**TABLE OF CONTENTS**

* ABSTRACT 1
* CHAPTER 1: INTRODUCTION 2 - 5
  + 1.1 Introduction 1 2
    - Purpose 3
    - Scope 4
    - Features 5
* CHAPTER 2: SYSTEM ANALYSIS 6 - 8
  + 2.1 Hardware Requirements 6
  + 2.2 Software Requirements 7 - 8
* CHAPTER 3: DEVELOPMENT ENVIRONMENT 9 - 10
  + 3.1 Language and Technologies Used 9 - 10
* CHAPTER 4: SYSTEM DESIGN & SPECIFICATIONS 11 - 15
  + 4.1 Flowchart 11
  + 4.2 DFD diagram 11.1
  + 4.3 Use case diagram 11.2
  + 4.4 Class Diagram 11.3
  + 4.5 Activity Diagram 11.4
  + 4.6 Screenshots of the Application 12 – 15
* CHAPTER 5: CODING IMPLEMENTATION 16 - 23
* CHAPTER 6: TESTING 24 - 15
* CONCLUSION 26
* LIMITATIONS 28
* REFERENCES 29

**ABSTRACT**

This report provides an in-depth overview of the development of a Heart Failure Detection Application designed to assist healthcare professionals and patients in assessing the risk of heart failure using machine learning techniques. Heart failure is a serious medical condition that requires early detection and intervention to improve patient outcomes, and this application aims to offer a predictive tool that can aid in the timely identification of individuals at risk. By leveraging clinical parameters such as age, sex, blood pressure, cholesterol levels, heart rate, and other relevant health data, the application employs machine learning algorithms to predict the likelihood of heart failure and provide actionable insights for both preventive measures and treatment planning.

The report covers the overall system design, highlighting the architecture, technologies, and tools used throughout the development process. It discusses the integration of various machine learning models for accurate prediction, the data preprocessing methods required to handle diverse medical datasets, and the user-friendly interface designed to ensure accessibility for both medical practitioners and patients. Additionally, the report elaborates on the database management system used to store and retrieve patient data, the use of Python for backend processing, and the deployment of the application on a suitable platform for widespread use.

Furthermore, the document includes a section on the testing procedures employed to evaluate the application’s performance. This encompasses model validation techniques, accuracy testing, and real-world simulation of heart failure risk predictions to ensure the application provides reliable and clinically meaningful results. The application has been rigorously tested for its predictive accuracy, robustness, and user experience to ensure its potential as a valuable tool for early diagnosis, treatment, and continuous patient monitoring.

By offering a data-driven approach to heart failure detection, this project aims to empower healthcare providers with a tool that enhances diagnostic accuracy, aids in making informed clinical decisions, and improves patient care. Ultimately, the goal of this Heart Failure Detection Application is to contribute to the prevention and management of heart failure, reducing its impact on patients' quality of life and healthcare systems. The application’s ability to provide personalized, data-backed insights underscores its potential as an innovative solution in the field of healthcare technology.

**CHAPTER - 1**

**1.1 INTRODUCTION**

Heart failure is a serious and growing health problem worldwide, affecting millions and leading to high mortality rates and substantial healthcare costs. It occurs when the heart cannot pump enough blood to meet the body's needs, often due to conditions such as coronary artery disease, high blood pressure, or previous heart attacks. This progressive condition has various stages, with early symptoms often being mild and easily overlooked. However, if left untreated, heart failure can rapidly progress to a life-threatening stage.

In recent years, predictive analytics has emerged as a promising tool in healthcare, particularly in diagnosing, monitoring, and managing chronic diseases such as heart failure. Predictive models trained on large datasets can analyze patient health data to identify risk factors, predict outcomes, and provide early warnings. This is especially crucial for heart failure, where early intervention can improve survival rates and quality of life.

**Current Heart Failure Statistics**

Heart failure is responsible for a significant portion of global healthcare burden. According to recent data from the World Health Organization (WHO), around 26 million people worldwide live with heart failure, with more than 1 million new cases diagnosed annually in the United States and Europe alone. In developed countries, heart failure affects 1-2% of the adult population, and the prevalence increases dramatically with age, reaching up to 10% in people over 70. Despite advances in treatment, the 5-year survival rate for heart failure remains low compared to other chronic conditions.

These statistics underline the critical need for solutions that can help detect heart failure risk factors early. This can prevent the disease from reaching severe stages, reduce hospital admissions, and improve patients' overall quality of life. As such, healthcare systems worldwide are increasingly investing in technology and tools that can assist with early detection, prompt intervention, and continuous monitoring of patients.

