EDITH: ECG BIOMETRICS AIDED BY DEEP LEARNING FOR RELIABLE INDIVIDUAL AUTHENTICATION

A PROJECT REPORT

Submitted by

ABOORVA S	**422419205002**
PRIYADHARSHINI M	**422419205027**
SAKTHI P	**422419205032**
SHANMATHI U	**422419205035**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"EDITH:ECG BIOMETRICS AIDED BY DEEP LEARNING FOR RELIABLE INDIVIDUAL AUTHENTICATION"** is the bonafide work of **ABOORVA S (422419205002), PRIYADHARSHINI M (422419205027)**, **SAKTHI P (422419205032)** and **SHANMATHI U (422419205035)** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MILTON GANESH,M.E,Ph.D.,
HEAD OF THE DEPARTMENT
Department of Information Technology,
University college of Engineering Melpakkam.
Tindivanam-604 307.

SIGNATURE

Dr.R.SHYAMALA,M.E,Ph.D.,
SUPERVISOR
Department of Information Technology,
University college of Engineering Melpakkam.
Tindivanam-604 307.

Submitted for the University Examination held on _____.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to thank our dean, **Dr.P.Tamizhazhagan,M.E.,Ph.D.,** for providing infrastructure facilities and wholehearted encouragement for completing our project successfully.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr.S.Milton Ganesh,M.E,Ph.D.,** Assistant professor and Head, Department of Information Technology, University College Of Engineering Tindivanam, for extending the facilities of the department towards our project and for his unstinting support.

We express our sincere thanks to our guide, panel member, our class advisor **Dr.R.Shyamala,M.E,Ph.D.,** Assistant professor, former Head of the department, Department of Information Technology, University College Of Engineering Tindivanam, for guiding us for every phase of the project. We appreciate her thoroughness, tolerance and ability to share a knowledge with us. We thank her for being easily approachable and quite thoughtful. We owe her harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could not have it to this extent.

We thank all our teaching and supporting staff members and our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project.

Last but not least,we extend our thanks to our family members,who have been a great source of inspiration and strength to us during the course of this project work.We sincerely thank all of them.

ABSTRACT

EDITH, a novel approach that leverages deep learning techniques to enhance the reliability of individual authentication using ECG (electrocardiogram) biometrics. Traditional authentication methods often rely on easily forgeable credentials or biometric features, which can be compromised or replicated. In contrast, the ECG signal represents a unique physiological pattern that is difficult to spoof or imitate, making it a promising modality for secure authentication. EDITH employs a deep learning framework to extract discriminative features from ECG signals and trains a classification model to accurately distinguish individuals. Experimental results demonstrate that EDITH achieves high authentication accuracy and outperforms existing methods in terms of robustness against spoofing attacks and variability in ECG recordings. The proposed approach holds significant potential for reliable individual authentication in various applications, such as healthcare, finance, and cybersecurity.

Index Terms- Authentication, ECG biometric, Deep Learning, CNN (Convolutional Neural Network)

ABOORVA S

PRIYADHARSHINI M

SAKTHI P

SHANMATHI U

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

ABBREVIATION	FULL FORM
ECG	Electrocardiogram
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
FAR	False Acceptance Rate
FRR	False Rejection Rate
EER	Equal Error Rate
ROC	Receiver Operating Characteristic

CHAPTER 1

INTRODUCTION

In an increasingly digital world where personal information and security are paramount concerns, the need for robust and reliable individual authentication methods has become more critical than ever. Traditional methods such as passwords, PINs, and fingerprint scans have their limitations, leading researchers and engineers to explore innovative approaches. One such promising solution is the utilization of electrocardiogram (ECG) biometrics aided by deep learning techniques, which offers a new frontier in accurate and secure individual authentication. ECG, a non-invasive and readily available signal, records the electrical activity of the heart and exhibits unique characteristics specific to each individual. By leveraging deep learning algorithms, which excel in extracting patterns from complex data, ECG biometric authentication systems can provide a highly accurate and reliable method of verifying an individual's identity.

The acronym EDITH (ECG Deep learning for Individual authentication with High accuracy) encapsulates the focus of this study: developing an advanced ECG biometric system empowered by deep learning models to achieve reliable individual authentication. The integration of deep learning allows for more sophisticated feature extraction, pattern recognition, and classification, enabling the system to distinguish genuine ECG signals from impostors or anomalies with heightened precision.

One of the main advantages of ECG biometrics is its resistance to forgery and spoofing. Unlike physical biometrics such as fingerprints, which can be replicated or manipulated, the ECG signal is an internal physiological characteristic that is nearly impossible to replicate. Furthermore, the ECG signal remains relatively stable over time, providing a persistent and consistent feature for long-term authentication.

The proposed authentication system aims to address several key challenges inherent in ECG-based biometrics. These challenges include dealing with noisy ECG signals caused by factors such as motion artifacts, variable heart rates, and different recording conditions. Deep learning models can effectively handle these challenges by learning complex representations and adapting to varying conditions, ultimately enhancing the system's accuracy and reliability.

This study also explores the potential applications of EDITH beyond traditional authentication scenarios. ECG biometric authentication can be seamlessly integrated into various domains, including healthcare, financial services, smart devices, and access control systems. Its non-invasive nature and compatibility with existing ECG monitoring technologies make it an appealing choice for industries seeking convenient and secure authentication methods.

In summary, this research aims to introduce EDITH, a cutting-edge ECG biometric authentication system that leverages deep learning techniques to achieve reliable individual identification. By harnessing the unique features of the ECG signal and the power of deep learning algorithms, EDITH promises to enhance security, convenience, and accuracy in authentication processes across diverse industries.

1.1 DEEP LEARNING

Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data .These neural networks are inspired by the structure and function of the human brain’s biological neurons,

and they are designed to learn from large amounts of data. It is automating and improving the learning process of computer without being actual programmed i.e., without human assistance .This process starts with good quality of dataset and then training by building deep neural network modules using data and different algorithm .The choose of algorithm depend on what type of data do we have and what kind of task we are typing to automate.

1.1.1 Supervised Learning

Supervised Learning in which the neural network learns to make predictions or classify data based on the labelled datasets. Labelled data is data that has a known output or target value associated with each input. In the training phase, the model is trained using a set of labelled data. Training data is to minimize the difference between the predicted output and the actual target value. They are trained by providing it with input and matching output patterns the neural network learns to make predictions based on the cost or error that comes from the difference between the predicted and the actual target, this process is known as back propagation. The testing phase involves evaluating the performance of the model on a set of new data that it has not seen before. This is done to check if the model can generalize well to new data. Deep learning algorithms like Convolutional neural networks, Recurrent neural networks are used for many supervised tasks like image classifications and recognition, sentiment analysis, language translations, etc.

1.1.1.1 Classification

In classification, the output variable is a categorical value, such as a binary classification problem like whether a plant is healthy or not, or a multiclass classification problem like identifying different species of flowers. The goal of classification is to learn a function that can correctly classify the input data into the appropriate category. Some of the popular algorithms used

for classification are logistic regression, decision trees, support vector machines, and neural networks.

1.1.1.2 Regression

In regression, the output variable is a continuous value. The goal of regression is to learn a function that can predict the output value given the input data. Linear regression, polynomial regression, and decision tree regression are some of the common algorithms used for regression.

1.1.2 Unsupervised Learning

The Unsupervised learning technique in which the neural network learns to discover the patterns or to cluster the dataset based on unlabelled datasets. Here there are no target variables. Only original date is required to start the analysis while the machine has to self-determined the hidden patterns or relationships within the datasets. Deep learning algorithms like auto encoders and generative models are used for unsupervised tasks like clustering, dimensionality reduction, and anomaly detection.

1.1.3 Reinforcement Learning

Reinforcement Learning is the technique in which an agent learns to make decisions in an environment to maximize a reward signal. The agent interacts with the environment by taking action and observing the resulting rewards. Deep learning can be used to learn policies, or a set of actions, that maximizes the cumulative reward over time. Deep reinforcement learning algorithms like Deep Q networks and Deep Deterministic Policy Gradient (DDPG) are used to reinforce tasks like robotics and game playing etc.

1.2 PRE-PROCESSING

Pre-processing in deep learning refers to the transformation of raw input data into a format that is suitable for training and inference with a deep

learning model. It involves various data manipulation techniques that are applied to the input data before it is fed into the deep learning model for training or prediction. Pre-processing is an important step in the deep learning pipeline as it can significantly impact the performance and accuracy of the model.

1.3 FEATURE EXTRACTION

Feature extraction involves extracting relevant features from the raw input data that can be used as input to the deep learning model. This may include techniques such as dimensionality reduction, where the number of input features is reduced while retaining important information, or feature selection, where a subset of the most informative features is selected for training the model.

1.4 CLASSIFICATION

Classification in deep learning refers to the process of categorizing input data into predefined classes or categories. It is a fundamental task in machine learning and has numerous applications, including image recognition, sentiment analysis, spam detection, and medical diagnosis.

1.5 FEASIBILITY STUDY

Feasibility studies aim to objectively and rationally uncover the system, weakness of the existing system and strength of proposed system, opportunities and threats as presented by environment, the resources required to Carry and ultimately the proposes for process. The feasibility study is conducted to see whether a project can be further preceded or discontinuing the project. The feasibility analysis is useful to determine final product will benefit to the users and organizations. Three aspects of feasibility study are

- Technical feasibility

- Economic feasibility
- Operational feasibility

1.5.1 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical feasibility requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The technical requirements are then compared to the technical capability of the organization. The project is considered.

1.5.2 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available.

1.5.3 Operational Feasibility

The aspect of study is to check the level of acceptance of system by the user. This includes the process of training the user to use the system efficiently.

CHAPTER 2

LITERATURE SURVEY

A Nabil Ibtehaz , “EDITH: ECG biometrics aided by deep learning for reliable Individual authentication” (2022) has proposed a deep learning-based framework for ECG biometrics authentication system. EDITH comprises a novel convolutional network architecture has been evaluated on four benchmark datasets and has consistently outperformed the priorworks. The parameter is accuracy. The advantage is inherent robustness against forgery. The disadvantage is the ECG signal from same person may deviate.

Muhammad Uzair Zahid, “Robust R-peak detection in low quality Holter ECGs using 1D convolutional neural network” (2022) has proposed a generic and robust system for R-peak detection in Holter ECG signals and algorithms that successfully addressed the problem of ECG R-peak detection. The parameter is positive predictive value. The advantage is reduced false positive and false negative rate. The disadvantage is complexity and less performance.

Sricharan Vijayarangan , “A deep learning approach for robust R-peak detection in noisy ECG” (2020) has proposed a novel application of the Unet combined with Inception and Residual blocks to perform the extraction of R-peaks from an ECG and the problem formulation also robustly deals with issues of variability and sparsity of ECG R-peaks. The parameters are precision, recall and F1 score. The advantage is robustness and versatility. The disadvantage is it is Hard to detect the R-peak accurately for an Untrained human eye.

Anthony Ngozichukwuka Uwaechia, “A Comprehensive survey on ECG Signals as new biometric modality for human authentication”(2020) has proposed the techniques employed for the ECG as biometrics for human authentication and an overview and discussion on ECG signal preprocessing, feature extraction, feature selection, and feature transformation for ECG-based biometric systems. The parameters are ECG recognition accuracy, PTB, MIT-BIH. The advantage is Increased recognition performance, enhanced security and fewer enrolment problems. The disadvantage is, this method is crucial for two factor authentication.

Mohit Ingale, “ECG biometric authentication: A comparative analysis” (2020) has proposed the impact of filtering type, segmentation, feature extraction, and health status on ECG biometric by using the evaluation metrics and the results have shown that the ECG biometric authentication outperforms existing methods lacking the ability to efficiently extract features, filtering, segmentation, and matching. The parameter is Five on-the-person and in-house off- the-person public ECG databases. The advantage is Better performance in identification rate, FAR, FRR, EER. The disadvantage is High chances of Spoofing.

Tomas repcik, “Biometric Authentication using the unique characteristics of the ECG signal” (2020) and tested three different ECG-based authentication methods on data measured by Maxim Integrated wristband and the first method extracted 22 time-domain features - intervals and amplitudes from each heartbeat and Hjorth descriptors of an average heartbeat. The parameter is false acceptance rate and false rejection rate. The advantages are high accuracy, non-intrusive and inconvenient. The disadvantages are hardware dependency and signal variability.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM REQUIREMENTS

The software requirements specification is acquired at the completion of analysis phase. The function and performance allocated to software in system engineering is elaborated by complete information description of software performance evaluation, design constraints and appropriate validation criteria.

