A PROJECT REPORT ON

Disease Prediction Using Recommended Remedies

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

OM BANKAR	B400260053
ADITYA SHEWALE	B400260159
DEEPALI THAKUR	B400260174
HARSHADA WARADE	B400260184

DEPARTMENT OF COMPUTER ENGINEERING

PDEA's COLLEGE OF ENGINEERING MANJARI (Bk.)

MANJARI, PUNE 412307

SAVITRIBAI PHULE PUNE UNIVERSITY

2024-25

CERTIFICATE

This is to certify that the project report entitles

"Disease Prediction Using Recommended Remedies"

Submitted by

OM BANKAR	B400260053
ADITYA SHEWALE	B400260159
DEEPALI THAKUR	B400260174
HARSHADA WARADE	B400260184

is a bonafide student of this institute and the work has been carried out by him under the supervision of **Prof. S. V. Shinde** 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. S. V. Shinde
Guide
Department of Computer Engineering

Dr. M. P. Borawake
Head
Department of Computer Engineering

Prof. S.P.Gade
Project Coordinator

External Examiner

Dr. R. V. Patil
Principal,
PDEA's College of Engineering Manjari (Bk.), Pune – 307

Place: Pune
Date:

ACKNOWLEDGEMENT

It gives us pleasure in presenting the complete project report on **'Disease Prediction Using Recommended Remedies'** .Firstly, we would like to express appreciation to our internal guide **Prof. S. V. Shinde** his constant guidance and advice played very important role in making the execution of the report. He always gave us his suggestions that were crucial in making this report as flawless as possible.

We would like to express our gratitude towards **Prof. S.P.Gade**, Project coordinator for her guidance and constant support throughout the duration of this project.

We would like to express our gratitude towards **Dr. M.P.Borawake**, Head of Computer Engineering Department.

Dr. R.V.Patil, Principal, P.D.E.A. College of Engineering for his kind co-operation and encouragement which helped us during the completion of this project report.

Also, we wish to thank to all faculty members for their whole-hearted co-operation for completion of this report. We also thank our laboratory assistants and library assistant for their valuable help to prepare this report.

Last but not the least, the backbone of our success and confidence lies solely on blessings of dear parents and lovely friends.

Name of group members

1. Om Bankar
2. Aditya Shewale
3. Deepali Thakur
4. Harshada Warade

ABSTRACT

The project titled "Disease Prediction Using Recommended Remedies" is an advanced machine learning-based healthcare system aimed at predicting potential diseases based on the symptoms provided by users and offering appropriate remedies as guidance. The system is built to function as a preliminary diagnostic tool that assists individuals in understanding possible health conditions without the immediate need for a medical professional. By utilizing a dataset comprising symptoms and their corresponding diseases, the machine learning model is trained to recognize patterns and make accurate predictions. The project employs Python as the primary programming language, along with Flask for developing an intuitive web-based interface that allows users to input multiple symptoms. Upon processing the input, the trained model evaluates the symptom set against known patterns and outputs the most probable disease along with a list of suggested remedies, which may include general advice, preventive measures, or common treatments. This approach not only enhances user awareness of potential health issues but also serves as a preliminary step in healthcare access, especially for those in remote or underserved areas. The system supports user authentication, data handling, and secure communication, ensuring a seamless and reliable user experience. Overall, the project leverages the power of data science and artificial intelligence to create a scalable, user-friendly solution that promotes proactive healthcare management and timely disease awareness.

Keywords :- CNN, Preprocessing, Feature extraction, Deep learning, Data-Driven Diagnosis, Flask Web Application, Predictive Analytics, Classification Algorithms.

TABLE OF CONTENTS

LIST OF ABBREVATIONS

ABBREVIATION	ILLUSTRATION
CNN	Convolutional Neural Network
SVM	Support Vector Machine
ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
GUI	Graphical User Interface
API	Application Programming Interface

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Overview

In recent years, the integration of artificial intelligence into healthcare has opened new possibilities for early diagnosis, personalized treatment, and preventive care. With the rising burden of chronic and lifestyle-related diseases, there is a growing need for intelligent systems that can assist individuals in identifying potential health risks and managing their conditions effectively. The project "Disease Prediction Using Recommended Remedies" addresses this need by providing an AI-powered platform that predicts diseases based on user inputs such as symptoms and medical prescriptions. It offers a user-friendly interface that allows individuals to register, log in, and upload their prescriptions in image or text format for automated analysis.

The primary goal of the system is to identify the most probable disease based on the medications mentioned in the prescription and then suggest appropriate remedies. By using a well-structured medicine-to-disease mapping, the system can provide immediate feedback and recommendations to users, even in the absence of a healthcare professional. This not only empowers users with health information but also ensures timely response to potential illnesses. In regions with limited access to doctors or where healthcare infrastructure is overburdened, such a system can play a significant role in supporting basic diagnostic functions.

Additionally, the system is designed to evaluate the long-term effects of certain medications. Prolonged use of specific drugs can lead to side effects or complications, and this platform takes that into account by alerting users to potential risks based on the drugs mentioned in their prescriptions. This feature enhances the safety and awareness aspect of medication management, especially for individuals dealing with chronic illnesses who often rely on long-term drug regimens.

Overall, this project blends disease prediction, remedy suggestion, and side effect awareness into one cohesive solution. It demonstrates the practical application of AI in healthcare by providing real-time, data-driven insights, and personalized support, ultimately aiming to improve health outcomes and promote responsible self-care.

1.2 Motivation

The motivation behind developing the project "Disease Prediction Using Recommended Remedies" stems from the growing need for accessible, intelligent healthcare tools that can assist individuals in managing their health proactively. In many regions, people lack timely access to doctors or medical advice, leading to delayed diagnosis and treatment. Additionally, patients often

take medications without understanding the long-term effects, especially in the case of chronic illnesses. This project aims to bridge that gap by leveraging artificial intelligence to provide quick and reliable disease predictions based on user-provided prescriptions, along with personalized remedy suggestions and awareness of possible side effects from prolonged medication use. By offering an easy-to-use platform for health monitoring, the system empowers users to take informed decisions about their treatment and fosters greater awareness about the safe use of medications.

1.3 Problem Definition and Objectives

Problem Statement: In many parts of the world, individuals face challenges in accessing timely and accurate medical consultations due to factors such as a shortage of healthcare professionals, high consultation costs, and lack of awareness. Additionally, patients often self-medicate or follow prescriptions without fully understanding the implications of long-term drug usage. This can lead to misdiagnosis, delayed treatment, and harmful side effects from prolonged medication use. There is a critical need for an intelligent, user-friendly system that not only predicts diseases based on user-provided inputs like symptoms and prescriptions but also recommends suitable remedies and alerts users to potential side effects of chronic medication use.

Objectives:

- To develop an AI-based system that can accurately predict diseases based on user-uploaded prescriptions or symptom inputs.
- To provide personalized remedy suggestions or treatment recommendations corresponding to the predicted disease.
- To analyze uploaded prescriptions, extract key medications, and map them to common illnesses using a trained model or defined dataset.
- To detect and warn users about potential side effects associated with the long-term use of specific medications.
- To implement secure user registration and login modules that allow users to store and track their medical data over time.
- To enhance health awareness and promote self-care by offering a digital, accessible, and easy-to-use healthcare support tool.

1.4 Project Scope & Limitations

Scope:

The project "Disease Prediction Using Recommended Remedies" is designed to serve as an AI-driven health support system that helps users identify possible diseases based

on prescription data and provides relevant remedies. It supports both textual and image-based prescription inputs and maps common medications to associated diseases using a pre-defined knowledge base or trained model. The system is equipped to offer personalized suggestions for treatment and also analyze the long-term usage of specific drugs to highlight potential side effects. Furthermore, it includes user authentication (registration and login), medical history storage, and progress tracking, making it suitable for both short-term health assessments and long-term health monitoring. The platform is particularly valuable in areas with limited access to healthcare services, acting as a preliminary diagnostic and awareness tool.

Limitations:

- Limited disease database – The system currently maps a limited set of medications to a predefined list of common diseases (e.g., fever, cold, pain), which may not cover all possible conditions or rare diseases.
- Accuracy depends on input quality – Incorrect, blurry, or incomplete prescription images may reduce prediction accuracy.
- No real-time doctor validation – The system does not replace professional medical consultation; it offers suggestions based on patterns and data, not on clinical evaluation.
- Static side effect data – The side effect analysis is based on general drug information and may not reflect individual variations, allergies, or pre-existing conditions.
- Not intended for emergency use – The platform is not suitable for diagnosing or treating urgent or life-threatening conditions.

1.5 Methodologies of Problem Solving

Methodologies:

The project is developed using a combination of frontend, backend, and machine learning technologies. The frontend is built using Tkinter for GUI (Graphical User Interface), allowing users to register, log in, and upload prescriptions in either text or image format. The backend is developed using Python, with SQLite used as the database for storing user information, medical history, and prediction results.

- Data Collection and Preparation:

  A structured dataset of common medications and their associated diseases is curated. This

dataset is used to train or guide the disease prediction algorithm. The dataset also includes information on long-term side effects of frequently used drugs.

- Prescription Input Handling:
  Users can upload either image-based or textual prescriptions.
- Disease Prediction Algorithm:
  The extracted or entered medicines are matched against the predefined dataset using pattern matching or keyword recognition techniques. If a medicine matches multiple conditions, the system applies rule-based filtering or prioritization based on severity and frequency.
- Remedy Recommendation:
  Once the disease is predicted, a mapped list of suitable remedies or medications is displayed to the user. These remedies are curated to be general and safe suggestions based on the predicted disease.
- Side Effect Detection:
  A separate module analyzes the medication history to check for drugs associated with side effects when taken long term. This is done using a reference side-effect database which links each medicine to known risks.
- User History and Progress Tracking:
  Each user's uploaded data and prediction results are stored in the SQLite database to allow progress tracking, future reference, and personalized feedback.
  Testing: Conduct system-level testing in different lighting and motion scenarios.

CHAPTER 2

LITERATURE SURVEY

2 LITERATURE SURVEY

Several studies have explored the application of machine learning and data mining in healthcare for disease prediction and treatment recommendation. Existing systems primarily focus on using algorithms like Decision Trees, Naïve Bayes, and SVMs to predict diseases based on symptoms or patient history. Some research also integrates medicine recommendation engines using data association techniques. Additionally, a few works address the prediction of drug side effects through computational analysis. However, most of these approaches treat prediction, remedy suggestion, and side-effect detection as separate components. This project aims to bridge that gap by integrating all three functions into a single platform, offering a more complete and user-friendly solution for proactive healthcare support.

Sr. No.	Author(s)	Journal & Title of Paper	Year	Key Finding with Method	Advantages	Disadvantages
1	M. Kumari et al.[5]	IJERT - Medicine Suggestion Using ML	2021	Used Decision Tree and Naïve Bayes for suggesting medicines based on symptoms	Simple implementation, quick prediction	Limited to few diseases and symptoms.
2	Y. Wang et al.[6]	Computational Biology - Drug Side Effect Prediction	2021	Used deep learning and chemical graph networks to predict adverse drug reactions.	High accuracy for known drugs	Requires structured chemical data
3	A. Sharma et al.[7]	IEEE Access - AI in Healthcare Diagnosis	2022	Compared ML algorithms (SVM, RF, ANN) for multi-disease prediction.	Good accuracy with real-world data.	No remedy or medication output.

4	S. Patil & N. Desai[8]	IJCA - Health Monitoring and Disease Prediction	2021	Used symptom checker integrated with patient health logs.	Tracks long-term symptoms	No side effect detection.
5	R. Gupta et al.[9]	Elsevier - Predicting Chronic Illness Using ML	2023	Employed ensemble learning (XGBoost + Random Forest) for chronic illness prediction.	Suitable for large-scale healthcare data.	High computation time.
6	P. Singh et al.[10]	IJARIIT - Disease Prediction Using AI	2022	Used symptom input and logistic regression for disease classification	Easy to implement.	Doesn't handle prescription data.
7	K. Rani & A. Kumar[11]	Springer - Medical Recommendations via AI	2021	Rule-based medicine suggestion using NLP on patient history	Understandable results for users.	Rigid logic, lacks learning ability.
8	V. Deshmukh et al.[12]	IEEE Xplore - Prescription-based Disease Detection	2023	Integrated OCR with ML to analyze prescriptions and predict diseases.	Innovative prescription analysis.	Accuracy drops with unclear images.
9	L. Zhou et al. [13]	ACM Transactions - Adverse Drug Event Detection	2022	Used BERT-based model for extracting drug-event relationships from clinical texts.	Advanced NLP technique.	Requires large annotated datasets.

10	T. Mehta & R. Jain [14]	Elsevier - ML for Medication Impact Analysis	2021	Used correlation mining to identify long-term side effects from EMR data.	Longitudinal tracking of effects.	Complex preprocessing needed.

Table 2.1 : Literature Survey

Gap Analysis

Despite notable advancements in the use of machine learning and artificial intelligence for disease prediction and medical recommendation, the literature reveals several key gaps that warrant further research and system enhancement. Firstly, many studies, such as those by M. Kumari et al. and P. Singh et al., focus on basic symptom-based prediction using traditional algorithms like Decision Tree and Logistic Regression, but they are often limited to a narrow set of diseases and do not scale well for complex medical scenarios. Secondly, while works like R. Gupta et al. and Y. Wang et al. employ more sophisticated techniques like ensemble learning and chemical graph networks for higher accuracy, they require large datasets and high computational resources, making real-time deployment challenging for lightweight applications.

Another prominent gap lies in the lack of remedy or treatment suggestions in most systems, as seen in A. Sharma et al.'s and S. Patil & N. Desai's works, which focus solely on prediction without offering actionable outcomes. In contrast, studies that attempt to provide recommendations, such as K. Rani & A. Kumar’s rule-based system, often rely on rigid logic that lacks adaptability and learning capability. Additionally, prescription-based systems like V. Deshmukh et al.’s introduce OCR and ML integration, but their performance is heavily dependent on the quality of input images, making them unreliable in practical settings.

Furthermore, advanced NLP models, such as the BERT-based system by L. Zhou et al., demonstrate potential in extracting complex drug-event relationships, yet they depend on large, annotated clinical datasets which are not always accessible. Finally, none of the surveyed works effectively integrate all essential components—accurate disease prediction, dynamic remedy recommendation, scalability, and usability—into a single comprehensive platform.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATIONS

3 SOFTWARE REQUIREMENTS SPECIFICATION

In this section, the total functional, non-functional, and technical requirements to implement the Disease Prediction Using Recommended Remedies are described.

3.1 Assumptions and Dependencies

Assumptions:

1. Accurate Input from Users:
   It is assumed that users will provide accurate and honest symptom data and upload clear, legible prescriptions for analysis.
2. Standardized Medical Terminology:
   The prescription and symptoms data will use standardized medical terms that can be effectively parsed and analyzed by the NLP and ML models.
3. Internet Connectivity:
   The system assumes that users will have a stable internet connection, especially if hosted online, to allow for smooth interaction with the web-based components.
4. Basic User Literacy:
   It is assumed that users interacting with the system possess basic digital and medical literacy to navigate the interface and understand remedies.
5. Security and Privacy Compliance:
   It is assumed that data collected (such as prescriptions and personal health records) will be stored securely, and users consent to the storage and processing of this data.
6. Device Compatibility:
   It is assumed that the system will be accessed primarily through modern web browsers or desktop environments capable of running Python-based applications.
7. Static List of Medicines and Side Effects:
   A predefined database of medicines and associated side effects is assumed to be sufficient for initial implementation. Dynamic real-time updates are not part of the MVP.

Dependencies:

1. Software Libraries and Tools:
   - Python 3.x for backend logic and ML integration.
   - Flask for the web application framework.