**Case Studies: The Impact of Early Detection on Patient Outcomes**

Case studies across healthcare institutions show that early detection and intervention can significantly impact outcomes for heart failure patients. For instance, a study conducted by the Cleveland Clinic found that implementing a predictive tool reduced hospitalization rates for heart failure patients by 30% within one year. Patients who were identified as high-risk received proactive care, such as medication adjustments and lifestyle recommendations, which contributed to the reduction in hospital visits.

Another study at a hospital in the United Kingdom tested an AI-based model that analyzed electronic health records to predict heart failure. This model allowed healthcare providers to identify patients at risk and intervene early, resulting in a 20% reduction in 30-day readmission rates for heart failure. These cases highlight the potential of predictive models to improve patient outcomes by enabling clinicians to make data-driven decisions.

**The Role of AI in Healthcare**

Artificial intelligence (AI) is transforming healthcare by providing insights from large volumes of data that would be challenging to analyse manually. In the context of heart failure prediction, AI models can process vast amounts of patient data—including age, gender, blood pressure, cholesterol levels, and other clinical parameters—to detect patterns associated with heart failure risk. Machine learning algorithms can learn from historical data and continuously improve their predictions, making them valuable tools for personalized patient care.

In recent years, numerous studies have demonstrated the effectiveness of AI in predicting disease outcomes. For example, a study published in the journal *Nature Medicine* found that an AI algorithm outperformed cardiologists in identifying heart failure from echocardiograms. Such AI-driven applications not only improve diagnostic accuracy but also enable healthcare systems to allocate resources more efficiently. By adopting AI in heart failure prediction, clinicians can focus on high-risk patients and provide timely intervention, ultimately improving survival rates and reducing costs.

**1.1.1 PURPOSE**

The primary purpose of this Heart Failure Detection Application is to assist healthcare providers in assessing patients' risk of heart failure using machine learning algorithms. By analyzing input data, the application generates a prediction indicating the likelihood of heart failure. This prediction can help clinicians prioritize at-risk patients and provide preventive measures, reducing the chances of hospitalization and severe health complications.

This application addresses several key challenges in healthcare:

1. **Early Detection**: By providing timely predictions, the application enables clinicians to identify high-risk patients before symptoms worsen.
2. **Cost Reduction**: Early intervention often requires less intensive treatment, which can lead to reduced healthcare costs.
3. **Improved Patient Outcomes**: Early identification and management of heart failure can significantly improve patients' quality of life, decrease the need for hospital visits, and extend lifespan.
4. **Data-Driven Decision-Making**: The application supports healthcare providers in making informed decisions based on patient data, enhancing the accuracy of risk assessment.

**1.1.2 SCOPE**

The scope of this project includes developing a predictive application that uses machine learning to analyze patient data and predict the risk of heart failure. The application is designed for use by healthcare providers in clinical settings and by individual users interested in monitoring their heart health. This multi-platform solution will provide insights via web and mobile applications, expanding its accessibility.

**Use Cases**

The Heart Failure Detection Application has various use cases, including:

* **Clinical Use**: Healthcare providers can use the application to evaluate the risk of heart failure in patients during regular check-ups or consultations. The prediction results can guide further testing, treatment adjustments, or specialist referrals.
* **Patient Monitoring**: Patients with risk factors for heart failure (e.g., those with hypertension or diabetes) can use the application for periodic assessments of their heart health.
* **Public Health Tool**: In regions where healthcare resources are limited, the application can support community health workers in screening large populations and identifying high-risk individuals who need further evaluation.

**Stakeholders**

The primary stakeholders for this application are:

* **Doctors and Healthcare Providers**: They benefit from having a quick and accurate tool to assess heart failure risk, allowing them to offer preventive care more effectively.
* **Patients**: The application offers patients greater insight into their heart health and encourages proactive health management.
* **Healthcare Administrators**: By using predictive analytics, administrators can improve resource allocation and manage patient care more efficiently.

**Limitations**

Despite its potential, the application faces certain limitations:

* **Data Privacy**: Managing sensitive patient data requires strict adherence to privacy regulations, such as the Health Insurance Portability and Accountability Act (HIPAA).
* **Accuracy of Predictions**: The model's predictions are based on historical data, which may not cover all patient profiles. Prediction accuracy may vary depending on factors like demographics and lifestyle.
* **Technical Requirements**: The application requires a reliable internet connection and computational resources for data processing and model inference, which may not be available in all settings.