3.1.1 Functional Requirements

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process. It lists the requirements of a particular software system including functional performance and security requirements. The requirements also provide usage scenarios from a user an operational and an administrative perspective. The purpose of software requirements specifications is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

3.1.2 Hardware Requirements

- Intel® Core™ i3-5005U CPU @ 2.00GHz or newer
- RAM /Hard Drive 4 GB.5 GB free disk space

3.1.3 Software Requirements

- Technology - Deep learning
- Backend - Python
- Frontend - Tkinter

- Operating System - Windows 10 or 11
- IDE - Google colab, Jupyter notebook

3.2 PYTHON

Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. Python is high level ,interpreted and general proposed dynamic programming language that focus on code readability ,and its syntax allow programmers to express their concept .Python is a programming language that let to work quickly and integrate the system more efficient .Python is interpreted language so it is flexible and easy to use language. It is used in many organizations as it supports multiple programming paradiagram it also supports automatic memory management . Extensive support libraries (OpenCV for computer vision, NumPy for numerical analysis, TensorFlow, eras for deep learning application and Task) Python clear object oriented design provides enhanced process control, image processing and integration capability as well as its own unit testing framework which make it more efficient.

3.3 JUPYTER NOTEBOOK

Jupyter notebooks basically provide an interactive computational environment for developing python-based Data science applications. A Jupyter Notebook app is a server –client application that allows running and editing notebook document via a web browser. Jupyter notebook combines live code, graphics, visualizations, and text in shareable notebooks that run in a web browser, the notebook itself could be hosted on your local machine or on a remote server. It is an open-source ,web-based interactive environment which allows you to create and share documents that contain live code, mathematical equation, graphics, maps, plots, visualizations and narrative text.

3.4 Google colab

It is a free, cloud-based platform provided by Google for running and developing machine learning and deep learning models. It is based on the Jupyter Notebook environment and provides a range of tools and resources for data analysis, visualization, and machine learning. Google Colab provides a free computing environment, including a virtual machine, that allows users to run Python code and execute machine learning models using popular libraries such as TensorFlow, Keras, and PyTorch. It also comes with pre-installed libraries for data analysis and visualization, such as NumPy, Pandas, and Matplotlib.it allows users to run code on powerful GPUs and TPUs (Tensor Processing Units) for free. This makes it particularly useful for running large deep learning models that require significant computational resources. Google Colab also provides collaboration features, allowing users to share notebooks and collaborate in real-time with others. It also allows users to easily import and export data from popular cloud storage services such as Google Drive, GitHub, and Dropbox. It is a popular choice for beginners and experts alike, due to its ease of use, free resources, and flexibility. It is particularly useful for experimenting with different machine learning models and developing prototypes quickly, without the need for expensive hardware or software.

3.5 Tkinter

Tkinter is utilized in the EDITH system to develop a graphical user interface (GUI) that facilitates user interaction. The Tkinter-based GUI enables users to register and authenticate their ECG data, visualizing ECG waveforms in a user-friendly manner. The GUI allows users to register their ECG data, authenticate their identity, and visualize ECG waveforms. It also provides feedback and notifications during the authentication process and allows system administrators to configure various parameters. Tkinter enhances the user

experience and interaction with the EDITH system for reliable individual authentication.

3.6 EXISTING SOLUTION

The existing method for ECG biometric authentication typically involve using deep learning techniques such as convolutional neural networks (CNNs) to extract features from ECG signals and authenticate individuals. The systems achieve high accuracy rates and can operate in real-time. They leverage the power of deep learning algorithms to learn complex patterns in ECG signals and provide reliable individual authentication. These solutions serve as valuable references for the development of the EDITH system for ECG biometric authentication.

CHAPTER 4

SYSTEM DESIGN

4.1 PROPOSED SYSTEM

The proposed solution for EDITH: ECG Biometric Aided by Deep Learning for Reliable Individual Authentication involves collecting ECG data, preprocessing it to remove noise, extracting features using deep learning techniques, training a model to identify individuals based on these features, integrating the system with a user-friendly GUI using Tkinter for easy interaction, and continuously evaluating and optimizing the system for improved accuracy and reliability.

4.2 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behaviour and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A System architecture can comprise system components, their interrelationships, and the principles and guidelines governing their design and evolution over time. It starts with collecting datasets from the Kaggle repository and importing them using the necessary libraries. The collected data is then analyzed to gain a better understanding of its structure and content.

To prepare the data for classification, several pre-processing steps are performed. This includes data normalization to ensure uniform scaling and re-sizing of images to have consistent dimensions. Additionally, data augmentation techniques are applied to increase the diversity and quantity of the data. Feature extraction techniques are then implemented to extract relevant information from the pre-processed data. These features capture the essential characteristics of the plant diseases and serve as inputs for the subsequent classifier system.

The extracted features are passed through a classifier system, which is trained using the labelled data. The classifier learns to recognize and differentiate between different plant diseases based on the provided features. Once the classifier is trained, it can make predictions or classify unseen data.

The performance of the classifier system is evaluated using metrics such as model accuracy. Model accuracy measures the percentage of correctly classified instances. Additionally, a confusion matrix is utilized to visualize the classification performance in more detail. The confusion matrix provides insights into the model's strengths and weaknesses by showing the number of true positives, true negatives, false positives, and false negatives.

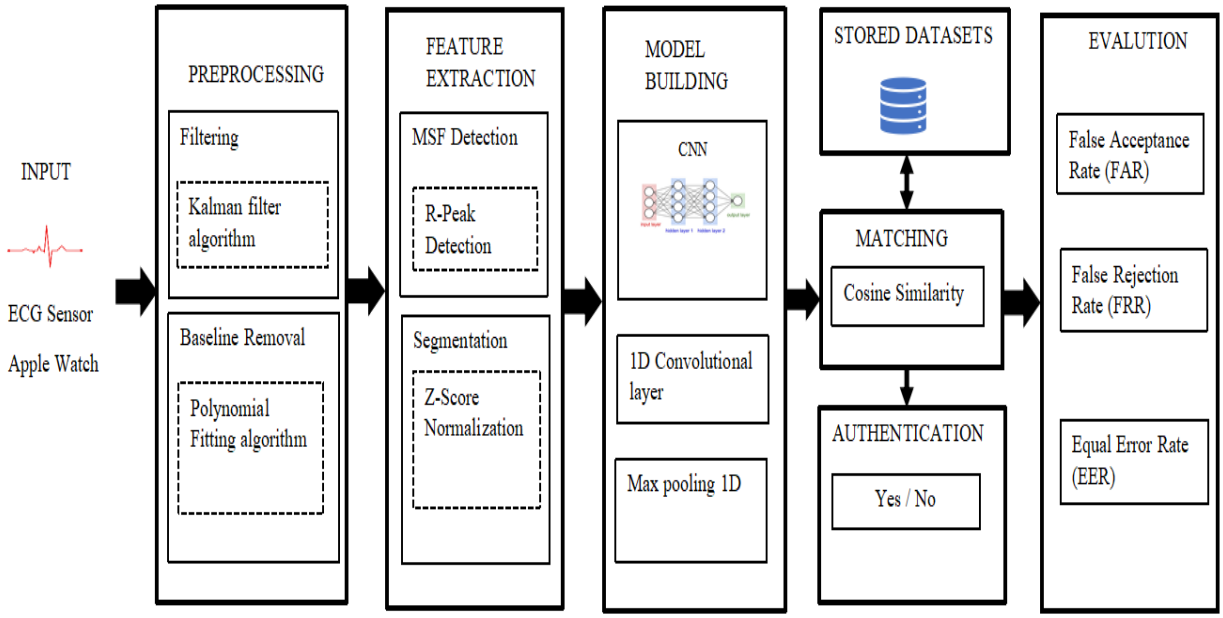


Figure 4.1 System Architecture

4.3 DATA FLOW DIAGRAM

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

4.3.1 DATA FLOW DIAGRAM Level-0

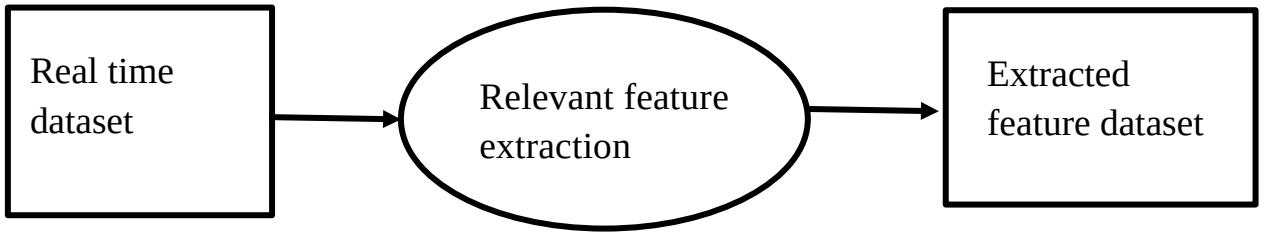


Figure 4.2 data flow diagram level-0

In the above Figure 4.2 the real time datasets are taken from the ECG Signals. The relevant features are extracted from the real time ECG dataset.

4.3.2 DATA FLOW DIAGRAM Level-1

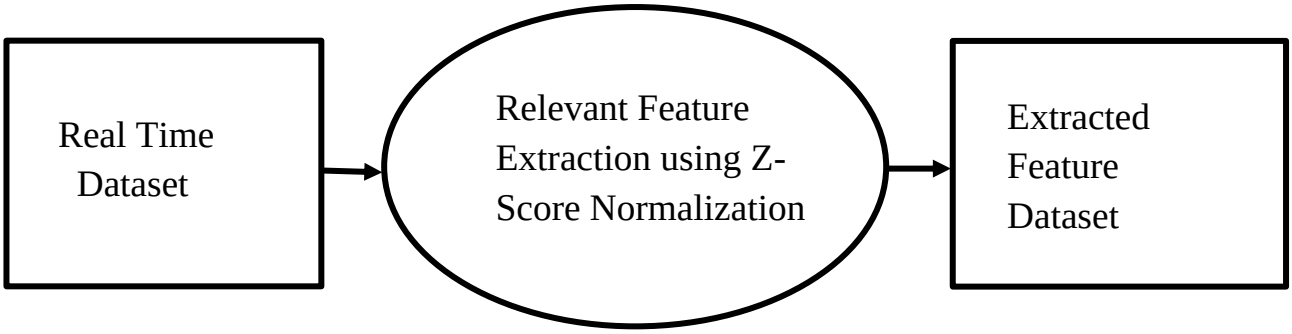


Figure 4.3 Data Flow Diagram level-1

The above figure 4.3 shows the first level of data flow diagram. Here the feature selection method is used for extracting the features by using Z- Score normalization.

4.3.3 DATA FLOW DIAGRAM Level-2

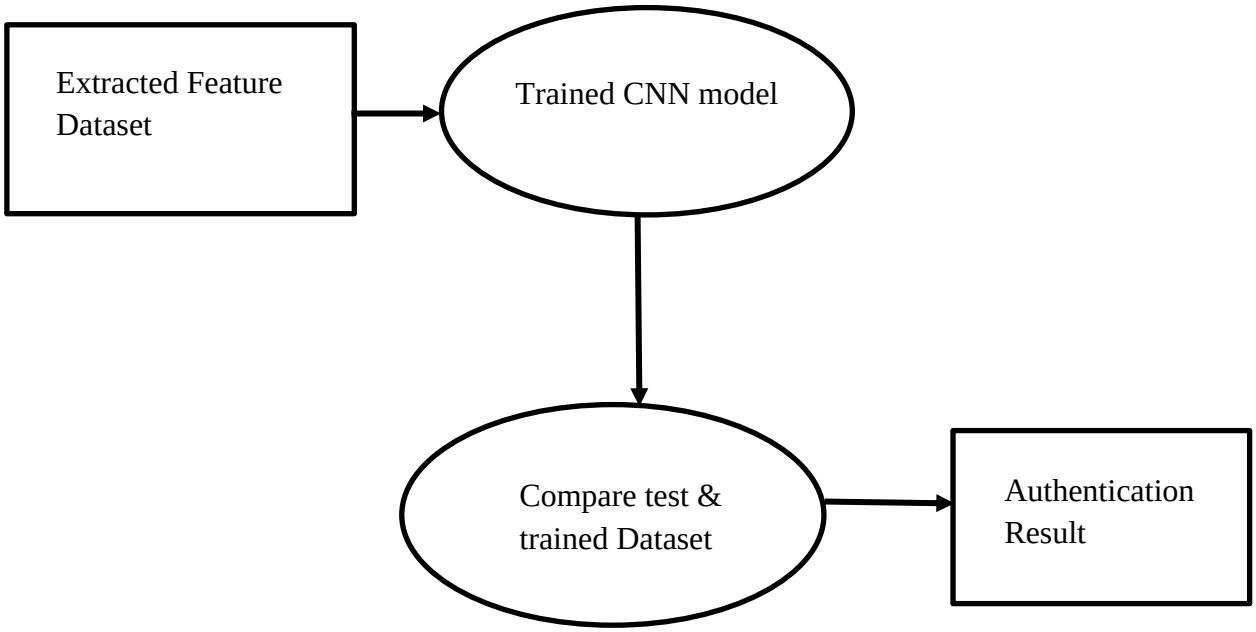


Figure 4.4 Data Flow Diagram level-2

The above Figure 4.4 shows the Second Level of Data Flow Diagram. The extracted feature dataset is trained using CNN model and then compare the test and trained dataset for the authentication result.

