DISEASE DETECTION SYSTEM

A PROJECT REPORT

Submitted by

Group No. 174
Shivam Krishna (23BCE10589)
Kanishk Tomar (23BCE10588)
Ankit Kumar (23BCE10519)
Myisha Porwal (23BCE11519)
Stuti S. Agrawal (23BCE11717)

in partial fulfilment for the award of the degree
of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE

MADHYA PRADESH - 466114

December 2024

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"DISEASE DETECTION SYSTEM"** is the bonafide work of "**Shivam Krishna (23BCE10589) Kanishk Tomar (23BCE10588) ,Ankit Kumar (23BCE10519) Myisha Porwal (23BCE11519) ,Stuti S. Agrawal (23BCE11717)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR	**PROJECT GUIDE**
Dr. Vikas Panthi	Dr. Ajeet Singh
Program Chair, CSE-Core	Assistant Professor
School of Computer Science	School of Computer Science
And Engineering	and Engineering
VIT BHOPAL UNIVERSITY	VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on _______________

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to **Dr. Vikas Panthi**, Head of the Department, School of Computing Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide **Mr. Ajeet Singh**, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

ABSTRACT

The Disease Prediction System aims to provide an efficient and accurate solution for predicting potential diseases based on user-provided symptoms and medical data. The purpose of this project is to assist healthcare professionals and individuals in the early identification of diseases, thereby facilitating timely intervention and treatment. By leveraging machine learning techniques, the system enhances diagnostic accuracy and improves decision-making in healthcare.

The methodology involves the use of supervised machine learning algorithms, including Naïve Bayes, and Support Vector Machines (SVM). A comprehensive dataset consisting of patient symptoms and corresponding disease labels has been used to train and validate the models. The system collects user inputs such as symptoms and relevant medical parameters, processes the data, and predicts possible diseases associated with severity levels. The project also incorporates a user-friendly interface to ensure accessibility and usability for non-technical users.

The findings demonstrate that the implemented machine learning models achieved a high level of accuracy in disease prediction. The system successfully predicts common diseases based on limited inputs, providing users with reliable diagnostic suggestions. This solution has the potential to support healthcare

delivery in remote areas, improve early detection, and reduce the workload on medical professionals.

In conclusion, our project is effective in decision-support tool that integrates machine learning to predict diseases accurately. Future enhancements may include real-time data integration, advanced deep learning for medical imaging, and wearable device compatibility to further improve healthcare outcomes.

LIST OF ABBREVIATIONS

NLP: Natural Language Processing

GUI: Graphical User Interface

ML: Machine Learning

SQL: Structured Query Language

API: Application Programming Interface

PyQt5: Python GUI Framework

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

TABLE OF CONTENTS

Chapter-1:
Project description and outline

1.1 Introduction

In recent years, healthcare systems have faced numerous challenges, including increasing disease prevalence, limited accessibility to medical resources, and delays in diagnosis. Early detection and timely intervention play a critical role in preventing disease progression and reducing healthcare costs. However, many individuals either lack access to healthcare facilities or delay seeking medical attention due to factors such as cost, geographic constraints, or lack of awareness.

The Disease Detecting System is designed to predict potential diseases based on user-provided symptoms. This project aims to assist users in identifying underlying health issues by leveraging advanced technologies like machine learning. With features like a user-friendly interface, voice and text-based symptom input, and visually appealing results, this project sets a new benchmark in healthcare accessibility and efficiency.

1.2 Motivation for the Work

Healthcare is one of the most critical domains where technological advancements can significantly impact human lives. The motivation for developing the **Disease Prediction System** stems from the need to address several persistent challenges in healthcare and improve the diagnostic process for users and healthcare providers.

Accessibility to Healthcare, Integration of AI in Healthcare, Bridging the Gap Between Technology and Healthcare

The motivation also stems from the aspiration to bridge the gap between cutting-edge technology and real-world healthcare challenges. By combining machine learning, voice recognition, and intuitive interfaces, the system showcases how technology can simplify and enhance diagnostic processes., Education and Awareness, Scalability for Future Applications

1.3 Introduction to the Project

The Disease Detection System is designed as an intelligent, user-centric solution for preliminary diagnosis. This project focuses on creating a system that collects and analyses symptoms input by users to suggest potential diseases. Using a robust database of symptom-disease mappings and machine learning models, the system offers accurate predictions while ensuring ease of use.

The system includes an interactive interface where users can input their symptoms. Advanced algorithms then process this information and generate a ranked list of possible diagnoses, along with confidence levels or recommendations for further medical action. The system is designed to be scalable, integrating real-time updates from medical databases and incorporating feedback from healthcare professionals to improve accuracy and adaptability.

Key features of the system include ease of access, multi-language support, and compatibility with telemedicine platforms. By offering an accessible and effective diagnostic tool, the project aims to empower individuals to take proactive steps toward managing their health.

1.4 Problem Statement

Access to timely and accurate diagnosis is a significant challenge in the healthcare domain. Many people, especially in rural or underserved areas, face barriers such as limited healthcare infrastructure, high costs of medical consultations, and lack of awareness about early symptoms. Additionally, healthcare systems are often overburdened, leading to delays in consultations and diagnostic procedures.

These challenges often result in undiagnosed or misdiagnosed conditions, which can progress into more severe stages. Furthermore, the reliance on traditional diagnostic processes requires significant time and resources, making it difficult for healthcare providers to address the needs of a growing population.

To address these challenges, there is a need for an intelligent, automated system capable of analysing symptoms and providing preliminary diagnoses. Such a system would act as a first point of contact for individuals, enabling them to understand potential health risks and seek appropriate medical attention without delays.

1.5 Objective of the Work

- **Enhance accessibility:** Provide an easy-to-use diagnostic tool for individuals in remote or underserved areas.
- **Support healthcare systems:** Act as a supplementary diagnostic tool, reducing the workload on healthcare professionals.
- **Ensure scalability:** Incorporate features such as real-time updates, multi-language support, and compatibility with telemedicine platforms.

