iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79569

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Medical Diagnosis Prediction Software

Prof.Tara Shende, Mr. Rahul Sukhdeve, Mr. Lokesh Rathod, Mrs. Rutuja Thool, Mr. Pranit Butale
Bachelor of Technology in Artificial Intelligence and Data Science Wainganga College of Engineering and Management, Nagpur, Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur

***Abstract:** AI-Based Disease Prediction using Machine Learning is a modern healthcare solution developed to improve the accuracy and speed of medical diagnosis. The system analyzes large amounts of patient data, including symptoms, medical history, demographic details, lifestyle habits, and laboratory test results. By applying powerful machine learning algorithms, it identifies patterns and correlations that may not be easily visible to doctors during manual examination. This helps in predicting diseases such as diabetes, heart disease, liver disorders, and certain types of cancer at an early stage. The system aims to support healthcare professionals by reducing diagnostic errors, providing risk assessments, and enabling timely treatment. It also enhances preventive healthcare by offering early warnings that allow patients to take corrective action before the disease progresses. Overall, this AI-based approach improves healthcare efficiency, supports better decision-making, and contributes to more reliable and accessible medical services.*

I. INTRODUCTION

In the last decade, the global healthcare sector has undergone a major transformation driven by rapid advancements in digital technologies, particularly Artificial Intelligence (AI) and Machine Learning (ML). Traditional diagnosis methods often rely on manual evaluation, clinical expertise, and time-consuming medical tests, which may sometimes lead to delays, inconsistencies, or human errors. As diseases become more complex and patient data grows exponentially, it has become increasingly difficult for healthcare professionals to manually analyze every detail with absolute precision. This challenge has created a strong need for intelligent systems that can support doctors by offering fast and accurate interpretations of medical data. Machine Learning algorithms excel in this area, as they can analyze massive datasets, identify hidden trends, and generate predictive insights that are not easily visible through conventional diagnostic processes. By incorporating patient symptoms, lifestyle factors, medical history, and laboratory test results, AI-based disease prediction systems help in identifying the likelihood of various diseases at an early stage, ensuring timely treatment and improved patient outcomes. The integration of AI in healthcare not only enhances diagnostic accuracy but also contributes to reducing healthcare costs, optimizing clinical workflows, and building a more efficient, reliable, and patient-centered medical ecosystem.

With the continuous evolution of technology, the healthcare industry has shifted from traditional diagnosis methods to more data-driven, intelligent, and automated systems. As the number of patients increases and healthcare records become more complex, the limitations of manual diagnosis have become more evident. Doctors often face challenges such as time pressure, incomplete information, misinterpretation of symptoms, and the complexity of certain diseases that require detailed evaluation. The emergence of Artificial Intelligence and Machine Learning offers a powerful solution to these problems by providing advanced computational techniques capable of analyzing and learning from large medical datasets. These intelligent models can detect subtle patterns, classify diseases, and predict health risks with high accuracy. AI-based disease prediction systems make use of datasets containing vital health indicators, demographic details, medical history, biochemical parameters, and lifestyle choices to assess the likelihood of diseases such as diabetes, cardiovascular disorders, liver disease, and cancer. By generating real-time predictions, these systems support healthcare professionals in making evidence-based decisions, improving treatment planning, and reducing diagnostic errors. Moreover, they play a crucial role in preventive healthcare, enabling early detection and helping patients adopt healthier lifestyle choices before the disease progresses. This technological shift represents a significant advancement toward smart healthcare solutions, where AI acts as a supportive tool that enhances human expertise and improves overall healthcare quality.

The integration of Artificial Intelligence in healthcare has emerged as one of the most revolutionary developments of the 21st century. In the past, the diagnosis of diseases relied heavily on manual assessment, physician experience, and time-consuming medical procedures. Although these methods have been effective, they often lack the ability to process large amounts of data quickly and accurately. Today, with the availability of electronic health records, laboratory data, wearable device information, and large medical datasets, there is a growing opportunity to utilize Machine Learning techniques to improve diagnostic accuracy and speed.

AI-based disease prediction systems are designed to analyze multiple patient parameters simultaneously, including symptoms, vital signs, clinical reports, environmental influences, genetic factors, and past medical records. By applying advanced algorithms such as Random Forest, Support Vector Machine, K-Nearest Neighbors, and Neural Networks, these systems can identify disease patterns and generate highly accurate risk predictions. This capability makes AI an essential tool for early diagnosis, especially for diseases that show mild or generic symptoms during initial stages. The adoption of such intelligent systems not only reduces the workload of healthcare professionals but also ensures faster decision-making and improves healthcare accessibility for patients living in remote or underserved areas. Additionally, these technologies contribute to personalized healthcare by offering tailored recommendations and early warnings based on individual health profiles.

Overall, the emergence of AI-based disease prediction marks a significant step toward building a smarter, more accurate, and more efficient healthcare environment that supports both preventive and curative strategies.

A. Problem Statement:

Healthcare systems face significant challenges in diagnosing diseases early, particularly when symptoms are unclear or when large volumes of patient data are involved. Manual diagnostic procedures often result in **inconsistencies**, time delays, and difficulty in recognizing subtle relationships within clinical datasets. With the increasing availability of digital medical records, laboratory results, and patient health data, there is a pressing need for automated systems capable of interpreting this information efficiently. However, existing diagnostic tools are either too specific, limited in scope, or lack the ability to adapt to diverse medical conditions. Thus, the problem addressed in this project is the development of an **automated, machine learning–based disease prediction model** that can analyze complex datasets, identify key risk factors, and accurately predict diseases such as diabetes, heart disease, and liver disorders. The goal is to create a system that enhances **diagnostic accuracy**, minimizes human error, and assists healthcare professionals in delivering faster and more reliable medical decisions.