4.4 UML DIAGRAM

4.4.1 Use Case Diagram

A use case is a methodology used in system analysis to identify, clarify and organize the system requirements. The use case is made up of a set of possible sequences of interactions between system and users in a particular

environment and related to a particular goal. It is represented using ellipse. Actor is any external entity that makes use of the system being modelled.

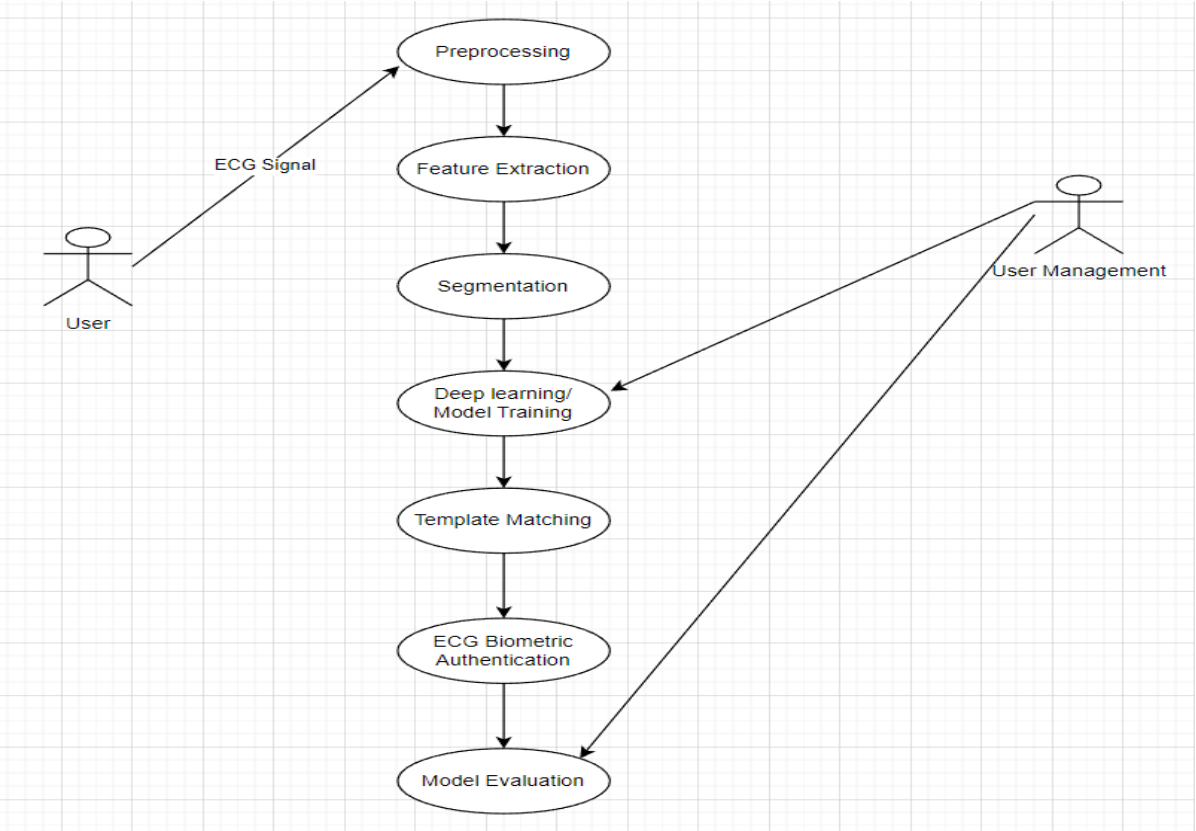


Figure 4.5 Use case Diagram

4.4.2 Class Diagram

The classes are arranged in groups that share common characteristics. A class diagram resembles a flow chart in which classes are potrayed as boxes, each box having three rectangles inside. The top, middle and lowest rectangle contains the name, attribute and methods of the class. The lines define the relationships, also called association between the classes.

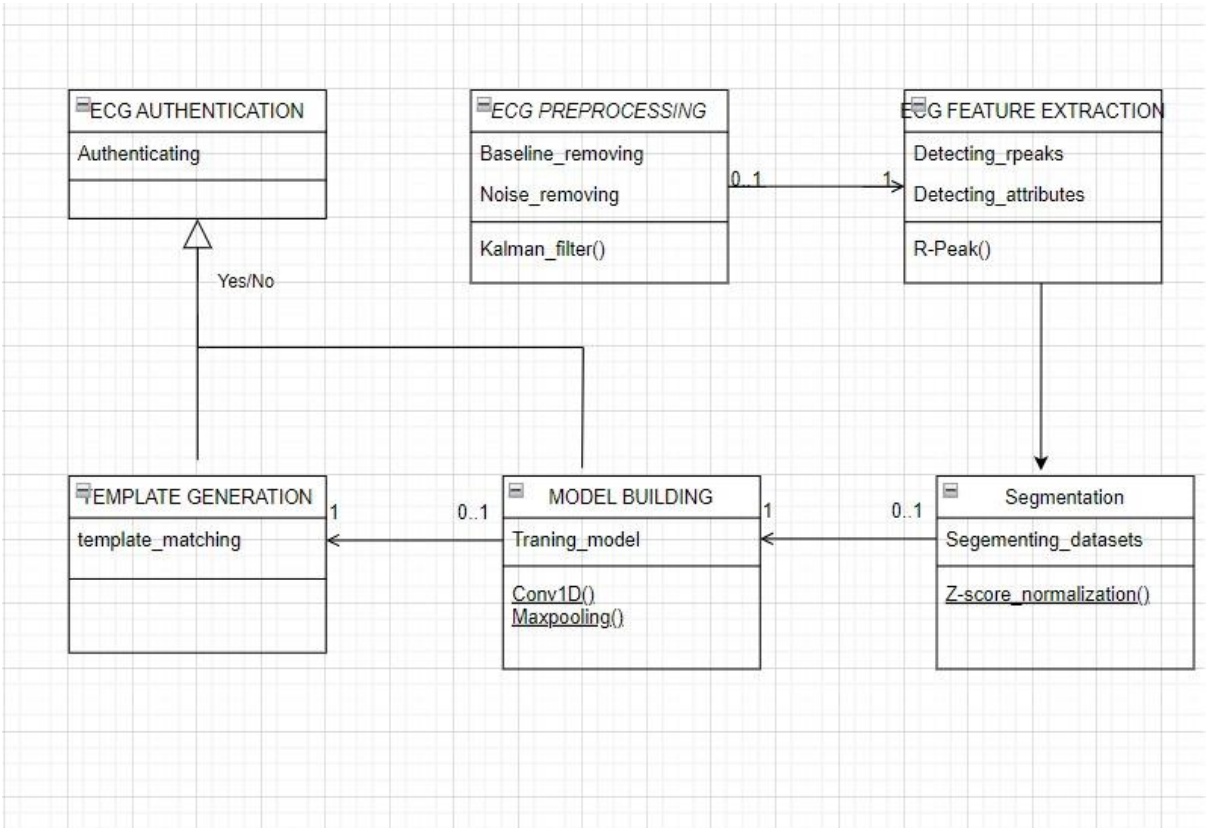


Figure 4.6 Class Diagram

4.4.3 Sequence diagram

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what it is a construct of a message sequence chart. The are two dimensions,

Vertical dimensions – represent time.

Horizontal dimension – represent different objects.

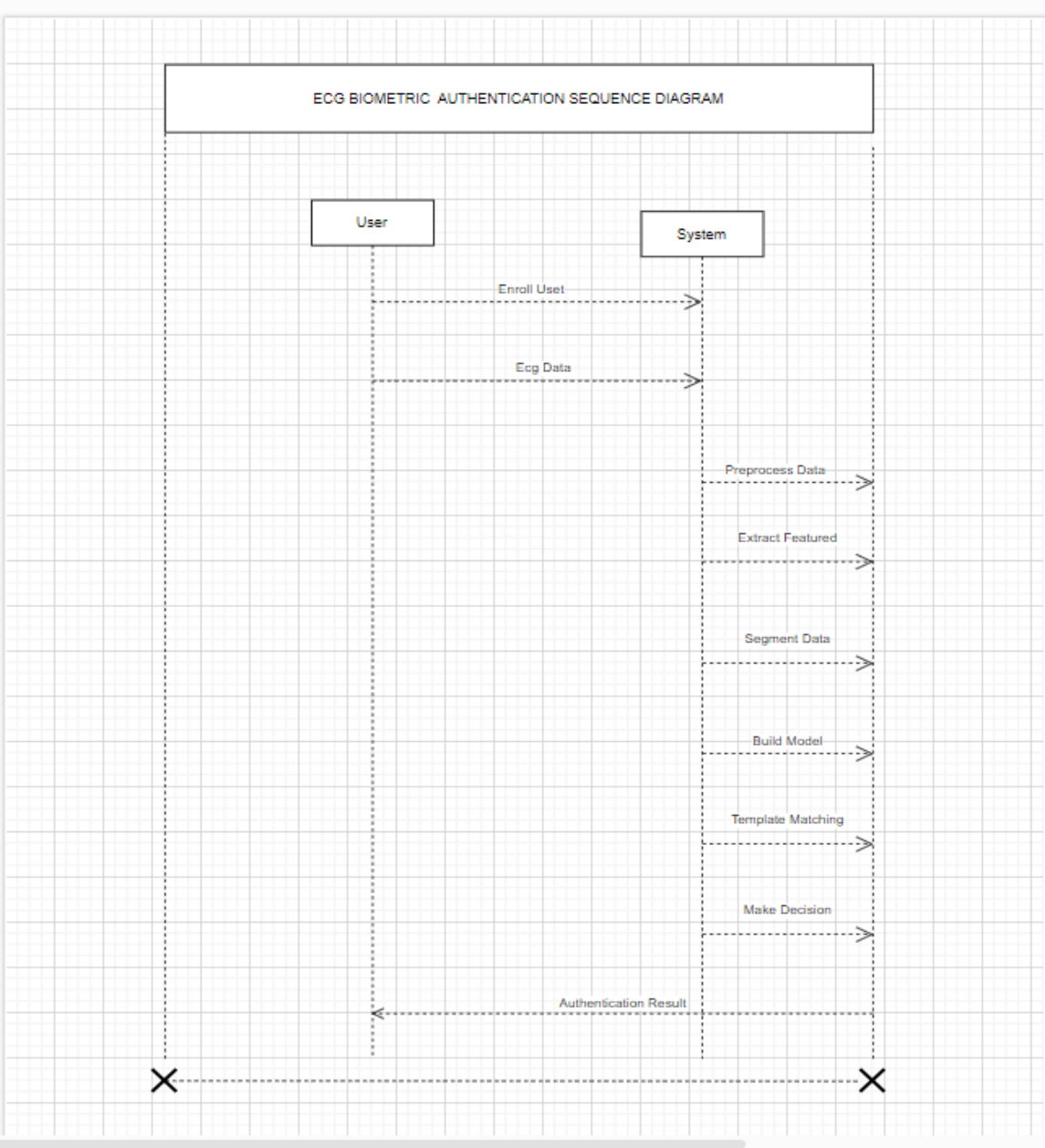


Figure 4.7 Sequence Diagram

4.4.4 Activity diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iterations and concurrency. In the

UML, activity diagrams can be used describe the business and operational stepby-
step workflows of components in a system. An activity diagram shows the overall flow of control. An activity diagram is shown as a rounded box containing the name of the operation.

