TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cardiovascular diseases (heart diseases) are the leading cause of death worldwide. The earlier they can be predicted and classified; the more lives can be saved. An electrocardiogram (ECG) is a common, inexpensive, and non-invasive tool for measuring the electrical activity of the heart and is used to detect cardiovascular disease. In this article, the power of deep learning techniques was used to predict the four major cardiac abnormalities: Abnormal heartbeat, Myocardial Infarction, History of Myocardial Infarction, and normal person classes using the public ECG images dataset of Cardiac Patients. First, the transfer learning approach was investigated using the pre-trained deep neural networks VGG16, RESNET50, and InceptionV3. The aforementioned pre-trained models were used as feature extraction tools for traditional machine learning algorithms, namely Support Vector Machine, K-nearest neighbours, Random Forest, and Naïve Bayes. Artificial intelligence plays an important role in improving the quality of life. In particular, early detection of diseases can help save lives. In this work, The proposed new lightweight CNN architecture has improved the accuracy rate of cardiovascular disease classification to 98.23% compared with the existing state-of-the-art methods.

Keywords: Cardiovascular, Deep Learning, Electrocardiogram (ECG) Images, Feature Extraction, Machine Learning, Transfer Learning.

CHAPTER-1 INTRODUCTION

Cardiovascular diseases (CVDs) remain the leading cause of mortality worldwide, accounting for millions of deaths each year. Early detection and timely intervention are critical to reducing the risk of severe cardiac events such as heart attacks and strokes. One of the most accessible and widely used diagnostic tools in clinical cardiology is the electrocardiogram (ECG), which records the electrical activity of the heart and helps in identifying various cardiac abnormalities.

However, manual interpretation of ECG signals is often time-consuming, prone to human error, and requires significant expertise. With the rapid advancement in artificial intelligence, particularly in machine learning (ML) and deep learning (DL), automated analysis of ECG data has gained substantial traction. These techniques offer the potential to enhance diagnostic accuracy, speed, and consistency by learning complex patterns from large datasets of ECG recordings.

This study aims to explore and implement ML and DL-based models for the automatic detection of cardiovascular diseases using ECG signals. By leveraging algorithms such as Support Vector Machines (SVM), Random Forest, Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks, the objective is to develop a robust and efficient system capable of identifying cardiac anomalies with high precision. The integration of such intelligent systems in healthcare can assist clinicians in making informed decisions and ultimately improve patient outcomes.

Cardiovascular diseases (CVDs) are among the most serious and prevalent health concerns globally, representing the leading cause of death in both developed and developing countries. According to the World Health Organization (WHO), an estimated 17.9 million people die each year due to CVDs, comprising 31% of global deaths. These diseases include a range of conditions such as coronary artery disease, arrhythmias, heart failure, and myocardial infarction, many of which can be detected early through careful monitoring of heart activity.

The electrocardiogram (ECG or EKG) is a fundamental, non-invasive diagnostic tool used to record the electrical activity of the heart over a period of time. It provides crucial information about the rhythm, rate, and morphology of cardiac signals, which can indicate various forms of cardiac abnormalities. Despite its widespread use and effectiveness, interpreting ECG signals accurately requires a high level of medical expertise and experience. Moreover, in high-demand healthcare settings, manual analysis can be time-consuming and susceptible to diagnostic errors.

To address these challenges, the integration of **machine learning (ML)** and **deep learning (DL)** techniques into ECG analysis has emerged as a promising solution. Machine learning algorithms can be trained on large datasets to recognize patterns and classify data based on learned features. Deep learning, particularly neural network-based models like **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**, goes a step further by automatically learning hierarchical feature representations from raw input signals, eliminating the need for manual feature extraction.

In recent years, numerous studies have demonstrated the effectiveness of ML and DL models in diagnosing cardiovascular conditions such as atrial fibrillation, ischemia, bradycardia, and tachycardia, often achieving diagnostic performance comparable to or exceeding that of trained cardiologists. These models can process massive volumes of ECG data quickly and consistently, making them ideal candidates for real-time, automated, and remote cardiac monitoring systems.

This project focuses on designing and evaluating machine learning and deep learning models for the detection of cardiovascular diseases using ECG signal datasets. The objectives are:

- To preprocess and normalize ECG data for optimal performance.
- To extract relevant features using signal processing and DL techniques.
- To train and validate various ML and DL models for classifying different types of cardiovascular diseases.
- To compare the accuracy, sensitivity, specificity, and overall performance of these models.

By leveraging artificial intelligence in ECG interpretation, this research aims to contribute to the development of reliable diagnostic tools that can assist healthcare professionals, especially in areas with limited access to cardiologists. The ultimate goal is to improve early diagnosis, treatment planning, and patient care outcomes in the domain of cardiovascular health.

1.1 MOTIVATION

The motivation behind this project stems from the pressing need for more reliable, accurate, and efficient diagnostic tools in the field of cardiology. Cardiovascular diseases (CVDs), including arrhythmias, myocardial infarctions, and heart failure, continue to be the leading cause of death globally, accounting for approximately 17.9 million deaths annually, according to the World Health Organization. Early diagnosis is crucial for effective treatment and prevention of complications, yet current diagnostic workflows often rely heavily on manual interpretation of electrocardiograms (ECGs), which can be time-consuming and prone to human error—especially in resource-limited settings.

ECG is a non-invasive, cost-effective, and widely used diagnostic tool for assessing the electrical activity of the heart. However, interpreting ECG signals accurately requires significant clinical expertise. Variations in interpretation among healthcare providers, particularly in complex or borderline cases, can lead to misdiagnosis or delayed treatment. This highlights a critical gap that can be addressed by artificial intelligence (AI)-based solutions.

1.2 PROBLEM STATEMENT

Cardiovascular diseases remain a leading cause of mortality worldwide, with early and accurate detection being crucial for effective treatment and prevention. Traditional methods of diagnosing these conditions through ECG images can be timeconsuming and prone to human error. This project aims to enhance the diagnostic process by leveraging machine learning and deep learning techniques to analyze ECG images more efficiently and accurately. By developing advanced algorithms capable of identifying and classifying various cardiovascular conditions, the goal is to improve diagnostic accuracy, reduce processing time, and ultimately contribute to better patient outcomes.

1.3 PURPOSE

The purpose of this project is to develop and implement advanced machine learning and deep learning models capable of automatically analyzing ECG (electrocardiogram) images for the detection and classification of various cardiovascular diseases. The project is motivated by the growing demand for accurate, efficient, and scalable diagnostic tools in the field of cardiology, where early detection is critical to successful treatment outcomes. By leveraging powerful algorithms such as convolutional neural networks (CNNs) and other deep learning architectures, the system aims to identify complex patterns in ECG signals that may indicate conditions like arrhythmias, myocardial infarction, and other cardiac abnormalities. The integration of AI into ECG analysis seeks to reduce diagnostic errors, minimize clinician workload, and provide decision support that complements the expertise of medical professionals.

1.4 SCOPE

The scope of this project involves the creation and implementation of machine learning and deep learning models for the automated detection and classification of cardiovascular diseases using ECG (electrocardiogram) images. This includes a comprehensive workflow starting with the acquisition and preprocessing of ECG data, where raw images are cleaned, normalized, and prepared for model training. The project then focuses on developing and optimizing advanced algorithms—such as convolutional neural networks (CNNs) and other deep learning architectures—to accurately identify patterns and anomalies associated with various cardiac conditions, including arrhythmias, myocardial infarctions, and atrial fibrillation. Key tasks also include fine-tuning model parameters, applying feature extraction techniques, and using validation strategies to ensure robust performance.

1.5 PROJECT OBJECTIVE

The primary objective of this project is to utilize advanced deep learning techniques to improve the detection and classification of major cardiac abnormalities by analyzing electrocardiogram (ECG) images. Cardiovascular diseases remain a leading cause of mortality worldwide, making timely and accurate diagnosis crucial for effective treatment and management. This project proposes a hybrid methodology that combines the power of deep learning-based feature extraction with the robustness of traditional machine learning classifiers. Specifically, it leverages transfer learning using well-established, pre-trained convolutional neural networks (CNNs) such as VGG16, ResNet50, and InceptionV3. These models are adept at capturing intricate spatial features from ECG image data. The extracted features are then fed into conventional machine learning algorithms including Support Vector Machines (SVM), KNearest Neighbors (KNN), Random Forest, and Naïve Bayes for classification tasks. This twostage approach aims to capitalize on the high-level feature representations learned by deep networks while maintaining the interpretability and efficiency of traditional classifiers.

1.6 LIMITATIONS

1. Data Limitations: The accuracy of predictions depends on the quality and balance of the ECG dataset, which can affect the model's generalizability.
2. Model Complexity: Combining pre-trained deep learning models with traditional algorithms can result in computational inefficiencies and deployment challenges.

CHAPTER-2 LITERATURE SURVEY

TITLE : Transforming ECG Diagnosis: An In-depth Review of Transformer-based Deep Learning Models in Cardiovascular Disease Detection

AUTHORS : Zibin Zhao

ABSTRACT : The emergence of deep learning has significantly enhanced the analysis of electrocardiograms (ECGs), a non-invasive method that is essential for assessing heart health. Despite the complexity of ECG interpretation, advanced deep learning models outperform traditional methods. However, the increasing complexity of ECG data and the need for real-time and accurate diagnosis necessitate exploring more robust architectures, such as transformers. Here, we present an in-depth review of transformer architectures that are applied to ECG classification. Originally developed for natural language processing, these models capture complex temporal relationships in ECG signals that other models might overlook. We conducted an extensive search of the latest transformer-based models and summarize them to discuss the advances and challenges in their application and suggest potential future improvements. This review serves as a valuable resource for researchers and practitioners and aims to shed light on this innovative application in ECG interpretation.

TITLE : Opportunities and Challenges of Deep Learning Methods for Electrocardiogram Data: A Systematic Review

AUTHORS : Shenda Hong, Yuxi Zhou, Junyuan Shang, Cao Xiao, Jimeng Sun

ABSTRACT : The electrocardiogram (ECG) is one of the most commonly used diagnostic tools in medicine and healthcare. Deep learning methods have achieved promising results on predictive healthcare tasks using ECG signals. This paper presents a systematic review of deep learning methods for ECG data from both modeling and

application perspectives. We extracted papers that applied deep learning (deep neural network) models to ECG data published between January 1st, 2010, and February 29th, 2020, from Google Scholar, PubMed, and the Digital Bibliography & Library Project. We then analyzed each article according to three factors: tasks, models, and data. Finally, we discuss open challenges and unsolved problems in this area. The total number of papers extracted was 191. Among these papers, 108 were published after 2019. Different deep learning architectures have been used in various ECG analytics tasks, such as disease detection/classification, annotation/localization, sleep staging, biometric human identification, and denoising. The number of works on deep learning for ECG data has grown explosively in recent years. Such works have achieved accuracy comparable to that of traditional feature-based approaches, and ensembles of multiple approaches can achieve even better results. Specifically, we found that a hybrid architecture of a convolutional neural network and recurrent neural network ensemble using expert features yields the best results. However, there are some new challenges and problems related to interpretability, scalability, and efficiency that must be addressed. Furthermore, it is also worth investigating new applications from the perspectives of datasets and methods. This paper summarizes existing deep learning research using ECG data from multiple perspectives and highlights existing challenges and problems to identify potential future research directions.

TITLE : Machine Learning-Based Heart Disease Diagnosis: A Systematic Literature Review

AUTHORS : Md Manjurul Ahsan, Zahed Siddique

ABSTRACT : Heart disease remains a significant global health challenge and a leading cause of mortality worldwide. Recent advancements in machine learning (ML) have demonstrated the feasibility of early-stage heart disease detection using electrocardiogram (ECG) and patient data. However, both ECG and patient datasets often exhibit class imbalances, posing challenges for traditional ML models to perform unbiasedly. Over the years, researchers have proposed various data-level and algorithm-level solutions to address these issues. This study presents a systematic literature review (SLR) to uncover the challenges associated with imbalanced data in heart disease predictions. A meta-analysis was conducted using 451 referenced articles

acquired from reputable journals between 2012 and November 15, 2021. For in-depth analysis, 49 articles were selected, considering factors such as heart disease type, algorithms, applications, and solutions. The SLR findings reveal that current approaches encounter several open problems when dealing with imbalanced data, hindering their practical applicability and functionality. The study emphasizes the need for further research to enhance the interpretability and explainability of ML algorithms in heart disease diagnosis.