1.6 Organization of the Project

This document is structured to provide a clear understanding of the project's scope, methodology, and outcomes:

Phase 1: Data Collection and Pre-processing:

• **Data Gathering**: Gather a diverse and representative dataset of medical images containing both benign and malignant cells for training and testing purposes.

• **Data Pre-processing**: Implement data pre-processing techniques, including, normalization, and feature extraction, to prepare the dataset for model training.

Phase 2: Model Development and Training:

• Algorithm Selection: Evaluate various machine learning algorithms, such as decision trees, support vector machines, and neural networks, to determine the most suitable for cancer cell classification.

• Model Training: Train the selected machine learning model using the preprocessed dataset, optimizing hyperparameters to achieve high accuracy and reliability.

• Validation: Employ cross-validation techniques to assess model performance and ensure generalizability.

Phase 3: Real-Time Classification Engine:

• Deployment: Deploy the trained machine learning model as a real time classification engine capable of receiving medical images for on the-spot diagnosis.

• Integration: Integrate the engine with a user-friendly interface for easy interaction and accessibility by healthcare professionals. The real-time system for efficiency, accuracy, and robustness.

Phase 4: Scalability and Expansion :

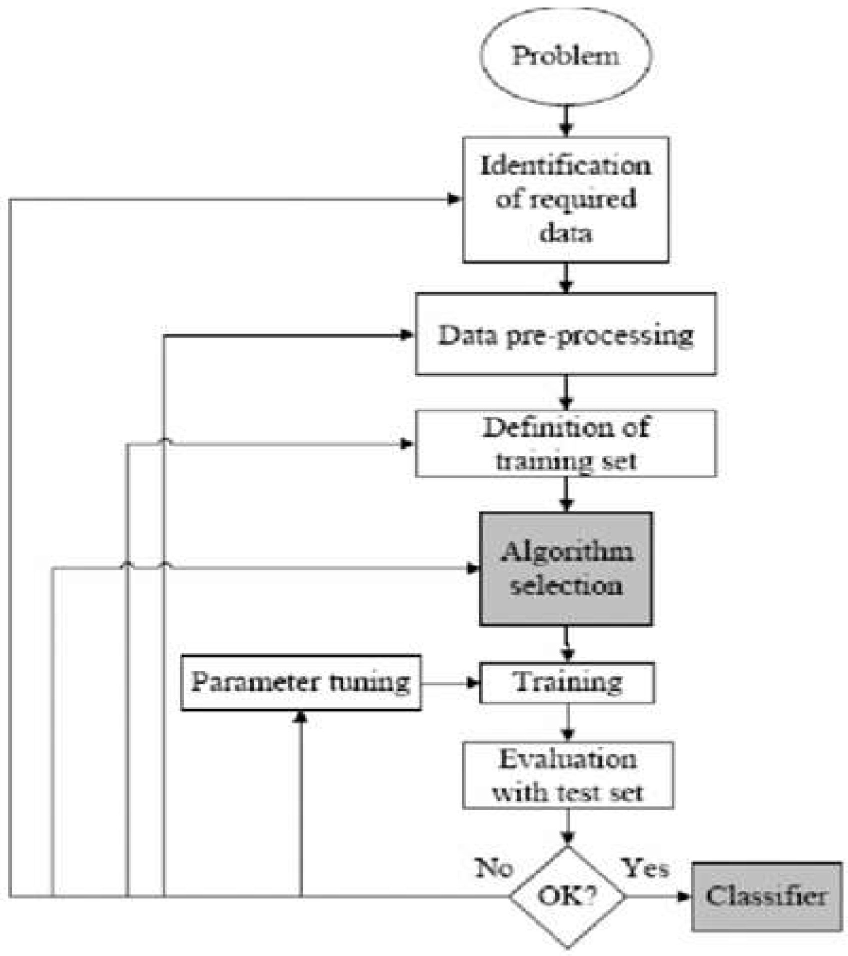
• Scalability: Explore strategies for scaling the system to handle a growing volume of data and users while maintaining performance.

• Enhancements: Continuously improve the system by incorporating user feedback and staying updated with advances in machine learning and healthcare technologies.

Phase 5: Reporting and Analytics :

• **Reporting Tools:** Develop reporting and analytics features to enable healthcare professionals to track trends, generate insights, and make data-driven decisions.

• Feedback Loop: Establish a feedback loop to gather user insights and iteratively improve the system's functionality and user experience.



FLOWCHART DESCRIBING ORGANIZATION OF PROJECT

System Design Diagram

Below is the **step-by-step workflow in diagram format**:

1. **Login System**:
    - User → Login/Registration → Database → Redirect to Prediction Page.
2. **Symptom Input**:
    - Text/Voice Input → Pre-processing (NLP) → Feature Vector Creation (CountVectorizer).

3. **Prediction**:
    - Feature Vector → ML Model → Predicted Disease.
4. **Result Display**:
    - Predicted Disease → Fetch Recommendations → Display on GUI.
5. **History Tracking**:
    - Store Predicted Disease + Symptoms in Database → View History.
6. **Exit**:
    - Logout or Close Application.

1.7. Summary

The Disease Prediction System is a user-centric application designed to predict diseases based on user-provided symptoms using advanced machine learning techniques. The project aims to bridge the gap in healthcare diagnostics by offering a scalable, accurate, and accessible solution that assists users in identifying potential diseases early, improving healthcare outcomes.

The project focuses on addressing critical challenges in the healthcare domain, such as diagnostic delays, unequal access to care, and overburdened healthcare systems. Through its user-friendly interface and scalable design, the system ensures accessibility for diverse user groups. The ultimate goal is to empower individuals to take proactive steps toward managing their health while supporting healthcare providers in delivering more efficient and effective care.