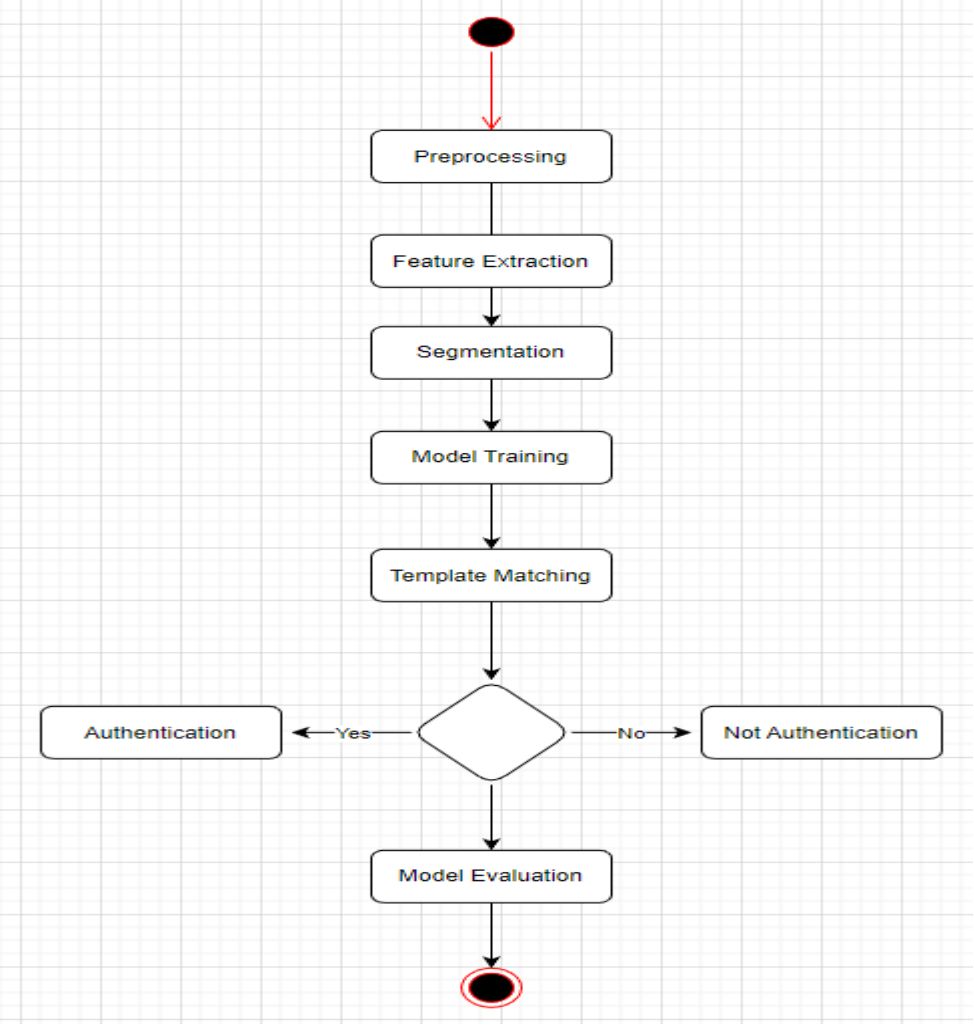


Figure 4.8 Activity Diagram

4.4.5 State chart Diagram

The State Chart Diagram represents the events and states of object and the behaviour of object in reaction to an event. The Start Chart diagram shows

the life cycle of the object. Sometimes it is not possible to show each and every event in the state diagram.

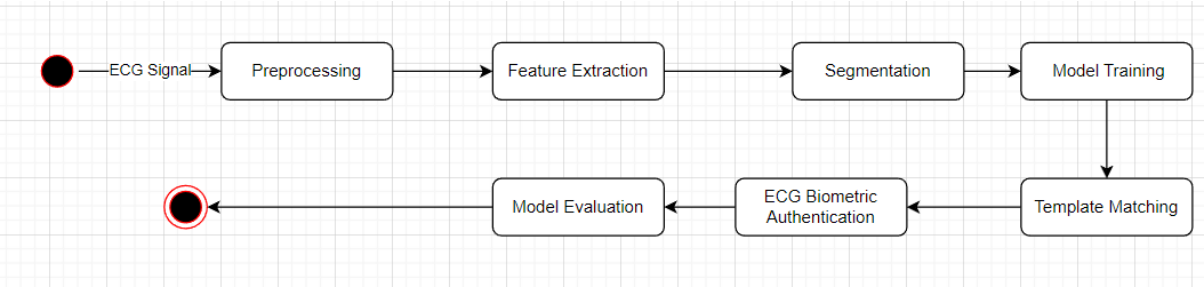


Figure 4.9 Statechart Diagram

4.4.6 Component diagram

A UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

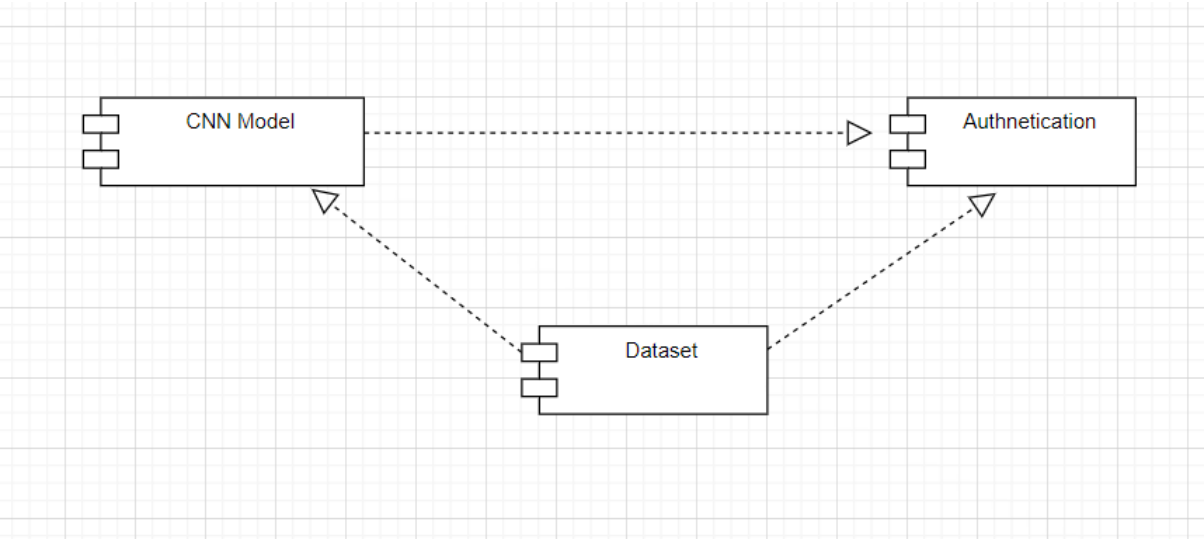


Figure 4.10 Component Diagram

4.4.7 Deployment diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view

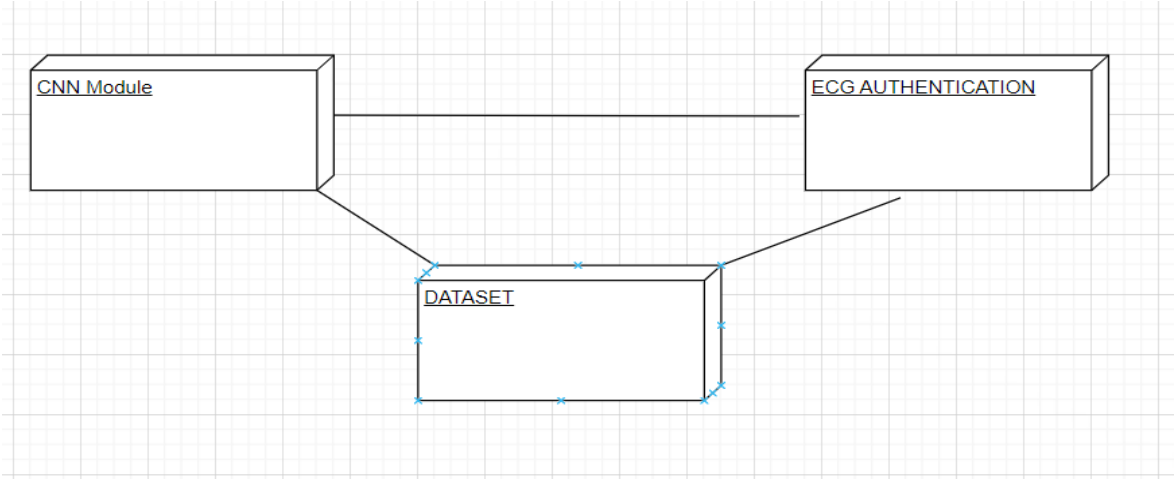


Figure 4.11 Deployment Diagram

CHAPTER 5

IMPLEMENTATION

5.1 MODULES

This project contains three different modules

i. Data collection
ii. Pre processing
iii. Feature extraction
iv. Model building

5.2 MODULE DESCRIPTION

5.2.1 Data Collection

Data collection is the process of gathering and measuring information from countless different sources. Select a diverse group of participants. Consider factors such as age, gender, ethnicity, and health conditions to ensure a representative dataset. We have collected a ECG data from each participant while they are at rest or engaged in specific activities.

5.2.2 Pre processing

The pre-processing module of an ECG architecture involves filtering and normalization of the raw ECG signal to remove noise and baseline wander.

i. Kalman filter can be used to remove noise and artifacts from the ECG signal.

ii. The Polynomial Fitting algorithm involves fitting a polynomial curve to the ECG signal to estimate and remove the baseline wander component.

5.2.3 Feature extraction

The feature extraction module in ECG biometric authentication involves extracting discriminative features from the pre-processed ECG signal, such as R-peak location, heart rate variability etc.,

i. Z-score normalization is a statistical technique used in ECG biometric authentication to standardize the features extracted from ECG signals by subtracting the mean and dividing by the standard deviation.

5.2.4 Model Building

CONVOLUTIONAL NEURAL NETWORK:

- 1D CONVOLUTIONAL LAYER

The convolutional filters can be stacked in multiple layers to extract more complex features from the signal. The parameters of the filters are learned during the training phase of the CNN, which enables the network to adapt to the specific patterns and variations in the ECG signals of each individual.

- MAX POOLING 1D

Max pooling is a downsampling operation commonly used in convolutional neural networks (CNNs) to reduce the spatial dimensions of feature maps. It operates by dividing the input into non-overlapping regions and selecting the maximum value within each region.

5.3 ALGORITHMS AND TECHNIQUES

5.3.1 Pre processing

5.3.1.1 Kalman filter Algorithm:

Input: Raw ECG data

Output: de-noising ECG data

1. Initialization

Initialize the state estimate: $\hat{x}_{0|0}$.

Initialize the error covariance matrix: $P_{0|0}$.

Initialize the process noise covariance matrix: Q.

Initialize the measurement noise covariance matrix: R.

2: Prediction (Time Update):

Project the state estimate ahead: $\hat{x}_{k|k-1} = F \cdot \hat{x}_{k-1|k-1} + B \cdot u_k$.

Project the error covariance ahead: $P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T + Q$.

3: Update (Measurement Update):

Compute the Kalman gain: $K_k = P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1}$.

Update the state estimate: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \cdot (z_k - H \cdot \hat{x}_{k|k-1})$.

Update the error covariance: $P_{k|k} = (I - K_k \cdot H) \cdot P_{k|k-1}$

5.3.1.2 Polynomial fitting Algorithm:

Input: Raw ECG data

Output: Remove baseline

Given a set of data points (x_i, y_i) where i = 1, 2, …., N

Step 1: Choose the degree of the polynomial, denoted by d.

Step 2: Set up the system of equations:

$$\sum(x_i)^2 \cdot a_j + \sum(x_i) \cdot a_{j+1} + ... + \sum(x_i)^{d-1} \cdot a_{j+}{}^{d-1} + N.\ a_j{}^d = \sum(x_i) \cdot y_i \text{ for } j = 0, 1,..d.$$

Step 3: Solve the system of equations to obtain the coefficients that minimize $a_0, a_1, ..., a\_d$ the squared error between the fitted polynomial curve and the data points.

