Fourth Year SC Project

8th Sem , MSc(CA & IT)

Medical Diagnosis System with Fuzzy Logic

Submitted By :

1) Darji Khushi Rakeshkumar (4198)
2) Rami Hinal Kirankumar (4241)
3) Keya Ashish Shah (4249)

Date of Submission :

5th May 2025

Name of Instructor :

Naisargi Oza Ma'am

Submitted to :

K. S. School Of Business Management and Information Technology

M.Sc.(CA & IT)

ACKNOWLEDGEMENT

It was a great experience working on the project titled **"Medical Diagnosis System with Fuzzy Logic."** We express our heartfelt gratitude to all those who were consistently involved with us throughout the development of this system. Their guidance and support will always hold a special place in our journey, both academically and professionally.

No technical project can be executed proficiently without the sharing of meticulous ideas, domain knowledge, and innovative thoughts contributed by individuals with both technical and non-technical expertise. The successful development of this system required a conducive learning environment and expert guidance, which helped us remain enthusiastic and focused throughout the project lifecycle.

We are especially grateful to all our academic supervisors for being approachable, supportive, and encouraging during every phase of the project. Their continuous feedback and direction made the implementation process smoother and more insightful.

We would also like to express our sincere gratitude to **Mrs. Naisargi Oza**, our project mentor, who provided us with this wonderful opportunity and continuously guided us in the correct direction. Her motivation, expert advice, and unwavering support were invaluable in the system analysis and design stages of this project.

Thank you for your constant support and proper guidance.

Thanking You,

4198 Darji Khushi Rakeshkumar

4241 Hinal Kirankumar

4249 Keya Ashish Shah

INDEX

❖ Abstract

- The Medical Diagnosis System is a Fuzzy Logic-based AI system designed for diagnosing diseases such as Diabetes Disease, Heart Disease, PCOD Disorder, Thyroid Disorder, and Anxiety Disorder.
- It processes uncertain and imprecise medical data, mimicking human reasoning to make diagnostic decisions.
- The system analyses symptoms, medical history, and test reports to provide risk assessment and early detection, improving healthcare outcomes with rule-based decision-making.
- Additionally, the system includes a file upload feature, allowing patients to upload medical files (such as reports), based on which the system predicts the likelihood of the presence of specific diseases.

❖ Introduction

➢ Background of the problem

Accurate medical diagnosis often faces challenges due to uncertainty, incomplete information, and variability in patient symptoms. Traditional diagnostic methods may struggle to handle such complexities effectively. Fuzzy logic provides a flexible approach to model this uncertainty, offering a more human-like reasoning process for supporting medical decisions.

➢ Purpose of the project

The purpose of this project is to design and develop a Medical Diagnosis System using Fuzzy Logic that assists in the identification and risk assessment of diabetes, thyroid disorders, anxiety, PCOD, and heart diseases. The system aims to enhance diagnostic support by processing clinical data through fuzzy inference. It allows for both manual symptom input and automated analysis through uploaded patient files, enabling broader usability and convenience.

➢ Scope and limitations

The system evaluates selected health parameters to predict the risk levels for specific diseases using fuzzy rule-based logic.

Limitations include:

- The system acts as a supportive tool and does not replace professional medical consultation.
- Results depend on the quality and accuracy of the input data.
- Fuzzy models are based on predefined rules and may require updates as medical knowledge evolves

❖ Literature Review

1. Existing Research Work

- **Diabetes Prediction**
  The PIMA Indian Diabetes Dataset is widely used. Smith et al. (2019) applied Decision Trees, Random Forest, and SVM, achieving over 80% accuracy. Deep learning models like neural networks improved precision with larger datasets.
- **Thyroid Disorder Detection**
  Zhang et al. (2020) applied ensemble methods like XGBoost and AdaBoost on UCI thyroid datasets, significantly improving detection of hyperthyroidism and hypothyroidism.
- **Heart Disease Prediction**
  The Cleveland Heart Disease dataset is a standard benchmark. Logistic Regression, Gradient Boosting, and Neural Networks (Patel and Prajapati, 2018) have achieved up to 90% accuracy in heart disease prediction.
- **PCOD Detection**
  Due to limited datasets, recent studies use hormonal and lifestyle data. Kumar et al. (2021) used Random Forests on health parameters, detecting PCOD with about 87% accuracy.
- **Anxiety Detection**
  AI in mental health diagnostics is growing. NLP techniques classify anxiety from patient responses. Lee et al. (2021) used BERT models on forum data for high-accuracy anxiety detection.

2. Similar Projects

- **IBM Watson Health**
  IBM's Watson Health integrates large datasets for disease diagnosis and treatment recommendations, illustrating AI's healthcare potential, although not disease-specific.
- **UCI ML Repository Projects**
  Open-source projects using UCI datasets (diabetes, thyroid, heart disease) apply algorithms like k-NN, Naive Bayes, and Deep Neural Networks to compare model performances.
- **Disease Prediction Android Applications**
  Apps like Ada Health and Babylon use user-reported symptoms and demographics to predict diseases, but they are generalized and do not specifically predict PCOD or anxiety.
- **Ensemble Disease Prediction Models**
  Recent projects combine multiple datasets for multi-disease prediction. Rao and Singh (2022) developed a CNN-LSTM hybrid model to classify diabetes, heart disease, and thyroid disorders.

❖ System Requirements

1. Software Requirements

Component	Details
Operating System	Windows 10/11
Programming Language	Python 3.8 or higher

2. Hardware Requirements

Component	Minimum	Recommended
Processor	Dual-core 2.0 GHz	Quad-core Intel i5 / AMD Ryzen 5 or better
RAM	4 GB	8 GB or more
Storage	1 GB free space	5 GB+ SSD storage
Internet	Required for downloading libraries	Required for online dataset & Streamlit UI
Graphics	Not required	Integrated GPU sufficient

❖ Methodology

➢ Steps followed in the project

1. **Problem Identification**
   Identified the need for a medical diagnosis system that can assess multiple diseases using fuzzy logic and provide early diagnosis support.

2. **Dataset Collection**
   Collected publicly available datasets for the following diseases:
   - Diabetes
   - Heart Disease
   - Thyroid
   - PCOD
   - Anxiety

3. **Data Preprocessing**
    - Removed missing or irrelevant values
    - Renamed columns for clarity
    - Normalized or standardized data as needed
    - Selected the most relevant features (input parameters)

4. **Fuzzy Logic System Design**
    - Defined fuzzy input variables (e.g., Glucose, Blood Pressure, etc.)
    - Defined fuzzy output variable (e.g., Diagnosis or Risk Level)
    - Created membership functions (Poor, Average, Good or Low, Medium, High)
    - Formulated fuzzy rules (IF-THEN logic)
    - Simulated and tested the fuzzy control systems

5. **Frontend Development using Streamlit**
    - Designed an interactive user interface for each disease module
    - Created sidebar navigation using streamlit-option-menu
    - Collected input values from the user dynamically
    - Displayed diagnosis results with simple outputs (e.g., Yes/No, Risk Level)

6. **Backend Integration**
    - Linked user input to fuzzy control system
    - Processed the input using scikit-fuzzy library
    - Displayed real-time diagnosis results

7. **Dataset Display Feature**
    - Added a “Dataset” tab to allow users to view raw data
    - Used st.dataframe() to display tabular data for all five diseases

8. **Testing and Validation**
    - Ran multiple test cases using known data to verify outputs
    - Ensured accuracy and reliability of fuzzy results

9. **Final Integration**
    - Combined all modules (Diabetes, Heart, Thyroid, PCOD, Anxiety) into a single Streamlit app
    - Ensured a consistent and user-friendly design

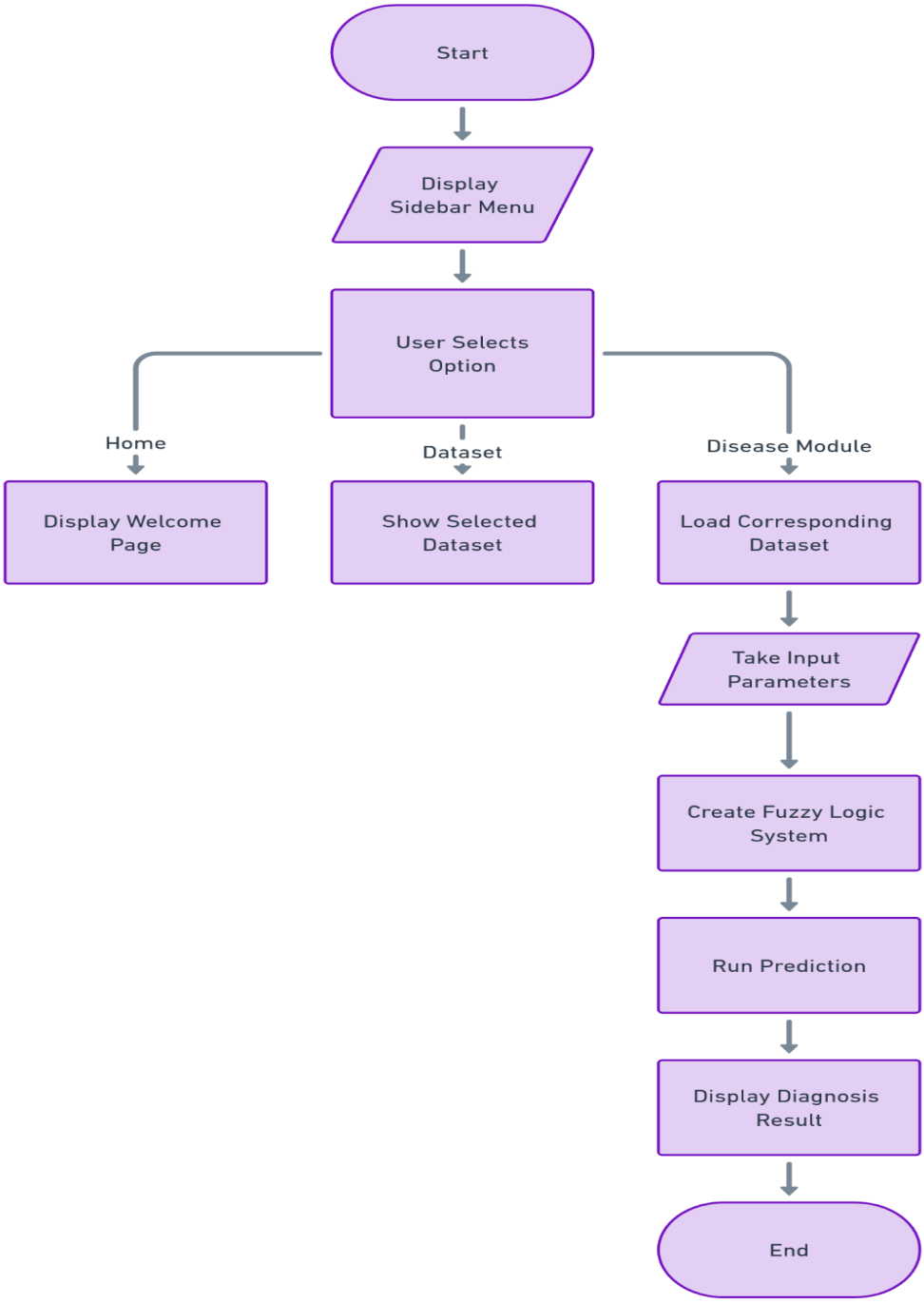
10. **Deployment and Execution**
    - Final version launched locally using the command:
    - streamlit run app.py

➢ Algorithms, flowcharts, or diagrams

- **Algorithms**

1. Fuzzification – Converts numeric inputs into fuzzy variables.
2. Fuzzy Rule-Based System – Uses expert-defined IF-THEN rules to determine medical risk.
3. Inference Engine – Applies fuzzy logic rules to evaluate risk levels.
4. Defuzzification – Converts fuzzy outputs into a Health risk.

- **Diagram :** System flow Diagram



➢ Tools and technologies used

Tools and Technology	
IDE / Code Editor	Visual Studio Code / Jupyter Notebook / Anaconda Navigator
Web Framework	Streamlit
Required Python Packages	numpy, pandas, scikit-fuzzy, streamlit, streamlit-option-menu
Browser	Google Chrome / Mozilla Firefox / Microsoft Edge
Optional Tools	Excel / Google Sheets (for viewing CSV files)

❖ Implementation & Development

➢ Description of coding, database, or system development

The medical diagnosis system was developed using **Python**, leveraging its powerful libraries and fuzzy logic capabilities to predict five key health conditions: **Heart Disease**, **Diabetes**, **Thyroid Disorders**, **PCOD**, and **Anxiety**.

- **Coding and Algorithm**
  We used **fuzzy logic algorithms** to manage the uncertainty in medical data and make reliable predictions based on user inputs such as age, symptoms, and lifestyle. Key libraries include:
  1. **NumPy**, **Pandas** – Data handling
  2. **scikit-fuzzy** – Fuzzy logic implementation
  3. **Flask** – For web deployment (optional)

  Each condition uses a dedicated fuzzy model with rules based on clinical research.

- **Database**
  Data was sourced from public online databases such as:
  1. **UCI Repository** – Heart disease, diabetes
  2. **Kaggle** – Thyroid, PCOD, and anxiety

  Data was pre-processed and used to train/test the system without including any personal information.

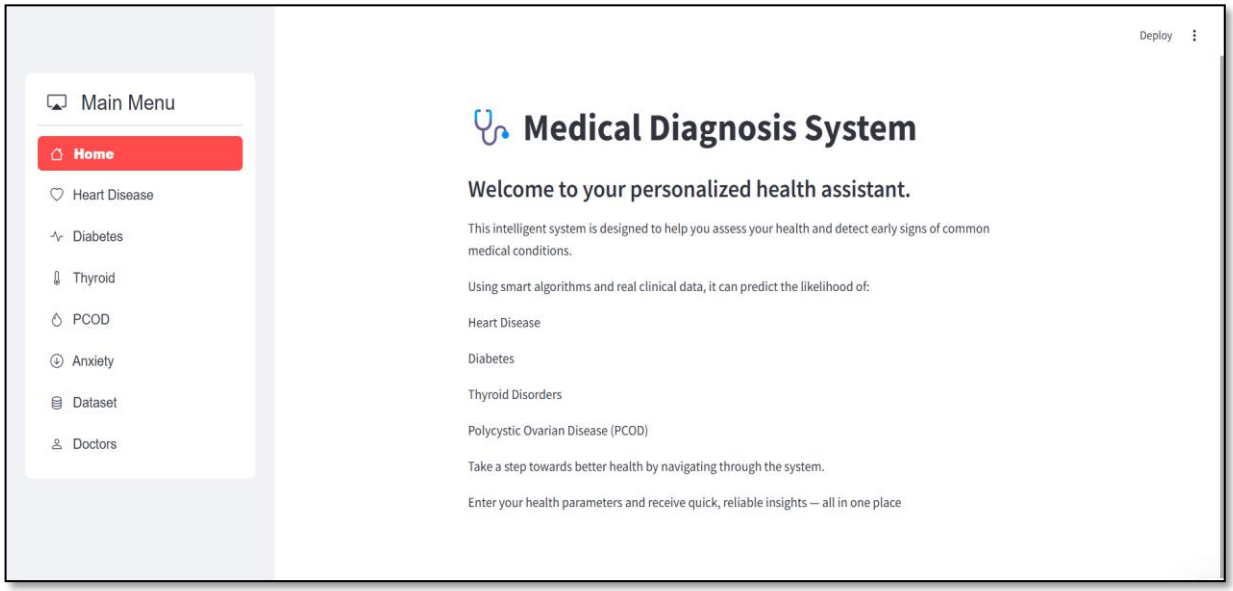
- **System Overview**

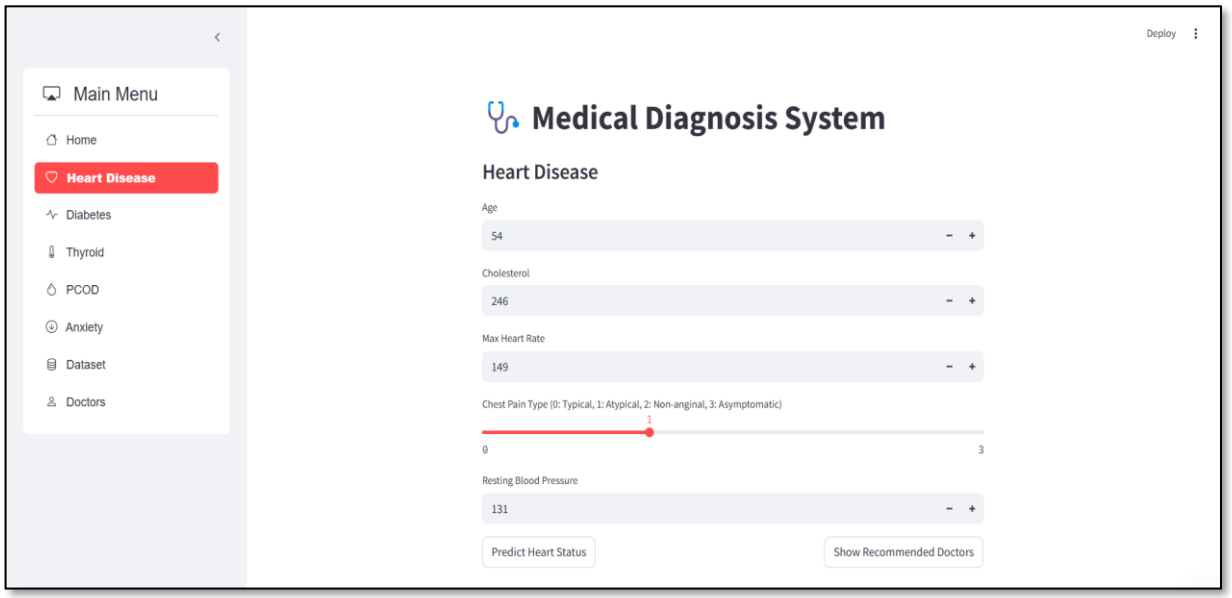
  Components include:

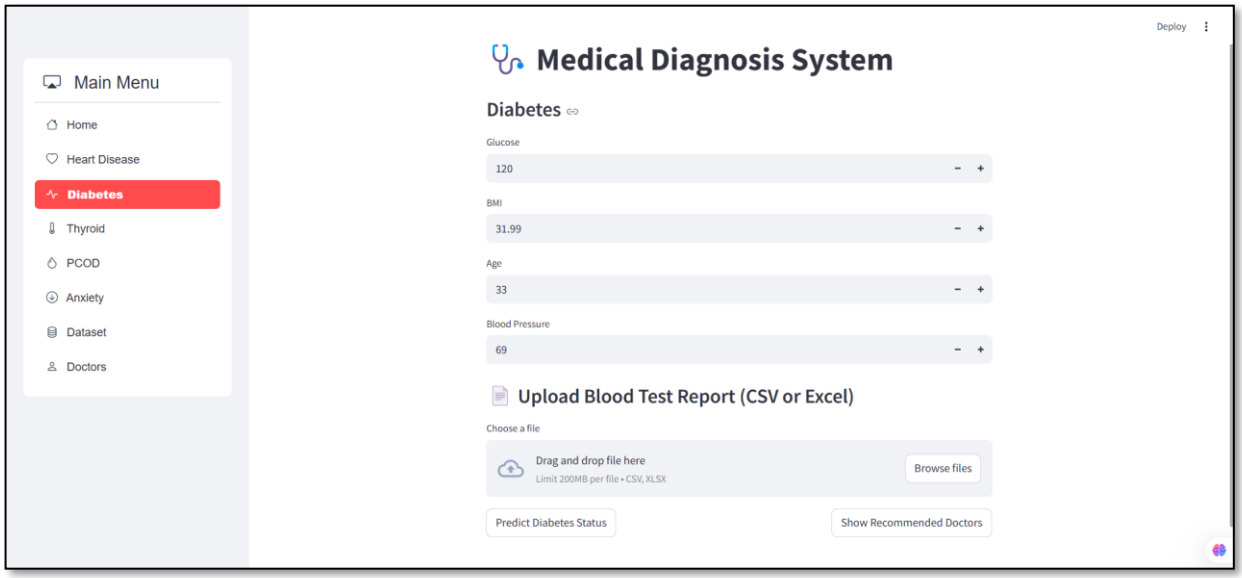
  1. **User Input Interface**
  2. **Fuzzy Inference Engine**
  3. **Prediction Output Display**
  4. **Database Access Layer**

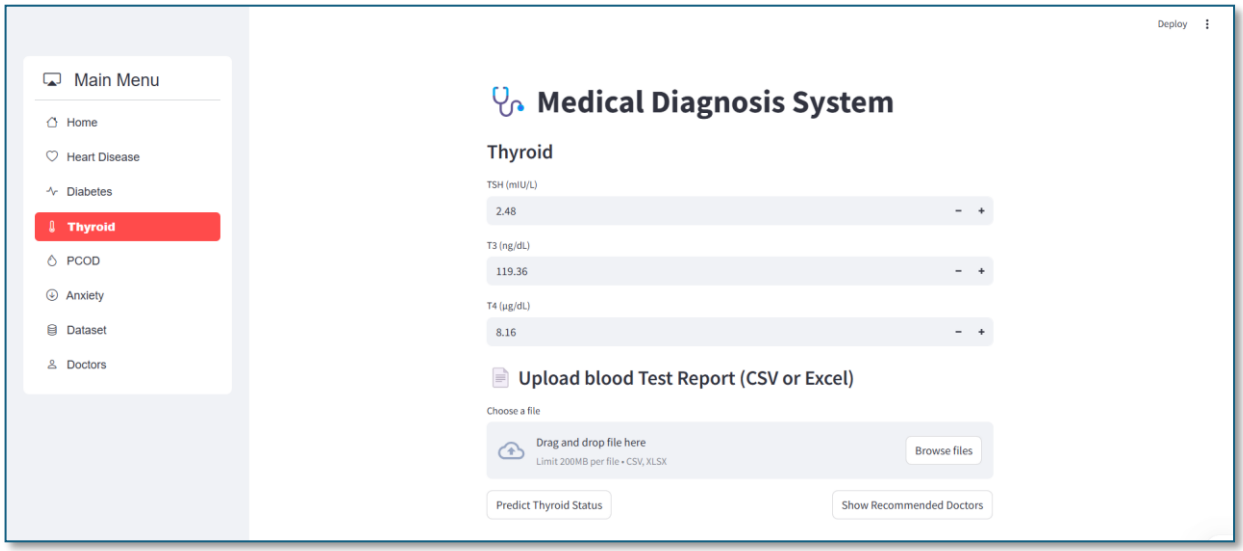
  The modular design supports future expansion and integration with machine learning models.

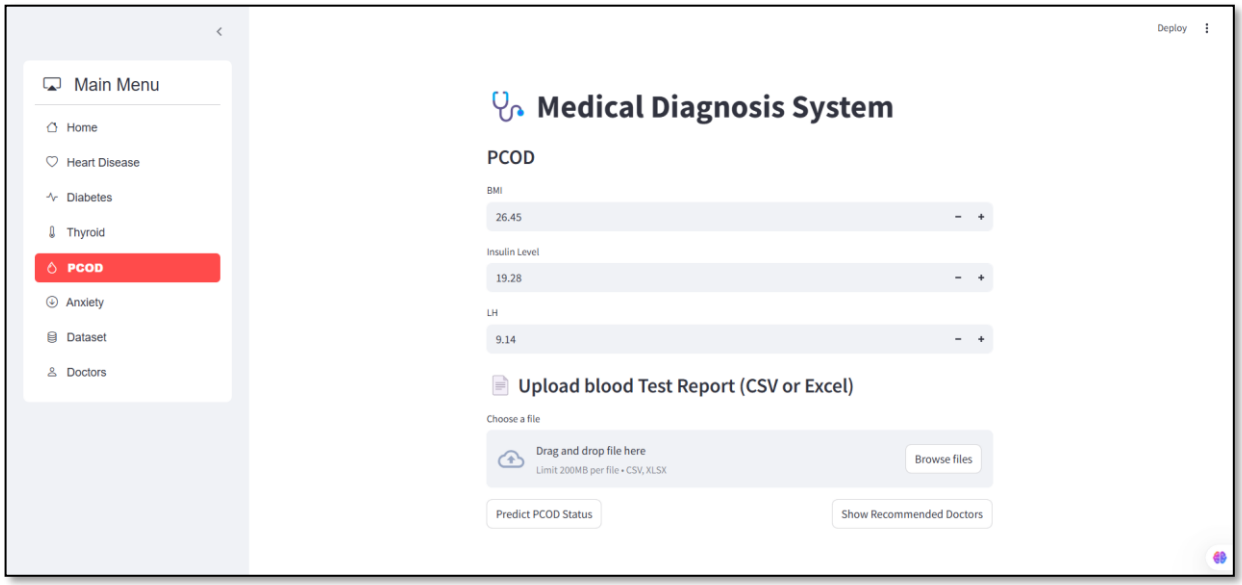
➢ Screenshots of the user interface

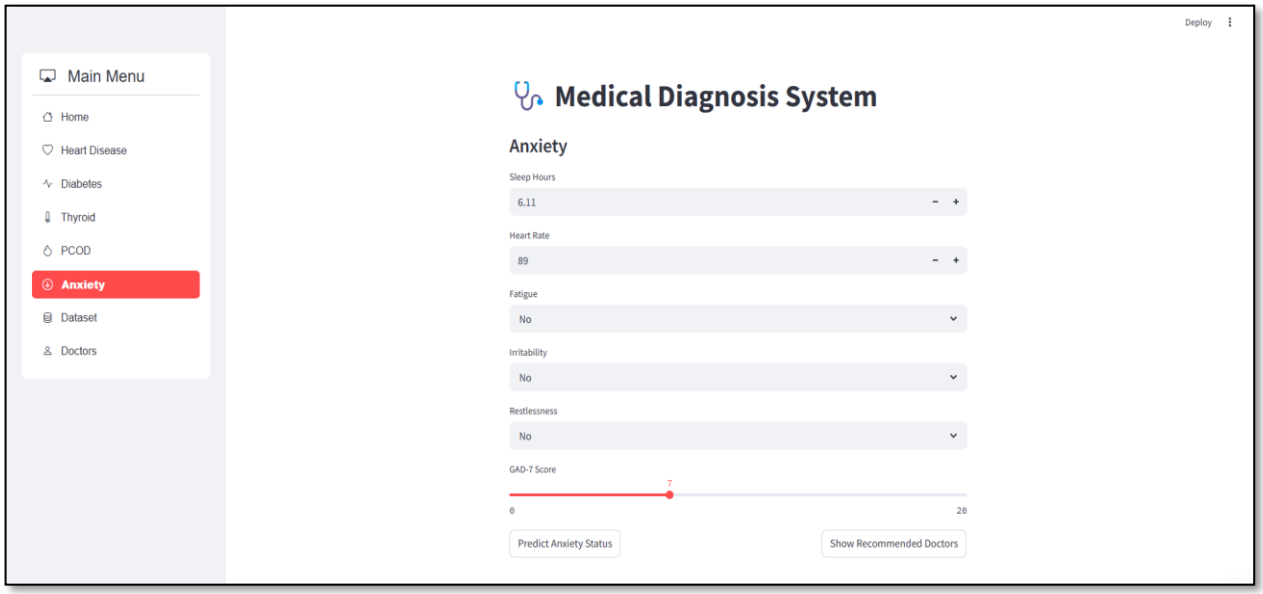


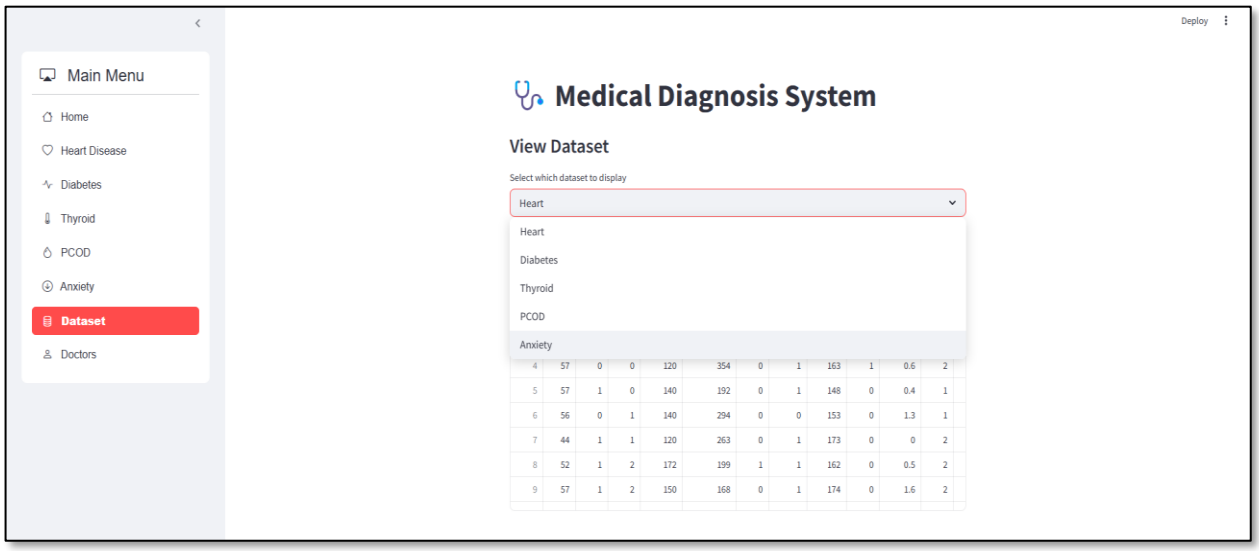



Deploy
Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Medical Diagnosis System
Diabetes
Glucose
120
BMI
31.99
Age
33
Blood Pressure
69
Upload Blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict Diabetes Status
Show Recommended Doctors

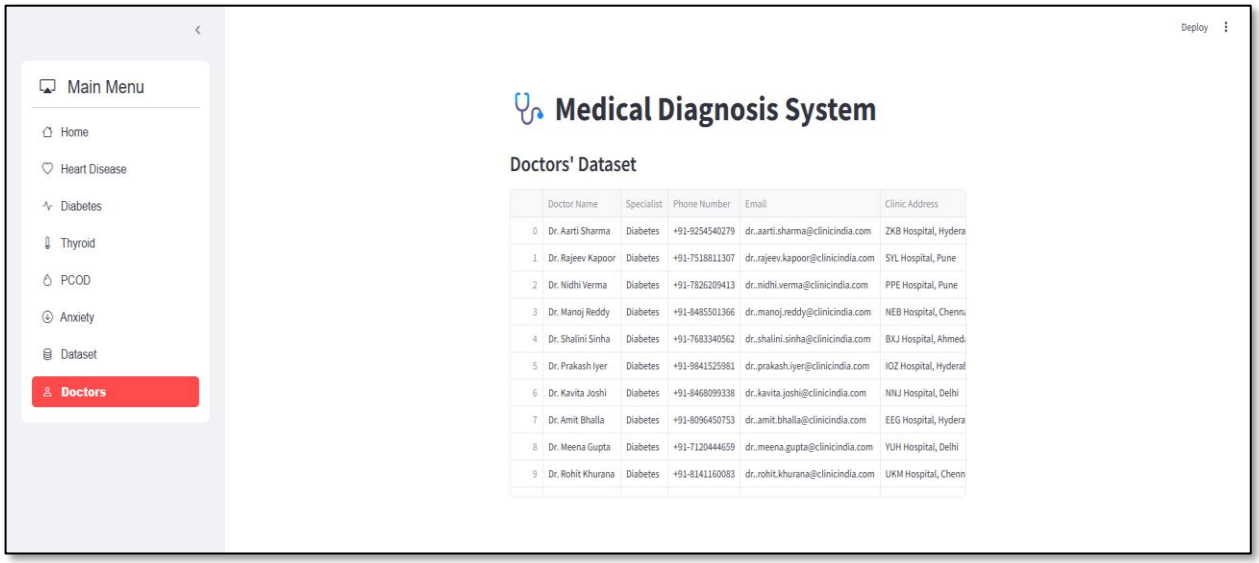

Deploy
Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Medical Diagnosis System
Thyroid
TSH (mIU/L)
2.48
T3 (ng/dL)
119.36
T4 (µg/dL)
8.16
Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict Thyroid Status
Show Recommended Doctors


Deploy
Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Medical Diagnosis System
PCOD
BMI
26.45
Insulin Level
19.28
LH
9.14
Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict PCOD Status
Show Recommended Doctors





4	57	0	0	120	354	0	1	163	1	0.6	2
5	57	1	0	140	192	0	1	148	0	0.4	1
6	56	0	1	140	294	0	0	153	0	1.3	1
7	44	1	1	120	263	0	1	173	0	0	2
8	52	1	2	172	199	1	1	162	0	0.5	2
9	57	1	2	150	168	0	1	174	0	1.6	2



	Doctor Name	Specialist	Phone Number	Email	Clinic Address
0	Dr. Aarti Sharma	Diabetes	+91-9254540279	dr..aarti.sharma@clinicindia.com	ZKB Hospital, Hydera
1	Dr. Rajeev Kapoor	Diabetes	+91-7518811307	dr..rajeev.kapoor@clinicindia.com	SYL Hospital, Pune
2	Dr. Nidhi Verma	Diabetes	+91-7826209413	dr..nidhi.verma@clinicindia.com	PPE Hospital, Pune
3	Dr. Manoj Reddy	Diabetes	+91-8485501366	dr..manoj.reddy@clinicindia.com	NEB Hospital, Chenn
4	Dr. Shalini Sinha	Diabetes	+91-7683340562	dr..shalini.sinha@clinicindia.com	BXJ Hospital, Ahmed
5	Dr. Prakash Iyer	Diabetes	+91-9841525981	dr..prakash.iyer@clinicindia.com	IOZ Hospital, Hyderal
6	Dr. Kavita Joshi	Diabetes	+91-8468099338	dr..kavita.joshi@clinicindia.com	NNJ Hospital, Delhi
7	Dr. Amit Bhalla	Diabetes	+91-8096450753	dr..amit.bhalla@clinicindia.com	EEG Hospital, Hydera
8	Dr. Meena Gupta	Diabetes	+91-7120444659	dr..meena.gupta@clinicindia.com	YUH Hospital, Delhi
9	Dr. Rohit Khurana	Diabetes	+91-8141160083	dr..rohit.khurana@clinicindia.com	UKM Hospital, Chenn

❖ Testing & Results

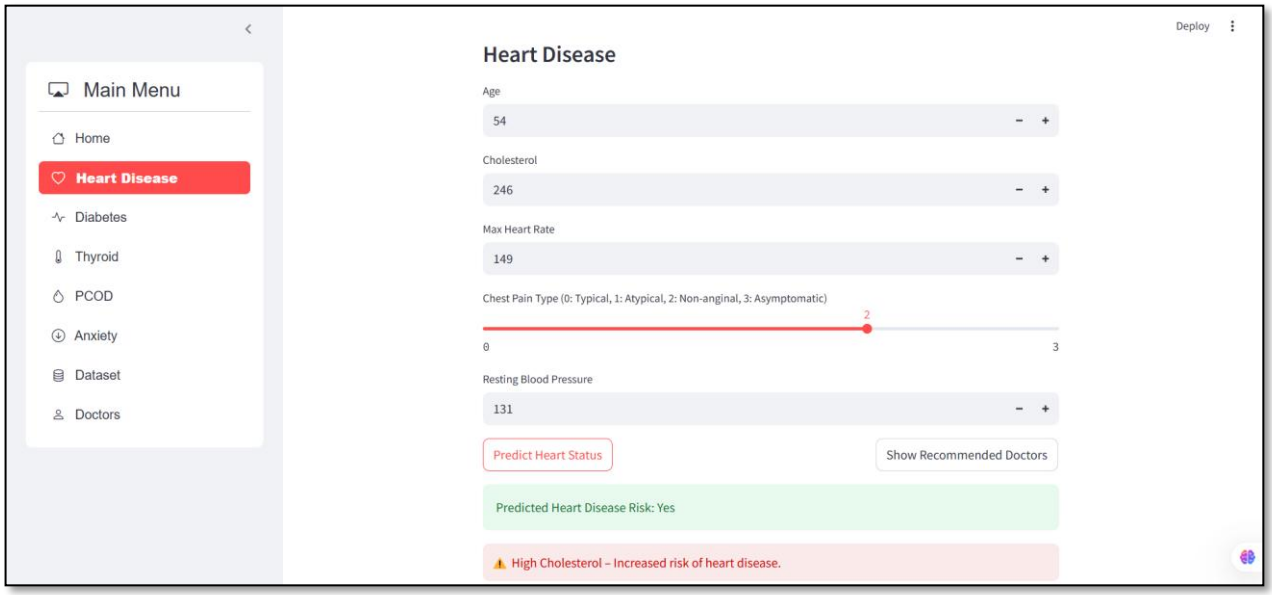
To ensure the reliability and accuracy of the medical diagnosis system, several testing strategies were employed throughout development. The system predicts five health conditions—**Heart Disease**, **Diabetes**, **Thyroid Disorders**, **PCOD**, and **Anxiety**—using Python-based fuzzy logic models.

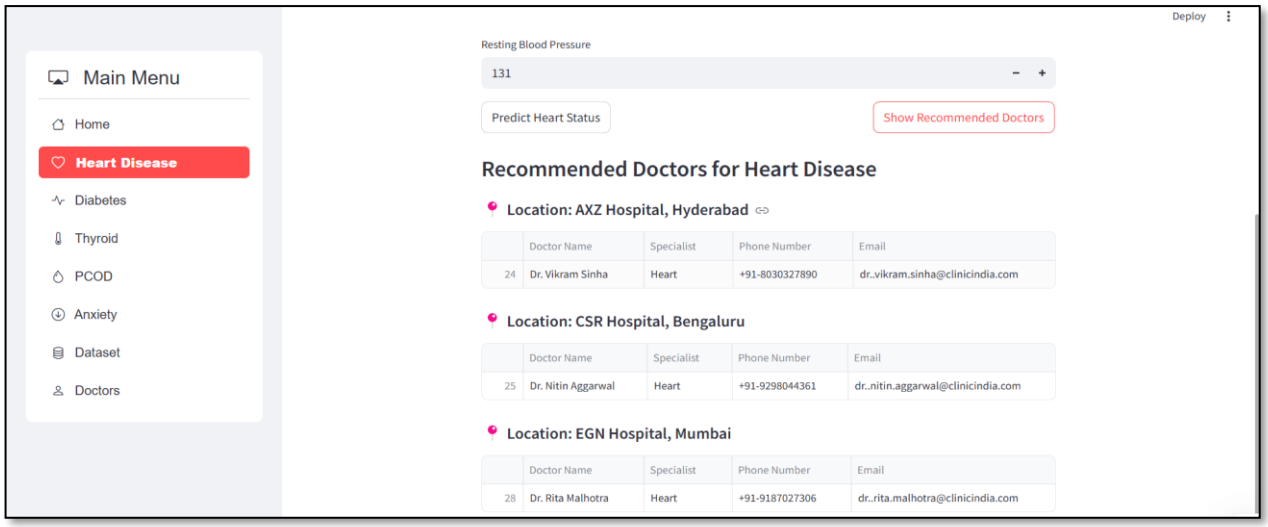
➢ Testing Strategies

- **Unit Testing:**
  Individual components such as data preprocessing functions, fuzzy logic modules, and input validation routines were tested independently to verify correctness and robustness.
- **Integration Testing:**
  Ensured smooth interaction between modules, including the user interface, fuzzy inference engine, and database layer.
- **System Testing:**
  The complete system was tested end-to-end to validate predictions across all five medical conditions. Multiple test scenarios were used, including normal, borderline, and extreme input cases.
- **Data Validation Testing:**
  The datasets from **UCI** and **Kaggle** were cross-checked for missing or inconsistent values. Preprocessing steps were tested to ensure clean and normalized input to the fuzzy logic models.

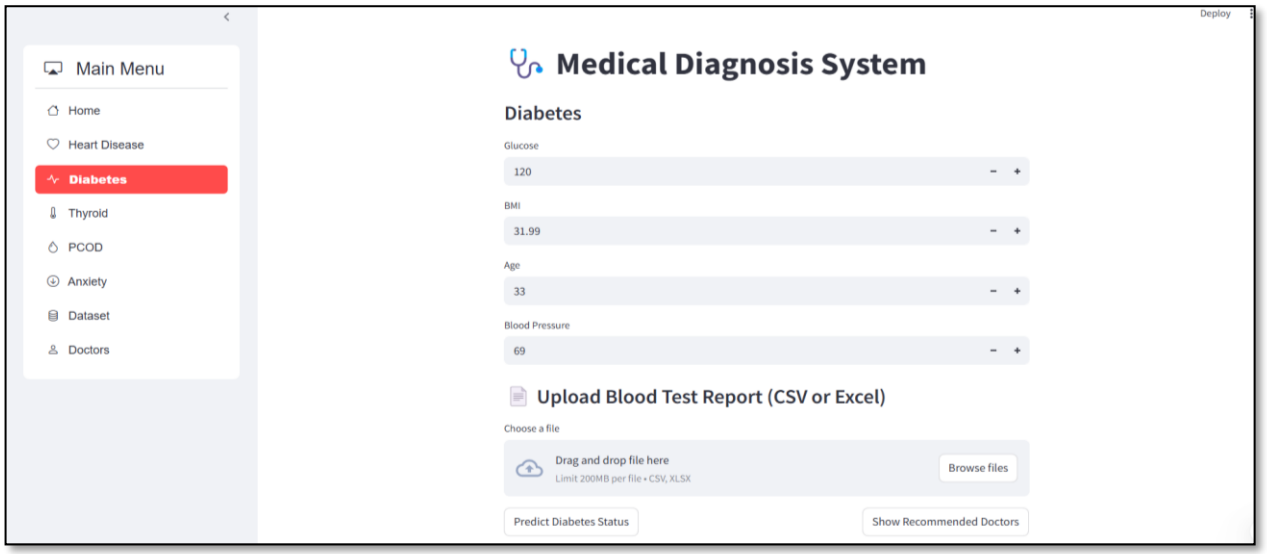
➢ Results

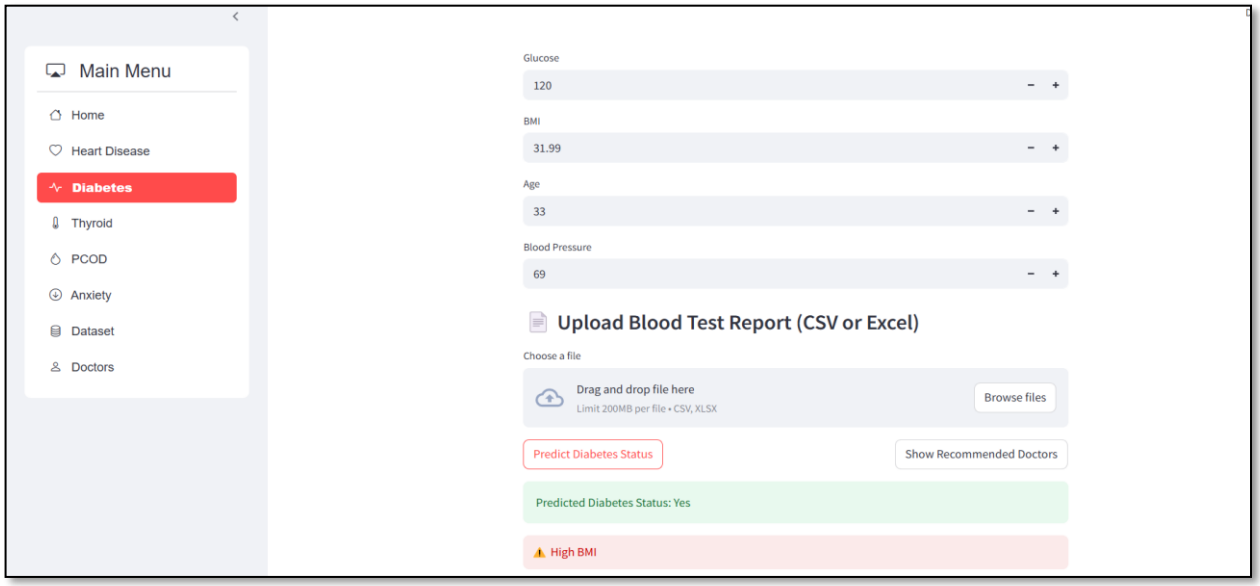
1. Heart Disease


Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Heart Disease
Age
54
Cholesterol
246
Max Heart Rate
149
Chest Pain Type (0: Typical, 1: Atypical, 2: Non-anginal, 3: Asymptomatic)
2
0
3
Resting Blood Pressure
131
Predict Heart Status
Show Recommended Doctors
Predicted Heart Disease Risk: Yes
High Cholesterol – Increased risk of heart disease.

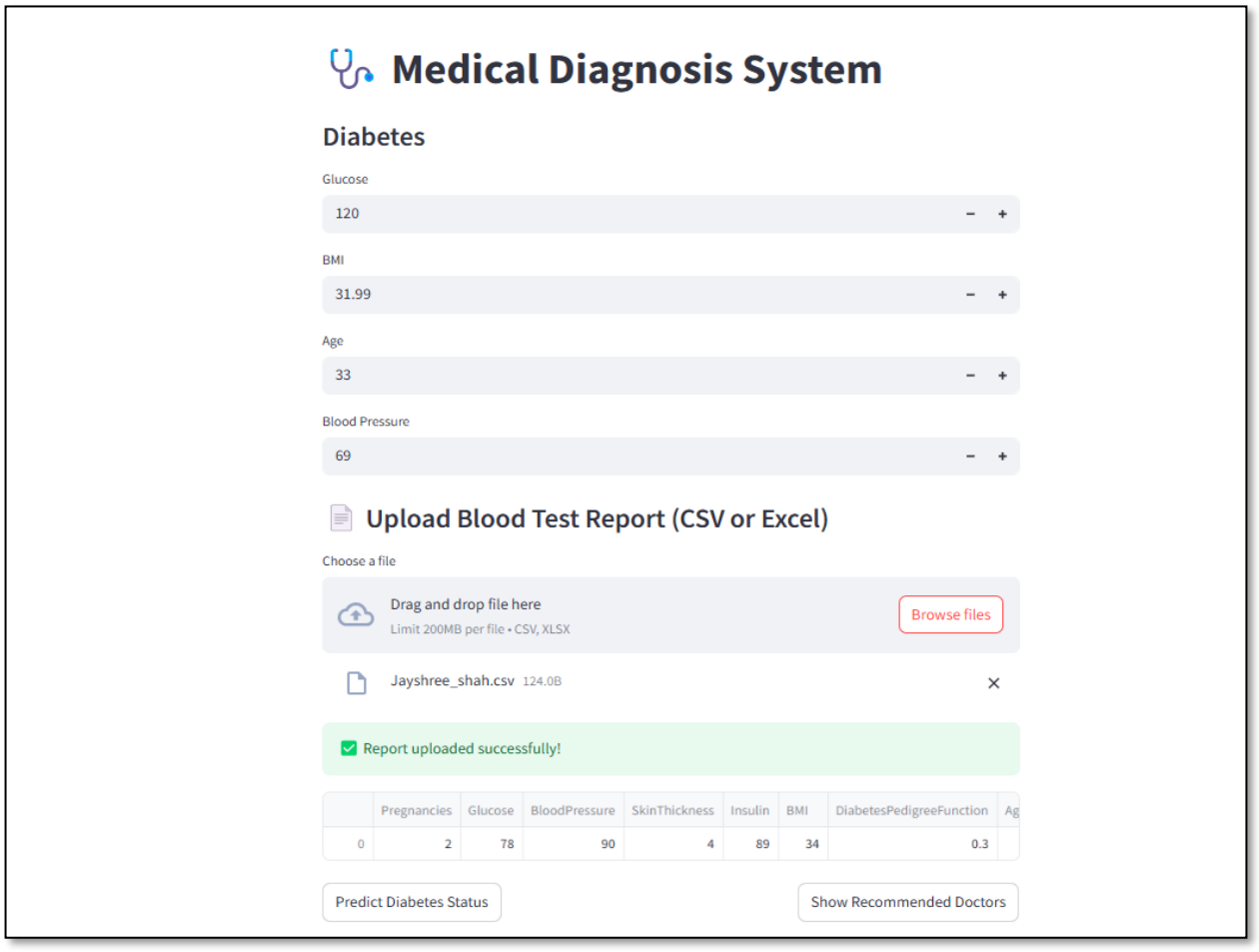

Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Resting Blood Pressure
131
Predict Heart Status
Show Recommended Doctors
Recommended Doctors for Heart Disease
Location: AXZ Hospital, Hyderabad
Doctor Name Specialist Phone Number Email
24 Dr. Vikram Sinha Heart +91-8030327890 dr..vikram.sinha@clinicindia.com
Location: CSR Hospital, Bengaluru
Doctor Name Specialist Phone Number Email
25 Dr. Nitin Aggarwal Heart +91-9298044361 dr..nitin.aggarwal@clinicindia.com
Location: EGN Hospital, Mumbai
Doctor Name Specialist Phone Number Email
28 Dr. Rita Malhotra Heart +91-9187027306 dr..rita.malhotra@clinicindia.com

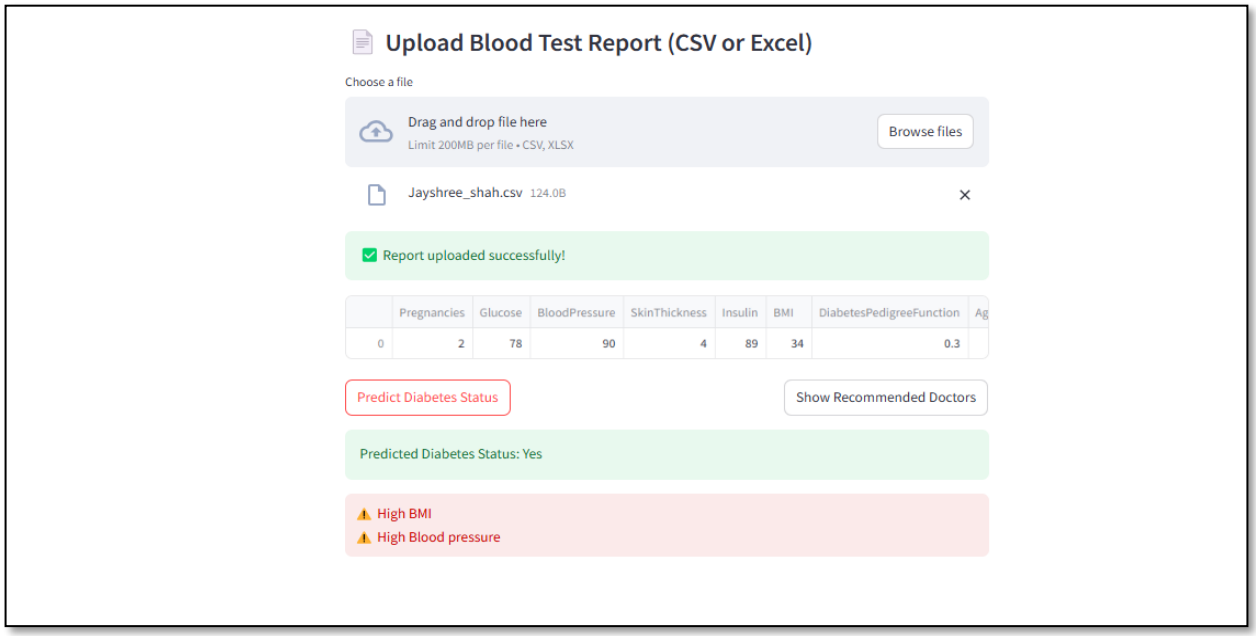
2. Diabetes

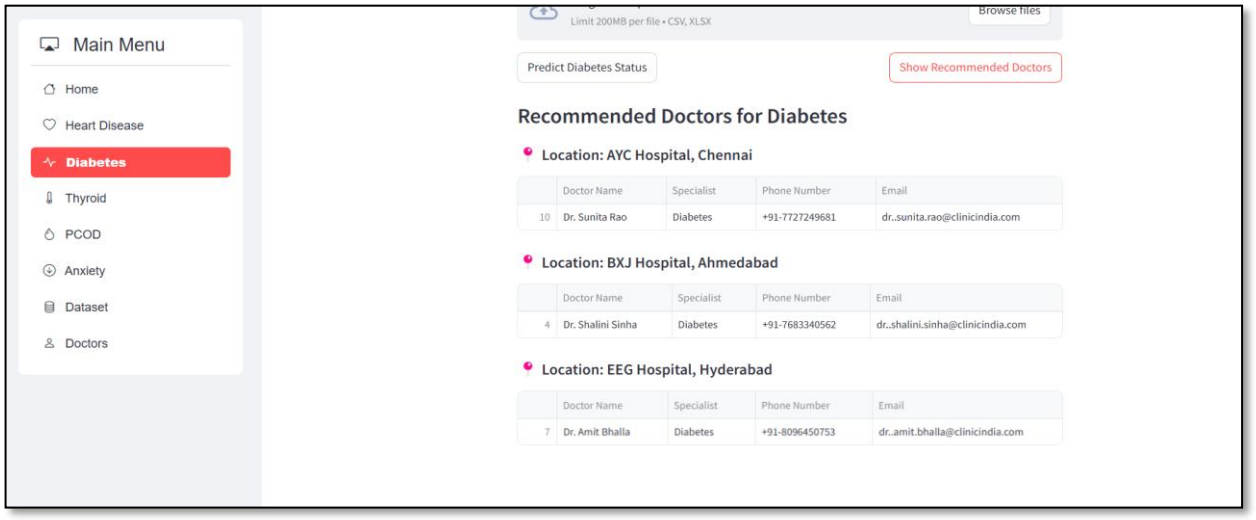

Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Medical Diagnosis System
Diabetes
Glucose
120
BMI
31.99
Age
33
Blood Pressure
69
Upload Blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict Diabetes Status
Show Recommended Doctors



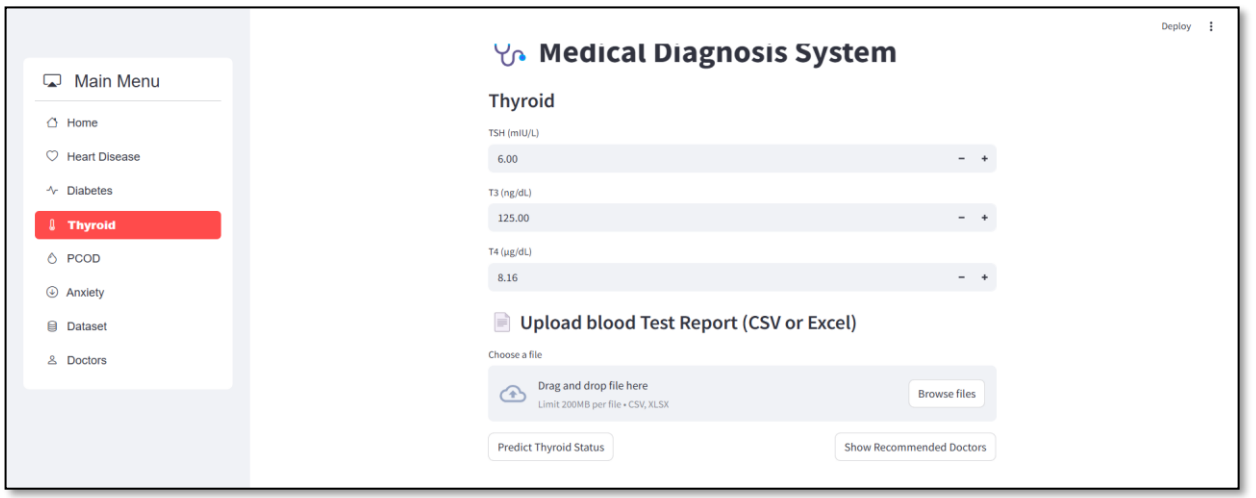
With Report Upload

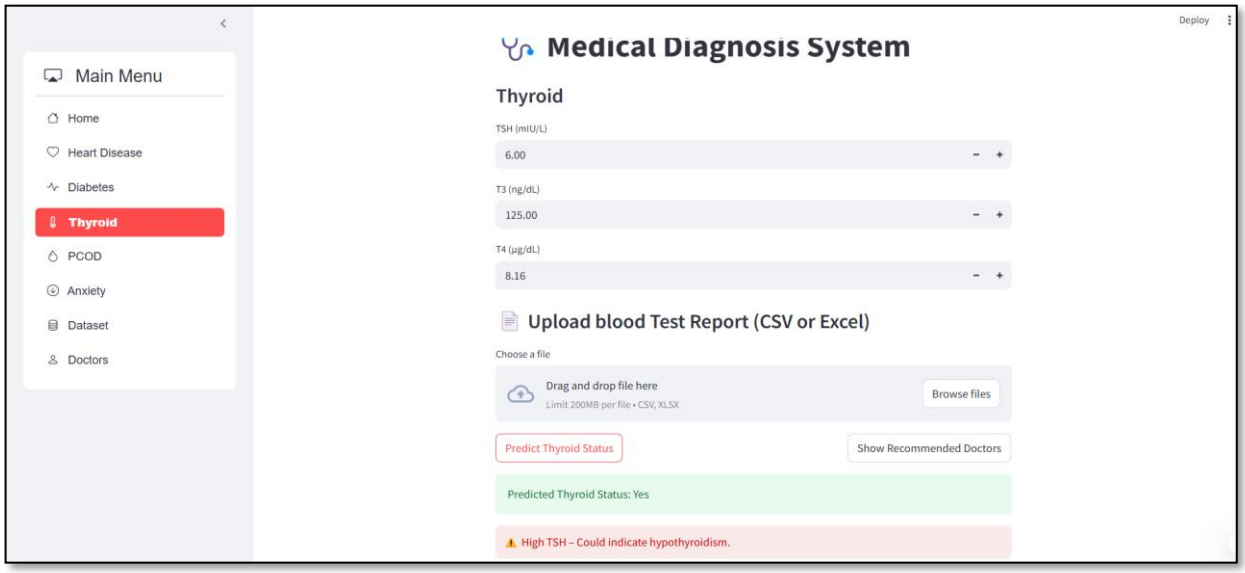



Upload Blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Jayshree_shah.csv 124.0B
Report uploaded successfully!
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Ag
0 2 78 90 4 89 34 0.3
Predict Diabetes Status
Show Recommended Doctors
Predicted Diabetes Status: Yes
High BMI
High Blood pressure

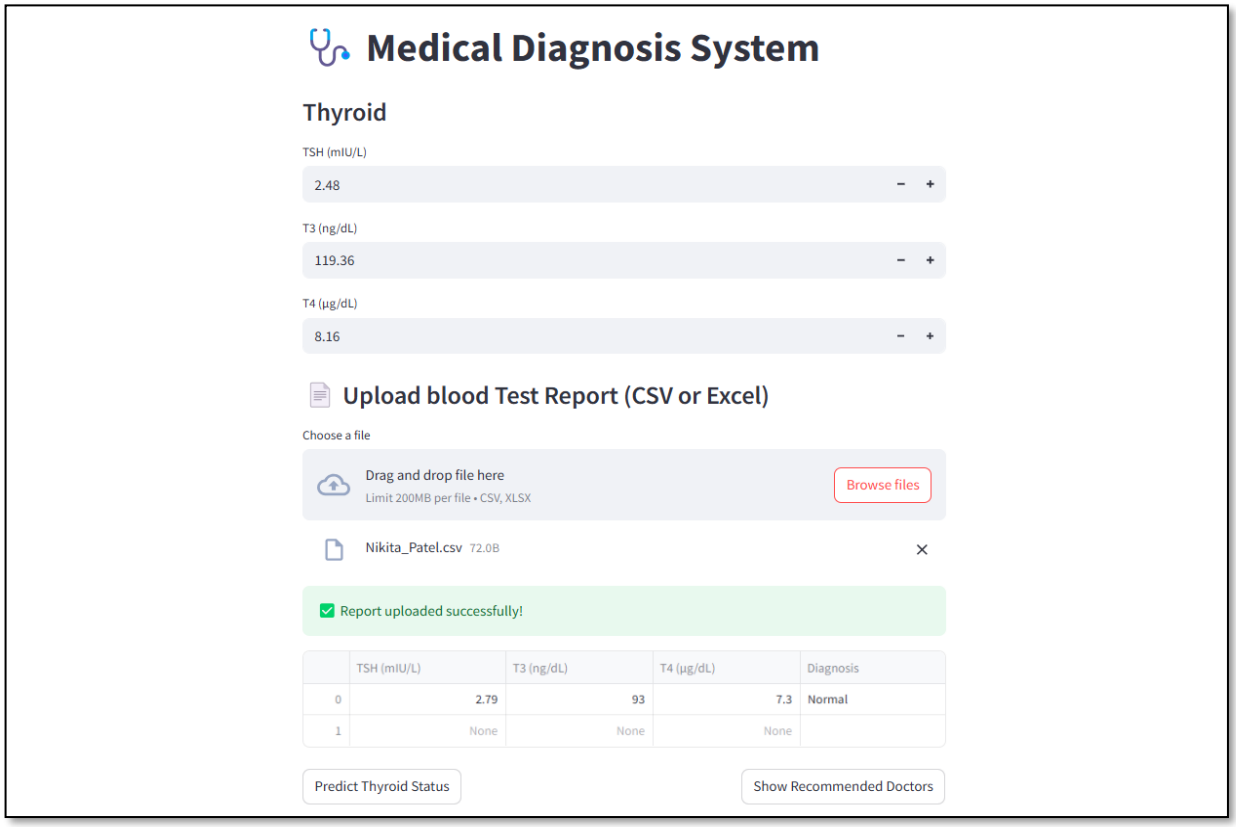

Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Predict Diabetes Status
Show Recommended Doctors
Recommended Doctors for Diabetes
Location: AYC Hospital, Chennai
Doctor Name Specialist Phone Number Email
10 Dr. Sunita Rao Diabetes +91-7727249681 dr..sunita.rao@clinicindia.com
Location: BXJ Hospital, Ahmedabad
Doctor Name Specialist Phone Number Email
4 Dr. Shalini Sinha Diabetes +91-7683340562 dr..shalini.sinha@clinicindia.com
Location: EEG Hospital, Hyderabad
Doctor Name Specialist Phone Number Email
7 Dr. Amit Bhalla Diabetes +91-8096450753 dr..amit.bhalla@clinicindia.com

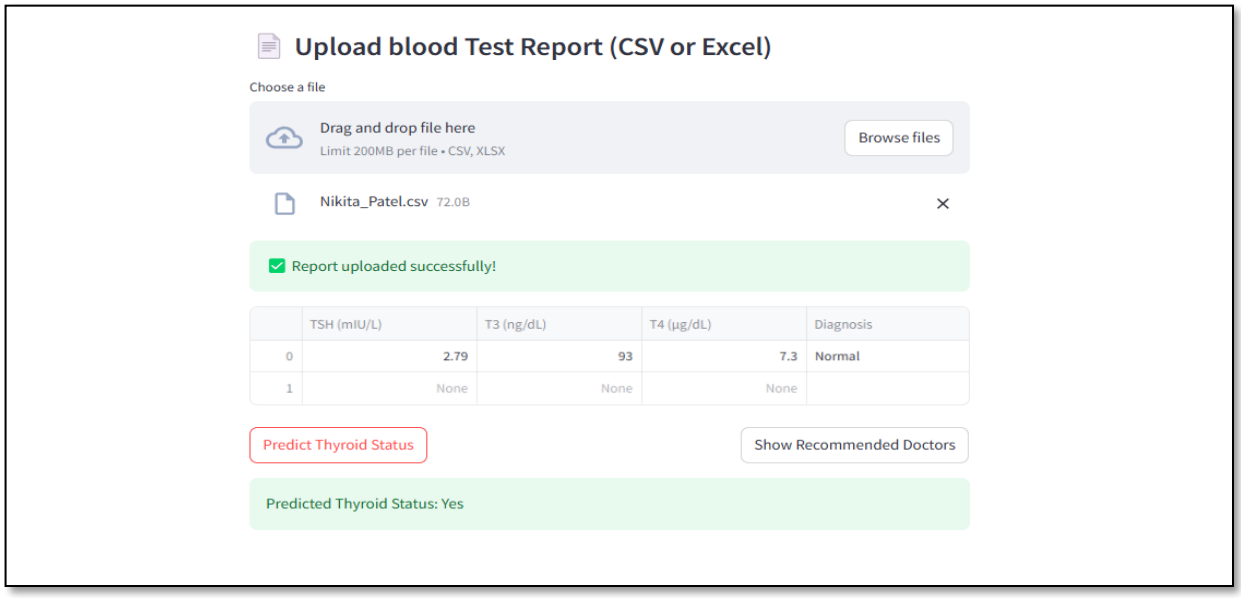
3. Thyroid

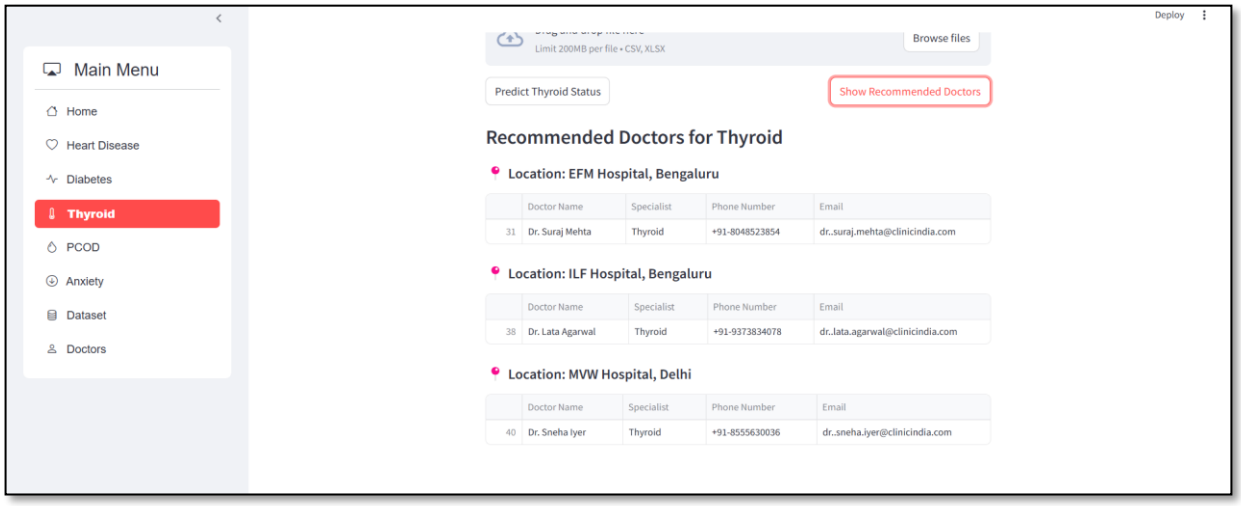

Deploy
Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Medical Diagnosis System
Thyroid
TSH (mIU/L)
6.00
T3 (ng/dL)
125.00
T4 (µg/dL)
8.16
Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict Thyroid Status
Show Recommended Doctors



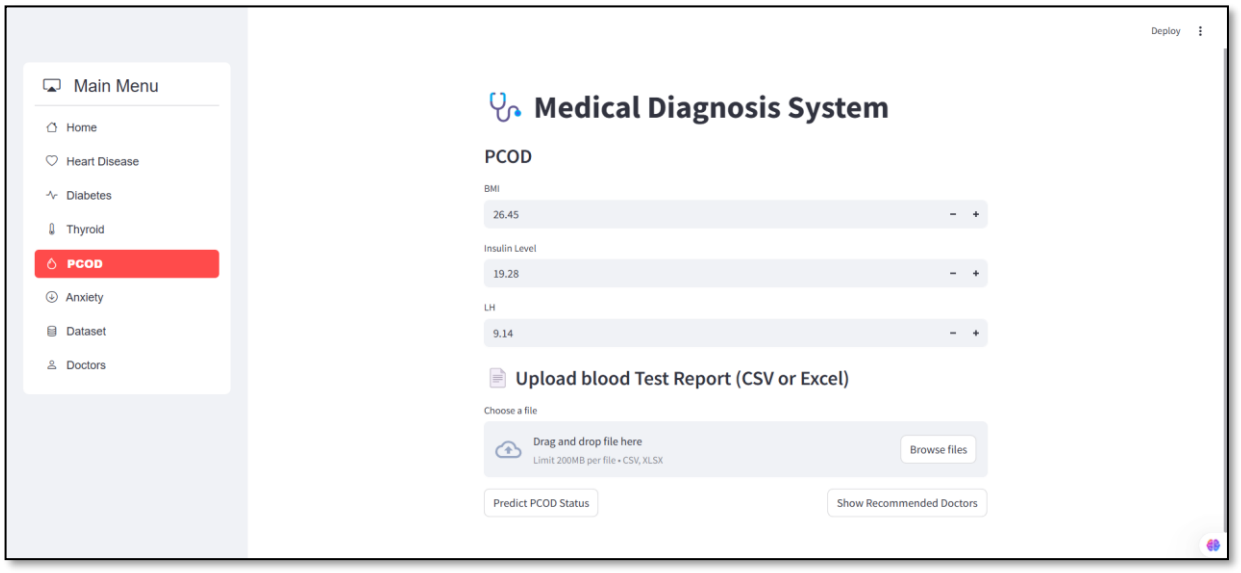
With Report Upload

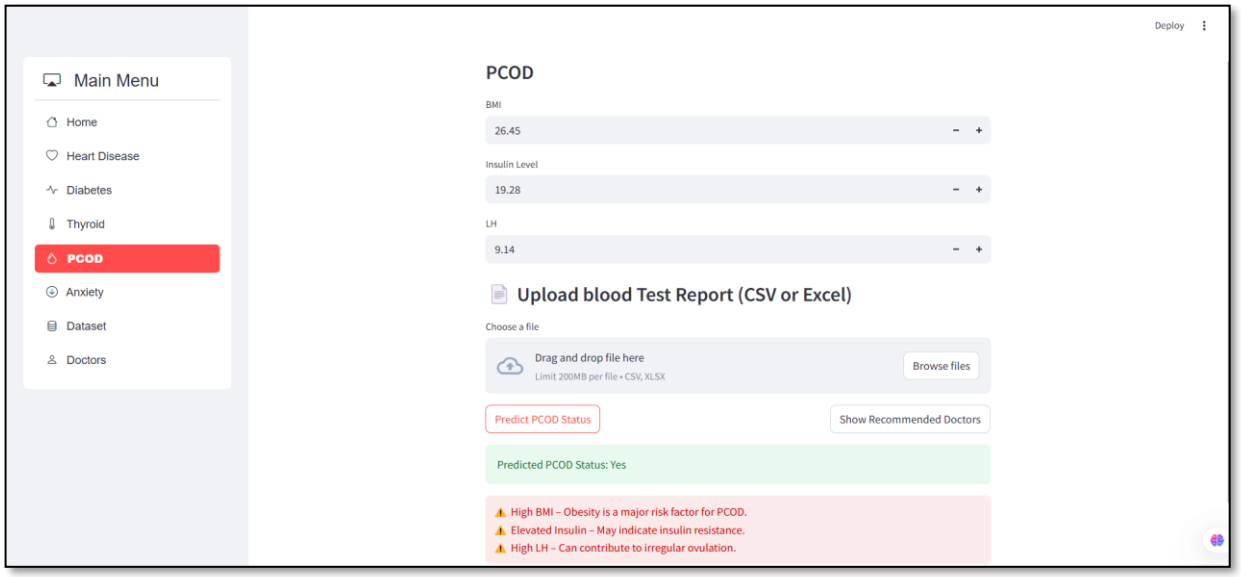



Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Nikita_Patel.csv 72.0B
Report uploaded successfully!
TSH (mIU/L) T3 (ng/dL) T4 (µg/dL) Diagnosis
0 2.79 93 7.3 Normal
1 None None None
Predict Thyroid Status
Show Recommended Doctors
Predicted Thyroid Status: Yes

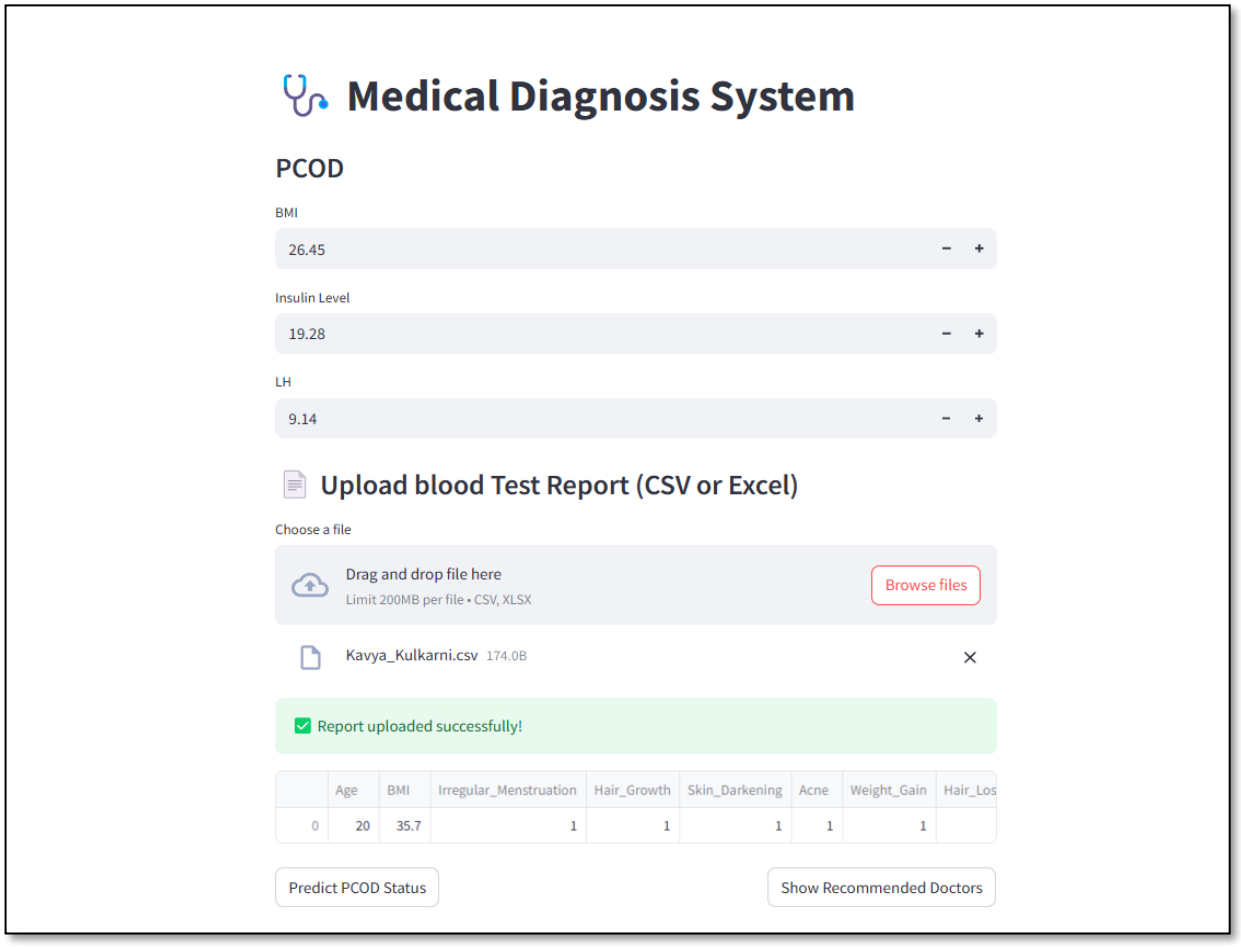

Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Browse files
Predict Thyroid Status
Show Recommended Doctors
Recommended Doctors for Thyroid
Location: EFM Hospital, Bengaluru
Doctor Name Specialist Phone Number Email
31 Dr. Suraj Mehta Thyroid +91-8048523854 dr..suraj.mehta@clinicindia.com
Location: ILF Hospital, Bengaluru
Doctor Name Specialist Phone Number Email
38 Dr. Lata Agarwal Thyroid +91-9373834078 dr..lata.agarwal@clinicindia.com
Location: MVW Hospital, Delhi
Doctor Name Specialist Phone Number Email
40 Dr. Sneha Iyer Thyroid +91-8555630036 dr..sneha.iyer@clinicindia.com

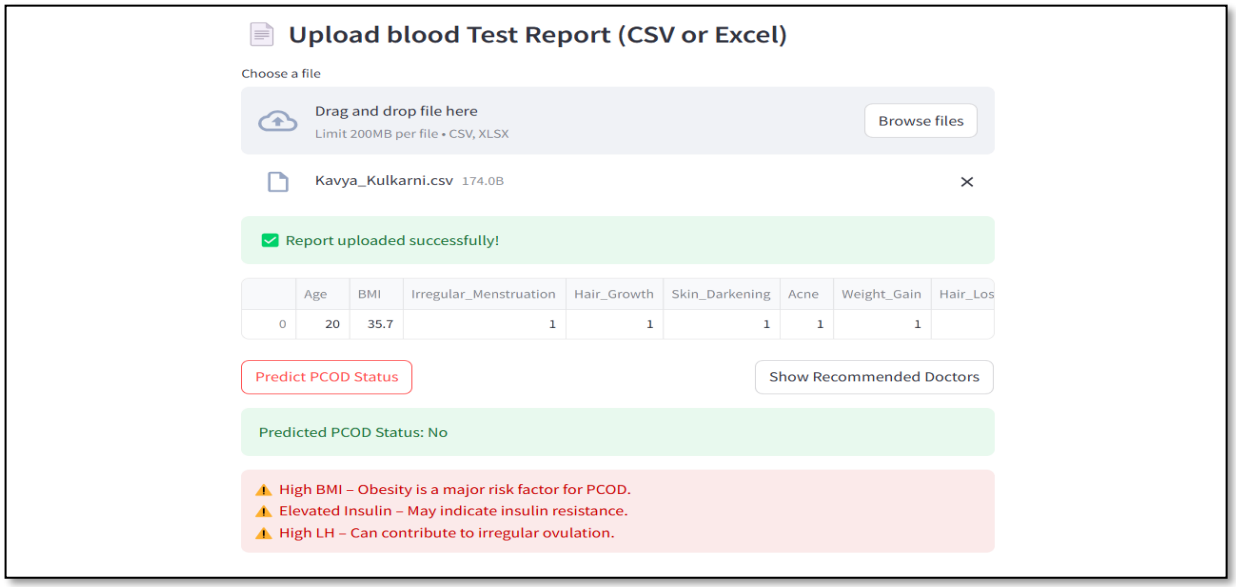
4. PCOD

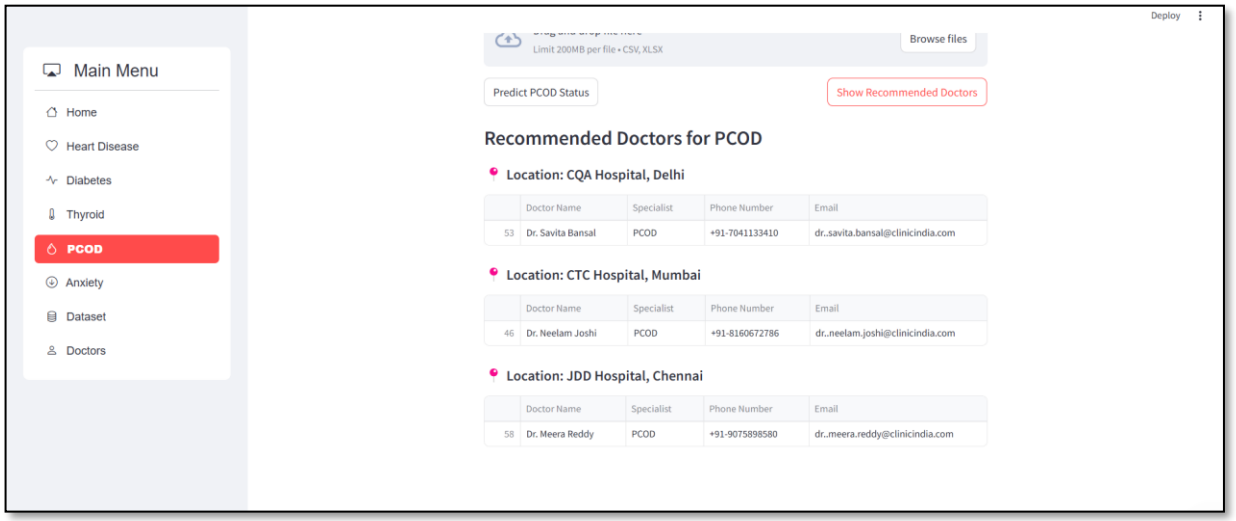

Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Medical Diagnosis System
PCOD
BMI
26.45
Insulin Level
19.28
LH
9.14
Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Predict PCOD Status
Show Recommended Doctors



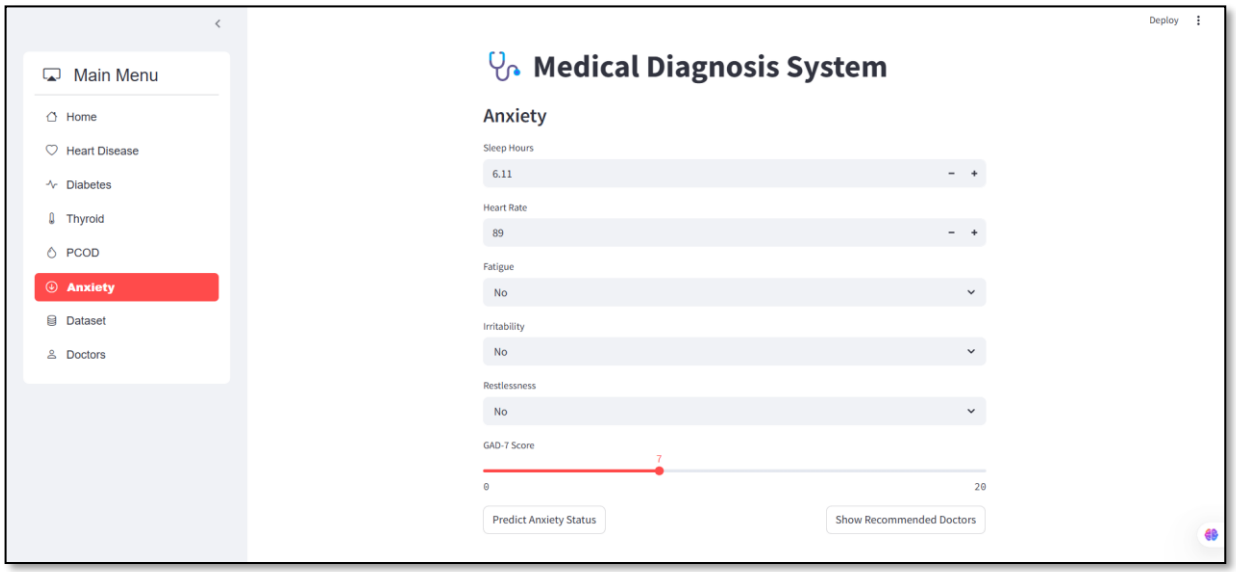
With Report Upload

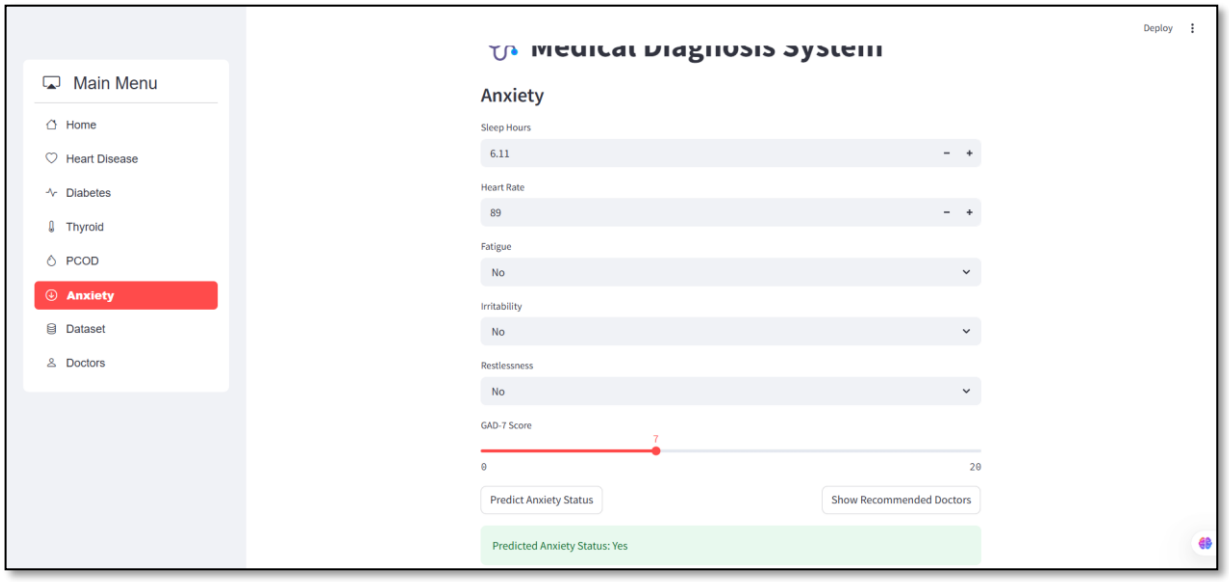


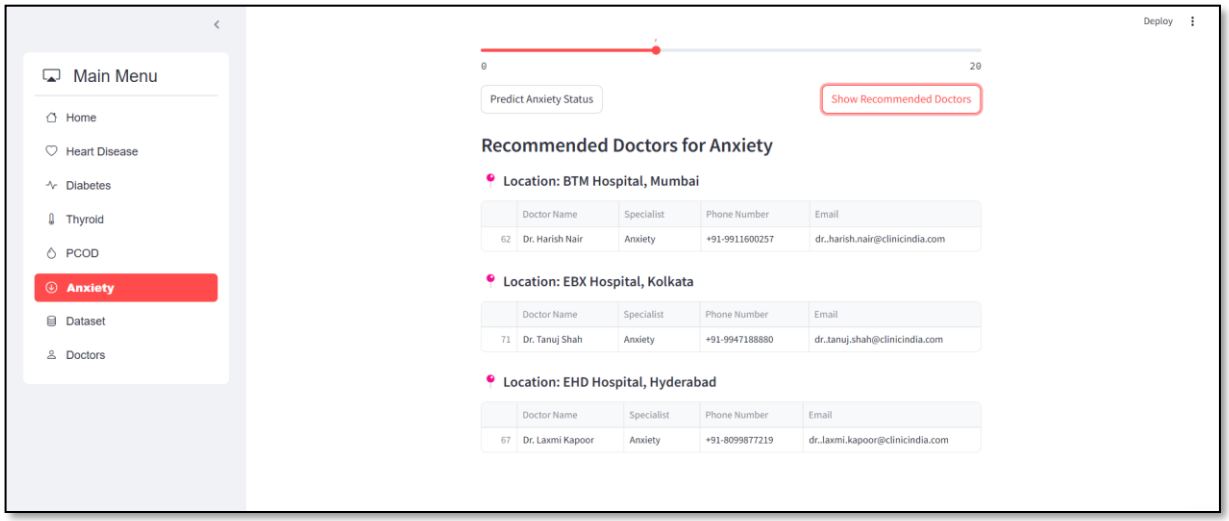

Upload blood Test Report (CSV or Excel)
Choose a file
Drag and drop file here
Limit 200MB per file • CSV, XLSX
Browse files
Kavya_Kulkarni.csv 174.0B
Report uploaded successfully!
Age BMI Irregular_Menstruation Hair_Growth Skin_Darkening Acne Weight_Gain Hair_Los
0 20 35.7 1 1 1 1 1
Predict PCOD Status
Show Recommended Doctors
Predicted PCOD Status: No
High BMI – Obesity is a major risk factor for PCOD.
Elevated Insulin – May indicate insulin resistance.
High LH – Can contribute to irregular ovulation.


Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Browse files
Limit 200MB per file • CSV, XLSX
Predict PCOD Status
Show Recommended Doctors
Recommended Doctors for PCOD
Location: CQA Hospital, Delhi
Doctor Name Specialist Phone Number Email
53 Dr. Savita Bansal PCOD +91-7041133410 dr..savita.bansal@clinicindia.com
Location: CTC Hospital, Mumbai
Doctor Name Specialist Phone Number Email
46 Dr. Neelam Joshi PCOD +91-8160672786 dr..neelam.joshi@clinicindia.com
Location: JDD Hospital, Chennai
Doctor Name Specialist Phone Number Email
58 Dr. Meera Reddy PCOD +91-9075898580 dr..meera.reddy@clinicindia.com

5. Anxiety


Main Menu
Home
Heart Disease
Diabetes
Thyroid
PCOD
Anxiety
Dataset
Doctors
Deploy
Medical Diagnosis System
Anxiety
Sleep Hours
6.11
Heart Rate
89
Fatigue
No
Irritability
No
Restlessness
No
GAD-7 Score
7
0
20
Predict Anxiety Status
Show Recommended Doctors





❖ Discussion & Challenges

➢ Problems Faced and Solutions

- **Data Issues:** Public datasets were often incomplete or imbalanced, especially for PCOD and anxiety.
  Solution: Applied data cleaning, imputation, and oversampling (e.g., SMOTE) to improve dataset quality.
- **Multi-Disease Fuzzy Logic Integration:** Each condition required unique fuzzy rules and parameter ranges.
  Solution: Developed separate fuzzy systems for each disease and integrated them under a common interface.
- **Overlapping Symptoms:** Shared symptoms caused diagnostic confusion.
  Solution: Introduced rule-based prioritization to differentiate conditions.

- **Accuracy Limitations:** Pure fuzzy logic had constraints in achieving high precision.
  Solution: Explored hybrid approaches with machine learning for future enhancement.

➢ Possible Improvements

- **Real-Time Data:** Integrate wearable or live health data for better predictions.
- **More Diseases:** Expand coverage to include additional conditions.
- **Better UI:** Develop a user-friendly interface using Flask/Django.
- **Hybrid Models:** Combine fuzzy logic with ML models for improved accuracy.
- **Clinical Testing:** Validate system outcomes with healthcare professionals.

❖ Conclusion

➢ Summary of Findings

We developed a medical diagnosis system using Python and fuzzy logic that predicts five major health conditions: diabetes, heart disease, thyroid disorders, PCOD, and anxiety. The system utilizes publicly available datasets to interpret user symptoms through a rule-based fuzzy inference approach. Our results show that fuzzy logic is effective in handling the uncertainties and overlaps often present in medical diagnosis, providing a practical tool for early screening and awareness.

➢ Future Scope of the Project

The project offers several opportunities for enhancement, both from a technical and user experience perspective:

- **Video Consultation:** Integrate video conferencing capabilities to connect patients with healthcare professionals directly from the platform.
- **Diet and Lifestyle Plans:** Offer AI-generated personalized diet recommendations and lifestyle improvement plans based on diagnosis results.
- **Symptom Tracker:** Add daily or weekly symptom logging features for long-term monitoring.
- **Interactive Chatbot:** Implement a virtual assistant to guide users through symptom input and answer health-related queries.
- **User Profile and Reports:** Allow users to download and share detailed diagnostic reports with doctors.
- **Multi-language Support:** Enhance accessibility by adding support for regional languages and voice-based inputs.

❖ References

- Google. (n.d.). *Google search*. Retrieved April 29, 2025, from https://www.google.com
- Wikipedia contributors. (n.d.). *Wikipedia, the free encyclopedia*. Wikipedia. Retrieved April 29, 2025, from https://www.wikipedia.org
- Kaggle. (n.d.). *Datasets for data science and machine learning*. Retrieved April 29, 2025, from https://www.kaggle.com
- Mayo Clinic. (n.d.). *Symptom Checker*. Retrieved April 29, 2025, from https://www.mayoclinic.org/symptom-checker/select-symptom/itt-20009075
- Patient.info. (n.d.). *Symptom Checker*. Retrieved April 29, 2025, from https://patient.info/symptom-checker

❖ Appendix

Medical_Diagnosis_System.py 3

Medical_Diagnosis_System.py > load_dataset

```python
import numpy as np
import pandas as pd
import skfuzzy as fuzz
import skfuzzy.control as ctrl
import streamlit as st
from streamlit_option_menu import option_menu

# --- Load datasets ---
def load_dataset(dataset_type):
    try:
        if dataset_type == "diabetes":
            return pd.read_csv("diabetes.csv")
        elif dataset_type == "heart":
            return pd.read_csv("heart.csv")
        elif dataset_type == "thyroid":
            file_path = "thyroid_dataset_300_rows.csv"
            df = pd.read_csv(file_path)
            df.columns = df.columns.str.strip()
            df.rename(columns={
                'TSH (mIU/L)': 'TSH',
                'T3 (ng/dL)': 'T3',
                'T4 (µg/dL)': 'T4'
            }, inplace=True)
            return df
        elif dataset_type == "pcod":
            return pd.read_csv("pcod.csv")
        elif dataset_type == "anxiety":
            return pd.read_csv("anxiety_dataset_300_modified.csv")
    except Exception as e:
        st.error(f"Error loading {dataset_type} dataset: {e}")
        return None


# --- Fuzzy Logic for Heart Disease ---
def create_fuzzy_heart(df):
    age = ctrl.Antecedent(np.arange(df['age'].min(), df['age'].max() + 1, 1), 'age')
    cholesterol = ctrl.Antecedent(np.arange(df['Cholesterol'].min(), df['Cholesterol'].max() + 1, 1), 'cholesterol')
```

Medical_Diagnosis_System.py > ...

```python
def create_fuzzy_heart(df):
    age = ctrl.Antecedent(np.arange(df['age'].min(), df['age'].max() + 1, 1), 'age')
    cholesterol = ctrl.Antecedent(np.arange(df['Cholesterol'].min(), df['Cholesterol'].max() + 1, 1), 'cholesterol')
    thalach = ctrl.Antecedent(np.arange(df['thalach'].min(), df['thalach'].max() + 1, 1), 'thalach')
    chest_pain = ctrl.Antecedent(np.arange(0, 4, 1), 'chest_pain')
    resting_bp = ctrl.Antecedent(np.arange(df['trestbps'].min(), df['trestbps'].max() + 1, 1), 'resting_bp')
    heart_risk = ctrl.Consequent(np.arange(0, 101, 1), 'heart_risk')

    age.automf(3)
    cholesterol.automf(3)
    thalach.automf(3)
    chest_pain.automf(3)
    resting_bp.automf(3)
    heart_risk.automf(3)

    rules = [
        ctrl.Rule(age['poor'] & cholesterol['poor'], heart_risk['poor']),
        ctrl.Rule(chest_pain['poor'] & resting_bp['poor'], heart_risk['poor']),
        ctrl.Rule(age['average'] & cholesterol['average'], heart_risk['average']),
        ctrl.Rule(thalach['good'] & cholesterol['good'], heart_risk['good']),
        ctrl.Rule(chest_pain['good'] & thalach['good'] & age['good'], heart_risk['good']),
    ]

    heart_ctrl = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(heart_ctrl)


# --- Fuzzy Logic for Diabetes ---
def create_fuzzy_diabetes(df):
    glucose = ctrl.Antecedent(np.arange(df['Glucose'].min(), df['Glucose'].max() + 1, 1), 'glucose')
    bmi = ctrl.Antecedent(np.arange(df['BMI'].min(), df['BMI'].max() + 1, 1), 'bmi')
    age = ctrl.Antecedent(np.arange(df['Age'].min(), df['Age'].max() + 1, 1), 'age')
    blood_pressure = ctrl.Antecedent(np.arange(df['BloodPressure'].min(), df['BloodPressure'].max() + 1, 1), 'blood_pressure')
    diabetes_risk = ctrl.Consequent(np.arange(0, 101, 1), 'diabetes_risk')

    glucose.automf(3)
    bmi.automf(3)
```

Medical_Diagnosis_System.py > ...

```python
def create_fuzzy_diabetes(df):
    bmi.automf(3)
    age.automf(3)
    blood_pressure.automf(3)
    diabetes_risk.automf(3)

    rules = [
        ctrl.Rule(glucose['poor'] & bmi['poor'] & age['poor'], diabetes_risk['poor']),
        ctrl.Rule(glucose['average'] & bmi['average'] & age['average'], diabetes_risk['average']),
        ctrl.Rule(glucose['good'] & bmi['good'] & age['good'], diabetes_risk['good']),
        ctrl.Rule(blood_pressure['poor'] & glucose['poor'], diabetes_risk['poor']),
        ctrl.Rule(blood_pressure['average'] & bmi['average'] & age['average'], diabetes_risk['average']),
        ctrl.Rule(glucose['average'] & blood_pressure['good'], diabetes_risk['average'])
    ]

    diabetes_ctrl = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(diabetes_ctrl)




# --- Fuzzy Logic for Thyroid ---
def create_fuzzy_thyroid(df):
    tsh = ctrl.Antecedent(np.arange(df['TSH'].min(), df['TSH'].max() + 1, 0.1), 'tsh')
    t3 = ctrl.Antecedent(np.arange(df['T3'].min(), df['T3'].max() + 1, 1), 't3')
    t4 = ctrl.Antecedent(np.arange(df['T4'].min(), df['T4'].max() + 1, 0.1), 't4')
    thyroid_risk = ctrl.Consequent(np.arange(0, 101, 1), 'thyroid_risk')

    tsh.automf(3)
    t3.automf(3)
    t4.automf(3)
    thyroid_risk.automf(3)

    rules = [
```

Medical_Diagnosis_System.py > ...

```python
def create_fuzzy_thyroid(df):
    rules = [
        ctrl.Rule(tsh['good'] & t3['good'] & t4['good'], thyroid_risk['poor']),
        ctrl.Rule(tsh['average'] | t3['average'] | t4['average'], thyroid_risk['average']),
        ctrl.Rule(tsh['poor'] | t3['poor'] | t4['poor'], thyroid_risk['good']),
    ]

    system = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(system)


# --- Fuzzy Logic for PCOD ---
def create_fuzzy_pcod(df):
    bmi = ctrl.Antecedent(np.arange(df['BMI'].min(), df['BMI'].max() + 1, 1), 'bmi')
    insulin = ctrl.Antecedent(np.arange(df['Insulin_Level'].min(), df['Insulin_Level'].max() + 1, 1), 'insulin')
    lh = ctrl.Antecedent(np.arange(df['LH'].min(), df['LH'].max() + 1, 1), 'lh')
    pcod_risk = ctrl.Consequent(np.arange(0, 101, 1), 'pcod_risk')

    bmi.automf(3)
    insulin.automf(3)
    lh.automf(3)
    pcod_risk.automf(3)

    rules = [
        ctrl.Rule(bmi['good'] & insulin['good'] & lh['good'], pcod_risk['poor']),
        ctrl.Rule(bmi['average'] | insulin['average'] | lh['average'], pcod_risk['average']),
        ctrl.Rule(bmi['poor'] | insulin['poor'] | lh['poor'], pcod_risk['good'])
    ]

    system = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(system)

# --- Fuzzy Logic for Anxiety ---

def create_fuzzy_anxiety(df):
    sleep = ctrl.Antecedent(np.arange(df['SleepHours'].min(), df['SleepHours'].max()+0.1, 0.1), 'sleep')
    heart_rate = ctrl.Antecedent(np.arange(df['HeartRate'].min(), df['HeartRate'].max()+1, 1), 'heart_rate')
```

Medical_Diagnosis_System.py > ...

```python
def create_fuzzy_anxiety(df):
    sleep = ctrl.Antecedent(np.arange(df['SleepHours'].min(), df['SleepHours'].max()+0.1, 0.1), 'sleep')
    heart_rate = ctrl.Antecedent(np.arange(df['HeartRate'].min(), df['HeartRate'].max()+1, 1), 'heart_rate')
    fatigue = ctrl.Antecedent(np.arange(0, 2, 1), 'fatigue')
    irritability = ctrl.Antecedent(np.arange(0, 2, 1), 'irritability')
    restlessness = ctrl.Antecedent(np.arange(0, 2, 1), 'restlessness')
    score = ctrl.Antecedent(np.arange(df['ScoreGAD7'].min(), df['ScoreGAD7'].max()+1, 1), 'score')
    anxiety_risk = ctrl.Consequent(np.arange(0, 101, 1), 'anxiety_risk')

    sleep.automf(3)
    heart_rate.automf(3)
    fatigue.automf(3)
    irritability.automf(3)
    restlessness.automf(3)
    score.automf(3)
    anxiety_risk.automf(3)

    rules = [
        ctrl.Rule(sleep['poor'] | score['good'] | heart_rate['good'], anxiety_risk['good']),
        ctrl.Rule(score['average'] | irritability['average'] | fatigue['average'], anxiety_risk['average']),
        ctrl.Rule(sleep['good'] & fatigue['poor'] & irritability['poor'], anxiety_risk['poor']),
        ctrl.Rule(restlessness['poor'] | fatigue['good'], anxiety_risk['good']),  # ✅ Added rule with restlessness
    ]


    system = ctrl.ControlSystem(rules)
    return ctrl.ControlSystemSimulation(system)


# --- Streamlit UI ---
def main():
    st.title("🩺 Medical Diagnosis System")

    with st.sidebar:
        selected = option_menu("Main Menu", ["Home","Heart Disease", "Diabetes", "Thyroid", "PCOD" , "Anxiety" , "Dataset","Doctors"],
                               icons=['house', 'heart', 'activity', 'thermometer', 'droplet', 'arrow-down-circle', 'database',"person"],
                               menu_icon="cast", default_index=0)
```

Medical_Diagnosis_System.py > ...

```
168  def main():
210      st.subheader("Heart Disease")
211      age = st.number_input("Age", int(df['age'].min()), int(df['age'].max()), int(df['age'].mean()))
212      cholesterol = st.number_input("Cholesterol", int(df['Cholesterol'].min()), int(df['Cholesterol'].max()), int(df['Cholesterol'].mean()))
213      thalach = st.number_input("Max Heart Rate", int(df['thalach'].min()), int(df['thalach'].max()), int(df['thalach'].mean()))
214      chest_pain = st.slider("Chest Pain Type (0: Typical, 1: Atypical, 2: Non-anginal, 3: Asymptomatic)", 0, 3, 1)
215      resting_bp = st.number_input("Resting Blood Pressure", int(df['trestbps'].min()), int(df['trestbps'].max()), int(df['trestbps'].mean()))
216
217      col1, col2, col3 = st.columns([1, 1, 1])
218      with col1:
219          predict = st.button("Predict Heart Status", key="predict_heart")
220      with col3:
221          show_doctors = st.button("Show Recommended Doctors", key="show_heart_doctors")
222
223      if predict:
224          sim.input['age'] = age
225          sim.input['cholesterol'] = cholesterol
226          sim.input['thalach'] = thalach #Max Heart Rate
227          sim.input['chest_pain'] = chest_pain
228          sim.input['resting_bp'] = resting_bp
229
230          sim.compute()
231          risk = sim.output['heart_risk']
232          diagnosis = "Yes" if risk >= 50 else "No"
233          st.success(f"Predicted Heart Disease Risk: {diagnosis}")
234
235          warnings = []
236          if cholesterol > 240:
237              warnings.append("⚠️ High Cholesterol – Increased risk of heart disease.")
238          if cholesterol < 125:
239              warnings.append("⚠️ Low Cholesterol – May indicate underlying health issues or malnutrition.")
240
241          if resting_bp > 140:
242              warnings.append("⚠️ High Resting Blood Pressure – May indicate hypertension.")
243          if resting_bp < 100:
244              warnings.append("⚠️ Low Resting Blood Pressure – May lead to dizziness or fainting.")
245
```

Medical_Diagnosis_System.py > ...

```
168  def main():
176      if selected == "Home":
177          st.markdown("###  Welcome to your personalized health assistant.")
178          st.write("This intelligent system is designed to help you assess your health and detect early signs of common medical conditions.")
179          st.write("Using smart algorithms and real clinical data, it can predict the likelihood of:")
180          st.write("Heart Disease")
181          st.write("Diabetes")
182          st.write("Thyroid Disorders")
183          st.write("Polycystic Ovarian Disease (PCOD)")
184          st.write("Take a step towards better health by navigating through the system.")
185          st.write("Enter your health parameters and receive quick, reliable insights – all in one place")
186
187      elif selected == "Doctors":
188          st.subheader("Doctors' Dataset")
189          try:
190              doctor_data = pd.read_csv("indian_doctors_dataset.csv")
191              st.dataframe(doctor_data)
192          except FileNotFoundError:
193              st.warning("Doctor dataset not found. Please upload or check the file.")
194      elif selected == "Dataset":
195          st.subheader(" View Dataset")
196          dataset_choice = st.selectbox("Select which dataset to display", ["Heart", "Diabetes", "Thyroid", "PCOD","Anxiety"])
197
198          df = load_dataset(dataset_choice.lower())
199          if df is not None:
200              st.dataframe(df)
201
202  # Heart disease logic
203
204      elif selected == "Heart Disease":
205          df = load_dataset("heart")
206          if df is None:
207              return
208          sim = create_fuzzy_heart(df)
209
210          st.subheader("Heart Disease")
211          age = st.number_input("Age", int(df['age'].min()), int(df['age'].max()), int(df['age'].mean()))
```

```python
Medical_Diagnosis_System.py > ...
def main():
            if thalach < 100:
                warnings.append("⚠️ Low Max Heart Rate – May be a sign of poor cardiovascular fitness.")
            if age > 60:
                warnings.append("⚠️ Age over 60 – Age is a major risk factor for heart disease.")
            if chest_pain == 3:
                warnings.append("⚠️ Asymptomatic Chest Pain Type – Often linked with higher heart disease risk.")

            if warnings:
                warning_html = '<div style="background-color:#fbeaea;padding:10px;border-radius:8px;">' + \
                               '<br>'.join(f'<span style="color:#d00000;">{w}</span>' for w in warnings) + \
                               '</div>'
                st.markdown(warning_html, unsafe_allow_html=True)
        elif show_doctors:
            st.subheader("Recommended Doctors for Heart Disease")
            try:
                doctor_data = pd.read_csv("indian_doctors_dataset.csv")
                h_doctors = doctor_data[
                    doctor_data['Specialist'].str.lower().str.contains("heart", na=False)
                ]
                if not h_doctors.empty:
                    grouped = h_doctors.groupby('Clinic Address')
                    count = 0
                    for name, group in grouped:
                        st.markdown(f"##### 📍 Location: {name}")
                        top5 = group.head(3)[['Doctor Name', 'Specialist', 'Phone Number', 'Email']]
                        st.dataframe(top5, use_container_width=True)
                        count += 1
                        if count >= 3:
                            break
                else:
                    st.info("No doctors found for Heart Disease.")
            except Exception as e:
                st.error(f"Error loading doctors data: {e}")

# Diabetes logic
```

```python
Medical_Diagnosis_System.py > ...
def main():
    elif selected == "Diabetes":
        df = load_dataset("diabetes")
        if df is None:
            return
        sim = create_fuzzy_diabetes(df)
        st.subheader("Diabetes ")
        #glucose = st.number_input("Glucose", int(df['Glucose'].min()), int(df['Glucose'].max()), int(df['Glucose'].mean()))
        glucose = st.number_input("Glucose", int(df['Glucose'].min()), int(df['Glucose'].max()), int(df['Glucose'].mean()), key="glucose_input")

        bmi = st.number_input("BMI", float(df['BMI'].min()), float(df['BMI'].max()), float(df['BMI'].mean()))
        age = st.number_input("Age", int(df['Age'].min()), int(df['Age'].max()), int(df['Age'].mean()))
        bp = st.number_input("Blood Pressure", int(df['BloodPressure'].min()), int(df['BloodPressure'].max()), int(df['BloodPressure'].mean()))

        st.markdown("### 📄 Upload Blood Test Report (CSV or Excel)")
        uploaded_file = st.file_uploader("Choose a file", type=["csv", "xlsx"])

        if uploaded_file:
            try:
                if uploaded_file.name.endswith('.csv'):
                    user_data = pd.read_csv(uploaded_file)
                else:
                    user_data = pd.read_excel(uploaded_file)

                st.success("✅ Report uploaded successfully!")
                st.dataframe(user_data)

                # Assuming the report has correct column names
                glucose = user_data['Glucose'][0]
                bmi = user_data['BMI'][0]
                age = user_data['Age'][0]
                bp = user_data['BloodPressure'][0]

            except Exception as e:
                st.error(f"⚠️ Error reading file: {e}")
                return
```

Medical_Diagnosis_System.py > ...

```
168  def main():
317          col1, col2, col3 = st.columns([1, 1, 1])
318          with col1:
319              predict = st.button("Predict Diabetes Status", key="predict_diabetes")
320          with col3:
321              show_doctors = st.button("Show Recommended Doctors", key="show_diabetes_doctors")
322
323          if predict:
324              sim.input['glucose'] = glucose
325              sim.input['bmi'] = bmi
326              sim.input['age'] = age
327              sim.input['blood_pressure'] = bp
328              sim.compute()
329              risk = sim.output['diabetes_risk']
330              diagnosis = "Yes" if risk >= 50 else "No"
331              st.success(f"Predicted Diabetes Status: {diagnosis}")
332
333              # Collect warnings
334              warnings = []
335              if glucose > 140:
336                  warnings.append("⚠️ High glucose")
337              if bmi > 30:
338                  warnings.append("⚠️ High BMI")
339              if bp > 80:
340                  warnings.append("⚠️ High Blood pressure")
341              if glucose < 70:
342                  warnings.append("⚠️ Low glucose – Your sugar level is too low. You might feel weak, shaky, or tired.")
343              if bmi < 18.5:
344                  warnings.append("⚠️ Low BMI – Your body weight is quite low. You may need to eat more to stay healthy.")
345              if bp < 60:
346                  warnings.append("⚠️ Low blood pressure – Your blood pressure is low. This can make you feel dizzy or faint.")
347              # Display warnings in one line using markdown
348              if warnings:
349                  warning_html = '<div style="background-color:#fbeaea;padding:10px;border-radius:8px;">' + \
350                                 ' <br> '.join(f'<span style="color:#d00000;">{w}</span>' for w in warnings) + \
351                                 '</div>'
```

Medical_Diagnosis_System.py > ...

```
168  def main():
351                                  '</div>'
352              st.markdown(warning_html, unsafe_allow_html=True)
353
354          elif show_doctors:
355              st.subheader("Recommended Doctors for Diabetes")
356              try:
357                  doctor_data = pd.read_csv("indian_doctors_dataset.csv")
358                  diabetes_doctors = doctor_data[
359                      doctor_data['Specialist'].str.lower().str.contains("diabet", na=False)
360                  ]
361                  if not diabetes_doctors.empty:
362                      grouped = diabetes_doctors.groupby('Clinic Address')
363                      count = 0
364                      for name, group in grouped:
365                          st.markdown(f"##### 📍 Location: {name}")
366                          top5 = group.head(3)[['Doctor Name', 'Specialist', 'Phone Number', 'Email']]
367                          st.dataframe(top5, use_container_width=True)
368                          count += 1
369                          if count >= 3:
370                              break
371                  else:
372                      st.info("No doctors found for Diabetes.")
373              except Exception as e:
374                  st.error(f"Error loading doctors data: {e}")
375
376  # Thyroid logic
377
378      elif selected == "Thyroid":
379          df = load_dataset("thyroid")
380          if df is None:
381              return
382          sim = create_fuzzy_thyroid(df)
383          st.subheader("Thyroid")
384          tsh = st.number_input("TSH (mIU/L)", float(df['TSH'].min()), float(df['TSH'].max()), float(df['TSH'].mean()))
385          t3 = st.number_input("T3 (ng/dL)", float(df['T3'].min()), float(df['T3'].max()), float(df['T3'].mean()))
```

```python
Medical_Diagnosis_System.py > ...
def main():
        t3 = st.number_input("T3 (ng/dL)", float(df['T3'].min()), float(df['T3'].max()), float(df['T3'].mean()))
        t4 = st.number_input("T4 (µg/dL)", float(df['T4'].min()), float(df['T4'].max()), float(df['T4'].mean()))

        st.markdown("### 📄 Upload blood Test Report (CSV or Excel)")
        uploaded_file = st.file_uploader("Choose a file", type=["csv", "xlsx"])


        if uploaded_file:
            try:
                if uploaded_file.name.endswith('.csv'):
                    user_data = pd.read_csv(uploaded_file)
                else:
                    user_data = pd.read_excel(uploaded_file)

                st.success("✅ Report uploaded successfully!")
                st.dataframe(user_data)

                # Assuming the report has correct column names
                tsh = user_data['TSH (mIU/L)'][0]
                t3 = user_data['T3 (ng/dL)'][0]
                t4 = user_data['T4 (µg/dL)'][0]

            except Exception as e:
                st.error(f"⚠️ Error reading file: {e}")
                return
        col1, col2, col3 = st.columns([1, 1, 1])
        with col1:
            predict = st.button("Predict Thyroid Status", key="predict_thyroid")
        with col3:
            show_doctors = st.button("Show Recommended Doctors", key="show_thyroid_doctors")

        if predict:
            sim.input['tsh'] = tsh
            sim.input['t3'] = t3
            sim.input['t4'] = t4
            sim.compute()
```

```python
Medical_Diagnosis_System.py > main
def main():
            risk = sim.output['thyroid_risk']
            diagnosis = "Yes" if risk >= 50 else "No"
            st.success(f"Predicted Thyroid Status: {diagnosis}")

            warnings = []
            if tsh > 4.0:
                warnings.append("⚠️ High TSH – Could indicate hypothyroidism.")
            if tsh < 0.4 :
                warnings.append("⚠️ Low TSH – Could be a sign of hyperthyroidism.")
            if t3 < 70:
                warnings.append("⚠️ Low T3 – Often seen in hypothyroidism.")
            if t3 > 200:
                warnings.append("⚠️ High T3 – Might indicate an overactive thyroid.")
            if t4 < 5.0:
                warnings.append("⚠️ Low T4 – May signal thyroid hormone deficiency.")

            if warnings:
                warning_html = '<div style="background-color:#fbeaea;padding:10px;border-radius:8px;">' + \
                               ' <br> '.join(f'<span style="color:#d00000;">{w}</span>' for w in warnings) + \
                               '</div>'
                st.markdown(warning_html, unsafe_allow_html=True)
        elif show_doctors:
            st.subheader("Recommended Doctors for Thyroid")
            try:
                doctor_data = pd.read_csv("indian_doctors_dataset.csv")
                t_doctors = doctor_data[
                    doctor_data['Specialist'].str.lower().str.contains("thyroid", na=False)
                ]
                if not t_doctors.empty:
                    grouped = t_doctors.groupby('Clinic Address')
                    count = 0
                    for name, group in grouped:
                        st.markdown(f"##### 📍 Location: {name}")
                        top5 = group.head(3)[['Doctor Name', 'Specialist', 'Phone Number', 'Email']]
                        st.dataframe(top5, use_container_width=True)
                        count += 1
```

Medical_Diagnosis_System.py > main

```
168  def main():
456                          count += 1
457                          if count >= 3:
458                              break
459                  else:
460                      st.info("No doctors found for Thyroid.")
461              except Exception as e:
462                  st.error(f"Error loading doctors data: {e}")
463
464  # PCOD Logic
465
466      elif selected == "PCOD":
467          df = load_dataset("pcod")
468          if df is None:
469              return
470          sim = create_fuzzy_pcod(df)
471          st.subheader("PCOD")
472          bmi = st.number_input("BMI", float(df['BMI'].min()), float(df['BMI'].max()), float(df['BMI'].mean()))
473          insulin = st.number_input("Insulin Level", float(df['Insulin_Level'].min()), float(df['Insulin_Level'].max()), float(df['Insulin_Level'].mean()))
474          lh = st.number_input("LH", float(df['LH'].min()), float(df['LH'].max()), float(df['LH'].mean()))
475
476          st.markdown("### 📄 Upload blood Test Report (CSV or Excel)")
477          uploaded_file = st.file_uploader("Choose a file", type=["csv", "xlsx"])
478
479          if uploaded_file:
480              try:
481                  if uploaded_file.name.endswith('.csv'):
482                      user_data = pd.read_csv(uploaded_file)
483                  else:
484                      user_data = pd.read_excel(uploaded_file)
485
486                  st.success("✅ Report uploaded successfully!")
487                  st.dataframe(user_data)
488
489                  # Assuming the report has correct column names
490                  bmi = user_data['BMI'][0]
491                  insulin = user_data["Insulin_Level"][0]
```

Medical_Diagnosis_System.py > main

```
168  def main():
492                  lh = user_data['LH'][0]
493
494              except Exception as e:
495                  st.error(f"⚠️ Error reading file: {e}")
496                  return
497
498          col1, col2, col3 = st.columns([1, 1, 1])
499          with col1:
500              predict = st.button("Predict PCOD Status", key="predict_pcod")
501          with col3:
502              show_doctors = st.button("Show Recommended Doctors", key="show_pcod_doctors")
503
504          if predict:
505              sim.input['bmi'] = bmi
506              sim.input['insulin'] = insulin
507              sim.input['lh'] = lh
508              sim.compute()
509              risk = sim.output['pcod_risk']
510              diagnosis = "Yes" if risk >= 50 else "No"
511              st.success(f"Predicted PCOD Status: {diagnosis}")
512
513              warnings = []
514              if bmi > 25:
515                  warnings.append("⚠️ High BMI – Obesity is a major risk factor for PCOD.")
516              if bmi < 18.5:
517                  warnings.append("⚠️ Low BMI – Consider monitoring nutritional health.")
518              if insulin > 15:
519                  warnings.append("⚠️ Elevated Insulin – May indicate insulin resistance.")
520              if insulin < 10:
521                  warnings.append("⚠️ Very Low Insulin – May need medical attention.")
522              if lh > 9:
523                  warnings.append("⚠️ High LH – Can contribute to irregular ovulation.")
524              if warnings:
525                  warning_html = '<div style="background-color:#fbeaea;padding:10px;border-radius:8px;">' + \
526                                 ' <br> '.join(f'<span style="color:#d00000;">{w}</span>' for w in warnings) + \
527                                 '</div>'
```

```python
Medical_Diagnosis_System.py > main
def main():
                st.markdown(warning_html, unsafe_allow_html=True)
        elif show_doctors:
            st.subheader("Recommended Doctors for PCOD")
            try:
                doctor_data = pd.read_csv("indian_doctors_dataset.csv")
                p_doctors = doctor_data[
                    doctor_data['Specialist'].str.lower().str.contains("pcod", na=False)
                ]
                if not p_doctors.empty:
                    grouped = p_doctors.groupby('Clinic Address')
                    count = 0
                    for name, group in grouped:
                        st.markdown(f"##### 📍 Location: {name}")
                        top5 = group.head(3)[['Doctor Name', 'Specialist', 'Phone Number', 'Email']]
                        st.dataframe(top5, use_container_width=True)
                        count += 1
                        if count >= 3:
                            break
                else:
                    st.info("No doctors found for PCOD.")
            except Exception as e:
                st.error(f"Error loading doctors data: {e}")

# Anxiety logic

    elif selected == "Anxiety":
        df = load_dataset("anxiety")
        if df is None:
            return
        sim = create_fuzzy_anxiety(df)
        st.subheader("Anxiety")
        sleep = st.number_input("Sleep Hours", float(df['SleepHours'].min()), float(df['SleepHours'].max()), float(df['SleepHours'].mean()))
        heart_rate = st.number_input("Heart Rate", int(df['HeartRate'].min()), int(df['HeartRate'].max()), int(df['HeartRate'].mean()))

        fatigue_str = st.selectbox("Fatigue", ["No", "Yes"])
        fatigue = 1 if fatigue_str == "Yes" else 0
```

```python
Medical_Diagnosis_System.py > main
def main():
        irritability_str = st.selectbox("Irritability", ["No", "Yes"])
        irritability = 1 if irritability_str == "Yes" else 0

        restlessness_str = st.selectbox("Restlessness", ["No", "Yes"])
        restlessness = 1 if restlessness_str == "Yes" else 0

        score = st.slider("GAD-7 Score", int(df['ScoreGAD7'].min()), int(df['ScoreGAD7'].max()), int(df['ScoreGAD7'].mean()))

        col1, col2, col3 = st.columns([1, 1, 1])
        with col1:
            predict = st.button("Predict Anxiety Status", key="predict_anxiety")
        with col3:
            show_doctors = st.button("Show Recommended Doctors", key="show_anxiety_doctors")

        if predict:
            sim.input['sleep'] = sleep
            sim.input['heart_rate'] = heart_rate
            sim.input['fatigue'] = fatigue
            sim.input['irritability'] = irritability
            sim.input['restlessness'] = restlessness
            sim.input['score'] = score
            sim.compute()

            risk = sim.output['anxiety_risk']
            diagnosis = "Yes" if risk >= 50 else "No"
            st.success(f"Predicted Anxiety Status: {diagnosis}")

            warnings = []

            if restlessness == 1:  # Assuming 1 = Yes, 0 = No
                warnings.append("⚠️ High Restlessness – Major contributor to anxiety.")
            if sleep < 6:
                warnings.append("⚠️ Poor Sleep – Can worsen anxiety.")
            if fatigue == 1:
                warnings.append("⚠️ Fatigue – Often linked with anxiety symptoms.")
            if irritability == 1:
```

Medical_Diagnosis_System.py > main

```
600             if irritability == 1:
601                 warnings.append("⚠️ Irritability – A strong emotional indicator.")
602             if sleep > 9:
603                 warnings.append("⚠️ Oversleeping – May signal underlying issues.")
604             if risk < 20:
605                 warnings.append("⚠️ Low Stress – That's good, keep it balanced!")
606 
607             if warnings:
608                 warning_html = '<div style="background-color:#fbeaea;padding:10px;border-radius:8px;">' + \
609                                ' <br> '.join(f'<span style="color:#d00000;">{w}</span>' for w in warnings) + \
610                                '</div>'
611                 st.markdown(warning_html, unsafe_allow_html=True)
612         elif show_doctors:
613             st.subheader("Recommended Doctors for Anxiety")
614             try:
615                 doctor_data = pd.read_csv("indian_doctors_dataset.csv")
616                 a_doctors = doctor_data[
617                     doctor_data['Specialist'].str.lower().str.contains("anxiety", na=False)
618                 ]
619                 if not a_doctors.empty:
620                     grouped = a_doctors.groupby('Clinic Address')
621                     count = 0
622                     for name, group in grouped:
623                         st.markdown(f"##### 📍 Location: {name}")
624                         top5 = group.head(3)[['Doctor Name', 'Specialist', 'Phone Number', 'Email']]
625                         st.dataframe(top5, use_container_width=True)
626                         count += 1
627                         if count >= 3:
628                             break
629                 else:
630                     st.info("No doctors found for Anxiety.")
631             except Exception as e:
632                 st.error(f"Error loading doctors data: {e}")
633 
634 if __name__ == "__main__":
635     main()
636 
```