CHAPTER-2:
RELATED WORK INVESTIGATION

2.1 Introduction

Our project focuses on developing an intuitive and user-friendly interface that helps users identify possible diseases based on their symptoms. The interface allows users to input multiple symptoms and, based on the data stored in a backend database, provides a list of potential diseases and relevant health recommendations. This system aims to empower users by offering preliminary insights into their health conditions.

In any research or project, a comprehensive understanding of the existing body of knowledge is essential to build upon prior work, identify gaps, and make valuable contributions to the field. The "Related Work Investigation" section serves as a crucial component of the project, offering a thorough examination of the relevant literature, studies, and projects that inform the current undertaking. This section is not merely a review of past research, but a meticulous investigation aimed at revealing key insights, methodologies, and findings of earlier works that have a direct bearing on the project at hand. By delving into the related work, we can identify the state of the art, assess the evolution of the field, and uncover the challenges and opportunities that have shaped the research landscape.

2.2 Core of the Project (Machine Learning integrated with NLP)

This core can be broken down into several key components:

• **Machine Learning Model:** At the heart of the project is the machine learning model, which is developed using the Scikit-Learn framework. This model is designed to accurately classify diseases. It represents the foundation of the automated classification system.

• **Early Detection:** The core objective of the project is to enable early detection of disease . Early detection is critical for improving patient outcomes and survival rates. Achieving this objective is a fundamental element of the project's mission.

• **Efficiency:** The project emphasizes the need to streamline the diagnostic process. By automating the classification of various types of diseases it reduces the time required for diagnosis. This increased efficiency is pivotal in ensuring timely treatment decisions.

• **Accuracy:** Achieving a high level of accuracy and precision in disease classification is at the core of the project's success. The machine learning model must surpass the capabilities of existing diagnostic methods to provide reliable and trustworthy results.

2.3 Existing Approaches/Methods

2.3.1 Approach -1- Using Tkinter {MANUAL DIAGNOSIS}

- Tkinter is a standard Python library used to create graphical user interfaces (GUIs). It allows developers to build simple, interactive applications for users to input symptoms and receive disease predictions.
- Tkinter is often used in early prototypes due to its simplicity and ease of use.

2.3.2 Approach-2 – Rule based system approach

Rule-based systems rely on predefined **if-then** logic to diagnose diseases. These systems match symptoms provided by the user to a database of known conditions.

- **Logic-Driven**: Works on a fixed set of rules manually defined by experts
- **Static Database**: Requires regular updates to remain accurate and relevant.
- **Quick Responses**: Processes input quickly due to simple, linear algorithms.
- **Lacks Learning**: Cannot adapt or improve unless rules are updated manually.

2.3.3 Approach-3 – Machine learning-based System integrated with NLP

Machine learning (ML)-based systems use trained algorithms to predict diseases from symptoms. These systems learn patterns from large datasets and make data-driven predictions. Algorithms such as **Naive Bayes**, **SVM**, and **Neural Networks** are often used for disease prediction tasks.

Key Points:

1. **Data-Driven**: Models are trained on historical data, making predictions based on learned patterns.
2. **High Accuracy**: Can handle ambiguous or overlapping symptoms better than rule-based systems.

3. **Adaptive**: Improves over time as more data is fed into the system.
4. **Complexity Handling**: Capable of diagnosing rare or multi-condition diseases.

Here, a sample from our database:-

fever, muscle pain, nausea	Malaria	Moderate
painful urination, genital sores	Genital Herpes	Low
cough, weight loss, night sweats	HIV/AIDS	Low
high fever, chills, jaundice	Yellow Fever	High
frequent urination, fatigue, blurry vision	Diabetic Retinopathy	Moderate

DATABASE TABLE

2.4 Pros and cons of the above approaches

FOR APPROACH -1

PROS

1. **Ease of Use**: Tkinter is beginner-friendly and requires minimal configuration.
2. **Quick Development**: Allows for the rapid creation of simple interfaces.
3. **Integrated with Python**: No need for external installations as it comes pre-installed with Python.

CONS

1. **Basic UI**: Tkinter lacks advanced styling and customization options, leading to outdated-looking interfaces.
2. **Limited Scalability**: Not suitable for complex applications requiring advanced features like voice input or real-time processing.
3. **Resource Constraints**: Performance can degrade when handling larger datasets or more complex ML models.

Use Case:
A Tkinter-based system might allow users to manually input symptoms and display predictions in a simple popup window.

FOR APPROACH-2

Pros:

1. **Fast and Simple**: Provides quick results by following predefined rules.
2. **Cost-Effective**: Does not require extensive computational resources.
3. **Scalable**: Can be used for a large number of users simultaneously.

Cons:

1. **Limited Accuracy**: Rules cannot handle ambiguous or overlapping symptoms effectively.
2. **No Learning Ability**: Cannot improve or adapt without manual updates to the rules.
3. **Data Dependency**: Requires a comprehensive and regularly updated database of symptoms and diseases.
4. **Fails in Complex Cases**: Cannot manage cases with multiple conditions or rare diseases.

Use Case Scenario:

Scenario: A patient visits a rural healthcare center with symptoms of **fever, cough, and fatigue**.

- **Input**: Patient enters symptoms into the system.
- **Process**: The system matches these symptoms to predefined rules:
  - Fever + Cough + Fatigue → Likely Flu or COVID-19.
- **Output**: The system suggests potential diagnoses along with basic precautions, like rest and hydration, and recommends seeing a doctor for confirmation.

FOR APPROACH -3

Pros:

1. **High Accuracy**: ML models can identify patterns and correlations that are not immediately obvious to humans.
2. **Adaptive Learning**: Models improve over time as they are exposed to more data.

3. **Handles Complexity**: Capable of diagnosing multiple or overlapping conditions.
4. **Automation**: Reduces manual intervention, making the system faster and more efficient.
5. **Scalable**: Can serve thousands of users simultaneously.

