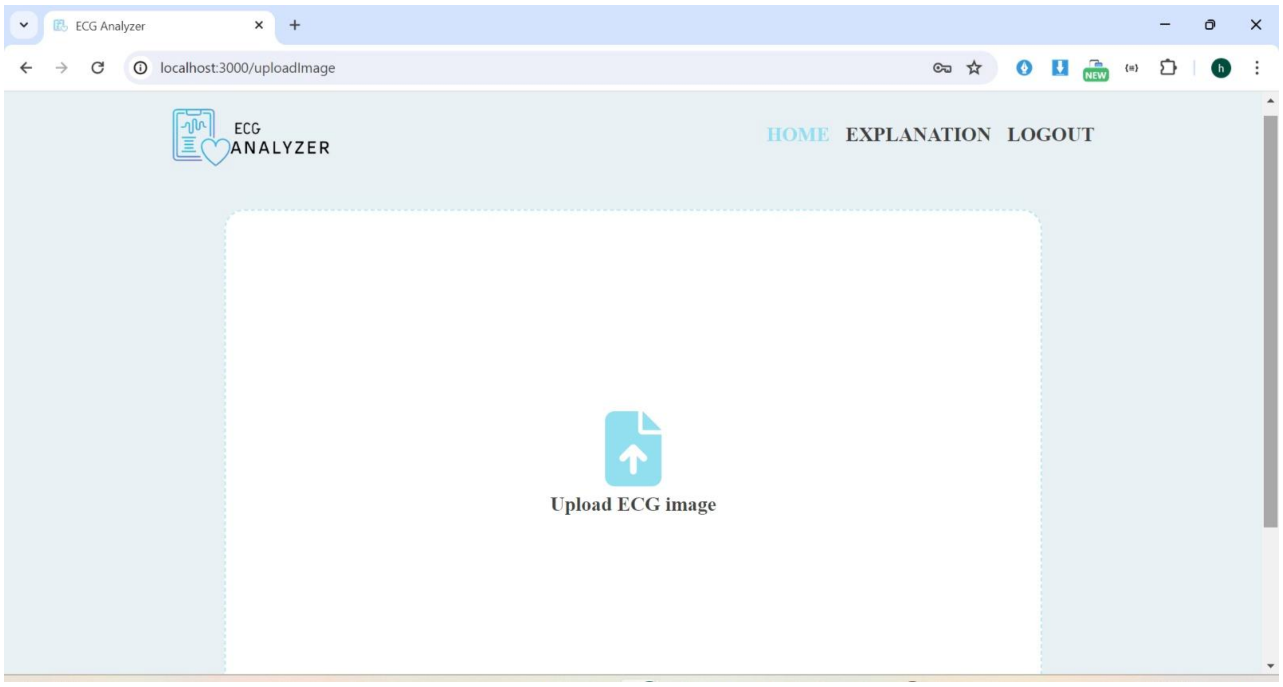


Figure 57: Upload ECG image page for medical students

- Whereas healthcare professionals do not.