**1.1.3 FEATURES**

The Heart Failure Detection Application has several core features designed to provide an accessible and accurate predictive tool for users. These features include:

1. **Input Fields and Validation**  
   Users are prompted to enter key health data, such as age, gender, cholesterol levels, and blood pressure. Each input field includes validation to ensure accurate data entry. For instance:
   * **Age and Blood Pressure**: Numeric input fields with ranges to prevent unrealistic values.
   * **Gender**: Drop-down options for standardized data entry.

Input validation reduces the risk of errors and ensures the model receives clean, usable data. These validations are implemented on both the frontend (using JavaScript) and backend (using Python) to reinforce data integrity.

1. **Multi-Platform User Interface (UI)**  
   The application is designed to work seamlessly across web and mobile platforms, making it accessible to a broad audience. The UI provides an intuitive, user-friendly experience, with clearly labeled fields and easy navigation.

**Web Interface (Streamlit)**: The web version of the application is built with Streamlit, a Python framework that simplifies the development of data-driven applications. The interface includes navigation tabs for input, prediction results, and history, allowing users to manage their data efficiently.

**CHAPTER - 2**

**SYSTEM ANALYSIS**

In this chapter, we delve into the system requirements necessary for developing, deploying, and running the Heart Failure Detection Application. The system analysis considers both hardware and software needs, outlining why specific resources were chosen for this application. Given the application's predictive capabilities, the goal is to ensure a reliable and responsive experience for both healthcare providers and patients.

**2.1 HARDWARE REQUIREMENTS**

The hardware requirements for the Heart Failure Detection Application depend on the environment in which the application is being used. We detail the specifications necessary for both **development** and **deployment** environments, noting the essential components and variations in each scenario.

**2.1.1 DEVELOPMENT ENVIRONMENT**

**Processor**: Intel Core i5 (minimum) or equivalent AMD processor

* + A robust processor is required for developing, testing, and running the application, especially when dealing with machine learning models. Intel Core i5 or higher ensures smooth code compilation and model inference, which is crucial when testing locally.

1. **RAM**: 8GB (minimum), 16GB (recommended)
   * For efficient multitasking and handling large data files, 8GB of RAM is a baseline requirement. However, 16GB is recommended for a smoother experience, especially when training or testing machine learning models locally.
2. **Storage**: SSD with at least 256GB
   * Since the development environment involves working with multiple datasets and software libraries, an SSD is recommended for faster read and write speeds. This improves the overall speed of loading datasets and deploying models.
3. **GPU (Optional)**: NVIDIA GTX 1050 or higher
   * For machine learning model training, having a dedicated GPU can significantly reduce the training time. While not essential for the deployment stage, a GPU is beneficial during development, allowing for more efficient model training and testing.
4. **Operating System**: Windows 10, macOS, or Ubuntu 18.04+
   * Compatibility with popular operating systems ensures a wider range of developer flexibility. Ubuntu is commonly used for its compatibility with Python and machine learning frameworks.

**2.1.2 DEPLOYMENT ENVIRONMENT**

**Cloud-based Servers**: AWS EC2, Google Cloud, or Azure Virtual Machine with 4 vCPUs and 8GB RAM (minimum)

* + Cloud-based servers allow for scalable deployment. A virtual machine with 4 vCPUs and at least 8GB of RAM supports the application’s backend processing and API request handling efficiently.

1. **GPU (for Model Deployment)**: NVIDIA Tesla T4 or equivalent (if complex model training is involved)
   * For real-time predictions and a more complex neural network model, a GPU-equipped server can be advantageous in deployment. However, for basic predictions, CPU-based deployment is sufficient.
2. **Storage**: 50GB SSD (minimum)
   * Cloud storage requirements are minimized, but an SSD with a minimum of 50GB ensures quick data retrieval for the predictive model and any user data caching.
3. **Network**: 1 Gbps or higher
   * High-speed network connectivity is vital for handling multiple concurrent requests, especially if the application is deployed in a hospital or clinical setting.

**2.2 SOFTWARE REQUIREMENTS**

This section lists the essential software tools, libraries, and frameworks used in the development and deployment of the Heart Failure Detection Application. Each software component has been chosen based on specific requirements, functionality, and reliability.

