A Project Report on

"Medical Recommendation System"

Submitted in the partial fulfilment for the award of degree for

Master of Computer Applications

Submitted By

Pratyak Mohapatra

(2370300)

Under the guidance of

Prof. sumit kumar Tetarave

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

School of Computer Applications

Bhubaneswar, Odisha – 751024

April 2025

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled "**Medical Recommendation System**" submitted to **School of Computer Applications, KIIT University** in partial fulfilment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS (MCA),** is an authentic and original work carried out by Master. **Pratyak Mohapatra ,**with Roll no. **2370300** and Regd. No. **237261103475** under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirements of any course of study.

…..…… ……..

Signature of the Student Signature of the Guide

Name:

Designation:

Date:……. Date:……..

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

CERTIFICATE

This is to certify that the project work entitled **"Medical Recommendation System"** Submitted by Master. **Pratyak Mohapatra** bearing roll no. **2370300,** is an authentic and original work.

Signature:

(Internal Examiner)

Date:…….

Name:

Designation:

Signature:

(External Examiner)

Date:……..

Name:

Designation:

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

DECLARATION

I, **Pratyak Mohapatra**, roll no **2370300** do hereby declare that the project report entitled "**Medical Recommendation System"** submitted to **School of Computer Applications, KIIT University, Bhubaneswar** for the award of the degree of **Master of Computer Applications (MCA) ,** is an authentic and original work carried out by me and my group members *(Pratyak Mohapatra 2370300, Pratyak Mahapatra 2370300, Pravat Das 2370303)* from 2nd January 2025 to 25th April 2025 at **School of Computer Applications, KIIT University** under the guidance of **Prof. Sumit kumar Tetarave** .

……

Signature of the student

Date: .....

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

ACKNOWLEDGEMENT

This satisfaction which accompanies the successful completion of any task is incomplete without the mention of those persons whose hands are behind the success. Because the success is the epitome of hard work, prevention, zeal, determination and the most encouraging guidance and advice serving as beacon of light and crowing our effort with success.

I am grateful to **Prof. Veena Goswami** , the Director General of SCA, and **Prof. Satya Ranjan Dash**,the Dean of SCA, KIIT Deemed to be University for the endless support and kind cooperation for completion of this project. I am also thankful to **Prof. Sumit kumar Tetarave**, KIIT Deemed to be University for his endless support and kind cooperation for completion of this project. I am also thankful to the faculty members of KIIT Deemed to be University who have constantly strived to support us.

……

Date: Signature of the Student

Place: Name of the Student:

Roll_no:

ABSTRACT

The exponential growth of medical data and the need for personalized healthcare have driven the development of intelligent recommendation systems. This project presents a Healthcare Recommendation System that leverages machine learning algorithms to assist in medical decision-making by providing tailored suggestions based on patient data.

The system analyzes various health parameters such as symptoms, patient history, and lifestyle information to generate accurate and relevant medical recommendations. By employing supervised learning techniques, the system is trained on real-world healthcare datasets to identify patterns and correlations within medical conditions and treatments.

The goal is to support healthcare professionals and patients in making informed decisions, improving diagnosis efficiency, and enhancing overall healthcare outcomes. The results demonstrate that machine learning can significantly contribute to delivering timely and personalized medical recommendations with high accuracy and reliability.

CONTENTS

INTRODUCTION

With the rapid advancements in Machine Learning (ML), healthcare systems have leveraged data-driven approaches to enhance disease diagnosis, treatment, and patient care. The integration of machine learning in medical decision-making enables the prediction of diseases based on symptoms, thereby assisting individuals in taking proactive health measures.

The Medical Recommendation System with Machine Learning is designed to predict potential diseases based on user-input symptoms and provide personalized health recommendations, including diet plans, medications, precautions, symptoms, and workout routines**.** In today's digital age, individuals often rely on online sources for self-diagnosis**,** which can lead to misinformation. This project aims to bridge this gap by offering a reliable machine learning-based recommendation system that enhances accessibility to preliminary healthcare insights before consulting medical professionals.

Project Overview

This project utilizes Training.csv as the primary dataset for data preprocessing, descriptive analysis, and predictive modelling**.** The system follows a structured workflow:

1. Data Preprocessing: Cleaning and transforming raw data for better model accuracy.
2. Descriptive Analysis: Exploring the relationship between symptoms and diseases through statistical and visual analysis.
3. Model Development: Implementing and training machine learning models to predict diseases.
4. System Deployment: Integrating the trained model into a web-based application for user interaction.

Once the user inputs symptoms, the system predicts the most probable disease and provides relevant recommendations, including diets, medications, precautions, and suitable workout plans**.**

Technologies and Tools

To develop an efficient and user-friendly Medical Recommendation System with Machine Learning**,** the following tools and technologies are used:

- Development Platforms: IntelliJ IDEA, Jupyter Notebook
- Programming Languages: Python (for backend and machine learning), HTML, CSS, JavaScript, Bootstrap (for frontend)
- Data Analysis & Visualization Libraries: Pandas, NumPy, Seaborn, Matplotlib
- Machine Learning Frameworks: Scikit-learn (sklearn)

The system follows a train-test approach, where multiple machine learning models are trained and evaluated to identify the most accurate model for disease prediction.

Significance of the Project

The Medical Recommendation System with Machine Learning aims to provide quick, accurate, and accessible healthcare insights, empowering individuals to make informed decisions about their health. Unlike traditional medical consultations that require physical visits and time-consuming diagnostic procedures, this system offers a machine learning-based approach to early disease detection and preventive care**.**

Additionally, healthcare professionals can use this system as a decision-support tool, improving diagnosis efficiency and patient management. The implementation of machine learning in disease prediction and medical recommendations enhances early detection, promotes lifestyle modifications, and contributes to better patient outcomes**.**

Conclusion

The Medical Recommendation System with Machine Learning represents an innovative step toward data-driven healthcare solutions. By integrating data analytics, machine learning, and web technologies, the system offers a reliable and user-friendly disease prediction and recommendation platform**.** This project enhances preliminary healthcare accessibility and encourages proactive health management, helping individuals take necessary precautions and make informed decisions about their well-being.

SYSTEM ANALYS

1. Identification of Need

The growing reliance on technology in the healthcare sector has led to the development of machine learning-based solutions that assist in early disease detection and personalized recommendations. Many individuals rely on online resources for self-diagnosis, which may not always be reliable. The Medical Recommendation System with Machine Learning addresses this issue by providing accurate disease predictions and tailored health recommendations based on user-input symptoms.

This system is designed to:

- Help users identify potential diseases based on their symptoms**.**
- Provide recommendations for diets, medications, precautions, and workouts**.**
- Reduce dependency on unreliable sources for self-diagnosis.
- Assist healthcare professionals by serving as a reference tool for symptom analysis.

a. Existing System Requirement

In the traditional healthcare system, diagnosing a disease requires physical consultations, laboratory tests, and expert medical evaluations. This approach has certain limitations:

- Time-consuming: Patients must schedule appointments and wait for test results.
- Limited accessibility: Not all individuals have immediate access to medical facilities.
- High costs: Medical tests and consultations can be expensive.
- Risk of self-misdiagnosis: Online searches often lead to incorrect or misleading health information.

b. How Computerization is Useful

The Medical Recommendation System with Machine Learning overcomes the limitations of traditional methods by offering:

- Instant Disease Prediction: Users receive predictions based on symptoms in real time.
- ML-Based Recommendations: The system suggests personalized treatment options, including diets, medications, and workouts.
- Remote Accessibility: Users can access healthcare insights from anywhere through a web-based platform.
- Cost-Effective Solution: Reduces the need for unnecessary medical visits by providing preliminary health analysis.

2. Feasibility Study

The feasibility study evaluates whether the proposed system is technically, operationally, and economically viable for implementation.

a. Technical Feasibility

This project utilizes well-established technologies and frameworks for data processing, machine learning, and web development:

- Programming Languages: Python (for backend and ML), HTML, CSS, JavaScript (for frontend).
- Libraries and Frameworks: Pandas, NumPy, Scikit-learn, Seaborn, Matplotlib.
- Development Tools: IntelliJ IDEA, Jupyter Notebook.
- Deployment Environment**:** Web-based interface for user interaction.

Given the availability of robust open-source tools and frameworks, the Medical Recommendation System with Machine Learning is technically feasible and can be implemented effectively.

b. Operational Feasibility

This system is designed to be user-friendly and efficient for both medical professionals and general users. Key operational benefits include:

- Ease of Use: A simple symptom input interface allows users to interact with the system easily.
- Automated Predictions: The machine learning-based system quickly analyzes symptoms and provides recommendations.
- Minimal Maintenance: The system requires periodic updates to enhance model accuracy but does not demand high operational costs.

With an intuitive web-based UI and backend integration, the system is operationally feasible and can be smoothly deployed.

c. Economic Feasibility

The economic feasibility examines the cost-effectiveness of developing and maintaining the system. Major cost components include:

- Development Costs: Initial investment in data collection, preprocessing, and model training**.**
- Operational Costs: Hosting the web application and maintaining the ML model.
- Long-Term Benefits: The system reduces healthcare costs by minimizing unnecessary medical visits and providing preventive healthcare insights.

Since the project utilizes open-source tools and frameworks, development costs are relatively low, making the system economically feasible. Additionally, it can generate revenue through premium features, API integrations, or partnerships with healthcare providers**.**

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

1. Introduction

The Medical Recommendation System with Machine Learning is a healthcare application that predicts diseases based on user-input symptoms and provides personalized recommendations, including diet plans, medications, precautions, and workout routines**.** The system leverages machine learning models to analyse symptoms and offer accurate healthcare insights, making it a valuable tool for both individuals and healthcare professionals.

2. Purpose

The primary goal of this system is to develop a machine learning-based healthcare assistant that:

- Predicts diseases based on user-provided symptoms.
- Recommends appropriate diets, medications, precautions, and workout plans.
- Enhances accessibility to preliminary medical diagnosis.
- Reduces dependency on self-diagnosis through unreliable online sources.
- Provides a user-friendly web-based interface for easy access to healthcare recommendations.

3. Scope

The Medical Recommendation System with Machine Learning is designed to provide accurate, data-driven health insights through a user-friendly web platform. The system includes:

- Symptom-based disease prediction using machine learning models.
- Health recommendations, including diet plans, medications, and precautions.
- A web-based interface for users to enter symptoms and receive medical insights.
- Data preprocessing and descriptive analysis to improve model accuracy.
- Supervised learning algorithms to classify diseases based on symptom patterns.
- Scalability to integrate with external healthcare databases or APIs in the future.

4. Benefits

For Users (Patients & General Public)

- Quick and reliable disease prediction based on symptoms.
- Instant medical recommendations without visiting a doctor.
- Remote accessibility via a web-based interface.
- Cost-effective alternative to initial medical consultations.