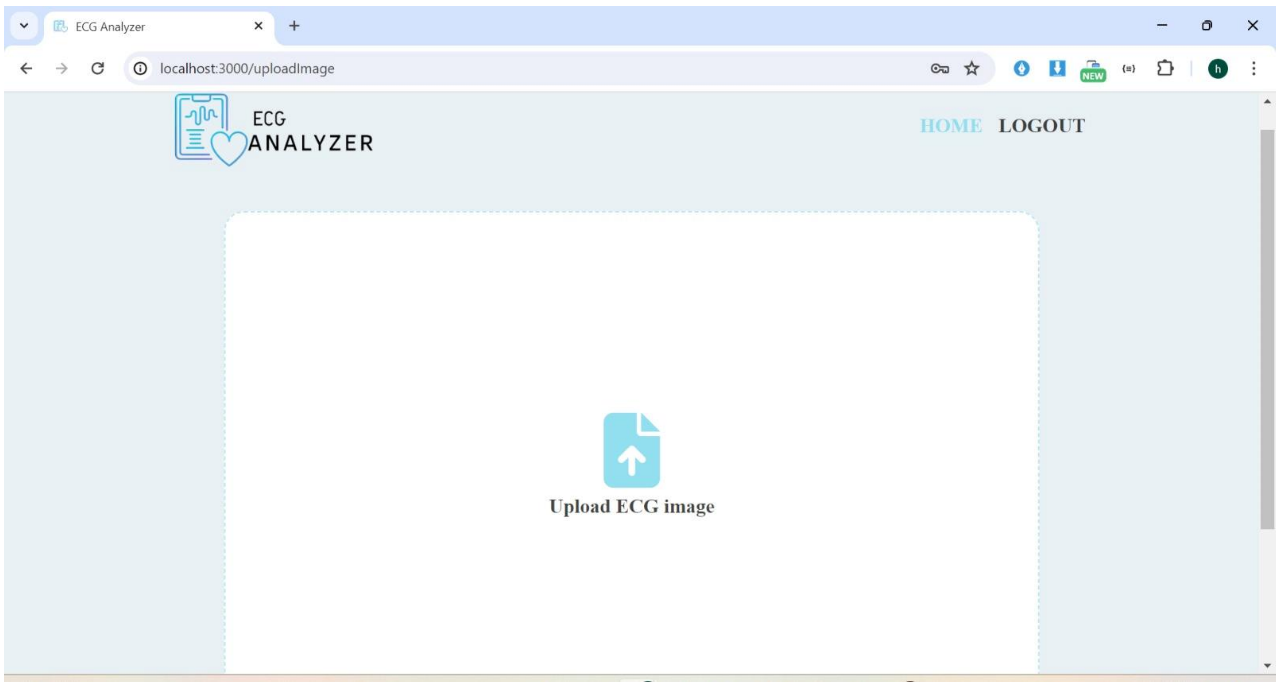


Figure 58: Upload ECG image page for health care professionals