B. Objectives:

1) To develop an AI-based system capable of predicting diseases using machine learning algorithms.
2) To analyze patient data such as symptoms, medical history, and lifestyle factors for accurate prediction.
3) To reduce diagnostic errors caused by manual interpretation and human limitations.
4) To provide early detection of common and critical diseases for timely treatment.
5) To assist doctors and healthcare professionals in making evidence-based clinical decisions.
6) To build a user-friendly interface for easy input and understanding of patient data and predictions.
7) To improve healthcare efficiency by reducing the time and cost involved in traditional diagnosis.
8) To promote preventive healthcare by providing risk assessments and early warnings.

II. LITERATURE SURVEY

S. No.	Title	Authors	Year	Technology Used	Key Contributions
1	AI-Based Multi-Disease Prediction System	R. Kaur, S. Gupta	2024	Random Forest, Python	Developed a model to predict multiple diseases with improved accuracy using ensemble ML.
2	Deep Learning Framework for Medical Diagnosis	L. Zhao, M. Reddy	2024	CNN, TensorFlow	Improved early detection of diseases using deep-learning with high performance on test data.
3	Machine Learning for Early Cancer Risk Detection	A. Bose, N. Roy	2023	XGBoost, Python	Used feature-based ML model for cancer risk prediction with improved sensitivity.
4	Explainable AI for Heart Disease Detection	P. Menon, R. Iyer	2023	XAI Tools, Logistic Regression	Provided interpretable predictions to help doctors understand key risk factors.

5	IoT and ML-Based Diabetes Monitoring System	S. Ahmed, V. Nair	2022	IoT Sensors, LSTM	Enabled real-time health monitoring and diabetes risk alert using sensor data.
6	Hybrid ML Model for Heart Disease Prediction	M. Patel, K. Shah	2021	SVM, Random Forest	Combined ML algorithms to improve accuracy and reduce diagnostic errors.

1) AI-Based Multi-Disease Prediction System (2024):-

This study focuses on building an AI model capable of predicting multiple diseases such as diabetes, heart disease, and kidney disorders. The authors used **Random Forest** because it works well with medical data and handles large numbers of features effectively. The system analyzes patient records and identifies key patterns that help in early detection. The model achieved high accuracy and proved helpful for hospitals to automate diagnosis and reduce errors.

2) Deep Learning Framework for Medical Diagnosis (2024):-

This research introduces a **CNN-based deep learning system** that analyzes medical data to detect diseases early. The model was trained using TensorFlow, allowing it to learn complex patterns in symptoms and clinical records. It showed strong performance in classification tasks and achieved better accuracy compared to traditional machine learning methods. The study supports the use of deep learning for fast and reliable diagnosis.

3) Machine Learning for Early Cancer Risk Detection (2023):-

This paper presents a prediction model using XGBoost, a powerful boosting algorithm. The model was designed to identify early cancer risks based on patient data like age, lifestyle, and medical history. The algorithm performed well in recognizing subtle signs of risk, helping doctors take preventative steps early. This study highlights the importance of machine learning in detecting dangerous diseases before they become severe.

4) Explainable AI for Heart Disease Detection (2023):-

The researchers developed a heart disease prediction system using Logistic Regression and XAI tools like SHAP and LIME. The aim was not just to predict disease but also to explain why a prediction was made. Doctors could see which features (like blood pressure, cholesterol, etc.) contributed most to the risk. This makes the model more trustworthy and easier for healthcare professionals to use in decision-making.

5) IoT and ML-Based Diabetes Monitoring System (2022):-

This study combines IoT and machine learning to create a real-time diabetes monitoring system. Wearable sensors collect health data such as glucose level and physical activity. An LSTM model then analyzes time-series data to predict sudden spikes or drops. The system helps diabetic patients receive early alerts and better manage their health. It shows how IoT and AI can work together for continuous monitoring.

6) Hybrid ML Model for Heart Disease Prediction (2021):-

The authors proposed a hybrid model that combines SVM and Random Forest to improve heart disease prediction. Features are selected using optimization techniques to reduce noise in data. The hybrid approach increases accuracy and reduces false predictions compared to using a single algorithm. This study emphasizes that combining multiple ML models can produce better and more stable medical predictions.

III. METHODOLOGY

1) Data Collection: -

The first step involves gathering relevant and high-quality medical datasets from reliable sources. These datasets include patient information such as age, gender, symptoms, medical history, laboratory test results, and lifestyle indicators (e.g., smoking habits, physical activity). Public datasets like UCI Machine Learning Repository, Kaggle medical datasets, and WHO health records are used to ensure diversity and completeness.

Multiple disease datasets (e.g., diabetes, heart disease, liver disease) are combined so that the system can predict various conditions. The collected data is stored in structured formats such as CSV or Excel for further processing.

2) Data Preprocessing: -

Medical data often contains missing values, noise, inconsistencies, and outliers, which can negatively affect model performance. Therefore, preprocessing is essential.

This step includes:

- Handling missing values using techniques such as mean/median imputation or KNN-based filling.
- Removing duplicates and irrelevant or corrupt entries.
- Normalizing and standardizing numerical attributes to keep them on a similar scale.
- Encoding categorical variables (e.g., gender, smoking status) using one-hot or label encoding.
- Outlier detection using IQR and Z-score methods to remove unrealistic values.