**2.2.1 FRONTEND DEVELOPMENT**

1. **Streamlit (for Web)**
   * **Description**: Streamlit is a Python-based web framework for building data-driven applications with a minimalistic UI.
   * **Justification**: Streamlit’s simplicity and compatibility with Python make it ideal for rapidly building and deploying data science applications. Unlike full-fledged frameworks like React or Angular, Streamlit requires minimal front-end coding, reducing development time. Its support for data visualization allows clinicians and users to understand model results more effectively

**2.2.2 MACHINE LEARNING LIBRARIES**

1. **scikit-learn**
   * **Description**: scikit-learn is a Python library used for model training, including logistic regression, random forest, and support vector machines.
   * **Justification**: scikit-learn offers a comprehensive suite of tools for model building, training, evaluation, and tuning. It is highly efficient and well-documented, making it the preferred choice for this application.
2. **pandas and NumPy**
   * **Description**: These libraries provide data handling capabilities, including data cleaning, manipulation, and preparation.
   * **Justification**: pandas and NumPy are widely used in data science for their flexibility and performance. They are essential for preparing input data before feeding it into the model, ensuring the prediction accuracy.
3. **Pickle**
   * **Description**: Pickle is a Python module used for serializing and deserializing Python objects.
   * **Justification**: Pickle is used to save the trained machine learning model, which can then be loaded by the Flask API for predictions. This serialization enables the application to use the pre-trained model without retraining each time, reducing computation time.

**2.2.3 ADDITIONAL TOOLS AND FRAMEWORKS**

1. **Git**
   * **Description**: Git is used for version control and code management.
   * **Justification**: Git allows developers to track changes, collaborate effectively, and revert to previous code versions if necessary. Its integration with platforms like GitHub facilitates team collaboration, which is essential for large-scale projects.

**CHAPTER - 3**

**DEVELOPMENT ENVIRONMENT**

**3.1 LANGUAGES AND TECHNOLOGIES USED**

In this section, we will discuss the primary languages and technologies utilized in the development of the AI-Powered Student Assistance Chatbot, emphasizing their functionalities and importance.

**Python**

Python is a versatile programming language known for its simplicity and readability. Its syntax is designed to be intuitive, allowing developers to express concepts in fewer lines of code compared to other languages. This ease of use has made Python a preferred choice for many developers, particularly in the fields of data science and machine learning.

* **Popular Libraries for Machine Learning**:
  + **Scikit-Learn**: This is a powerful library for machine learning that provides a range of supervised and unsupervised learning algorithms. Scikit-Learn simplifies the model training process through its consistent API and built-in cross-validation methods.
  + **NumPy**: NumPy is a fundamental package for scientific computing in Python. It offers support for large multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is crucial for preprocessing data, as it enables efficient handling of large datasets, which is essential for machine learning tasks.
  + **Pandas**: Often used alongside NumPy, Pandas provides data structures for efficiently storing and manipulating large datasets. Its DataFrame object is particularly useful for data preprocessing and cleaning tasks.

**JavaScript and HTML**

JavaScript and HTML are the backbone of front-end development, enabling interactive web applications. HTML structures the content of web pages, while JavaScript adds dynamic functionality, allowing for real-time user interaction.

**Customization in Front-end Elements**:

* **HTML**: The use of HTML5 provides semantic tags that enhance the accessibility and SEO of web applications. Customizing the structure of HTML elements enables better user interfaces (UIs) and user experiences (UX). For example, using <section>, <article>, and <nav> elements helps create a clear hierarchy of content.
* **JavaScript**: JavaScript allows developers to manipulate the Document Object Model (DOM), enabling dynamic updates to the web page without reloading. Customizations often involve creating event listeners, managing user inputs, and displaying responses from the server. Using modern JavaScript frameworks like React or Vue.js can significantly improve the structure and maintainability of front-end code.

**Machine Learning Libraries**

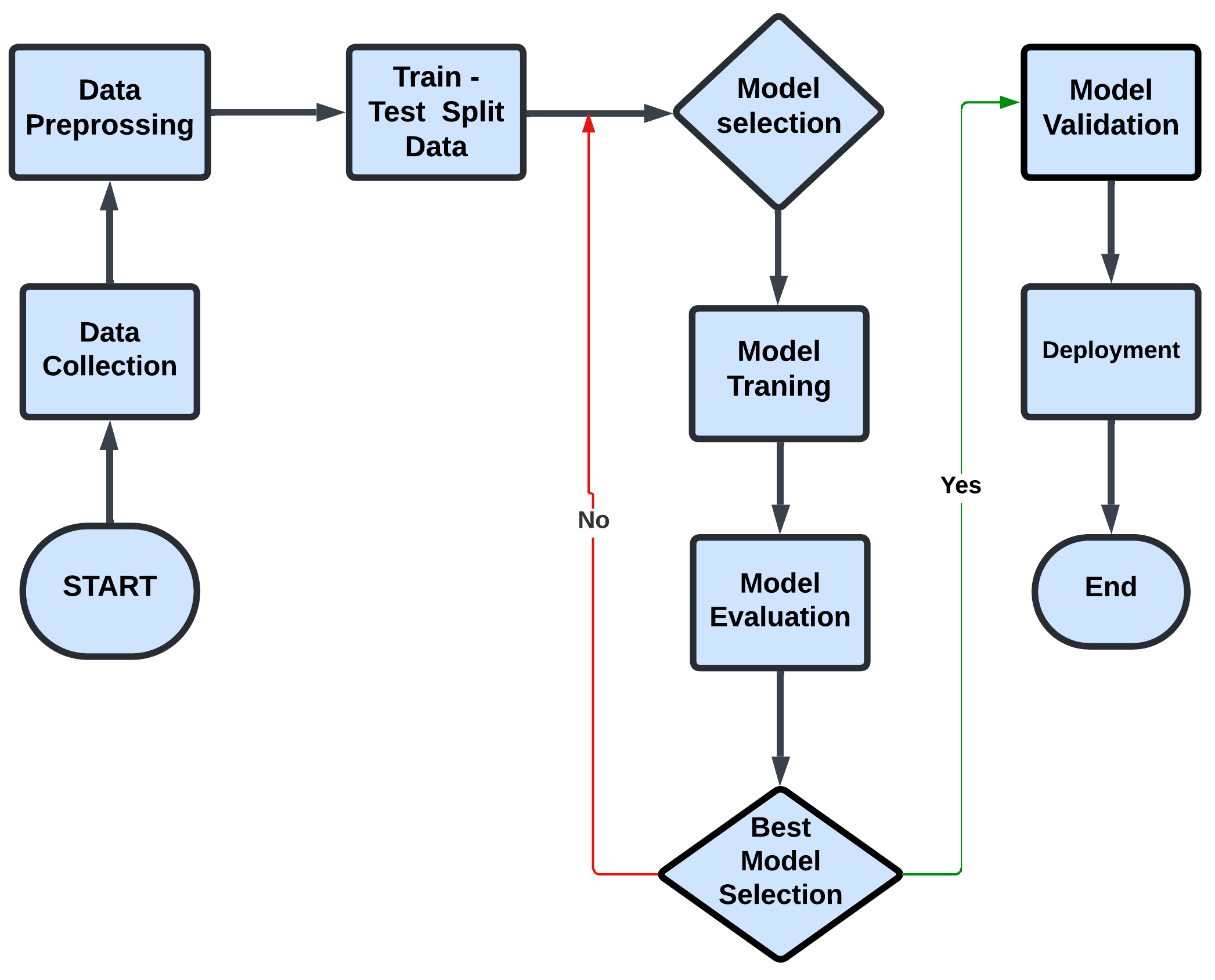
Machine learning libraries play a pivotal role in the development of AI models, handling everything from data preprocessing to model training and evaluation.

* **Scikit-Learn**: This library is essential for implementing various machine learning algorithms. It provides tools for:
  + **Data Preprocessing**: Scikit-Learn includes functions for scaling, normalizing, and transforming data, which are crucial for improving model performance. For instance, using StandardScaler to standardize features by removing the mean and scaling to unit variance can lead to better convergence during training.
  + **Model Training**: Scikit-Learn supports a wide array of algorithms, including linear regression, decision trees, and support vector machines. The library offers a straightforward way to train and evaluate models through its fit and predict methods.
* **NumPy**: In addition to its role in handling arrays, NumPy is integral in performing numerical operations essential for machine learning tasks. It supports:
  + **Array Manipulations**: NumPy enables efficient operations on large datasets, which is critical when dealing with the high-dimensional data often encountered in machine learning.
  + **Mathematical Functions**: Many machine learning algorithms rely on matrix operations, and NumPy provides a robust framework for linear algebra operations, such as dot products, eigenvalues, and singular value decomposition (SVD).
* **Pandas**: This library complements Scikit-Learn and NumPy by providing tools for data manipulation. Key functionalities include:
  + **Data Cleaning**: Pandas helps handle missing values, duplicates, and incorrect data types, which are common in real-world datasets. Functions like fillna() and drop\_duplicates() are particularly useful.
  + **Data Exploration**: With Pandas, developers can quickly explore datasets using methods like describe() and info(), enabling them to understand the distribution and characteristics of the data before training models.