- First, users select an image from their devices.

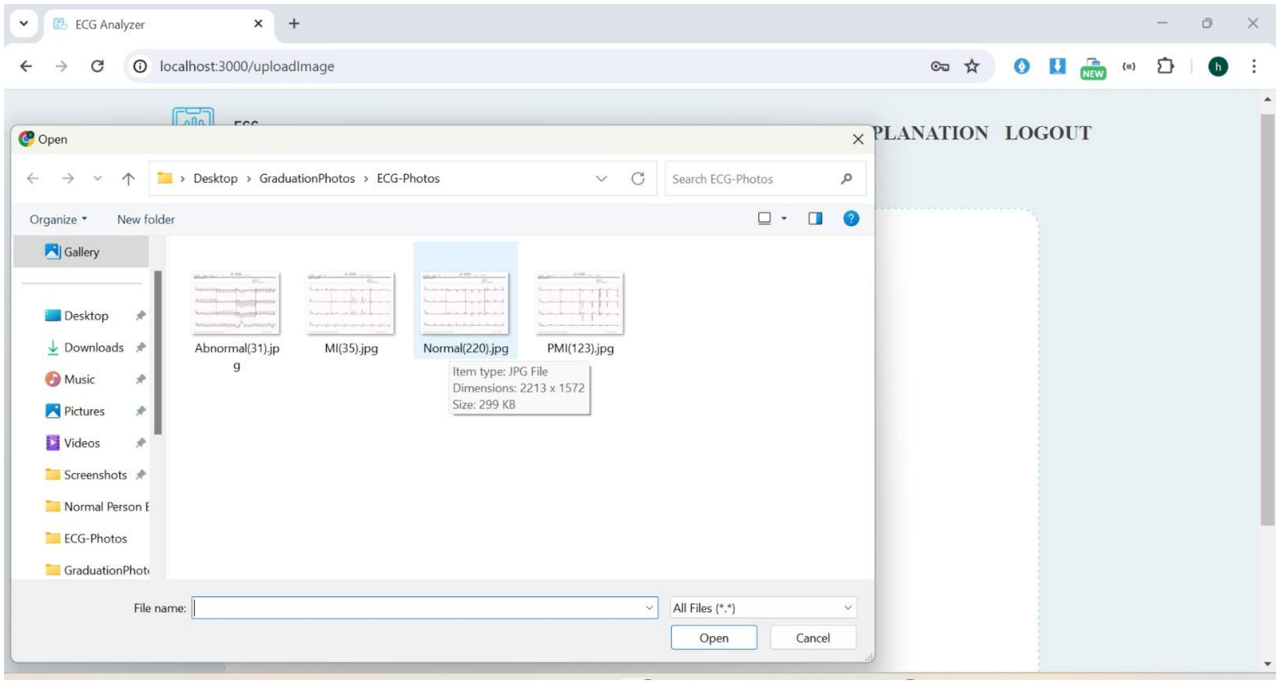
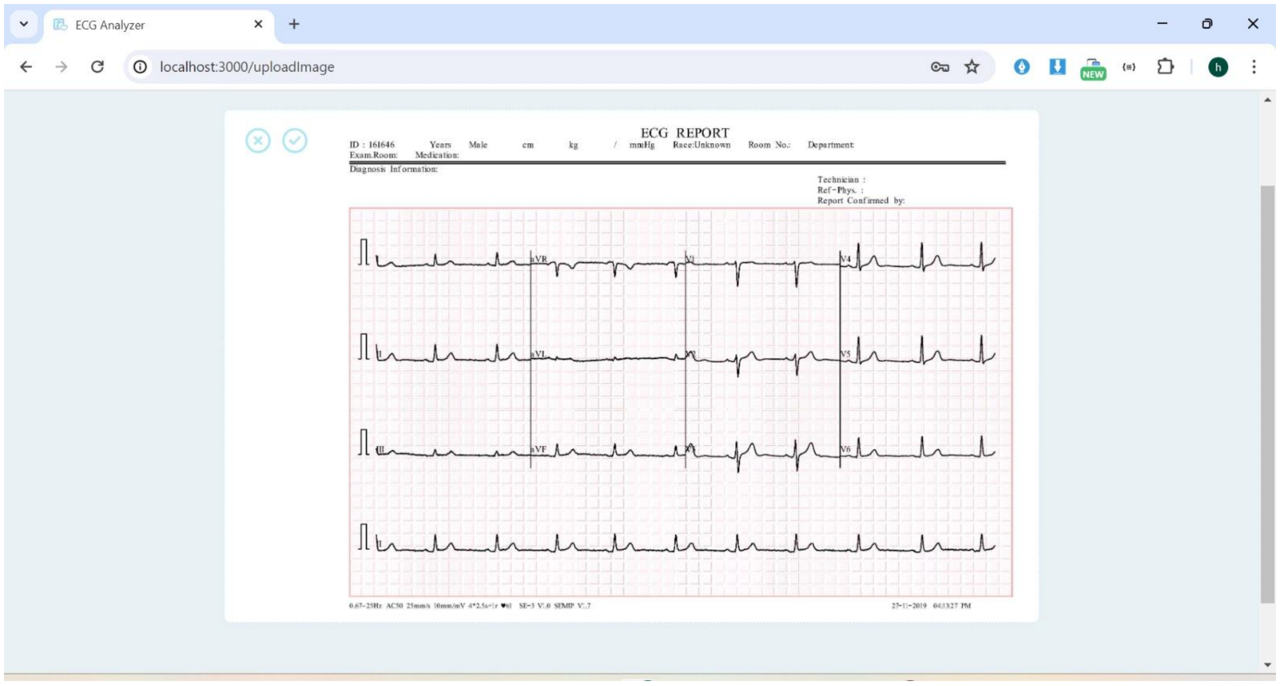


Figure 59: Select image to upload

- They can view the ECG image and then decide whether to confirm or remove it.



ECG REPORT

ID : 161646 Years Male cm kg / mmHg Race:Unknown Room No.: Department:
Exam.Room: Medication:

Diagnosis Information:

Technician :
Ref-Phys. :
Report Confirmed by:

Figure 60: Making a decision for confirm sending image or cancel it

- The system will analyze the image and display the result.

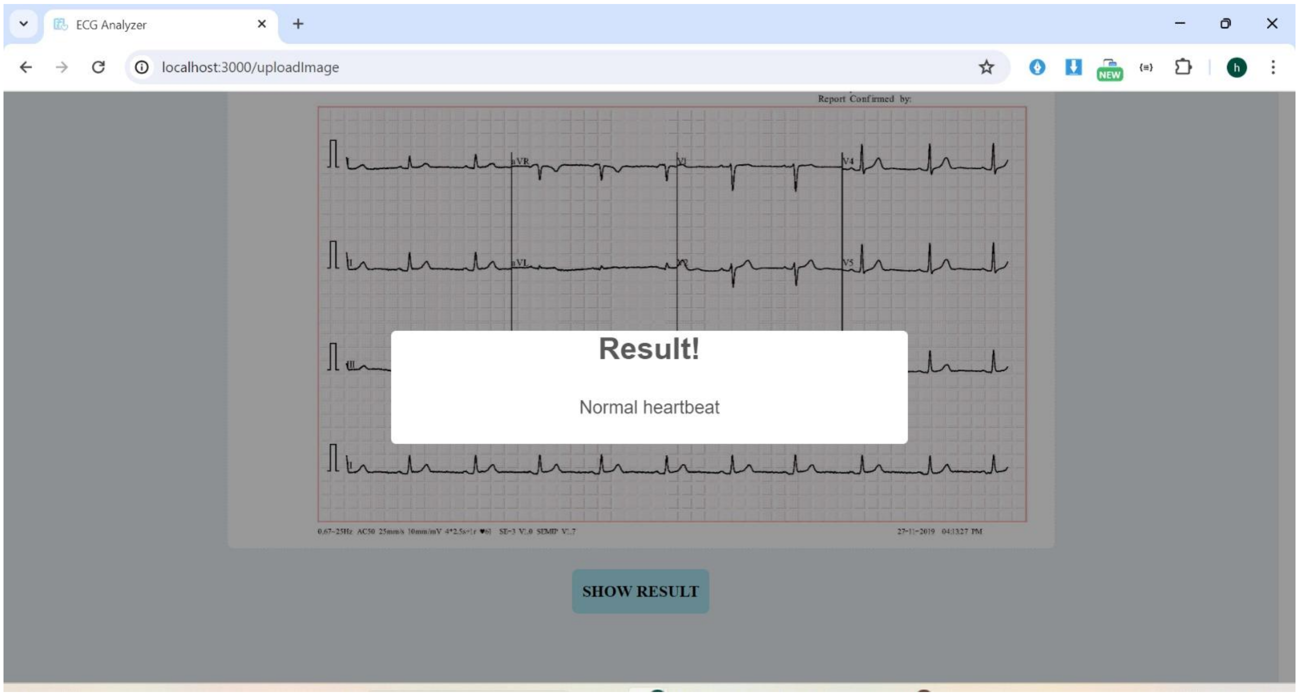
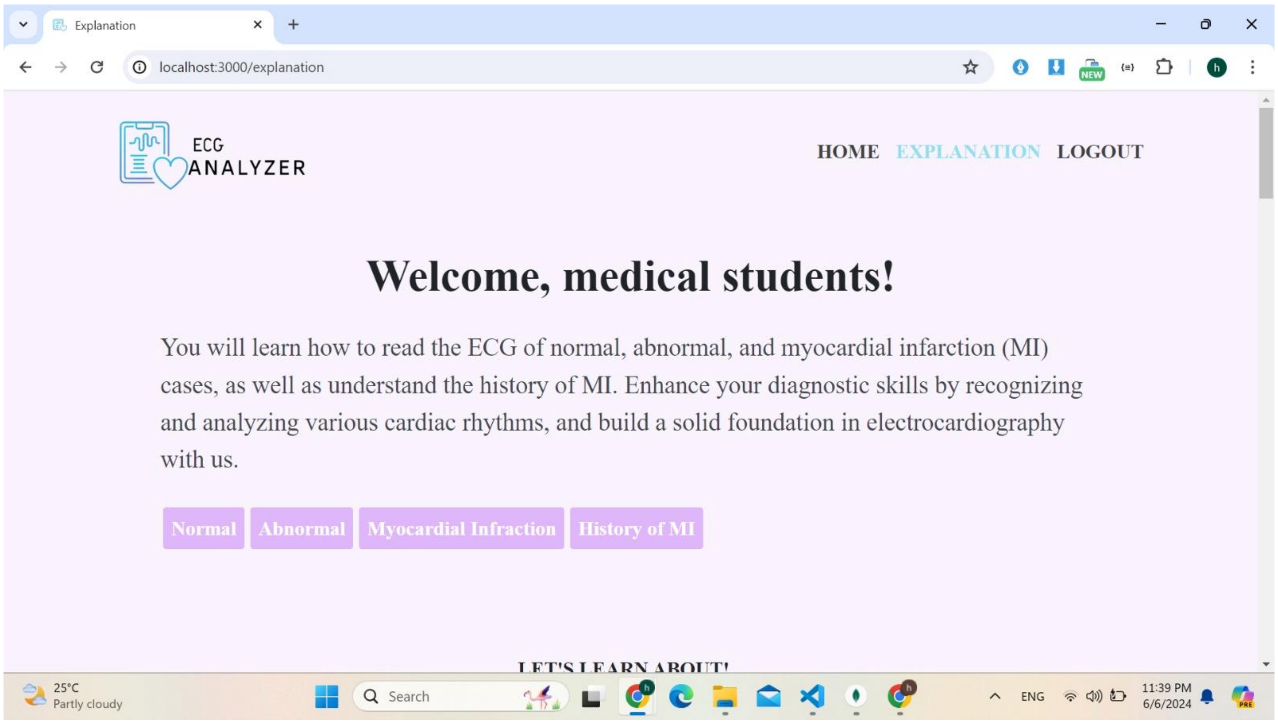


Figure 61: Show the result (medical diagnosis)

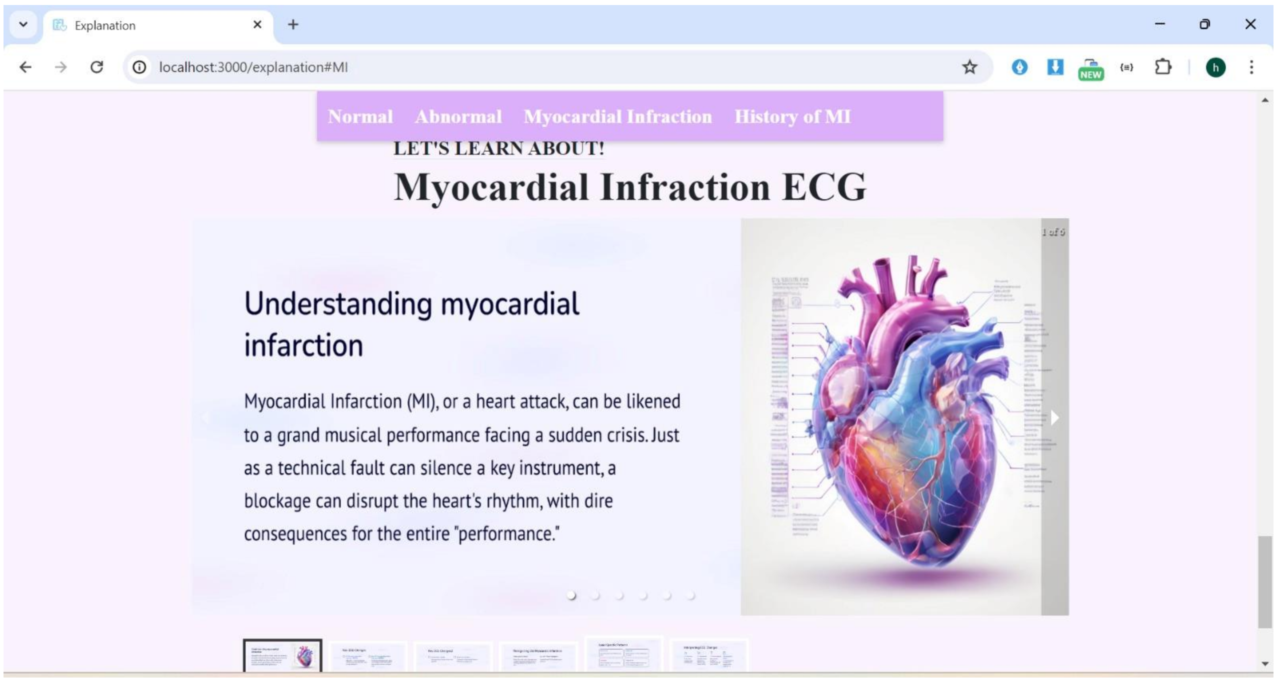
- Medical students can access a medical explanation page containing information about heart diagnosis and how to interpret ECG images.



Welcome, medical students!

You will learn how to read the ECG of normal, abnormal, and myocardial infarction (MI) cases, as well as understand the history of MI. Enhance your diagnostic skills by recognizing and analyzing various cardiac rhythms, and build a solid foundation in electrocardiography with us.

Figure 62: View medical explanation page-part1


Explanation

localhost:3000/explanation#MI

LET'S LEARN ABOUT!