This stage ensures the input data is clean, consistent, and machine-readable.

3) Exploratory Data Analysis (EDA): -

EDA provides a detailed understanding of the dataset. Various statistical and visualization techniques are applied to identify patterns, correlations, and trends.

This includes:

- Distribution plots for each feature
- Correlation heatmaps to check relationships between parameters
- Histogram and box-plot analysis to understand variability
- Class imbalance checking (e.g., how many patients have disease vs. no disease)

The insights from EDA guide decisions in feature selection and model choice.

4) Feature Engineering: -

Feature engineering improves model performance by extracting meaningful information from raw data.

Common steps include:

- Selecting key features that have the highest impact on disease prediction
- Removing redundant or correlated features to reduce overfitting
- Creating new features, such as BMI from height and weight
- Scaling features using Min-Max Scaling or Standard Scaler
- Balancing the dataset using SMOTE if the data is imbalanced

This step enhances the predictive power and accuracy of the model.

5) Model Selection: -

Multiple machine learning algorithms are considered to find the best one for disease prediction. The models usually include:

- Logistic Regression
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Gradient Boosting & XGBoost
- Naïve Bayes

Each model is tested with training data, and its performance is compared using evaluation metrics. The best-performing models are selected for final training.

6) Model Training: -

After selecting suitable models, the training process begins. The dataset is split into training and testing sets (typically 80% training, 20% testing).

During training:

- The model learns the patterns and relationships between features and outcomes.
- Hyperparameters such as learning rate, number of trees, kernel type, and maximum depth are fine-tuned.
- Techniques like cross-validation (k-fold) are used to ensure stability and prevent overfitting.

The training phase ensures the model can generalize well to unseen data.

7) Model Evaluation: -

To check how well the model performs, several performance metrics are used:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC Curve
- Confusion Matrix

These metrics help identify whether the model is reliable for medical diagnosis. Models with higher recall are generally preferred for disease detection to reduce false negatives.

8) Model Optimization: -

If necessary, additional tuning is performed to improve accuracy, such as:

- Hyperparameter optimization using Grid Search or Random Search
- Feature selection refinement
- Reducing model complexity to avoid overfitting
- Using ensemble techniques for higher stability

Optimization ensures that the final model is both accurate and computationally efficient.

9) System Integration & User Interface Development: -

Once the prediction model is finalized, it is integrated into a user-friendly application.
This includes:

- Designing a web-based or mobile interface (e.g., using HTML, CSS, JavaScript, Flask, or Streamlit).
- Creating a form for users to enter symptoms or medical parameters.
- Connecting the front-end to the trained ML model.
- Ensuring real-time prediction generation.

The interface is made simple so that even non-technical users can easily access the system.

10) Model Deployment: -

The final step is deploying the model so it can be accessed by doctors and patients. Deployment can be done using:

- Flask API
- Streamlit web app
- Cloud platforms like AWS, Heroku, or Google Cloud

Once deployed, the system provides instant disease predictions based on user inputs. Regular updates and retraining ensure long-term reliability.

A. Requirements: -

1) Software Requirements: -

- OS: Windows 10/11
- Editor/IDE: VS Code / PyCharm
- Language: Python / Java / HTML-CSS-JS
- Framework: Django / Flask (if web)

- Database: MySQL / SQLite
- Others: GitHub, MS Office, Chrome

2) Hardware Requirements (Short): -

- Processor: Minimum i3, Recommended i5
- RAM: 4 GB (min), 8 GB (recommended)
- Storage: 256 GB (min), 512 GB SSD (recommended)
- Display: 14"–15.6" FHD
- Network: Stable internet
- Extra: Mouse, Keyboard

IV. IMPLEMENTATION

A. Steps :-

1) Environment Setup: -

To begin the implementation, a development environment is created using essential tools and libraries:

- Programming Language: Python
- Libraries: NumPy, Pandas, Scikit-Learn, Matplotlib, Seaborn, XGBoost, TensorFlow (optional)
- Backend Framework: Flask or Django
- Database: SQLite / Firebase / MySQL (optional)
- Development Tools: Jupyter Notebook, VS Code, PyCharm

All required packages are installed using **pip**, ensuring that the environment supports data processing, visualization, and machine learning functions.

2) Dataset Preparation & Preprocessing: -

The collected datasets (such as diabetes, heart disease, and liver disease) are imported into the system using Pandas. Each dataset undergoes preprocessing steps implemented directly in Python:

- Handling missing values using imputation techniques
- Label encoding categorical variables like gender or chest pain type
- Standardizing numerical features
- Removing duplicate records
- Splitting data into training and testing sets

These preprocessing steps are implemented with Scikit-Learn tools such as StandardScaler, LabelEncoder, and train_test_split.

3) Machine Learning Model Development: -

Multiple machine learning algorithms are implemented to compare performance. Each model is trained using the prepared dataset:
Models Implemented:

- Logistic Regression
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Naïve Bayes
- Gradient Boosting / XGBoost

Each model is coded in Python using Scikit-Learn functions.
Model Evaluation & Selection
After training, each model is tested on X_test data and evaluated using:

- Accuracy
- Precision & Recall
- F1-score

- Confusion Matrix

The best-performing model (often Random Forest or XGBoost) is selected for final deployment.
The model is then saved as a .pkl file using pickle so it can be loaded inside the web application.