Step 4: The fitted polynomial function can be represented as:

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a\_d \cdot x^d$$

5.3.2 Feature extraction

Z- Score Normalization:

Step 1: Compute the mean (μ) and standard deviation (σ) of the dataset:

$$\mu = (1/N) * \sum(x\_i) \text{ for } i = 1 \text{ to } N$$

$$\sigma = \text{sqrt}((1/N) * \sum((x\_i - \mu)^2)) \text{ for } i = 1 \text{ to } N$$

Step 2: Normalize each data point, x_i, using the Z-score formula:

$$z\_i = (x\_i - \mu) / \sigma$$

5.3.3 Model Building

5.3.3.1 Convolutional 1D:

Output Length = (Input Length – Kernel Size + 2 * Padding) / Stride + 1

Where:

Output Length: The length of the output feature map.

Input Length: The length of the input sequence.

Kernel Size: The size (length) of the convolutional kernel.

Padding: The number of zero-padding elements added to the input sequence.

Stride: The step size of the convolution operation, which determines the amount of shift between successive convolutions.

5.3.3.2 MAX POOLING 1D:

Output Length =floor((Input Length – Pool Size) / Stride) + 1

Where:

Output Length: The length of the output feature map.

Input Length: The length of the input sequence.

Pool Size: The size (length) of the pooling window.

Stride: The step size of the pooling operation, which determines the amount of shift between successive pooling operations.

5.4 MODEL EVALUATION:

- Accuracy = Correct predictions / All predictions
- Precision

  Precision = True positive / (True positive + False positive)
- Recall

  Recall = True positive / (True positive + False Negative)
- F1 Score

  F1 Score = 2 ((Precision * Recall) / Precision + Recall

CHAPTER 6

CODING AND SYSTEM TESTING

6.1 CODING

Once the design aspect of the system finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it should be easily screwed into the system.

6.2 DEVELOPING TECHNOLOGIES

The test process is initiated by developing a compressive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

6.3 SYSTEM TESTING

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes logical assumptions that if all the parts of the system are incorrect, it will handle successfully.

During testing, the program to be tested is executed with set of data and output of the program for the test data is evaluated to determine if the programs are performing as expected. A series of testing are performed for the proposed system before the system is ready for user acceptance testing.

The testing are

- Unit testing
- Functional testing
- Integration testing
- Validation testing
- Output testing
- User acceptance testing
- White box testing
- Black box testing

6.3.1 Unit Testing

In this different test modules are tested against the specification of the modules. Unit testing was done for the verification of the code produced during the coding phase to test the internal logic or modules. It refers to the verification of the single program module in installed environment.

6.3.2 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional is spotlighted at the following items:

Valid Input : Identified classes of valid input must be accepted.

Invalid Input : Identified classes of invalid input must be rejected.

Functions : Identified functions must be exercised.

Output : Identified classes of application output must be exercised.

Systems/Procedures: Interfacing systems or procedures must be invoked. Two types of testing in Functional test:

• Performance testing

• Structure testing

6.3.2.1 Performance Testing

It determines the amount of execution time spent in various parts of the unit, program throughput and response time and device utilization by the program unit.

6.3.2.2 Structure Testing

It is the test designed to intentionally break the system as unit into many small parts or units.

6.3.3 Integration Testing

In this project modules are integrated properly, the emphasis being and testing interfaces between modules. Internal interfaces and external interfaces are tested as each module is incorporated into the structure. This test is designed to uncover errors associated with local or global data structures. It is also designed to verify performance levels established when software design is conducted. Thus, all these modules are combined, verified and the information about the item is properly carried on to the next module and then it is checked.

6.3.4 Validation Testing

At the culmination of integral testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests validation may begin.

Validation can be defined in many ways but a simple definition is that validation succeeds when software function in a manner that can be reasonably expected by the customer.

6.3.5 Output testing

After performing the validation, next step is the output testing of the proposed system. Since no system could be useful if it does not produce the output in the specified formats. The output generator or displayed by the system under consideration is tested for user acceptance by constantly keeping touch with perspective system and user at the time of developing and making changes whenever required.

6.3.6 User Acceptance Testing

Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. This is done regarding to the following points.

- Input Screen design
- Output Screen design
- Format of the report and other output

6.3.7 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least s purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

There are different kinds of Basic path testing:

- Flow graph notation
- Cyclomatic complexity
- Deriving test cases

6.3.8 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. It is a testing in which the software under test is treated, as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

CHAPTER 7

PERFORMANCE ANALYSIS

Performance analysis and result reporting are essential components of any deep learning project. The evaluation of a model's performance is necessary to determine how well it is performing in a given task. There are various performance metrics to evaluate the model, such as accuracy, precision, recall, F1-score.

In deep learning, the performance of the model is often evaluated on a separate test dataset that was not used during training. This approach ensures that the model's performance is generalizable and not just good at memorizing the training data.

ROC Curve and AUC: Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model. It plots the true positive rate (TPR) against the false positive rate (FPR) at different threshold settings. The area under the curve (AUC) of the ROC curve is used as a measure of the model's performance.

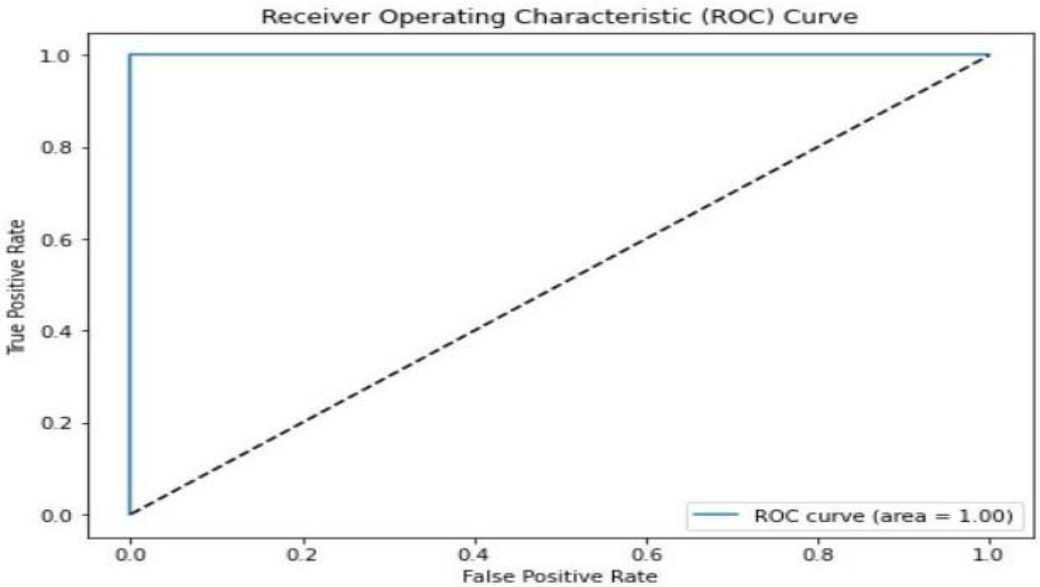


Figure 7.1 ROC Curve

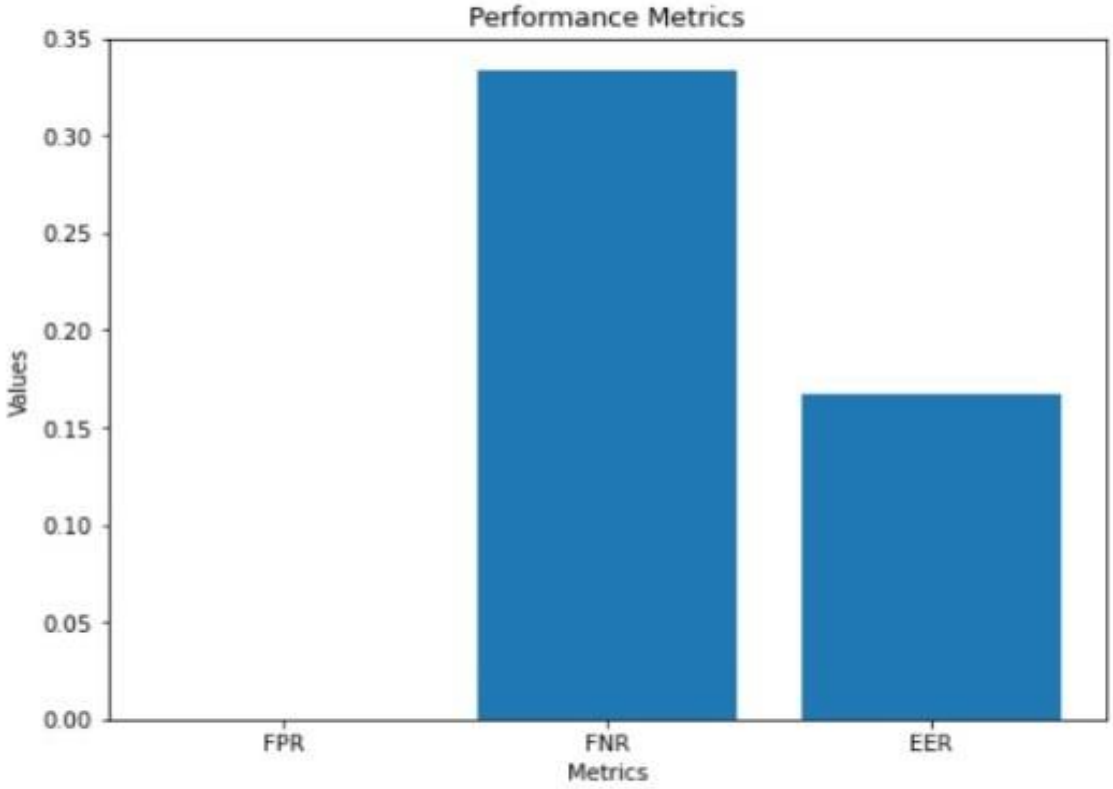
Precision, Recall, and F1-score: These metrics are commonly used in binary and multi-class classification tasks. Precision measures the proportion of true positives among all instances predicted as positive. Recall measures the proportion of true positives among all actual positive instances. F1-score is the harmonic mean of precision and recall and provides a balanced measure of performance for imbalanced datasets.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       809
           1       1.00      0.67      0.80         3

    accuracy                           1.00       812
   macro avg       1.00      0.83      0.90       812
weighted avg       1.00      1.00      1.00       812

Precision: 1.0
Recall: 0.6666666666666666
F1 Score: 0.8
Accuracy: 0.9987684729064039
```

Figure 7.2: Precision, Recall, F1 Score

FAR represents the rate at which the system incorrectly accepts an imposter or unauthorized user. FRR measures the rate at which the system incorrectly rejects a legitimate user. In ECG biometrics authentication, a high FRR indicates that the system is overly cautious and may reject valid users, resulting in inconvenience and decreased usability. ERR is the point where the FAR and FRR values are equal. It represents the balance between false acceptances and false rejections. A lower ERR indicates a more accurate and balanced authentication system.