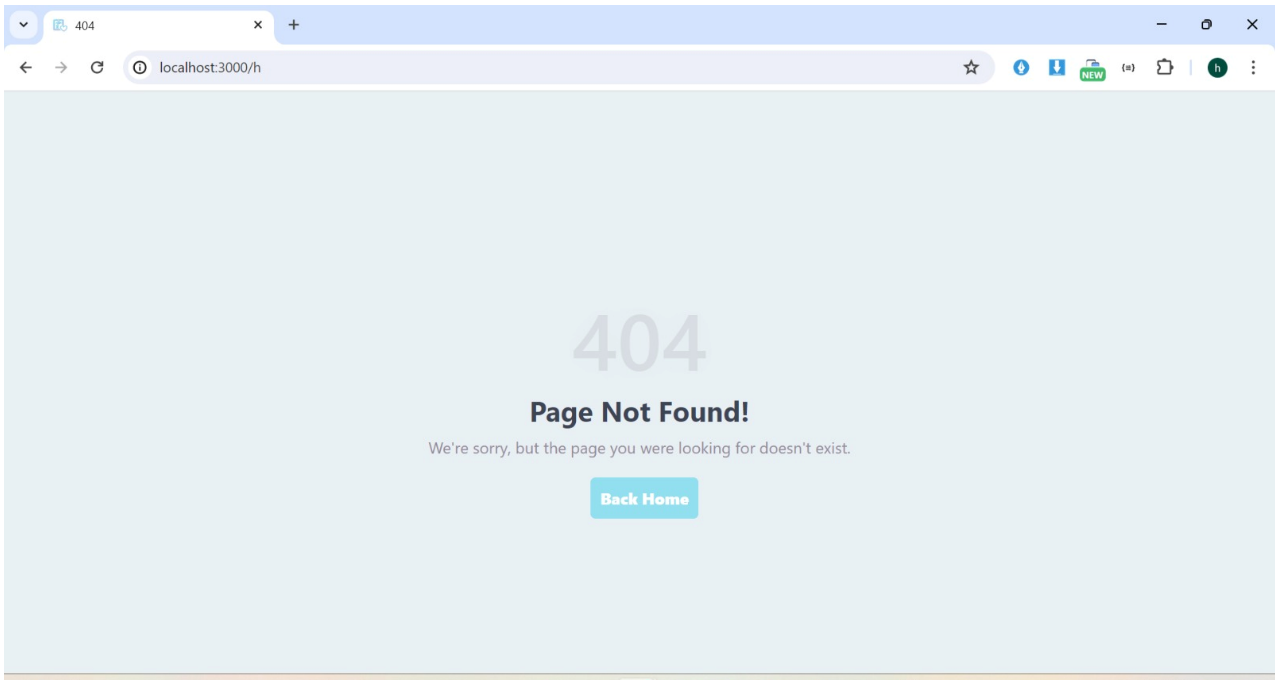
Myocardial Infraction ECG

Understanding myocardial infarction

Myocardial Infarction (MI), or a heart attack, can be likened to a grand musical performance facing a sudden crisis. Just as a technical fault can silence a key instrument, a blockage can disrupt the heart's rhythm, with dire consequences for the entire "performance."

Figure 63: View medical explanation page-part2

- If the user requests an incorrect path, a "Page Not Found" will appear.


404

localhost:3000/h

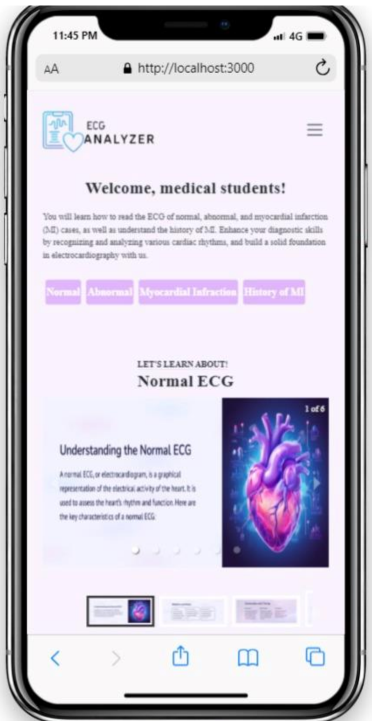
404

Page Not Found!

We're sorry, but the page you were looking for doesn't exist.

Figure 64: Page not found

- Our website is accessible on all devices regardless of screen size, as we have utilized media queries to ensure optimal display across various screen sizes.



11:45 PM 4G

http://localhost:3000

ECG ANALYZER

Welcome, medical students!

You will learn how to read the ECG of normal, abnormal, and myocardial infarction (MI) cases, as well as understand the history of MI. Enhance your diagnostic skills by recognizing and analyzing various cardiac rhythms, and build a solid foundation in electrocardiography with us.

Normal | Abnormal | Myocardial Infraction | History of MI

LET'S LEARN ABOUT!