4) Backend Development: -

The backend is developed using **Flask**, where APIs are created to handle user inputs and generate predictions.

Process:

- Create a Flask app (app.py)
- Load the saved trained model using pickle
- Make routes for:
  - Home page
  - Prediction page
  - API request handling

The backend ensures that the machine learning model runs efficiently and provides real-time results.

5) Front-End Interface Development: -

A simple, clean, and user-friendly interface is created using:

- HTML
- CSS
- Python

Features of the Interface:

- Input fields for symptoms and medical parameters
- Form submission button
- Display area for prediction result
- Error validation for incorrect inputs

When the user enters details and clicks **Predict**, the data is sent to Flask via POST request, and the predicted disease result appears instantly.

6) Integration of Frontend and Backend: -

After the machine learning model, backend API, and frontend interface are completed, they are integrated:

- Frontend HTML sends user input to Flask using HTTP request
- Flask processes the data and feeds it into the ML model
- Model predicts the disease
- Flask returns the output to the user interface
- The result is displayed on the webpage

This integration ensures a smooth flow from input → prediction → output.

7) Testing and Debugging

The system is tested extensively to identify and fix errors:

- Functional Testing: Ensures features like prediction, input validation, and navigation work correctly.
- Performance Testing: Checks how fast predictions are generated.
- Accuracy Testing: Compares predicted results with known medical data.

Testing ensures the system is reliable and ready for use.

8) Deployment: -

The final step is deploying the system so that users can access it online. Deployment can be done using:

- PythonAnywhere
- AWS / GCP / Azure (for advanced deployment)

The trained ML model and Flask app are uploaded to the server, and the system becomes accessible via a public link.

B. Programming :-

1) Fronted Code : -

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Clinical Notes Disease Prediction</title>
<link rel="stylesheet" href="/static/styles.css">
</head>
<style>
    body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #fff;
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 400px;
    text-align: center;
}

h1 {
    color: #333;
    font-size: 24px;
}

input[type="file"] {
    margin: 20px 0;
    display: block;
    width: 100%;
    padding: 10px;
    border-radius: 5px;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 12px 20px;
```

```
    border: none;
    cursor: pointer;
    width: 100%;
    font-size: 16px;
    border-radius: 5px;
}

button:hover {
    background-color: #45a049;
}

.spinner {
    margin: 20px auto;
    border: 4px solid #f3f3f3;
    border-top: 4px solid #3498db;
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 2s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

.result {
    margin-top: 20px;
    font-size: 18px;
    color: #333;
}

</style>
<body>
<div class="container">
<h1>Clinical Notes Disease Prediction</h1>
<form action="/predict/" method="post" enctype="multipart/form-data" id="uploadForm">
<input type="file" name="file" accept=".txt,.pdf" required>
<button type="submit" id="submit-btn">Predict Disease</button>
</form>
<div id="spinner" class="spinner" style="display:none;"></div>
<div id="result" class="result" style="display:none;"></div>

</div>

<script>
    const form = document.getElementById("uploadForm");
    const spinner = document.getElementById("spinner");
    const resultDiv = document.getElementById("result");
```

```
    form.onsubmit = async (event) => {
    event.preventDefault();
    spinner.style.display = "block"; // Show the spinner
    resultDiv.innerHTML = "";        // Clear previous results
    resultDiv.style.display = "none"; // Hide result while loading

    try {
        // Handle the file upload
        const formData = new FormData(form);
        const response = await fetch('/predict/', {
            method: 'POST',
            body: formData,
        });

        if (!response.ok) {
            throw new Error("Error in fetching the prediction.");
        }

        const data = await response.json();
        spinner.style.display = "none"; // Hide the spinner

        // Render results
        resultDiv.innerHTML = `
<h3 style="text-align: center; color: #333;">Patient Clinical Report</h3>
<div style="text-align: left; border: 1px solid #ccc; padding: 15px; border-radius: 8px; background-color: #fff;">
<table style="width: 100%; border-collapse: collapse;">
<tr style="border-bottom: 1px solid #ddd;">
<td style="text-align:left; padding: 8px; font-weight: bold;">Predicted Disease:</td>
<td style="style="text-align:left; padding: 8px;">${data.predicted_disease}</td>
</tr>
<tr style="border-bottom: 1px solid #ddd;">
<td style="style="text-align:left; padding: 8px; font-weight: bold;">Description:</td>
<td style="style="text-align:left; padding: 8px;">${data.description}</td>
</tr>
<tr style="border-bottom: 1px solid #ddd;">
<td style="style="text-align:left; padding: 8px; font-weight: bold;">Recommended Medicines:</td>
<td style="style="text-align:left; padding: 8px;">${data.medicines.join(", ")}</td>
</tr>
<tr style="border-bottom: 1px solid #ddd;">
<td style="style="text-align:left; padding: 8px; font-weight: bold;">Specialists:</td>
<td style="style="text-align:left; padding: 8px;">${data.specialists.join(", ")}</td>
</tr>
</table>
</div>
        `;
        resultDiv.style.display = "block"; // Show results
    } catch (error) {
        spinner.style.display = "none";
        resultDiv.innerHTML = `<p style="color: red;">Error: ${error.message}</p>`;
```

```
        resultDiv.style.display = "block"; // Show error
    }
};