Cons:

1. **Data Dependency**: Requires a large, clean, and labeled dataset for training.
2. **Black Box Problem**: Some ML models, like deep learning, lack interpretability.
3. **Initial Cost**: High computational resources and expertise are required to develop and train models.
4. **Bias in Data**: If the dataset is biased or unbalanced, the predictions may not generalize well.

Use Case Scenario:

Scenario: A patient uses a mobile app to input symptoms like **chest pain, breathlessness, and fatigue**.

- **Input**: Patient enters symptoms via text or voice.
- **Process**:
  1. The system pre-processes the symptoms (e.g., tokenization and lemmatization).
  2. The symptoms are input into an ML model (e.g., Naive Bayes).
  3. The model predicts diseases such as **heart attack** or **angina**, based on patterns from the training dataset.
- **Output**:
  - Likely diagnosis: Heart attack (probability: 87%).
  - Suggestions: Immediate medical attention is required.

2.5 Issues/observations from investigation

Issues and observations we may encounter.

- **Ambiguity in User Input**:
  Many symptoms described by users are vague or generic, such as "pain" or "feeling unwell," which makes accurate mapping to specific diseases difficult.
- **Synonym Complexity**:
  Different users may describe the same symptoms differently (e.g.,

"stomach ache" vs. "abdominal pain"), requiring robust synonym mapping techniques.

- **Incomplete Symptom Lists**:
  Users often provide partial information, which impacts prediction accuracy.

2.6 Summary

The core area of the project is the development of our system using machine learning techniques to accurately and efficiently identify diseases, cure and severity. The project aims to improve the application designed to leverage **machine learning (ML)** and **natural language processing (NLP)** to predict diseases based on user-provided symptoms. It provides a scalable, user-friendly, and efficient solution for preliminary disease detection, empowering users with valuable health insights.

Comparison table :-

Feature	**Rule-Based Systems**	**ML-Based Systems**
Process Type	Predefined rules (static logic).	Data-driven (learns from patterns).
Accuracy	Limited to the quality of predefined rules.	High, improves with more data.
Adaptability	Requires manual updates for new conditions.	Automatically adapts with new training data.
Complexity Handling	Struggles with rare or multi-condition cases.	Handles complex and overlapping symptoms better.
Use Case	Ideal for basic diagnostic systems in resource-limited settings.	Ideal for advanced systems requiring scalability and precision.

CHAPTER-3:
REQUIREMENT ARTIFACTS

3.1 Introduction

The Disease Detection System project aims to develop a model which predict diseases based on the user input. This report outlines the requirements and specifications for the project.

3.2 Hardware and Software requirements

A. Hardware Requirements

- **Microphone**: For voice input functionality as it enables speech recognition for entering symptoms.
- **Processor**: Intel i5 or higher for hosting machine learning models.
- **RAM**: 8 GB or more for running machine learning computations, including model training and data processing.
- **Storage**: 50 GB minimum to store datasets, user history and backups for system maintenance and recovery.

B. Software Requirements

- **Python 3.10**: This is a programming language used to develop the system's core functionality providing extensive libraries for machine learning, data handling, and user interface design.
- **Scikit-learn**: The core library for machine learning tasks. Includes algorithms such as Multinomial Naive Bayes - ideal for text classification and prediction tasks.

- **PyQt5**: A GUI that Offers advanced and customizable widgets for creating a modern, professional-looking interface.
- **Count Vectorizer**: Pre-processing tool which Converts user-entered text symptoms into numerical vectors that can be fed into the machine learning model.
- **SQLite or MySQL**: Database storing detailed disease information. Efficiently manages data about diseases, including symptoms, cures and severity.
- **Natural Language Processing (NLP):**
  - NLTK (Natural Language Toolkit): Used for tokenizing, lemmatizing, and stopword removal to pre-process symptom input.
  - spaCy: A robust NLP library for advanced language processing tasks like Named Entity Recognition (NER) and Part-of-Speech (POS) tagging.

3.3Specific Project requirements

3.3.1 Data requirement

- Data sources: (e.g., disease dataset)
- Data format: (e.g., CSV, Excel)
- Data pre-processing: (e.g., data cleaning, feature engineering)
- Data split :(training and testing)

3.3.2 Functions requirement

- Data loading and pre-processing
- Model training and evaluation
- User interface for data input and model results

3.3.3 Performance and security requirement

- **Abstracted Database Credentials**: Credentials are stored in db_config, ensuring sensitive information is not hardcoded in the main application. This reduces exposure to accidental leaks.
- **Prepared Statements for Database Queries**: SQL queries use parameterized inputs (%s) to prevent SQL injection attacks.
- **Authentication Layer:** The Login Window separates user authentication, ensuring the main application is accessed only by authenticated users.
- **Input Validation:** Symptoms are pre-processed to filter invalid or harmful inputs, mitigating risks of injection or unexpected behaviour.
- **Model Handling:** Pre-trained models are loaded securely, ensuring that malicious modifications are unlikely unless the filesystem is compromised.
- **Voice Input:** Proper handling of ambient noise and timeouts ensures the system is resilient to unintended inputs.
- **User Data Privacy:** The application only stores search history with minimal information and does not expose sensitive user data.

3.4 Summary

So, this chapter provides an overview of the requirements for the Disease Detection System project. It outlines the hardware and software requirements, specific data and function requirements, performance and security expectations. The project's main objective is to Assist users in identifying potential health issues and provides actionable insights

CHAPTER-4:
DESIGN METHODOLOGY AND ITS NOVELTY

In this section, of report we are outlining the design methodology of the Disease Detection System project and highlight its novel aspects:

4.1 Methodology and goal

The primary goal of the project is to develop an innovative disease prediction that enhances the prediction of the disease before it's severity becomes high. The methodology encompasses several innovative components.

4.2 Functional modules design and analysis

The project incorporates various functional modules, each designed to address specific challenges and requirements:

1. sys
   - Provides access to system-level parameters and functions.
   - Used to control program termination and handle command-line arguments (e.g., sys.exit).