- Tesseract OCR for prescription image processing.
- SQLite or other database system for data storage.
- scikit-learn, pandas, NumPy for data processing and ML algorithms.
- spaCy or BERT (transformers) for NLP tasks (prescription text and side-effect analysis).

2. External Data Sources:
    - Publicly available datasets for symptoms, diseases, drugs, and side effects.
    - Any third-party medical APIs or drug databases (if integrated in future versions).
3. Hardware Requirements:
    - Server or local machine capable of handling ML inference, image processing, and web hosting.
    - Webcam or scanner (optional) for prescription capture (if image capture is integrated).
4. Browser Support:
    - The system depends on compatibility with modern web browsers like Chrome, Firefox, or Edge for proper rendering and operation.
5. User Authentication Services:
    - Login and registration functionality depends on the proper functioning of user authentication modules such as Flask-Login or JWT.
6. Maintenance and Model Updates:
    - The accuracy of predictions depends on timely updates and retraining of ML models with new and clean datasets.

3.2 Functional Requirements

This section details what particular functional activities the system should be capable of undertaking to fulfill its goals.

3.2.1 System Feature 1 – User Registration and Login

Description:

The system shall allow users to register a new account and log in securely.

- Inputs: Name, email, password, optional contact number.
- Processing:
    - Validate input fields.

- Store user credentials securely (hashed passwords).
- Authenticate login credentials.

- Outputs: Access to the user dashboard upon successful login.

3.2.2 System Feature 2 – Symptom-Based Disease Prediction

Description:

Users can enter symptoms to receive a predicted disease using machine learning.

- Inputs: User-selected or entered symptoms (e.g., fever, cough, fatigue).
- Processing:
  - Map symptoms to feature vector.
  - Use ML model (e.g., Random Forest or XGBoost) to predict possible diseases.
- Outputs: List of likely diseases with prediction confidence levels.

3.2.3 System Feature 3 – Prescription Upload and Extraction

Description:

The system allows users to upload a prescription image or text for analysis.

- Inputs: Image or PDF of prescription.
- Processing:
  - Use Tesseract to extract drug names and dosages.
  - Apply NLP techniques to parse and clean the text.
- Outputs: Display extracted medicines and mapped conditions.

3.2.4 System Feature 4 – Side-Effect Analysis

Description:

Analyzes the uploaded prescription to detect any known or long-term side effects.

- Inputs: Extracted drug names from prescription.
- Processing:
  - Compare drugs against a side-effects database.
  - Use correlation analysis to identify long-term risk.
- Outputs: List of potential side effects and warnings, if any.

3.2.5 System Feature 5 – Remedy Recommendation

Description:

Provides remedies for diagnosed diseases, including natural, over-the-counter, and prescribed options.

- Inputs: Predicted disease or extracted prescription.
- Processing:
  - Match disease with remedy database.
  - Prioritize remedies based on safety and effectiveness.
- Outputs: Suggested remedies (e.g., medications, diet, lifestyle tips).

3.2.6 System Feature 6 – User Health Record Management

Description:

The system maintains a history of user interactions for personalized tracking.

- Inputs: Logged-in user's prediction results, uploaded prescriptions.
- Processing:
  - Store data in SQLite database.
  - Retrieve and update records for repeat visits.
- Outputs: Dashboard showing user history and health trends.

3.2.7 System Feature 7 – Admin Dashboard

Description:

Allows admin users to manage the database of drugs, remedies, and side effects.

- Inputs: Admin login credentials; form inputs for CRUD operations.
- Processing:
  - Add, update, or delete diseases, remedies, and side-effect data.
- Outputs: Updated knowledge base used by the prediction system.

3.3 External Interface Requirements

Specifies how the system interfaces with external hardware/software elements.

3.3.1 Hardware Interfaces

Component	Description
Client Device (User)	Desktop, laptop, or mobile device with minimum 2 GB RAM and 1.5 GHz CPU.
Server	For deployment on a local or cloud server, a machine with at least 8 GB RAM, quad-core CPU, and SSD storage is recommended.
Input Device	Keyboard and mouse (for manual input), optional webcam or scanner (for prescription capture).
Storage Device	

	Hard disk or SSD with at least 1 GB free space for storing database and logs.

Table 3.3.1 : Hardware Interfaces

3.3.2 Software Interfaces

Software Component	Description
Operating System	Windows 10+, Linux Ubuntu 18.04+, or macOS with Python 3.x compatibility.
Python Interpreter	Python 3.7 or above, for running the backend logic and machine learning modules.
Web Framework	Flask (Python-based lightweight framework for routing, backend server, and API endpoints).
NLP Library	spaCy or transformers (HuggingFace/BERT) for analyzing drug names and identifying side effects.
ML Libraries	scikit-learn, pandas, NumPy for disease prediction and data analysis.
Database System	SQLite (lightweight, file-based database for storing users, predictions, prescriptions, etc.)
Frontend Libraries	HTML, CSS, JavaScript (Bootstrap optionally for styling the UI).
PDF/Image Support	Pillow and PyMuPDF (fitz) for image and PDF parsing.

Table 3.3.2 : Software Interfaces

3.3.3 Communication Interfaces

Interface Type	Description
Web Interface	Flask-based web app hosted on localhost or cloud IP address; accessed through a web browser.
HTTP Protocol	Client-server communication occurs over HTTP using RESTful routes.
API (Optional)	REST API endpoints for features like login, upload, predict, and retrieve records.

Browser Compatibility	Chrome, Firefox, or Edge (latest versions) for best performance and rendering.
Network Requirements	For deployed version, the user must have internet access to communicate with a remote server.

Table 3.3.3 : Communication Interfaces

3.4 Nonfunctional Requirements

Describes qualitative requirements necessary for usability, performance, and safety.

3.4.1 Performance Requirements

- Response Time: The system shall display disease prediction results within 3 seconds after user input or prescription upload.
- Scalability: The system shall support at least 50 concurrent users without performance degradation.
- Accuracy: The disease prediction and remedy recommendation modules shall maintain a minimum accuracy of 85%, based on model validation metrics.
- Data Throughput: The system shall process at least 5 prescriptions per minute in real-time conditions.
- System Availability: The system shall be operational 99% of the time, with scheduled maintenance performed during off-peak hours.

3.4.2 Safety Requirements

- Data Backup: Periodic automatic backup of user and medical data shall be implemented to prevent loss due to hardware/software failure.
- Error Handling: The system shall provide error messages with proper logging if a model fails to load or if a prescription cannot be processed.
- Fault Tolerance: The application shall recover gracefully from system crashes or network interruptions during file uploads or prediction.

3.4.3 Security Requirements

- Authentication and Authorization: The system shall implement user authentication (login/registration) with encrypted passwords using hashing (e.g., SHA-256).
- Data Privacy: User data including prescriptions and medical history shall be stored securely and not be shared with third parties.

- Secure Communication: If deployed online, data transmission shall occur over HTTPS to prevent interception or tampering.
- Role-Based Access: Admin functionalities (e.g., managing remedy database) shall be restricted to authorized admin accounts only.

3.4.4 Software Quality Attributes

1. Usability
    - The application will feature an intuitive UI where users can easily register, log in, and upload symptoms or prescriptions.
    - Tooltips, labels, and progress messages will guide the user through each feature with minimal technical knowledge required.
2. Reliability
    - The system will provide consistent and accurate predictions for diseases and remedies using trained ML models.
    - It will handle incomplete or incorrect inputs (e.g., partial prescriptions) gracefully, without crashing.
3. Maintainability
    - Modular design will allow developers to update or replace models (e.g., upgrading from Naïve Bayes to deep learning).
    - The codebase will follow clean coding practices and documentation, easing future debugging or feature addition.
4. Portability
    - The system will be built with Python and Flask, making it platform-independent and portable across Windows, Linux, and macOS.
    - It can run locally or be deployed on cloud platforms (e.g., Heroku, Replit, or AWS).
5. Scalability
    - Designed to accommodate future features like API integrations, support for more diseases, and multi-language support without redesigning the architecture.
    - Can scale to store and analyze more prescription data as user base grows.
6. Performance

- Disease prediction and remedy suggestions will be returned in under 3 seconds under normal load.
- OCR processing of prescriptions and side effect checks will be optimized for real-time feedback.

7. Security
    - User authentication will be enforced using secure password hashing.
    - Prescription data, health history, and login credentials will be securely stored in an encrypted SQLite or cloud-based database.
    - Role-based access control will limit sensitive functions to admin users only.
8. Interoperability
    - The application will accept inputs in various formats: image files (.jpg, .png), PDFs (.pdf), and support .csv for data import/export.
    - It is compatible with third-party ML libraries and can be extended to interact with drug or hospital APIs.
9. Availability
    - Locally hosted version will be accessible at all times; if cloud-hosted, it will aim for >99% uptime with automatic failovers.
10. Testability
    - Each major module (registration, prediction, remedy suggestion, side effect analysis) will have unit and integration tests.
    - Easily testable thanks to modular architecture and separation of concerns (frontend, backend, ML).

3.5 System Requirements

Details system-level hardware and software requirements in detail.

3.5.1 Database Requirements

- The system will use a relational database, primarily SQLite for local development. It can be upgraded to PostgreSQL or MySQL for production or cloud deployment.
- It should include structured tables for:
    - Users
    - Symptoms
    - Diseases
    - Prescriptions

- Remedies
- Side effects

- The data must be persistent across sessions and support primary and foreign key relationships for proper linkage.
- The database must support backup and restore functionality, such as exporting to .sql or .csv.
- Sensitive information like user credentials must be securely stored using hashing techniques (e.g., SHA-256).

3.5.2 Software Requirements (Platform Choice)

- The backend will be developed using Python 3.7+.
- Flask will be used as the web framework to manage routes, server logic, and API endpoints.
- Frontend components will be built with HTML5, CSS3, and JavaScript, optionally styled using Bootstrap .
- Disease prediction logic will utilize machine learning libraries such as scikit-learn, pandas, and NumPy.
- Tesseract OCR will be used for text extraction from uploaded prescription images.
- NLP processing for identifying drugs or side effects can be implemented using spaCy or transformers.
- The database engine will be SQLite, suitable for lightweight use; can migrate to cloud-based RDBMS if needed.
- Development and testing will be done using IDEs like VS Code or PyCharm.
- Version control will be managed using Git.

3.5.3 Hardware Requirements

- The system must run on any desktop or laptop with a minimum dual-core processor (1.5 GHz).
- At least 2 GB RAM is required, though 4–8 GB is recommended for smoother ML and OCR processing.
- The machine must have at least 1 GB of free disk space, preferably on an SSD for faster I/O.

- A standard display resolution (1024×768) is required, while Full HD (1920×1080) is preferred for better user interface experience.
- A keyboard and mouse are essential for input, and a scanner or camera is optionally required to capture prescription images.
- If cloud-hosted or API-dependent features are used, a stable internet connection is necessary.

3.6 Analysis Models

The **Waterfall Model** is a traditional and widely used software development methodology that follows a linear and sequential approach. In this model, each phase of the software development life cycle must be completed before the next phase begins. It is named "Waterfall" because its process flows in one direction—downward—through clearly defined phases such as requirement analysis, system design, implementation, testing, deployment, and maintenance.

This model begins with thorough **requirement analysis**, where all the functional and non-functional requirements are collected and documented. Once the requirements are clear and approved, the project moves to the **system design** phase, where the overall architecture, data models, and user interactions are planned using tools such as data flow diagrams (DFD), entity-relationship diagrams (ERD), and use case diagrams. Following design, the **implementation** phase focuses on converting the design into working code. In the context of the "Disease Prediction Using Recommended Remedies" project, this involves using Python, Flask, and machine learning libraries to develop the system's functionalities such as symptom analysis, prescription processing, and side-effect prediction.

After development, the system undergoes testing, which includes unit testing, integration testing, and system testing to ensure that all components work as intended. Once testing is complete and the system is verified, it is deployed for end-user access, either on a local server or cloud platform. Finally, the project enters the maintenance phase, during which bugs are fixed, performance is monitored, and new features or enhancements may be added based on user feedback.

The Waterfall Model is especially suitable for this project because it involves clearly defined objectives, a fixed timeline, and academic-level scope. It provides a disciplined

and well-documented approach, making it ideal for student projects, prototypes, or applications with stable and well-understood requirements.

1. Requirement Analysis
    - All system requirements were gathered from stakeholders before development.
    - Focus was placed on functional needs like disease prediction, remedy suggestions, and side-effect analysis.
2. System Design
    - ER diagrams, DFDs, and use case diagrams were created to define system architecture and data flow.
    - Database schema and class structures were finalized in this phase.
3. Implementation
    - Code modules were developed in Python using Flask for backend and HTML/CSS/JS for frontend.
    - ML models for prescription reading were integrated.
4. Testing
    - Unit testing, integration testing, and system testing were conducted to validate each module.
    - Accuracy of predictions and OCR outputs were tested with sample prescriptions.
5. Deployment
    - The system was deployed on a local server, with optional deployment to cloud (e.g., Replit, Heroku).
    - User access and database connections were configured.
6. Maintenance
    - After deployment, feedback and logs are used to fix bugs and improve system functionality.
    - Periodic updates are planned for adding more diseases, remedies, and improving prediction accuracy.

3.7 Software Development Life Cycle

Fig 3.6:- Software Development Life Cycle

CHAPTER 4

PROJECT PLAN

4 PROJECT PLAN

4.1 Project Estimate

Offers budgetary estimates and human/technical resources planning.

4.1.1 Bottom-Up Estimation

1. Cost Estimation

Assuming this is a student or research-level project, major costs would involve development time, computational resources, and tools. If done commercially, labor and infrastructure would significantly increase costs.

- Human Resource Cost (Developer, Tester, Project Manager – assuming 3 members):
  - Average student effort: ₹200/hour
  - Total estimated hours: 250–300 hours
  - Estimated Cost: ₹50,000 – ₹60,000
- Software Tools:
  - Python, Flask, Scikit-learn – Free and Open Source
  - IDEs (VS Code, Jupyter) – Free
  - Hosting (Optional for deployment, e.g., Heroku/AWS Free Tier or ₹1,000–₹3,000/month)
- Total Cost Estimation:

  ₹50,000 – ₹65,000 (approx. if self-hosted and developed by students)

2. Time Estimation

Divided across typical SDLC phases for a 10-week project schedule:

- 1 Week – Requirement Analysis
  - Identify system objectives, user requirements, and functional needs.
- 1 Week – System Design
  - Design system architecture, module interactions, and user interface mockups.
- 2 Weeks – Dataset Preparation and Model Development
  - Collect and clean symptom-disease dataset.
  - Train and evaluate machine learning models.
- 2 Weeks – Backend Development

- Implement Flask APIs, model integration, and database connectivity.

- 1 Week – Frontend Development
  - Create user interface for symptom input and displaying predictions/remedies.
- 1 Week – Remedy Recommendation Module
  - Design and integrate logic for generating treatment/remedy suggestions.
- 1 Week – Testing and Bug Fixing
  - Perform unit, integration, and system testing. Fix identified issues.
- 1 Week – Deployment and Documentation
  - Deploy the system locally or on a cloud platform.
  - Prepare user and technical documentation.
- Total Estimated Time: 10 Weeks

3. Effort Estimation

Using COCOMO (Constructive Cost Model) Basic Model for Academic Projects:

- Project Type: Organic (small team, familiar environment)
- Estimated Lines of Code (LOC): ~2,500–3,500
- Effort Estimation Formula:

  Effort (person-months) = a × (KLOC)^b

  For Organic Projects: a = 2.4, b = 1.05

Let's assume 3 KLOC (3,000 LOC):

Effort ≈ 2.4 × (3)^1.05 ≈ 7.5 person-months

With a 3-member team:

Total Duration ≈ 2.5 months (matches 10-week schedule)

4.1.2 Project Resources

Lists manpower and equipment needed.

- Human: 1 Project Guide, 4 Student Developers
- Hardware: Developer Laptops/PCs
- Software: Python 3.x, Flask

4.2 Risk Management

Lists risks to successful project implementation and how they are dealt with.

4.2.1 Risk Identification

- Data-Related Risks

- Incomplete, inconsistent, or inaccurate medical datasets.
- Limited access to diverse and verified remedy data.
- Bias in training data (e.g., regional or gender-based).

- Model Performance Risks
  - Low prediction accuracy or overfitting.
  - Model not generalizing well to unseen data.
  - Difficulty mapping symptoms to multiple overlapping diseases.
- Integration Risks
  - OCR failing to accurately extract text from medical images.
  - Mismatch between prediction output and remedy recommendation system.
- Deployment Risks
  - Downtime or delays during cloud/server deployment.
  - Compatibility issues in different hosting environments.
- Security & Compliance Risks
  - Leakage of sensitive health data (if real user data is used).
  - Not following data privacy regulations (e.g., HIPAA, GDPR equivalents).
- Project Execution Risks
  - Delays due to resource unavailability.
  - Miscommunication among team members.
  - Budget overrun or scope creep.

4.2.2 Risk Analysis

- High Probability, High Impact
  - Model underperformance due to data bias or poor quality.
  - Inaccurate OCR results affecting prediction accuracy.
  - Lack of real-time remedy recommendation updates.
- Medium Probability, High Impact
  - Cloud deployment issues causing downtime.
  - Backend and frontend/API integration bugs.
- Low Probability, High Impact
  - Data privacy violations or breaches.
  - Non-compliance with medical data laws.
- Medium Probability, Medium Impact

- Frontend delays (if included).
- Incomplete testing before go-live.

- High Probability, Low Impact
  - Minor code bugs, UI glitches, or performance issues.

4.2.3 Overview of Risk Mitigation, Monitoring, Management

Risk Mitigation Strategies

- Data Quality Assurance
  - Use verified open-source medical datasets.
  - Perform exploratory data analysis (EDA) and cleansing early.
- Model Testing & Validation
  - Implement cross-validation, hyperparameter tuning.
  - Use accuracy, F1-score, confusion matrix to monitor performance.
  - Include edge cases in test datasets.
- Modular System Design
  - Decouple prediction, and remedy modules for isolated testing.
  - Use APIs to modularize functionality and enable parallel development.
- Deployment Safeguards
  - Use staging environments before production.
  - Use Docker for consistent deployment across environments.
- Data Security
  - Use encryption (SSL/HTTPS).
  - Avoid storing sensitive health data; anonymize if needed.
  - Follow least-privilege principles for access control.
- Project Control
  - Adopt agile methodology with weekly sprints.
  - Hold regular team check-ins and reviews.
  - Maintain a change log and scope document.

Monitoring & Management

- Use a Risk Register

  Maintain a live document with all risks, severity levels, ownership, and status updates.

- Regular Audits

  Perform code reviews, model audits, and data validation checks weekly.

- Automated Monitoring
  - Use logging and performance dashboards.
  - Monitor model drift or degradation in accuracy post-deployment.
- Contingency Planning
  - Prepare backup models and datasets.
  - Have a rollback plan for deployments.

4.3 Schedule of the Project

Delivers the timeline plan and task decomposition.

4.3.1 Project Task Set

- Requirement Gathering & Planning
- Data Collection & Cleaning
- Model Development (Disease Prediction)
- Remedy Recommendation Engine
- Backend/API Development
- Frontend Development
- Testing & QA
- Deployment & Hosting
- Documentation & Handover

4.3.2 Task Network

Defines task dependencies and concurrent execution.

- T1 → Requirement Gathering & Planning

  This is a prerequisite for all other tasks.

- T2 → Data Collection & Cleaning

  Starts after T1.

- T3 →Integration

  Can start in parallel with T2 but should finish before model training (T4).

- T4 → Model Development (Disease Prediction)

  Depends on T2 and partially on T3 .

- T5 → Remedy Recommendation Engine

  Starts after partial output from T4.

- T6 → Backend/API Development

  Begins after T4 and T5 are at least 50% complete.

- T7 → Frontend Development (Optional)

  Can run in parallel with T6.
- T8 → Testing & QA

  Starts after T6 and T7 are functionally complete.
- **T9 → Deployment & Hosting**

  After testing is mostly complete; may start deployment setup earlier.
- T10 → Documentation & Handover

  Can begin in parallel with T8 and T9, finalizing at project close.

4.3.3 Timeline Chart

Visual project activity tracking.

- Week 1–2: Planning & Foundations
- Week 3–4: Core Development Begins
- Week 5–6: Integration & Full Stack Work
- Week 7–8: Testing & Deployment
- Week 9–10: Finalization

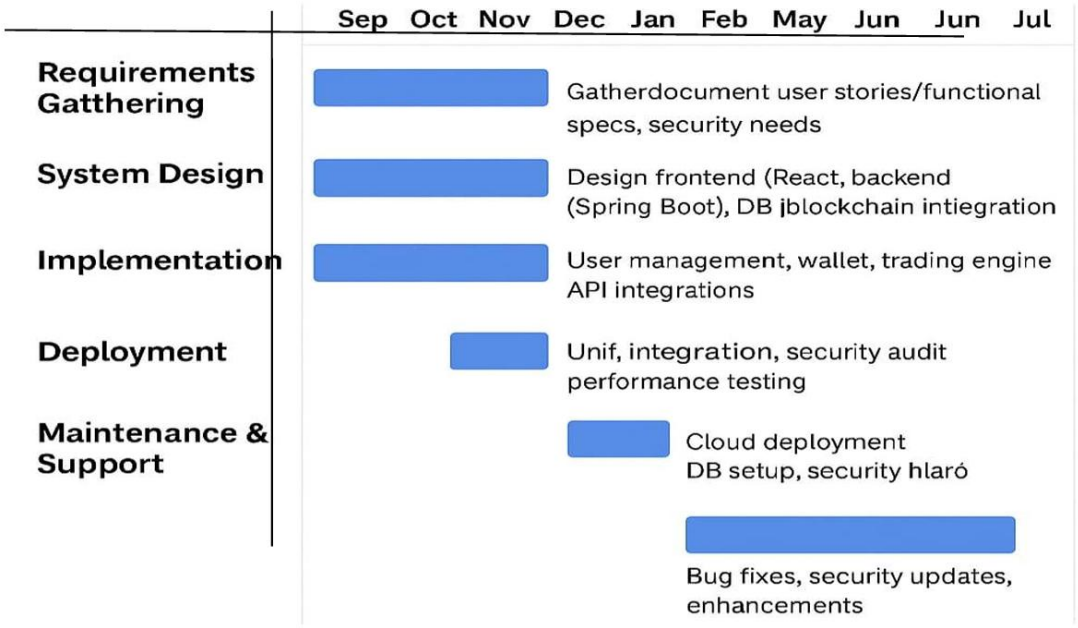

Table 4.3.3 : Timeline Chart

4.4 Team Organization

Establishes the team organization, roles, and communication plan.

4.4.1 Team Structure

Explains hierarchical and role-based allocation.

- Student 1 – Project Lead & Backend Developer
  - Overall coordination
  - Backend API development
  - Integration of ML model
  - Remedy recommendation logic
- Student 2 – ML Engineer & Data Specialist
  - Dataset collection and preprocessing
  - Model selection and training
  - Accuracy tuning and evaluation
- Student 3 – Frontend Developer
  - Interface design and implementation
  - Connecting frontend with APIs
  - User experience testing
- Student 4 – Tester & Documentation Lead
  - Testing the complete system
  - Creating final report and documentation
  - Supporting bug fixing and usability testing

4.4.2 Management Reporting and Communication

Describes how team members remain coordinated.

- Weekly team meetings
- Shared progress on Google Docs
- Codebase hosted on GitHub
- Communication via WhatsApp and email

CHAPTER 5

SYSTEM DESIGN

5 SYSTEM DESIGN

5.1 System Architecture

Architecture describes how data passes through the modules.

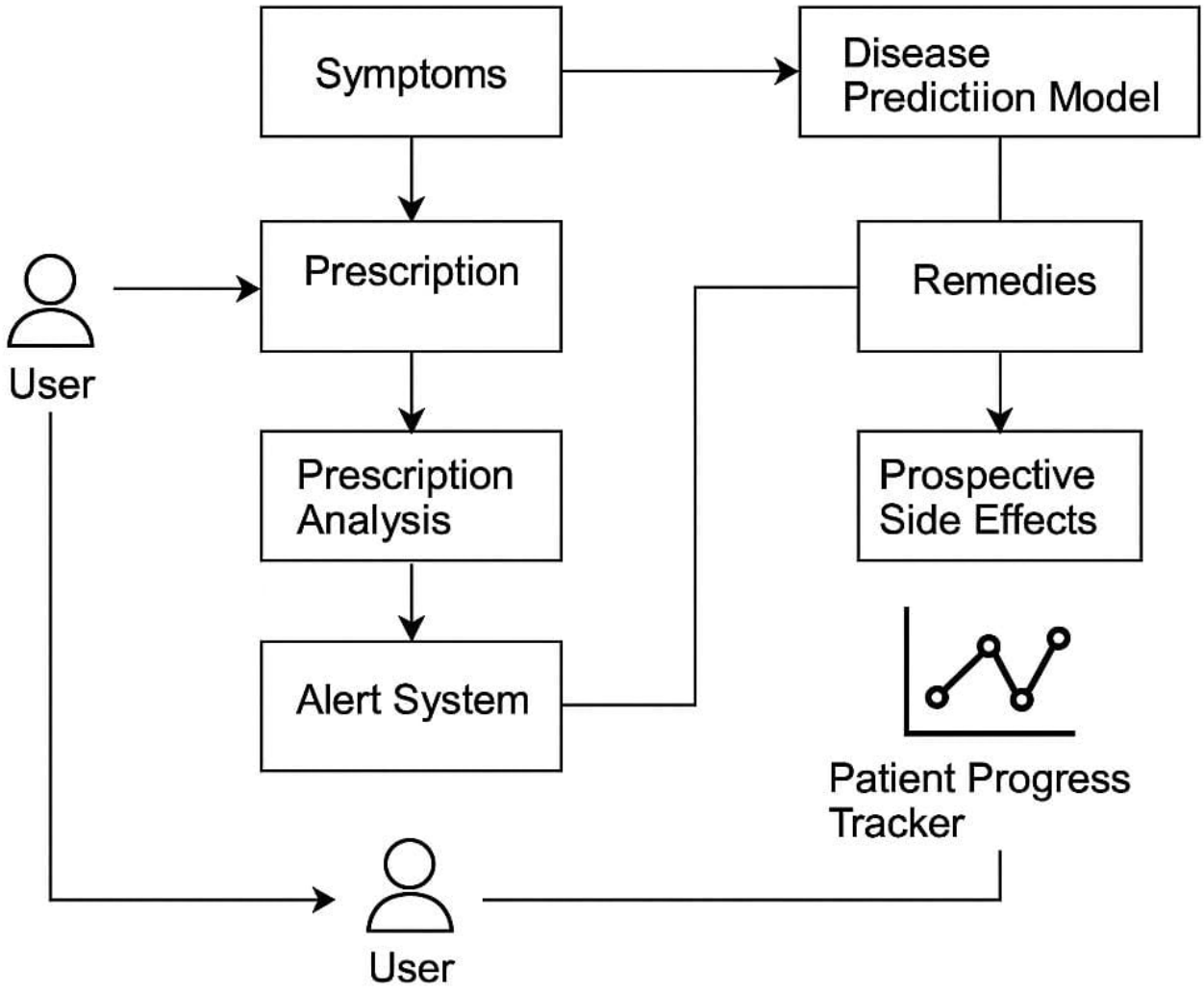

Fig 5.1 : System Architecture

5.2 Mathematical Model

Let the system be represented as:

S = {I, O, F, DD, NDD, M}

1. Input Set (I):

   The input consists of user-provided symptoms and optional user data.

   I = {s_1, s_2, ..., s_n, u}, where:

   $s_1, s_2, \ldots, s_n$ = symptoms selected or entered by the user

   u = user metadata (age, gender, medical history – optional)

2. Output Set (O):

The output consists of predicted disease(s) and corresponding remedies.

O = {d, r}, where:

- *d* = predicted disease
- *r* = recommended remedies (medicine, rest, lifestyle suggestions)

3. Function Set (F):

The functions used to convert input into output.

F = {f_1, f_2, f_3, f_4}

- *f_1:* Preprocessing function → f_1: I → I'
- *f_2:* Feature vector creation → f_2: I' → X
- *f_3:* Disease prediction (ML Model) → f_3: X → d
- *f_4:* Remedy recommendation based on disease → f_4: d → r

4. Deterministic Data (DD):

Known datasets and values used in the model.

DD = {Symptom-Disease Dataset, Remedy Mapping Table}

5. Non-Deterministic Data (NDD):

Dynamic or user-specific values not fixed in advance.

NDD = {User symptoms, User feedback, Environmental factors}

6. ML Model (M):

Model used for disease classification (e.g., Naive Bayes, Decision Tree, Random Forest, etc.)

M: X → d, where

- *X* = vector of input features
- *d* = predicted disease

5.3 Data Flow Diagrams (DFD)

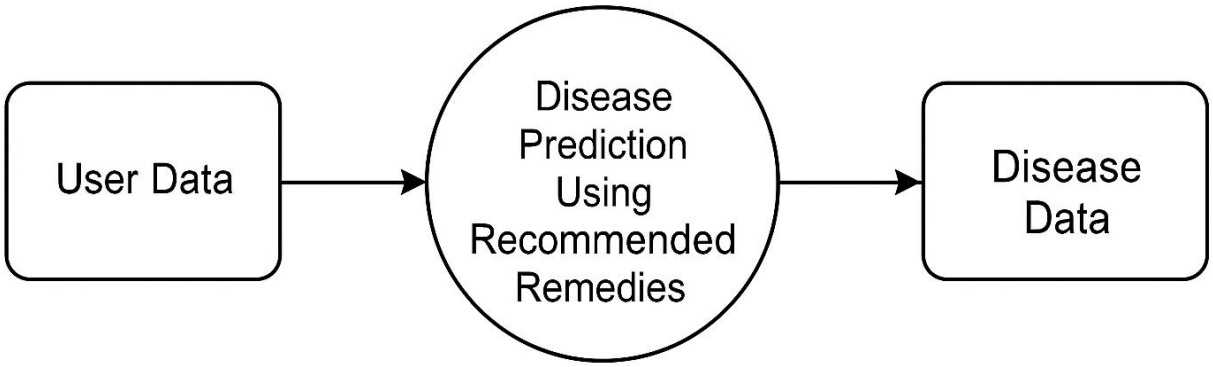

Fig 5.3.1 **:-** Data Flow diagram level 0

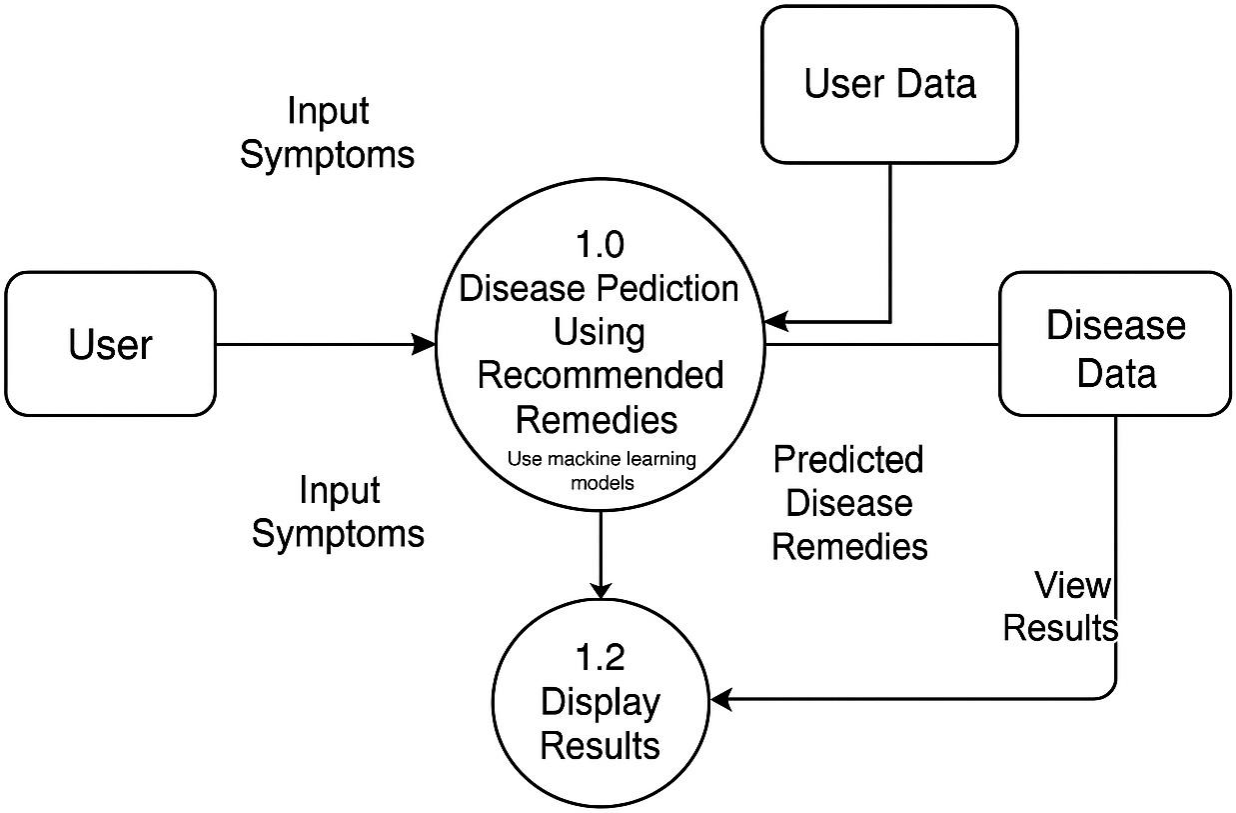

Figure 5.3.2: Data Flow diagram 1 level

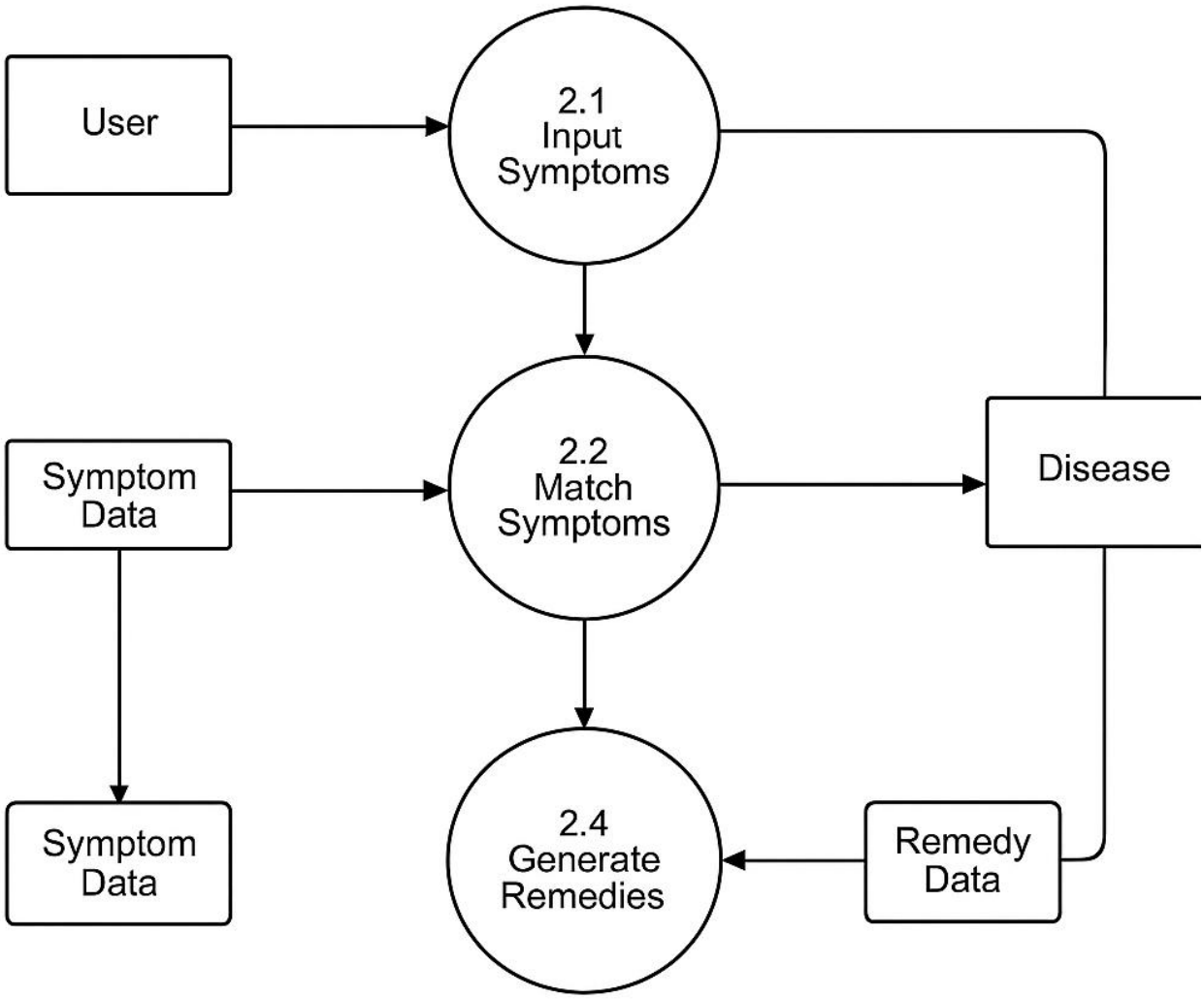

Figure 5.3.3: Data Flow Diagram 2 level

In Data Flow Diagram, we Show that flow of data in our system in DFD0 we show that base DFD in which rectangle present input as well as output and circle show our system, In DFD1 we show actual input and actual output of system input of our system is text or image and output is rumor detected like wise in DFD 2 we present operation of user as well as admin.

5.4 Entity Relationship Diagram (ERD)

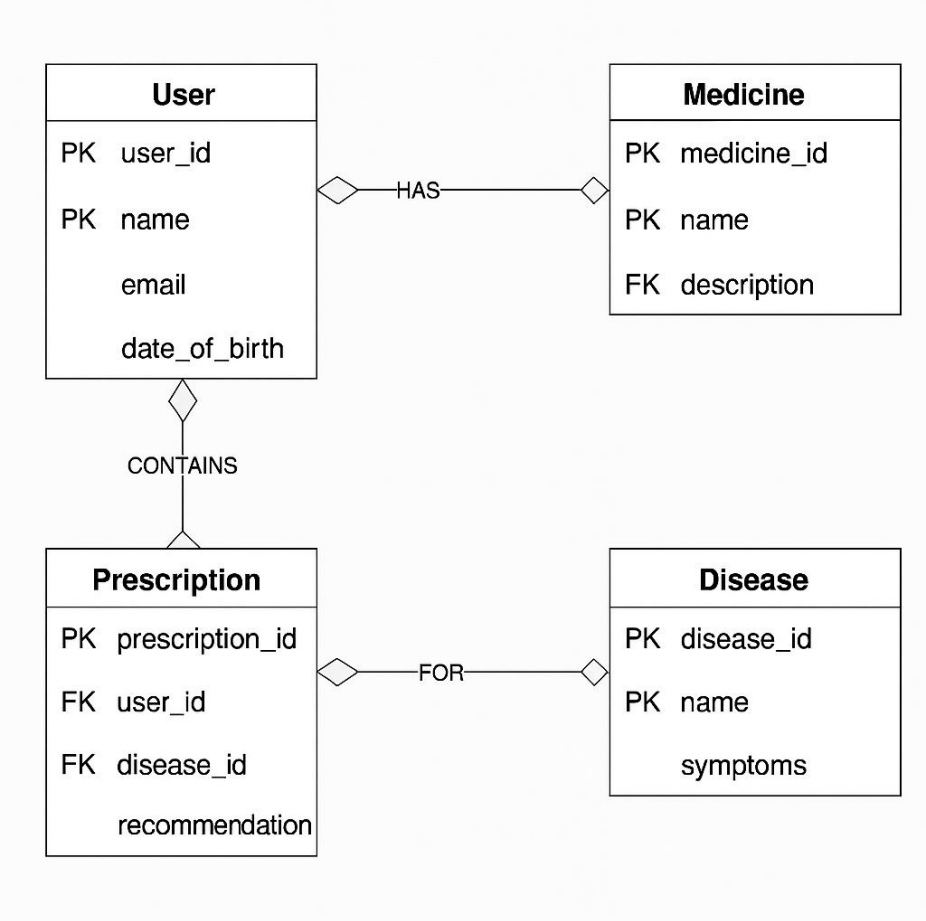

Fig 5.4 :- ER diagram

5.5 UML Diagrams

Represent system interaction and architecture.

5.5.1 Use Case diagram for user and system interaction

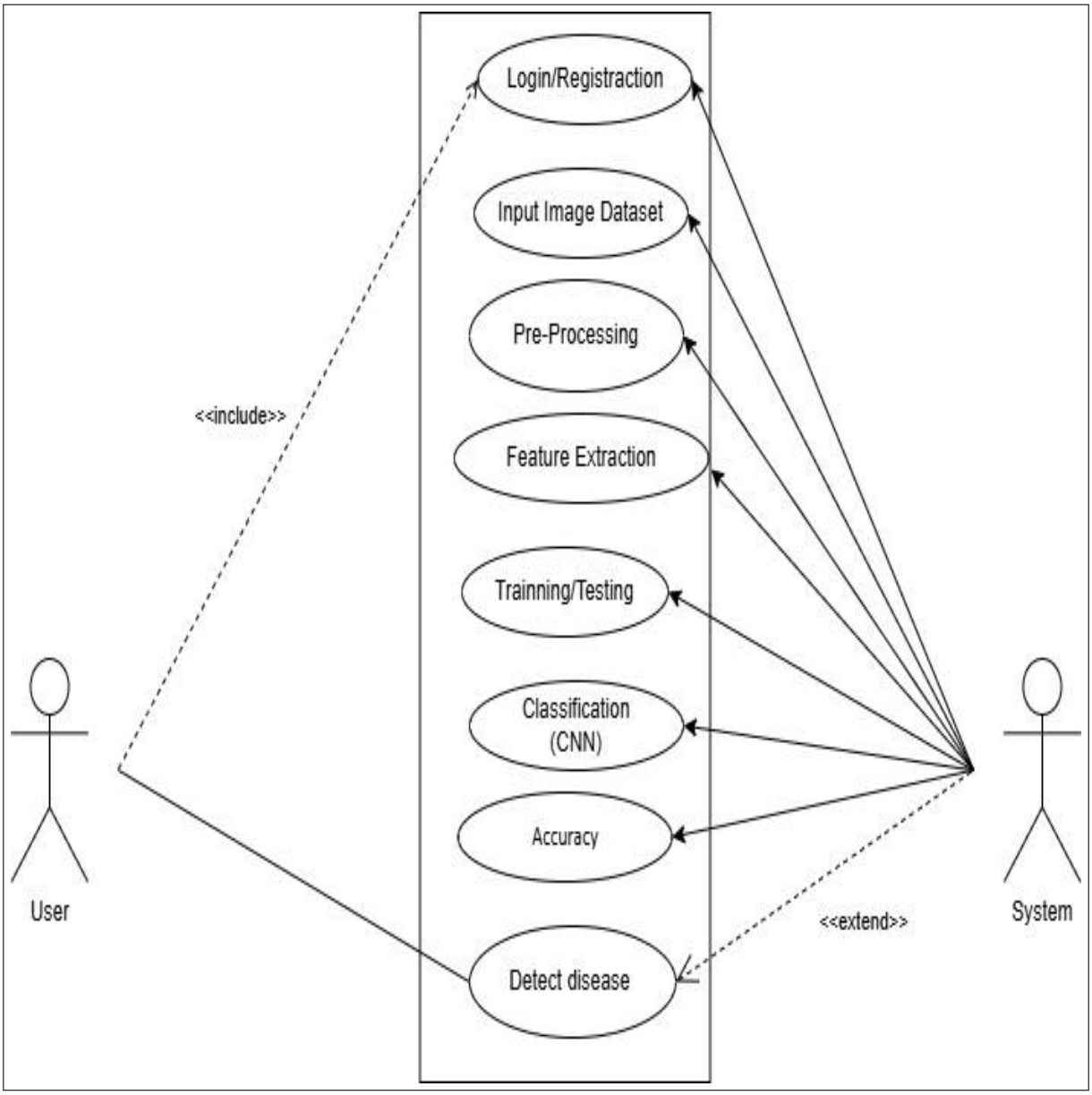

Fig 5.5.1 :- Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

5.5.2 Activity diagram showing disease prediction

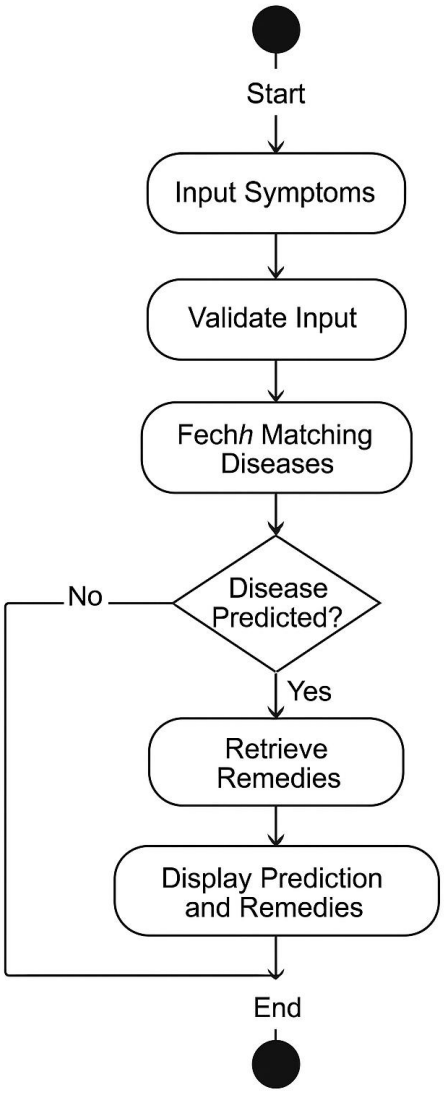

Fig 5.5.2 :- Activity Diagram

Activity diagrams are graphical representations of workflows of step wise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control they can also include elements showing the flow of data between activities through one or more data stores.