</script>
</body>
</html>
```

2) Backend Code : -

```
# pip install fastapi uvicorn torch transformers PyPDF2 aiofiles scikit-learn nltk python-multipart

from fastapi import FastAPI, File, UploadFile, Form
from fastapi.responses import HTMLResponse, JSONResponse
from fastapi.templating import Jinja2Templates
from transformers import BertForSequenceClassification, BertTokenizer
import torch
import PyPDF2
from io import BytesIO
import pickle
from fastapi import Request
import numpy as np
import re
from nltk.corpus import stopwords

# Make sure to download stopwords from nltk
import nltk
nltk.download('stopwords')


# Load the model, tokenizer, and label encoder
model = BertForSequenceClassification.from_pretrained('./patient_model')
tokenizer = BertTokenizer.from_pretrained('./patient_model')
label_encoder = pickle.load(open("label_encoder.pkl", 'rb'))

# FastAPI instance
app = FastAPI()

# Set up templates and static file directory
templates = Jinja2Templates(directory="templates")


# Disease data
disease_data = {
    "Peptic Ulcer Disease": {
        "description": "A sore that develops on the lining of the esophagus, stomach, or small intestine.",
        "medicines": ["Omeprazole", "Pantoprazole", "Ranitidine", "Esomeprazole", "Amoxicillin"],
        "specialists": ["Gastroenterologist", "General Physician", "Internal Medicine Specialist"]
    },
    "Type 2 Diabetes Mellitus": {
        "description": "A chronic condition that affects the way the body processes blood sugar (glucose).",
```

"medicines": ["Metformin", "Glipizide", "Insulin", "Sitagliptin", "Canagliflozin"],
"specialists": ["Endocrinologist", "Diabetologist", "Nutritionist"]
},
"Acute Myocardial Infarction": {
"description": "A medical emergency where the blood flow to the heart is blocked.",
"medicines": ["Aspirin", "Clopidogrel", "Statins", "Beta Blockers", "ACE Inhibitors"],
"specialists": ["Cardiologist", "Emergency Medicine Specialist"]
},
"Chronic Obstructive Pulmonary Disease": {
"description": "A group of lung diseases that block airflow and make breathing difficult.",
"medicines": ["Tiotropium", "Albuterol", "Ipratropium", "Fluticasone", "Salmeterol"],
"specialists": ["Pulmonologist", "General Physician", "Respiratory Therapist"]
},
"Cerebrovascular Accident (Stroke)": {
"description": "A condition caused by the interruption of blood flow to the brain.",
"medicines": ["Alteplase", "Aspirin", "Clopidogrel", "Warfarin", "Atorvastatin"],
"specialists": ["Neurologist", "Rehabilitation Specialist", "Neurosurgeon"]
},
"Deep Vein Thrombosis": {
"description": "A blood clot forms in a deep vein, usually in the legs.",
"medicines": ["Warfarin", "Heparin", "Apixaban", "Dabigatran", "Rivaroxaban"],
"specialists": ["Hematologist", "Vascular Surgeon", "Cardiologist"]
},
"Chronic Kidney Disease": {
"description": "The gradual loss of kidney function over time.",
"medicines": ["Erythropoietin", "Phosphate Binders", "ACE Inhibitors", "Diuretics", "Calcitriol"],
"specialists": ["Nephrologist", "Dietitian", "Internal Medicine Specialist"]
},
"Community-Acquired Pneumonia": {
"description": "A lung infection acquired outside of a hospital setting.",
"medicines": ["Amoxicillin", "Azithromycin", "Clarithromycin", "Ceftriaxone", "Levofloxacin"],
"specialists": ["Pulmonologist", "Infectious Disease Specialist", "General Physician"]
},
"Septic Shock": {
"description": "A severe infection leading to dangerously low blood pressure.",
"medicines": ["Norepinephrine", "Vancomycin", "Meropenem", "Hydrocortisone", "Dopamine"],
"specialists": ["Intensivist", "Infectious Disease Specialist", "Emergency Medicine Specialist"]
},
"Rheumatoid Arthritis": {
"description": "An autoimmune disorder causing inflammation in joints.",
"medicines": ["Methotrexate", "Sulfasalazine", "Hydroxychloroquine", "Adalimumab", "Etanercept"],
"specialists": ["Rheumatologist", "Orthopedic Specialist", "Physical Therapist"]
},
"Congestive Heart Failure": {
"description": "A chronic condition where the heart doesn't pump blood effectively.",
"medicines": ["ACE Inhibitors", "Beta Blockers", "Diuretics", "Spironolactone", "Digoxin"],
"specialists": ["Cardiologist", "General Physician", "Cardiac Surgeon"]
},
"Pulmonary Embolism": {
"description": "A blockage in one of the pulmonary arteries in the lungs.",