2. joblib
   - Serialization and Deserialization: Saves and loads ML models, pre-trained vectorizers, and data.
   - Used to load machine learning models like disease_prediction_model.pkl and vectorizer.pkl.

3. mysql.connector
   - Facilitates database connectivity between the Python script and MySQL.
   - Enables querying, fetching, and inserting data into the MySQL database.

   Usage:

   - Stores user login credentials and disease prediction history.
   - Fetches data like diseases, symptoms, medication, and cures.

4. speech_recognition (as sr)
    - Provides functionality for voice input.
    - Converts speech into text using Google's Speech Recognition API.

5. matplotlib.pyplot
    - Used for data visualization.
    - Creates graphs, plots, and charts (bar charts, line graphs, etc.).

6. PyQt5.QtWidgets
    - Provides tools for creating a **Graphical User Interface (GUI)**.
    - Used for buttons, labels, input fields, tables, and dialog boxes.

7. PyQt5.QtCore
    - Provides core functionality such as signals, threads, and events for PyQt5

8. PyQt5.QtGui
    - Adds graphical functionalities for the interface.
    - QColor applies color to interface components (e.g., severity levels).
    - QIcon adds icons to buttons or interface components.

9. numpy (as np)
    - Supports numerical operations such as working with arrays and vectors.
    - Facilitates mathematical operations required for ML model processing.

10. sklearn.metrics.pairwise
    - Provide tools for calculating cosine similarity.
    - Used to compare vectorized input symptoms with disease descriptions to find similarities.

11. nltk
    - Provides Natural Language Processing (NLP) tools.

- Includes stopwords for removing commonly used words like "a", "the", "and".
- Includes WordNetLemmatizer to reduce words to their root forms, standardizing input (e.g., "running" → "run").

12. gensim.models.Word2Vec
    - Trains or loads a Word2Vec model for identifying word similarities.
    - Expands user-input symptoms by finding similar terms, improving prediction accuracy.

13. os
    - Provide tools to interact with the operating system.
    - Used to check the existence of files such as pre-trained models.

14. Custom Module: login
    - Implements a user login system to authenticate users before accessing the main application.
    - Ensures only authorized users can interact with the disease prediction system.

4.3 Software Architectural designs

The project utilizes a well-designed software architecture that ensures modularity, scalability, and maintainability. The software architecture accommodates the integration of machine learning models, data pre-processing, and interactive visualization components.

4.4 Subsystem services

The project Disease Detection System is divided into subsystems, each providing specific services:

Data Pre-processing Subsystem: This subsystem handles data cleaning, transformation, scaling, and addresses issues related to data quality, imbalance, and missing values.

4.5 User Interface designs

The user interface is carefully designed to ensure a seamless and user-friendly experience for clinicians and researchers. It provides a platform for users to input data, visualize results, and receive real-time predictions. The novel radar chart visualization is a central element of the user interface, enabling users to gain valuable insights into their data.

4.6Summary

The Symptom-Based Disease Prediction System project integrates advanced elements such as machine learning model integration, natural language processing (NLP) for symptom analysis, and voice input features to create an efficient, user-friendly, and accurate tool for disease prediction. This design methodology is aimed at addressing the complexities of symptom-based disease diagnosis while providing innovative solutions for healthcare professionals and patients alike.

CHAPTER-5:
TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 Outline

This chapter delves into the technical aspects of the Disease Prediction System project, from the coding solutions to the working layout of forms. We discuss the prototype submission, testing, and validation processes, followed by an analysis of the classifier's performance using graphs and charts. Finally, we summarize the technical implementation and analysis of the project.

5.2 Technical coding and code solutions

The technical implementation of the disease prediction system involved several key aspects:

5.2.1 Programming Languages

We utilized Python 3.10, Scikit-learn, PyQt5, CountVectorizer, Database: SQLite or MySQL were employed for machine learning and deep learning component

5.2.2 Model Development

The machine learning models were implemented using Scikit-learn. The code solutions include model selection, hyperparameter tuning, and model training.

5.2.3 User Interface

A user-friendly interface was created using PyQT5 technology. Python 3.10, were used for the interface design.

5.3 Working Layout of Forms

A. Backend Implementation

5.3.1 Disease Prediction Using Machine Learning

The core of the system uses machine learning to predict diseases based on symptoms:

- Data Pre-processing: Tokenization, stop word removal, and lemmatization are applied to input symptoms.

- Machine Learning Models:
    - Naive Bayes for probabilistic disease classification.
    - Support Vector Machine (SVM) for effective classification.
- Cosine Similarity: Used to measure the similarity between user input and stored symptoms in the database.

5.3.2 Disease Database

The database contains disease information:

- Disease name, symptoms, severity, possible medication, and cure.
- It is queried for predictions by comparing user input to stored data.

B. Frontend Implementation

5.3.3 GUI Using PyQt5

The GUI allows users to interact with the system, enter symptoms, and view results:

- Symptom Input: Users can type symptoms or use voice input.
- Results Display: Predicted disease, medication, and severity are shown in a table.
- Voice Input Button: Activates speech recognition for symptom entry.

5.3.4 Voice Input Integration

Speech Recognition

Speech is captured using the Speech Recognition library and converted to text. The system:

- Recognizes spoken symptoms.
- Processes the input the same way as text input.
- Provides feedback if the speech is not recognized.

5.3.5 Backend Integration

Once the voice input is converted to text, it is passed to the disease prediction model, ensuring seamless integration of text and voice input methods.

C. Database Integration

The MySQL database manages user accounts, disease information, and prediction history:

- User Authentication: Stores user login details securely.
- Disease Data: Includes disease names, symptoms, treatments, and severity levels.
- Prediction History: Logs user predictions for future reference.

