A PROJECT REPORT ON

"Automatic cardiac Arrhythmia classification using SVM"

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

Ankita Thorat	72036943G
Deepak Mustare	72036761B
Nikita Sonar	71922843B
Siddhesh Chavan	71922488J

DEPARTMENT OF COMPUTER ENGINEERING

DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH

AKURDI, PUNE 411044

SAVITRIBAI PHULE PUNE UNIVERSITY

2022 -2023

CERTIFICATE

This is to certify that the project report entitles

“Automatic cardiac Arrhythmia classification

using SVM”

Submitted by

Ankita Thorat	72036943G
Deepak Mustare	72036761B
Nikita Sonar	71922843B
Siddhesh Chavan	71922488J

are bonafide students of this institute and the work has been carried out by him/her under the supervision of **Prof. Pooja Mishra** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

Prof. Pooja Mishra
Guide
Computer Engineering

Prof. P. P. Shevatekar
Head of the Department
ComputerEngineering

Dr. Anupama V. Patil
Principal,
Dr. D. Y. Patil Institute of Engineering, Management & Research Akurdi Pune – 411044

Place : Pune
Date :

External Examiner’s Name:
Sign:

ACKNOWLEDGEMENT

This is a great pleasure and immense satisfaction to express our deepest sense of gratitude and thanks to everyone who has directly or indirectly helped us in completing our project work successfully.

We express my gratitude towards Project Guide **Prof. Pooja Mishra** and **Mrs. P. P. Shevatekar**, Head of Department of Computer Engineering, Dr. D Y Patil Institute of Engineering, Management and Research, Akurdi who guided and encouraged us in completing the seminar work in scheduled time. We would like to thanks our Principal Dr. A. V. Patil, for allowing us to pursue our project in this institute.

Finally, we would like to thank our friends who have directly or indirectly helped us in our project work.

Ankita Thorat
Deepak Mustare
Nikita Sonar
Siddhesh Chavan
(B.E. COMPUTER ENG.)

ABSTRACT

ECG plays a vital role in diagnosis of many such cardiac disorders. Some arrhythmias including ventricular fibrillation and premature ventricular contraction can be fatal if not dealt on time Ventricular and atrial tachycardias are some of the most common clinical cardiac arrhythmias. Therefore, accurate automated arrhythmia detection systems for various types of arrhythmias are required in clinical settings. Recently, deep learning has been actively used for arrhythmia diagnosis, resulting in significant performance improvements. In this study, we classify Cardiac Arrythmia using Support-Vector machine. Cardiac arrhythmias occur in a short duration; therefore, it is difficult to distinguish them using the human eye directly. Cardiac arrhythmia is associated with abnormal electrical activities of the heart, which can be reflected by altered characteristics of electrocardiogram (ECG). Due to the simplicity and noninvasive nature, the ECG has been widely used for detecting arrhythmias and there is an urgent need for automatic ECG detection. We used the Support Vector Machine (SVM) classifier and the best result we chose had a global accuracy greater 80%, while still obtaining relatively high Sensitivities and Positive Predictive Value and low False Positive Rates. Thus, this method presents a noble, accurate, and reliable method for classification of cardiac arrhythmia beats.

TABLE OF CONTENTS

LIST OF ABBREVATIONS

ABBREVIATION	ILLUSTRATION
HTTP	Hyper Text Transfer Protocol
MQTT	Message Query Telemetry Transport
SVM	Support Vector Machine
GUI	Graphic User Interface
ECG	Electrocardiogram

LIST OF FIGURES

1. INTRODUCTION

1.1 OVERVIEW

Different cardiologists hold varying perspectives on the differentiation of arrhythmias. Consequently, there is a need for precise automated systems capable of detecting various types of arrhythmias in clinical environments. Recently, deep learning has emerged as a prominent approach for diagnosing arrhythmias, leading to notable advancements in performance. However, challenges persist when it comes to implementing these systems practically. One key challenge involves extracting distinctive features specific to arrhythmias from electrocardiogram (ECG) signals using deep learning algorithms. ECG signals exhibit unique morphological and temporal characteristics, resulting in diverse ECG morphologies for a single arrhythmia across different patients. Consequently, extracting distinctive ECG features for specific cardiac conditions using deep learning poses a significant challenge.
By successfully completing this project, a reliable and user-friendly system is created that can aid in the classification of cardiac arrhythmias using an SVM algorithm. The Tkinter GUI enhances the usability and accessibility of the system, allowing healthcare professionals or users to input data and obtain real-time predictions conveniently.

1.2 MOTIVATION

The motivation behind the project stems from the need to improve the accuracy and efficiency of diagnosing cardiac arrhythmias. Cardiac arrhythmias are abnormal heart rhythms that can have serious health implications and may require timely intervention for proper treatment. However, accurately classifying different types of arrhythmias can be challenging for medical professionals due to the complexity and variability of these conditions.

Overall, the project's motivation lies in leveraging machine learning techniques, specifically SVM, and GUI development to create an automated classification system that improves the accuracy, efficiency, and consistency of diagnosing cardiac arrhythmias. By addressing these aspects, the project aims to enhance patient care, support medical professionals, and potentially save lives through timely and accurate arrhythmia identification.

1.3 PROBLEM DEFINITION

The project aims to develop a system that can accurately classify different types of cardiac arrhythmias based on input data, using a Support Vector Machine (SVM) algorithm. The system will be integrated with a graphical user interface (GUI) built using Tkinter, allowing users to conveniently input data and receive real-time predictions.

By addressing these aspects, the project aims to provide a reliable and user-friendly system for accurate classification of cardiac arrhythmias using SVM. The integration of Tkinter GUI enhances the usability and accessibility of the system, allowing healthcare professionals to conveniently input data and receive real-time predictions, aiding in the diagnosis and treatment of cardiac arrhythmias.

1.4 LIMITATIONS

- Users must have knowledge of English.
- Users must have all required software to run the application.
- It requires internet connection.
- Dependency on Data Quality:
- Limited Generalization
- Interpretable Feature Extraction
- Complexity of Arrhythmia Patterns

It is important to acknowledge these limitations and conduct thorough analysis, validation, and continuous improvement of the system to mitigate these challenges and ensure its safe and effective use in a clinical setting.

1.5 METHODOLOGIES OF PROBLEM SOLVING

To approach problem-solving we followed these methodologies:

1. Define the Problem: Clearly understand the problem statement and the goals of the project. Identify the specific requirements and constraints, such as the expected accuracy of classification, the types of arrhythmias to be classified, and the features to be used.

2. Gather Information: Gather relevant information about cardiac arrhythmia, SVM classification, and GUI development using Tkinter. Understand the basics of ECG data, arrhythmia classification methods, and the SVM algorithm. Explore the functionalities and capabilities of Tkinter for building GUI applications.

3. Break down the Problem: Break down the problem into smaller, manageable tasks. Identify the steps involved, such as data preprocessing, SVM model training, GUI design, and result display.

4. Design the Solution: Design a solution by determining the appropriate data preprocessing techniques, SVM model parameters, and GUI layout. Decide on the input fields, buttons, and result display mechanisms in the Tkinter GUI. Consider the user experience and ease of use.

5. Implement the Solution: Implement the solution by writing the code. Develop the data preprocessing functions, SVM classification algorithms, and Tkinter GUI components based on the defined design. Test each component independently to ensure they work as expected.

6. Integrate and Test: Integrate the individual components, such as data preprocessing, SVM classification, and Tkinter GUI, to create a complete system. Test the integrated system thoroughly to verify its functionality, accuracy, and performance. Write unit tests to validate the correctness of the implementation.

7. Debug and Refine: Identify and fix any bugs or issues that arise during testing. Refine the code and algorithms to improve the accuracy and usability of the classification system. Consider feedback from users or stakeholders to enhance the GUI interface and overall user experience.

8. Validate and Evaluate: Validate the classification system by comparing its results with known ground truth or expert diagnoses. Evaluate the system's performance using appropriate metrics, such as accuracy, precision, recall, or F1 score. Assess the usability and effectiveness of the Tkinter GUI based on user feedback.

9. Document and Communicate: Document the entire process, including the problem statement, methodologies, implementation details, and evaluation results. Create clear and concise documentation for future reference. Communicate the findings, limitations, and recommendations to stakeholders or peers involved in the project.

10. Iterate and Improve: Based on the evaluation and feedback, iterate on the solution to improve its performance, accuracy, and user experience. Consider incorporating additional features or techniques to enhance the classification system. Continuously refine and update the system as needed.