```
        "medicines": ["Heparin", "Warfarin", "Alteplase", "Rivaroxaban", "Dabigatran"],
        "specialists": ["Pulmonologist", "Hematologist", "Emergency Medicine Specialist"]
    },
    "Sepsis": {
        "description": "A life-threatening organ dysfunction caused by a dysregulated immune response to infection.",
        "medicines": ["Vancomycin", "Meropenem", "Piperacillin-Tazobactam", "Cefepime", "Dopamine"],
        "specialists": ["Infectious Disease Specialist", "Intensivist", "Emergency Medicine Specialist"]
    },
    "Liver Cirrhosis": {
        "description": "A late-stage liver disease caused by liver scarring and damage.",
        "medicines": ["Spironolactone", "Furosemide", "Lactulose", "Nadolol", "Rifaximin"],
        "specialists": ["Hepatologist", "Gastroenterologist", "Nutritionist"]
    },
    "Acute Renal Failure": {
        "description": "A sudden loss of kidney function.",
        "medicines": ["Diuretics", "Dopamine", "Calcium Gluconate", "Sodium Bicarbonate", "Epoetin"],
        "specialists": ["Nephrologist", "Critical Care Specialist", "Internal Medicine Specialist"]
    },
    "Urinary Tract Infection": {
        "description": "An infection in any part of the urinary system.",
        "medicines": ["Nitrofurantoin", "Ciprofloxacin", "Amoxicillin-Clavulanate", "Trimethoprim-Sulfamethoxazole",
"Cephalexin"],
        "specialists": ["Urologist", "General Physician", "Infectious Disease Specialist"]
    },
    "Hypertension": {
        "description": "A condition in which the force of the blood against the artery walls is too high.",
        "medicines": ["Lisinopril", "Amlodipine", "Losartan", "Hydrochlorothiazide", "Metoprolol"],
        "specialists": ["Cardiologist", "General Physician", "Nephrologist"]
    },
    "Asthma": {
        "description": "A condition in which the airways narrow and swell, causing difficulty in breathing.",
        "medicines": ["Albuterol", "Fluticasone", "Montelukast", "Budesonide", "Salmeterol"],
        "specialists": ["Pulmonologist", "Allergist", "General Physician"]
    },
    "Gastroesophageal Reflux Disease (GERD)": {
        "description": "A digestive disorder where stomach acid irritates the esophagus.",
        "medicines": ["Omeprazole", "Esomeprazole", "Ranitidine", "Lansoprazole", "Pantoprazole"],
        "specialists": ["Gastroenterologist", "General Physician", "Dietitian"]
    }
}

# Extended clean_text function with more steps
def clean_text(text):
    stop_words = set(stopwords.words('english'))

    # Convert to string and lowercase the text
    text = str(text).lower()

    # Remove any numbers (you may want to modify this if numbers are important)
    text = re.sub(r'\d+', '', text)
```

```
    # Remove special characters, punctuation, and non-alphabetical characters
    text = re.sub(r'[^a-z\s]', '', text)

    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()

    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])

    return text

# Function to make prediction
def predict_disease(patient_note, model, tokenizer, label_encoder):
    patient_note = clean_text(patient_note)

    # Tokenize the input patient note
    inputs = tokenizer(patient_note, return_tensors="pt", padding=True, truncation=True, max_length=512)

    # Make prediction
    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits

    # Get the predicted label
    predicted_label = torch.argmax(logits, dim=1).item()

    # Convert the predicted label to the corresponding disease name
    predicted_disease = label_encoder.inverse_transform([predicted_label])[0]

    return predicted_disease


# Route for rendering the index page
@app.get("/", response_class=HTMLResponse)
async def upload_form(request: Request):
    return templates.TemplateResponse("index.html", {"request": request})



# Function to get disease details
def get_disease_details(disease_name):
    if disease_name in disease_data:
        return disease_data[disease_name]
    return {
        "description": "No details available for this disease.",
        "medicines": [],
        "specialists": []
    }
```

```
# Updated predict endpoint
@app.post("/predict/")
async def predict(file: UploadFile = File(...)):
    content = await file.read()
    text = ""

    # Extract text from PDF or TXT file
    if file.filename.endswith(".pdf"):
        pdf_reader = PyPDF2.PdfReader(BytesIO(content))
        for page in pdf_reader.pages:
            text += page.extract_text()
    elif file.filename.endswith(".txt"):
        text = content.decode("utf-8")

    # Predict disease
    predicted_disease = predict_disease(text, model, tokenizer, label_encoder)
    disease_details = get_disease_details(predicted_disease)

    # Return result
    return JSONResponse(content={
        "predicted_disease": predicted_disease,
        "description": disease_details["description"],
        "medicines": disease_details["medicines"],
        "specialists": disease_details["specialists"]
    })