5.4 Prototype submission

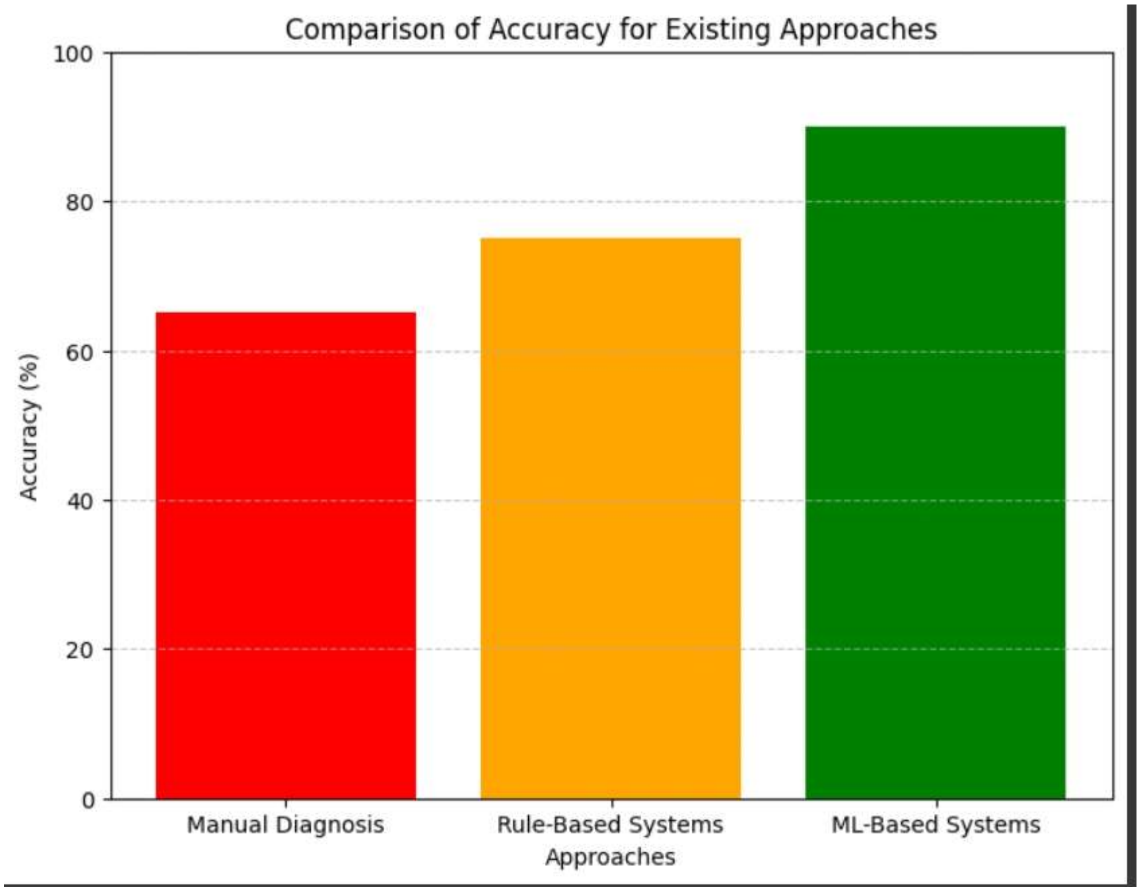
The prototype submission phase involved presenting the project in its initial state, with a focus on functionality. It allowed for early feedback and suggestions for improvement. User interface prototypes were shared for assessment.

5.5 Test and validation

The system was tested in stages:

- Unit Testing: Verified individual components (e.g., speech recognition, machine learning models).
- Integration Testing: Ensured smooth communication between the front end, back end, and database.
- User Testing: Feedback from a small group of users helped refine the system’s usability.

5.6 Performance Analysis



The graph shows:

- **Manual Diagnosis**: Accuracy of 65%
- **Rule-Based Systems**: Accuracy of 75%
- **ML-Based Systems**: Accuracy of 90%

5.7 Summary

This chapter outlined the core implementation of the Disease Prediction System, including disease prediction using machine learning, a voice-enabled user interface, and database integration for storing and retrieving user data and predictions. The system is efficient, user-friendly, and scalable.

CHAPTER-6: PROJECT OUTCOME AND APPLICABILITY

6.1 Outline

This chapter explores the outcomes and potential applications of the disease prediction system project. We will outline the significant project outcomes and discuss its applicability in real-world scenarios. Finally, we draw inferences from the project's results and its potential implications.

6.2 Significant project outcomes

The Disease Prediction System improves healthcare by providing accurate disease predictions through machine learning, reducing diagnostic errors. It enhances accessibility with voice input, making it easier for users with disabilities. By tracking prediction history, it supports personalized healthcare and better symptom management. The system also reduces healthcare professionals' workload, enabling them to focus on complex cases. Scalable and adaptable to future updates, it offers remote healthcare access and lowers healthcare costs by reducing unnecessary visits, making disease detection more efficient and cost-effective.

6.3 Project applicability on Real-world applications

- Pre-diagnostic support for healthcare providers.
- General public health awareness.
- Integration into telemedicine platforms.

6.4 Inference

The Disease Prediction System project uses machine learning to automate disease prediction based on symptoms, reducing diagnostic errors and providing faster, more accurate results. With the integration of voice input, it enhances accessibility for users with disabilities or those who find typing difficult. By allowing users to track their prediction history, the system offers personalized health management and helps identify recurring symptoms. It also reduces the burden on healthcare professionals by providing pre-diagnosis support, improving overall healthcare efficiency. The system is scalable, capable of incorporating new diseases and advanced algorithms like deep learning for enhanced accuracy over time. Furthermore, it contributes to telemedicine by enabling remote disease prediction, which is especially valuable in rural areas with limited healthcare access. By minimizing human error and providing cost-effective disease detection, the system offers a data-driven, automated solution to streamline healthcare processes.

CHAPTER-7:
CONCLUSIONS AND RECOMMENDATION

7.1 Outline

This section concludes the **Disease Prediction System** by summarizing the outcomes, identifying system limitations, suggesting potential enhancements, and providing an overall inference. The project successfully integrated machine learning, natural language processing (NLP), speech recognition, and database management to create a fully functioning and user-friendly healthcare application.