```
FAR: 0.0
FRR: 0.3333333333333333
EER: 0.16666666666666666
```




```
False Positive Rate (FPR): [0. 0. 0. 1.]
False Negative Rate (FNR): 0.3333333333333333
Equal Error Rate (EER): 0.16666666666666666
```

Figure 7.3 Evaluation Metrics

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

EDITH: ECG Biometric Aided by Deep Learning for Reliable Individual Authentication is a promising technology that utilizes deep learning algorithms to authenticate individuals based on their unique electrocardiogram (ECG) signals. The use of ECG biometrics for individual authentication is a novel and secure method that has the potential to revolutionize the field of biometric authentication.

The unique electrical activity of the heart provides a highly personalized biometric signature that is difficult to replicate or steal. The use of deep learning algorithms enhances the accuracy and robustness of the authentication system.

However, there are still some challenges that need to be addressed in the development and implementation of ECG biometric authentication systems, such as the need for specialized ECG sensors, potential privacy concerns, and the need for further research to validate the effectiveness of the system. Nevertheless, ECG biometric authentication has the potential to provide a secure and convenient means of authentication in various applications, including healthcare, finance, and security

8.2 FUTURE WORK

Future enhancements for EDITH could include improving the system's performance with noisy ECG signals, as well as expanding the system's capabilities to include multi-modal biometric authentication. The use of wearable ECG sensors would make EDITH more practical and accessible,

allowing individuals to authenticate themselves on-the-go, without the need for specialized equipment. Additionally, the system could be integrated with wearable devices to provide continuous authentication and monitoring of the user's ECG signals. Another potential enhancement could be the development of a user-friendly interface for EDITH, making it more accessible to a wider range of users. It could also be interesting to investigate the possibility of using EDITH for continuous authentication in high-security environments, such as military or government facilities, where reliable individual authentication is crucial.

APPENDICES

APPENDIX 1: SAMPLE CODING

PREPROCESSING:

```
import numpy as np
from scipy.signal import savgol_filter
from pykalman import KalmanFilter

 # Load ECG data from CSV file
ecg = np.genfromtxt('', delimiter=',')

# Remove NaN values
ecg = ecg[~np.isnan(ecg).any(axis=1)]

# Apply Savitzky-Golay filter to remove noise
ecg = savgol_filter(ecg, window_length=5, polyorder=2, axis=0)

# Apply Kalman filter to remove baseline drift
kf = KalmanFilter(n_dim_obs=ecg.shape[1], n_dim_state=ecg.shape[1])
ecg = kf.em(ecg).smooth(ecg)[0]

# Apply polynomial fitting algorithm to remove baseline wander
for i in range(ecg.shape[1]):
    p = np.polyfit(np.arange(ecg.shape[0]), ecg[:, i], deg=5)
    ecg[:, i] = ecg[:, i] - np.polyval(p, np.arange(ecg.shape[0]))

# Save preprocessed ECG data to CSV file
```

```
np.savetxt('preprocessed_ecg_data1.csv', ecg, delimiter=',')
```

FEATURE EXTRACTION:

```
import numpy as np
import pandas as pd
from scipy.stats import kurtosis, skew
from scipy.signal import welch

# Load the preprocessed dataset
preprocessed_data = pd.read_csv('preprocessed_ecg_data1.csv')

# Extract ECG signals from the dataset
ecg_signals = preprocessed_data.iloc[:, 1:].values  # Assuming ECG signals are in columns 1 to 4

# Initialize an empty list to store the extracted features
 features = []

 # Iterate over each ECG signal in the dataset
for ecg_signal in ecg_signals:

 # Statistical features
 mean = np.mean(ecg_signal)
 std = np.std(ecg_signal)
 kurt = kurtosis(ecg_signal)
 skewness = skew(ecg_signal)

 # Frequency-domain features
```

```
 f, psd = welch(ecg_signal)

peak_freq = f[np.argmax(psd)]

total_power = np.sum(psd)

# Append the features to the list

features.append([mean, std, kurt, skewness, peak_freq, total_power])

# Convert the list of features to a NumPy array

features = np.array(features)

# Save the extracted features to a new CSV file

feature_df = pd.DataFrame(features, columns=['mean', 'std', 'kurtosis', 'skewness', 'peak_freq', 'total_power'])

feature_df.to_csv('ecg_features9091.csv', index=False)
```

MODEL BUILDING:

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense

from tensorflow.keras.optimizers import Adam

import pandas as pd

from tensorflow import keras

# Load the segmented dataset from CSV
```

```
dataset = pd.read_csv('segment101.csv')

# Extract features and labels
X = dataset.iloc[:, :5].values  # Adjust the column range according to your dataset
y = dataset.iloc[:, 5].values

# Set the threshold for converting continuous labels to binary
threshold = 0.5

# Convert labels to binary based on the threshold
y_binary = np.where(y >= threshold, 1, 0)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2, random_state=42)

# Reshape the data for model input
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Define the CNN model
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(5, 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
```

```
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])


# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test,
y_test))

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_binary = np.where(y_pred >= threshold, 1, 0)
accuracy = (y_test == y_pred_binary).mean()
print("Accuracy:", accuracy)

# Load the model
model.save('cnn_model10100.h5')
```

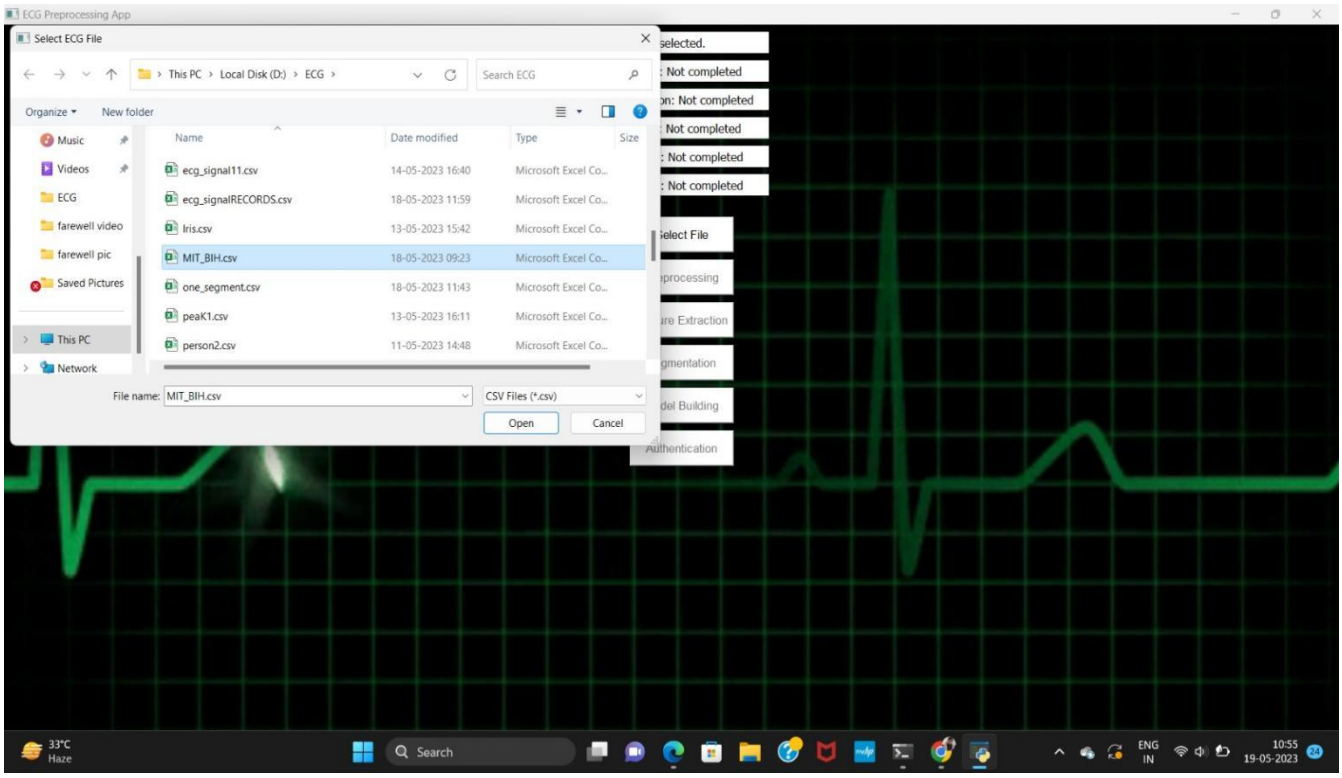
COMPARING:

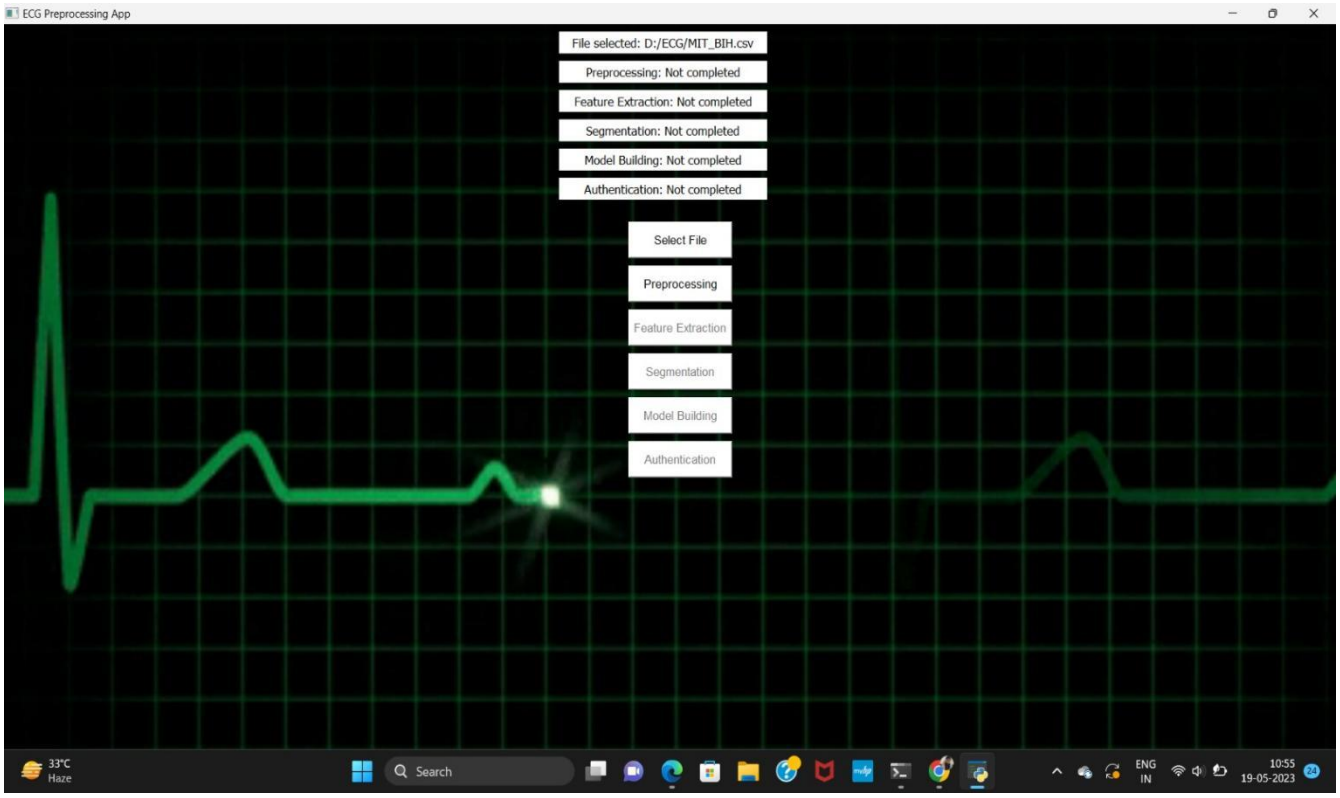
```
import pandas as pd
import numpy as np
from keras.models import load_model

# Load the pre-trained model
model = load_model(r"C:\Users\Shan\Desktop\ECG\cnn_model10100.h5")
```

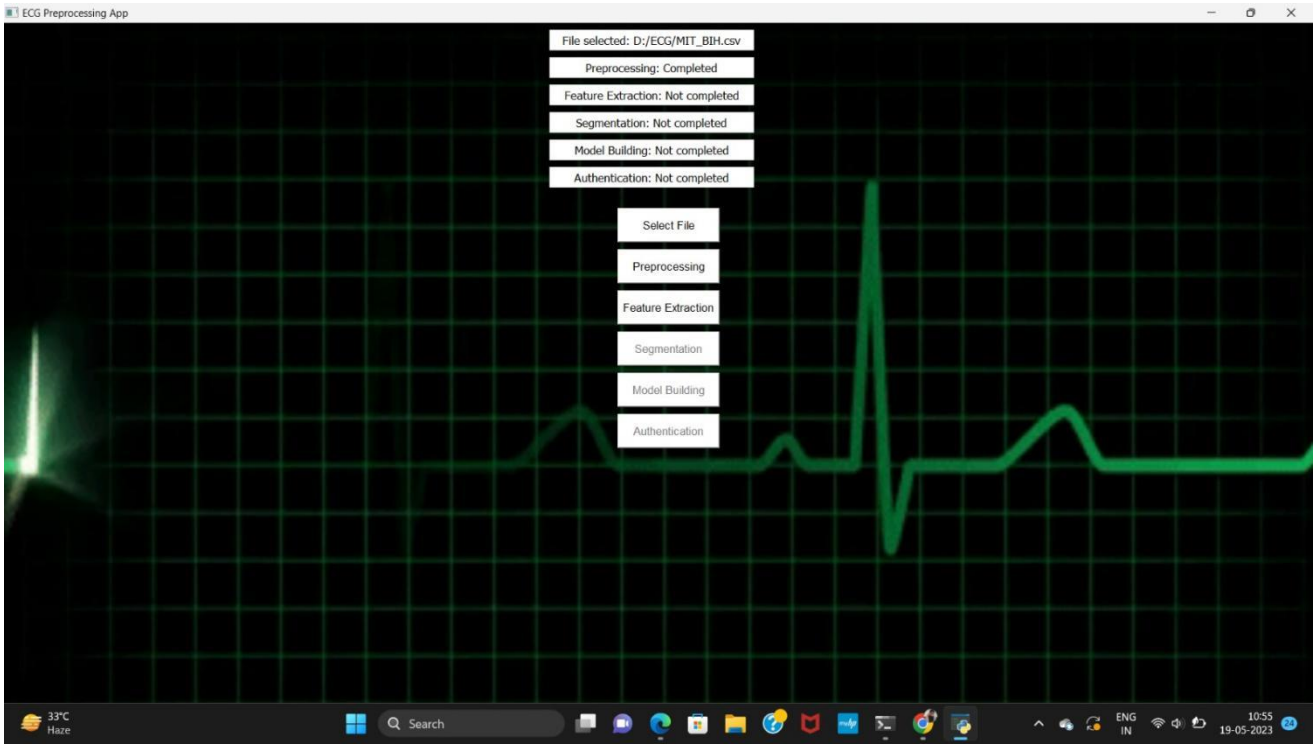
```
# Load the segmented dataset from CSV
segmented_data = pd.read_csv(r"C:\Users\Shan\Desktop\ECG\ecg_signal.csv")

# Adjust the shape of the segmented data
segmented_data = segmented_data.values  # Convert to numpy array
segmented_data = np.reshape(segmented_data[:, :5], (segmented_data.shape[0], 5, 1))  # Reshape to (n_samples, 5, 1)

# Feed the reshaped data to the model
prediction = model.predict(segmented_data)


# Make the authentication decision
threshold = 0.5  # Define your threshold
is_authenticated = np.all(prediction > threshold)


# Print the authentication decision
if is_authenticated:
    print("Authentication successful")
else:
    print("Authentication failed")
```