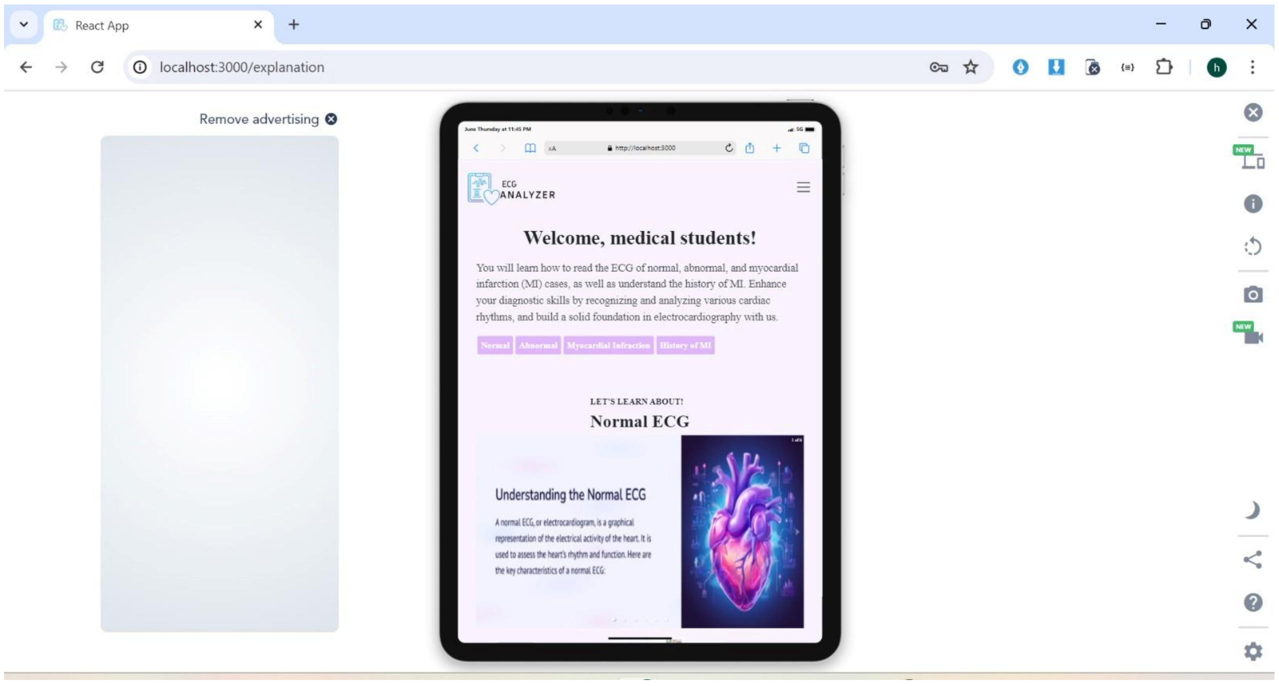
Normal ECG

Understanding the Normal ECG

A normal ECG, or electrocardiogram, is a graphical representation of the electrical activity of the heart. It is used to assess the heart's rhythm and function. Here are the key characteristics of a normal ECG.

1 of 6

Figure 65: ECG analyzer website on phones devices



React App

localhost:3000/explanation

Remove advertising

June Thursday at 11:45 PM 5G

http://localhost:3000

ECG ANALYZER

Welcome, medical students!

You will learn how to read the ECG of normal, abnormal, and myocardial infarction (MI) cases, as well as understand the history of MI. Enhance your diagnostic skills by recognizing and analyzing various cardiac rhythms, and build a solid foundation in electrocardiography with us.

Normal | Abnormal | Myocardial Infraction | History of MI

LET'S LEARN ABOUT!

Normal ECG

Understanding the Normal ECG

A normal ECG, or electrocardiogram, is a graphical representation of the electrical activity of the heart. It is used to assess the heart's rhythm and function. Here are the key characteristics of a normal ECG:

Figure 66: ECG analyzer website on medium-sized devices

6.3 Conclusion

In conclusion, the development of our ECG analyzer website represents a significant advancement in medical education and healthcare technology. By providing medical students and healthcare professionals with a platform to upload ECG images and receive automated classification results using deep learning techniques, our website offers a valuable tool for diagnosis support and learning. Moreover, the inclusion of comprehensive medical information and tutorials on interpreting ECGs enhances the educational value of the platform, empowering users to deepen their understanding of cardiac diagnostics.

Throughout the development process, we focused on usability, accessibility, and accuracy. The integration of media queries ensures that our website is accessible on all devices, providing a seamless experience for users regardless of screen size. Additionally, the implementation of deep learning algorithms specifically ResNet50 ensures reliable and efficient ECG classification, contributing to improved diagnosis accuracy and efficiency.

CHAPTER 7
FUTURE VISION

7.1 Online Deployment with Targeted Access and Security

Our future plan is to deploy the project online, making the platform accessible over the internet to enhance convenience and usability for users worldwide. To maintain the integrity and focus of our platform, access will be restricted to verified institutions, universities, and hospitals through a secure authentication system. This ensures that only authorized medical students and healthcare employees can utilize the website, fostering a professional and educational environment while preventing misuse by the general public. By combining online accessibility with targeted access and robust security measures, we aim to provide a reliable and effective resource for medical education and practice.

7.2 Patient Management Functionality

Future enhancements for our project include the addition of patient management functionality tailored for each doctor. This feature will allow doctors to manage their own medical records, facilitating efficient patient care and record-keeping.