2. LITERATURE SURVEY

Sr. No	Paper Title	Journal Name	Authors & Publication Date	Methodology
1	A Telemetry System for the Study of Spontaneous Cardiac Arrhythmias	IEEE	Dennis L. Rollins, Cheryl R. Killingsworth, Gregory P. Walcott, R. Kyle Justice	To study arrhythmias in animal models, an eightchannel implantable radio telemetry system has been developed to record continuously cardiac electrograms over a period of weeks to months, with maintenance restricted to changing batteries.
2	Comparison of Arrhythmia Prevalence in NUVANT Mobile Cardiac Telemetry System Patients in the US and India.	IEEE	Jonathan M. Engel, Niranjan Chakravarthy, Grant Nosbush, Matt Merkert, Richard Fogoros M.D., Abhi Chavan	Analysis of a sample of 2231 US and 1053 Indian NUVANT patients has revealed interesting statistics on the prevalence of various cardiac arrhythmias in the patient populations of the two nations. The population is non-randomized and consists of US patients where often a traditional Holter Monitor study was negative and self-pay Indian patients.
3	Classifying Normal Sinus Rhythm and Cardiac Arrhythmias in ECG Signals Using Statistical Features in Temporal Domain	IEEE	Mavera Mazhar bhatt, Usman Akram, Shoab A. Khan	Any morphological abnormality or atypical group of conditions in a cardiac rhythm indicates a typical class of arrhythmias. ECG plays a vital role in diagnosis of many such cardiac disorders. Some arrhythmias including ventricular fibrillation and premature ventricular contraction can be fatal if not dealt on time.

4	Cardiac propagation Pattern Mapping with Vector Field for Helping Tachyarrhythmias Diagnosis with Clinical Tridimensional Electro Anatomical Mapping Tools	IEEE	Corentin Dallet, Caroline Roney, Ruairidh Martin, Takeshi Kitamura, Stephane Puyo, Josselin Duchateau, Carole DumasPomier, Gwladys Ravon	Invasive electro-anatomical mapping for an accurate diagnosis using electrograms (EGMs) acquired with intra-cardiac catheters and localized on the surface mesh of the studied cavities and non-invasive electrocardiographic imaging (ECGi) for a global view of the arrhythmia, with EGMs mathematically reconstructed from body surface electrocardiograms using 3-D cardio-thoracic surface meshes obtained from CT-scans
5	A Wearable Auto-Patient Adaptive ECG Processor for Shockable Cardiac Arrhythmia	IEEE	Syed Muhammad Abubakar, Muhammad Rizwan Khan, Wala Saadeh, Muhammad Awais Bin Altaf.	A non-machine learning patient-specific shockable cardiac arrhythmia-(SCA) classification processor based on single lead electrocardiogram (ECG)is presented. The proposed SCA detection processor integrates a hardware efficient reduced-set-of-five (RSF5) Computer Engineering department, DYPIEMR,‘22-’23 14 feature extraction engine to extract SCA and non-SCA, self-adaptive patient-specific threshold engine for the peak and interval detection from the ECG, and simplified decision logic to discriminate the arrhythmia in real-time

3. SOFTWARE REQUIREMENT

3.1. ASSUMPTIONS AND DEPENDENCIES

Input: Input as textual dataset

Domain: Machine Learning

3.2. FUNCTIONAL REQUIREMENTS

Proposed system consists of 4 modules:

- Feature point extraction: Feature points of each image gets detected.
- Feature correspondence matching: Matching of selected feature points across various image frames.
- Point estimation: Position estimation and vision system orientation during navigation.
- Position refinement: Location estimate based, accurate location derivation.

3.3. EXTERNAL INTERFACE REQUIREMENTS

3.3.1. USER INTERFACES

Application of Tkinter Framework to build a User-friendly UI.

3.3.2. HARDWARE INTERFACES

- RAM: 8 GB (As we are using Machine Learning Algorithm and Various High-Level Libraries)
- Laptop RAM: minimum 8 GB.
- Hard Disk: 40 GB.
- Processor: Intel i5 Processor (Spyder ide environment is to be used and data loading should be fast hence fast processor is required)
- IDE: Spyder
- Coding Language: Python Version 3.5 (Highly specified Programming Language for Machine Learning because of availability of High-Performance Libraries.)
- Operating System: Windows 11 (Latest Operating System that supports all type of installation and development Environment)

3.3.3. SOFTWARE INTERFACES

• Operating System: Windows 11

• IDE: Spyder

• Programming Language: Python

3.4. NONFUNCTIONAL REQUIREMENTS

3.4.1. PERFORMANCE REQUIEMENTS

- The performance of the functions and every module must be well.
- The overall performance of the software will enable the users to work efficiently.
- Performance of encryption of data should be fast.
- Performance of the providing virtual environment should be fast Safety Requirement.
- The application is designed in modules where errors can be detected. This makes it easier to install and update new functionality if required.

3.4.2. SAFETY REQUIREMENTS

The application is designed in modules where errors can be detected and fixed easily.

This makes it easier to install and update new functionality if required.

3.4.3. SOFTWARE QUALITY ATTRIBUTE:

Our software has many quality attributes that are given below:

Adaptability: This software is adaptable by all users.

Availability: This software is freely available to all users. The availability of the software is easy for everyone.

Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.

Reliability: The performance of the software is better which will increase the reliability of the Software.

User Friendliness: Since, the software is a GUI application; the output generated is much user friendly in its behavior.

Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.

Security: Users are authenticated using many security phases so reliable security is provided.

Testability: The software will be tested considering all the aspects.

3.5. SYSTEM REQUIREMENTS

3.5.1 DATABASE REQUIREMENTS

- Browser for SQLite (DBMS) is a high quality, visual, open-source tool to create, design, and edit database files compatible with SQLite.
- DBMS is for users and developers who want to create, search, and edit databases.
- DBMS uses a familiar spreadsheet-like interface, and complicated SQL commands do not have to be learned.
- Controls and wizards are available for users to:
    - Create and compact database files
    - Create, define, modify, and delete tables
    - Create, define, and delete indexes o Browse, edit, add, and delete records o Search records o Import and export records as text
    - Import and export tables from/to CSV files
    - Import and export databases from/to SQL dump files
    - Issue SQL queries and inspect the results

3.5.2 SOFTWARE REQUIREMENTS

Anaconda Navigator: Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

• Operating System	:	Windows 10
• IDE	:	Spyder
• Programming Language	:	Python

3.5.3 HARDWARE REQUIREMENTS

RAM: 8 GB (As we are using Machine Learning Algorithm and Various High-Level Libraries)

- Laptop RAM: minimum 8 GB.
- Hard Disk: 40 GB.
- Processor: Intel i5 Processor (Spyder ide environment is to be used and data loading should be fast hence fast processor is required)
- IDE: Spyder
- Coding Language: Python Version 3.5 (Highly specified Programming Language for Machine Learning because of availability of High-Performance Libraries)
- Operating System: Windows 11 (Latest Operating System that supports all type of installation and development Environment)

3.6. ANALYSIS MODELS: SDLC MODEL TO BE APPLIED

SDLC Models stands for Software Development Life Cycle Models. In this article, we explore the most widely used SDLC methodologies such as Agile. Each software development life cycle model starts with the analysis, in which the Also, here are defined the technologies used in the project, team load. One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models. SDLC – is a continuous process, which starts from the moment, when it’s made a decision to launch the project, and it ends at the moment of its full remove from the exploitation. There is no one single SDLC model. They are divided into main groups, each with its features and weaknesses.

1. Requirement Analysis - Requirement Analysis is the most important and necessary stage in SDLC. The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry. Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage. Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

2. System Design - The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

3. Implementation - In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers must follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

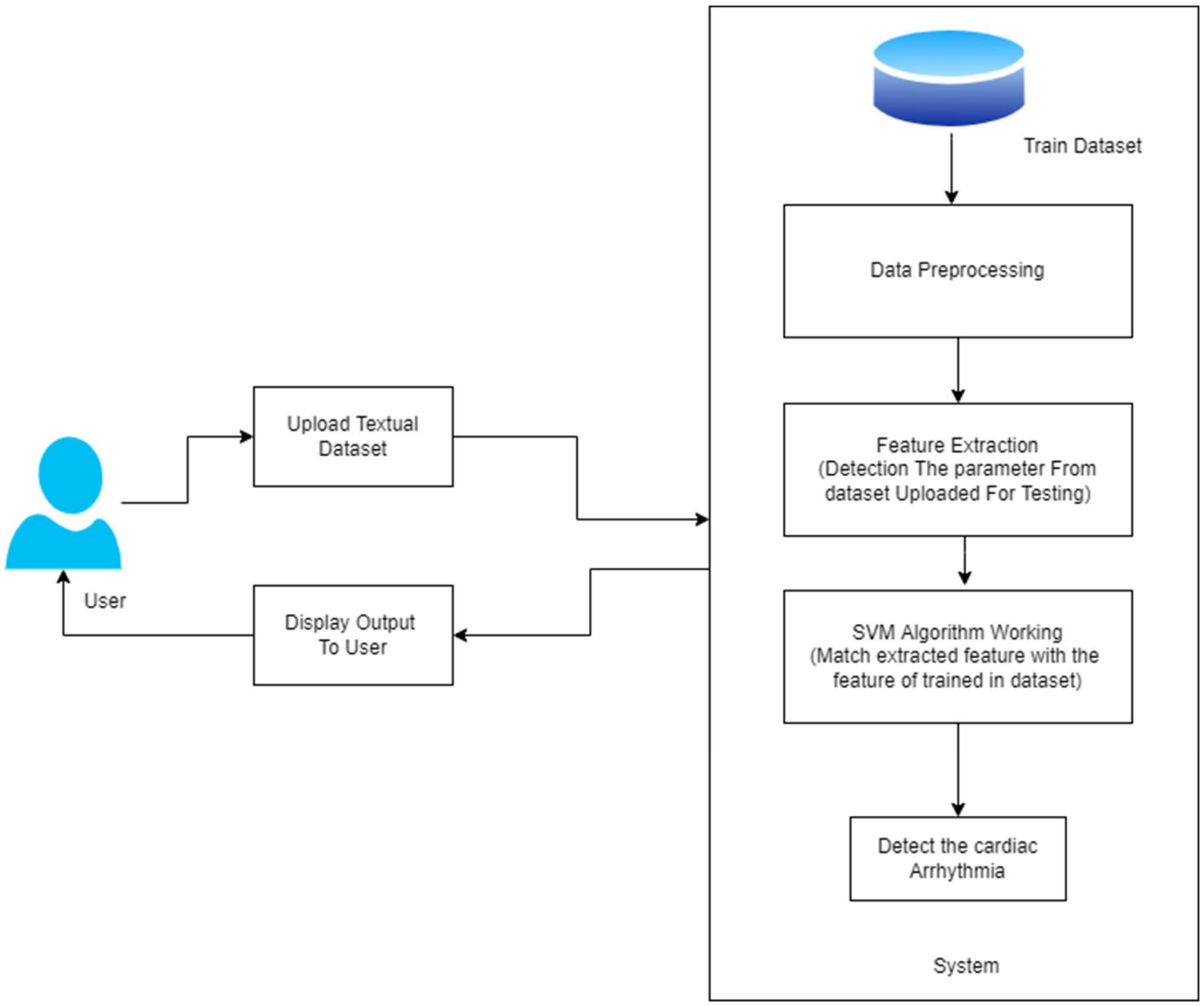
4. Testing - After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage. During this stage, unit testing, integration testing, system testing, acceptance testing is done.

5. Deployment - Once the software is certified, and no bugs or errors are stated, then it is deployed. Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment. After the software is deployed, then its maintenance begins.

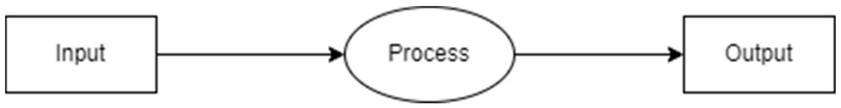
6. Maintenance - Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time. This procedure where the care is taken for the developed product is known as maintenance.

4. SYSTEM DESIGN

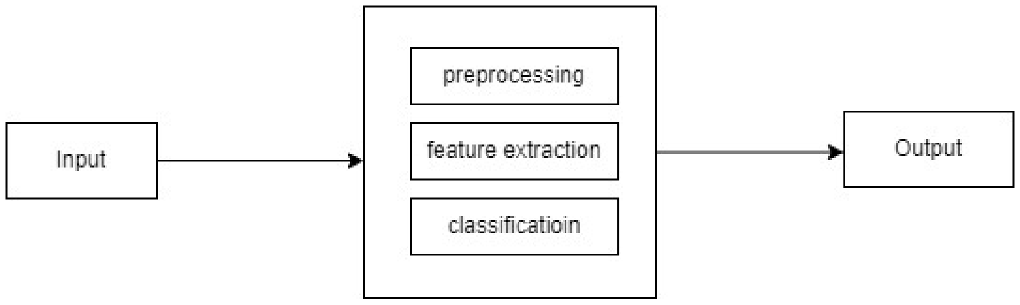
4.1 SYSTEM ARCHICTURE


Train Dataset
Data Preprocessing
Feature Extraction (Detection The parameter From dataset Uploaded For Testing)
SVM Algorithm Working (Match extracted feature with the feature of trained in dataset)
Detect the cardiac Arrhythmia
System
Upload Textual Dataset
Display Output To User
User

4.2 DATA FLOW DIAGRAM

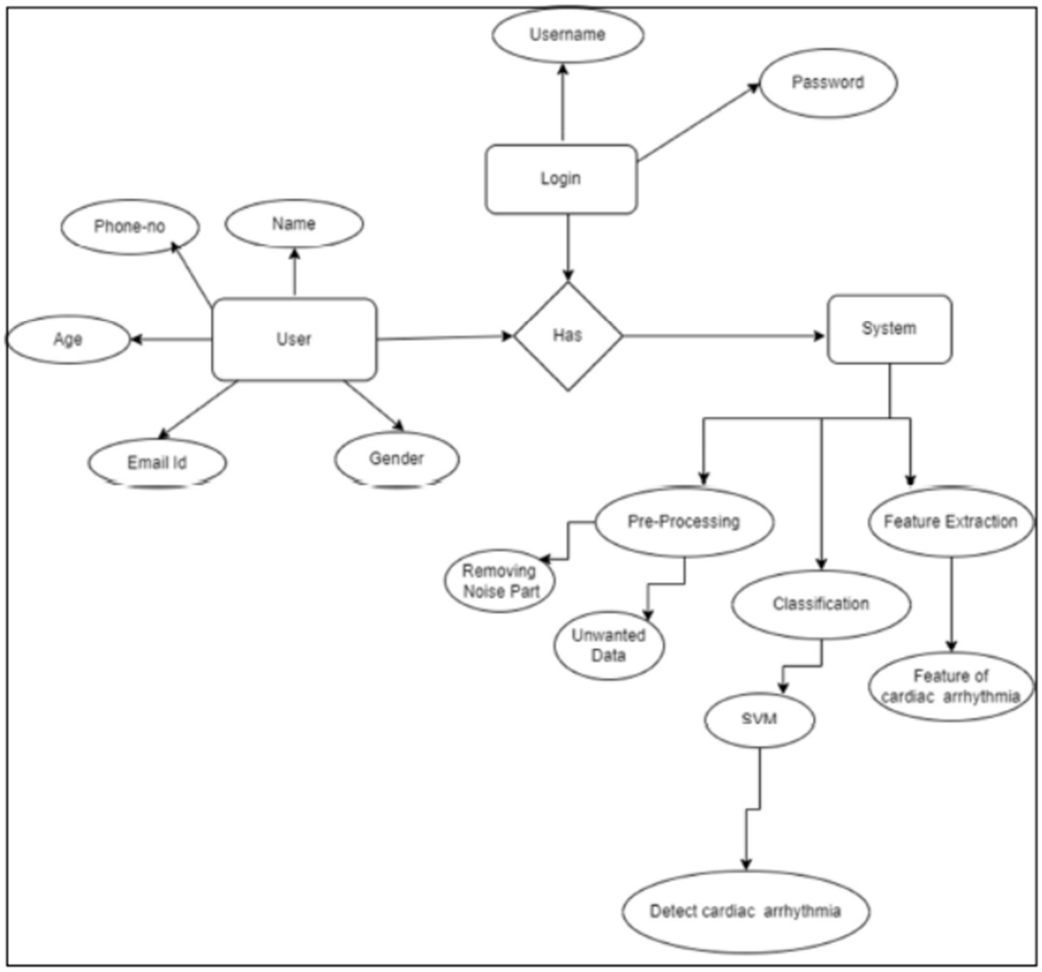


(DATA FLOW DIAGRAM 1)



(DATA FLOW DIAGRAM 2)

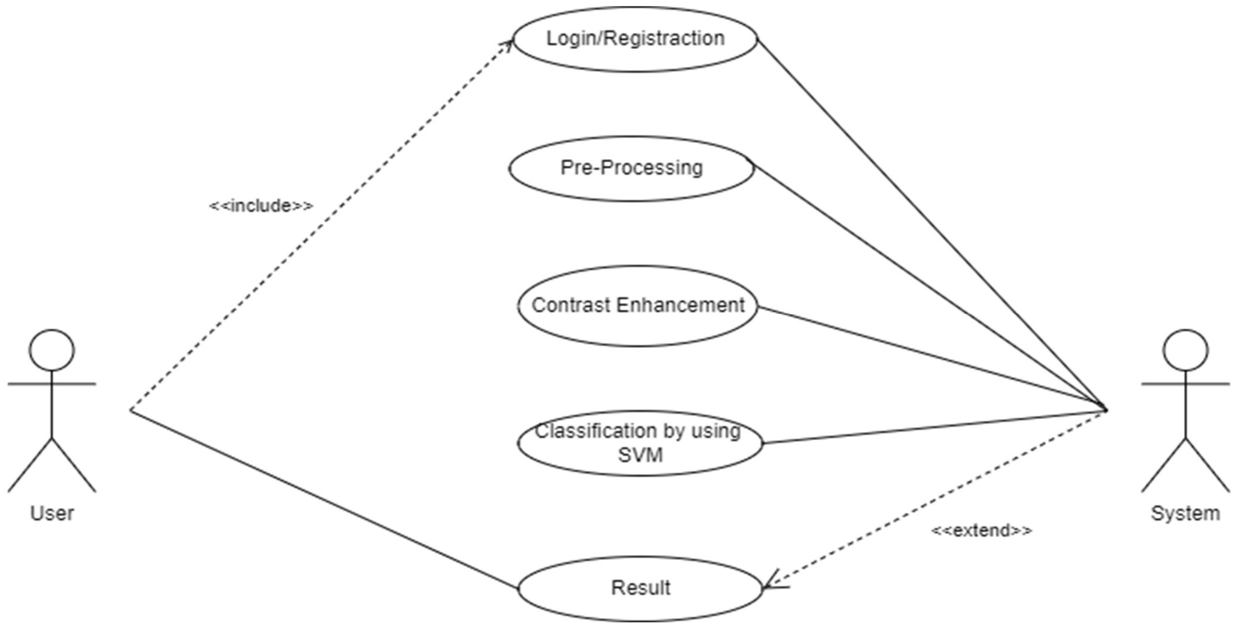
4.3 ENTITY RELATIONSHIP DIAGRAM



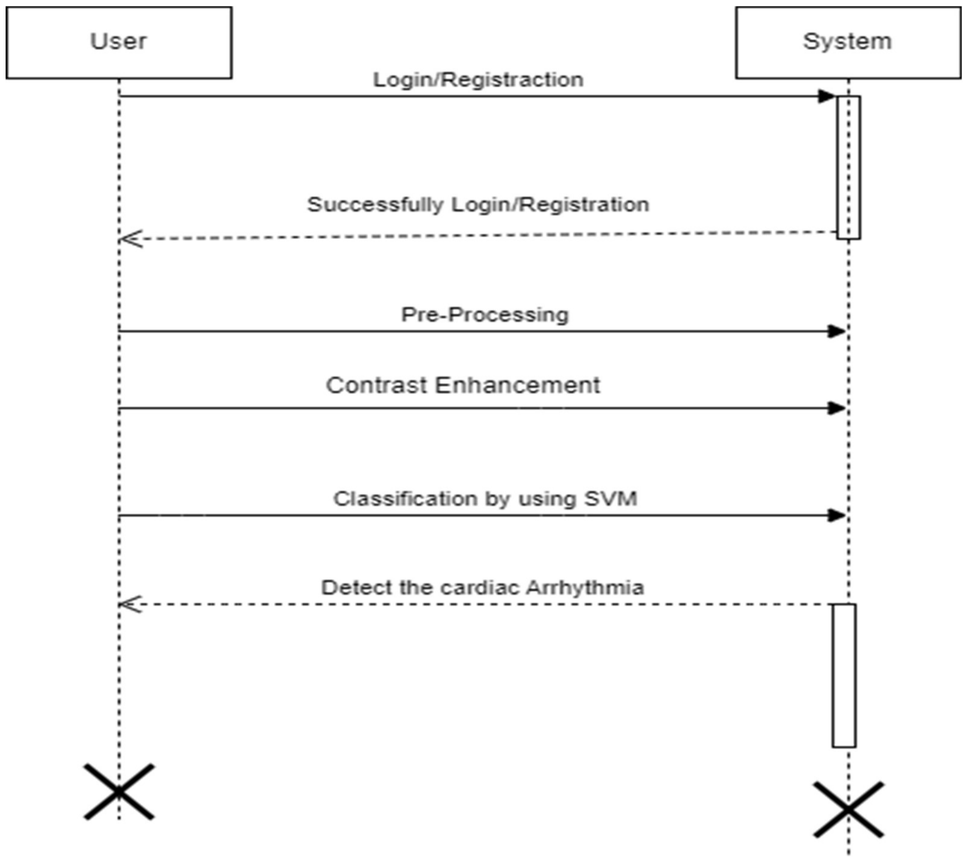
(ENTITY RELATIONSHIP DIAGRAM)

4.4. UML DIAGRAMS

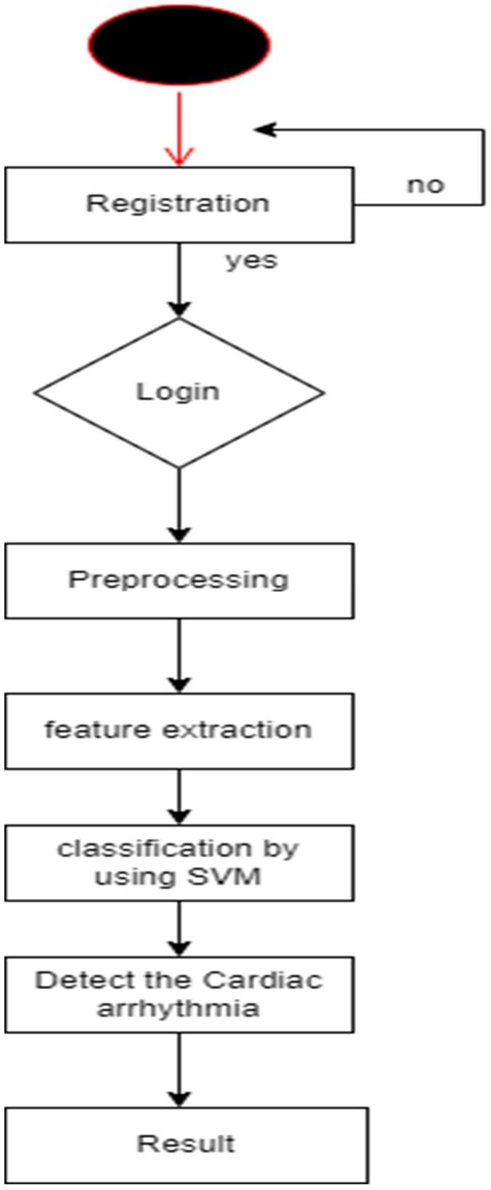
4.4.1 USE CASE DIAGRAM



4.4.2 SEQUENCE DIAGRAM


User
System
Login/Registraction
Successfully Login/Registration
Pre-Processing
Contrast Enhancement
Classification by using SVM
Detect the cardiac Arrhythmia

4.4.3 ACTIVITY DIAGRAM



5. OTHER SPECIFICATION

5.1 ADVANTAGES

1. Accurate Arrhythmia Classification: By utilizing SVM, a robust and well-established machine learning algorithm, the project can achieve accurate classification of different types of cardiac arrhythmias. SVM has been widely used for pattern recognition tasks, including medical diagnosis, and has shown promising results in arrhythmia classification.

2. Automated Diagnosis: The project provides an automated arrhythmia detection system, reducing the dependence on manual interpretation by cardiologists. This automation can enhance efficiency and speed up the diagnosis process, leading to quicker medical interventions and improved patient care.

3. Objective and Consistent Results: The use of machine learning algorithms, such as SVM, helps eliminate subjective biases that can occur in human interpretation. The project's objective approach ensures consistent and reliable results, reducing the potential for human errors in arrhythmia classification.

4. User-Friendly Interface: The integration of Tkinter GUI framework provides a user-friendly interface for interacting with the arrhythmia classification system. The GUI allows healthcare professionals to easily input ECG data, view classification results, and navigate through the application. This simplifies the user experience and enhances usability.

5. Potential for Real-Time Monitoring: The project lays the foundation for potential real-time monitoring of cardiac arrhythmias. By integrating the system with live ECG devices or telemetry systems, it can enable continuous monitoring and timely detection of arrhythmias, facilitating proactive medical interventions.

6. Basis for Further Development: The project can serve as a starting point for future enhancements and research in arrhythmia classification. It provides a foundation for incorporating advanced machine learning techniques, expanding the classification system to cover more arrhythmia types, and integrating with electronic health records or other healthcare systems.

5.2 LIMITATIONS

1. Dependency on Input Quality: The accuracy of the arrhythmia classification heavily relies on the quality and accuracy of the input electrocardiogram (ECG) data. Inaccurate or noisy input data may lead to incorrect classification results. It is crucial to ensure the reliability and accuracy of the ECG data sources used in the project.

2. Generalization to Unseen Data: The project's classification model may have limitations in generalizing well to unseen or out-of-distribution data. It is important to evaluate the performance of the model on a diverse range of data, including cases that were not part of the training set, to assess its robustness and generalization capability.

3. Human Expertise and Validation: The project should be used as a supportive tool and not as a substitute for professional medical expertise. It is crucial to involve cardiologists and domain experts throughout the development and validation process to ensure that the classification results align with clinical knowledge and standards.

5.3 APPLICATIONS

1. Clinical Decision Support System: The project can be utilized as a clinical decision support tool for healthcare professionals. It can assist cardiologists and healthcare providers in accurately classifying and diagnosing different types of cardiac arrhythmias, providing valuable insights for treatment planning and decision-making.

2. Screening and Early Detection: The project can be employed as a screening tool to identify individuals at risk of cardiac arrhythmias. By analyzing ECG signals, the system can detect abnormal patterns and alert individuals to seek further medical evaluation. Early detection can lead to timely interventions, potentially preventing adverse cardiac events.

3. Research and Data Analysis: The project can support research activities in the field of cardiac arrhythmias. Researchers can utilize the system to analyze large datasets of ECG signals, investigate patterns and correlations, and generate insights for further studies in arrhythmia management, risk assessment, and treatment optimization.

4. Clinical Trials and Outcome Analysis: The project's classification capabilities can be utilized in clinical trials related to cardiac arrhythmias. It can assist in analyzing ECG data from participants, monitoring treatment outcomes, and evaluating the effectiveness of interventions or medications.

5. Healthcare Data Analytics: By leveraging the classification results generated by the project, healthcare organizations can perform data analytics to gain insights into the prevalence, distribution, and characteristics of different types of cardiac arrhythmias. This information can help in resource planning, quality improvement initiatives, and population health management.

6. Decision Support for Medical Devices: The project's classification module can be integrated into medical devices such as ECG monitors or defibrillators. It can provide real-time feedback and alerts to healthcare providers regarding the presence of arrhythmias, supporting timely interventions and device programming decisions.

6. PROJECT IMPLEMENTATION

6.1 OVERVIEW OF PROJECT MODULES

1. Data Collection: Obtain a dataset of cardiac arrhythmia cases, which may include electrocardiogram (ECG) recordings or other relevant data.

2. Data Preprocessing: Prepare the collected data for training and classification. This step may involve cleaning the data, removing noise, and addressing missing values or outliers. Additionally, feature extraction techniques can be applied to derive meaningful features from the raw data, such as extracting heart rate variability metrics or frequency domain features.

3. Feature Selection: Choose a subset of relevant features from the preprocessed data. This step helps in reducing the dimensionality of the data and removing any redundant or irrelevant features that may hinder the classification performance.

4. Feature Scaling: Normalize or scale the selected features to ensure that they have a similar range or distribution. Common techniques include z-score normalization or min-max scaling, which help prevent certain features from dominating the SVM training process due to their larger magnitudes.

5.Training/Test Split: Split the preprocessed and scaled data into training and testing sets. The training set is used to train the SVM model, while the testing set is used to evaluate its performance.

6. SVM Model Training: Utilize the training set to train an SVM classifier. SVMs aim to find an optimal hyperplane that separates the different classes of cardiac arrhythmia in the feature space. The choice of the kernel function (e.g., linear, polynomial, radial basis function) and tuning of hyperparameters (e.g., regularization parameter C, kernel parameters) can significantly impact the model's performance.

7. Model Evaluation: Assess the performance of the trained SVM model using the testing set. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1 score, and receiver operating characteristic (ROC) curve analysis. These metrics provide insights into how well the model can classify different types of arrhythmias.

8. Model Optimization: Fine-tune the SVM model by adjusting hyperparameters or exploring different kernel functions to optimize its performance. Techniques like grid search or random search can be used to systematically search through the hyperparameter space and find the best combination.

9. Prediction: Apply the trained SVM model to classify new, unseen cases of cardiac arrhythmia. The model takes the relevant features of an input sample and predicts the corresponding arrhythmia class based on the learned decision boundary.

6.2 TOOLS AND TECHNOLOGIES USED

1. Programming Language: Python is a popular choice due to its extensive libraries for machine learning and data analysis.
2. Machine Learning Libraries: Various libraries in Python provide SVM implementations and other relevant functions. The most commonly used library for SVM is scikit-learn, which offers robust SVM models with different kernel functions and hyperparameter tuning options.
3. Data Analysis and Manipulation: Libraries such as NumPy and Pandas are commonly used for data manipulation, preprocessing, and feature engineering. They provide efficient data structures and functions to handle numerical computations and data handling tasks.
4. Data Visualization: Matplotlib and Seaborn are widely used libraries for data visualization in Python. They help create plots, charts, and graphs to visualize the data distribution, feature correlations, and classification results.
5. Feature Selection and Dimensionality Reduction: Scikit-learn provides various methods for feature selection and dimensionality reduction, such as Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA).
6. Cross-Validation and Model Evaluation: Scikit-learn includes modules for cross-validation, evaluation metrics, and model selection. Functions like cross_val_score, GridSearchCV, and classification_report can be utilized to perform cross-validation, hyperparameter tuning, and evaluate the model's performance.
7. Model Deployment: Tkinter, popular Python web frameworks, can be used to deploy the trained SVM model as a web application or API, allowing users to make predictions on new data.
8. Integrated Development Environment (IDE): IDEs such as PyCharm, Jupyter Notebook, or Visual Studio Code provide a development environment with code editing, debugging, and interactive execution capabilities, making the development process more efficient.
9. Data Storage: Depending on the size and nature of the dataset, various data storage solutions can be employed, including CSV files, databases (e.g., SQLite, MySQL, or PostgreSQL), or cloud-based storage services like Amazon S3 or Google Cloud Storage.

6.3 ALGORITHM

1. Import the necessary libraries for data preprocessing, SVM classification, and Tkinter GUI development.
2. Load the dataset containing cardiac arrhythmia data.
3. Perform data preprocessing tasks such as cleaning, filtering, and feature extraction on the dataset.
4. Split the preprocessed dataset into training and testing sets.
5. Implement the SVM model using a suitable kernel function (e.g., linear, polynomial, radial basis function) and train it using the training set.
6. Create a Tkinter GUI window with appropriate widgets for user interaction.
7. Design and place input fields and options in the GUI to allow users to enter or select relevant parameters for classification (e.g., heart rate, intervals, or ECG waveforms).
8. Create a button widget in the GUI to trigger the classification process.
9. Define a function that retrieves the user input from the GUI, preprocesses it according to the training data preprocessing steps, and feeds it to the trained SVM model for prediction.
10. Display the predicted arrhythmia class in the GUI, either through a label or a message box.
11. Test the GUI by running the application and verifying that the classification process and result display are functioning correctly.
12. Refine and enhance the GUI design and functionality based on user feedback and testing.
13. Optionally, implement additional features such as visualizations or data summaries to provide more insights into the classification results.
14. Iterate on the design and implementation to improve usability, accuracy, and user experience.
15. Deploy the final GUI application for users to classify cardiac arrhythmia cases using the SVM model.

7. SOFTWARE TESTING

7.1. TYPE OF TESTING

7.1.1. Unit testing:

Unit testing for this project involves testing individual components and functionalities of the System to ensure their proper functionality. Here are some unit testing scenarios that can be performed:

- Input Validation: Test the input validation mechanism of the GUI to ensure that only valid and appropriate data can be entered. This includes testing for correct handling of empty fields, invalid characters, and input range validation.
- SVM Classification: Test the SVM classification module by providing known input data and verifying that the expected output (arrhythmia type) matches the actual output. This includes testing for various arrhythmia types and representative input samples for each type.
- GUI Functionality: Test the functionality of the Tkinter GUI, including buttons, menus, and data entry fields. Verify that the GUI elements are responsive, and interactions with the GUI do not result in unexpected errors or crashes.
- Error Handling: Test the system's error handling mechanisms by intentionally providing invalid or unexpected input data. Verify that appropriate error messages are displayed, and the system gracefully handles and recovers from errors.
- Performance Testing: Evaluate the performance of the system by testing its response time and resource utilization. This includes measuring the time taken for classification and GUI responsiveness under different workloads.

7.1.2 White Box Testing

White box testing takes advantage of extensive knowledge of an application's internals to develop highly-targeted test cases. Examples of tests that might be performed during white box testing include:

- Path Checking: White box testing can be used to explore the various execution paths within an application to ensure that all conditional statements are correct, necessary, and efficient.
- Output Validation: This enumerates the various potential inputs to a function and ensures that each produces the expected result.
- Security Testing: Static code analysis and other white box testing techniques are used to identify potential vulnerabilities within an application and validate that it follows secure development best practices.
- Loop Testing: Tests the loops within an application to ensure that they are correct, efficient, and properly manage the variables within their scope.
- Data Flow Testing: Tracks variables throughout the execution paths of a program to ensure that variables are declared, initialized, used, and properly manipulated.

7.1.3 Black Box Testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.
In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not.
If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

7.2 TEST CASE & RESULT

Test Case	**User registration**
Objective	**User should register**
Expected Result	**User registered successfully**

Table 7.2.1 Test case 1

Test Case	**User Login**
Objective	**User should Login**
Expected Result	**User Logged-in successfully**

Table 7.2.2 Test case 2

Test Case	**Submit Details**
Objective	**User should Submit details**
Expected Result	**Details Submitted successfully**

Table 7.2.3 Test case 3

Test Case	**Final Result**
Objective	**User should obtain final result**
Expected Result	**User received result successfully**

Table 7.2.4 Test case 4

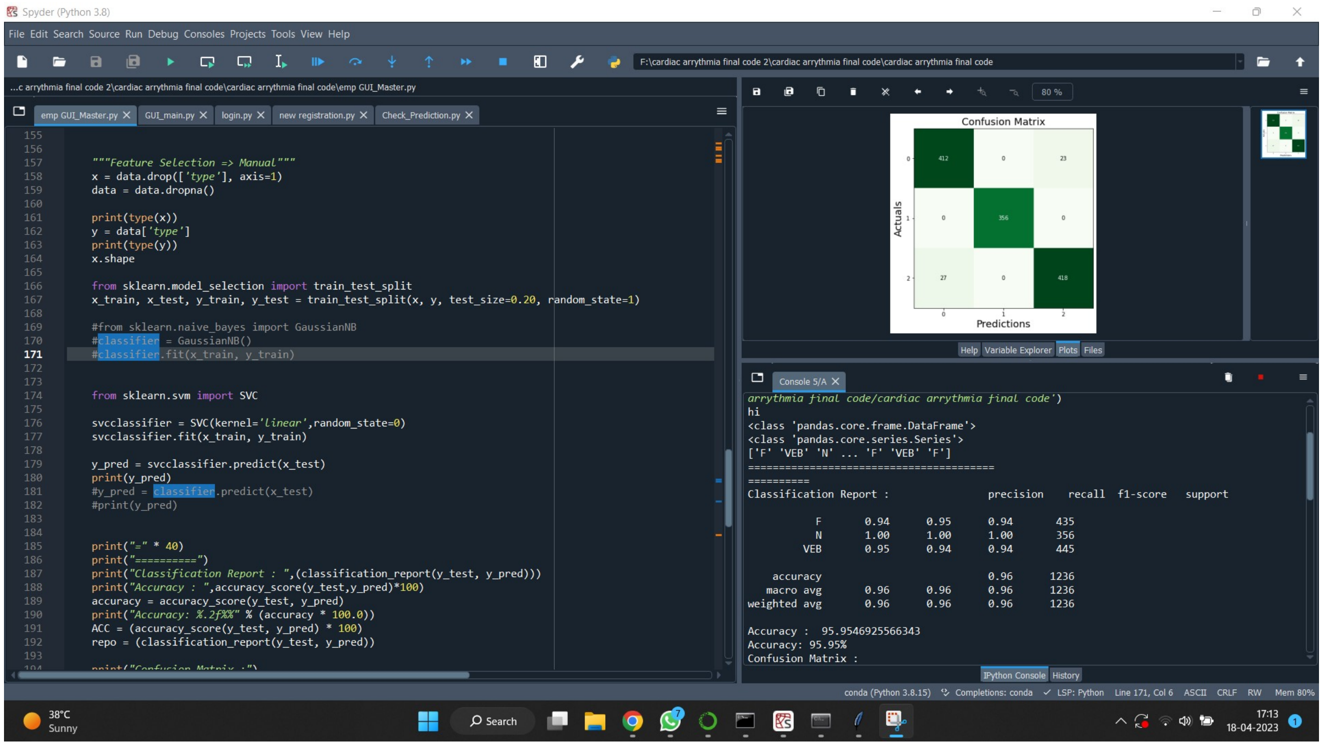
Test Case: SVM Classification Accuracy

Test Description:
This test verifies the accuracy of the SVM classification module by providing known input data and comparing the expected output with the actual output.

Test Steps:
Prepare a set of representative input data samples for different types of cardiac arrhythmias, including atrial fibrillation, ventricular tachycardia, and bradycardia.
Load the trained SVM model used for classification.
For each input data sample:
a. Input the sample into the SVM classification module.
b. Obtain the actual output (arrhythmia type) from the classification module.
c. Compare the actual output with the expected output for that specific input sample.
d. Repeat the above steps for all representative input samples.

Test Assertions:
The actual output should match the expected output for each input data sample.
The SVM classification module should achieve a high accuracy in classifying different types of arrhythmias.
Test Pass Criteria:
All actual outputs should match the expected outputs for the provided input data samples, resulting in a 100% accuracy rate.

Test Fail Criteria:
If any actual output does not match the expected output for the provided input data samples, the test will fail, indicating a potential issue with the SVM classification module.



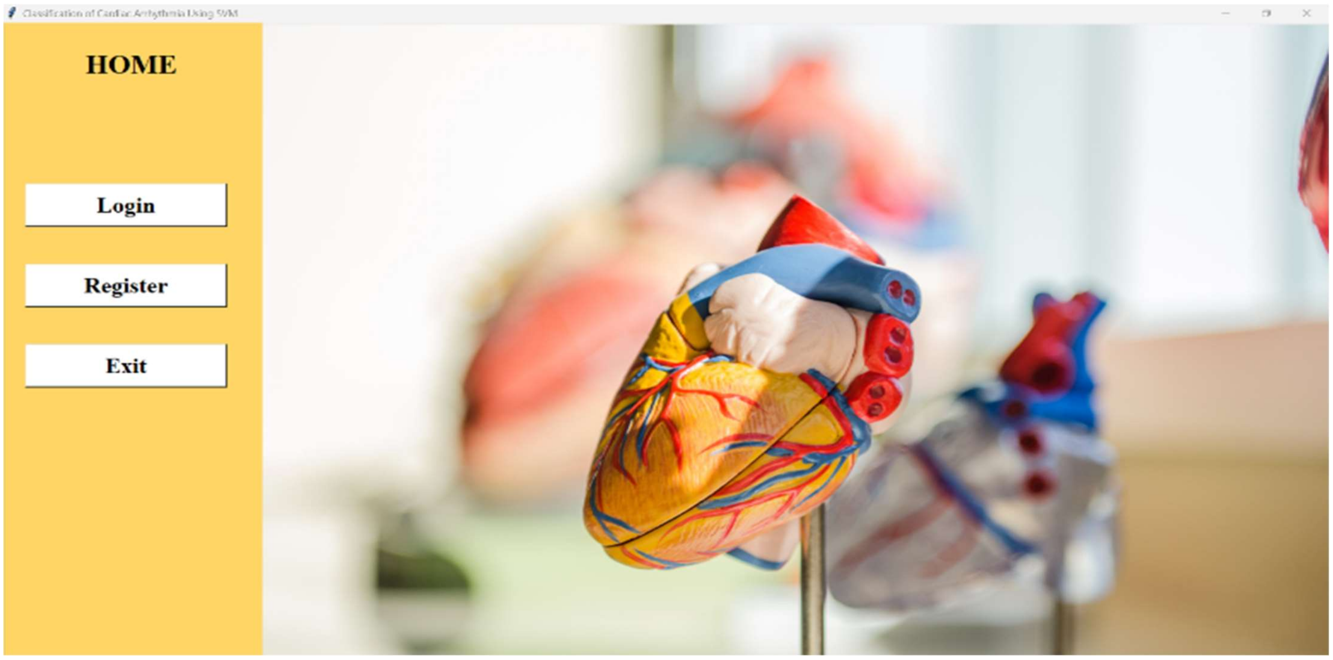
```python


    """Feature Selection => Manual"""
    x = data.drop(['type'], axis=1)
    data = data.dropna()

    print(type(x))
    y = data['type']
    print(type(y))
    x.shape

    from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=1)

    #from sklearn.naive_bayes import GaussianNB
    #classifier = GaussianNB()
    #classifier.fit(x_train, y_train)


    from sklearn.svm import SVC

    svcclassifier = SVC(kernel='linear',random_state=0)
    svcclassifier.fit(x_train, y_train)

    y_pred = svcclassifier.predict(x_test)
    print(y_pred)
    #y_pred = classifier.predict(x_test)
    #print(y_pred)


    print("=" * 40)
    print("==========")
    print("Classification Report : ",(classification_report(y_test, y_pred)))
    print("Accuracy : ",accuracy_score(y_test,y_pred)*100)
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy: %.2f%%" % (accuracy * 100.0))
    ACC = (accuracy_score(y_test, y_pred) * 100)
    repo = (classification_report(y_test, y_pred))
```

```
arrythmia final code/cardiac arrythmia final code')
hi
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
['F' 'VEB' 'N' ... 'F' 'VEB' 'F']
========================================
==========
Classification Report :               precision    recall  f1-score   support

           F       0.94      0.95      0.94       435
           N       1.00      1.00      1.00       356
         VEB       0.95      0.94      0.94       445

    accuracy                           0.96      1236
   macro avg       0.96      0.96      0.96      1236
weighted avg       0.96      0.96      0.96      1236

Accuracy :  95.9546925566343
Accuracy: 95.95%
Confusion Matrix :
```

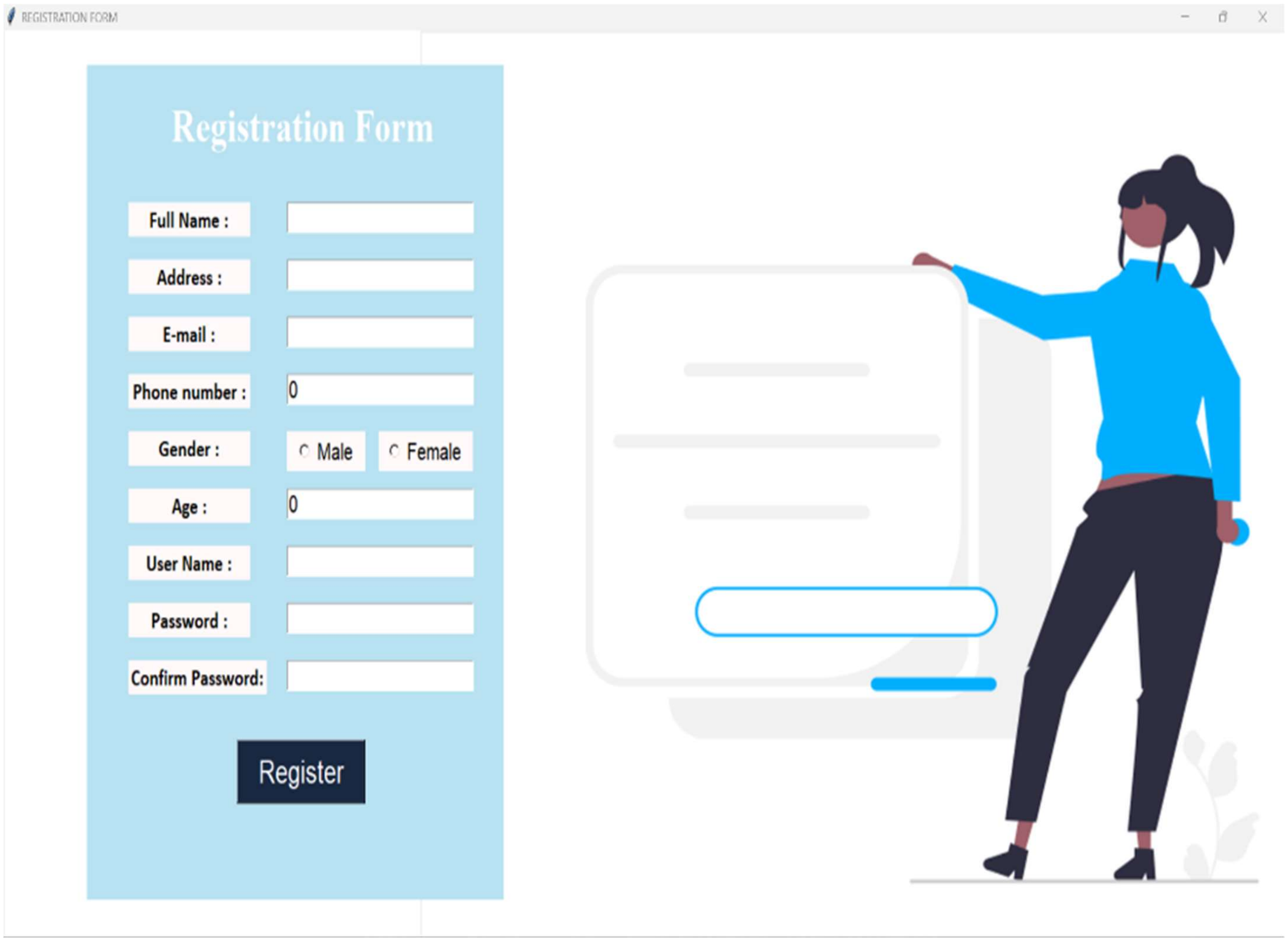
(TEST RESULT)

8. RESULT:

8.1 OUTCOMES:



(LOGIN/REGISTRATION PAGE)


REGISTRATION FORM

Registration Form

Full Name :

Address :

E-mail :

Phone number : 0

Gender : Male Female

Age : 0

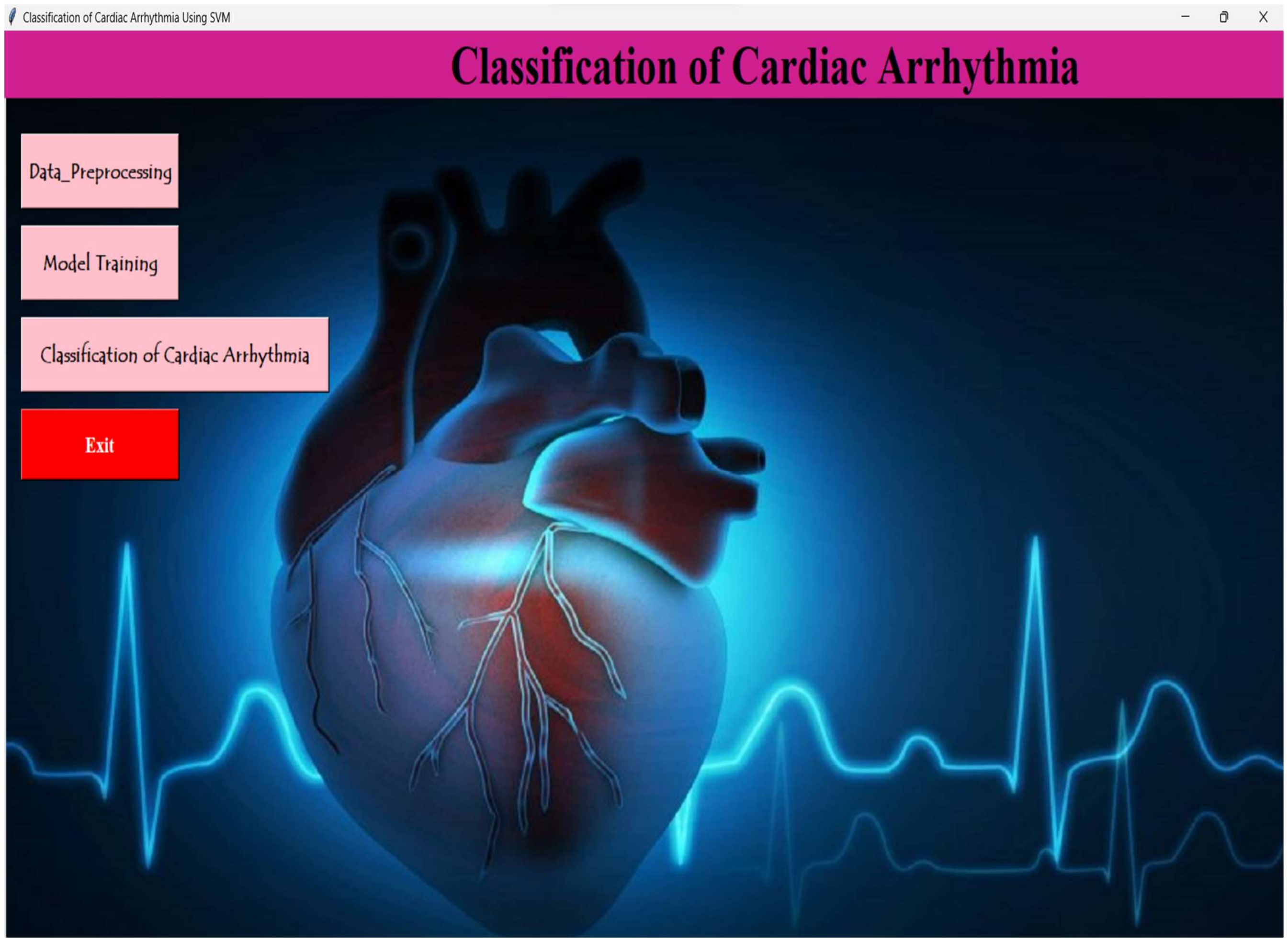
User Name :

Password :

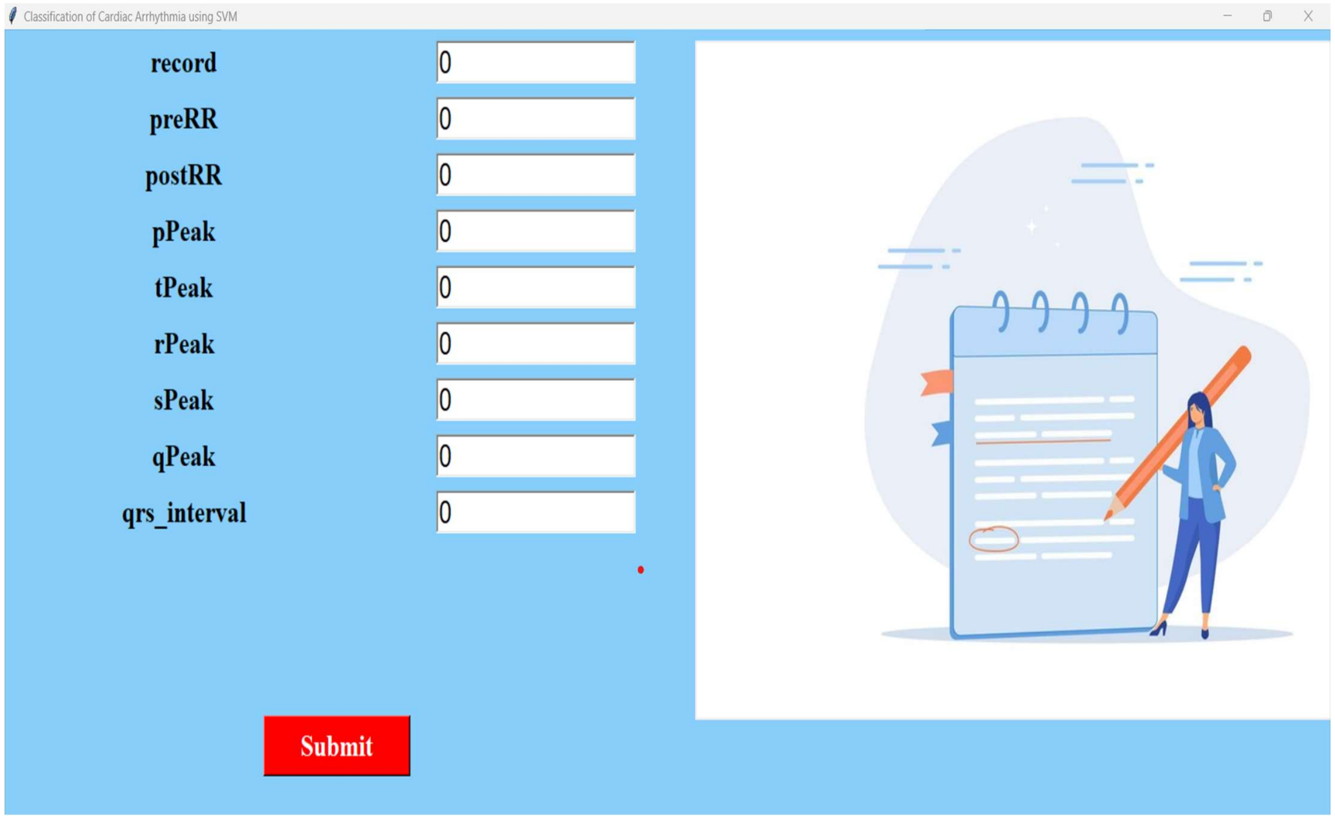
Confirm Password:

Register

(REGISTRATION PAGE)


Classification of Cardiac Arrhythmia Using SVM
Classification of Cardiac Arrhythmia
Data_Preprocessing
Model Training
Classification of Cardiac Arrhythmia
Exit

(HOME PAGE)



record	0
preRR	0
postRR	0
pPeak	0
tPeak	0
rPeak	0
sPeak	0
qPeak	0
qrs_interval	0

(DATA SUBMISSION PAGE)

9. CONCLUSIONS

9.1 CONCLUSION

In conclusion, the project "Classification of Cardiac Arrhythmia using SVM with Tkinter GUI" aims to develop an accurate and user-friendly system for classifying different types of cardiac arrhythmias. By integrating a Support Vector Machine (SVM) algorithm with a Tkinter graphical user interface (GUI), the project addresses the need for efficient arrhythmia diagnosis and classification.

The project's objectives include improving the accuracy of arrhythmia classification, enhancing clinical workflow, and providing decision support to healthcare professionals. By leveraging SVM, a powerful machine learning algorithm, the system aims to achieve high accuracy in identifying and differentiating between various arrhythmia types.

9.2 FUTURE WORK

The current project focuses on classifying a specific set of arrhythmia types. Future work can involve expanding the classification system to cover a broader range of arrhythmias, including rare or atypical ones.
Extending the project to incorporate real-time ECG signal acquisition and analysis would enable continuous
monitoring of arrhythmias. This could involve connecting the system to live ECG devices or integrating it
with a telemetry system for remote monitoring. Providing real-time feedback and alerts to healthcare professionals would aid in timely interventions.
The proposed model can be useful for long-term ECG monitoring using single-lead wearable devices in clinical settings in the future.

APPENDIX A

References:

[1] B. de Luna, P. Coumel, and J. F. Leclercq, “Ambulatory sudden cardiacdeath: Mechanisms of production of fatal arrhythmia on the basis of data from 157 cases,” Amer. Heart J., vol. 117, pp. 151–159, 1989.

[2] J. F. Leclercq, P. Coumel, P. Maison-Blanche, B. Cauchemez, M. Zimmer-mann, F.Chouty, and R. Slama, “Mise en evidence des mecanismes d ́ eterminants de la mort subite,” Arch Mal Coeur, vol. 79, pp. 1024–1033, 1986.

[3] H. A. Fozzard, “Electromechanical dissociation and its possible role in sudden cardiac death,” J. Amer. Coll. Cardiol., vol. 5, pp. 31B–34B, 1985.

[4] K. Dawson, A. S. Leon, and H. L. Taylor, “Effect of pentobarbital anesthesia on vulnerability to ventricular fibrillation,” Amer. J. Physiol., vol. 239, pp.H427–H431, 1980.

[5] T. L. Wenger, F. E. Harrell Jr., K. K. Brown, S. Lederman, and H. C. Strauss, “Ventricular fibrillation following canine coronary artery reperfusion: Different outcomes with pentobarbital and alpha-chloralose,” Can. J. Physiol. Phar-macol., vol. 62, pp. 224–228, 1984.

[6] F. H. De Jonge, E. A. Bokkers, W. G. Schouten, and F. A. Helmond, “Rearing piglets in a poor environment: Developmental aspects of social stress in pigs,” Physiol. Behavior, vol. 60, pp. 389–396, 1996.

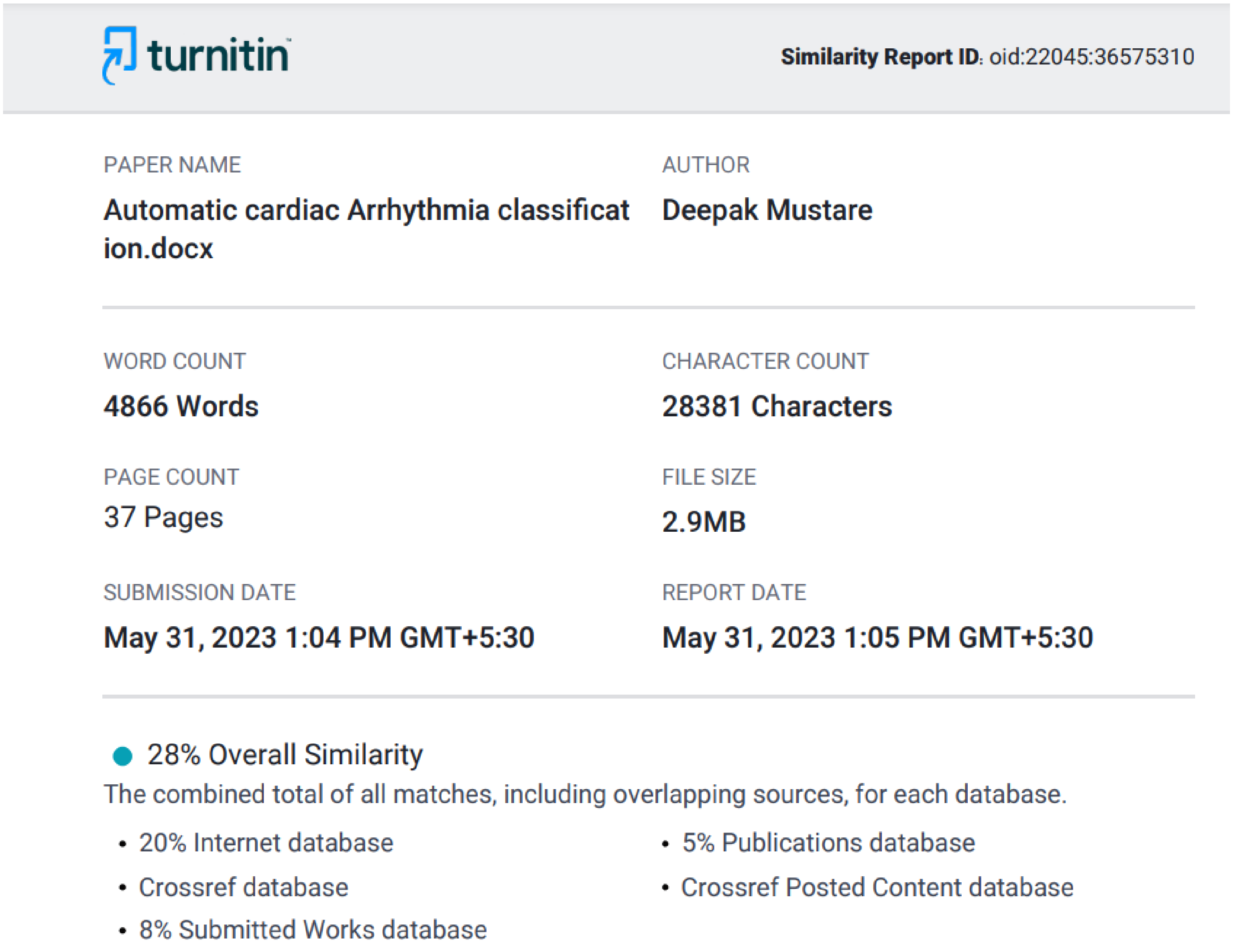
[7] W. G. Schouten and V. M. Wiegant, “Individual responses to acute and chronic stress in pigs,” Acta Physiologica Scandinavica. Supplementum, vol. 640, pp.88–91, 1997.

[8] M. Roelke, H. Garan, B. A. McGovern, and J. N. Ruskin, “Analysis of the initiation of spontaneous monomorphic ventricular tachycardia by stored in- tracardiac electrograms,” J. Amer. Coll. Cardiol., vol. 23, pp. 117–122, 1994.

[9] S. B. Knisley and W. M. Smith, “Experimental techniques for investigating cardiac electrical activity and response to electrical stimuli,”Proc. IEEE,
vol.84, pp. 417–427, Mar. 19

APPENDIX C

PLAGIARISM REPORT



turnitin

Similarity Report ID: oid:22045:36575310

PAPER NAME	AUTHOR
Automatic cardiac Arrhythmia classificat ion.docx	Deepak Mustare
WORD COUNT	CHARACTER COUNT
4866 Words	28381 Characters
PAGE COUNT	FILE SIZE
37 Pages	2.9MB
SUBMISSION DATE	REPORT DATE
May 31, 2023 1:04 PM GMT+5:30	May 31, 2023 1:05 PM GMT+5:30

● 28% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 20% Internet database
- Crossref database
- 8% Submitted Works database
- 5% Publications database
- Crossref Posted Content database