For Healthcare Professionals

- Acts as a decision-support system by analysing symptoms efficiently.
- Reduces unnecessary hospital visits for minor conditions.
- Helps in early detection of diseases, leading to timely treatment.

For the Healthcare Industry

- Promotes machine learning-driven diagnostics for enhanced healthcare accessibility.
- Provides data-driven insights for medical research and disease pattern analysis.

5. Overall Description

The Medical Recommendation System with Machine Learning functions as a machine learning-powered healthcare assistant that takes user-input symptoms, processes the data using classification algorithms**,** and provides disease predictions along with medical recommendations.

- Data Source**:** Uses the Training.csv dataset for data preprocessing, descriptive analysis, and predictive modeling.
- Technology Stack**:** Built using Python (backend & ML), HTML, CSS, JavaScript, Bootstrap (frontend).
- Machine Learning Models**:** Various classification models such as SVC, Random Forest, Gradient Boosting, K-Neighbors, and GaussianNB are used.
- User Interface**:** A web-based application where users input symptoms and receive predictions.
- Output**:** Disease prediction along with recommended treatments, precautions, and workout plans.

This system represents an innovative step toward machine learning-based healthcare solutions**,** making health insights more accessible, data-driven, and user-friendly.

SOFTWARE ENGINEERING PARADIGM APPLIED:

Agile Model

The Agile model emphasizes iterative development, frequent user feedback, and adaptive planning. Each iteration in this project builds upon the previous one, gradually adding functionality and improving system performance.

Iteration-wise Breakdown:

1 – Dataset Integration & Preprocessing

Planning:

- Identify and gather the dataset (Training.csv)
- Define preprocessing steps (handle missing values, normalize data)

Requirement Analysis:

- Understand the dataset structure and target (Disease prediction)
- Identify preprocessing needs

Design:

- Design data pipeline structure for cleaning and transformation

System Building Phase:

- Implement data loading, cleaning, label encoding, and transformation

System Testing Phase:

- Verify that the cleaned data is in the desired format and free of inconsistencies

2 – Exploratory Data Analysis (EDA)

Planning:

- Explore relationships between symptoms and diseases

Requirement Analysis:

- Identify visual and statistical techniques for data insights

Design:

- Design EDA reports using graphs and summaries

System Building Phase:

- Use Pandas, Seaborn, Matplotlib to create heatmaps, correlation plots, etc.

System Testing Phase:

- Validate findings with sample cases, peer review of data patterns

3 – Model Development & Evaluation

Planning:

- Select ML algorithms for disease prediction

Requirement Analysis:

- Choose models (e.g., Decision Tree, Random Forest, Naive Bayes, etc.)

Design:

- Design training-validation split
- Define accuracy metrics

System Building Phase:

- Train models using scikit-learn
- Evaluate performance using cross-validation

System Testing Phase:

- Test with unseen data, compare accuracy, precision, recall

4 – Web UI & Basic Integration

Planning:

- Plan frontend and backend integration

Requirement Analysis:

- Determine key UI inputs (symptoms) and expected outputs (predictions, recommendations)

Design:

- UI wireframes
- Backend API endpoints for predictions

System Building Phase:

- Build HTML/CSS/JS UI with Bootstrap
- Flask backend to connect model to UI

System Testing Phase:

- Test frontend form submissions, model integration, and prediction output

5 – Recommendation Module

Planning:

- Plan personalized health recommendations (diet, precautions, workouts)

Requirement Analysis:

- Curate recommendations for each predicted disease

Design:

- Map predicted diseases to recommendation content

System Building Phase:

- Implement logic to display recommendations post-prediction

System Testing Phase:

- Validate that correct recommendations appear for each disease

6 – Final Deployment & Feedback

Planning:

- Prepare the system for deployment and collect user feedback

Requirement Analysis:

- Final check of all features and scalability options

Design:

- Deployment pipeline design (e.g., using Heroku/Render/Flask Server)

System Building Phase:

- Deploy complete application
- Integrate model, UI, and recommendations

System Testing Phase:

- Perform usability testing, collect feedback from users
- Fix bugs and optimize performance

Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The Dfd does not mention anything about how data flows through the system. Admin (module) login add members detail share and give feedback share & give feedback application home page member (module) login view own details pay individual maintenance fees view member details & their maintenancefees18thereisaprominentdifferencebetweendfdandflowchart. The flowchart depicts flow of control in program modules. Dfds depict flow of data in the system at various levels. Dfd does not contain any control or branch elements.

· **Types of DFD**

Data Flow Diagrams are either Logical or Physical.

· **Logical DFD**

This type of dfd concentrates on the system processes, and flow of data in the system. For example in a banking software system, how data is moved between different entities.

· **Physical DFD**

This type of dfd shows how the data flow is actually implemented in the system. It is more specific and close to the implementation

DFD Components

DFD can represent source, destination, storage and flow of data using the following set of components

Figure-1

Entities:

Entities are source and destination of information data. Entities are represented by a rectangle with their respective names.

Process:

Activities and action taken on the data are represented by circle or round-edged rectangles.

Data Storage:

There are two variants of data storage- it can either be represented as a rectangle with absence of both smaller sides or as an open sided rectangle with only one side missing.

Data Flow:

Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Context level DFD:

The context level DFD is showing the overall process which can be seen in a system, so news application two modules are here who can interact with the application and operating all the process in this system as shown in the context level DFD below:

Context DFD (Level 0)

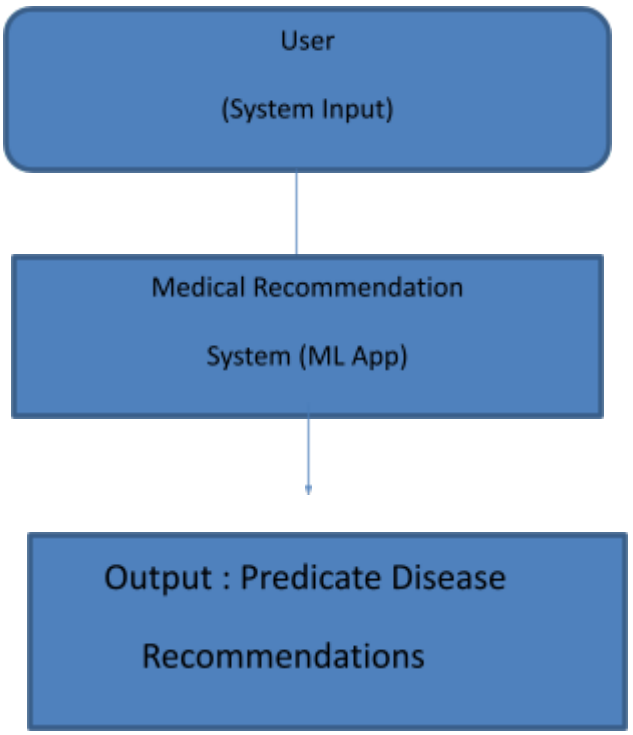


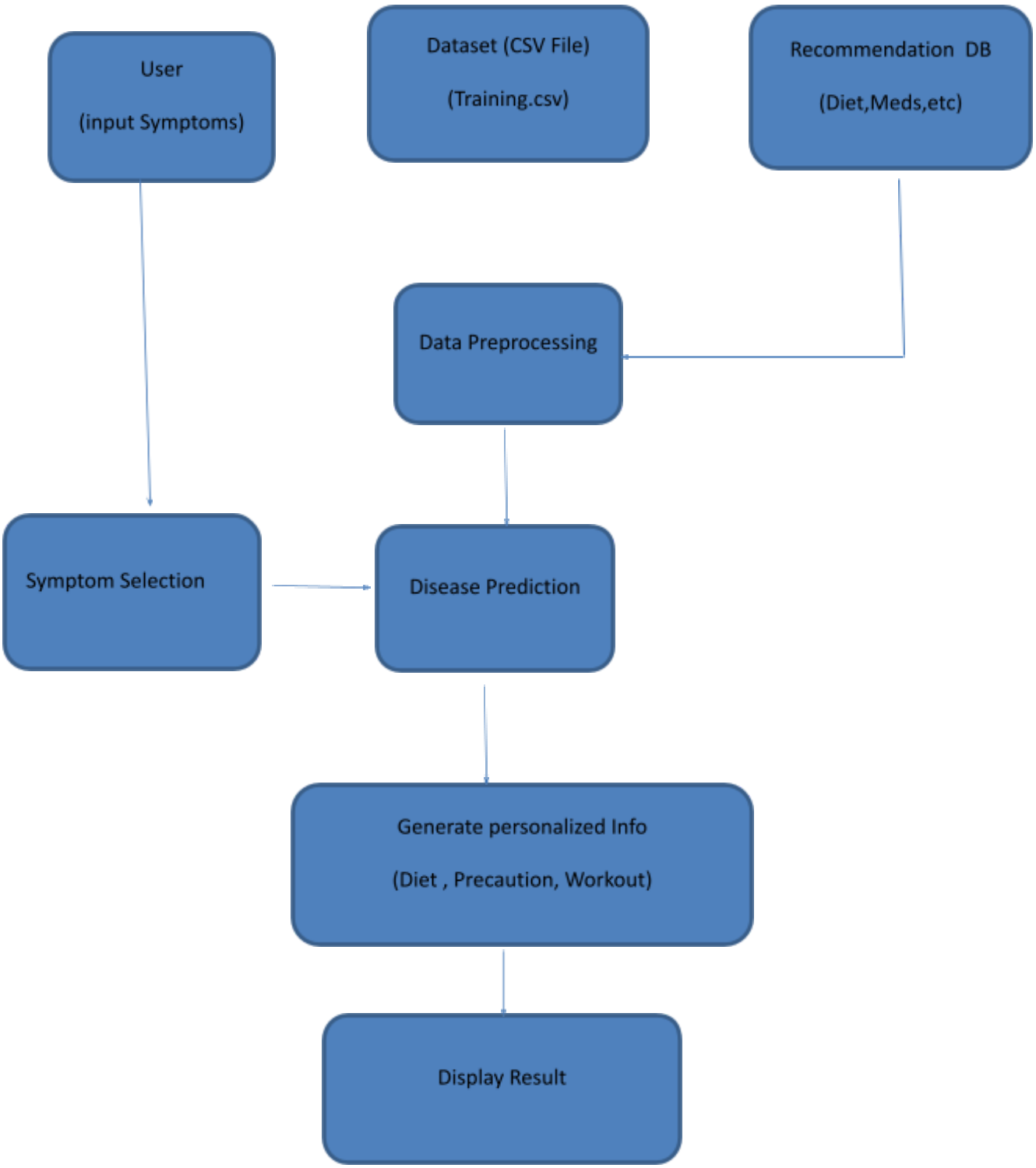
Figure-2

Level 1 DFD:

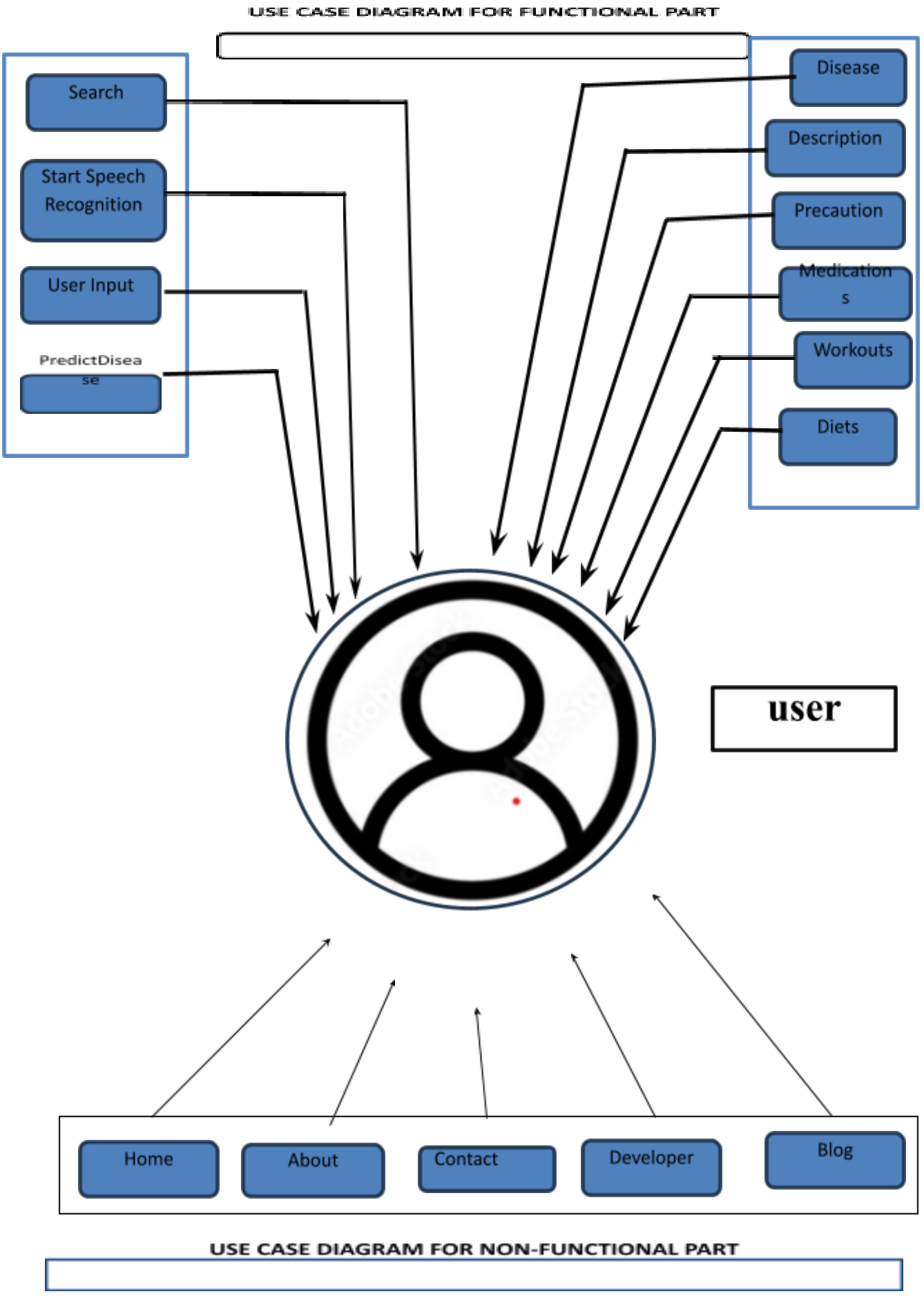
Context level of DFD broken down into sub– processes consists in

the news application, sub– processes are shown in the (figure 3)

below:



Use case diagram for News Application (figure-4)



System Design

User Interface Design:

Create a simple user interface where users (patients/doctors) can enter symptoms and get recommendations.

Tools used are Flask + HTML/CSS/JS.

Goal: Make the system easy and accessible to use and to make the system user-friendly, especially for non-technical users.

Register Page:

Using the register page we can register the name of the patient so that we can keep the record of the patient. we can record the email and password so that only data of authenticate user can be recorded.

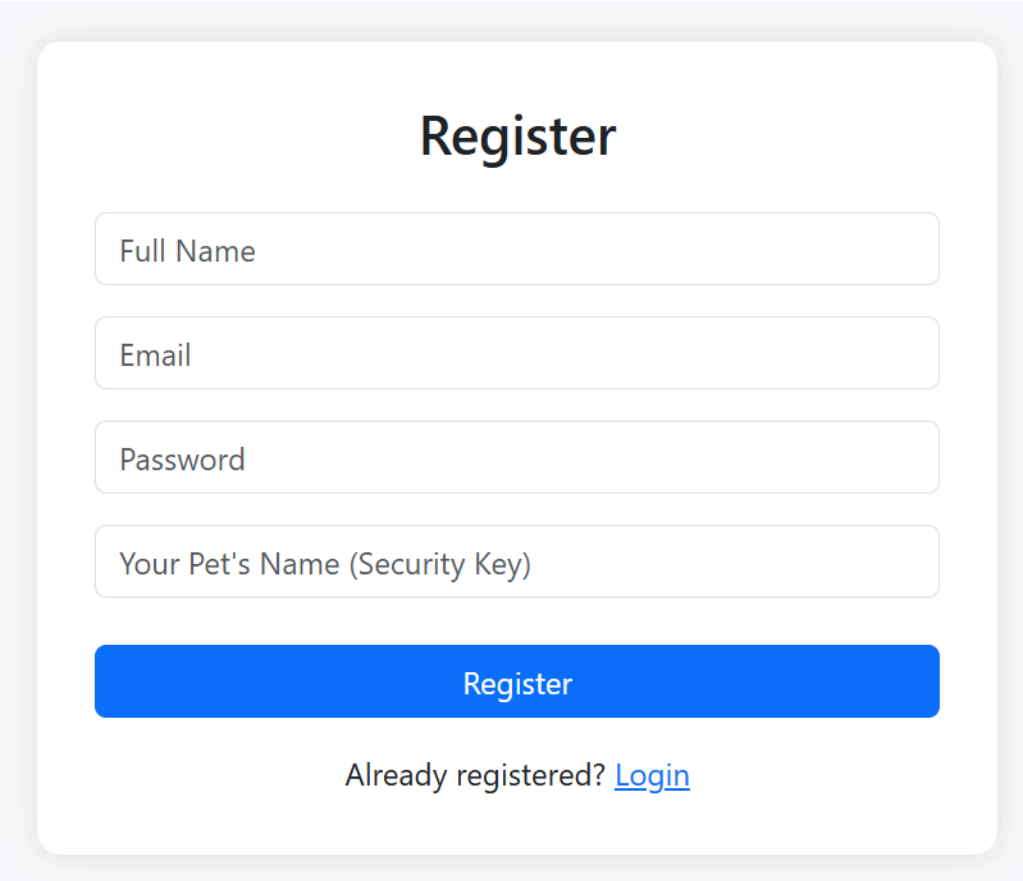


Figure5

Code:

```
<div class="container mt-5">
  <h3 class="text-center mb-4">Register</h3>
  <form action="register.php" method="post">
    <input type="text" class="form-control mb-3" name="name" placeholder="Full
Name" required>
    <input type="email" class="form-control mb-3" name="email" placeholder="Email"
required>
    <input type="password" class="form-control mb-3" name="password"
placeholder="Password" required>
    <input type="text" class="form-control mb-4" name="pet" placeholder="Your Pet's
Name (Security Key)" required>
    <button type="submit" class="btn btn-primary w-100">Register</button>
    <div class="text-center mt-3">
      Already registered? <a href="login.html">Login</a>
    </div>
  </form>
</div>
```

Login page:

Login page is used so that only authenticate user can use the page with proper authentication like using password.

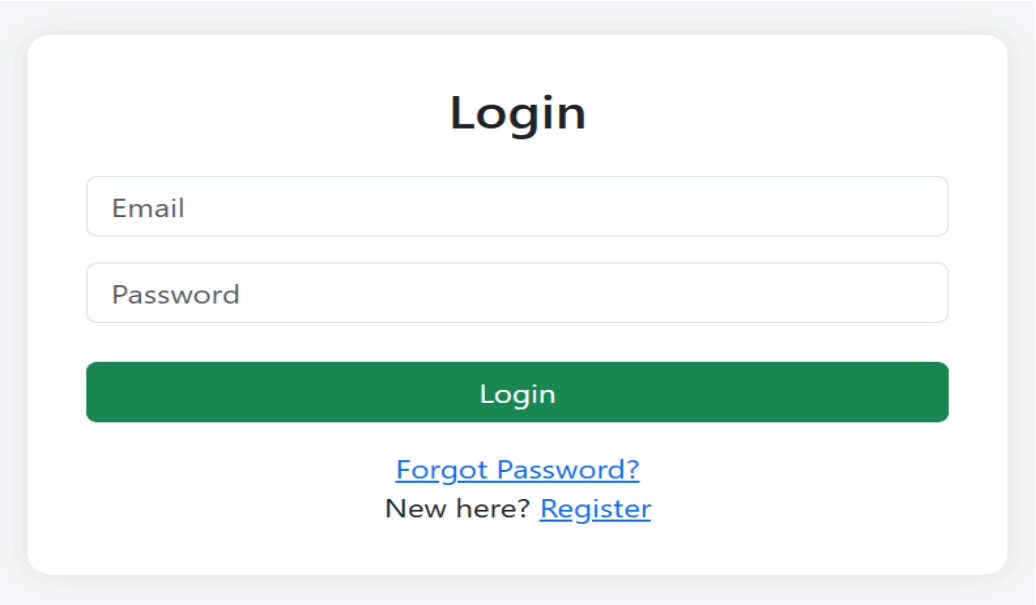


Figure 6

Code:

```
<div class="container mt-5">
  <h3 class="text-center mb-4">Login</h3>
  <form action="login.php" method="post">
    <input type="email" class="form-control mb-3" name="email" placeholder="Email"
required>
    <input type="password" class="form-control mb-4" name="password"
placeholder="Password" required>
    <button type="submit" class="btn btn-success w-100">Login</button>
    <div class="text-center mt-3">
      <a href="forgot_password.html">Forgot Password?</a><br>
      New here? <a href="register.html">Register</a>
    </div>
  </form>
</div>
```

Reset Password:

If any user if forgot the password then this page help to reset password using some authentic data that is only known to the user like pet's name so that password can reset and the authentic user can access the account .

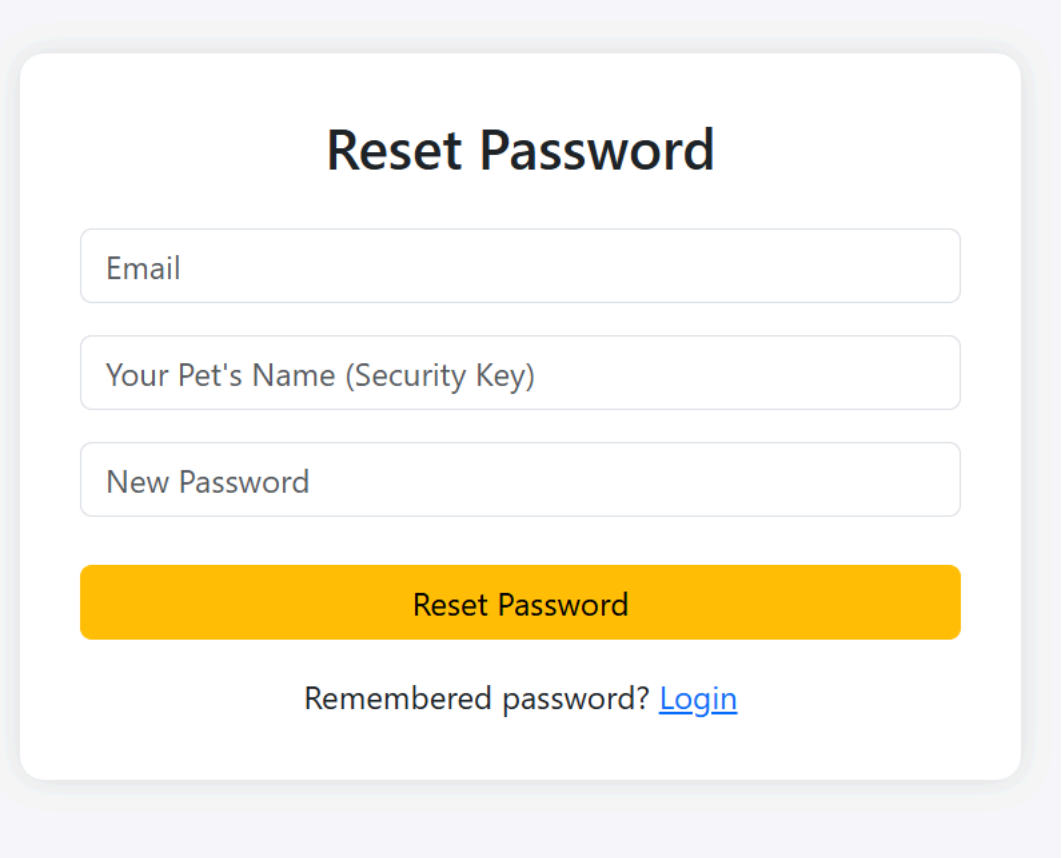


Figure 7

Code:

```
<div class="container mt-5">
  <h3 class="text-center mb-4">Reset Password</h3>
  <form action="forgot_password.php" method="post">
    <input type="email" class="form-control mb-3" name="email" placeholder="Email"
required>
    <input type="text" class="form-control mb-3" name="pet" placeholder="Your Pet's
Name (Security Key)" required>
    <input type="password" class="form-control mb-4" name="new_password"
placeholder="New Password" required>
    <button type="submit" class="btn btn-warning w-100">Reset Password</button>
    <div class="text-center mt-3">
      Remembered password? <a href="login.html">Login</a>
    </div>
  </form>
</div>
```

HOME PAGE:

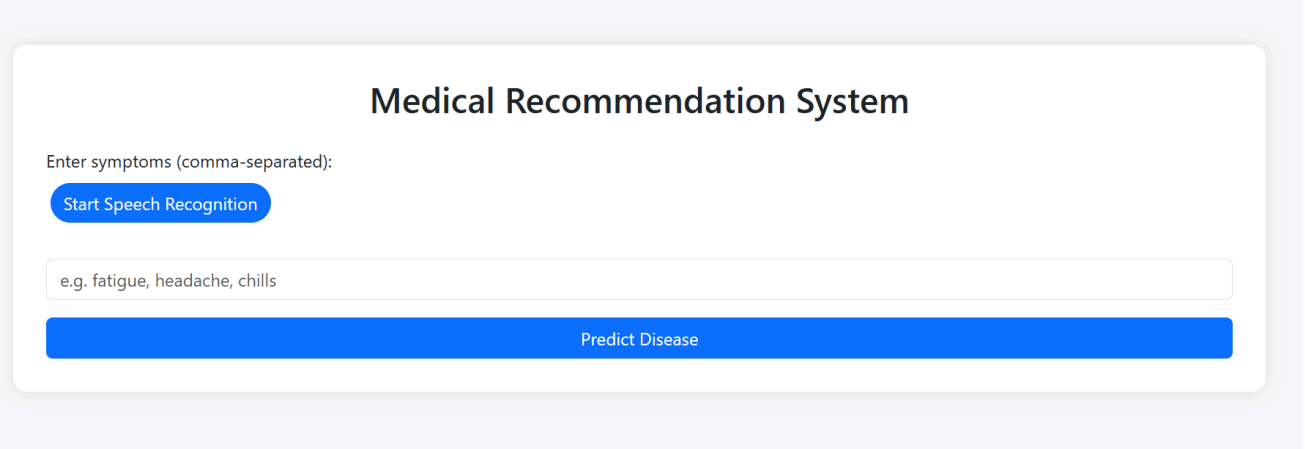


Figure-8

In the home page we can get all information and required items ,such as search button, predict button ,speech recognition button so that we can enter our requirement and the machine will suggest us all the results.

ABOUT:

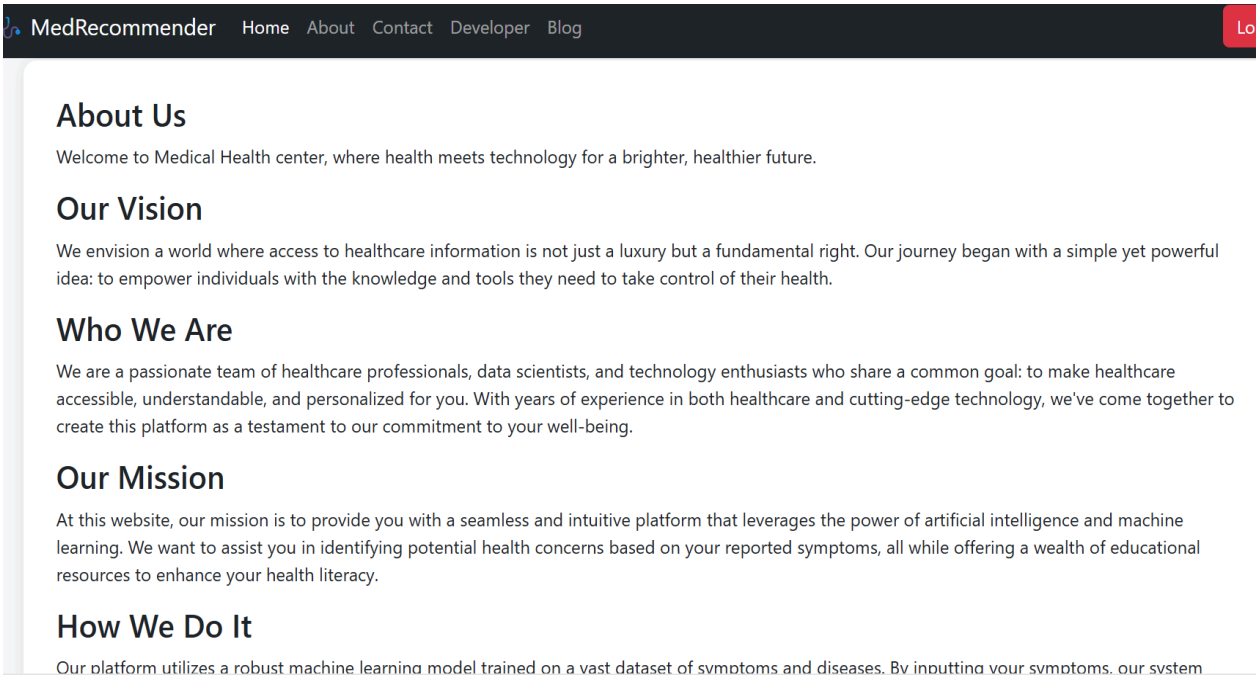


Figure 9

In the about page there is information about who we are , our vision, our mission and how we did our project ,different processes etc.

CONTACT:

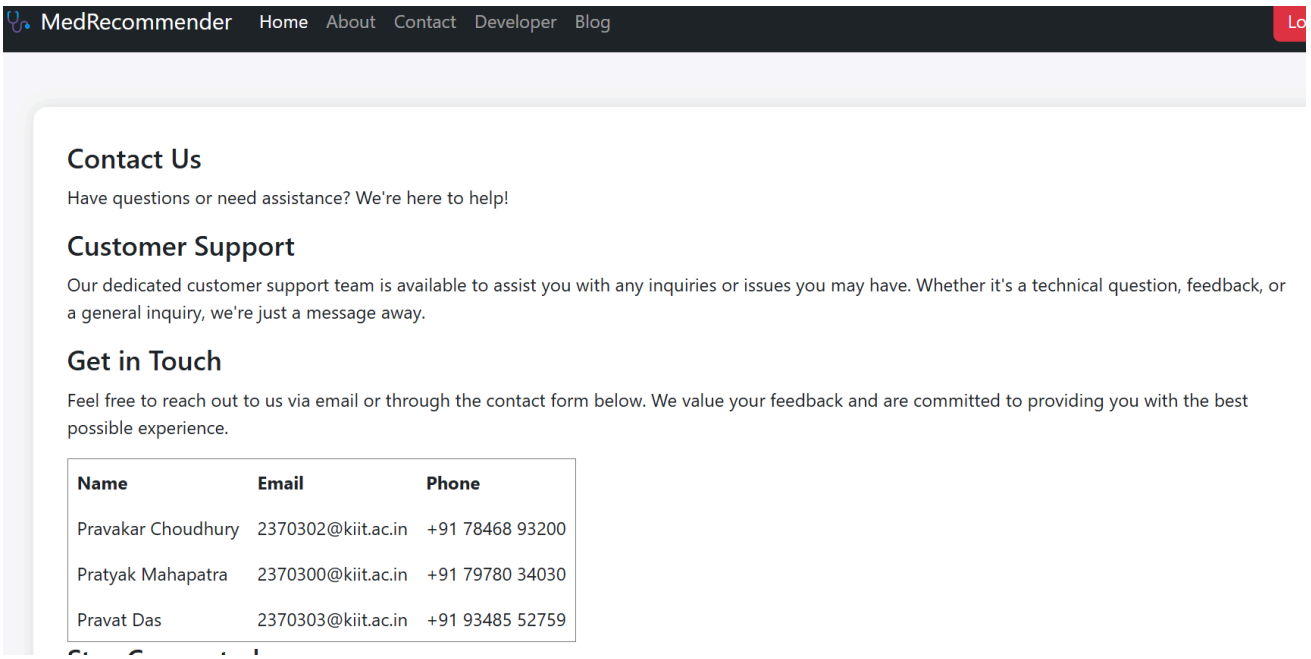


Figure 10

In the contact page we have our contact details, information and locations etc.

DEVELOPER:

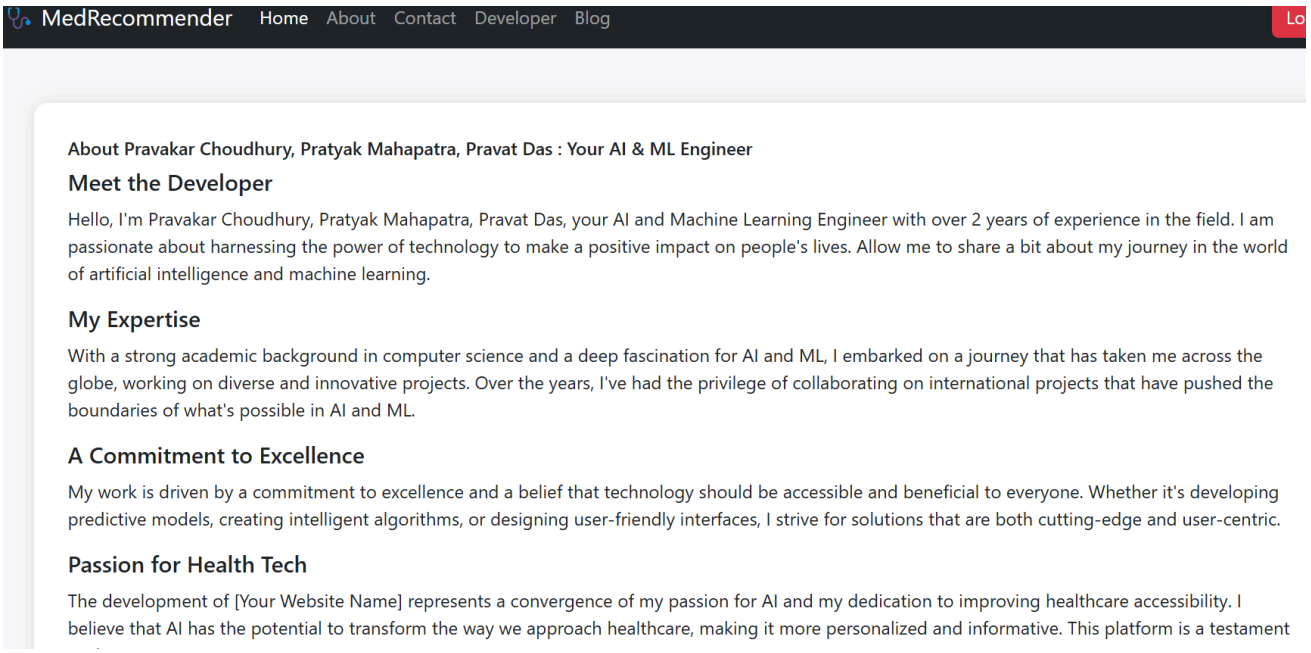


Figure 11

In the developer page there is information about the developers who, developed the projects, there journey and their commitment to the project.

BLOG:

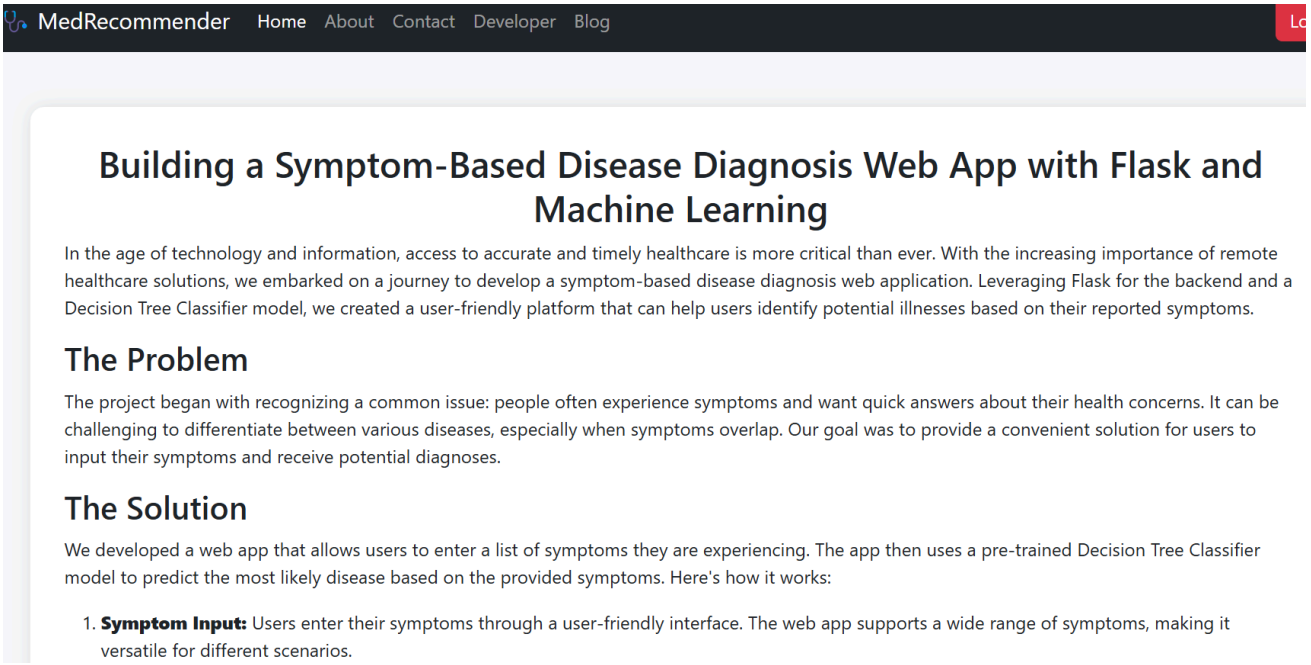


Figure-12

In the blog page we have information about the project and the solution to the problems occur in the medical system.

SEARCH:

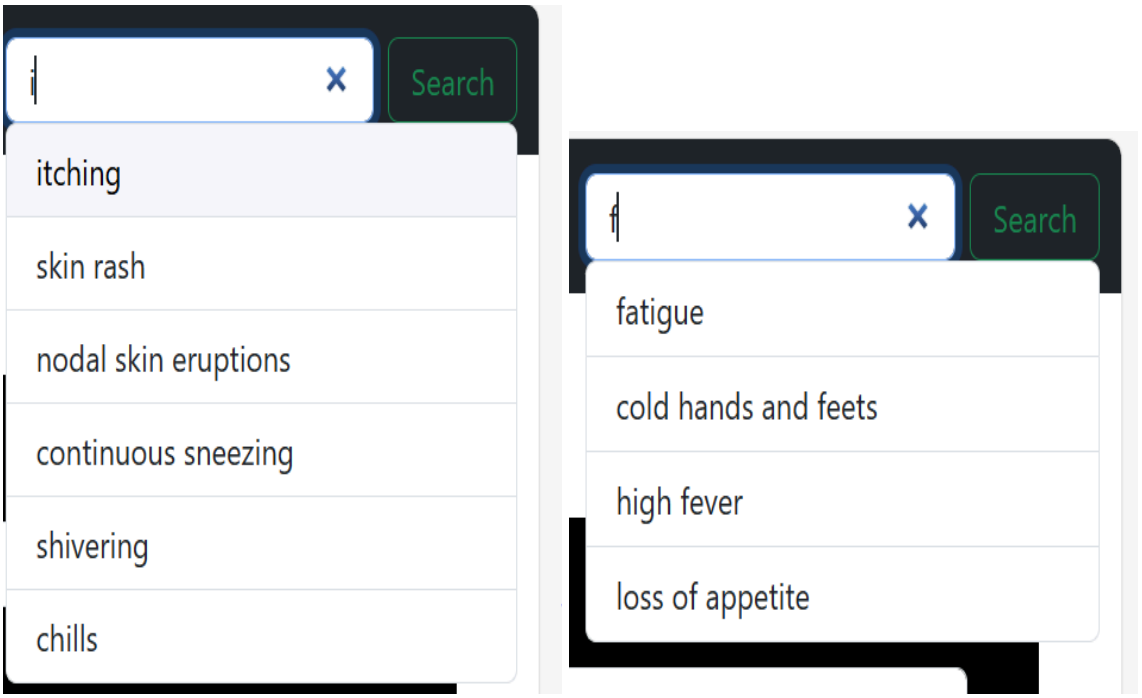


Figure-13

This icon help us in finding the names of the symptoms so that we paste it and predict our result.

SELECT SYMBOL:

Figure 14

In the select icon we can select the symptoms of the disease that we are facing.

PREDICT:

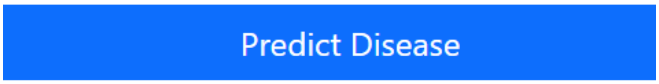


Figure-15

In the predict icon we can predict about the diseases ,the precautions we can take ,workouts etc.

DISEASES PREDICT:

Figure-16

Disease predict icon predicts what type of disease the person is suffering, for example the disease could be fungal, viral or any other type.

DISEASE DESCRIPTION:

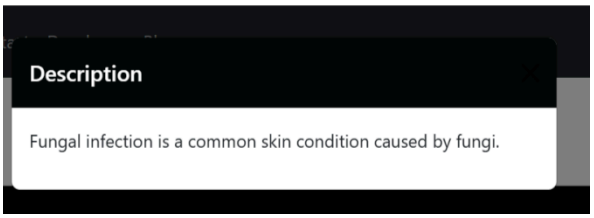


Figure-17

Disease description describes the type of disease and the cause of the disease.

PRECAUTION:

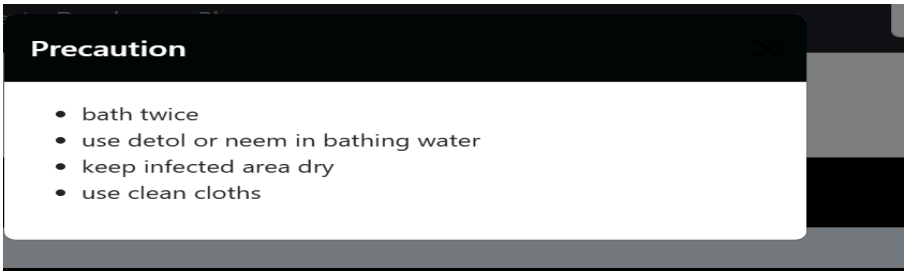


Figure 18

Precaution icon shows what the patient's daily routine should be so that is could be further get protected from the disease.

MEDICATION:

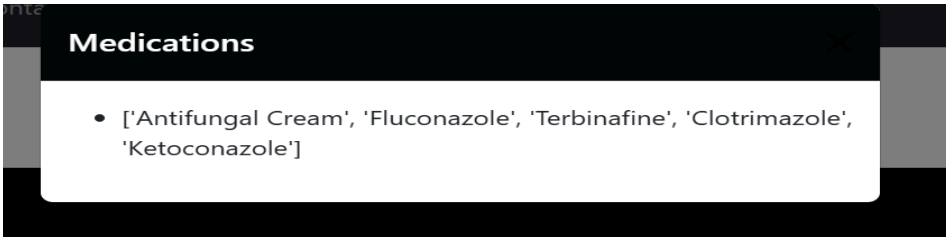


Figure-19

Medication icon shows what type of medical procedure we should acquire when we are suffering from this kind of diseases.

WORKOUTS:

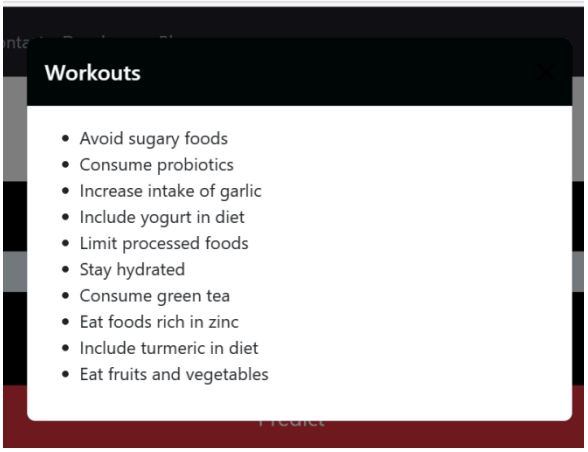


Figure -20

Workout icon shows that the what the patient should avoid or what the person should try when that person is suffering from a particular diseases.

DIETS:

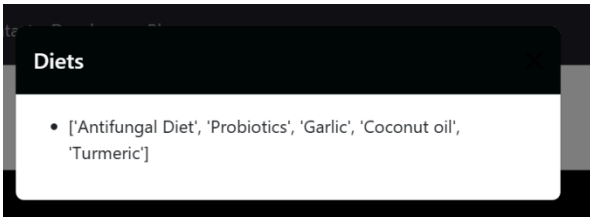


Figure-21

Diets icon shows what diet should a patient follow when he /she is suffering any disease.

MACHINE LEARNING

Data Loading and pre-processing

Data collection is the foundation of any machine learning project Data Collection is the process of gathering raw medical data from sources (e.g., CSV files, hospital databases, APIs).

This is the foundational step where we gather relevant data for your project. In healthcare, this could include patient records, symptoms, medications, diets, etc. Once collected, the data often needs to be cleaned and prepared for analysis.

Pre-processing is the process of cleaning the data (removing duplicates, handling missing values, converting categorical data, normalizing features).

Pre-processing includes handling missing values, correcting inconsistencies, converting categorical variables into numerical formats (encoding), normalizing or standardizing data, and ensuring all data is in a usable structure. This step ensures the dataset is accurate, complete, and ready for further processing.

Goal: Make the data usable for machine learning.

Dataset collection

We collect datat on which we can perfrom data analysis**.**

https://www.kaggle.com/code/syedfaizanalii/medicine-recommendation-system

https://www.kaggle.com/code/syedfaizanalii/medicine-recommendation-system/input?select=precautions_df.csv

https://www.kaggle.com/code/syedfaizanalii/medicine-recommendation-system/input?select=symtoms_df.csv

https://www.kaggle.com/code/syedfaizanalii/medicine-recommendation-system/input?select=diets.csv

https://www.kaggle.com/code/syedfaizanalii/medicine-recommendation-system/input?select=workout_df.csv

Install Libraries

```
import subprocess

# List of required libraries
libraries = ["numpy", "pandas", "matplotlib", "seaborn", "plotly", "scikit-learn", "basemap"]

#libraries = ["pandas", "scikit-learn"]

# Install each library
for lib in libraries:
    try:
        subprocess.check_call(["pip", "install", lib])
        print(f"Successfully installed {lib}")
    except subprocess.CalledProcessError:
        print(f"Failed to install {lib}")

print("All installations attempted.")
```

```
Successfully installed numpy
Successfully installed pandas
Successfully installed matplotlib
```

Check all Library Install or not

```
import importlib

# List of required libraries
libraries = ["numpy", "pandas", "matplotlib", "seaborn", "plotly", "scikit-learn", "basemap"]

# Check if each library is installed
for lib in libraries:
    try:
        importlib.import_module(lib)
        print(f"{lib} is installed ✅")
    except ImportError:
        print(f"{lib} is NOT installed ❌")

print("Library check completed.")
```

```
numpy is installed ✅
pandas is installed ✅
matplotlib is installed ✅
seaborn is installed ✅
```

Figure-22

Load dataset

```
import  pandas as pd
```

```
# dataset = pd.read_csv(r'C:\Users\rajes\PRAVAKAR PROJECT\DATASET\Training.csv')
dataset = pd.read_csv("DATASET/Training.csv")
dataset_copy = pd.read_csv("DATASET/Training.csv")
```

```
### preprocessing
```

```
dataset
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads
0	1	1	1	0	0	0	0	0	0	0	...	0
1	0	1	1	0	0	0	0	0	0	0	...	0
2	1	0	1	0	0	0	0	0	0	0	...	0
3	1	1	0	0	0	0	0	0	0	0	...	0

```
dataset_copy
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads
0	1	1	1	0	0	0	0	0	0	0	...	0
1	0	1	1	0	0	0	0	0	0	0	...	0
2	1	0	1	0	0	0	0	0	0	0	...	0
3	1	1	0	0	0	0	0	0	0	0	...	0
4	1	1	1	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
4915	0	0	0	0	0	0	0	0	0	0	...	0
4916	0	1	0	0	0	0	0	0	0	0	...	1

```
In [6]: dataset.head().T
```

Out[6]:

	0	1	2	3	4
itching	1	0	1	1	1
skin_rash	1	1	0	1	1
nodal_skin_eruptions	1	1	1	0	1
continuous_sneezing	0	0	0	0	0
shivering	0	0	0	0	0
...	...	...	...	...	...
inflammatory_nails	0	0	0	0	0
blister	0	0	0	0	0
red_sore_around_nose	0	0	0	0	0
yellow_crust_ooze	0	0	0	0	0
prognosis	Fungal infection	Fungal infection	Fungal infection	Fungal infection	Fungal infection

Figure-23

Exploratory Data Analysis (EDA)

EDA is all about understanding the data. EDA is the process of analyzing datasets to summarize their main characteristics using visual methods.

This involves:

Visualizing data using graphs like histograms, boxplots, scatter plots. Statistical analysis to understand distributions, correlations, and trends.

Identifying outliers, missing data, and possible data imbalances. EDA helps form hypotheses and guides feature selection.

```
# Check for missing values
dataset.isnull().sum()
```

```
itching                 0
skin_rash               0
nodal_skin_eruptions    0
continuous_sneezing     0
shivering               0
                       ..
inflammatory_nails      0
blister                 0
red_sore_around_nose    0
yellow_crust_ooze       0
prognosis               0
Length: 133, dtype: int64
```

Figure-24

Descriptive analysis

```
### Descriptive analysis
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
# Display basic info
dataset_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4920 entries, 0 to 4919
Columns: 133 entries, itching to prognosis
dtypes: int64(132), object(1)
memory usage: 5.0+ MB
```

```
# Check for missing values
missing_values = dataset_copy.isnull().sum()
print("Missing Values:\n", missing_values[missing_values > 0])
```

```
Missing Values:
 Series([], dtype: int64)
```

Figure-25

Descriptive analysis is the process of examining and summarizing the raw data to understand its basic characteristics. It helps you gain insights into the structure and quality of the data before applying any machine learning algorithms. In the medical domain, descriptive analysis helps identify trends, patterns, and anomalies related to patients, symptoms, diagnoses, treatments, or outcomes.

For example, in your project, descriptive analysis might include:

Identifying the most common symptoms reported by patients.

Understanding the distribution of patients across different age groups or gender.

Checking how often a particular disease or condition occurs.

Analyzing the average time taken for recovery across various treatments.

These steps provides the foundation for EDA and helps in formulating hypotheses or selecting features that are clinically meaningful.

Descriptive Statistics

```
# Descriptive statistics
print("Descriptive Statistics:\n", dataset_copy.describe())
```

```
Descriptive Statistics:
            itching    skin_rash  nodal_skin_eruptions  continuous_sneezing  \
count  4920.000000  4920.000000           4920.000000          4920.000000
mean      0.137805     0.159756              0.021951             0.045122
std       0.344730     0.366417              0.146539             0.207593
min       0.000000     0.000000              0.000000             0.000000
25%       0.000000     0.000000              0.000000             0.000000
50%       0.000000     0.000000              0.000000             0.000000
75%       0.000000     0.000000              0.000000             0.000000
max       1.000000     1.000000              1.000000             1.000000

         shivering       chills   joint_pain  stomach_pain      acidity  \
count  4920.000000  4920.000000  4920.000000   4920.000000  4920.000000
mean      0.021951     0.162195     0.139024      0.045122     0.045122
std       0.146539     0.368667     0.346007      0.207593     0.207593
min       0.000000     0.000000     0.000000      0.000000     0.000000
25%       0.000000     0.000000     0.000000      0.000000     0.000000
50%       0.000000     0.000000     0.000000      0.000000     0.000000
75%       0.000000     0.000000     0.000000      0.000000     0.000000
```

```
# Identify numerical and categorical columns
numeric_cols = dataset_copy.select_dtypes(include=['number']).columns
categorical_cols = dataset_copy.select_dtypes(include=['object']).columns
```

```
# Detect and remove duplicate rows
dataset_copy = dataset_copy.drop_duplicates()
```

```
# Feature scaling (example: standardization)
scaler = StandardScaler()
dataset_copy[numeric_cols] = scaler.fit_transform(dataset_copy[numeric_cols])
```

```
# Descriptive statistics
print("Descriptive Statistics:\n", dataset_copy.describe())
```

```
Descriptive Statistics:
             itching     skin_rash  nodal_skin_eruptions  continuous_sneezing  \
count  3.040000e+02  3.040000e+02          3.040000e+02         3.040000e+02
mean  -4.674623e-17 -3.505967e-17          2.337312e-17         1.168656e-17
std    1.001649e+00  1.001649e+00          1.001649e+00         1.001649e+00
min   -3.948336e-01 -4.003815e-01         -1.154701e-01        -2.027212e-01
25%   -3.948336e-01 -4.003815e-01         -1.154701e-01        -2.027212e-01
```

Figure-26

Descriptive Statistics

Descriptive statistics are the numerical or graphical methods used in descriptive analysis to summarize and describe the features of a dataset. They provide a quantitative summary of the data.