APPENDIX 2: SCREENSHOTS

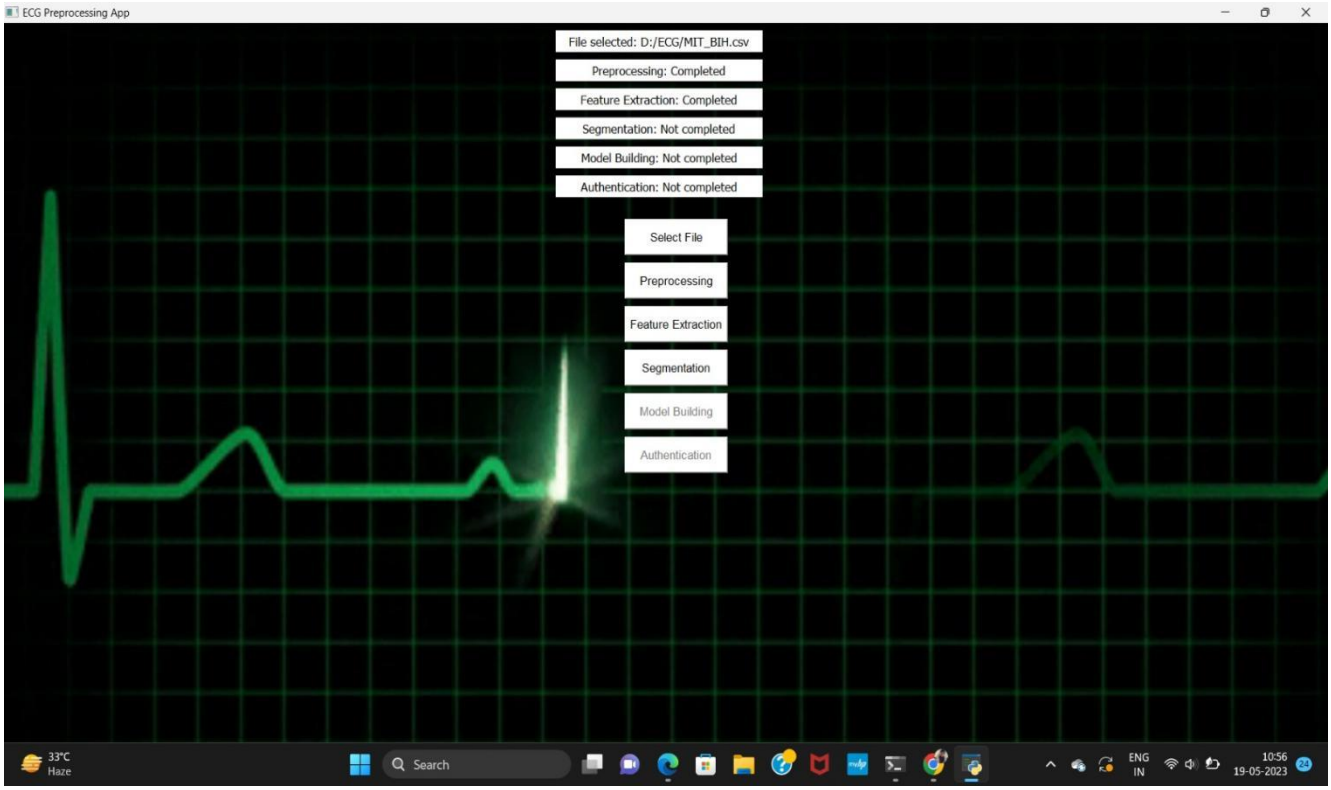
MODULE 1: PREPROCESSING

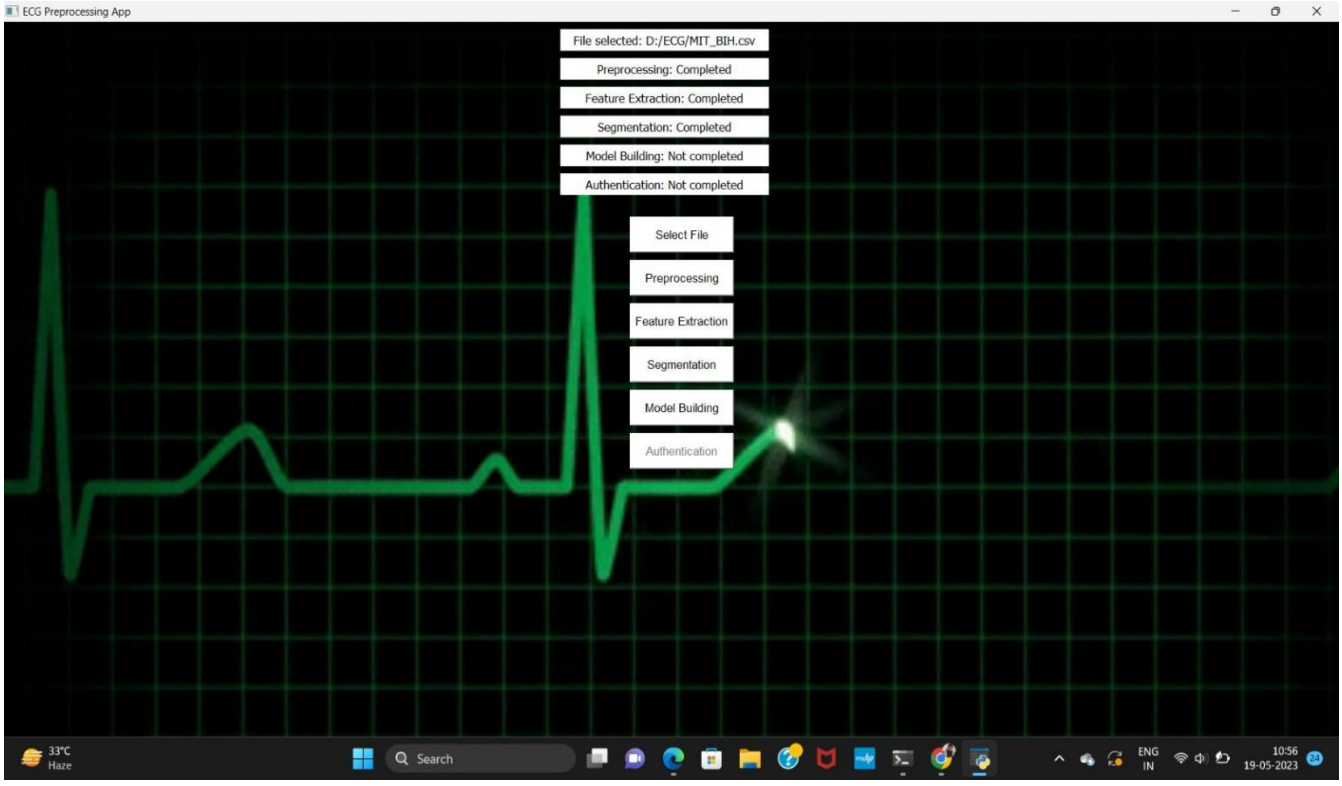




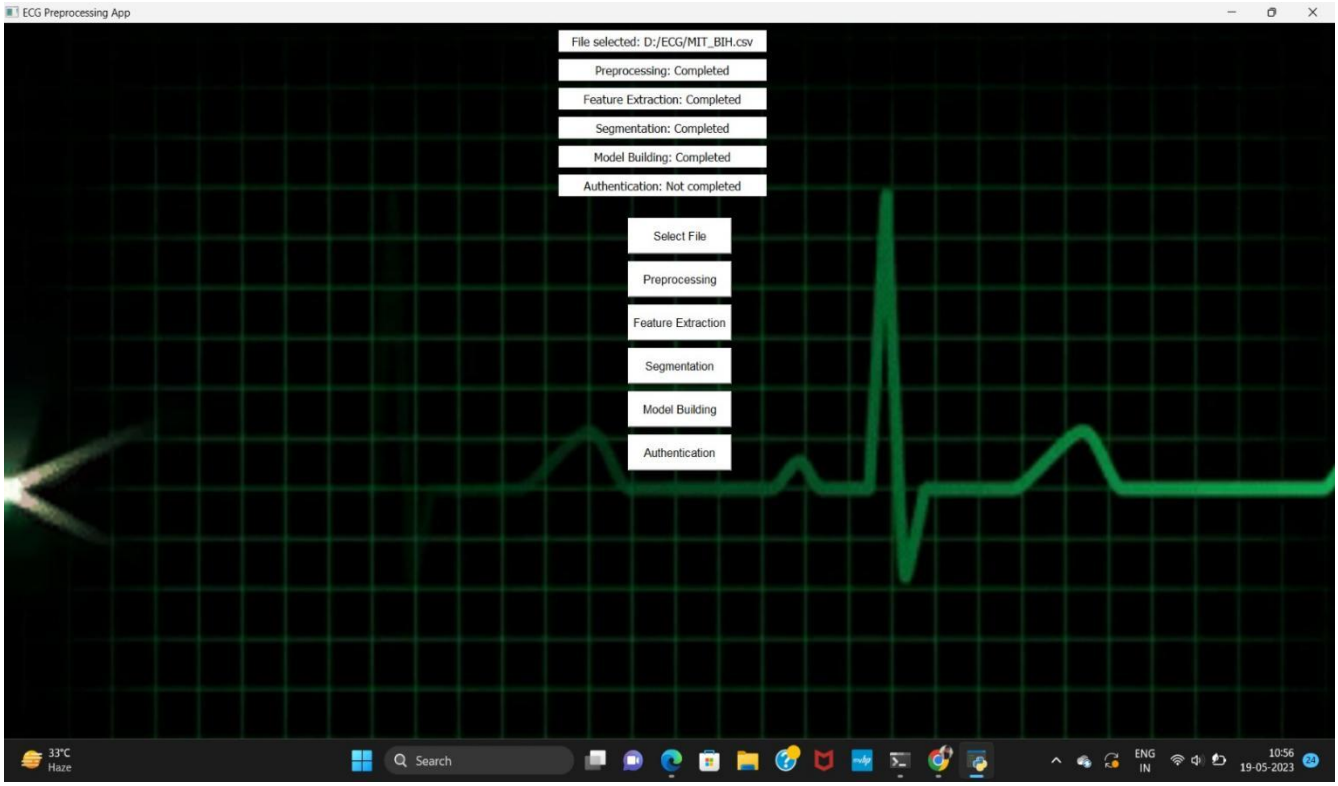


MODULE 2: FEATURE EXTRACTION

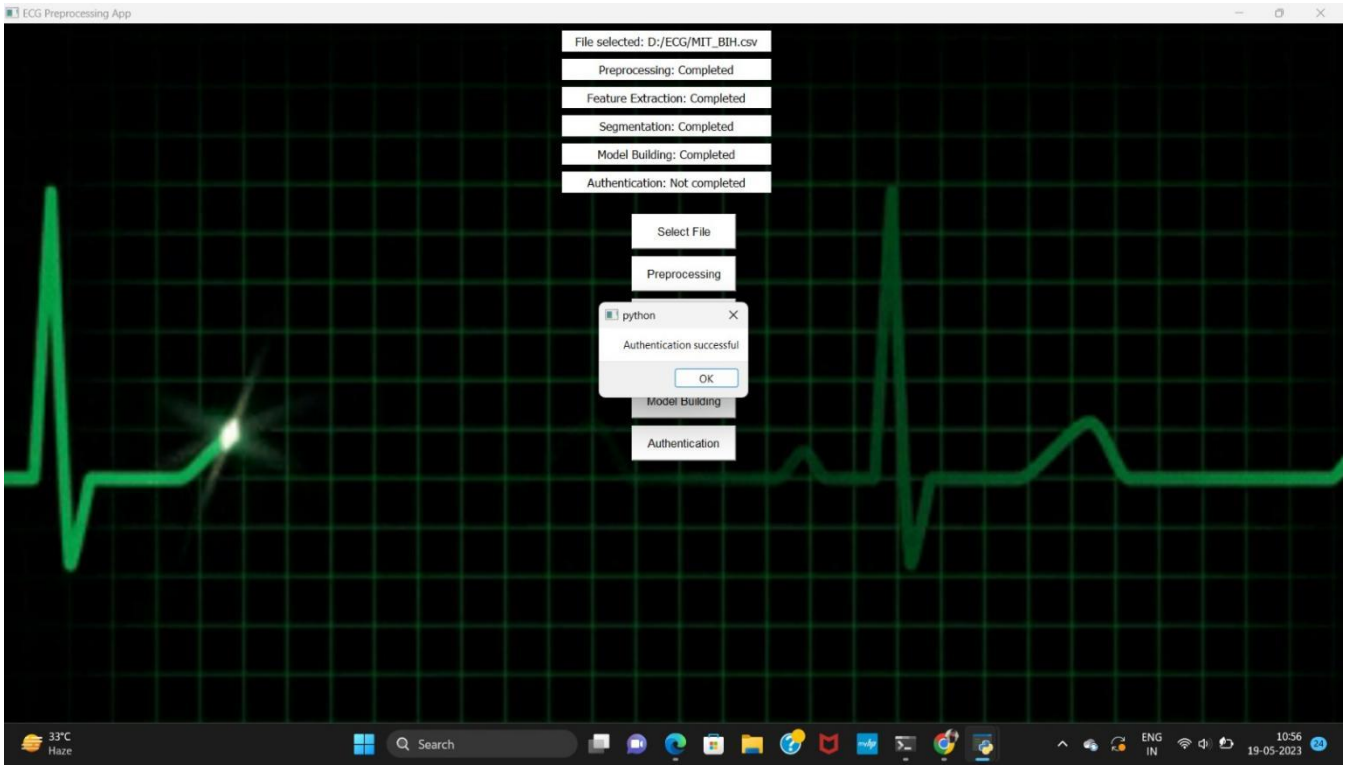


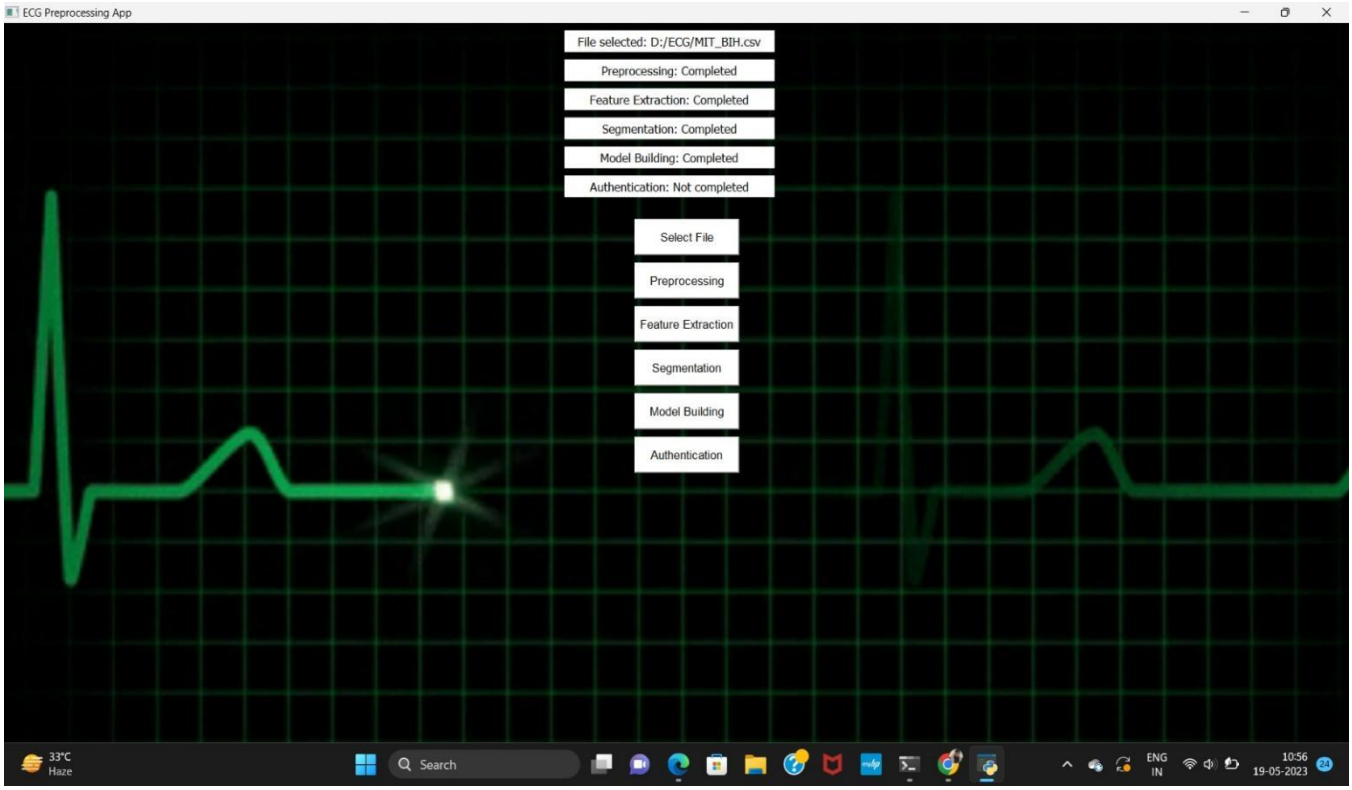


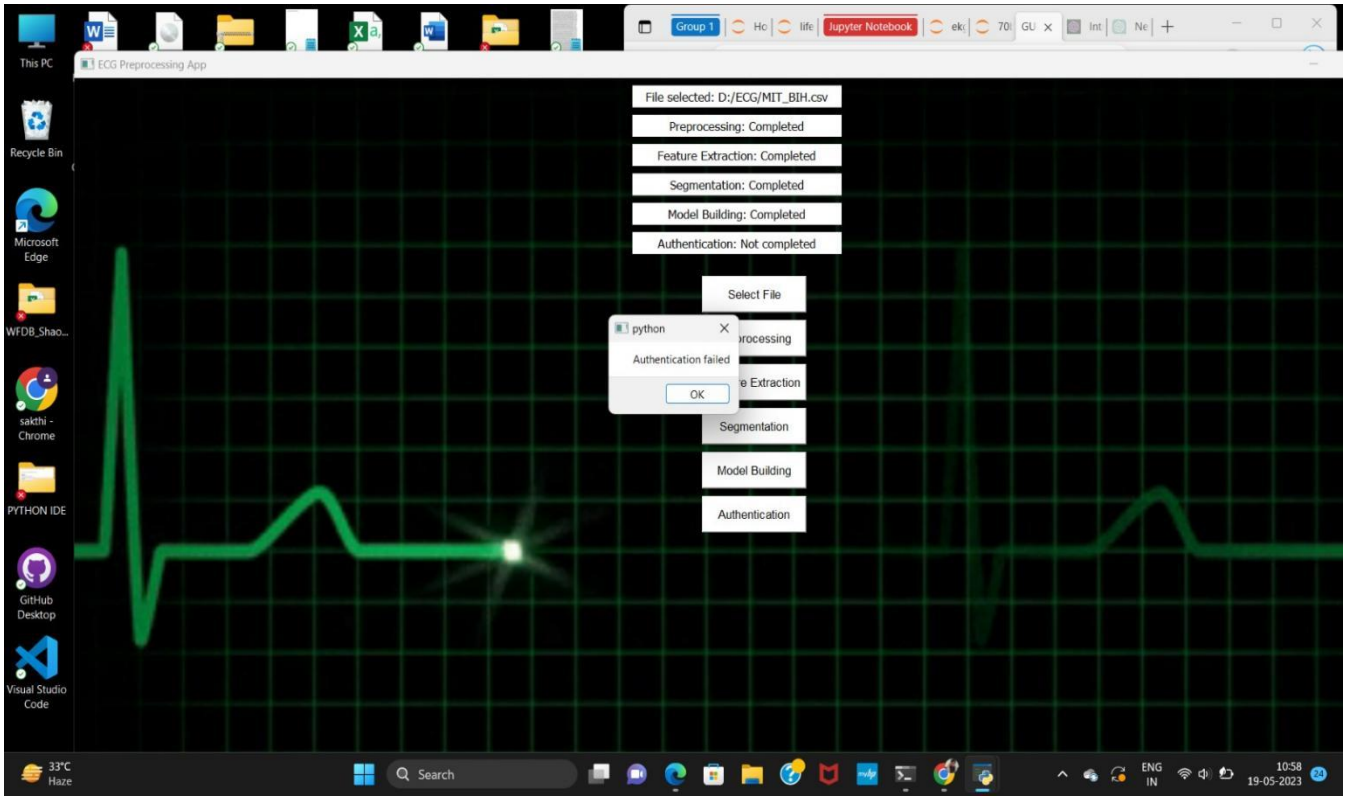
MODULE 3: MODEL BUILDING


ECG Preprocessing App
File selected: D:/ECG/MIT_BIH.csv
Preprocessing: Completed
Feature Extraction: Completed
Segmentation: Completed
Model Building: Completed
Authentication: Not completed
Select File
Preprocessing
Feature Extraction
Segmentation
Model Building
Authentication
33°C
Haze
Search
ENG
IN
10:56
19-05-2023

RESULT:


ECG Preprocessing App
File selected: D:/ECG/MIT_BIH.csv
Preprocessing: Completed
Feature Extraction: Completed
Segmentation: Completed
Model Building: Completed
Authentication: Not completed
Select File
Preprocessing
python
Authentication successful
OK
Model Building
Authentication
33°C
Haze
Search
ENG
IN
10:56
19-05-2023


ECG Preprocessing App
File selected: D:/ECG/MIT_BIH.csv
Preprocessing: Completed
Feature Extraction: Completed
Segmentation: Completed
Model Building: Completed
Authentication: Not completed
Select File
Preprocessing
Feature Extraction
Segmentation