TITLE : CNN-Based Detection of Cardiovascular Diseases from ECG Images

AUTHORS : Irem Sayin, Rana Gursoy, Buse Cicek, Yunus Emre Mert, Fatih Ozturk, Taha Emre Pamukcu, Ceylin Deniz Sevimli, Huseyin Uvet

ABSTRACT : Electrocardiogram (ECG) signals are crucial for diagnosing and monitoring various cardiovascular diseases (CVDs). This research develops a robust algorithm to accurately classify ECG signals, even in the presence of environmental noise. The study proposes a one-dimensional (1D) convolutional neural network (CNN) with two convolutional layers, two down-sampling layers, and a fully connected layer. The 1D data is transformed into two-dimensional (2D) images to enhance classification accuracy. A 2D CNN model consisting of input and output layers, three 2D convolutional layers, three down-sampling layers, and a fully connected layer is applied. The proposed 1D and 2D models achieve classification accuracies of 97.38% and 99.02%, respectively, when tested on the Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) arrhythmia database. Both models outperform corresponding state-of-the-art classification algorithms, validating their effectiveness.

Cardiovascular diseases (CVDs) are a leading cause of death worldwide. Accurate detection of CVDs is crucial for early treatment. This study proposes an ensemble-based lightweight deep learning model for predicting CVDs from electrocardiogram (ECG) images. The model applies pre-processing methods to remove artifacts and improve ECG image quality. Four renowned transfer learning models are investigated to predict CVDs, and their performance is evaluated. A lightweight convolutional neural network (CNN) is designed to extract prominent features from ECG images. Classification algorithms with optimal parameters are applied for individual prediction. An optimized weight model is developed for the ensemble of classifiers for final prediction. The proposed model achieves the highest accuracy of 99.29% when tested on real ECG datasets, outperforming baseline methods.

TITLE : Interpretable Deep Learning for Automatic Diagnosis of 12-lead Electrocardiogram

AUTHORS : Dongdong Zhang, Xiaohui Yuan, Ping Zhang

ABSTRACT : Electrocardiogram (ECG) is a widely used reliable, non-invasive approach for cardiovascular disease diagnosis. With the rapid growth of ECG examinations and the insufficiency of cardiologists, accurate and automatic diagnosis of ECG signals has become a hot research topic. In this paper, we developed a deep neural network for multi-label classification of cardiac arrhythmias in 12-lead ECG recordings. Experiments on a public 12-lead ECG dataset showed the effectiveness of our method. The proposed model achieved an average area under the receiver operating characteristic curve (AUC) of 0.970 and an average F1 score of 0.813. The deep model showed superior performance than 4 machine learning methods learned from extracted expert features. Besides, the deep models trained on single-lead ECGs produce lower performance than using all 12 leads simultaneously. The best-performing leads are lead I, aVR, and V5 among 12 leads. Finally, we employed the SHapley Additive exPlanations (SHAP) method to interpret the model's behavior at both patient level and population level.

CHAPTER-3 SYSTEM ANALYSIS

3.1 INTRODUCTION

In the current era, systems are being developed for the automatic detection of cardiac-related issues. These systems predict high-accuracy results based on onedimensional ECG beat signals but are still not adopted as tools in healthcare institutes. The main areas that affect the success of these approaches, i.e., selection of features, extraction techniques, types of classification algorithms, and most importantly, the use of imbalanced data for classification can reduce the recognition accuracy of the minority class. Moreover, state-of-the-art studies represent the classical methods to show high accuracy in classification and detection tasks on one lead-based ECG image. Furthermore, the healthcare institutes used various ECG equipment, presenting results in nonuniform formats of ECG images. The state-of-the-art research was unable to propose a generalized methodology for nonuniform formats of ECG images. In this research, the main focus is to provide a novel automatic detection tool relatively similar and adaptable for cardiac hospitals to process lead-based ECG images.

In what can be termed as the information age, research on the methods that can be used in the timely diagnosis is extensive. Forecasting is important because it eases the control of the rate and probability of someone suffering from the disease and simultaneously makes it possible to reinforce the necessary measures to control the disease. Risk prediction of heart diseases is important because it aids in decision-making on lifestyle choices and the pros and cons of some habits such as smoking, obesity, and alcohol consumption. It all depends on the individual's vulnerability in getting a heart disease. Early diagnosis on the other hand makes treatment cheap which can reduce the complications as well as to reduce the cost of treatment. The revolution of the healthcare system is very rapid. For a long time, scientists have improvised automated methods that have made forecasting of the risk easy and more accurate. Many researchers developed deep learning algorithms for ECG arrhythmia classification. The computation efficiency and speed are achieved through a convolutional neural network for an individual patient.

3.2 EXISTING SYSTEM

According to the World Health Organization, cardiovascular diseases (heart diseases) are the leading cause of death worldwide. They claim an estimated 17.9 million lives each year, accounting for 32% of all deaths worldwide. About 85% of all deaths from heart disease are due to heart attacks, also known as myocardial infarctions (MI). Many lives can be saved if an efficient diagnosis of cardiovascular disease is detected at an earlier stage. Different techniques are used in the healthcare system to detect heart diseases, such as electrocardiogram (ECG), echocardiography (echo), cardiac magnetic resonance imaging, computed tomography, blood tests, etc. The ECG is a common, inexpensive, and non-invasive tool for measuring the electrical activity of the heart. A highly skilled clinician can detect heart disease from the ECG waves.
However, this manual process can lead to inaccurate results and is very time consuming.

• **Problems with the Existing system:**

On the other hand, deep learning, which is a subfield of machine learning, automatically extracts important features and patterns from the training datasets for the classification phase without the intervention of separate entities for feature extraction and selection. In deep learning, a model is created by constructing multiple hidden layers of NNs. Convolutional neural network (CNN) learning, which has achieved satisfactory results on image classification tasks.

3.3 PROPOSED SYSTEM

The power of deep learning and pre-trained networks can be used for feature extraction without having to retrain the whole network, transfer learning, and classification. In this research, the pre-trained networks, i.e., VGG16, RESNET50, and InceptionV3 are used as a transfer learning approach to study their performance in heart disease classification and as feature extraction for traditional machine learning methods for heart disease classification using ECG images.

- **Advantages of Proposed System :**

The proposed system offers several key advantages that significantly improve overall performance and user experience. By automating manual tasks, it enhances efficiency and reduces the chance of human error, leading to more accurate and reliable results. The system is designed to be scalable and flexible, allowing it to grow and adapt according to the evolving needs of users. Its user-friendly interface ensures that even those with minimal technical skills can operate it with ease, minimizing the need for extensive training. Additionally, the system provides real-time access to data, enabling quicker and better-informed decision-making. Enhanced security features protect sensitive information, while seamless integration with existing platforms ensures smooth workflow and data consistency. Overall, these benefits contribute to cost savings, improved productivity, and more effective management within the organization.

CHAPTER-4 SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

- **Data Processing**:

  The system must preprocess ECG images to standardize their format and resolution for analysis.

- **Feature Extraction**:

  The system should use pre-trained models (VGG16, RESNET50, InceptionV3) to extract relevant features from ECG images.

- **Classification:**

  The system must classify ECG images into one of the four categories: abnormal heartbeat, myocardial infarction, history of myocardial infarction, or normal.

- **Integration with Machine Learning Algorithms:**

  The system should apply traditional machine learning algorithms (SVM,KNN, Random Forest, Naïve Bayes) on the extracted features to Enhance classification accuracy.

- **Performance Evaluation:**

  The system must provide metrics (e.g., accuracy, precision, recall) to evaluate the performance of the classification models.

4.2 NON-FUNCTIONAL REQUIREMENTS

- **Scalability:**

  The system should handle large volumes of ECG images efficiently and be scalable to accommodate increasing data sizes.

- **Accuracy:**

  The classification models should achieve high accuracy in detecting and differentiating cardiac abnormalities.

- **Usability:**

The system should be user-friendly, with an intuitive interface for inputting ECG

images and viewing results.

- **Security:**

The system must ensure the confidentiality and integrity of patient data during processing and storage.

- **Performance:**

The system should provide results in a reasonable time frame to ensure timely

diagnosis and intervention.

4.3 INTERFACE REQUIREMENTS

4.3.1 User Requirements :

- SGUI Applications
- Web frameworks
- Image processing
- Web scraping
- Test frameworks
- Multimedia

4.3.2 System Requirements:

- **Hardware Requirements:**

Processor : Any Update Processor

Ram : Min 4 GB Hard Disk : Min 250 GB

- **Software Requirements:**

Operating System	:	Windows family
Technology	:	Python 3.8
Front-end Technology	:	HTML, CSS, JS
Back-end Technology	:	MySQL
Code Development Tool	:	PyCharm IDE

CHAPTER-5 SYSTEM DESIGN

The system design for the detection of cardiovascular diseases using ECG with machine learning and deep learning methods involves a streamlined, modular pipeline. It begins with the **data acquisition module**, which collects ECG signals from public datasets or real-time sensors. These signals pass through a **preprocessing module** where noise is filtered, signals are normalized, and segmented into manageable windows or heartbeats. In the **feature extraction module**, traditional ML approaches rely on manually derived features such as RR intervals and PQRST peaks, while DL models like CNNs and LSTMs use raw ECG signals or images directly. The **classification module** applies ML models (e.g., SVM, Random Forest) or DL models (e.g., CNN, LSTM, hybrid CNN-LSTM) to categorize the data into normal or abnormal classes, such as arrhythmia or myocardial infarction. The final **interpretation module** generates outputs with confidence scores and visualizations, optionally using explainability tools like SHAP or Grad-CAM for clinical transparency. This system can be deployed as a desktop tool, cloud service, or mobile application and integrates easily with healthcare databases or IoT-based ECG devices for real-time diagnostics.

5.1 SYSTEM ARCHITECTURE

The system architecture for detecting cardiovascular diseases using ECG signals involves a sequence of integrated stages designed to ensure accurate and efficient diagnosis. The process begins with **ECG signal acquisition**, which can be sourced from publicly available datasets like MIT-BIH or PTB-XL, or obtained through real-time sensors and ECG devices. These signals, typically in the form of one-dimensional time-series data, are then passed through the **data preprocessing** phase. This stage involves noise removal using filters to eliminate baseline drift and power line interference, signal normalization to bring all signals to a common scale, and segmentation to divide long recordings into manageable parts for analysis.

Following preprocessing, the system enters the **feature extraction** phase. In traditional machine learning models, this involves extracting key characteristics from the ECG such as RR intervals, PQRST complex features, and heart rate variability (HRV). In contrast, deep learning models like CNNs and LSTMs can work directly on raw signals or spectrograms without the need for manual feature engineering, allowing them to learn complex patterns automatically.

Once features are prepared, they are passed into the **model training and classification** stage. Here, either machine learning algorithms (like Support Vector Machines, Random Forests, or k-Nearest Neighbors) or deep learning models (such as Convolutional Neural Networks, Long Short-Term Memory networks, or hybrid CNN-LSTM architectures) are employed to classify the input data. The output of this stage is the **prediction of cardiovascular conditions**, which may include categories such as normal rhythm, arrhythmia, myocardial infarction, or atrial fibrillation.

Finally, the system presents **results and interpretation**. This includes the predicted class along with confidence scores, and optionally, visual tools like SHAP or Grad-CAM can be used to interpret the model's decisions, enhancing clinical trust. This end-to-end pipeline can be extended with real-time monitoring, cloud-based inference, or integration into mobile health applications, providing scalable and accessible heart disease diagnostics for patients and healthcare providers.

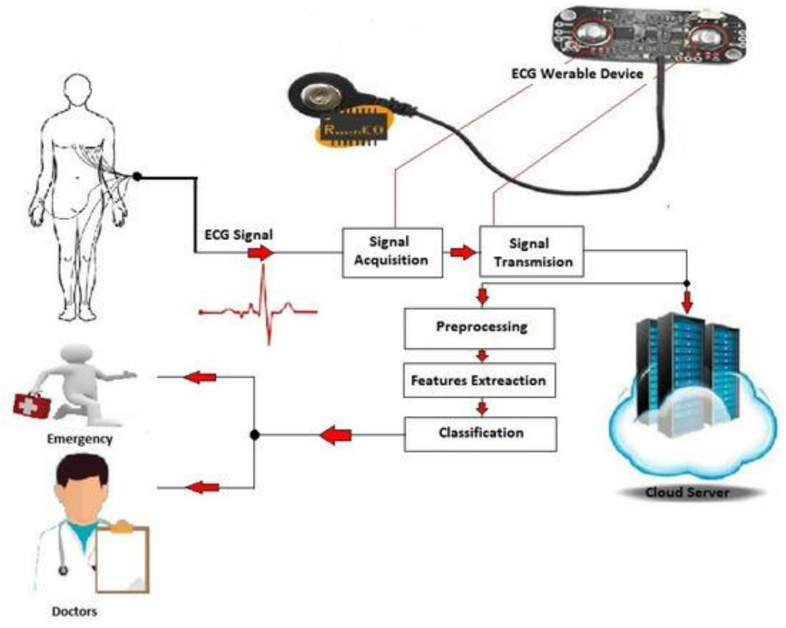


Fig :5.1.1 System Architecture

5.2 UML DIAGRAMS

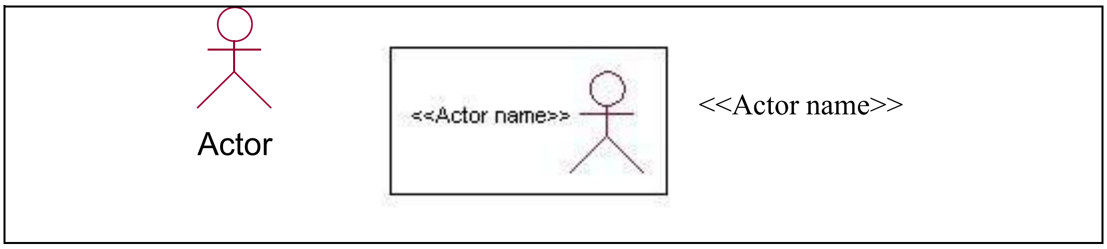
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, humanmachine interfaces, detailed design, processing logic, and external interfaces.

- **Global Use Case Diagrams:**

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases. Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

2 Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?

- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

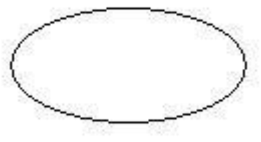
The actors identified in this system are:

a. **Systessm Administrator**
b. **Customer**
c. **Customer Care**

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor Use cases provide a means to:
- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will Be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar. This makes the description less ambiguous Questions to identify use cases:
- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

➢ Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows Construction of Usecase diagrams:

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations

Relationships in use cases:

1. **Communication:**

   The communication relationship of an actor in a usecase is shown by connecting the actor symbol to the usecase symbol with a solid path. The actor is said to communicate with the usecase.

2. **Uses:**

   A Uses relationship between the usecases is shown by generalization arrow from the usecase.

3. **Extends:**

   The extend relationship is used when we have one usecase that is similar to another usecase but does a bit more. In essence it is like subclass.

5.2.1 Use Case Diagram:

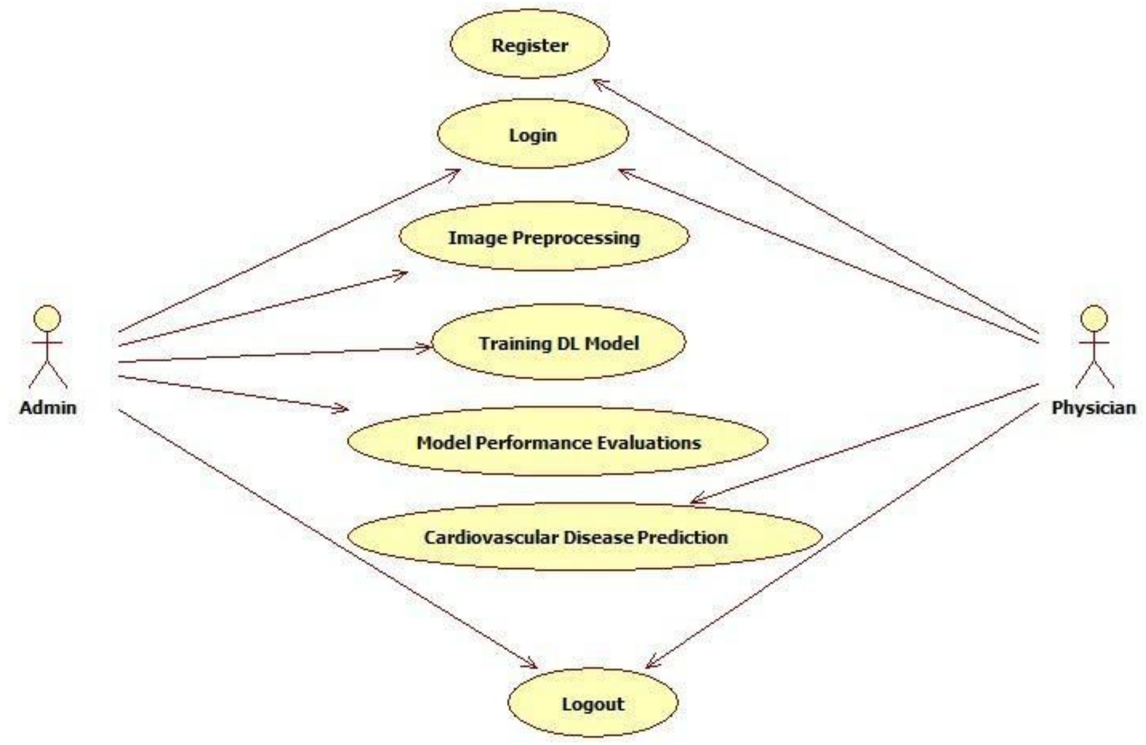


Fig :5.2.1 Usecase Diagram

5.2.2 Class Diagram:

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

There are 4 approaches for identifying classes:

a. Noun phrase approach.
b. Common class pattern approach.
c. Use case Driven Sequence or Collaboration approach.
d. Classes , Responsibilities and collaborators Approach.

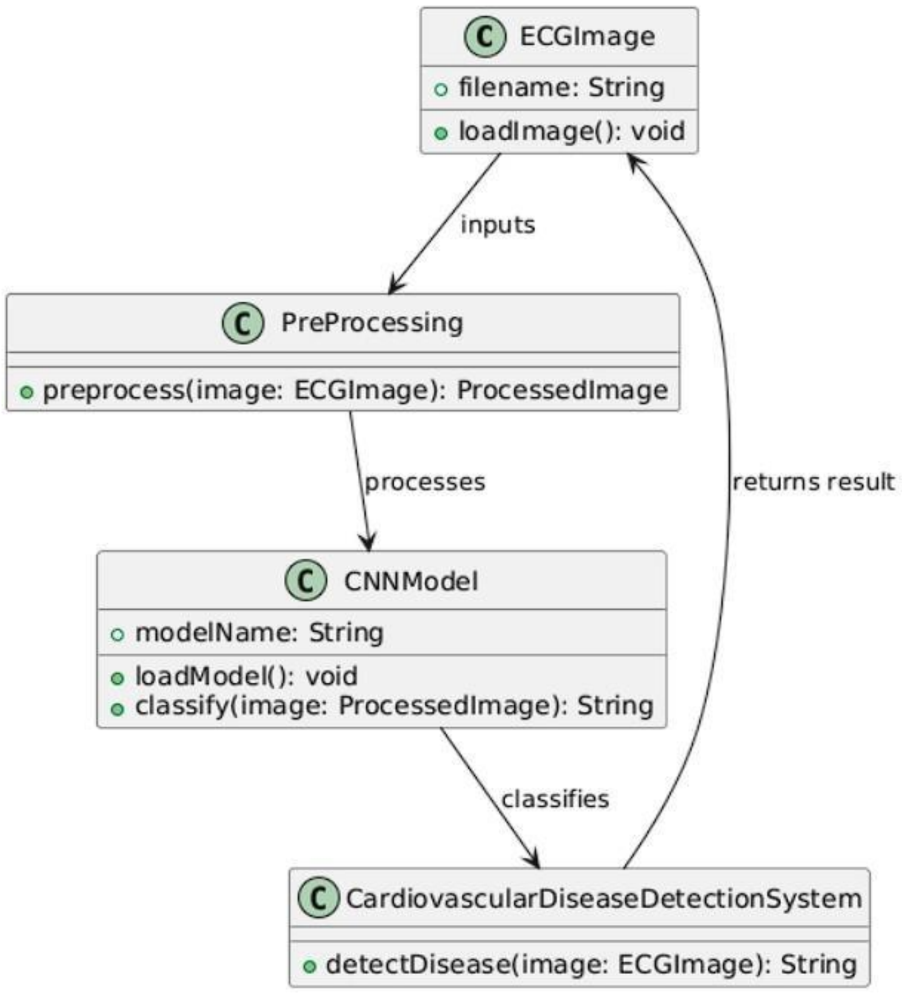

ECGImage
filename: String
loadImage(): void
inputs
PreProcessing
preprocess(image: ECGImage): ProcessedImage
processes
returns result
CNNModel
modelName: String
loadModel(): void
classify(image: ProcessedImage): String
classifies
CardiovascularDiseaseDetectionSystem
detectDisease(image: ECGImage): String

Fig :5.2.2 Class Diagram

5.2.3 Sequence Diagram:

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

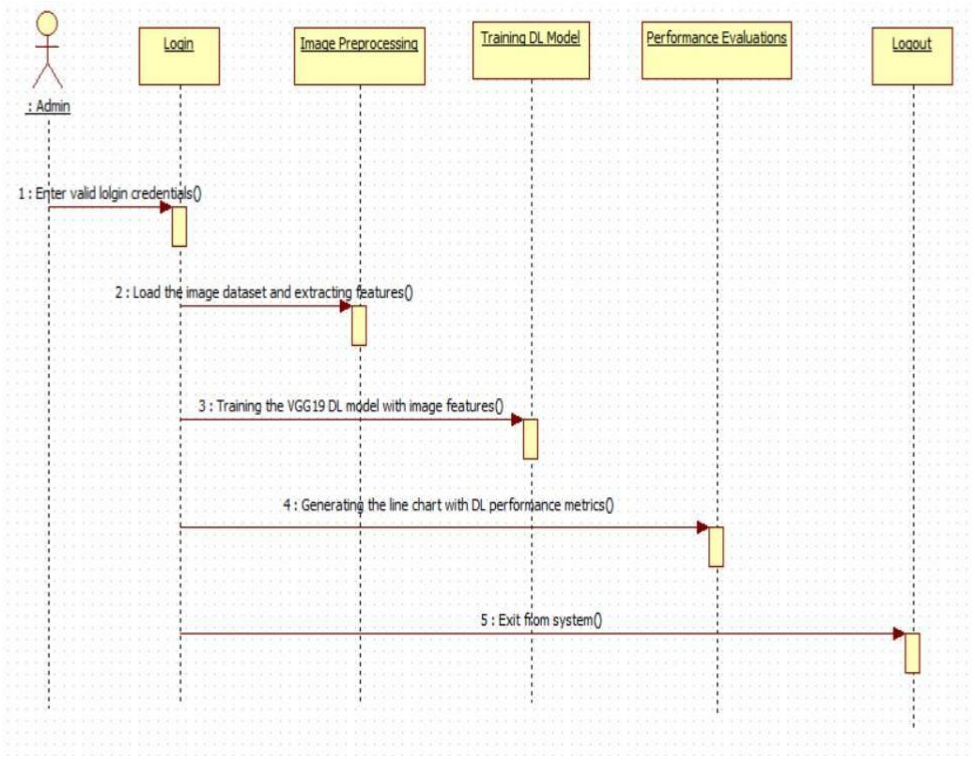

: Admin
Login
Image Preprocessing
Training DL Model
Performance Evaluations
Logout
1 : Enter valid lolgin credentials()
2 : Load the image dataset and extracting features()
3 : Training the VGG19 DL model with image features()
4 : Generating the line chart with DL performance metrics()
5 : Exit from system()

Fig :5.2.3 Sequence Diagram

This sequence diagram demonstrates the linear flow of actions taken by the physician, from registration through using the system for prediction, and finally logging out.