Descriptive statistics might include:

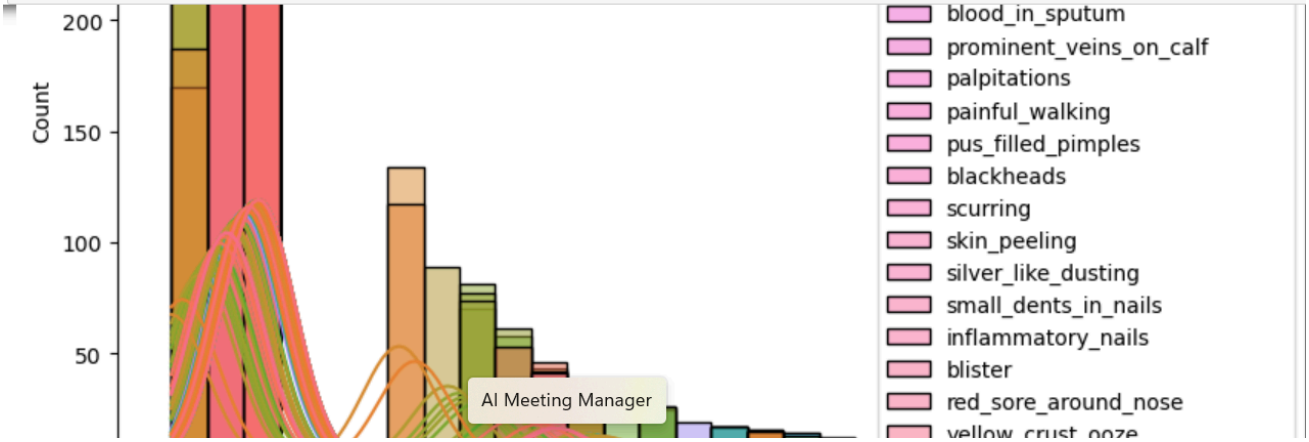
Measures of central tendency: Mean, median, and mode of patient ages, blood pressure levels, cholesterol, etc.

Measures of dispersion: Standard deviation, variance, range, and interquartile range to understand how spread out the health values are.

Frequency distributions: How often each symptom or diagnosis appears in the dataset.

Graphs and visualizations: Bar charts for symptom frequency, histograms for age distribution, pie charts for gender ratios, etc.

```
# Visualizing distributions
plt.figure(figsize=(10, 6))
sns.histplot(dataset_copy[numeric_cols].melt(), x="value", hue="variable", kde=True, bins=30)
plt.title("Distribution of Numerical Features")
plt.show()
```




```
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(dataset_copy.corr(), annot=True, cmap="coolwarm", fmt='.2f')
plt.title("Feature Correlation Matrix")
plt.show()
```

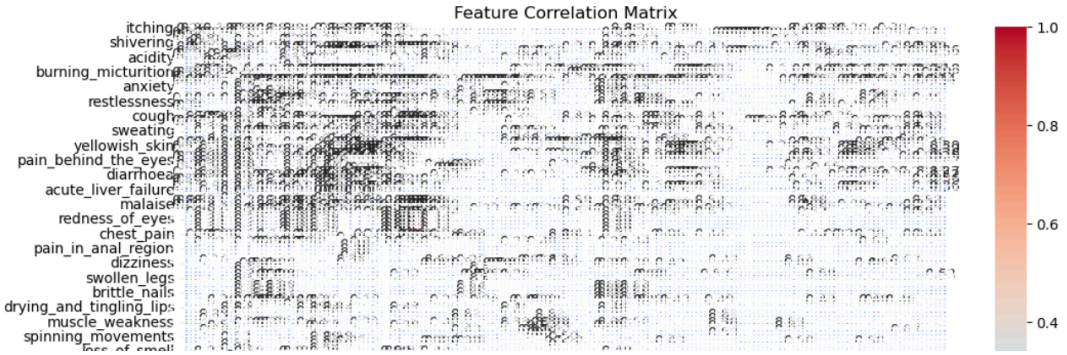


Figure-27

Train ML models

Model Training (ML Algorithms) is the process of applying machine learning algorithms (eg; Random forest, GradientBoosting, KNeighbors) to train a model.

It is the process that split data into training and testing sets, fit the model using training data.

Goal: Build a model that learns patterns in the data.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
X = dataset.drop('prognosis', axis=1)
y = dataset['prognosis']

# ecoding prognonsis
le = LabelEncoder()
le.fit(y)
Y = le.transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=20)
```

```
X
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	pus_filled_p
0	1	1	1	0	0	0	0	0	0	0	...	
1	0	1	1	0	0	0	0	0	0	0	...	
2	1	0	1	0	0	0	0	0	0	0	...	

```
Y
```

```
array([15, 15, 15, ..., 38, 35, 27])
```

```
y
```

```
0                              Fungal infection
1                              Fungal infection
2                              Fungal infection
3                              Fungal infection
4                              Fungal infection
                          ...
4915    (vertigo) Paroymsal  Positional Vertigo
4916                                      Acne
4917                   Urinary tract infection
4918                                  Psoriasis
4919                                   Impetigo
Name: prognosis, Length: 4920, dtype: object
```

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (3444, 132)
X_test shape: (1476, 132)
y_train shape: (3444,)
y_test shape: (1476,)
```

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
import os


# Fix CPU core issue (Optional)
os.environ["LOKY_MAX_CPU_COUNT"] = "4"  # Set to a valid number of logical cores
```

```python
# Convert to NumPy arrays & Handle NaNs
X_train = np.nan_to_num(np.array(X_train, dtype=np.float64))
X_test = np.nan_to_num(np.array(X_test, dtype=np.float64))

# Dictionary of Models
models = {
    'SVC': SVC(kernel='linear'),
    'RandomForest': RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=1),  # Limit cores
    'GradientBoosting': GradientBoostingClassifier(n_estimators=100, random_state=42),
    'KNeighbors': KNeighborsClassifier(n_neighbors=5),
    'GaussianNB': GaussianNB()
}

# Train & Evaluate Models
for model_name, model in models.items():
    # TRAIN MODEL
    model.fit(X_train, y_train)
    # TEST MODEL
    predictions = model.predict(X_test)

    accuracy = accuracy_score(y_test, predictions)
    print(f"{model_name} Accuracy: {accuracy:.4f}")
```

```python
    cm = confusion_matrix(y_test, predictions)
    print(f"{model_name} Confusion Matrix:\n{cm}")

    print("\n" + "="*40 + "\n")
```

```
SVC Accuracy: 1.0000
SVC Confusion Matrix:
[[40  0  0 ...  0  0  0]
 [ 0 43  0 ...  0  0  0]
 [ 0  0 28 ...  0  0  0]
 ...
 [ 0  0  0 ... 34  0  0]
 [ 0  0  0 ...  0 41  0]
 [ 0  0  0 ...  0  0 31]]

========================================

RandomForest Accuracy: 1.0000
RandomForest Confusion Matrix:
[[40  0  0 ...  0  0  0]
 [ 0 43  0 ...  0  0  0]
 [ 0  0 28 ...  0  0  0]
```

```
X
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	pus_filled_p
0	1	1	1	0	0	0	0	0	0	0	...	
1	0	1	1	0	0	0	0	0	0	0	...	
2	1	0	1	0	0	0	0	0	0	0	...	
3	1	1	0	0	0	0	0	0	0	0	...	
4	1	1	1	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4915	0	0	0	0	0	0	0	0	0	0	...	
4916	0	1	0	0	0	0	0	0	0	0	...	
4917	0	0	0	0	0	0	0	0	0	0	...	
4918	0	1	0	0	0	0	1	0	0	0	...	
4919	0	1	0	0	0	0	0	0	0	0	...	

```
y
```

```
0                              Fungal infection
1                              Fungal infection
2                              Fungal infection
3                              Fungal infection
4                              Fungal infection
                              ...
4915    (vertigo) Paroymsal  Positional Vertigo
4916                                      Acne
4917                   Urinary tract infection
4918                                 Psoriasis
4919                                  Impetigo
Name: prognosis, Length: 4920, dtype: object
```

```
Y
```

```
array([15, 15, 15, ..., 38, 35, 27])
```

Figure-28

Prediction and Recommendation

Making Recommendations is the process that uses the trained model to give medical advice/recommendations (e.g., suggest potential diseases or treatments).

Based on the trained model's predictions we can suggest diagnoses or treatments, recommend lifestyle or medication changes, provide alerts or risk levels to healthcare providers or patients.

Goal: Deliver meaningful, data-driven suggestions to the end-user or doctor.

```
# selecting svc
svc = SVC(kernel='linear')
svc.fit(X_train,y_train)
ypred = svc.predict(X_test)
accuracy_score(y_test,ypred)
```

```
1.0
```

```
# save svc
import pickle
pickle.dump(svc,open("MODELS/svc.pkl",'wb'))
```

```
# load model
svc = pickle.load(open("MODELS/svc.pkl",'rb'))
```

```
X_test
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [1., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
# Test 1: Single Prediction
print("Predicted disease:", svc.predict(X_test[0].reshape(1, -1)))
print("Actual Disease:", y_test[0])
```

```
Predicted disease: [40]
Actual Disease: 40
```

```
# Test 2:
print("Predicted disease:", svc.predict(X_test[100].reshape(1, -1)))
print("Actual Disease:", y_test[100])
```

```
Predicted disease: [39]
Actual Disease: 39
```

```
# Test with different samples
print("Predicted disease:", svc.predict(X_test[10].reshape(1, -1)))
print("Actual Disease:", y_test[10])

print("Predicted disease:", svc.predict(X_test[25].reshape(1, -1)))
print("Actual Disease:", y_test[25])
```

```
Predicted disease: [20]
Actual Disease: 20
Predicted disease: [17]
Actual Disease: 17
```

```
sym_des = pd.read_csv("DATASET/symtoms_df.csv")
precautions = pd.read_csv("DATASET/precautions_df.csv")
workout = pd.read_csv("DATASET/workout_df.csv")
description = pd.read_csv("DATASET/description.csv")
medications = pd.read_csv('DATASET/medications.csv')
diets = pd.read_csv("DATASET/diets.csv")
```

```
#=========================================================
# custome and helping functions
#==========================helper funtions================
def helper(dis):
    desc = description[description['Disease'] == predicted_disease]['Description']
    desc = " ".join([w for w in desc])

    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2', 'Precaution_3', 'Precaution_4']]
    pre = [col for col in pre.values]

    med = medications[medications['Disease'] == dis]['Medication']
    med = [med for med in med.values]

    die = diets[diets['Disease'] == dis]['Diet']
    die = [die for die in die.values]

    wrkout = workout[workout['disease'] == dis] ['workout']


    return desc,pre,med,die,wrkout

symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills':
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic
```

```
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills':
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic

# Model Prediction function
def get_predicted_value(patient_symptoms):
    input_vector = np.zeros(len(symptoms_dict))
    for item in patient_symptoms:
        input_vector[symptoms_dict[item]] = 1
    return diseases_list[svc.predict([input_vector])[0]]
```

```
# Test 1
# Split the user's input into a list of symptoms (assuming they are comma-separated) # itching,skin_rash,nodal_skin_eruptions
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)

desc, pre, med, die, wrkout = helper(predicted_disease)

print("==================predicted disease============")
```

```
desc, pre, med, die, wrkout = helper(predicted_disease)

print("==================predicted disease============")
print(predicted_disease)
print("==================description==================")
print(desc)
print("==================precautions==================")
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1

print("==================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1

print("==================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1
```

```
print("==================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1
```

```
Enter your symptoms.......itching,skin_rash,nodal_skin_eruptions
==================predicted disease============
Fungal infection
==================description==================
Fungal infection is a common skin condition caused by fungi.
==================precautions==================
1 :  bath twice
2 :  use detol or neem in bathing water
3 :  keep infected area dry
4 :  use clean cloths
==================medications==================
5 :  ['Antifungal Cream', 'Fluconazole', 'Terbinafine', 'Clotrimazole', 'Ketoconazole']
==================workout==================
6 :  Avoid sugary foods
7 :  Consume probiotics
```

```
# Test 1
# Split the user's input into a list of symptoms (assuming they are comma-separated) # yellow_crust_ooze,red_sore_around_nose
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)

desc, pre, med, die, wrkout = helper(predicted_disease)

print("=================predicted disease============")
print(predicted_disease)
print("=================description==================")
print(desc)
print("=================precautions==================")
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1

print("=================medications==================")
for m_i in med:
```

```
print("=================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1

print("=================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1

print("=================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1
```

```
Enter your symptoms.......yellow_crust_ooze,red_sore_around_nose,small_dents_in_nails,inflammatory_nails,blister
=================predicted disease============
Impetigo
=================description==================
Impetigo is a highly contagious skin infection causing red sores that can break open.
```

```
8 :  Consume nutrient-rich foods
9 :  Limit sugary foods and beverages
10 :  Include foods rich in vitamin C
11 :  Consult a healthcare professional
12 :  Follow medical recommendations
13 :  Avoid scratching
14 :  Take prescribed antibiotics
15 :  Practice wound care
=================diets==================
16 :  ['Impetigo Diet', 'Antibiotic treatment', 'Fruits and vegetables', 'Hydration', 'Protein-rich foods']
```

```
# let's use pycharm flask app
# but install this version in pycharm
import sklearn
print(sklearn.__version__)
```

```
1.3.0
```

Figure-29

TESTING

Unit Testing:

In the unit test case if we gave any value , we get similar result in every case.In unit test case single value is given.

```
# Test 1: Single Prediction
print("Predicted disease:", svc.predict(X_test[0].reshape(1, -1)))
print("Actual Disease:", y_test[0])
```

```
Predicted disease: [40]
Actual Disease: 40
```

```
# Test 2:
print("Predicted disease:", svc.predict(X_test[100].reshape(1, -1)))
print("Actual Disease:", y_test[100])
```

```
Predicted disease: [39]
Actual Disease: 39
```

```
# Test with different samples
print("Predicted disease:", svc.predict(X_test[10].reshape(1, -1)))
print("Actual Disease:", y_test[10])

print("Predicted disease:", svc.predict(X_test[25].reshape(1, -1)))
print("Actual Disease:", y_test[25])
```

Figure-30

Multiple Unit Test

Test 1:

In multiple testing if we gave multiple value like itching, skin_rash, nodal-skin_erupton etc then we also get multiple result like bath twice, use clean cloths etc.

```
# Test 1
# Split the user's input into a list of symptoms (assuming they are comma-separated) # itching,skin_rash,nodal_skin_eruptions
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)

desc, pre, med, die, wrkout = helper(predicted_disease)

print("=================predicted disease============")
print(predicted_disease)
print("=================description==================")
print(desc)
print("=================precautions==================")
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1

print("=================medications==================")
for m i in med:
```

```
print("==================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1

print("==================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1

print("==================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1
```

```
Enter your symptoms.......itching,skin_rash,nodal_skin_eruptions
==================predicted disease============
Fungal infection
==================description==================
Fungal infection is a common skin condition caused by fungi.
------------------precautions------------------
```

```
1 :  bath twice
2 :  use detol or neem in bathing water
3 :  keep infected area dry
4 :  use clean cloths
==================medications==================
5 :  ['Antifungal Cream', 'Fluconazole', 'Terbinafine', 'Clotrimazole', 'Ketoconazole']
==================workout==================
6 :  Avoid sugary foods
7 :  Consume probiotics
8 :  Increase intake of garlic
9 :  Include yogurt in diet
10 :  Limit processed foods
11 :  Stay hydrated
12 :  Consume green tea
13 :  Eat foods rich in zinc
14 :  Include turmeric in diet
```

Figure-31

Test 2:

In multiple testing if we gave multiple value like inflammatory nails, small dents, blisters etc then we also get multiple result like use antibiotics, stay hydrated etc.

```
# Test 2
# Split the user's input into a list of symptoms (assuming they are comma-separated) # yellow_crust_ooze,red_sore_around_nose
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)

desc, pre, med, die, wrkout = helper(predicted_disease)

print("=================predicted disease============")
print(predicted_disease)
print("=================description==================")
print(desc)
print("=================precautions==================")
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1

print("=================medications==================")
for m_i in med:
```

```
print("=================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1

print("=================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1

print("=================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1
```

```
Enter your symptoms.......yellow_crust_ooze,red_sore_around_nose,small_dents_in_nails,inflammatory_nails,blister
=================predicted disease============
Impetigo
=================description==================
Impetigo is a highly contagious skin infection causing red sores that can break open.
==================precautions==================
1 :  soak affected area in warm water
2 :  use antibiotics
3 :  remove scabs with wet compressed cloth
4 :  consult doctor
==================medications==================
5 :  ['Topical antibiotics', 'Oral antibiotics', 'Antiseptics', 'Ointments', 'Warm compresses']
==================workout==================
6 :  Maintain good hygiene
7 :  Stay hydrated
8 :  Consume nutrient-rich foods
9 :  Limit sugary foods and beverages
10 :  Include foods rich in vitamin C
11 :  Consult a healthcare professional
12 :  Follow medical recommendations
13 :  Avoid scratching
14 :  Take prescribed antibiotics
15 :  Practice wound care
```

Figure-32

Future Scope of the Project

1. **Integration with Real-Time Health Monitoring Devices**
    - Connect wearable devices (smartwatches, fitness bands) to fetch live health parameters (heart rate, BP, oxygen levels) for more accurate diagnosis.
2. **Addition of More Advanced Machine Learning Algorithms**
    - Incorporate deep learning models (e.g., RNNs, CNNs) or ensemble methods for higher prediction accuracy.
3. **User Authentication and Personal Health Dashboard**
    - Implement secure login and personalized dashboards to track symptom history, diagnosis, and recommendations over time.
4. **Chatbot for Initial Symptom Collection**
    - Add an intelligent chatbot to interactively collect symptoms and provide instant preliminary advice.
5. **Multi-language Support**
    - Localize the app to support various regional languages for wider accessibility in diverse communities.
6. **Integration with Pharmacy APIs**
    - Suggest nearby pharmacies and allow users to order recommended medicines directly through the app.
7. **Doctor Consultation Feature**
    - Provide an option to book telemedicine consultations based on diagnosis for verified treatment.
8. **Automated Diet and Workout Plans**
    - Generate dynamic diet/workout plans based on diagnosed disease, user preferences, and lifestyle data.
9. **Mobile App Version**
    - Create a cross-platform mobile app (using Flutter or React Native) for wider usability and on-the-go access.

10. **Real-Time Disease Outbreak Analytics**

- Track and analyze the frequency of certain diseases in regions to identify and alert users of local outbreaks.

11. **Feedback & Learning System**

- Let users give feedback on prediction accuracy and outcomes, improving the model using continuous learning.

Assumptions

1. **Symptoms Are Provided Accurately by the User**
    - It is assumed that users provide correct and relevant symptoms without misreporting or omissions.
2. **Single Disease Diagnosis per Prediction**
    - The system assumes that only one disease is associated with the given symptoms at a time, not multiple conditions.
3. **Preprocessed and Clean Datasets**
    - The datasets used (symptoms_df.csv, Symptom-severity.csv, etc.) are assumed to be cleaned, validated, and free of major inconsistencies or missing values.
4. **Machine Learning Model is Trained Properly**
    - It is assumed that the svc.pkl model has been trained on sufficient and balanced data to make reliable predictions.
5. **Static Dataset**
    - The dataset does not change dynamically or update in real time; any updates must be done manually.
6. **User Understands the Application's Output**
    - It is assumed that users are capable of interpreting the result (disease name, precautions, medicine suggestions, etc.) as general guidance, not a medical prescription.
7. **No Emergency Medical Conditions Handled**
    - The app is not designed to handle urgent or life-threatening conditions and assumes users will seek professional help in such cases.
8. **Server and Backend are Always Available**
    - It is assumed that the Flask backend and application server are always available and running without crashes.
9. **Environment Compatibility**

- The application is assumed to run on a standard Python environment (Python 3.13 in your case) with all necessary dependencies installed.

10. **Limited Scope of Diseases Covered**

- The model is trained to predict only a limited set of diseases listed in the dataset, not all possible medical conditions.

Glossary

Term	Definition
Symptom	A physical or mental feature that indicates a condition or disease.
Disease	A disorder affecting the body or mind, diagnosed based on symptoms.
Diagnosis	The process of identifying a disease from its symptoms.
Machine Learning (ML)	A branch of AI that enables systems to learn and make predictions from data.
Model (svc.pkl)	A trained Support Vector Classifier used to predict diseases based on symptoms.
CSV File	A data file format (Comma-Separated Values) used to store tabular data.
Dataset	A structured collection of data used for training or prediction.
Flask	A lightweight Python web framework used to build the backend of the application.
Frontend	The user interface of the web app, where users input symptoms and view results.
Backend	The server-side logic that handles input, runs the ML model, and sends output.
Precaution	A preventive measure suggested to reduce the risk of worsening a condition.
Medication	The suggested treatment or drug related to the diagnosed disease.
Workout Plan	A suggested set of physical exercises based on the diagnosed condition.
Diet Plan	A recommended set of foods suitable for managing the predicted disease.

HTML Templates	Predefined frontend page layouts rendered by Flask (index.html, etc.).
Pickle (.pkl)	A file format used to save trained machine learning models in Python.
User Input	The list of symptoms provided by the user for diagnosis.
Output Prediction	The disease predicted by the ML model along with related recommendations.

References

1. Brownlee, J. (2016). Machine Learning Mastery With Python: Understand Your Data, Create Accurate Models, and Work Projects End-To-End. Machine Learning Mastery.

2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

4. Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine Learning in Medicine. New England Journal of Medicine, 380(14), 1347–1358. https://doi.org/10.1056/NEJMra1814259

5. Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

6. Sharma, A., & Kaushik, S. (2020). Medical Recommendation System using Machine Learning Algorithms. International Journal of Scientific Research in Computer Science, 8(2), 44–49.