7.3 Comprehensive Diagnostic Capabilities

We aim to broaden the diagnostic capabilities of our model to encompass all known heart diseases. This will involve continuous training and refinement of our algorithms with diverse and comprehensive datasets to ensure high accuracy across a wide range of cardiac conditions.

7.4 Improving the educational content for students

We envision significant enhancements to our medical explanation reference system to offer a more comprehensive and versatile learning experience. By incorporating multimedia elements such as instructional videos, interactive simulations, and detailed infographics, we aim to cater to diverse learning preferences and improve the accessibility of complex medical information. Additionally, as our dataset expands to cover more than four diagnoses.

References

[1] Abdullatif Husseini, Heidar Abu Ghosh, Nadim Barghuthi, Nahed Mikki, Niveen ME Abu-Rmeileh, Prof. Tarik M Ramahi (2009) – "Cardiovascular diseases, diabetes mellitus, and cancer in the occupied Palestinian territory." Published online at The Lancet Journal.

Retrieved from: 'https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(09)60109-4/fulltext'.

[2] Saloni Dattani, Veronika Samborska, Hannah Ritchie, and Max Roser (2023) - "Cardiovascular Diseases." Published online at OurWorldInData.org.

Retrieved from: 'https://ourworldindata.org/cardiovascular-diseases'.

[3] David A. Cook, So-Young Oh, and Martin V. Pusic (2020) - "Accuracy of Physicians' Electrocardiogram Interpretations." Published online at PubMed Central.

Retrieved from: 'https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7522782/'.

[4] Keyvan Amini, Alireza Mirzaei, Mirtohid Hosseini, Hamed Zandian, Islam Azizpour, and Yagoob Haghi (2022) - "Assessment of electrocardiogram interpretation competency among healthcare professionals and students of Ardabil University of Medical Sciences: a multidisciplinary study." Published online at PubMed Central. Retrieved from: 'https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9179219/'.

[5] Brad Munt, James M Brophy, John Rottger, Lyall Higginson, Merril L Knudtson, Rob Beanlands (2006) – "Treating the right patient at the right time: Access to specialist consultation and noninvasive testing." Published online at the National Library of Medicine.

Retrieved from: 'https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2569014/'.

[6] John Lynn Jefferies, Academic Editor (2023) - "Current and Future Use of Artificial Intelligence in Electrocardiography." Published online at PubMed Central.

Retrieved from: 'https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10145690/'.

[7] Kachuee, M., Fazeli, S., & Sarrafzadeh, M. (2018, June). ECG heartbeat classification: A deep transferable representation. In 2018 IEEE international conference on healthcare informatics (ICHI) (pp. 443-444). IEEE. Published online at Cornell University.
Retrieved from: '**https://arxiv.org.com**' **or** '**https://www.kaggle.com/datasets/ecg-image-data**'.

[8] Khan, Ali Haider; Hussain, Muzammil (2021), "ECG Images dataset of Cardiac Patients ", Mendeley Data, V2, doi: 10.17632/gwbz3fsgp8.2. University of Management and Technology. Published online at Mendeleye data.
Retrieved from: 'www.data.mendeley.com/datasets'.

[9] PMcardio-ECG Analysis: 'https://www.powerfulmedical.com/pmcardio'.

[10] ECG+ | Analyzer for QTc & HRV: 'www.ECG+ | Analyzer for QTc & HRV.apple.com'.

[11] ECG for Doctors: 'www.ECG for Doctors.apps.apple.com'.

[12] Datt, L., Zhang, J., Humaidi, A.J. et al (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53. Published online at Springer Open.

Retrieved from: 'www.journalofbigdata.springeropen.com'.

[13] Lavian, A. (2019). Transfer learning with a small data set- "nanos gigantum humeris insidentes". Published online at Towards Data Science.
Retrieved from: ' www.towardsdatascience.com'.

[14] Yang, Y., Zhang, L., Du, M., Bo, J., Liu, H., Ren, L., Li, X., Deen, M. J. (2021). A comparative analysis of eleven neural networks architectures for small datasets of lung images of COVID-19 patients toward improved clinical decisions. Library Computers in Biology and Medicine, Volume: 139. Published online at ScienceDirect. Retrieved from: 'www.sciencedirect.com'.

[15] Changchong, L., Weihai L., (2018). Ship Classification in High-Resolution SAR Images via Transfer Learning with Small Training Dataset. Department of Electronic Engineering and Information Science, University of Science and Technology of China. Published online at ResearchGate. Retrieved from: 'www.researchgate.com'.

[16] Hall, J. E. (2011). Guyton and Hall textbook of medical physiology Book (12th ed.). Saunders.