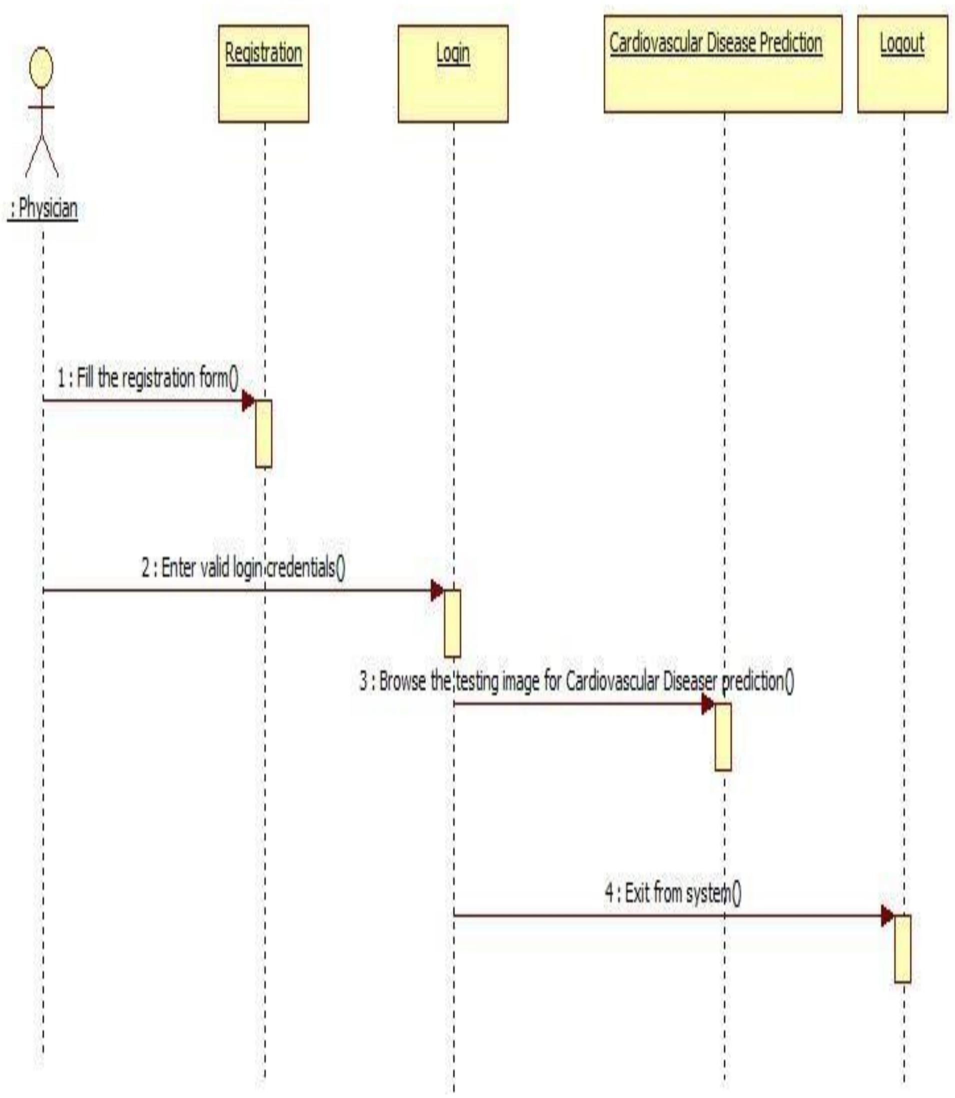

: Physician
Registration
Login
Cardiovascular Disease Prediction
Logout
1 : Fill the registration form()
2 : Enter valid login credentials()
3 : Browse the testing image for Cardiovascular Diseaser prediction()
4 : Exit from system()

Fig :5.2.3 Sequence Diagram

5.2.4 Activity Diagram:

This activity diagram demonstrates the main operations of the system and how users interact with it based on their roles.

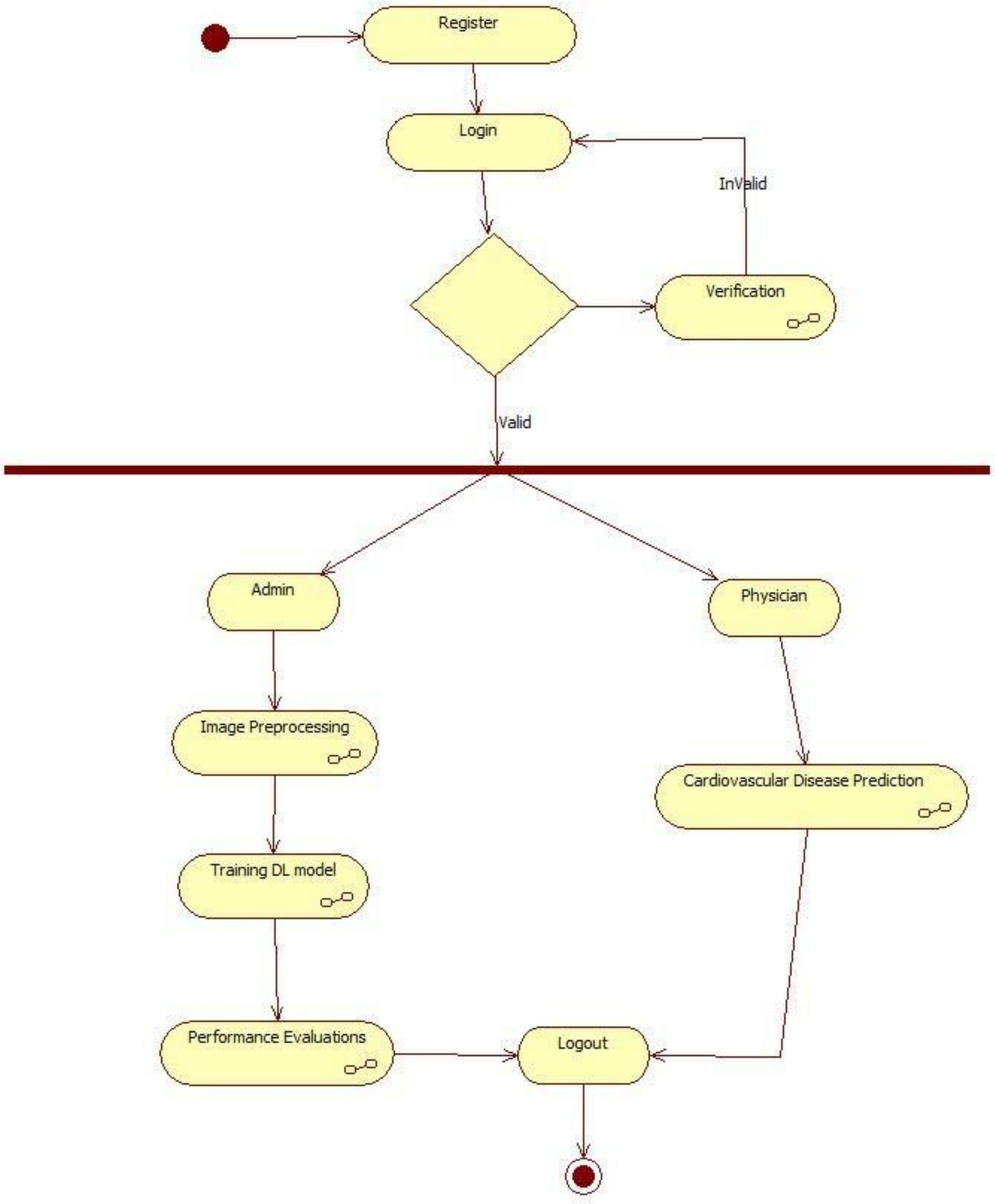


Fig :5.2.4 Activity Diagram

5.3 Modules in the proposed system:

- **Dataset Collection:**

  In this proposed system, we are acquiring the ECG Images dataset of Cardiac Patients from the internet resources. This dataset contains 928 ECG images with 4 classes *Abnormal Heartbeat*, *History of Myocardial Infarction*, *Myocardial Infarction,* and *Normal*. Each class *Abnormal Heartbeat* has 233 images, *History of Myocardial Infarction* has 172 images, *Myocardial Infarction* has 239 images and *Normal* has 284 images.

- **Image Pre-processing:**

  The dataset contains folder structures with images, so it needs to read the images from that directory with Python libraries *OS* and *path*. The machines cannot understand images directly; therefore, it is mandatory to convert the images into pixel format with the Python library NUMPY to get features from images. Later the image dataset will be separated into independent and dependent features. Here independent features are like image pixels which are stored in a list and disease names or classed or target values are treated as dependent values which they can also store in a separate list.

- **Split Dataset:**

  Based on values of independent features and target features, the dataset will be split with 70 to 30 ratios. Here 70 percent indicates the training set and 30 percent indicates the testing set.

- **Training the Model:**

  Here the CNN architecture will be created based on the pre-trained model and image features will be extracted by using this pre-trained model. Thereafter, the ML model will be trained with a training set and generated the training model which will be used for the detection of cardiovascular diseases further.

- **Performance Evaluations:**

  The performance evaluations will be generated with the help of testing the image dataset. Here the trained ML model will used to calculate performance metrics such as accuracy, precision, recall, and f1_score by taking the input of the testing dataset.

- **Cardiovascular Diseases Detection:**

  In this stage, it has to browse the ECG image as an input image to detect cardiovascular diseases. Here from this testing input image, this system extracts the features of the image by using the DL pre-trained model and feeds these features to the machine learning model then this predictor model can classify the cardiovascular diseases.

➢ **Future scope:**

In future work, optimization techniques can be used to obtain optimized values for the hyperparameters of the proposed CNN model. The proposed model can also be used for predicting other types of problems. Since, the proposed model belongs to the family of low-scale deep learning methods in
terms of the number of layers, parameters, and depth. Therefore, a study on using the proposed model in the Industrial Internet of Things domain for classification purposes can be explored.

➢ **Feasibility study**

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions.

➢ **Technical Feasibility**

Technical resources need for project Development

- Windows family Operating System
- Python 3.6 Technology
- Vs Code
- Mysql
- Sqlyog

➢ **Economic Fesibility**

Cost/ benefits analysis of the project as over project is academic project we will not have only basic cost for learning of the technologies

➢ **Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project.

CHAPTER-6 IMPLEMENTATION AND RESULTS

6.1 METHOD OF IMPLEMENTATION

➢ **Architecture of the proposed system:**

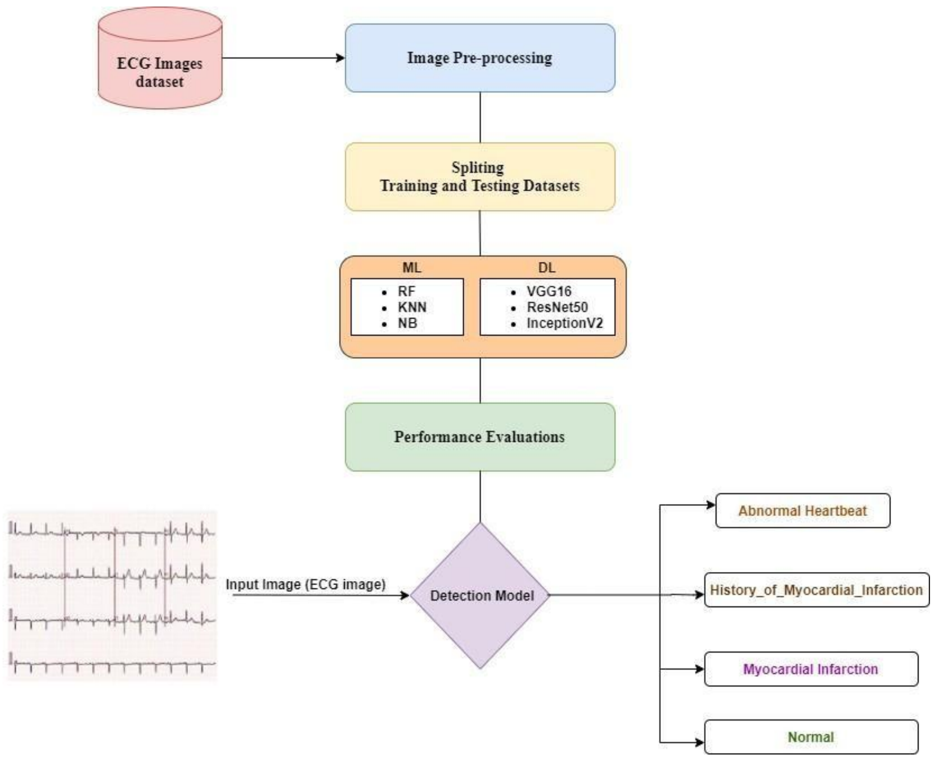


Figure.6.1.1 depicts the cardiovascular disease detection model and explains briefly in the following:

6.2 SOFTWARE OVER VIEW:

➢ **History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

➢ Input as CSV File

Reading data from CSV(comma separated values) is a fundamental necessity in Data Science. Often, we get data from various sources which can get exported to CSV format so that they can be used by other systems. The Panadas library provides features using which we can read the CSV file in full as well as in parts for only a selected group of columns and rows.

The CSV file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named input.csv.You can create this file using windows notepad by copying and pasting this data. Save the file as input.csv using the save As All files(*.*) option in notepad.

import pandas as pd data=

pd.read_csv('path/input.csv') print(data)

➢ Operations using NumPy

NumPy is a Python package which stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations -related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

➢ Key Features of Pandas:

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.

- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

➢ **Python Flask Tutorial**

Flask Tutorial provides the basic and advanced concepts of the Python Flask framework.

Our Flask tutorial is designed for beginners and professionals.

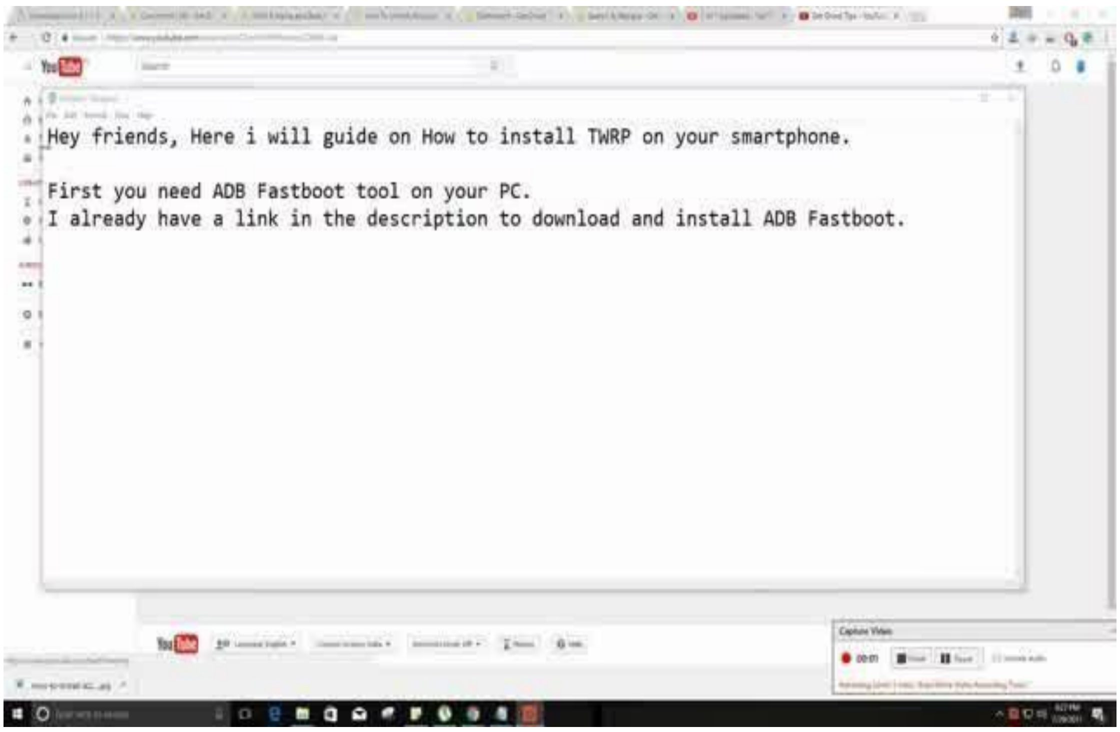
Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by **Armin Ronacher** who leads an international group of python enthusiasts (POCCO).

What is Flask?

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by **Armin Ronacher** who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

What is WSGI?

It is an acronym for web server gateway interface which is a standard for python web application development. It is considered as the specification for the universal interface between the web server and web application.



```
Hey friends, Here i will guide on How to install TWRP on your smartphone.

First you need ADB Fastboot tool on your PC.
I already have a link in the description to download and install ADB Fastboot.
```

What is Jinja2?

Jinja2 is a web template engine which combines a template with a certain data source to render the dynamic web pages.

Flask Environment Setup

To install flask on the system, we need to have python 2.7 or higher installed on our system. However, we suggest using python 3 for the development in the flask. Install virtual environment (virtualenv) virtualenv is considered as the virtual python environment builder which is used to create the multiple python virtual environment side by side. It can be installed by using the following command.

1. $ pip install virtualenv

Once it is installed, we can create the new virtual environment into a folder as given below.

1. $ mkdir new
2. $ cd new
3. $ virtualenv venv

To activate the corresponding environment, use the following command on the Linux operating system.

1. $ venv/bin/activate

On windows, use the following command.

1. $ venv\scripts\activate

We can now install the flask by using the following command.

1. $ pip install flask

However, we cannstall the flask using the above command without creating the virtual environment.

➢ Pycharm Tutorial

PyCharm is the most popular IDE for Python, and includes great features such as excellent code completion and inspection with advanced debugger and support for web programming and various frameworks. PyCharm is created by Czech company, Jet brains which focusses on creating integrated development environment for various web development languages like JavaScript and PHP.

➢ Audience

This tutorial has been prepared for Python developers who focus on using IDE with complete package of running, debugging and creating projects in various python frameworks. Also, interested learners with a basic knowledge of any IDE can take up this tutorial.

➢ Prerequisites

Before proceeding with this tutorial, you need a basic knowledge of any integrated development environment of Python like Sublime Text or most popular IDE like NetBeans. If you are a beginner, we suggest you to go through tutorials related to these topics first before proceeding further on this tutorial.

PyCharm is the most popular IDE used for Python scripting language. This chapter will give you an introduction to PyCharm and explains its features. PyCharm offers some of the best features to its users and developers in the following aspects −

- Code completion and inspection
- Advanced debugging
- Support for web programming and frameworks such as Django and Flask Features of PyCharm

Besides, a developer will find PyCharm comfortable to work with because of the features mentioned below − Code Completion

PyCharm enables smoother code completion whether it is for built in or for an external package.

SQLAlchemy as Debugger

You can set a breakpoint, pause in the debugger and can see the SQL representation of the user expression for SQL Language code.

Git Visualization in Editor

When coding in Python, queries are normal for a developer. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one.

Code Coverage in Editor

You can run **.py** files outside PyCharm Editor as well marking it as code coverage details elsewhere in the project tree, in the summary section etc.

Package Management

All the installed packages are displayed with proper visual representation. This includes list of installed packages and the ability to search and add new packages.

Local History

Local History is always keeping track of the changes in a way that complements like Git.

Local history in PyCharm gives complete details of what is needed to rollback and what is to be added.

Refactoring

Refactoring is the process of renaming one or more files at a time and PyCharm includes various shortcuts for a smooth refactoring process.

User Interface of PyCharm Editor

The user interface of PyCharm editor is shown in the screenshot given below. Observe that the editor includes various features to create a new project or import from an existing project.

You can download the PyCharm Editor and read its official documentation at this link − https://www.jetbrains.com/pycharm/

DATABASE

➢ About MySQL:

MySQL is a relational database management system (RDBMS)[1] that runs as a server providing multi-user access to a number of databases.

The SQL phrase stands for Structured Query Language. Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large- scale World Wide Web products, including Wikipedia, Google , Facebook, and Twitter.

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout it's history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com.

The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius, who had worked together since the 1980's. More historical information on MySQL is

➢ Data Base Tables

```
/*
SQLyog Community Edition- MySQL GUI v6.07
Host - 5.5.30 : Database - cardio_disease
*****************************************************************
*
**
Server version : 5.5.30
*/
/*!40101 SET NAMES utf8 */;
/*!40101 SET SQL_MODE=''*/;
create database if not exists `cardio_disease`;
USE `cardio_disease`;
/*!40014 SET
@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*Table structure for table `dl_ml_evaluations` */
DROP TABLE IF EXISTS `dl_ml_evaluations`;
CREATE TABLE `dl_ml_evaluations` (
`dl_algm` varchar(50) DEFAULT NULL,
`ml_algm` varchar(50) DEFAULT NULL,
`accuracy` varchar(50) DEFAULT NULL,
`precision` varchar(50) DEFAULT NULL,
`recall` varchar(50) DEFAULT NULL,
`fscore` varchar(50) DEFAULT NULL
)
```

```
ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*Table structure for table `physician` */
DROP TABLE IF EXISTS `physician`;
CREATE TABLE `physician`
(
`name` varchar(100) DEFAULT NULL,
`username` varchar(100) DEFAULT NULL,
`passwrd` varchar(100) DEFAULT NULL,
`email` varchar(100) DEFAULT NULL,
`mno` varchar(100) DEFAULT NULL
)
ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
```

6.3 EXPLANATION OF KEY FUNCTION :

In particular, these functions extend the evaluation of deep learning and machine learning algorithms by fetching data from the database, processing it, and visualizing the results. This is an extension of the key functionality of evaluation within the application. Machine learning models are evaluated with helper functions like dl_RF_evaluations(), dl_KNN_evaluations(), and dl_NB_evaluations().

6.3.1 CODE :

```
def dl_RF_evaluations():
database = DBConnection.getConnection()
cursor = database.cursor()
cursor.execute("SELECT * FROM dl_ml_evaluations where
ml_algm='RF'")
rows = cursor.fetchall()
   # Visualize the result using barchart
   barchart(rows, "VGG_RF", "ResNet_RF", "Inception_RF", "RF")
   print("rows=", rows)
   def dl_KNN_evaluations():
   database = DBConnection.getConnection()
   cursor = database.cursor()
   cursor.execute("SELECT * FROM dl_ml_evaluations where
ml_algm='KNN'")
   rows = cursor.fetchall()
   # Visualize the result using barchart
   barchart(rows, "VGG_KNN", "ResNet_KNN", "Inception_KNN",
"KNN")
   print("rows=", rows)
    def dl_NB_evaluations():
   database = DBConnection.getConnection()
   cursor = database.cursor()
   cursor.execute("SELECT * FROM dl_ml_evaluations where
ml_algm='NB'")
   rows = cursor.fetchall()
   # Visualize the result using barchart

   barchart(rows, "VGG_NB", "ResNet_NB", "Inception_NB",
"NB")

    print("rows=", rows)
```

6.4 Source Code :

```
from flask import Flask, render_template, request, flash, session

from DBConfig import DBConnection

from werkzeug.utils import secure_filename

from Graphs import barchart

from CD_Prediction import prediction_image

import os

import warnings

import sys



warnings.filterwarnings('ignore')
```

```
app = Flask(_name_)

app.secret_key = "1234"



# Home

@app.route('/')

def index():

    return render_template('index.html')



# Admin routes

@app.route("/admin")

def admin():

    return render_template("admin_login.html")



@app.route("/adminlogin_check", methods=["POST"])

def adminlogin():

    uid = request.form.get("unm")
    pwd = request.form.get("pwd")
    if uid == "admin" and pwd == "admin":
        return render_template("admin_home.html")
    else:
        return render_template("admin_login.html", msg="Invalid
Credentials")


@app.route('/admin_home')
def admin_home():
    return render_template('admin_home.html')



# User routes
@app.route("/user_home")
def user_home():
    return render_template("user_home.html")



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

```
def user_login():

    return render_template("user_login.html")



@app.route("/newuser")

def newuser():

    return render_template("user_register.html")



@app.route("/user_register", methods=["POST"])

def user_register():

    try:

        name = request.form.get('name')

        uid = request.form.get('unm')

        pwd = request.form.get('pwd')

        mno = request.form.get('mno')

        email = request.form.get('email')
        database = DBConnection.getConnection()

        cursor = database.cursor()



        # Check if username exists

        sql = "SELECT COUNT(*) FROM physician WHERE
username = %s"

        cursor.execute(sql, (uid,))

        res = cursor.fetchone()[0]



        if res > 0:

            return render_template("user_register.html",
msg="Username already exists..!")



        # Insert user

        sql = "INSERT INTO physician (name, username,
password, email, mno) VALUES (%s, %s, %s, %s, %s)"

        values = (name, uid, pwd, email, mno)
        cursor.execute(sql, values)

        database.commit()



        return                    render_template("user_login.html",
msg="Registered Successfully..! Login Here.")



    except Exception as e:

        print("Registration Error:", e)

        return  render_template("user_register.html",  msg="Error
occurred. Try again.")



@app.route("/userlogin_check", methods=["POST"])

def userlogin_check():

    uid = request.form.get("unm")

    pwd = request.form.get("pwd")



    try:
        database = DBConnection.getConnection()

        cursor = database.cursor()

        sql = "SELECT COUNT(*) FROM physician WHERE
username = %s AND password = %s"

        cursor.execute(sql, (uid, pwd))

        res = cursor.fetchone()[0]



        if res > 0:

            session['uid'] = uid

            return render_template("user_home.html")

        else:

            return                     render_template("user_login.html",
msg2="Invalid Credentials")



    except Exception as e:

        print("Login Error:", e)
        return render_template("user_login.html", msg2="Login
error occurred.")



# Detection

@app.route("/detection")

def detection():

    return render_template("detection.html")



@app.route("/prediction", methods=["POST"])

def prediction():

    try:

        image = request.files['file']



        # Save to absolute path

        folder_path =
"D:/CD_Detection/UpdateCode/CD_Detection/test_image"

        os.makedirs(folder_path, exist_ok=True)
        image_path = os.path.join(folder_path, "test_img.jpg")

        image.save(image_path)



        result = prediction_image(image_path)

        return                  render_template("detection_result.html",
result=result)



    except Exception as e:

        print("Prediction Error:", e)

        return                  render_template("detection_result.html",
result="Prediction failed due to an error.")



# Evaluations

@app.route("/dl_evaluations")

def dl_evaluations():

    try:

        database = DBConnection.getConnection()
        cursor = database.cursor()

        cursor.execute("SELECT * FROM dl_ml_evaluations")

        rows = cursor.fetchall()



        dl_RF_evaluations()

        dl_KNN_evaluations()

        dl_NB_evaluations()



        return          render_template("models_evaluations.html",
rawdata=rows)



    except Exception as e:

        print("Error:", e)

        tb = sys.exc_info()[2]

        print("Line:", tb.tb_lineno)

        return          render_template("models_evaluations.html",
rawdata=[])
```

```
def dl_RF_evaluations():

    database = DBConnection.getConnection()

    cursor = database.cursor()

    cursor.execute("SELECT    *    FROM    dl_ml_evaluations
WHERE ml_algm = 'RF'")

    rows = cursor.fetchall()

    barchart(rows,  "VGG_RF",  "ResNet_RF",  "Inception_RF",
"RF")



def dl_KNN_evaluations():

    database = DBConnection.getConnection()

    cursor = database.cursor()

    cursor.execute("SELECT    *    FROM    dl_ml_evaluations
WHERE ml_algm = 'KNN'")

    rows = cursor.fetchall()

    barchart(rows,           "VGG_KNN",           "ResNet_KNN",
"Inception_KNN", "KNN")
```

```
def dl_NB_evaluations():

    database = DBConnection.getConnection()

    cursor = database.cursor()

    cursor.execute("SELECT    *    FROM    dl_ml_evaluations
WHERE ml_algm = 'NB'")

    rows = cursor.fetchall()

    barchart(rows, "VGG_NB", "ResNet_NB", "Inception_NB",
"NB")



# Start Flask app

if _name_ == '_main_':

    app.run(host="localhost", port=1234, debug=True)
```

6.5 Training &Evaluation :

6.5.1 Training :

1. **Data Preparation**:

- The code loads and preprocesses images for both training and testing using cv2.imread() and resizes them. Labels are encoded using LabelEncoder.
- Data is normalized and split into training and testing sets.

2. **Model Training**:

- **Random Forest (RF)**: The features are extracted from the InceptionV3 model, and the **Random Forest Classifier** is trained using RF_model.fit(features, y_train).
- **K-Nearest Neighbors (KNN)**: Similarly, the **KNN model** is trained using KNN_model.fit(features, y_train).
- **Naive Bayes (NB)**: The **Naive Bayes model** is trained using NB_model.fit(features, y_train).

CODE:

```
RF_model.fit(features, y_train) # Training Random Forest

KNN_model.fit(features,        y_train)        #        Training        KNN

NB_model.fit(features, y_train) # Training Naive Bayes
```

6.5.2 Evaluation :

1. **Feature Extraction**:

- The test data is passed through the **InceptionV3** model to extract features using inceptn_model.predict(x_test).

2. **Model Evaluation**:

- After predicting with the trained models (RF, KNN, NB), the code evaluates the accuracy, precision, recall, and F1-score for each model using metrics from sklearn.
- The evaluation results are then inserted into the database (dl_ml_evaluations) using SQL queries.

6.6 OUTPUT SCREENS :

6.6.1 Home Page :

Welcome to our platform dedicated to the intelligent detection of cardiovascular diseases through advanced ECG analysis. Leveraging the power of machine learning and deep learning techniques, our system aims to enhance early diagnosis and improve patient outcomes. Cardiovascular diseases remain a leading cause of mortality worldwide, and timely detection is critical. By analyzing ECG signals with sophisticated algorithms, our solution can identify patterns and anomalies often missed by traditional methods. The integration of AI allows for accurate, fast, and scalable assessment of heart conditions, supporting healthcare professionals in clinical decision-making. This platform showcases our research, methodology, datasets, results, and potential applications in real-world scenarios. Whether you're a medical professional, researcher, or enthusiast, explore how cutting-edge technology is reshaping cardiac care. Join us in our mission to make cardiac diagnostics more efficient, accessible, and precise through innovation in machine learning and deep learning.

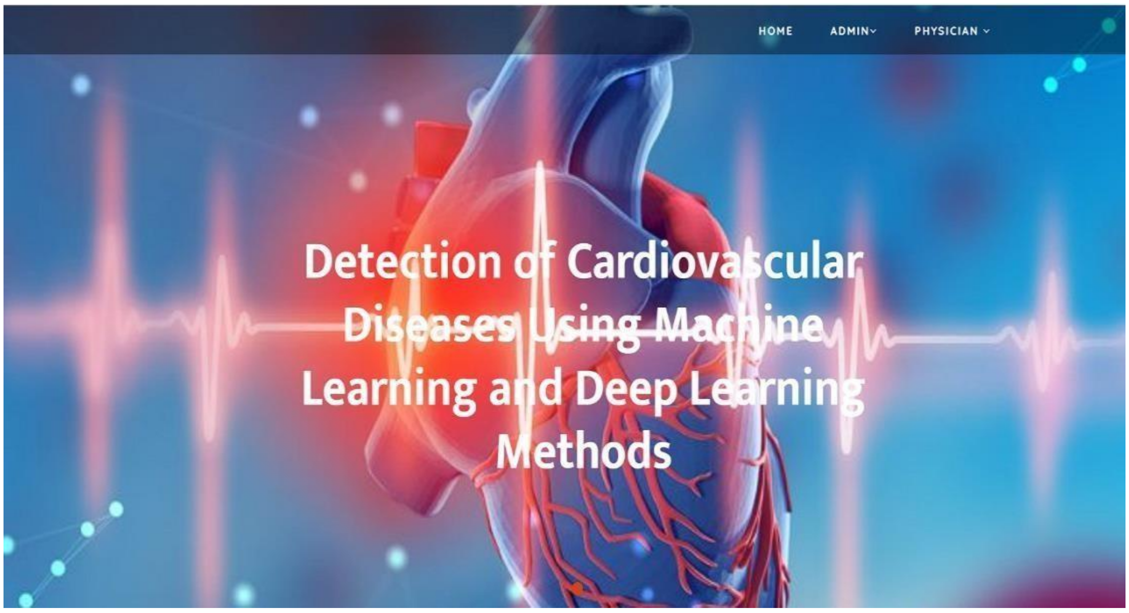
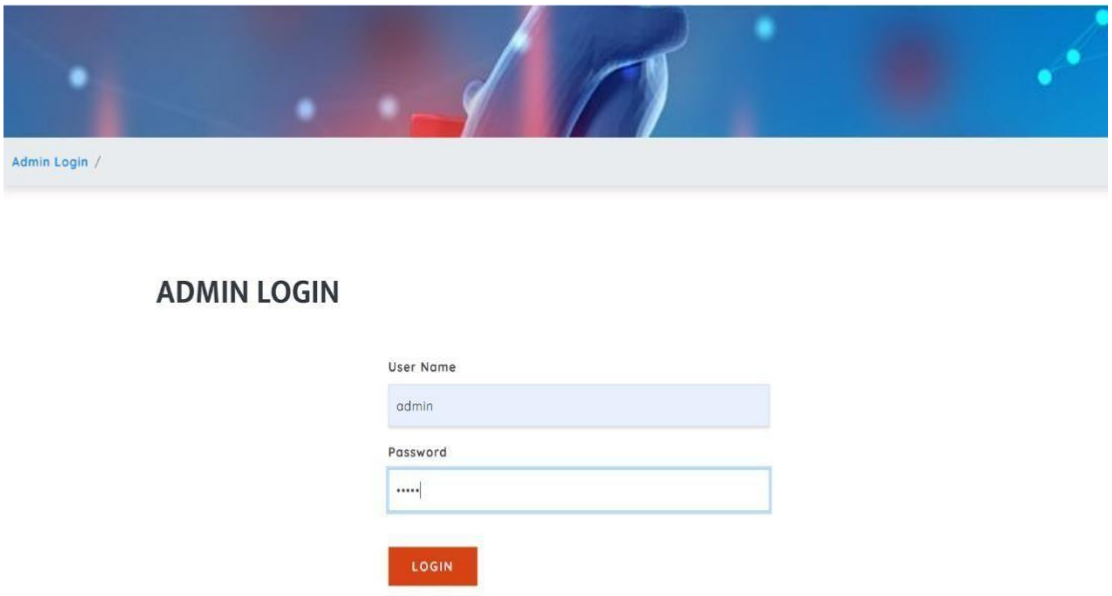

HOME
ADMIN
PHYSICIAN
Detection of Cardiovascular Diseases Using Machine Learning and Deep Learning Methods

Fig:6.6.1 Home Page

6.6.2 Admin Login :

The admin login for the "Detection of Cardiovascular Diseases in ECG using Machine Learning and Deep Learning Methods" system provides secure access to manage patient data, monitor model performance, and oversee system operations. It ensures that only authorized personnel can view sensitive ECG data and control prediction modules. Implemented with authentication features, the login interface typically includes a username and password field. Once logged in, administrators can upload ECG datasets, initiate training or testing of ML/DL models, and review diagnostic outputs. This login system is vital for maintaining data privacy, ensuring system integrity, and complying with medical data security standards.

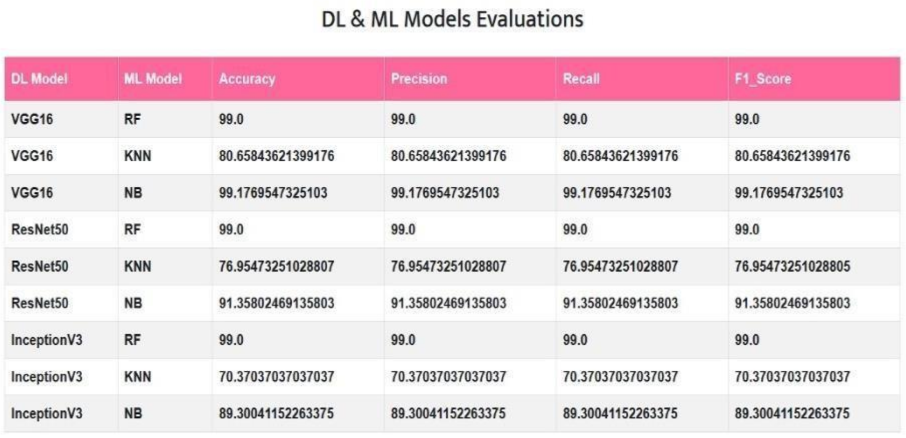


User Name
admin
Password
•••••

Fig:6.6.2 Admin Login

6.6.3 DL & ML Models Evaluations:

The performance of ML and DL models for ECG-based cardiovascular disease detection is evaluated using metrics such as **accuracy, precision, recall, F1-score**, and **AUC-ROC**. Machine Learning models like SVM, Random Forest, and KNN are typically assessed using cross-validation and confusion matrices. Deep Learning models, especially CNNs, LSTMs, or hybrid architectures, are evaluated on large ECG datasets like MIT-BIH using training-validation-test splits. DL models often outperform ML models in capturing complex ECG patterns due to automatic feature extraction. Robust evaluation ensures that the models are generalizable, clinically relevant, and safe for real-world medical applications.

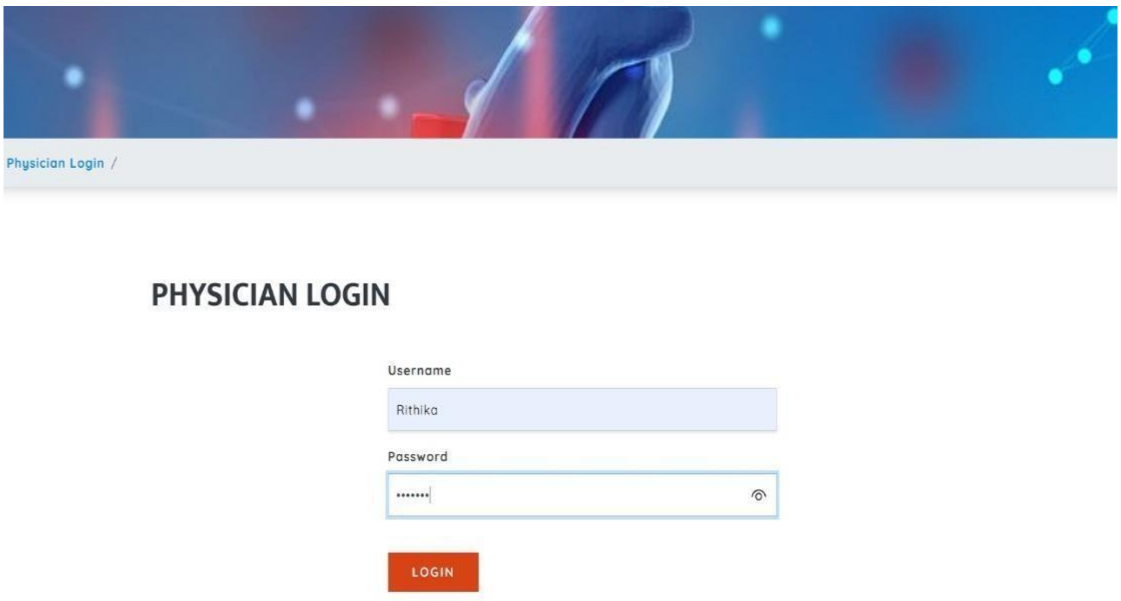

DL & ML Models Evaluations

DL Model	ML Model	Accuracy	Precision	Recall	F1_Score
VGG16	RF	99.0	99.0	99.0	99.0
VGG16	KNN	80.65843621399176	80.65843621399176	80.65843621399176	80.65843621399176
VGG16	NB	99.1769547325103	99.1769547325103	99.1769547325103	99.1769547325103
ResNet50	RF	99.0	99.0	99.0	99.0
ResNet50	KNN	76.95473251028807	76.95473251028807	76.95473251028807	76.95473251028805
ResNet50	NB	91.35802469135803	91.35802469135803	91.35802469135803	91.35802469135803
InceptionV3	RF	99.0	99.0	99.0	99.0
InceptionV3	KNN	70.37037037037037	70.37037037037037	70.37037037037037	70.37037037037037
InceptionV3	NB	89.30041152263375	89.30041152263375	89.30041152263375	89.30041152263375

Fig:6.6.3 DL & ML Models Evaluations

6.6.4 Physician Login :

The physician login provides secure access for doctors to use the ECG-based cardiovascular disease detection system powered by machine learning and deep learning. After logging in with authorized credentials, physicians can upload patient ECG data, view diagnostic predictions, and interpret results supported by AI. The interface offers visualizations of ECG signals, probability scores, and disease classifications to assist in clinical decisions. Role-based access ensures patient data confidentiality and compliance with healthcare regulations. This login system enables efficient interaction between medical professionals and AI tools, improving diagnostic accuracy and facilitating timely intervention in cardiovascular disease management.


Physician Login /

PHYSICIAN LOGIN

Username

Rithika

Password

•••••••

Fig:6.6.4 Physician Login

6.6.5 Prediction :

In the ECG-based cardiovascular disease detection system, the prediction phase involves analyzing patient ECG signals using trained machine learning or deep learning models. Preprocessed ECG data is input into the model, which identifies abnormal patterns linked to conditions such as arrhythmia, myocardial infarction, or ischemia. Machine learning models use extracted features like heart rate variability, while deep learning models (e.g., CNNs, LSTMs) analyze raw ECG signals directly. The system outputs a predicted disease class along with confidence scores. These predictions assist physicians in early diagnosis and decision-making, improving patient outcomes through faster and more accurate detection of cardiovascular conditions.

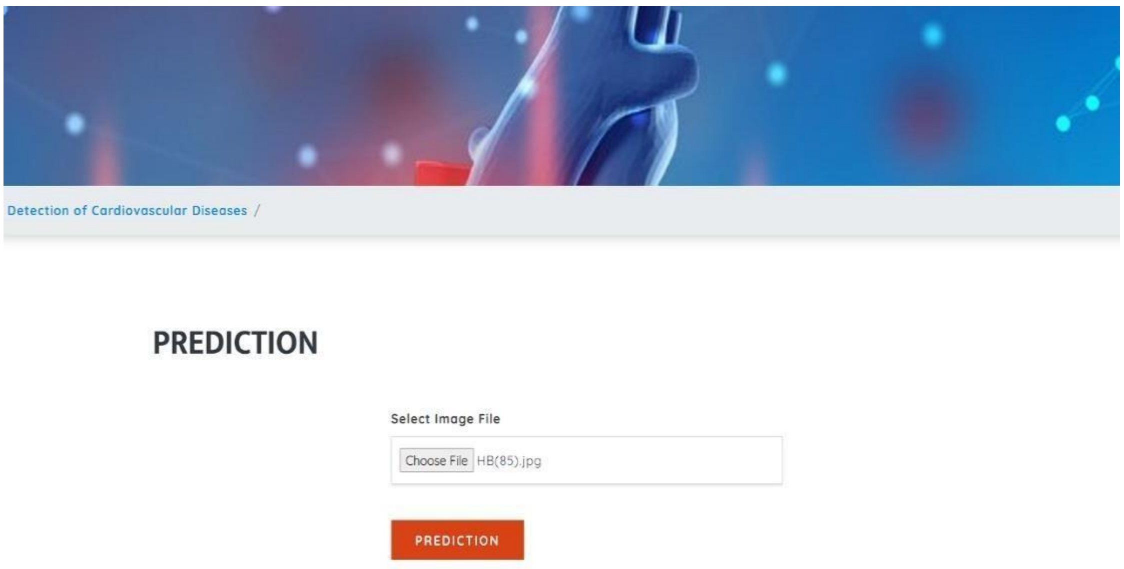
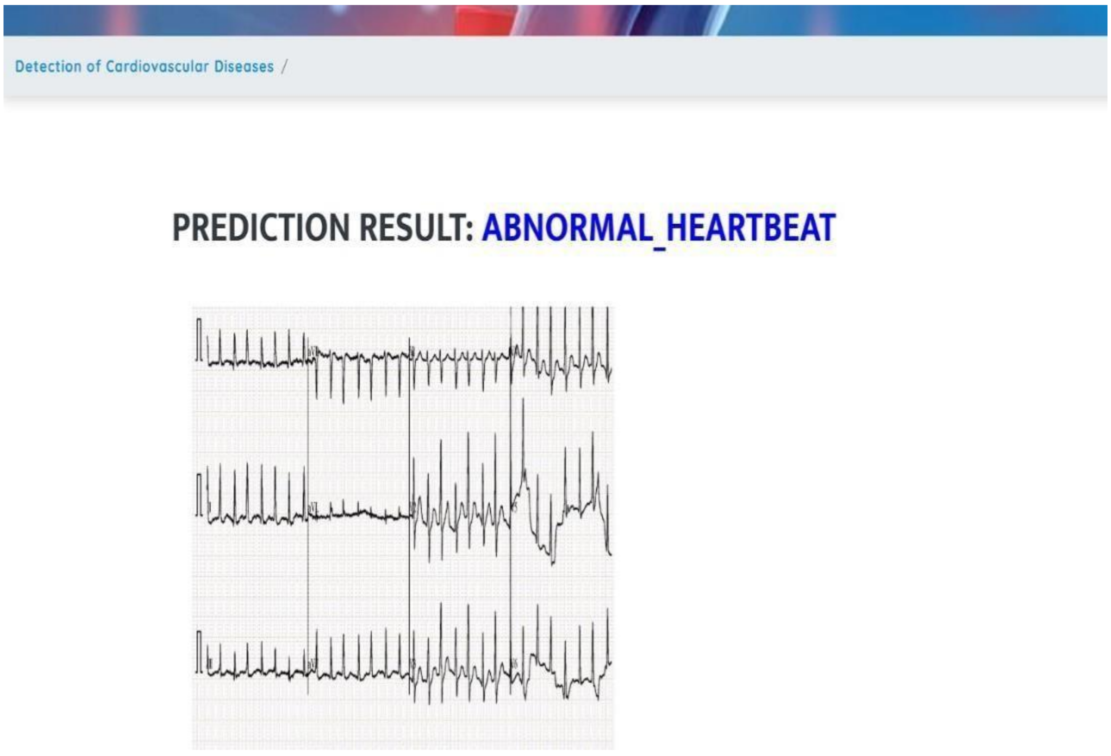


Fig:6.6.5 Prediction

6.6.6 Prediction Result :



Detection of Cardiovascular Diseases /

PREDICTION RESULT: ABNORMAL_HEARTBEAT

Fig:6.6.6 Prediction Result

CHAPTER-7 SYSTEM TESTING

7.1 INTRODUCTION FOR TESTING :

Software testing is one of the main stages of project development life cycle to provide our cessation utilizer with information about the quality of the application and ours, in our Project we have under gone some stages of testing like unit testing where it’s done in development stage of the project when we are in implementation of the application after the Project is yare we have done manual testing with different Case of all the different modules in the application we have even done browser compatibility testing in different web browsers in market, even we have done Client side validation testing on our application.

➢ Unit testing

The unit testing is done in the stage of implementation of the project only the error are solved in development stage some of the error we come across in development are given below.

➢ TESTING

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

➢ Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.A successful test is one that uncovers an as yet undiscovered error. A good test case is one that has probability of finding an error.

- **Levels of Testing:**In order to uncover present in different phases we have the concept of levels of testing.
- **The basic levels of Testing:**

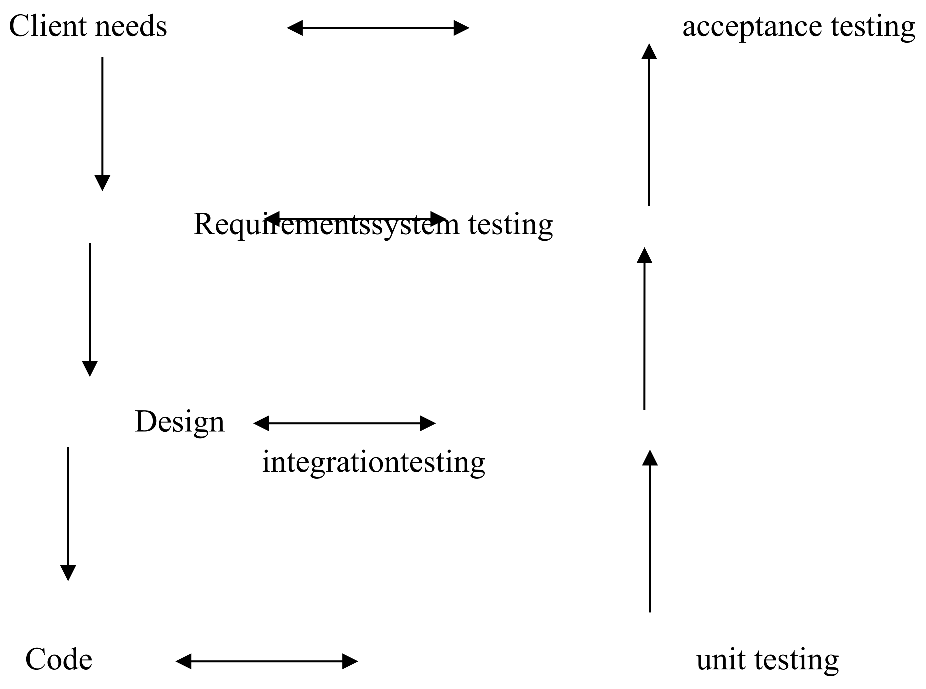


Fig7.1.1: Levels of Testing

- **Code testing:**

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

- **Specification Testing:**

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

- **Unit testing:**

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

➢ **What is Black Box Testing?**

Black box testing is a software testing techniques in which **functionality of the software under test (SUT) is tested without looking at the internal code structure**, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Input
Black Box
Output

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing – Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them
- correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the
- SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

➢ Types of Black Box Testing

What do you verify in White Box Testing ?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development.

One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique.

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application.

This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

➢ **System testing:**

Once the individual module testing is completed, modules are assembled and integrated to perform as a system. The top down testing, which began from upper level to lower level module, was carried out to check whether the entire system is performing satisfactorily.

There are three main kinds of System testing:

i. Alpha Testing
ii. Beta Testing iii. Acceptance Testing

➢ **Alpha Testing:**

This refers to the system testing that is carried out by the test team with the Organization. ➢ **Beta Testing**:

This refers to the system testing that is performed by a selected group of friendly customers