# Run the application with Uvicorn
# Command: uvicorn app:app --reload
```

V. RESULT

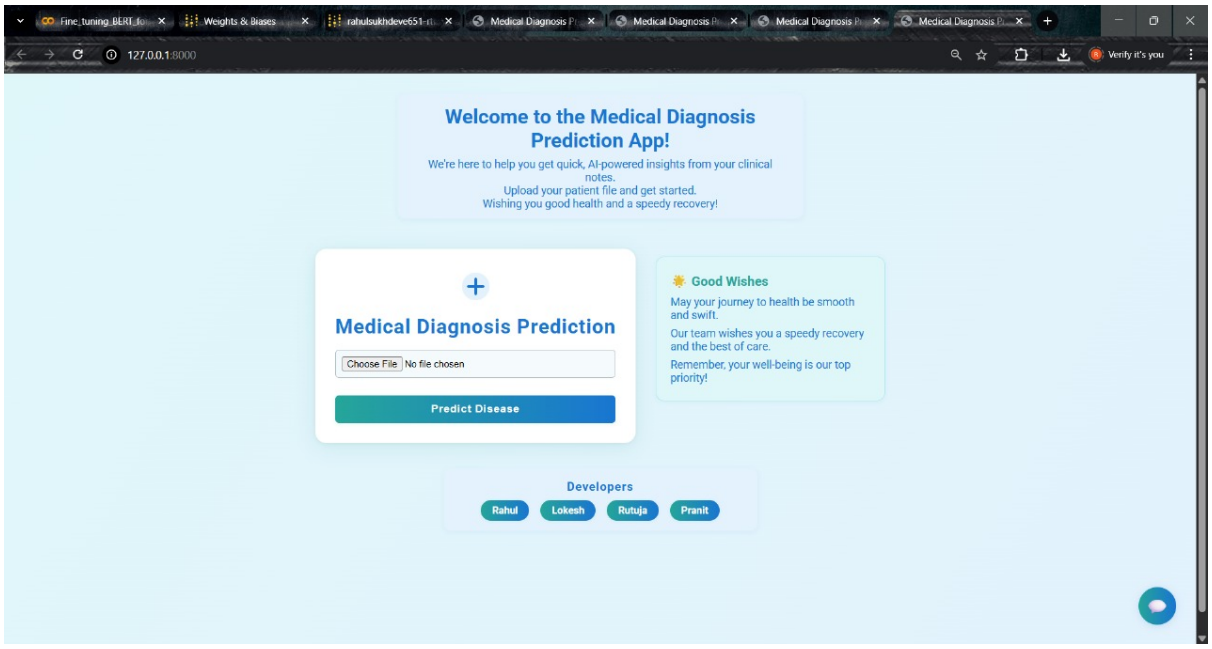


Fig 5.1 : - Before Report Uploading

The images show the interface of the Medical Diagnosis Prediction App, designed with a clean medical-themed layout. In the first screen, the user is welcomed with a friendly message and provided with an option to upload a patient report using the *Choose File* button. Once the file is selected, the user can click the *Predict Disease* button to start the AI-based analysis. A "Good Wishes" panel appears on the right side, offering positive and supportive messages, and the bottom section displays the names of the developers.

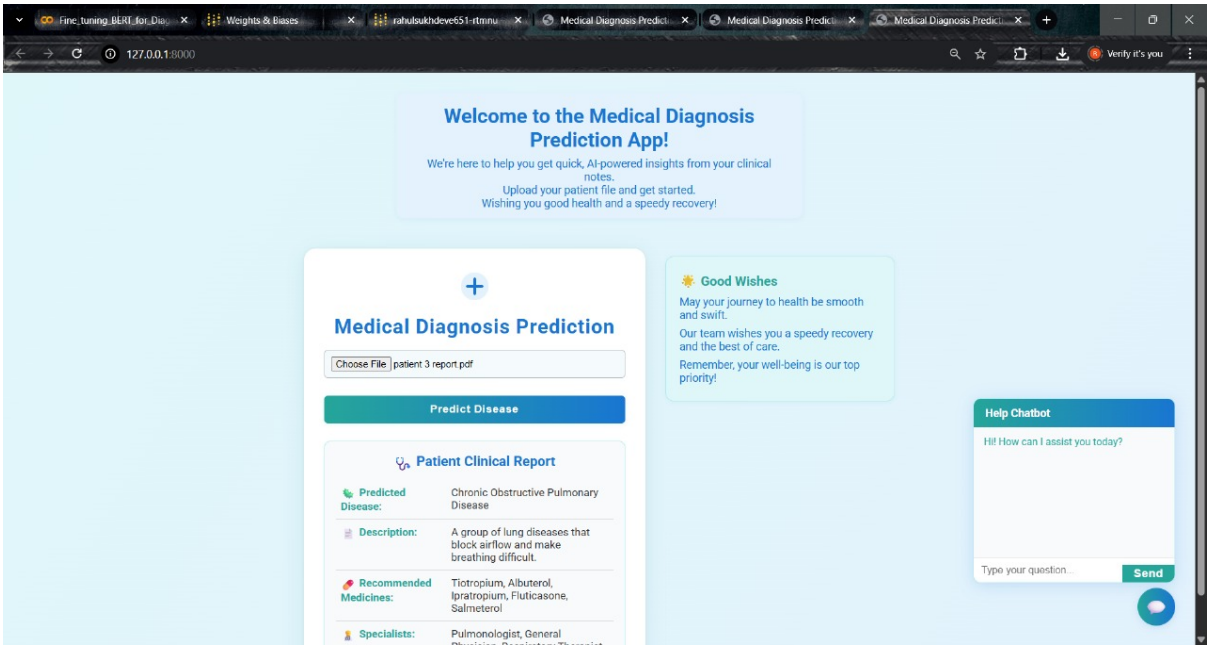


Fig 5.2 : - After Uploading the report

In the second screen, after uploading patient 3 report.pdf, the app displays the generated Patient Clinical Report, which includes the predicted disease (COPD), a short description, recommended medicines, and suggested specialists. The design remains simple and user-friendly. On the right side of the interface, a Help Chatbot is available, greeting the user and allowing them to ask questions for assistance. The chatbot enhances the usability of the application by providing quick support and guidance.

VI. FUTURE SCOPE

1) Integration of advanced AI and machine learning models for improved prediction accuracy.
2) Expansion of the dataset to make the system more robust and reliable.
3) Real-time data processing for faster and more efficient results.
4) Development of a mobile application or web interface for easier accessibility.
5) Addition of predictive analytics and reporting features for better decision-making.
6) Scalability improvements to handle larger datasets and multiple users.
7) Incorporation of user feedback mechanisms to enhance usability and functionality.
8) Implementation of cloud-based solutions for better performance and storage.

VII. ACKNOLEDGMENT

