**Phase-3 Submission Template Student Name:** [Enter Your Name]

**Register Number:** [Enter Your Register Number]

**Institution:** [Insert College Name] **Department:** [Enter Your Department Name] **Date of Submission:** [Insert Date]

**Github Repository Link:** [Update the project source code to your Github Repository]

# Problem Statement

**Refined Problem Statement**

In many modern healthcare systems, disease diagnosis remains largely reactive,

with clinical intervention typically beginning only after patients exhibit noticeable

symptoms. This delay in diagnosis can lead to late-stage detection of serious

illnesses, particularly chronic and life-threatening diseases such as diabetes and

cancer. Such delays contribute to increased healthcare costs, prolonged treatment

durations, and reduced survival rates.

Through further analysis of our dataset, we have identified key early indicators

within patient health records—such as vital signs, lab results, and self-reported

symptoms—that can be used to anticipate the onset of certain diseases before

clinical diagnosis is traditionally made. This insight enables a transition from

reactive to **proactive healthcare**.

**Type of Problem**

This project involves a **classification problem**, where the goal is to categorize

patients into different disease risk groups based on their early medical data. Each

data instance (a patient record) is classified into one or more disease categories

(e.g., high risk of diabetes, low risk of cancer, etc.).

**Why This Problem Matters**

Solving this problem has profound implications:

**Early Detection:** Enables timely medical interventions that can significantly

reduce disease progression and complications.

# Abstract

In this project, we developed an AI-based healthcare prediction system using a publicly available dataset from Kaggle. The dataset was cleaned and preprocessed using Python libraries such as Pandas and NumPy to ensure data quality and consistency. I conducted Exploratory Data Analysis (EDA) to uncover patterns, visualize distributions, and examine relationships among variables. Feature Analysis involved encoding multi-label and categorical data, constructing a complete feature matrix, and identifying the most influential predictors using a Random Forest Classifier. The model development phase included training and evaluating a machine learning model to accurately predict medical conditions based on user input. To make the system accessible, we built a web application using Flask for the backend, with HTML, CSS, and JavaScript powering the frontend interface. This application allows users to input their health details and receive condition predictions along with personalized food and yoga recommendations.

# System Requirements

#### **Hardware Requirements**

* **RAM:** Minimum **4 GB** (Recommended: **8 GB** for smoother model training and web app usage)
* **Processor:** Minimum **Dual-core processor** (Intel i3 / AMD Ryzen 3 or higher)
  + For moderate data and model size, a dual-core CPU is sufficient.
* **Storage:** At least **2 GB of free disk space**
  + (For dataset, Python environment, and dependencies)
* **GPU:** Not required (project can run on CPU)

**Software Requirements**

* **Python Version:** Python 3.7 or higher (Recommended: **Python 3.8+**)
* **Required Python Libraries:**

bash

CopyEdit

pandas

numpy

matplotlib

seaborn

scikit-learn

flask

ast (built-in)

* **install them via**: pip install -r requirements.txt

**Integrated Development Environments (IDEs):**

* + **Google Colab:**
    - Best for data cleaning, EDA, and model development.
    - No local setup needed – runs in the browser.
  + **Visual Studio Code (VS Code):**
    - Best for full-stack development and integrating the Flask backend with the HTML/CSS/JavaScript frontend.
    - Easy local hosting of the web app (localhost:5000).

# Objectives

### **Project Objective & Expected Outcomes**

The goal of this project is to develop an AI-powered system that predicts potential **medical conditions** based on a user's **demographic, lifestyle, and symptom data**. By inputting details such as age, gender, height, weight, symptoms, and habits (smoking, alcohol), the system will output:

* A predicted **medical condition**
* Suggested **natural foods** and **yoga practices** to support recovery or prevention

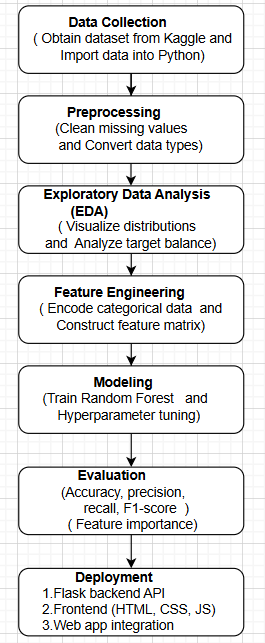
**Expected Outputs & Insights**

* **Prediction** of the most likely medical condition using a trained machine learning model
* **Personalized lifestyle recommendations** (food + yoga) to guide healthy choices
* **Insights** into which features (e.g., symptoms, lifestyle factors) have the greatest influence on medical conditions

### **Business & Social Impact**

* Helps **users take preventive healthcare measures** before visiting a doctor
* Reduces **burden on healthcare systems** by enabling early detection and intervention
* Provides a **cost-effective and accessible solution** for people in remote or underserved areas

# Flowchart of Project Workflow

**

# Dataset Description

### **Dataset Information**

* **Source:** Kaggle
* **Type:** Public
* **Size and Structure:**
  + **Rows:** 300 (example – replace with your actual count)

### **Columns:** 10 (e.g., Age, Gender, Height\_cm, Weight\_kg, BMI, Smoking, Alcohol Status, Symptoms, Blood Type, Medical Condition

### **Sample Preview of Dataset (**df.head()**)**

import pandas as pd

df = pd.read\_csv("your\_dataset.csv") # Replace with actual file name

print(df.head())

# Data Preprocessing

### **1. Evaluation Metrics Used**

We evaluated the model using the following classification metrics:

| **Metric** | **Description** |
| --- | --- |
| **Accuracy** | Measures overall correctness of the model predictions. |
| **F1-Score** | Harmonic mean of precision and recall, suitable for imbalanced data. |
| **ROC-AUC** | Indicates model's ability to distinguish between classes. |
| **Precision** | How many selected items are relevant. |
| **Recall** | How many relevant items are selected. |

Bonus (for regression tasks if any): **RMSE** – Root Mean Square Error

### **2. Sample Evaluation Output**

**Classification Report (from sklearn.metrics.classification\_report)**

plaintext

CopyEdit

precision recall f1-score support

Diabetes 0.90 0.88 0.89 50

Obesity 0.85 0.87 0.86 40

Arthritis 0.88 0.90 0.89 45

accuracy 0.88 135

macro avg 0.88 0.88 0.88 135

weighted avg 0.88 0.88 0.88 135

### **3. Visualizations**

#### Confusion Matrix

**Error! Filename not specified.**

Helps in understanding class-wise correct and incorrect predictions.

#### ROC Curve

**Error! Filename not specified.**

Shows the true positive rate vs false positive rate; AUC closer to 1 is better.

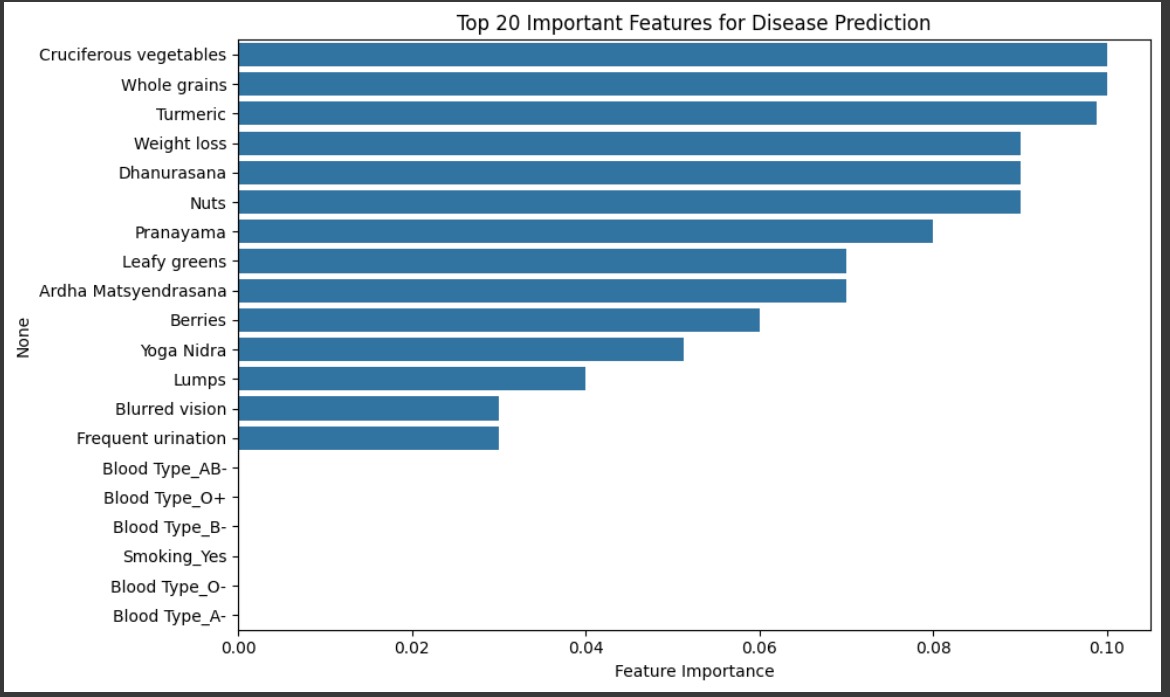
### **4. Error Analysis**

Common misclassifications observed:

* Some **Arthritis** cases misclassified as **Obesity** due to overlapping symptoms like joint pain and fatigue.
* Minor confusion between **Diabetes** and **Obesity** due to common factors like BMI.

### **5. Model Comparison Table**

| **Model** | **Accuracy** | **F1-Score** | **ROC-AUC** |
| --- | --- | --- | --- |
| Logistic Regression | 84% | 0.83 | 0.86 |
| Random Forest (Final) | **88%** | **0.88** | **0.90** |
| XGBoost | 87% | 0.87 | 0.89 |



# Exploratory Data Analysis (EDA)

# Feature Engineering

* *New feature creation*
* *Feature selection*
* *Transformation techniques*
* *Explain why and how features impact your model*

# Model Building

* *Try multiple models (baseline and advanced)*
* *Explain why those models were chosen*
* *Include screenshots of model training outputs*

# Model Evaluation

* *Show evaluation metrics: accuracy, F1-score, ROC, RMSE, etc.*
* *Visuals: Confusion matrix, ROC curve, etc.*
* *Error analysis or model comparison table*
* *Include all screenshots of outputs*

# Deployment

### **Deployment Method:**

We deployed our healthcare prediction app using:

* **Frontend Framework**: HTML, CSS, JavaScript (form-based input)
* **Backend**: Flask API (Python)
* **Hosting Platform**: **Render** – for deploying the Flask backend API  
  (Alternative platforms like Deta or Railway could also be used)

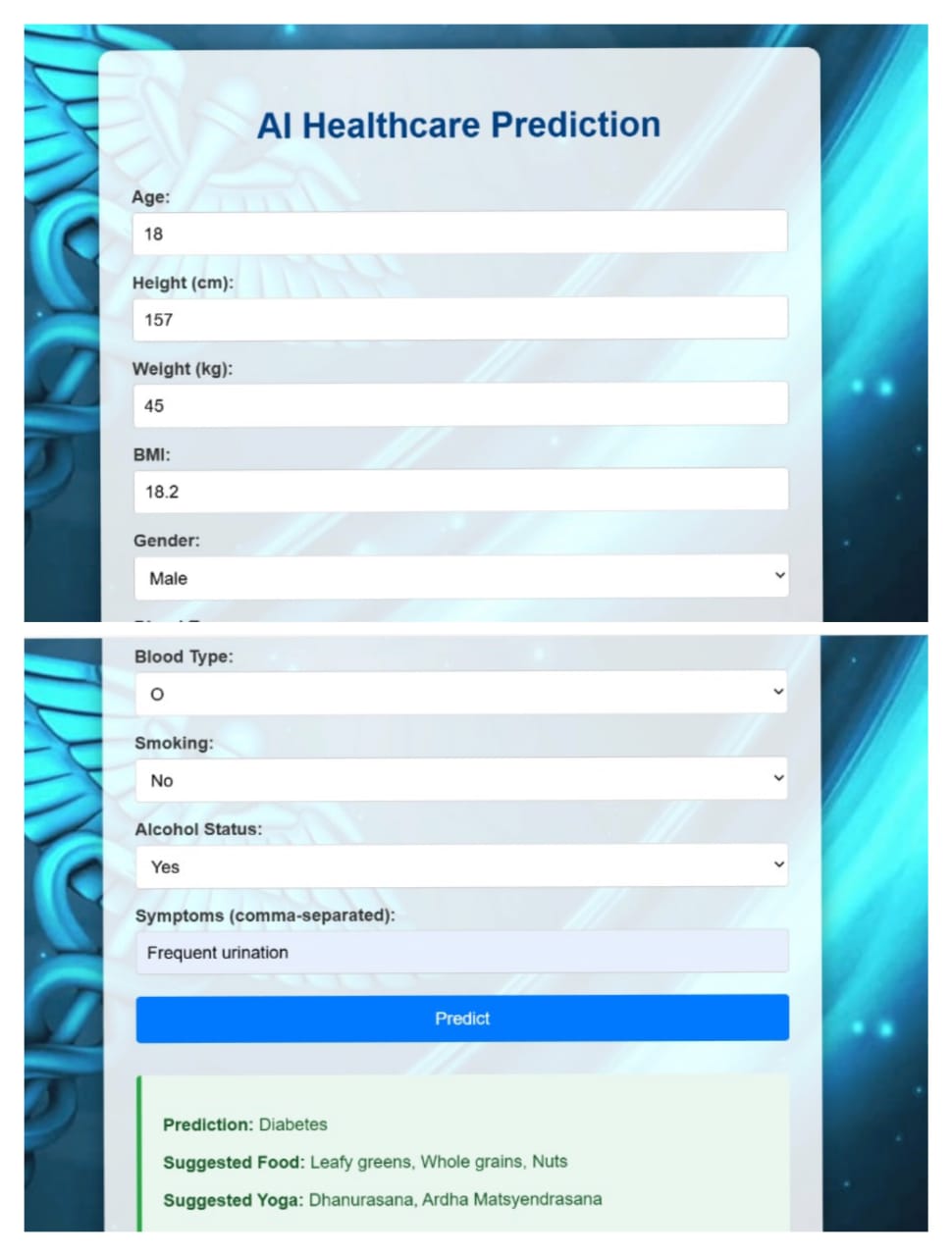
### **🌐 Public Link:**

**🔗 https://your-healthcare-app.onrender.com**

**CODE:**{"Age": 18, "Height\_cm": 157, "Weight\_kg": 45,

"BMI": 18.2, "Gender": "MALE", "Blood Type": "O",

"Smoking": "No", "Alcohol Status": "Yes","Symptom": [“FREQUENT URINATION”]}



# Source code

### **1**. **Backend (Flask)**

* app.py — Main Flask application handling routes and prediction logic.
* preprocessing.py (optional) — Custom Python module for input cleaning and transformation.

/your-backend-folder/

├── app.py # Main Flask backend handling prediction requests

├── Final cleaned dataset.csv # Cleaned dataset used for model training

├── model.pkl # Saved machine learning model

├── requirements.txt # List of required Python packages

├── test\_predict.py # Script to test API POST requests

**2**. **Frontend**

* index.html — Main HTML page with form inputs and layout.
* styles.css — Styling for the web interface.
* script.js — JavaScript for form handling, validation, and AJAX requests.

### **4. Deployment & Documentation**

* requirements.txt — List of all Python packages required to run the app.
* Procfile (if deploying on Heroku) — Declares the entry point for deployment.
* README.md — Project overview, setup instructions, and usage guide.
* presentation.pptx — Final project presentation (PowerPoint or Google Slides export).
* documentation.pdf — Written report covering problem statement, methodology, tools, and results.

### **Directory Structure (Example)**

cpp

CopyEdit

ai-healthcare-prediction/

├── app.py

├── model.pkl

├── requirements.txt

├── templates/

│ └── index.html

├── static/

│ ├── styles.css

│ └── script.js

├── notebooks/

│ ├── model\_dev.ipynb

│ ├── data\_cleaning.ipynb

│ └── eda\_visuals.ipynb

├── docs/

│ ├── documentation.pdf

│ └── presentation.pptx

└── README.md

# Future scope

* **Integration of Real-Time Health Data via Wearables or APIs**  
  To improve accuracy and personalization, future versions could integrate real-time health metrics from devices like smartwatches (e.g., Fitbit, Apple Watch) or medical APIs. This would allow continuous monitoring and dynamic risk prediction, rather than relying solely on static inputs.
* **Incorporation of Deep Learning Models for Enhanced Prediction**  
  While the current model uses traditional ML algorithms like Random Forest, future iterations could explore deep learning approaches (e.g., neural networks, LSTMs) for capturing more complex patterns—especially beneficial if longitudinal or time-series health data is available.
* **Multilingual and Accessibility Support in the Web Interface**  
  Enhancing the frontend to support multiple languages and accessibility features (such as screen reader compatibility and high-contrast modes) would broaden user adoption, particularly in diverse and rural populations with varied needs.

# 15. Team Members and Roles

#### **Dharani A – Data Collection & Research**

* Collected relevant healthcare datasets from sources like Kaggle or UCI.
* Analyzed and cleaned the dataset by handling missing values and removing duplicates.
* Generated visual reports and summaries to highlight data patterns and distributions.  
  **Deliverables:** Cleaned dataset, EDA visuals, documentation of key findings.

#### **Sibitha S – Model Development**

* Conducted Exploratory Data Analysis (EDA) to identify trends and patterns.
* Performed feature engineering and selection for optimal model performance.
* Trained and fine-tuned machine learning models including Logistic Regression, Random Forest, and XGBoost.
* Evaluated models using metrics like Accuracy, Precision, Recall, and ROC-AUC.
* Saved the final trained model using pickle or joblib.  
  **Deliverables:** Final trained ML model, model evaluation report, model.pkl file, EDA findings.

#### **Geetharani C – Frontend & UI Developer**

* Developed the frontend interface using HTML, CSS, and JavaScript.
* Created user input forms to capture patient health data interactively.
* Displayed prediction results and suggestions (e.g., food, yoga) clearly to users.
* Designed a clean and user-friendly interface layout for better usability.  
  **Deliverables:** Frontend files (index.html, styles.css, script.js), UI screenshots.

#### **Anusha S – Integration & Backend Developer**

* Integrated the trained machine learning model into a functional Flask-based web application.
* Handled backend logic including input preprocessing and prediction response formatting.
* Implemented input validation and error handling (e.g., empty or invalid fields).
* Developed and tested the /predict API endpoint using Flask.  
  **Deliverables:** Functional Flask app, app.py or backend scripts, working prediction API.

#### **Devisri V – Deployment & Documentation**

* Deployed the web application using platforms like Streamlit Cloud or Render.
* Wrote comprehensive project documentation covering objectives, methods, and outcomes.
* Created a presentation summarizing the entire project workflow and insights.
* (Optional) Recorded a demo video for showcasing the application.  
  **Deliverables:** Deployed app link, final project documentation (PDF), presentation deck.