➢ **Acceptance Testing:**

This refers to the system testing that is performed by the customer to determine whether or not to accept the delivery of the system.

➢ **Integration Testing:**

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

➢ **Output testing:**

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system.

7.2 VARIOUS TESTCASE SCENARIOS

TEST CASES

Test Case ID	#1	**Test Case Description** - Validations in Registration Form		
S#	**Prerequisites**	**S#**	**Test Data Requirement**	
1	User should be Registered	1	Data should be valid	
Test Condition				
Entering data in registration form				
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass/Fail/Not Executed/Suspended**
1	User gives First and Last Name	Pop showing email verification message	Enter valid email/password	Fail
2	Submitting the form without entering any details	Pop showing email verification message	Enter email /password	Fail
3	User enters invalid format of email id	Pop showing email verification message	Enter valid email id	Fail
4	User enters a phone number with < 10 digits	Pop showing email verification message	Enter valid phone number	Fail

5	Entering valid username and password	Pop showing email verification message	Pop showing email verification message	Pass

Table 1 Registration test case

Test Case ID #2		**Test Case Description -** Validations in Login Form		
S#	**Prerequisites**	**S#**	**Test Data Requirement**	
1	User should have an email id	1	Data should be valid	
Test Condition				
Entering data in login form				
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass/Fail/Not Executed/Suspended**
1	User gives aemail or password of <6 characters	User logged in	Enter valid email/password	Fail
2	Submitting the form without entering any details	User logged in	Enter email /password	Fail
3	User enters wrong Email and (or) password	User logged in	Enter correct email /password	Fail

Table 7.2.1: Various test case scenarios

CHAPTER-8 CONCLUSION

8.1 PROJECT CONCLUSION :

In this article, we propose a lightweight CNN-based model to classify the four major cardiac abnormalities, i.e., AH, MI, H. MI, and NP classes, using public ECG images dataset of cardiac patients. According to the results of the experiments, the proposed CNN model achieves remarkable results in cardiovascular disease classification and can also be used as a feature extraction tool for the traditional machine learning classifiers. Thus, the proposed CNN model can be used as an assistance tool for clinicians in the medical field to detect cardiac diseases from ECG images and bypass the manual process that leads to inaccurate and time-consuming results. In the future work, optimization techniques can be used to obtain optimized values for the hyperparameters of the proposed CNNmodel. The proposed model can also be used for predicting other types of problems. Since, the proposed model belongs to the family of low-scale deep learning methods in terms of the number of layers, parameters, and depth. Therefore, a study on using the proposed model in the Industrial Internet of Things domain for classification purposes can be explored.

8.2 FUTURE ENHANCEMENT :

Here are some potential future enhancements for your project on detecting cardiovascular diseases in ECG images using machine learning and deep learning:

1. **Real-Time Analysis and Monitoring**: Implement a real-time system to continuously monitor ECG signals and predict cardiovascular issues as they occur. This can be done through wearable devices integrated with cloud-based machine learning models.
2. **Integration of Explainable AI (XAI)**: Enhance the system by integrating explainable AI techniques to provide transparency into how the deep learning model makes predictions, making the system more trustworthy for medical professionals.
3. **Transfer Learning and Pre-trained Models**: Use transfer learning with pre-trained models like ResNet or EfficientNet to improve model accuracy, especially when limited ECG data is available for training.

4. **Multi-Modal Data Fusion**: Combine ECG images with other data types like medical history, blood pressure, or genetic data to enhance prediction accuracy by using hybrid models that process both image and numerical data.
5. **Federated Learning**: Implement federated learning to train models on decentralized datasets from different hospitals or clinics without compromising patient privacy, allowing for broader and more generalized model training.
6. **Advanced Signal Processing Techniques**: Integrate advanced signal processing methods, such as wavelet transforms or Fourier analysis, to preprocess ECG signals before feeding them into the model for improved feature extraction.
7. **Personalized Prediction Models**: Create personalized cardiovascular disease prediction models based on an individual's medical history, lifestyle factors, and ECG data, tailoring predictions to specific patient profiles.
8. **Cloud-Based Deployment with API Access**: Deploy the model on cloud platforms and provide API access for healthcare applications to integrate the model for broader use in hospitals or mobile health apps.
9. **Enhanced Anomaly Detection**: Implement unsupervised learning techniques to detect rare or unseen cardiovascular anomalies in ECG data, improving the system's robustness in handling edge cases.
10. **3D ECG Signal Analysis**: Develop methods to convert 2D ECG signals into 3D representations, offering a different perspective for the detection of complex cardiovascular conditions.

CHAPTER-9 REFERENCES

9.1 PAPER REFERENCES :

[1] U. R. Acharya, H. Fujita, S. L. Oh, Y. Hagiwara, J. H. Tan, and M. Adam, "Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals," Inf. Sci., vol. 415 416, pp. 190–198,2017.

[2] R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande, and P. Singh, "Prediction of heart disease using a combination of machine learning and deep learning," Comput. Intell. Neurosci., vol. 2021, 2021, Art. no. 8387680.

[3] M. Zubair, J. Kim, and C. Yoon, "An automated ECG beat classification system using convolutional neural networks," in Proc. 6th Int. Conf. IT Convergence Secur., 2016, pp. 15.

[4] A.H. Khan,M.Hussain, and M.K. Malik,"Cardiac disorder classification by electrocardiogram sensing using deep neural network," Complexity, vol. 2021, 2021, Art. no. 5512243.

[5] A. Pal, R. Srivastva, and Y. N. Singh, "CardioNet: An efficient ECG arrhythmia classification system using transfer learning," BigDataRes., vol. 26, 2021, Art. no. 100271.

[6] World Health Organization (WHO), "Cardiovascular diseases," Jun. 11, 2021. Accessed: Dec. 27, 2021. [Online]. Available: https://www.who. int/healthtopics/cardiovasculardiseases.

[7] Government of Westren Australia, Department of Health, "Com mon medical tests to diagnose heart conditions," Accessed: Dec. 29, 2021. [Online]. Available: https://www.healthywa.wa.gov.au/Articles/A_ E/Common-medical-tests-to-diagnose-
heartconditions.

[8] M. Swathy and K. Saruladha, "A comparative study of classification and prediction of cardio-vascular diseases (CVD) using machine learning and deep learning techniques," ICT

Exp., to be published, 2021. [Online]. Available: https://doi.org/10.1016/j.icte.2021.08.021. [9] R. R. Lopes et al., "Improving electrocardiogram-based detection of rare genetic heart disease using transfer learning: An application to phos pholamban p.Arg14del mutation carriers," Comput. Biol. Med., vol. 131, 2021, Art. no. 104262. [Online].

Available: https://doi.org/10.1016/j. compbiomed.2021.104262.

[10] R. J. Martis, U. R. Acharya, and H. Adeli, "Current methods in electro cardiogram characterization," Comput. Biol. Med., vol. 48, pp. 133–149, 2014. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2014. 02.012.

9.2 WEBSITES :

[1] Available: https://www.who.int/health-topics/cardiovascular-diseases.

[2] Available: https://www.healthywa.wa.gov.au/Articles/A_E/Common-medical-teststodiagnose-heart-conditions.

[3] Available: https://doi.org/10.1016/j.icte.2021.08.021.

[4] Available: https://doi.org/10.1016/j.compbiomed.2021.104262.

[5] Available: https://doi.org/10.1109/ACCESS.2017.2707460.

[6] Available: https://doi.org/10.1109/ACCESS.2017.2707460.

[7] Available: https://doi.org/10.3390/sym8060047.

[8] Available: https://doi.org/10.1145/3342999.3343015.

9.3 TEXT BOOKS:

Here are some textbooks that could be valuable for detecting cardiovascular diseases in ECG images using machine learning and deep learning:

1. **"Deep Learning for Biomedical Applications"** by Mehmet Kaya. This book explores deep learning methods applied to biomedical imaging, including ECG analysis.
2. **"Machine Learning in Medicine"** by Ton J. Cleophas and Aeilko H. Zwinderman. It discusses various machine learning models and their applications in medical diagnostics.
3. **"Biomedical Signal Processing and Artificial Intelligence in Healthcare"** by Walid A. Zgallai. It covers signal processing techniques and AI applications in ECG and healthcare.
4. **"Artificial Intelligence in Healthcare"** by Parashar Shah and Anil Kumar. This book provides insights into AI methodologies, including machine learning and deep learning techniques, specifically in the context of healthcare applications like ECG analysis.