The attainment and final outcome of this project required a lot of supervision and assistance from many people and I am extremely privileged to have got this all along the completion of our project. Whatever I have done is only due to such guidance and assistance and I would not forget to express thanks to them.
My earnest gratitude goes to my Guide, Prof. Tara Shende Assistant Professor, Department of ArtificialIntelligence&DataScience Engineering, for her support, guidance, inspiration and encouragement throughout the period this work was carried out. Her willingness for meeting at all times, her lucrative comments, her concern and support even with practical things have been very helpful.
I express my gratitude to our H.O.D., Prof. Aparna Gale for her encouragement and guidance to complete our project work.

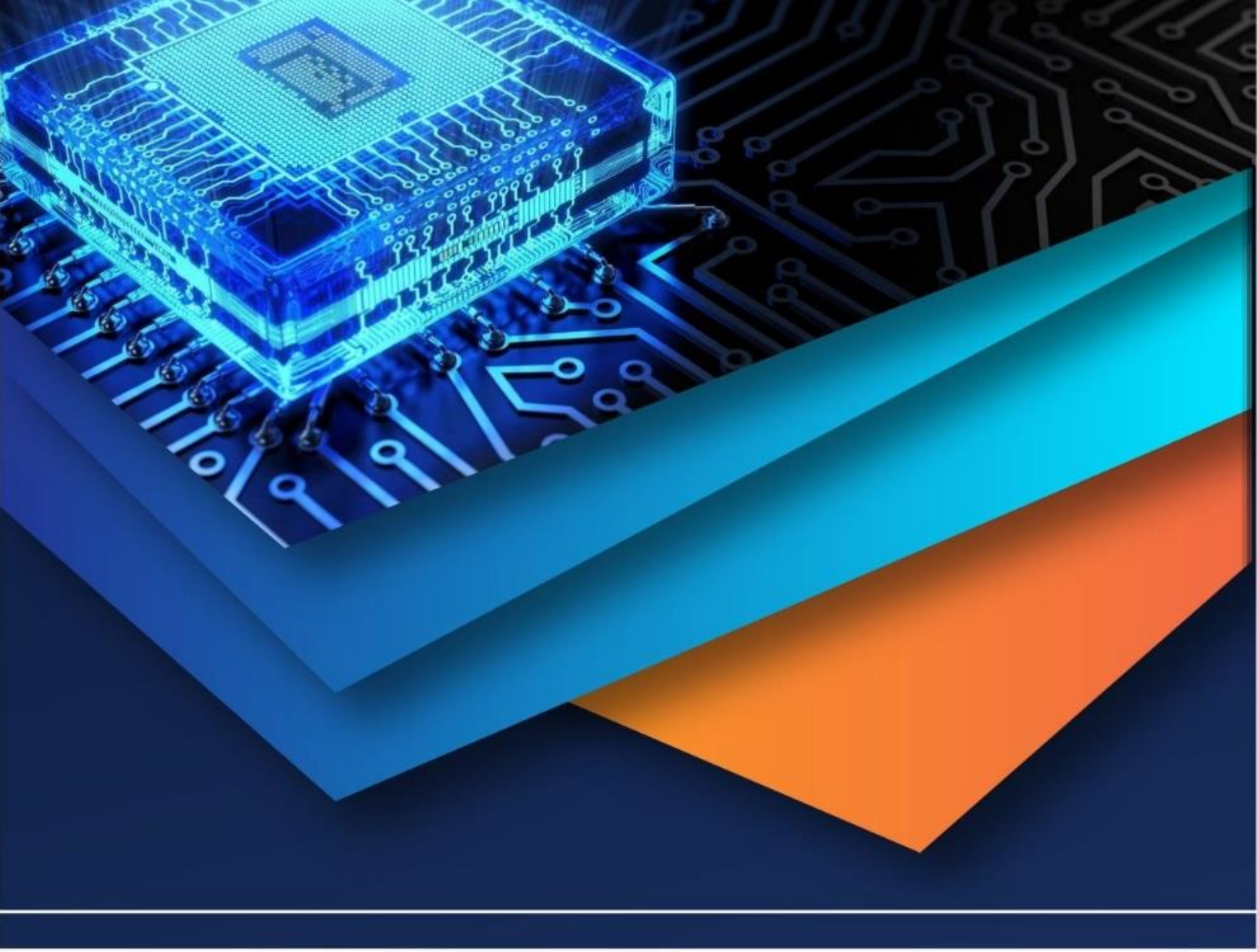
My sincere thanks to the higher authorities and Director, Dr. Sangita Deshmukh for providing me necessary facilities to carry out the work.

I would also like to thank all teaching and non-teaching staff members for their constant support and timely help in various ways for the completion of this thesis.

Last, but not the least, I would like to a vote of appreciation for my Parents and fellow friends for their cooperation.

REFERENCES

[1] AI-Based Multi-Disease Prediction System — Tiwari K., Dubey V., Gupta V. (2024). Random Forest for multi-disease prediction. Link
[2] Deep Learning Framework for Medical Diagnosis — Subramani S., Varshney N., Anand M. V. (2023). CNN-based medical diagnosis. Link
[3] Machine Learning for Early Cancer Risk Detection — Majhi B., Kashyap A. (2024). XGBoost for cancer risk prediction. Link
[4] Explainable AI for Heart Disease Detection — Majhi B., Kashyap A. (2024). XAI + Logistic Regression for interpretable heart disease prediction. Link
[5] IoT and ML-Based Diabetes Monitoring System — Menon S. P., Shukla P. K., Sethi P. (2023). IoT + LSTM for real-time diabetes monitoring. Link
[6] Hybrid ML Model for Heart Disease Prediction — Lamir A.A., Razzagzadeh S., Rezaei Z. (2025). SVM + Random Forest for improved heart disease prediction. Link



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY