Predictive Healthcare

A Project report submitted in partial fulfillment of the requirements
for the award of the degree of

Bachelor of Computer Applications (BCA)

Semester V

July – December, 2025

Submitted by:
Mohit Dhanotiya (2311160)
Khushi Kulshrestha (2311146)
Kuldeep Verma (2311149)

Under the guidance of:
Dr. Tarjani Sevak

School of Computer Science & IT

Devi Ahilya Vishwavidyalaya, Indore, (MP) - 452017, India

School of Computer Science & IT

Devi Ahilya Vishwavidyalaya, Indore, (MP) - 452017, India

DECLARATION

We hereby declare that the project report titled **"Predictive Healthcare"** is an original work completed by us as part of the requirements for the **Bachelor of Computer Applications (BCA)** program.

This project has not been submitted to any other university or institution for the award of any degree or diploma.
We have carried out this work under the guidance of **Dr. Tarjani Sevak**, and all referenced sources have been acknowledged properly.

Date:

Signature of Student:

Mohit Dhanotiya (2311160):

Khushi Kulshrestha (2311146):

Kuldeep Verma (2311149):

School of Computer Science & IT

Devi Ahilya Vishwavidyalaya, Indore, (MP) - 452017, India

CERTIFICATE FROM GUIDE

It is to certify that dissertation on **"Predictive Healthcare"** submitted by **Mohit Dhanotiya (2311160), Khushi Kulshrestha (2311146)** and **Kuldeep Verma (2311149)** to the School of Computer Science & IT, Devi Ahilya Vishwavidyalaya, Indore has been completed under my supervision and the work is carried out and presented in a manner required for its acceptance in partial fulfilment for the award of the degree of **Bachelor of Computer Applications (BCA).**

Project Guide

Dr. Tarjani Sevak

Signature:

Date:

School of Computer Science & IT

Devi Ahilya Vishwavidyalaya, Indore, (MP) - 452017, India

APPROVAL CERTIFICATE FROM EXAMINER

It is to certify that we have examined the dissertation on "**Predictive Healthcare**", submitted by **Mohit Dhanotiya (2311160), Khushi Kulshrestha (2311146)** and **Kuldeep Verma (2311149)** to the School of Computer Science & IT, Devi Ahilya Vishwavidyalaya, Indore and hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfilment for the award of the degree of **Bachelor of Computer Applications (BCA).**

Internal Examiner	**External Examiner**
Signature:	Signature:
Name:	Name:
Date:	Date:

School of Computer Science & IT

Devi Ahilya Vishwavidyalaya, Indore, (MP) - 452017, India

ACKNOWLEDGEMENT

We would like to thank our project guide, **Dr. Tarjani Sevak**, for their continuous support, clear guidance, and helpful suggestions during the development of our project titled **"Predictive Healthcare"**. Their inputs helped us understand the concepts better and complete our work on time.

We also thank the faculty members for providing useful knowledge throughout the course and for helping us whenever we needed assistance related to the project.

We are also grateful to our college for giving us the opportunity and resources to work on this project.

Student Names:

Mohit Dhanotiya (2311160)

Khushi Kulshrestha (2311146)

Kuldeep Verma (2311149)

Table of Contents

List of Figures

List of Tables

List of Algorithms

ABSTRACT

The project **Predictive Healthcare** focuses on building an AI-based system that predicts possible diseases using the symptoms provided by the user. Many people struggle to understand medical information on the internet, which often leads to confusion or incorrect self-diagnosis. This project aims to offer a clearer and more reliable approach by using machine learning techniques to analyze symptoms and suggest likely health conditions.

The system takes user-entered symptoms as input, processes them, and then uses a trained machine learning model to generate disease predictions. The goal is to make the process simple, fast, and useful for early awareness. The project includes stages such as requirement analysis, model training, interface design, implementation, and testing. The final system presents the predicted disease along with confidence information and encourages users to seek professional medical consultation when necessary.

Overall, **Predictive Healthcare** provides a supportive tool for basic health understanding and demonstrates how artificial intelligence can assist in the initial stages of medical decision-making.

CHAPTER 1

INTRODUCTION

Technology has brought major improvements in the healthcare field, especially with the use of artificial intelligence. Many people try to understand their health issues by searching symptoms online, but the information they find is often confusing, unreliable, or difficult to understand. This creates a need for a simple tool that gives clearer guidance based on proper data.

The project **Predictive Healthcare** aims to provide such guidance by predicting possible diseases using machine learning. Instead of typing symptoms, users can select their symptoms from a predefined list using a multiselect interface. This makes the system easy to use and reduces errors. Once the user selects their symptoms, the system processes the selection and uses a trained machine learning model to suggest the most likely disease.
The system is not a replacement for medical professionals, but it helps users understand their condition better and encourages timely medical consultation.

1.1 Background

Online health information is available in large amounts, but users often face difficulty in understanding it correctly. Searching for symptoms can lead to misleading results, and the user may not know which information is relevant. With the help of machine learning, it is possible to study patterns of symptoms and predict diseases more reliably.

This project uses a simple symptom-selection approach, where the user chooses symptoms from a list. These selected symptoms are then processed and passed to a prediction model trained on medical data. The aim is to give users a clearer and more focused understanding of their possible health issue.

1.2 Purpose of the Project

The purpose of this project is to create a tool that helps users quickly understand what disease they might be facing based on their symptoms. The project aims to:

- Provide simple and fast disease predictions
- Reduce confusion caused by online searches
- Offer a more structured and reliable result
- Encourage timely medical consultation when needed

1.3 Project Scope

The scope of the project includes:

- Allowing users to select symptoms from a predefined list
- Processing the selected symptoms for prediction
- Using a trained machine learning model to identify possible diseases
- Displaying the prediction in a clear and easy-to-understand manner
- Giving basic guidance for next steps

Future enhancements may include features such as:

- Uploading medical reports (images/PDFs)
- More detailed predictions
- A wider range of diseases and improved accuracy
- Providing basic treatment or doctor suggestions

1.4 Project Objectives

The main objectives of this project are:

1. To design a system that predicts diseases using user-entered symptoms
2. To convert selected symptoms into a format suitable for prediction
3. To train and use a machine learning model that can generate accurate predictions
4. To create a simple and user-friendly interface
5. To present the prediction in a clear and helpful manner
6. To support users in making informed decisions about their health

CHAPTER 2

PROJECT PLANNING AND SCHEDULING

Proper planning is important for completing any project in a smooth and organized way. It helps divide the work into clear steps, assign responsibilities, and maintain steady progress throughout development. In the **Predictive Healthcare** project, the work was planned in stages to ensure clarity and avoid delays.

2.1 Project Plan

The work of the project was divided into the following phases:

1. Requirement understanding
2. Dataset creation and preparation
3. Model selection and training
4. System design
5. Interface design
6. Integration of model and interface
7. Testing of system modules
8. Documentation and report preparation

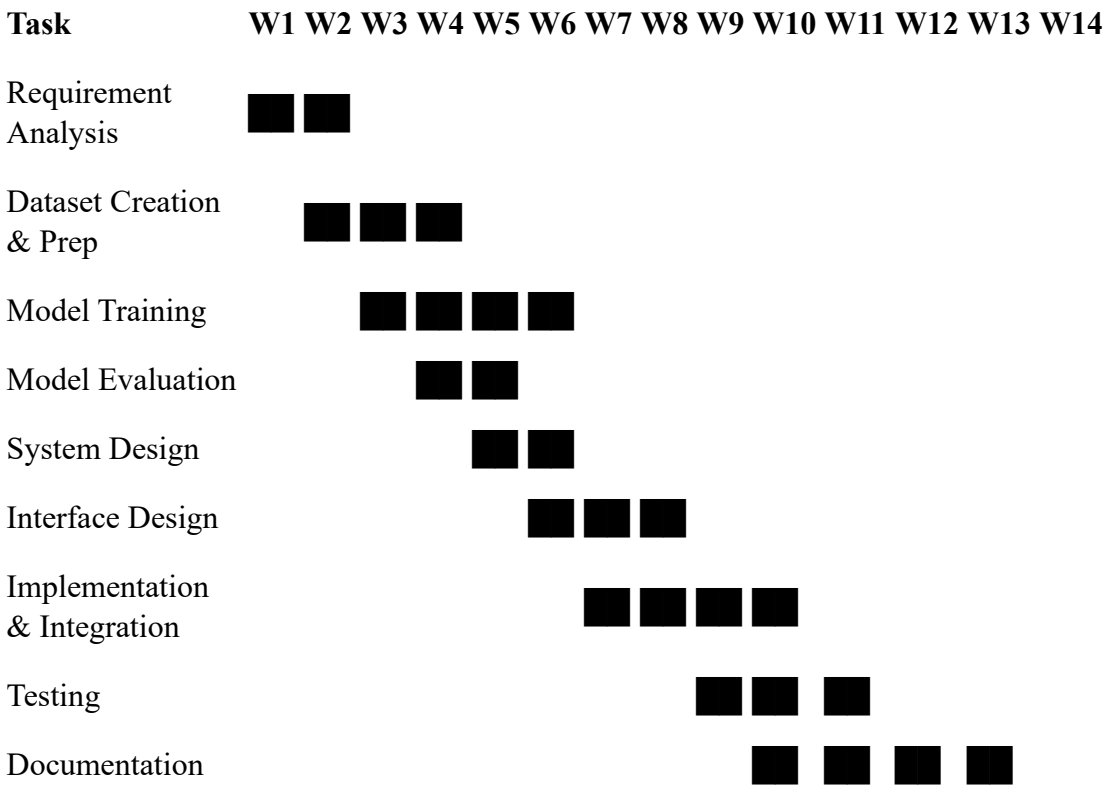
Each phase was completed carefully to maintain a structured development flow.

2.2 Work Breakdown Structure (WBS)

Level	Task	Description
1	Project Planning	Understanding goals and defining scope
2.1	Requirement Analysis	Identifying system needs and user flow
2.2	Dataset Preparation	Creating dataset and organizing symptom–disease values
3.1	Model Training	Selecting and training the ML model
3.2	Model Evaluation	Checking accuracy and performance

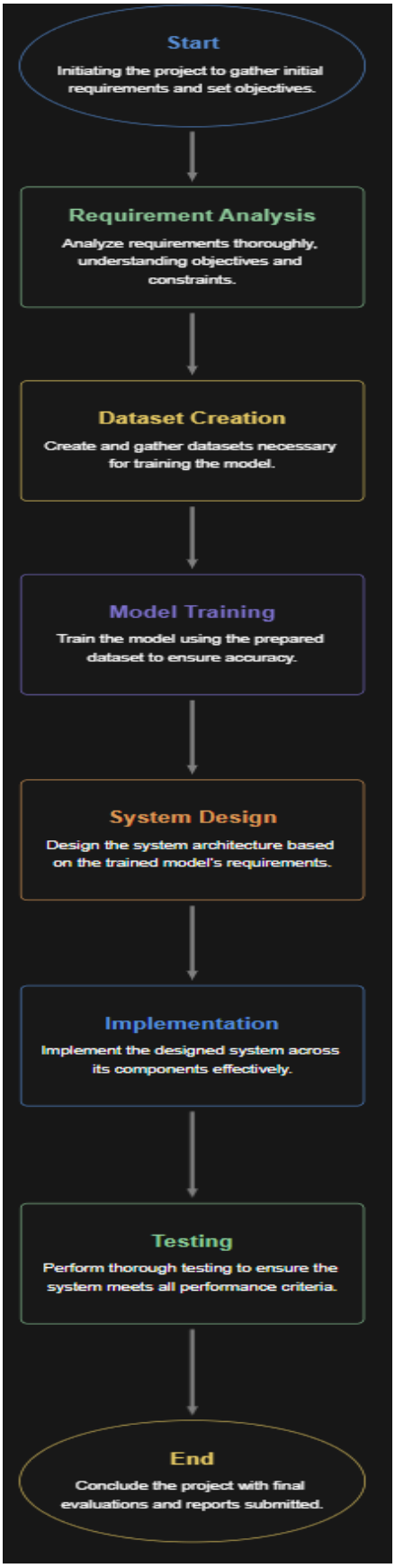
Level	Task	Description
4.1	System Design	Architecture and diagram creation
4.2	Interface Design	Designing input and output screens
5.1	Implementation	Model–UI integration
5.2	Testing	Unit, system, and acceptance tests
6	Documentation	Writing report and preparing screenshots

2.3 Gantt Chart (14-Week Plan)



2.4 PERT Chart

PERT Activity Flow



2.5 Team Structure and Responsibilities

The project was completed through a collaborative team effort, where all members contributed to every part of the work. Instead of dividing tasks strictly, the team worked together in planning, design, implementation, testing, and documentation. This helped maintain consistency and ensured that every member clearly understood the entire system.

The team collectively participated in:

- Understanding requirements
- Creating the dataset
- Designing diagrams and system flow
- Training and improving the model
- Developing and integrating the interface
- Testing and refining the system
- Preparing the final documentation

All decisions were taken together, and each member supported the others whenever needed. This shared approach helped maintain good coordination and played an important role in completing the project successfully.

2.6 Project Development Methodology

The project used the **Iterative Development Method**, where each cycle involved:

Plan → Design → Develop → Test → Improve

This method allowed continuous refinement and helped maintain a simple and flexible development approach.

2.7 Hardware and Software Requirements

Hardware Requirements

- **Processor:** Intel Core i5 (8th Gen or higher) / AMD Ryzen 5 or above
- **RAM:** Minimum 4 GB (Recommended: 8 GB or above for smooth model operation)
- **Storage:** 256 GB SSD or 500 GB HDD minimum
- **Internet Connection:** Minimum 5 Mbps speed required for Streamlit app

- **Power Backup:** Recommended for model testing sessions

Software Requirements

- **Operating System:** Windows 10 or above
- **Programming Language:** Python 3.11+
- **Framework:** Streamlit (v1.33+)
- **Libraries:** Pandas (v2.2+), Scikit-learn (v1.4+), Joblib (v1.4+)
- **Dataset:** CSV (Symptoms – Disease Mapping)
- Visual Studio Code (latest version) / PyCharm
- Jupyter Notebook
- Git & GitHub (for tracking and code storage)
- Chrome / Edge Browser (for Streamlit)

CHAPTER 3

SYSTEM ANALYSIS

System analysis helps in understanding the problem clearly and defining what the system should do. It involves identifying the requirements, studying the flow of data, and understanding how users will interact with the system. In the **Predictive Healthcare** project, system analysis was performed to understand the disease prediction process, the required inputs, the expected outputs, and the internal working of the system.

3.1 Problem Description

3.1.1 Problem Definition

There is a need for a system that can provide quick predictions about possible diseases.

Many users search for their symptoms online, but the information they find is often unorganized, unreliable, and difficult to understand. This leads to confusion and incorrect assumptions about their health condition. Without a simple and structured guidance tool, users struggle to identify which information is relevant and what disease their symptoms might indicate. A system that offers clear and fast predictions based on selected symptoms can help reduce this confusion and support users in understanding their condition better.

3.1.2 Proposed Solution

The proposed system, **Predictive Healthcare**, provides disease predictions based on symptoms selected by the user.
The solution includes:

- A list of symptoms that users can select
- A machine learning model trained on a custom dataset
- A prediction output that shows the most likely disease
- A simple guidance message for the user

This makes the prediction process structured, reliable, and easy for users to understand.

3.2 Requirements

3.2.1 Functional Requirements

1. The system should allow users to select symptoms from a predefined list.
2. The system should convert selected symptoms into a format suitable for prediction.
3. The system should use the trained model to predict the disease.
4. The system should display the predicted disease clearly.
5. The system should show the confidence of prediction (if available).
6. The system should allow users to reselect or modify symptoms.
7. The system should show a simple guidance message after prediction.

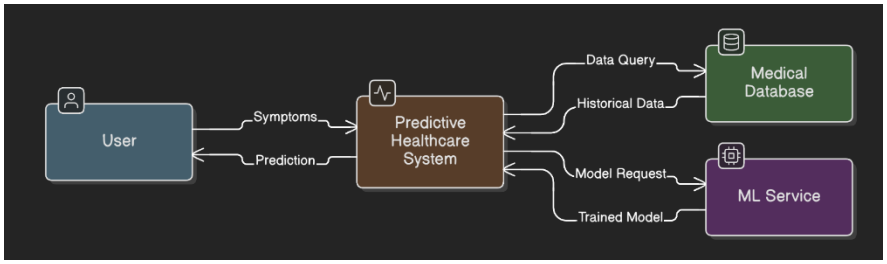
3.2.2 Non-Functional Requirements

1. **Accuracy:** Predictions should be reliable based on the training data.
2. **Usability:** The interface should be clean and easy to navigate.
3. **Efficiency:** Predictions should be generated quickly.
4. **Scalability:** More symptoms and diseases can be added in the future.
5. **Maintainability:** The system should be easy to update when required.
6. **Compatibility:** It should work smoothly on common operating systems.

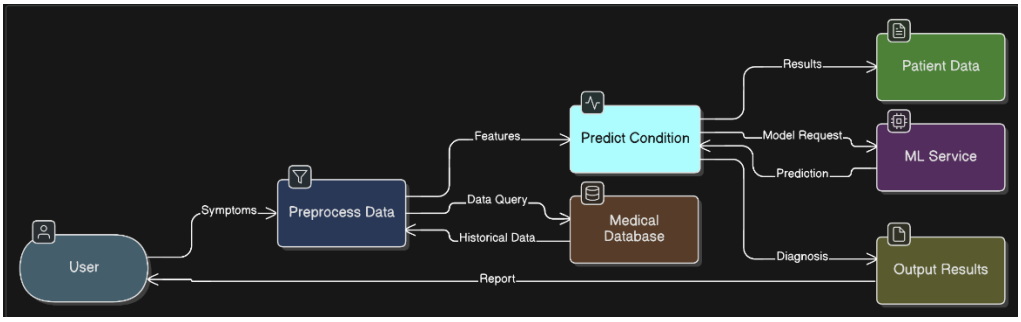
3.3 Analysis Diagrams

3.3.1 Data Flow Diagram (DFD)

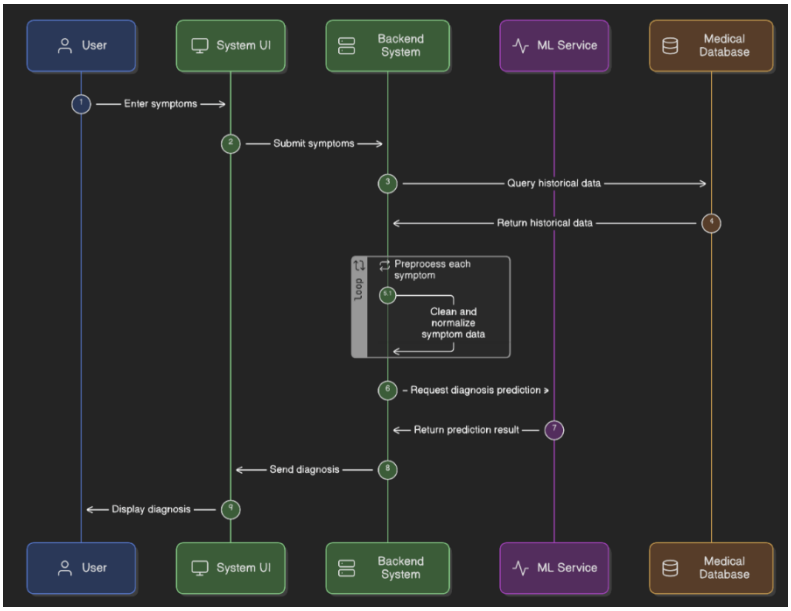
DFD Level 0 (Context Diagram)



DFD Level 1



3.3.2 Sequence Diagram



CHAPTER 4

SYSTEM DESIGN

System design provides a detailed understanding of how the **Predictive Healthcare System** works internally. It describes the architecture, major components, module interactions, input/output design, and the algorithms used for prediction.

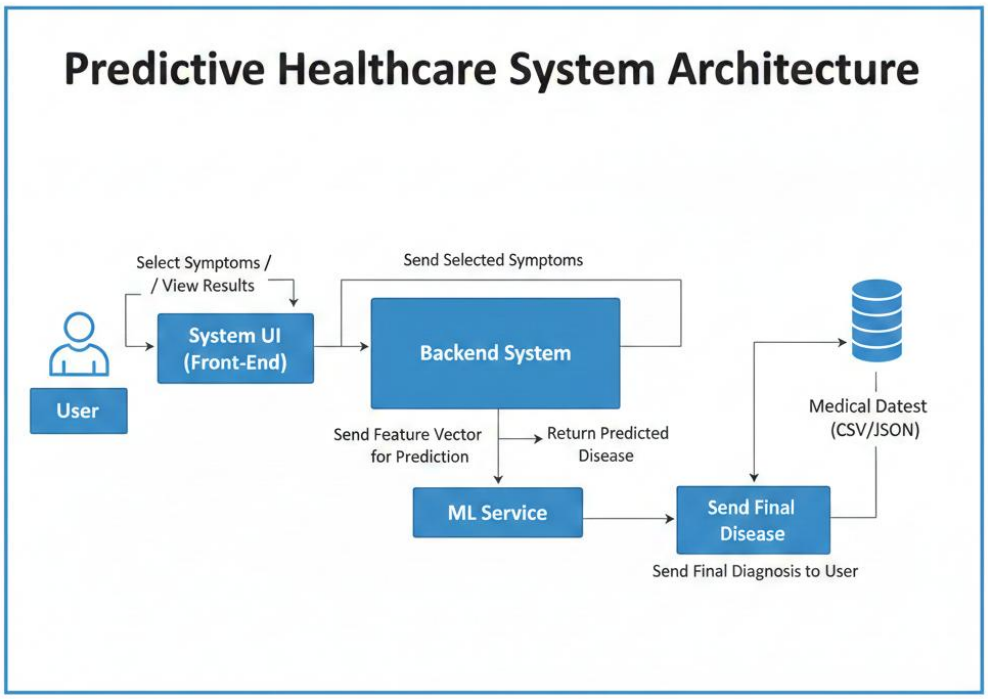
4.1 System Architecture

The system follows a simple and modular architecture consisting of:

- **User Interface (UI):** Accepts symptoms from the user and displays the final prediction.
- **Backend System:** Processes the symptoms, converts them into a feature vector, interacts with the model, and returns the output.
- **ML Service:** Loads the trained model and provides the disease prediction.
- **Dataset File:** Stores the symptom–disease mapping used during model training.

The overall workflow ensures a smooth transition from user input to machine learning-based disease prediction.

4.1.1 Architecture Diagram



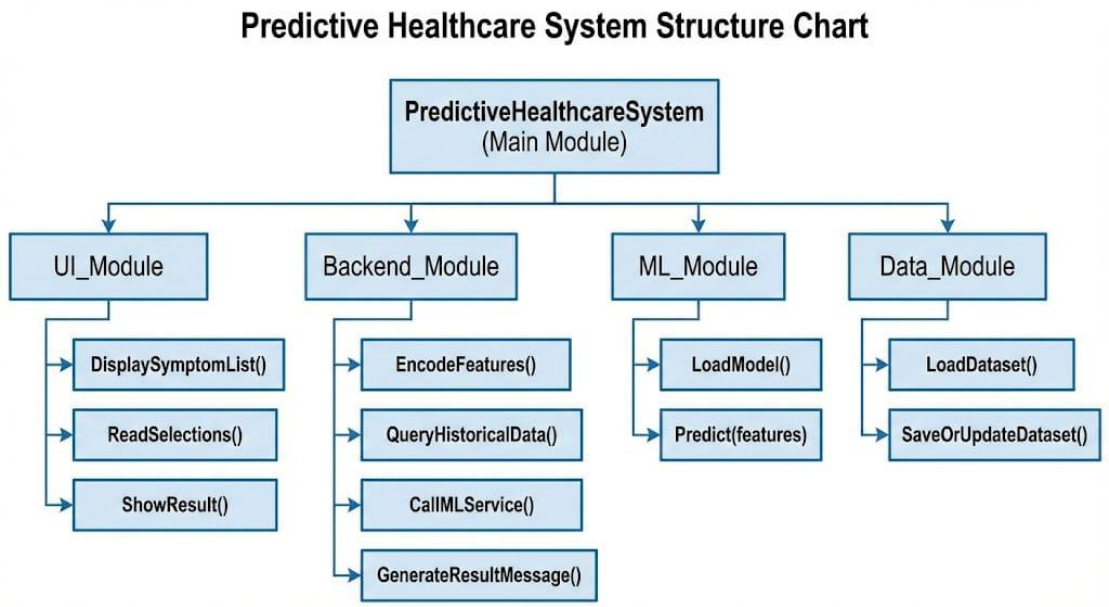
4.2 Physical Design

4.2.1 Deployment Structure

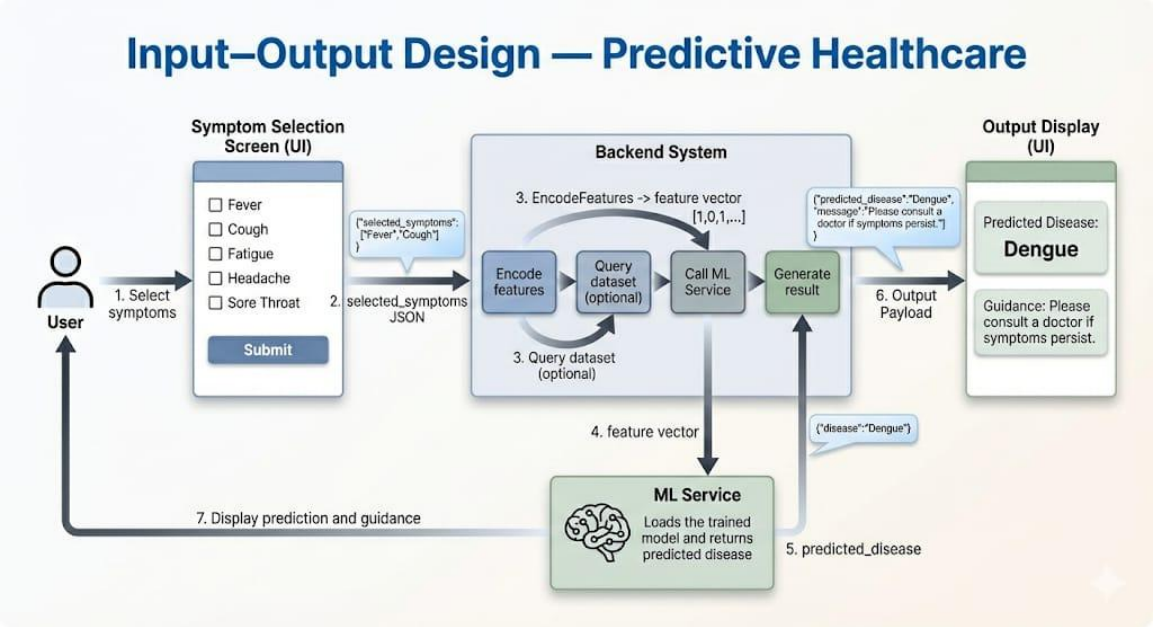
- The **User Interface** runs on the user's device (web app or local interface).
- The **Backend System** runs locally or on a simple server, handling processing and communication.
- The **ML Service** runs as a module that loads the trained machine learning model.
- The **Dataset** remains stored as CSV/JSON and is used only during preprocessing or retraining.

4.2.2 Structure Diagram

The structure diagram shows the main modules of the system and their internal functions.



4.3 Input and Output Design



4.3.1 Input Design

The input is taken through a **Symptom Selection Screen** where users can choose symptoms from a predefined checklist.
This method ensures clear and structured input without confusion.

Main elements of the screen:

- Symptom checklist (multi-select)
- Submit button

4.3.2 Output Design

After the model processes the symptoms, the system displays:

- **Predicted Disease**
- **A simple guidance message** (e.g., “Consult a doctor if symptoms persist.”)

The output screen includes:

- Disease result
- Guidance note

4.4 Algorithmic Design

This section explains the algorithms used for converting symptoms to features, predicting diseases, and generating results.

Algorithm 1 — Symptom Selection Processing

1. Start
2. Display the list of symptoms.
3. Allow user to select multiple symptoms.
4. For each symptom:

   If selected, assign 1

   Else assign 0
5. Create the feature vector from these values.
6. Send the feature vector to the prediction module.
7. End

Algorithm 2 — Disease Prediction

1. Start
2. Receive the feature vector from the backend.
3. Pass the vector to the trained machine learning model.
4. The model returns the predicted disease.
5. Send the predicted disease to the result generation module.
6. End

Algorithm 3 — Result Generation

1. Start
2. Receive the predicted disease from the prediction module.
3. Prepare the final message including predicted disease and a short guidance note.
4. Display the final output on the user interface.
5. End

CHAPTER 5

IMPLEMENTATION

Implementation refers to converting the system design into an operational software product. This phase describes the integration among modules in the system.

5.1 Integration of Modules / Files

This section describes how all modules communicate with each other, as required in the format.

5.1.1 UI ↔ Backend Integration

1. User selects symptoms
2. UI sends JSON request:

   ```
   {
           "selected_symptoms": ["Fever", "Headache"]
   }
   ```
3. Backend receives request → Encodes symptoms → Sends vector to ML Module
4. Backend receives predicted disease → Creates result message
5. UI displays the output

5.1.2 Backend ↔ ML Module Integration

- Backend calls:

  CallMLService(feature_vector)
- ML Module returns only:

  "Dengue"
- Backend wraps it into final output.

CHAPTER 6

SYSTEM TESTING

6.1 Test Case Design

Testing was carried out to ensure that the Predictive Healthcare System functions correctly, handles user inputs properly, and generates accurate predictions based on the selected symptoms. The test case design covers unit testing, integration testing, system testing, and acceptance testing.

6.1.1 Unit Testing

Unit testing focuses on individual modules of the system. Each module was tested independently to verify that it works as expected.

Modules Tested:

1. **UI Module**
    - Display of symptom list
    - Selection and clearing of symptoms
    - Navigation to result screen
2. **Backend Module**
    - Encoding symptoms into a feature vector
    - Proper handling of invalid/empty inputs
    - JSON request and response formatting
3. **ML Module**
    - Loading of trained model file
    - Returning the correct predicted disease

Unit Testing Result:

All individual modules worked correctly and produced the expected outputs.

6.1.2 Integration Testing

Integration testing ensures that modules interact correctly with one another.

Interactions Tested:

1. **UI → Backend**
    - Inputs sent in JSON format
    - Backend receives selected symptoms correctly
2. **Backend → ML Service**
    - Feature vector sent to model
    - Model returns predicted disease
3. **Backend → UI**
    - Output message displayed correctly on the result screen

Integration Testing Result:

Module interactions performed smoothly with no communication errors.

6.1.3 System Testing

System testing checks the complete end-to-end workflow.

Scenarios Tested:

1. Selecting valid symptoms → Correct prediction shown
2. Selecting multiple symptoms → System handles combinations correctly
3. Submitting without selecting symptoms → Appropriate error message displayed

System Testing Result:

The system behaved as expected in all scenarios and satisfied functional requirements.

6.1.4 Acceptance Testing

Acceptance testing verifies that the system meets user expectations and project goals.

Acceptance Criteria:

- Easy-to-use interface
- Fast prediction response
- Clear output message

- Accurate mapping between symptoms and diseases
- No app crash or failure

Acceptance Testing Result:

Users found the system simple, understandable, and accurate.
All acceptance criteria were satisfied.

6.2 Specific System Testing

The following specific tests were performed to handle edge cases and confirm robustness:

Test Case	Input	Expected Output	Result
TC01	Fever, Headache	Correct disease prediction	Pass
TC02	Fever, Cough, Fatigue	Prediction shown properly	Pass
TC03	No symptoms selected	Error: “Please select at least one symptom.”	Pass
TC04	Random symptom combinations	Stable prediction output	Pass

The system handled all specific test conditions successfully.

6.3 Test Reports

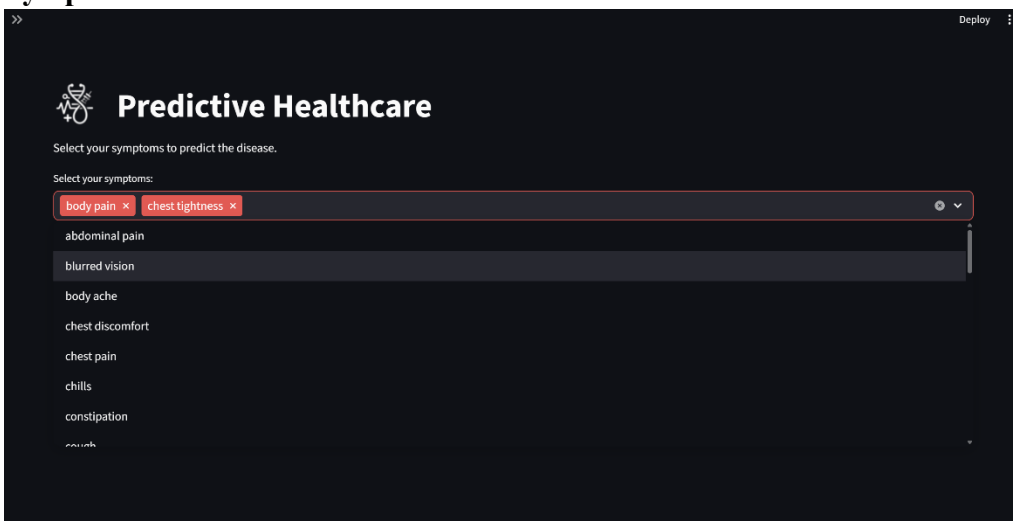
Below are summarized reports of the tests conducted.

Test Summary Table

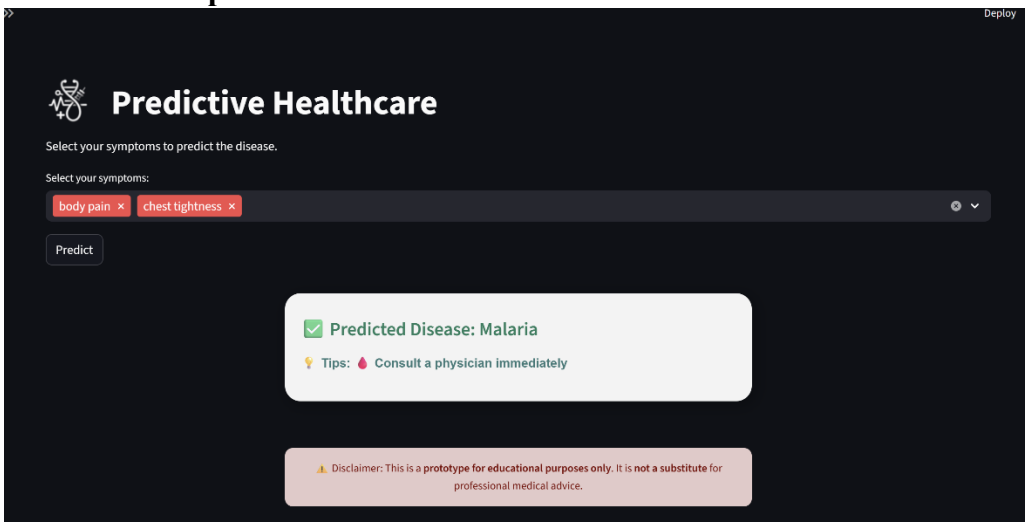
Test Category	Status
Unit Testing	✓ Passed
Integration Testing	✓ Passed
System Testing	✓ Passed
Acceptance Testing	✓ Passed
Specific System Tests	✓ Passed

6.3.1 Sample Test Evidence (Screenshots)

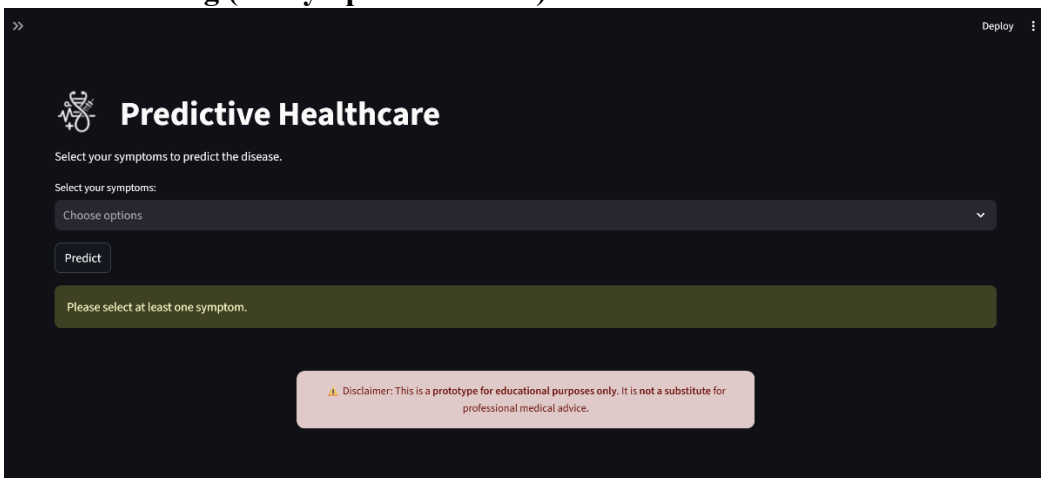
- **Symptom Selection Screen**



- **Prediction Output Screen**



- **Error Handling (No Symptom Selected)**



CHAPTER 7

CONCLUSION OF THE PROJECT

7.1 Results

The Predictive Healthcare System successfully performs disease prediction based on user-selected symptoms. The system generates results quickly and displays them in a clear and understandable format. The main outcomes achieved are:

- The system accurately converts selected symptoms into a feature vector.
- The trained machine learning model correctly returns a predicted disease based on the input.
- The output is displayed in a simple format containing the disease name and a short guidance message.
- The user interface allows smooth navigation between symptom entry and result viewing.
- The system handles invalid inputs, such as no symptom selection, by showing appropriate error messages.

Overall, the implemented system meets the project objectives and provides reliable performance during testing.

7.1.1 Accuracy results

The performance of the machine learning model was evaluated using two standard methods: **Train–Test Split** and **K-Fold Cross-Validation**. In the train–test approach, the model achieved an accuracy of **95%**, showing strong performance on the test dataset. To further assess the model's consistency, K-Fold Cross-Validation was applied, resulting in an average accuracy of **80%** across multiple folds. This indicates that while the model performs very well on the test split, its generalized performance is moderately lower but still acceptable for the scope of this student-level project. Overall, the evaluation confirms that the model is capable of generating reliable disease predictions based on symptoms.

Accuracy Comparison Table

Evaluation Method	**Accuracy**
Train–Test Split	**95%**
K-Fold Cross-Validation	**80%**

7.2 Limitations of the Project

Although the system works effectively for the intended purpose, it has certain limitations:

- The predictions depend entirely on the dataset created manually, which may not cover all diseases.
- The system identifies only one predicted disease, without ranking alternatives.
- The symptom list is fixed and limited; users cannot input new or uncommon symptoms.
- The system does not store patient data or learning history for personalized outputs.
- The model accuracy is restricted by dataset size and quality.
- No real-time medical data or external APIs are integrated.

These limitations arise mainly due to project scope and student-level constraints.

7.3 Conclusion

The Predictive Healthcare System demonstrates how machine learning can be applied to assist users in identifying possible diseases using basic symptom inputs. The project fulfills its aim of offering a simple, fast, and user-friendly platform for health awareness. The modular design ensures that each component—UI, backend, ML model, and dataset—works effectively together.
Testing confirms that the system operates smoothly under multiple scenarios and provides meaningful output to users.

While this system is not meant to replace medical diagnosis, it acts as a useful preliminary tool that can guide users toward better understanding of their symptoms. The project also lays a strong foundation for future improvements and expansion.

7.4 Future Work

Several enhancements can improve the system in future versions:

- Expanding the dataset with more diseases and symptoms.
- Integrating real medical datasets or APIs for improved accuracy.
- Allowing users to upload medical reports (PDF/Images) for enhanced predictions.

- Adding multi-disease prediction instead of single output.
- Including severity and risk-level estimation.
- Introducing multilingual support in the user interface.
- Implementing cloud deployment for wider accessibility.
- Adding doctor recommendation or appointment booking functionality.

These improvements can significantly increase the system's usefulness and reliability.

7.5 Lessons Learned

During the development of this project, the team gained valuable technical and practical experience:

- Understanding how machine learning models are trained and integrated into applications.
- Learning the process of converting raw symptoms into structured feature vectors.
- Gaining experience in UI design, backend development, and module integration.
- Realizing the importance of clean dataset preparation for accurate predictions.
- Improving skills in testing, debugging, and validation.
- Learning teamwork, coordination, and documentation practices.

This project provided hands-on exposure to real-world problem-solving using machine learning and software development principles.

REFERENCES

- Scikit-learn Documentation. *Machine Learning in Python*. https://scikit-learn.org/
- Python Documentation. *Official Python Language Reference*. https://docs.python.org/3/
- W3Schools. *Basic Web and UI Concepts*. https://www.w3schools.com/
- GeeksforGeeks. *Machine Learning and Python Articles*. https://www.geeksforgeeks.org/
- Streamlit Documentation (used for interface understanding). https://docs.streamlit.io/