**CHAPTER - 4**

**SYSTEM DESIGN & SPECIFICATIONS**

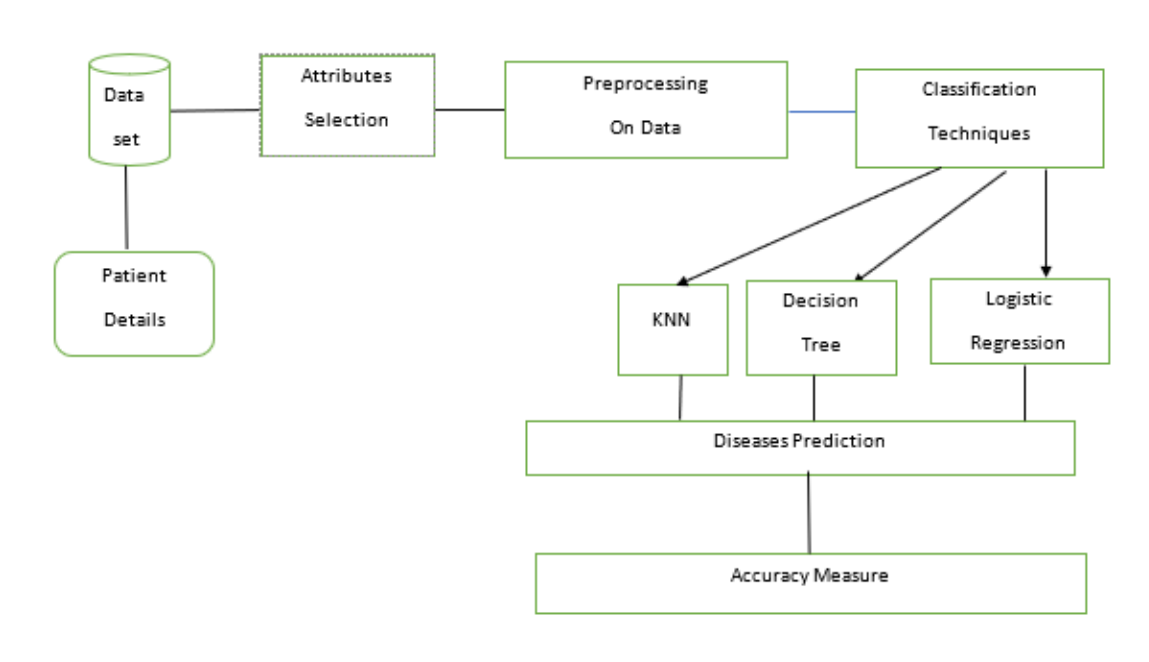
**Flowchart:**



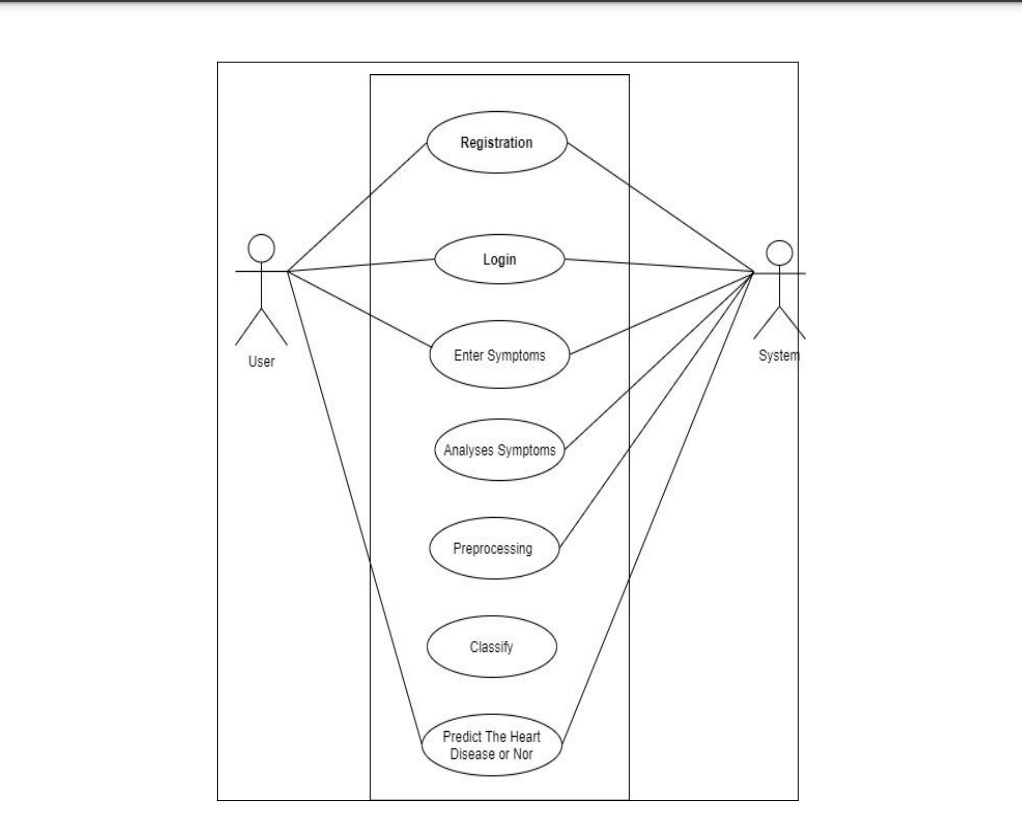
**4.2 DFD diagram:**

**Data Flow Diagram (DFD) – Level 1**

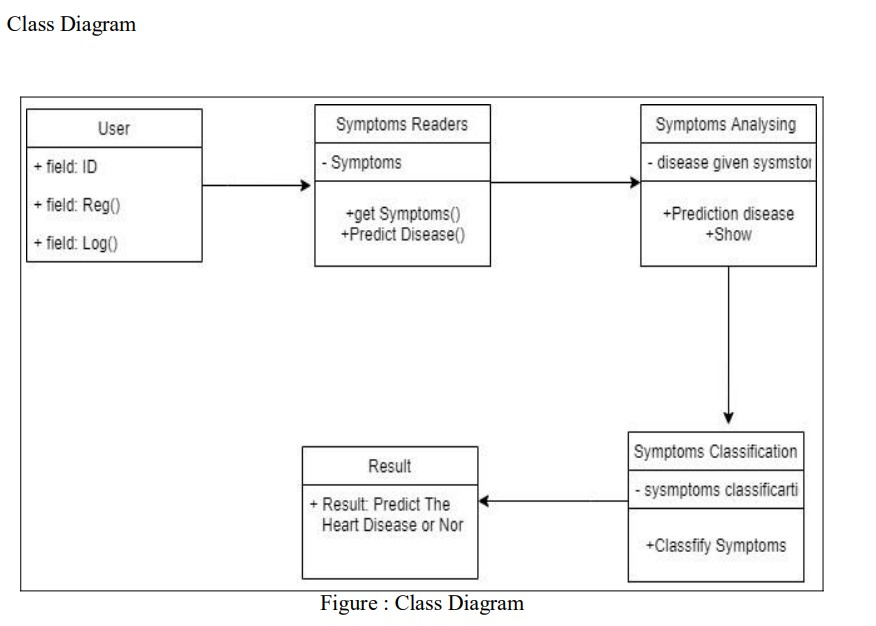
1. **Entities**:
   * **User**: Inputs health data.
   * **Email Service**: Sends reports via email.
2. **Processes**:
   * **Data Input**: User inputs health metrics.
   * **Data Validation**: Ensures data format correctness (e.g., numeric checks).
   * **Prediction Model**: Uses validated data to predict heart failure risk.
   * **Send Report (Email Service)**: Sends the generated prediction report to the user's email if requested.
3. **Data Stores**:
   * **User Data Store**: Temporarily holds user input during the session.
   * **Prediction Result Store**: Holds the current session’s prediction result for inclusion in the report.



**4.3 Use case diagram :**



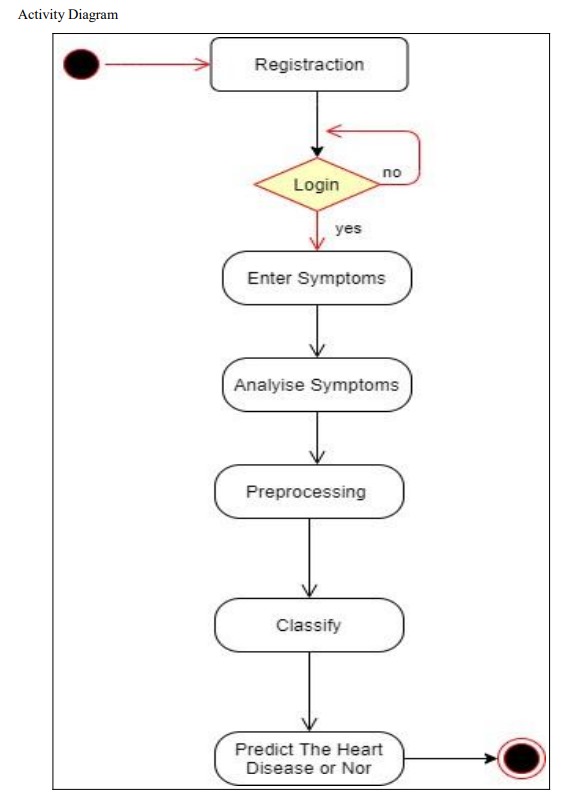
A Use Case Diagram visually represents the various interactions between the system and its users. This diagram serves as a roadmap to show how different actors interact with the system's functionalities. For the heart failure predictive model, we can define a few key actors and their respective use cases.