5.5.3 Class diagram for classes

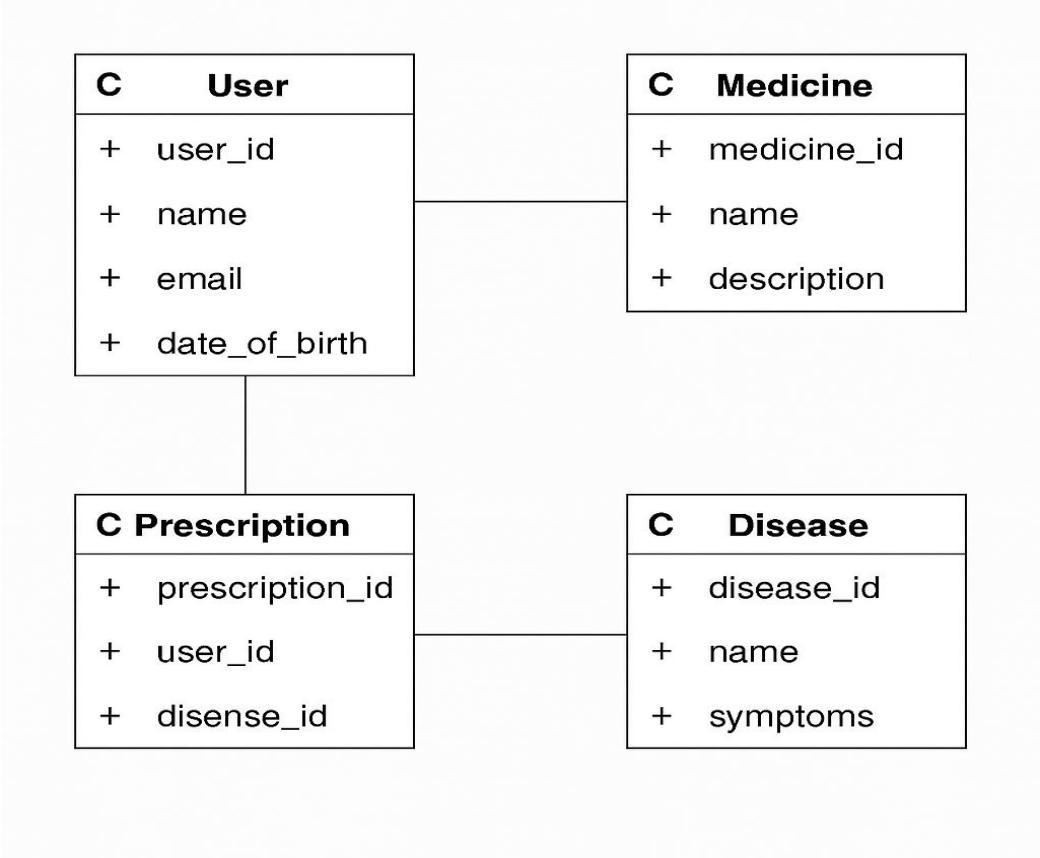

Fig 5.5.3 :- Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. [1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed

5.5.4 Sequence Diagram for user and system interaction

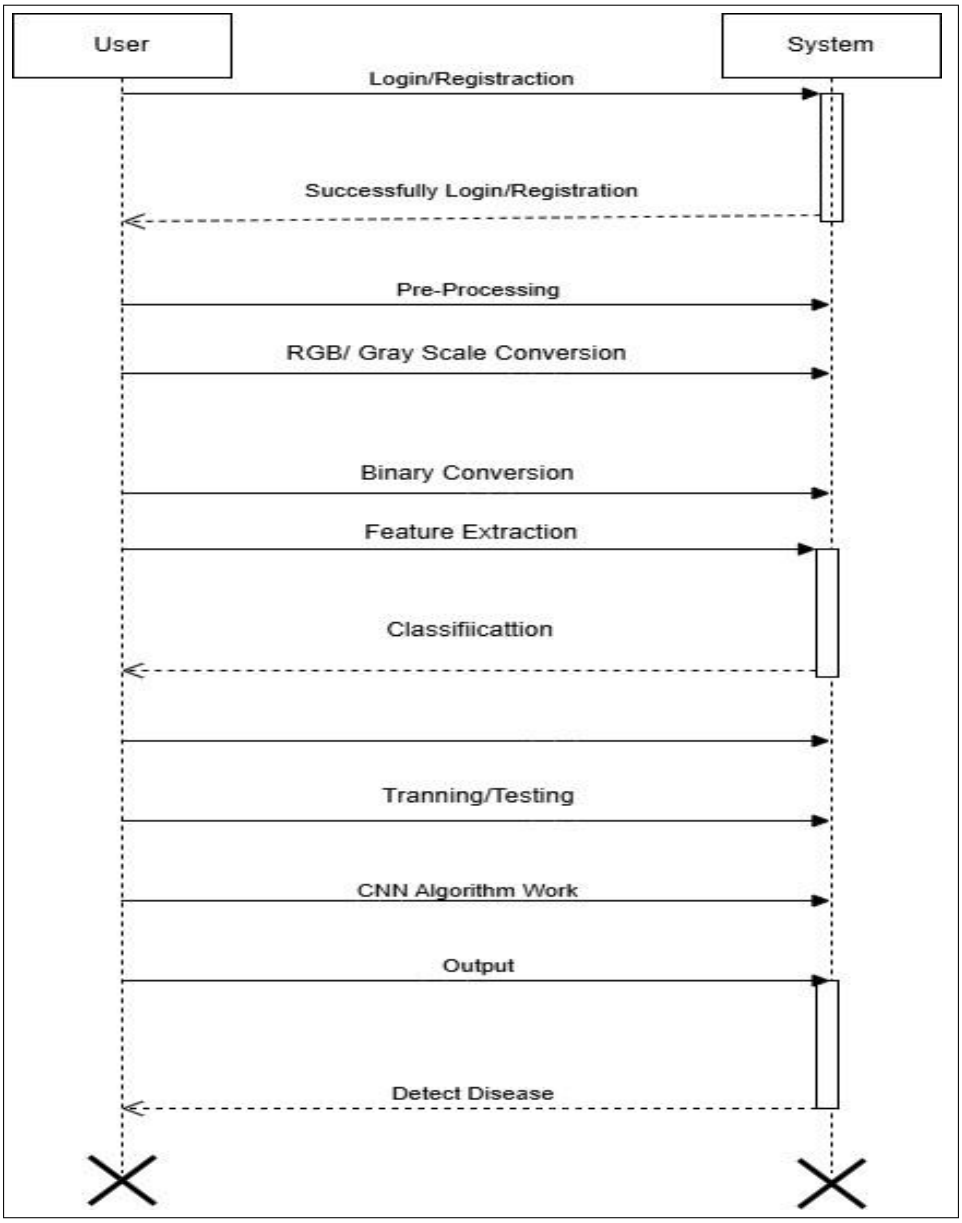

Fig 5.5.4 :- Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

5.5.5 State diagram for state to detect disease

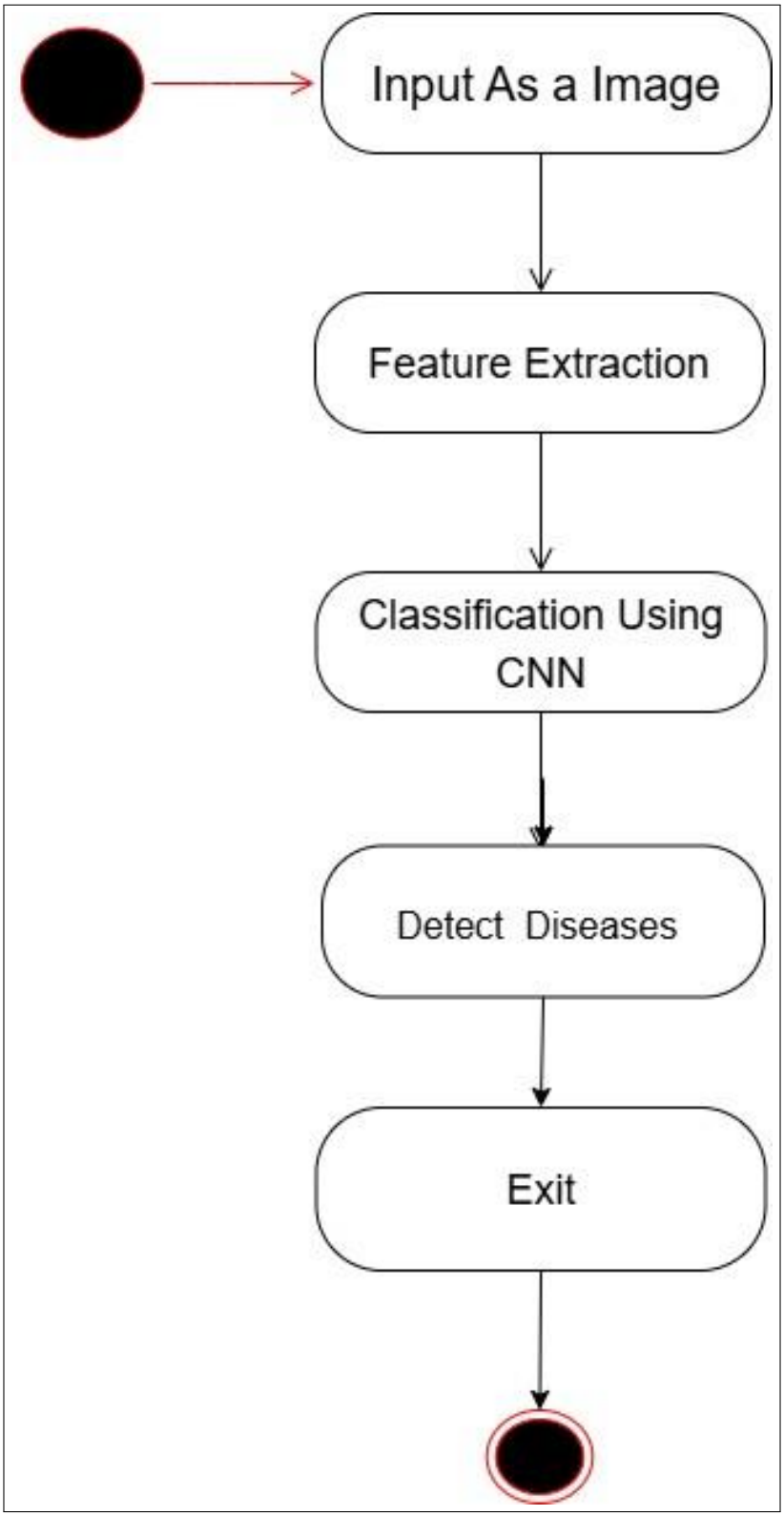

Fig 5.5.5 :- State Diagram for detect disease

5.5.6 Component diagram

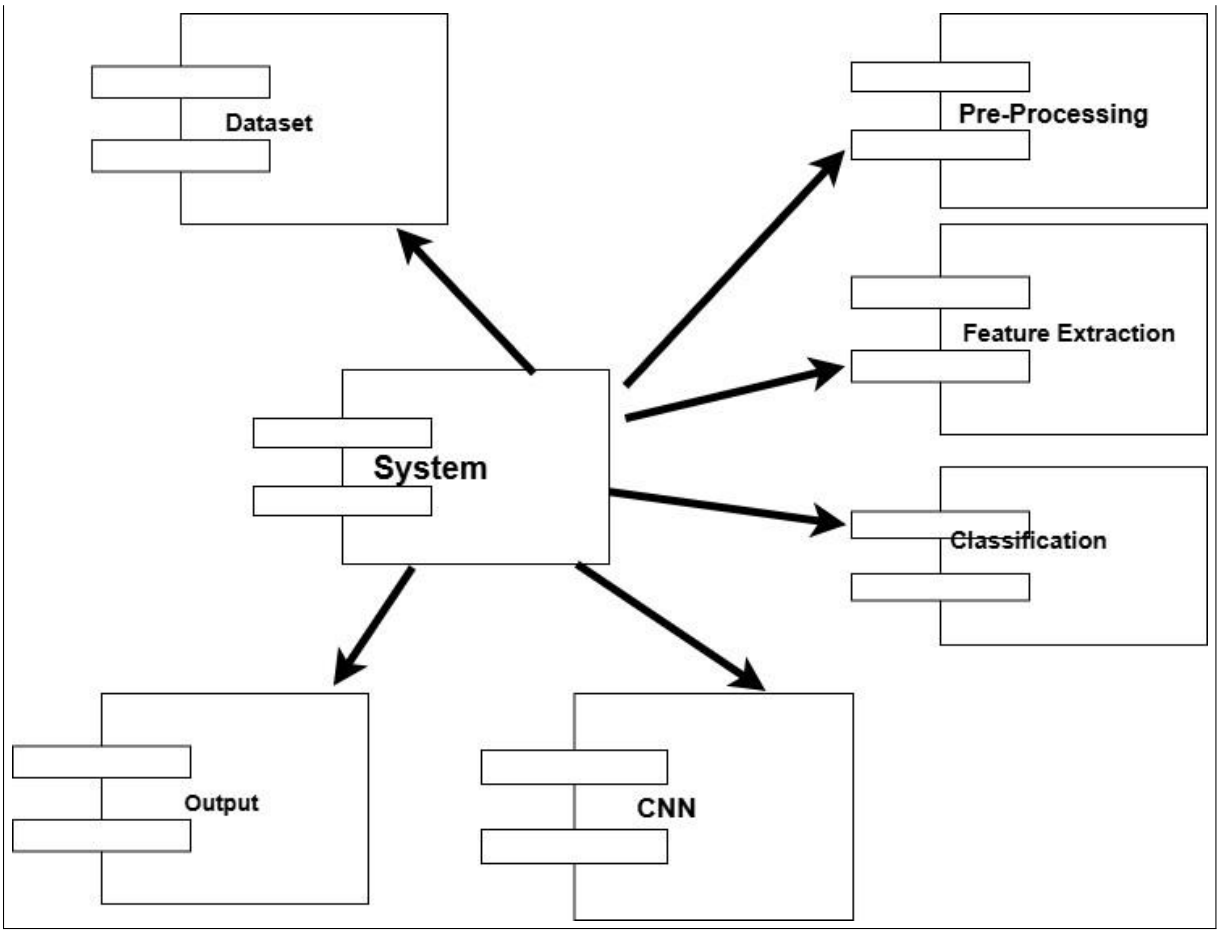

Fig 5.5.6 :- Component Diagram

5.5.7 Object Diagram

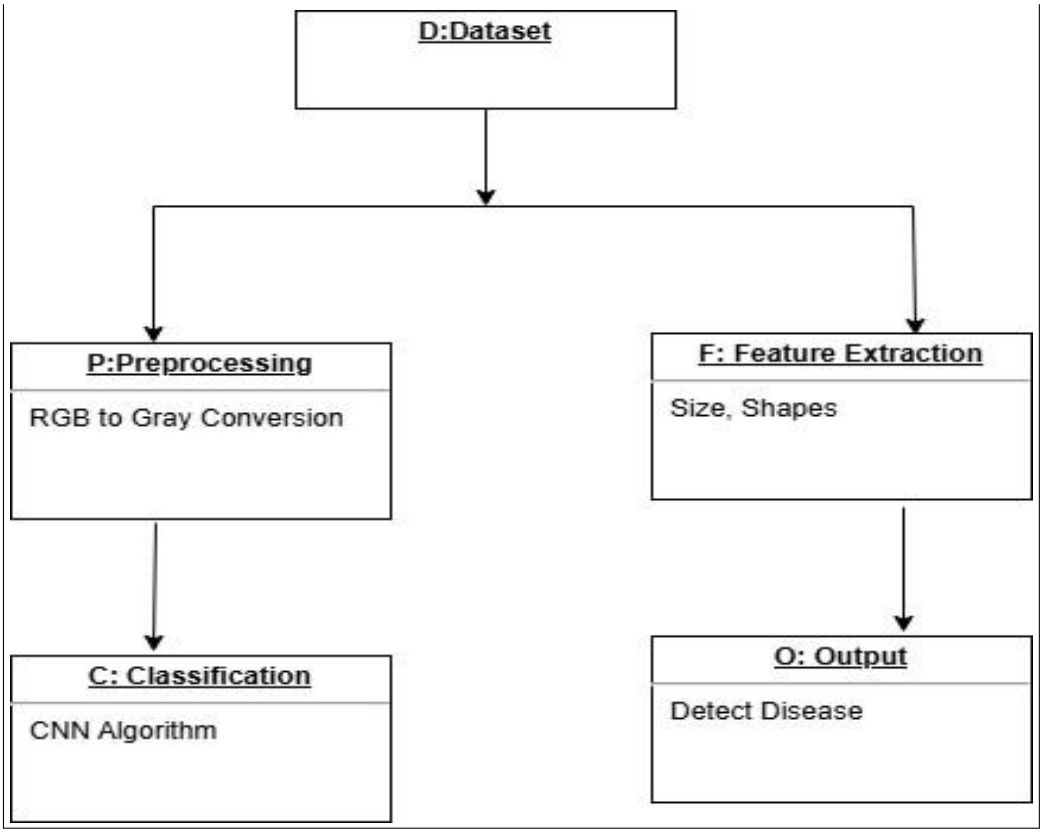

Fig 5.5.7 :- Object Diagram

5.5.8 Communication

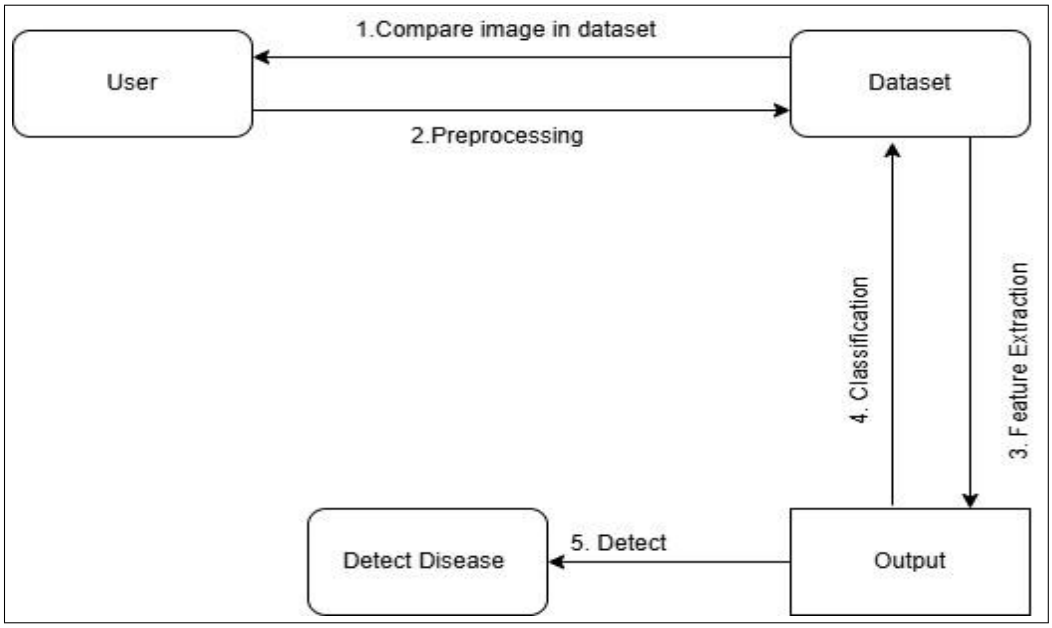

Fig 5.5.8 :- Communication Diagram

5.5.9 Deployment Diagram

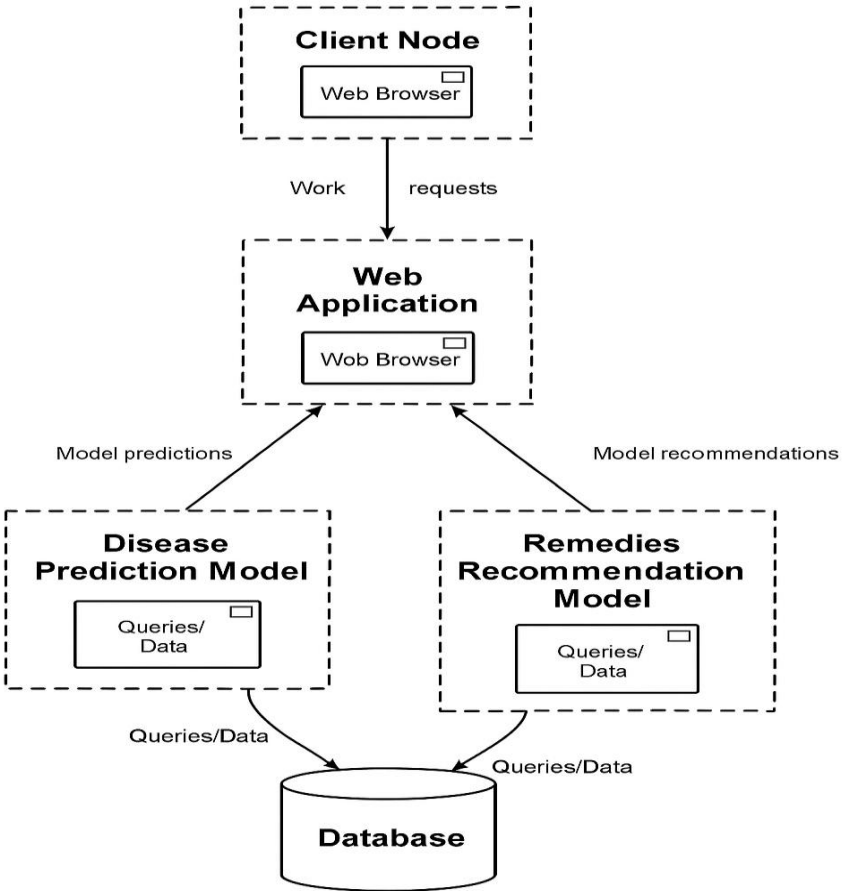

Fig 5.5.9 :- Deployment Diagram

CHAPTER 6

PROJECT IMPLEMENTATION

6 PROJECT IMPLEMENTATION

This section illustrates the process of developing, implementing, and testing the system. It explains the major modules, technologies, and algorithms involved in the Disease Prediction Using Recommended Remedies.

6.1 Overview of Project Modules

This subsection presents the various modules that together form the disease prediction using recommended remedies:

Division of logical components and assignments.

- User Interface Module
  - Function: Frontend interface for user input and result display.
  - Components: Web forms, file upload (for medical reports), input symptoms.
  - Technology: Could be built using HTML/CSS, JS, or integrated in a Python web framework like Flask/FastAPI.
- Symptom Analysis Module
  - Function: Extracts and processes symptoms from either user input or OCR output.
  - Processing: Tokenization, symptom matching, validation.
  - Integration: Works closely with the disease prediction model.
- Disease Prediction Module
  - Function: Uses ML/AI model to predict possible diseases based on symptoms.
  - Files: disease_predictor.py
  - Process:
    - Loads trained ML model
    - Preprocesses input
    - Outputs predicted disease(s)
- Routing/API Module
  - Function: Connects frontend with backend; handles input and output routing.
  - Files: routes.py, app.py
  - Endpoints: May include /predict, /upload, /recommend, etc.
- Data Models & Validation Module
  - Function: Defines request and response models using schema validation.

- Files: models.py
- Use Case: Ensures correct data formats and types for API transactions.

- Configuration & Deployment Module
  - Function: Stores app configuration, secrets, environment variables.
  - Files: .replit, om.env, pyproject.toml
  - Use Case: Smooth deployment and secure configuration.

6.2 Tools and Technologies Used

This project was built with the following tools and frameworks:

1) **Python**
   - Core programming language for the entire backend, ML model.
2) **FastAPI**
   - Web framework used to build the REST API endpoints for user input, predictions, and remedies.
3) **Pydantic**
   - Used with FastAPI for defining and validating data models (input/output schemas).
4) **Uvicorn**
   - ASGI server used to run the FastAPI application efficiently.
5) **scikit-learn**
   - Machine learning library used for building and using the disease prediction model.
6) **Pandas**
   - Used for handling tabular data and preprocessing before feeding into ML models.
7) **NumPy**
   - Supports numerical operations and array manipulations needed during prediction.
8) **joblib / pickle**
   - Used for model serialization and loading trained models at runtime.
9) **pytesseract**
   - Python wrapper for Tesseract to extract text from uploaded medical images.

10) **OpenCV**
    - Image processing library used to clean, resize, or adjust images before OCR.

11) **python-dotenv / .env files**
    - For managing configuration variables like API keys or environment settings.

12) **Replit**
    - Cloud-based development environment used for coding and hosting the project.

13) **Git**
    - Version control system used to track code changes and manage project history.

14) **pyproject.toml**
    - Used for dependency and project configuration in modern Python projects.

6.3 Algorithm Details

This sub-section outlines the fundamental algorithms employed within the project to detect diseases.

6.3.1 Algorithm 1 – Symptom Vectorization Algorithm

- Algorithm Type: Binary Vector Mapping
- Purpose: Converts list of symptoms into a fixed-length binary vector to feed into ML models.
- Working:
    - A master list of known symptoms is maintained.
    - For any given input, the presence (`1`) or absence (`0`) of each symptom is recorded.
    - Example: Input = [‘fever’, ‘cough’]
      Master list = [‘fever’, ‘cold’, ‘cough’, ‘headache’]
      Output vector = [1, 0, 1, 0]

6.3.2 Algorithm 2 – Machine Learning Algorithm for Disease Prediction

- Most Likely Algorithm Used:
    - Decision Tree Classifier
    - (Possibly also tested: Random Forest, Naive Bayes, KNN)

- Details (for Decision Tree):
    - Type: Supervised Classification Algorithm
    - Purpose: Predict the most probable disease based on symptom vector.
- Training Phase:
    - Input: Symptom vectors (X) and disease labels (Y)
    - Output: Trained model saved using joblib or pickle
- Prediction Phase:
    - Input: User's symptom vector
    - Output: Predicted disease label (e.g., "Malaria")

6.3.4 Algorithm

1. **Input collection**:

   Accept symptoms as user input (via form or UI).
2. **Data preprocessing**:

   Convert symptoms to appropriate feature vectors.
3. **Model prediction**:

   Use a trained machine learning model (e.g., Decision Tree, Random Forest) to predict the disease based on the input symptoms.
4. **Remedy recommendation**:

   Fetch precautions corresponding to the predicted disease.
5. **Output results**:

   Display predicted disease.

   Display precautions.

6.4 Flowchart Of Disease Prediction Using Recommended Remedies

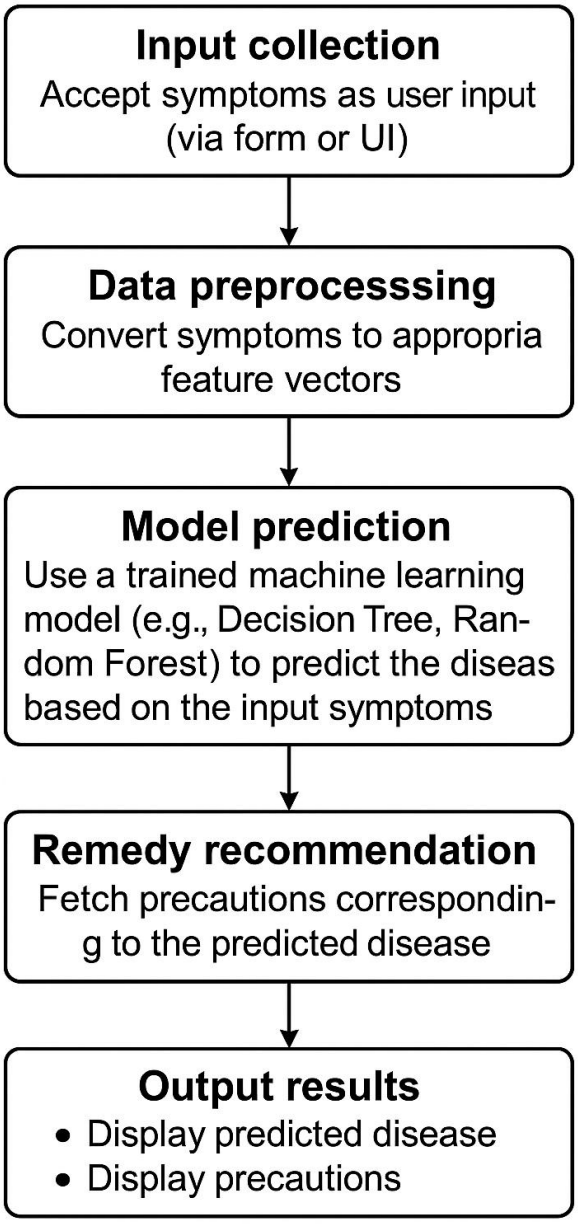

Fig 6.4:- Flowchart

CHAPTER 7

SOFTWARE TESTING

7 SOFTWARE TESTING

Testing was conducted to ensure the system's accuracy, reliability, and performance in various real-world scenarios.

7.1 Types of Testing

Following testing mechanisms were used:

- **Unit Testing**
  - Tests individual functions or modules like symptom input validation, disease prediction function, remedy retrieval, etc.
- **Integration Testing**
  - Verifies the interaction between modules (e.g., symptom input → prediction model → remedy recommendation).
- **System Testing**
  - End-to-end testing of the entire application including UI, database, prediction engine, and output results.
- **Functional Testing**
  - Ensures that features such as "Enter Symptoms", "Predict Disease", "Recommend Remedies" work as specified.
- **Non-Functional Testing**
  - **Performance Testing:** Checks how fast prediction and recommendation modules respond.
  - **Usability Testing:** Verifies the UI is user-friendly for both technical and non-technical users.
- **Regression Testing**
  - Ensures that adding new symptoms, diseases, or remedies doesn't break existing functionality.

7.2 Test Cases and Results

Some sample test cases are given below:

- Test Case ID: TC_Disease_01

  Feature: Disease Prediction

  Input: ["metformin", "glibenclamide"]

Expected Result: Predicted Disease: Diabetes; Remedies: Advise low-sugar diet, regular exercise, medication adherence

Actual Result: Predicted Disease: Diabetes; Remedies suggested correctly

Status: Pass

- Test Case ID: TC_Disease_02

  Feature: Disease Prediction

  Input: ["salbutamol", "budesonide"]

  Expected Result: Predicted Disease: Asthma; Remedies: Inhaler use, avoid allergens, regular checkups

  Actual Result: Predicted Disease: Asthma; Remedies suggested correctly

  Status: Pass

- Test Case ID: TC_Disease_03

  Feature: Disease Prediction

  Input: ["diosmin", "lidocaine cream"]

  Expected Result: Predicted Disease: Haemorrhoids; Remedies: Topical cream use, fiber-rich diet

  Actual Result: Predicted Disease: Haemorrhoids; Remedies suggested correctly

  Status: Pass

- Test Case ID: TC_Disease_04

  Feature: Multi-disease Detection

  Input: ["metformin", "montelukast"]

  Expected Result: Predicted Diseases: Diabetes, Asthma; Remedies for both

  Actual Result: Both diseases detected; separate remedies suggested

  Status: Pass

- Test Case ID: TC_Disease_05

  Feature: Unknown Medicine Handling

  Input: ["xyzmed"]

  Expected Result: Prompt: Medicine not recognized

  Actual Result: Prompt displayed correctly

  Status: Pass

CHAPTER 8

RESULTS

8 RESULTS

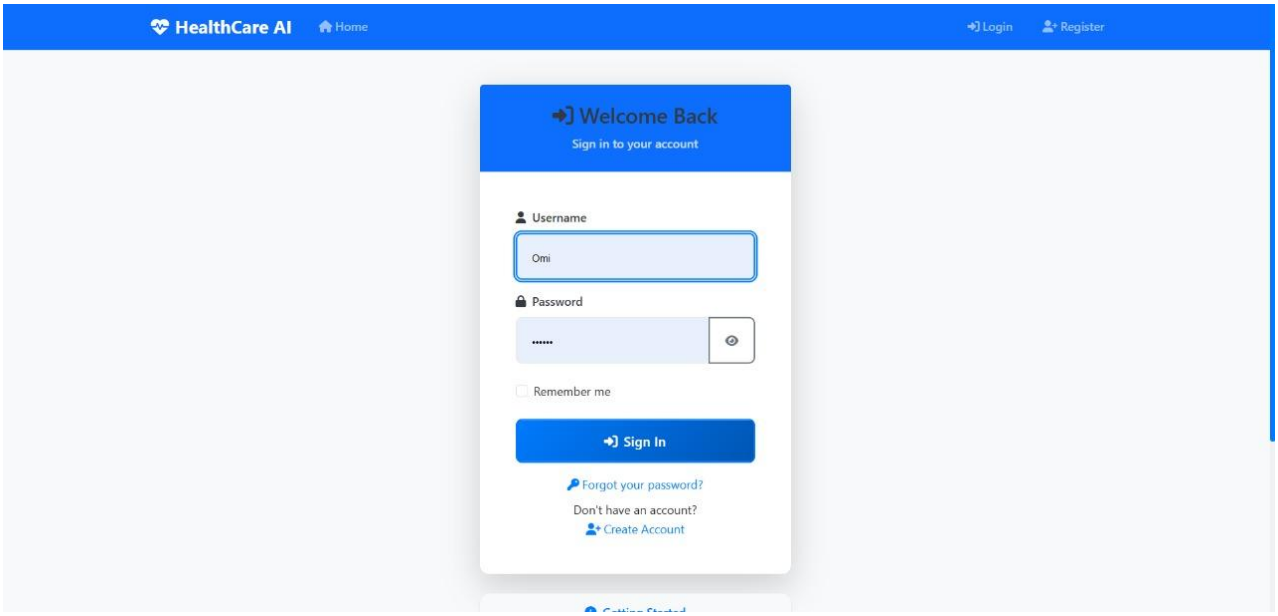

Fig 8.1:- Login Page

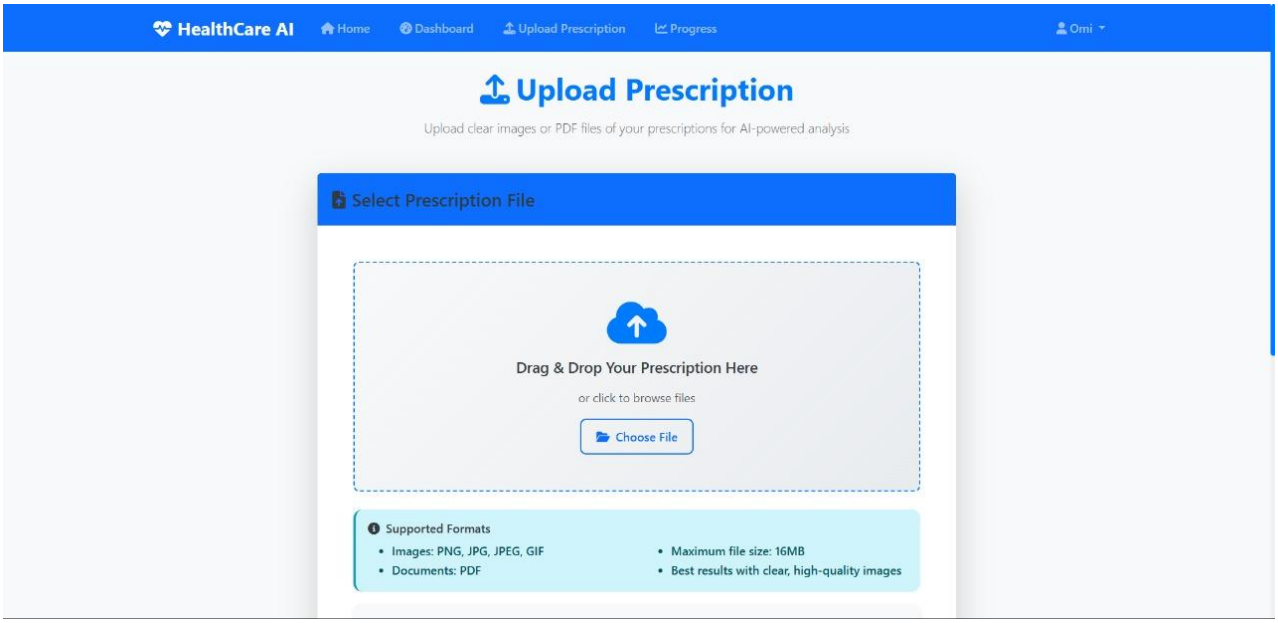

Fig 8.2:- Upload Prescription Page

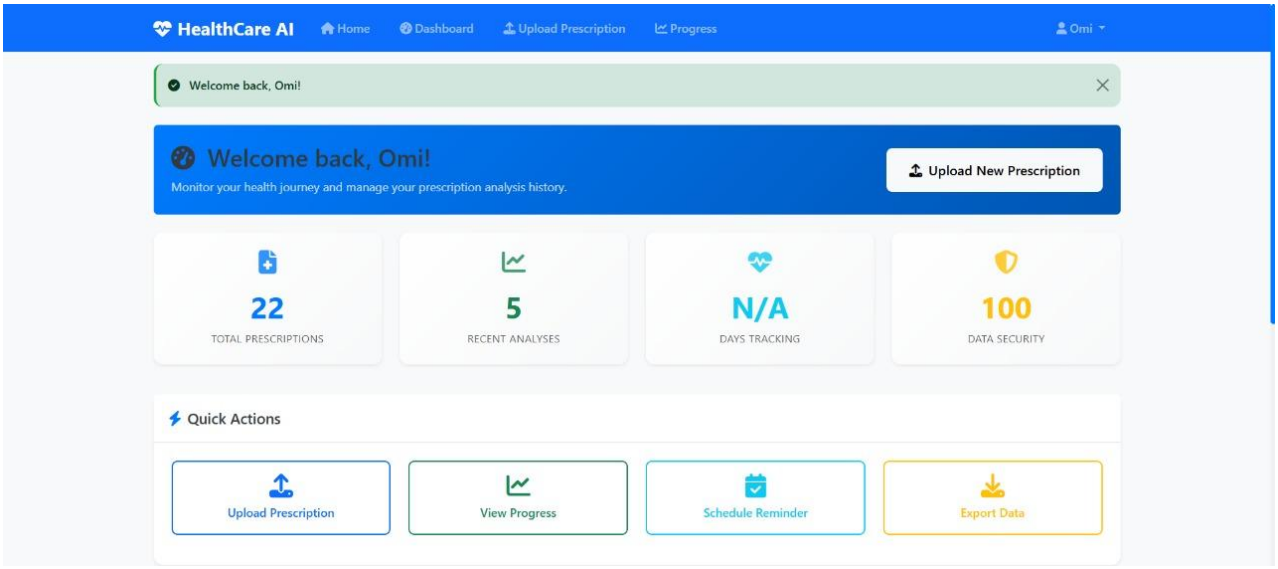

Fig 8.3:- Dashboard Page

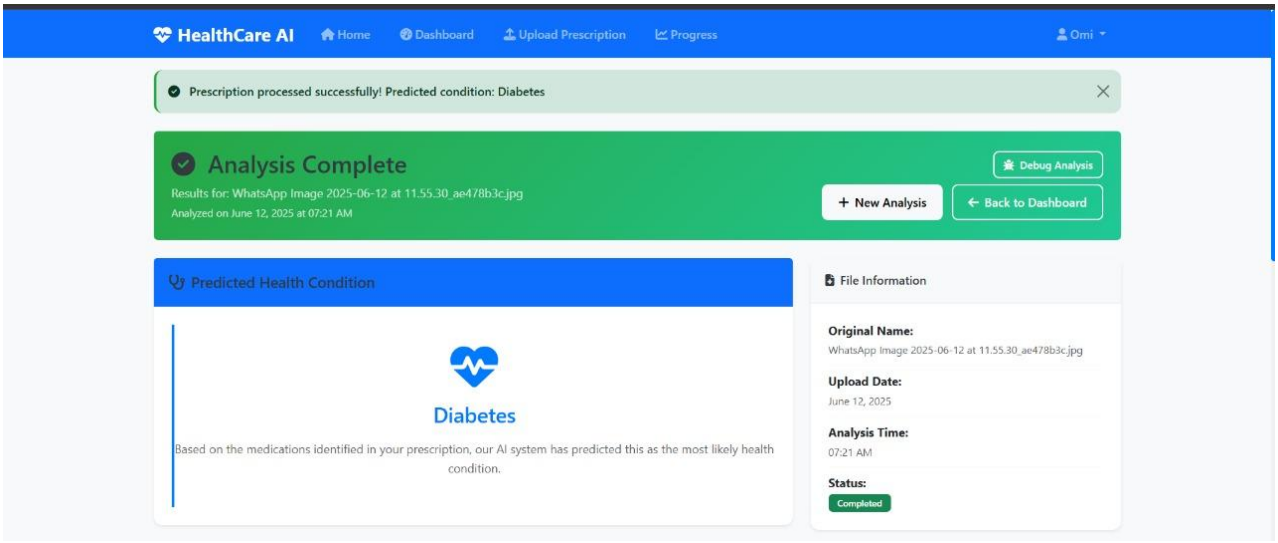

Fig 8.4:- Disease Predicted

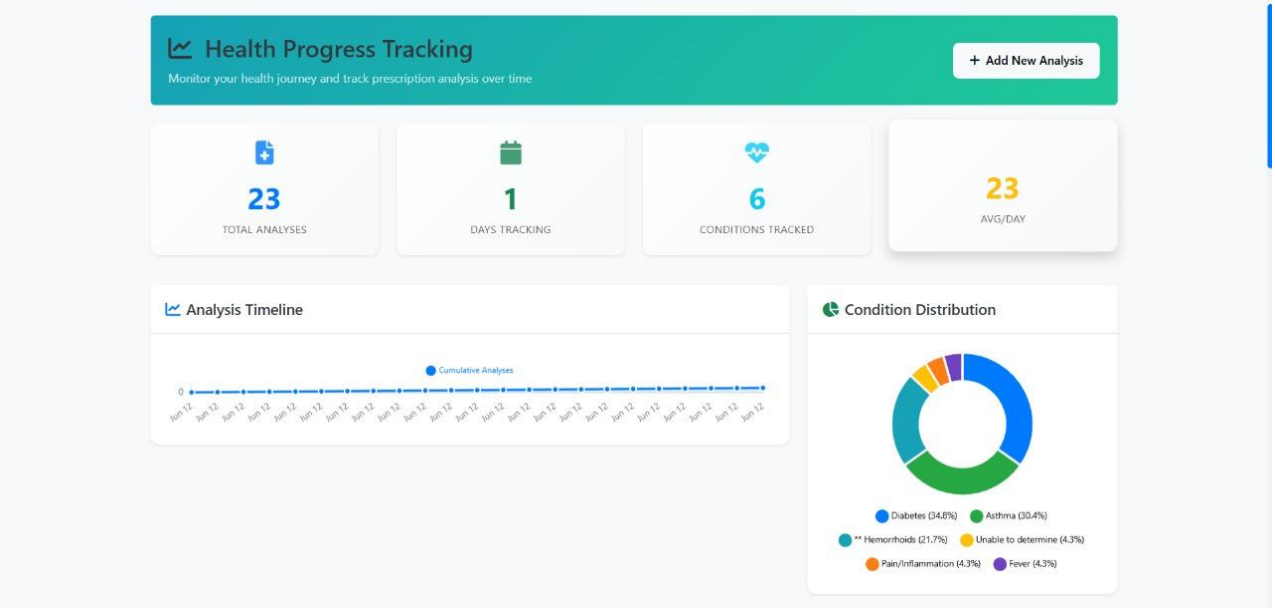

Fig 8.5:- Analysis graph

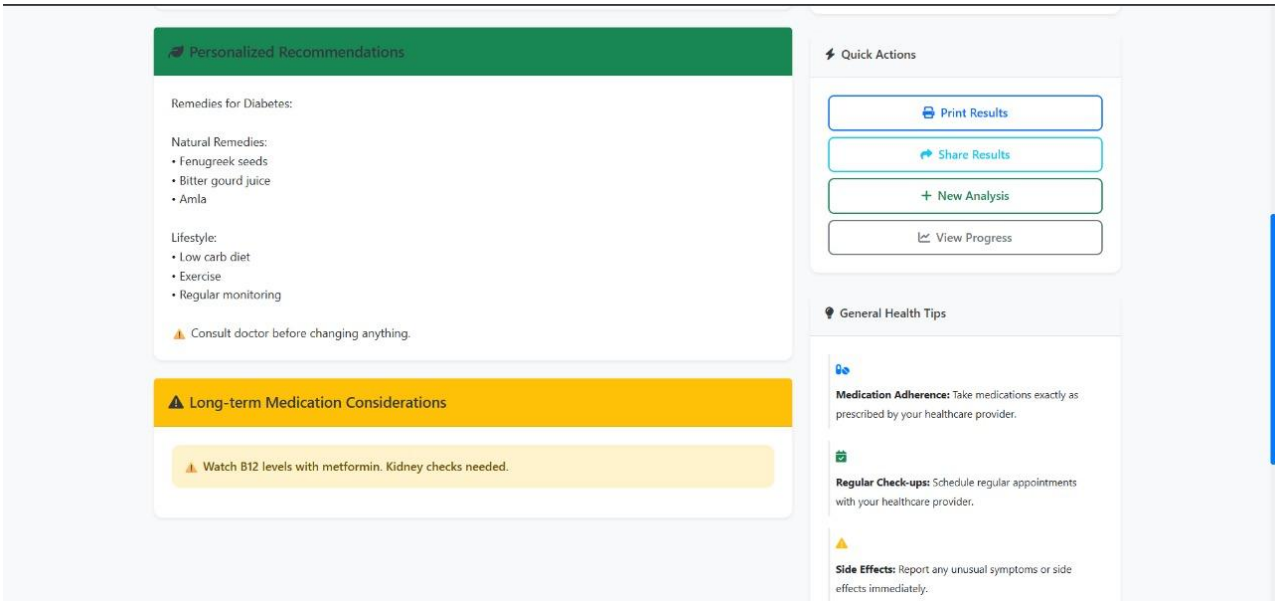

Fig 8.6:- Recommendations

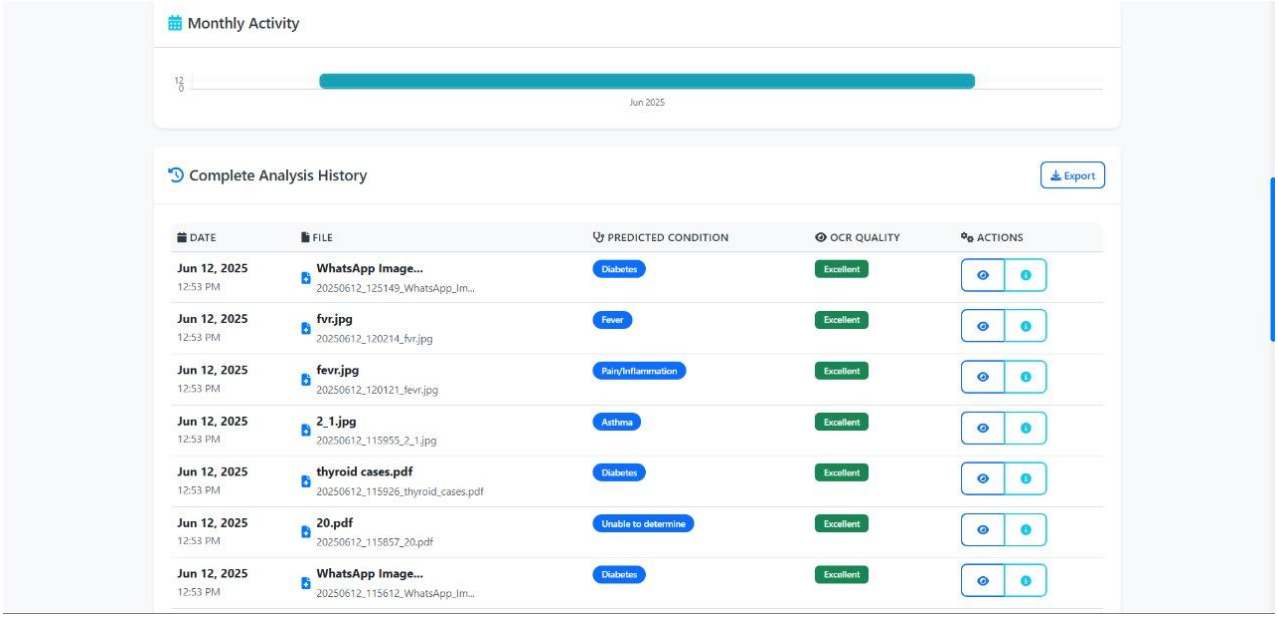

Fig 8.7:- History of patient

CHAPTER 9

CONCLUSION

9 CONCLUSION

9.1 Conclusion

The long term Disease detection system using medical prescriptions shows promising potential for improving healthcare delivery by leveraging deep learning techniques. By analyzing prescription data, the system can identify patterns and correlations between prescribed medications and potential tong term Diseases, assisting healthcare professionals in diagnosing and suggesting appropriate treatments. The model can quickly analyze large volumes of prescription data, enabling faster diagnosis and treatment recommendations. The integration of historical prescription data with deep learning algorithms enhances the accuracy of long term Disease detection by recognizing patterns not easily identifiable by human clinicians.

9.2 Future Work

Ensuring high quality standardized prescription data is essential for the system accuracy. Future work should focus on developing methods to handle incomplete or erroneous data, as Well as ensuring uniformity across different healthcare system.Future research could explore Seamless integration with EHR systems, enabling a more comprehensive view of patient Health data, which could improve the long term Disease detection process and treatment Recommendations. Incorporating information on drug interations and potential side effect Could help refine the diagnosis and avoid potential complications in treatment plans.

9.3 Applications

1. **Online Health Self-Diagnosis Portals**
    - Helps users perform a basic health assessment by inputting symptoms and receiving likely disease predictions along with suggested remedies.
2. **Primary Healthcare Assistance in Rural Areas**
    - Assists communities with limited access to medical professionals by providing immediate guidance based on common symptoms and recommended treatments.
3. **Mobile Health (mHealth) Applications**
    - Can be integrated into mobile apps for on-the-go users, offering health insights without the need to visit clinics unless necessary.
4. **Clinical Decision Support System (CDSS)**
    - Supports doctors in making faster decisions by cross-verifying symptoms and diseases with a system-generated prediction.

5. **Health Chatbots and Virtual Assistants**
    - Powers intelligent chatbots that interact with users, understand symptoms, predict diseases, and recommend OTC remedies or medical advice.
6. **Educational Tools for Medical Students**
    - Helps students learn symptom-disease relationships and enhances diagnostic skills through interactive case-based scenarios.
7. **Telemedicine Platforms**
    - Enhances online doctor consultations by providing a preliminary diagnosis and treatment path based on patient-reported symptoms.
8. **Healthcare Analytics and Data Mining**
    - Aggregated prediction data can be analyzed for outbreak tracking, common illness trends, and public health planning.
9. **Pharmacy Recommendation System**
    - Suggests suitable over-the-counter medicines or home remedies for common conditions, potentially integrated into online pharmacy platforms.
10. **Health Monitoring for Wearables**
    - When combined with wearable health data (like temperature, pulse, etc.), it can improve accuracy in symptom analysis and predictions.

ANNEXURE A

APPENDIX A

A1 PROBLEM STATEMENT FEASIBILITY ASSESSMENT USING, SAT- ISFIABILITY ANALYSIS AND NP HARD, NP-COMPLETE OR P TYPE USING MODERN ALGEBRA AND RELEVANT MATHEMATICAL MODELS.

Problem Statement:

Design a system that predicts probable diseases from given symptoms and recommends appropriate remedies using an efficient algorithm.

When solving problems, we must decide the difficulty level of our problem. There are three types of classes provided for that. These are as follows:

P Class

NP Complete Class

NP hard Class

Feasibility using Satisfiability (SAT):

Representation:

- Let each symptom be represented as a Boolean variable.
- Disease rules (symptom sets that indicate a disease) are expressed as logical clauses.

Feasibility:

- We can model disease prediction as a SAT problem, where the goal is to check if there exists a combination of symptoms that satisfies the disease conditions.
- Modern SAT solvers can efficiently process such boolean logic, even at scale.
- So, disease prediction is reducible to SAT, proving the feasibility of the logical approach.

Complexity (P, NP, etc):

- **P Class (Polynomial Time):**
  - Basic rule-based disease matching (lookup tables or decision trees) is in P, since it can be done in polynomial time.
- **NP-Complete Aspects:**
  - If you're selecting the optimal remedy plan from a large space of possible remedies under constraints (like allergies, medication conflicts, availability), this becomes:
  - A constraint satisfaction or **s**ubset selection problem.

- This can be reduced to the SAT or Knapsack problem, making it NP-Complete.

- **NP-Hard:**
  - If multiple diseases interact or co-occur, and you're finding optimal multi-disease remedy combinations, this may require solving:
  - A variant of multi-objective optimization, which is NP-Hard.

Use Of Math:

a) Set Theory & Logic:

- Symptoms (S), Diseases (D), Remedies (R) modeled as sets:
  - D={d1,d2,...,dn}
  - S={s1,s2,...,sm}
  - R={r1,r2,...,rk}
- Disease prediction: f:S→D , Remedy mapping: g:D→R

b) Graph Theory:

- Bipartite graph:
  - Nodes: Symptoms ↔ Diseases ↔ Remedies
  - Paths represent dependency and flow.

c) Boolean Algebra:

- Used for SAT modeling and expression of logical rules for symptoms → disease mapping.

d) Matrix Algebra:

- Symptom–Disease incidence matrix AijA_{ij}Aij, where:
  - Aij=1 if symptom si is associated with disease dj

Conclusion:

The project is computationally and mathematically feasible. It can be effectively modeled using Boolean logic and analyzed through SAT-Satisfiability, where disease prediction rules can be expressed in logical clauses and efficiently evaluated using SAT solvers.

ANNEXURE B

APPENDIX B

1) Paper Title: Disease Prediction Using Recommended Remedies.
   1. Name of the Conference/Journal where paper submitted:
      - Sies School Of Business
   2. Paper accepted/rejected: Accepted.
   3. Review comments by reviewer: No.
   4. Corrective actions if any: No

2) Paper Title: Disease Prediction Using Recommended Remedies.
   1. Name of the Conference/Journal where paper submitted:
      - International Journal of Research And Analytical Reviews(IJRAR).
   2. Paper accepted/rejected: Accepted.
   3. Review comments by reviewer: No.
   4. Corrective actions if any: No

SIES School of Business Studies
RISE WITH EDUCATION

CERTIFICATE

This is to certify that **Harshada Warade**
of **PDEA's College of Engineering, Manjari**
has presented a paper entitled
Disease Prediction Using Recommended Remedies
in the"International Management Research Conference 2025" A Whole New World:
Business Landscape in the Emerging Era
organized by SIESSBS, Navi Mumbai
from 13th February to 15th February, 2025.

Dr. Swati Checker, Convener | Dr.Deepa Dixit, Director | Dr. Gunjan Hasijani, Convener

Fig :- Certificate 1

SIES School of Business Studies
RISE WITH EDUCATION

CERTIFICATE

This is to certify that **Deepali Thakur**
of **PDEA's College of Engineering, Manjari**
has presented a paper entitled
Disease Prediction Using Recommended Remedies
in the"International Management Research Conference 2025" A Whole New World:
Business Landscape in the Emerging Era
organized by SIESSBS, Navi Mumbai
from 13th February to 15th February, 2025.

Dr. Swati Checker, Convener | Dr.Deepa Dixit, Director | Dr. Gunjan Hasijani, Convener

Fig :- Certificate 2

SIES School of Business Studies
RISE WITH EDUCATION

CERTIFICATE

This is to certify that **Om Bankar**
of **PDEA's College of Engineering, Manjari**
has presented a paper entitled
Disease Prediction Using Recommended Remedies
in the"International Management Research Conference 2025" A Whole New World:
Business Landscape in the Emerging Era
organized by SIESSBS, Navi Mumbai
from 13th February to 15th February, 2025.

Dr. Swati Checker, Convener | Dr.Deepa Dixit, Director | Dr. Gunjan Hasijani, Convener

Fig :- Certificate 3

SIES School of Business Studies
RISE WITH EDUCATION

CERTIFICATE

This is to certify that **Aditya Shewale**
of **PDEA's College of Engineering, Manjari**
has presented a paper entitled
Disease Prediction Using Recommended Remedies
in the"International Management Research Conference 2025" A Whole New World:
Business Landscape in the Emerging Era
organized by SIESSBS, Navi Mumbai
from 13th February to 15th February, 2025.

Dr. Swati Checker, Convener | Dr.Deepa Dixit, Director | Dr. Gunjan Hasijani, Convener

Fig :- Certificate 4

Fig :- Certificate 5

Fig :- Certificate 6

Fig :- Certificate 7

Fig :- Certificate 8

ANNEXURE C

APPENDIX C

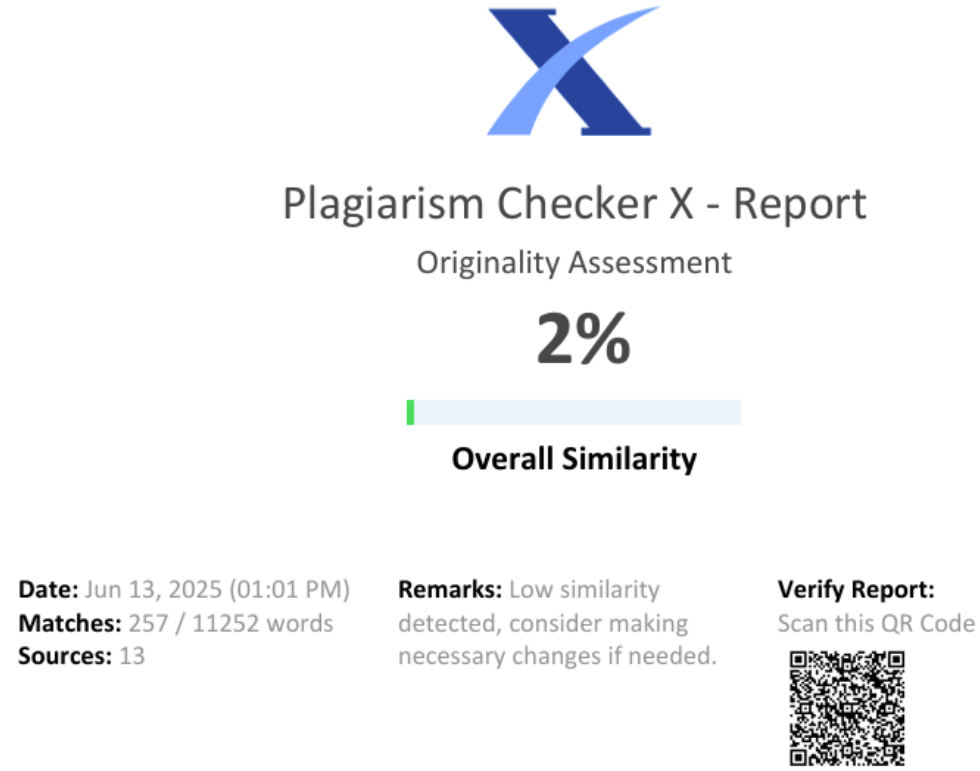

Fig:- Plagiarism Report

REFERENCES

Books

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[2] Tom M. Mitchell, *Machine Learning*, McGraw Hill, 1997

[3]Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, 7th ed., McGraw-Hill, 2012.

[4] Michael Sipser, *Introduction to the Theory of Computation*, 3rd ed., Cengage, 2012.

Journals/Conferences

[5] M. Kumari, S. Yadav, and R. Singh, "Medicine Suggestion Using Machine Learning," *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 5, pp. 256–260, 2021.

[6] Y. Wang, J. Liu, and H. Chen, "Drug Side Effect Prediction Based on Deep Learning and Chemical Graph Networks," *Computational Biology and Chemistry*, vol. 95, Art. no. 107587, 2021.

[7] A. Sharma, R. Agrawal, and S. Verma, "AI-based Comparative Study for Healthcare Diagnosis," *IEEE Access*, vol. 10, pp. 54231–54242, 2022.

[8] S. Patil and N. Desai, "Health Monitoring and Disease Prediction System Using Machine Learning," *International Journal of Computer Applications (IJCA)*, vol. 183, no. 12, pp. 12–17, 2021.

[9] R. Gupta, M. Jain, and K. Bansal, "Predicting Chronic Illness Using Ensemble Learning Models," *Journal of Biomedical Informatics*, vol. 139, p. 104215, Elsevier, 2023.

[10] P. Singh, A. Tiwari, and R. Kapoor, "Disease Prediction Using AI," *International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT)*, vol. 8, no. 2, pp. 134–138, 2022.

[11] K. Rani and A. Kumar, "AI-Based Medical Recommendation System Using Patient History," in *Lecture Notes in Computational Vision and Biomechanics*, Springer, vol. 37, pp. 145–154, 2021.

[12] V. Deshmukh, S. Gawande, and P. Kulkarni, "Prescription-Based Disease Detection Using OCR and Machine Learning," *Proceedings of the IEEE International Conference on Artificial Intelligence in Healthcare (ICAIH)*, pp. 92–98, 2023.

[13] L. Zhou, F. Li, and T. Zhang, "Adverse Drug Event Detection via Pretrained Language Models," *ACM Transactions on Computing for Healthcare*, vol. 3, no. 2, pp. 1–20, 2022.

[14] T. Mehta and R. Jain, “Machine Learning for Medication Impact Analysis in Longitudinal Healthcare Data,” *Journal of Biomedical Informatics*, vol. 118, Art. no. 103777, Elsevier, 2021.

Websites/URLs

[15] “Scikit-learn: Machine Learning in Python,” https://scikit-learn.org

[16] “UCI Machine Learning Repository,” https://archive.ics.uci.edu/ml/index.php

ANNEXURE D

Group Member-1:

1. Name: Om Babasaheb Bankar
2. DOB: 23/11/2003
3. Gender: Male
4. Address: Kedgaon Tal, Nagar Dist, A,Nagar
5. Email: oombankar990@gmail.com
6. Roll no: 62
7. Cell no: 9527498878
8. Paper Published: Yes

Group Member-2:

1. Name: Aditya Sanjay Shewale
2. DOB: 30/09/2002
3. Gender: Male
4. Address: Parbhani
5. Email: adityashewale77@gmail.com
6. Roll no: 37
7. Cell no: 9158083005
8. Paper Published: Yes

Group Member-3:

1. Name: Deepali Santoshsingh Thakur
2. DOB: 21/12/2003
3. Gender: Female
4. Address: Udgir Tal. Latur
5. Email: thakurdeepali1521@gmail.com
6. Roll no: 50
7. Cell no: 7262853865
8. Paper Published: Yes

Group Member-4:

1. Name: Harshada Suryakant Warade
2. DOB: 12/04/2003
3. Gender: Female
4. Address: Chh. Sambhajinagar
5. Email: harshwarade12@gmail.com
6. Roll no: 58
7. Cell no: 9307468339
8. Paper Published: Yes