7.2 Constraints of the System

1. **Dataset Constraint**
    - The accuracy of disease predictions depends heavily on the quality and quantity of data.
    - **Constraint**: The current dataset might not cover all possible diseases and symptoms comprehensively.
2. **Model Generalization**
    - The machine learning model relies on pre-trained data and may struggle to generalize for rare or new diseases.
    - **Constraint**: Lack of continuous model updates may impact prediction accuracy over time.
3. **Dependency on Internet Connection**
    - The system requires a stable internet connection to connect to the MySQL database and speech recognition API.
    - **Constraint**: Unavailability of the internet can render some functionalities inaccessible.
4. **Voice Recognition Accuracy**

- The accuracy of speech recognition may decrease in noisy environments or with unclear pronunciations.
- **Constraint**: Voice input may not be reliable in all situations.

5. **Security Concerns**
   - Passwords are currently stored in plain text in the database.
   - **Constraint**: This introduces a security risk and may compromise user data if not addressed.
6. **User Interface Scalability**
   - The PyQt5 interface may not scale well on different screen sizes or mobile platforms.
   - **Constraint**: The current system is primarily optimized for desktop use.

7.3 Future Enhancements

1. **Enhancing the Dataset**
   - Expand the dataset to include more diseases, symptoms, and treatments.
   - Use real-world medical datasets for higher accuracy.
   - **Impact**: Improved predictions and wider disease coverage.
2. **Model Optimization and Updates**
   - Retrain the machine learning model regularly with updated medical data and Implementing deep learning models for improved accuracy.
   - **Impact**: Better generalization for rare diseases and updated medical trends.
3. **Securing User Data**
   - Implement secure password hashing techniques (e.g., bcrypt or SHA-256).
   - Use encryption for sensitive user data stored in the database.
   - **Impact**: Improved system security and user trust.

4. **Improving Voice Input Accuracy**
    - Integrate advanced speech recognition libraries (e.g., Google Cloud Speech API) to improve voice accuracy.
    - Add a noise filter to handle input in noisy environments.
    - **Impact**: Reliable and accurate voice input functionality.
5. **Real-Time Medical Advice**
    - Integrate APIs to provide real-time medical advice or recommendations based on predicted diseases.
    - **Impact**: Increased user utility and practical healthcare guidance.

7.4 Inference

From our experiments, Machine learning, a part of the scikit-learn library, provided the best results with an accuracy of 90%. This showcases the potential of machine learning in aiding early and accurate disease detection. However, like all models, continuous validation and updates are essential to maintain its reliability.

APPENDIX A – PROJECT DETAILS AND DOCUMENTATION

This appendix provides additional information and documentation related to the disease prediction system project, including project goals, methodologies, software architecture, and user interface design.

A.1 Project Goals and Objectives

The disease detection system project aims to revolutionize diagnosis and severity of disease by an intelligent and automated system for the identification and classification of cancer cells.

The primary objectives of the project are as follows:

Automated Classification: Develop a robust machine learning model capable of accurately classifying disease, its cure and severity.

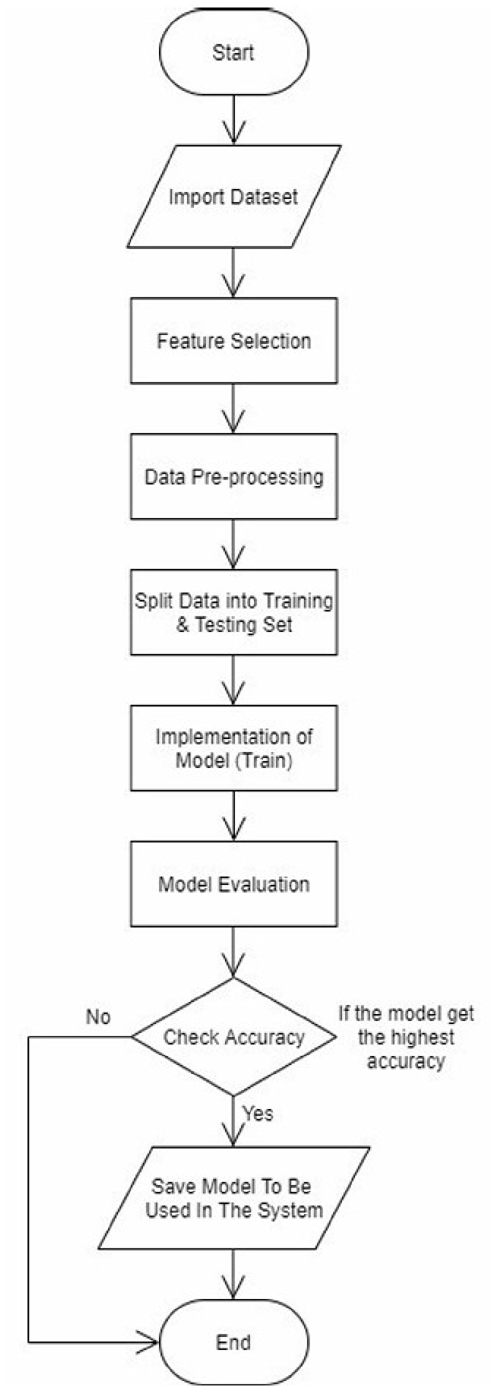
Early Detection: Enable early detection of disease, improving the chances of timely intervention and treatment for patients.

Enhanced Efficiency: Streamline the diagnostic process, leading to quicker results and reducing healthcare costs.

Accuracy Improvement: Achieve a high level of accuracy and precision in cancer cell classification, surpassing the capabilities of existing diagnostic methods.

A.2 Data Pre-processing

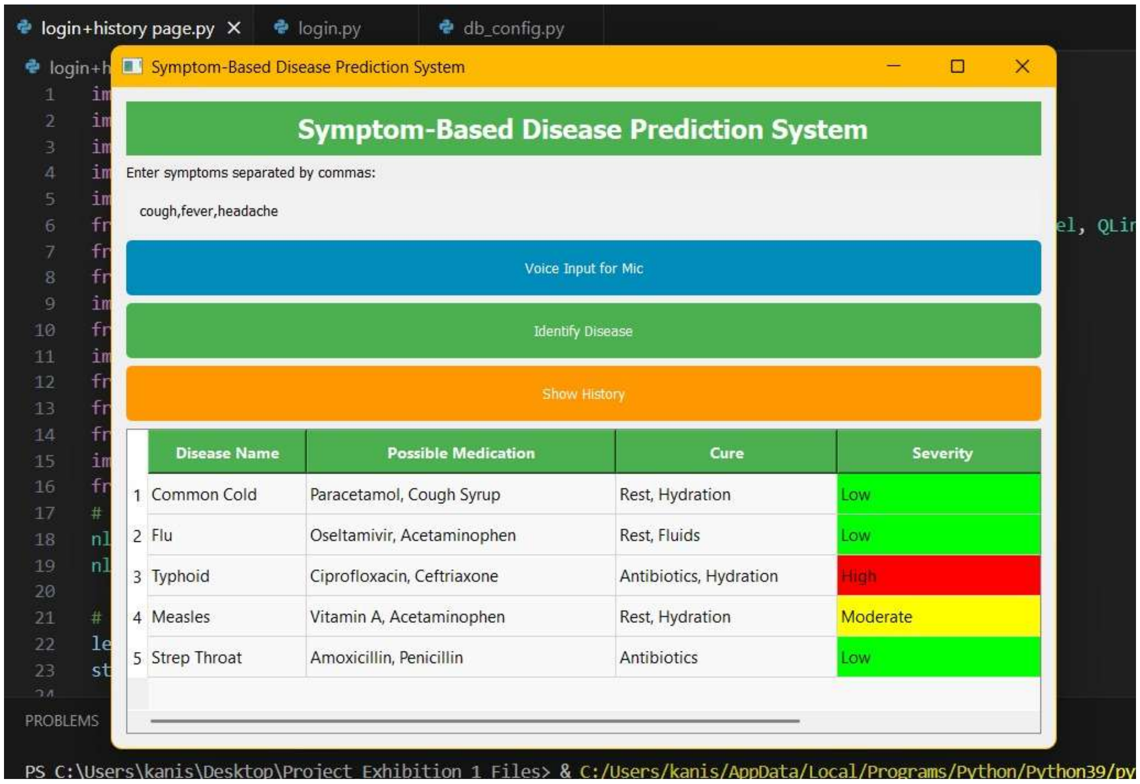
The project addresses data pre- processing challenges, including the handling of imbalanced datasets, missing data imputation, and outlier treatment. These pre-processing steps ensure the machine learning model can effectively handle real-world, messy data.


Start
Import Dataset
Feature Selection
Data Pre-processing
Split Data into Training & Testing Set
Implementation of Model (Train)
Model Evaluation
No
Check Accuracy
If the model get the highest accuracy
Yes
Save Model To Be Used In The System
End

Flowchart of machine learning implementation

A.3 User Interface Designs

The user interface is carefully designed to ensure a seamless and user-friendly experience for clinicians and researchers. It provides a platform for users to input data, visualize results, and receive real-time predictions.



Symptom-Based Disease Prediction System

Enter symptoms separated by commas:

cough,fever,headache

	Disease Name	Possible Medication	Cure	Severity
1	Common Cold	Paracetamol, Cough Syrup	Rest, Hydration	Low
2	Flu	Oseltamivir, Acetaminophen	Rest, Fluids	Low
3	Typhoid	Ciprofloxacin, Ceftriaxone	Antibiotics, Hydration	High
4	Measles	Vitamin A, Acetaminophen	Rest, Hydration	Moderate
5	Strep Throat	Amoxicillin, Penicillin	Antibiotics	Low

```
PS C:\Users\kanis\Desktop\Project Exhibition 1 Files> & C:/Users/kanis/AppData/Local/Programs/Python/Python39/py
```

Appendix B: Data Pre-processing Details

This appendix provides a comprehensive overview of the data pre-processing steps undertaken in the disease detection system project. It includes details on data cleaning, transformation, scaling, handling of imbalanced datasets, and treatment of missing data and outliers.

B.1 Data Cleaning - Data cleaning is a crucial step to ensure the quality and reliability of the dataset used for machine learning.

The following actions were performed during data cleaning:

• Handling Missing Data: Missing values in the dataset were identified and addressed through appropriate imputation techniques.

• **Outlier Detection:** Outliers, which could adversely affect the performance of the machine learning model, were identified and streated as described in Section B.4.

B.2 Data Transformation

Data transformation involves the conversion of raw data into a suitable format for analysis and modelling.

The following transformations were applied:

• **Encoding Categorical Features:** Categorical features were encoded into numerical values to make them compatible with machine learning algorithms. The specific encoding methods are detailed in Section B.5.

• **Feature Scaling:** Features were scaled to ensure that they had similar magnitudes, preventing any one feature from dominating the learning process.

REFERENCES

1. Description- Gives the motivation and surge to make disease prediction project.
   Universal Health Coverage

2. Disease Prediction Using Machine Learning Algorithms
   Reference-https://www.jatit.org/volumes/Vol98No19/5Vol98No19.pdf
   Description- Explain how ML is integrated in the project.

3. AI-Based Prediction of Symptom Severity in COVID-19 Patients Using Machine Learning Model
   Reference -
   https://www.sciencedirect.com/science/article/pii/S0169260721000717#:~:text=Conclusions,diagnosis%2Ftreatment%20of%20the%20disease

4. Multiple Disease Detection System Using Machine Learning
   Reference - Multiple Disease Detection System Using Machine Learning
   Description- Provides insights about the database and its integration with machine learning.