Model Building
Authentication
33°C
Haze
Search
ENG
IN
10:56
19-05-2023


This PC
Recycle Bin
Microsoft Edge
WFDB_Shao...
sakthi - Chrome
PYTHON IDE
GitHub Desktop
Visual Studio Code
ECG Preprocessing App
File selected: D:/ECG/MIT_BIH.csv
Preprocessing: Completed
Feature Extraction: Completed
Segmentation: Completed
Model Building: Completed
Authentication: Not completed
Select File
python
Authentication failed
OK
Segmentation
Model Building
Authentication
33°C
Haze
Search
ENG
IN
10:58
19-05-2023

REFERENCE

1. E.Maiorana (2019), EEG-based biometric verification using siamese CNNs, in Proc. Int. Conf. Image Anal. Process, pp. 3–11.

2. A. K. Jain and K. Nandakumar, Biometric authentication: System security and user privacy, IEEE Comput., vol. 45, no. 11, pp. 87–92.

3. L. Biel, O. Pettersson, L. Philipson, and P. Wide, ECG analysis: A new approach in human identification, IEEE Trans. Instrum. Meas., vol. 50, no. 3, pp. 808–812.

4. D. Belo, N. Bento, H. Silva, A. Fred, and H. Gamboa, ECG biometrics using deep learning and relative score threshold classification, Sensors, vol. 20, no. 15 (2020), Art. no. 4078.

5. J. Unar, W. C. Seng, and A. Abbasi, A review of biometric technology along with trends and prospects, Pattern Recognit., vol. 47, no. 8, pp. 2673–2688.