**4.4 Class Diagram :**

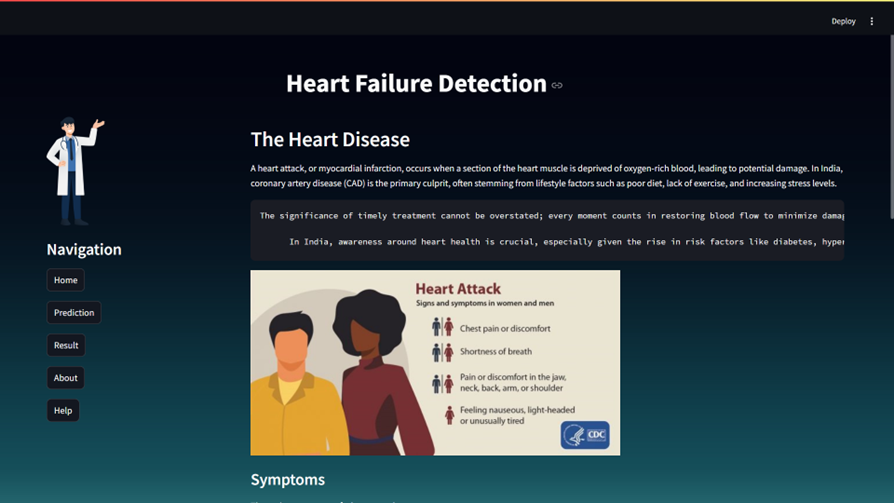
The Class Diagram provides an intricate blueprint of the system's core components, highlighting the system’s key classes, their attributes, methods, and relationships. The class diagram for this system delves into the core objects that intertwine to create the heart failure prediction and reporting system.

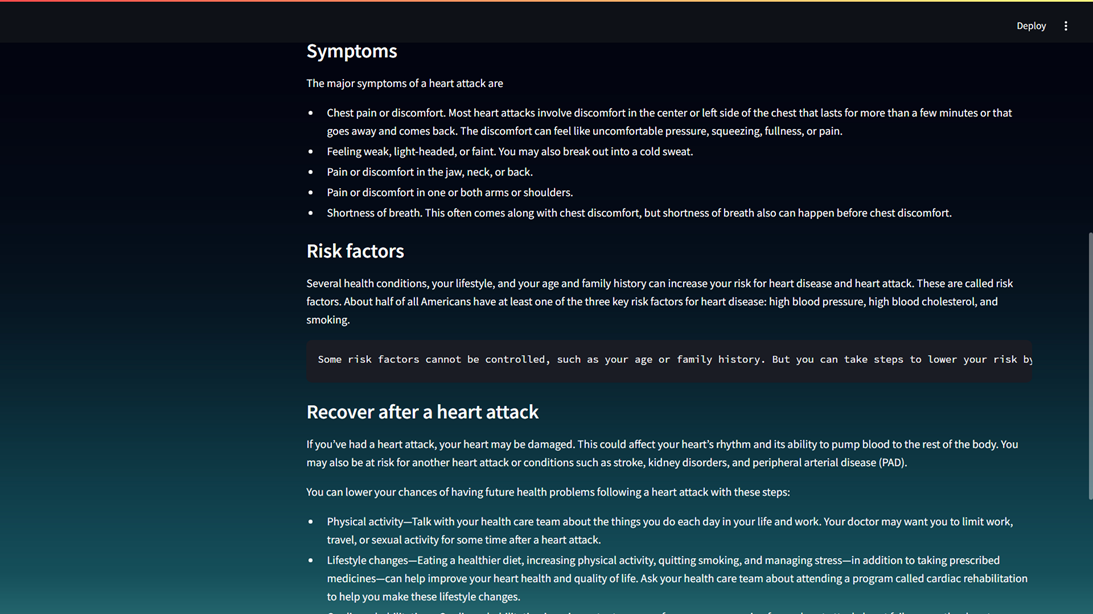
**4.5 Activity Diagram:**

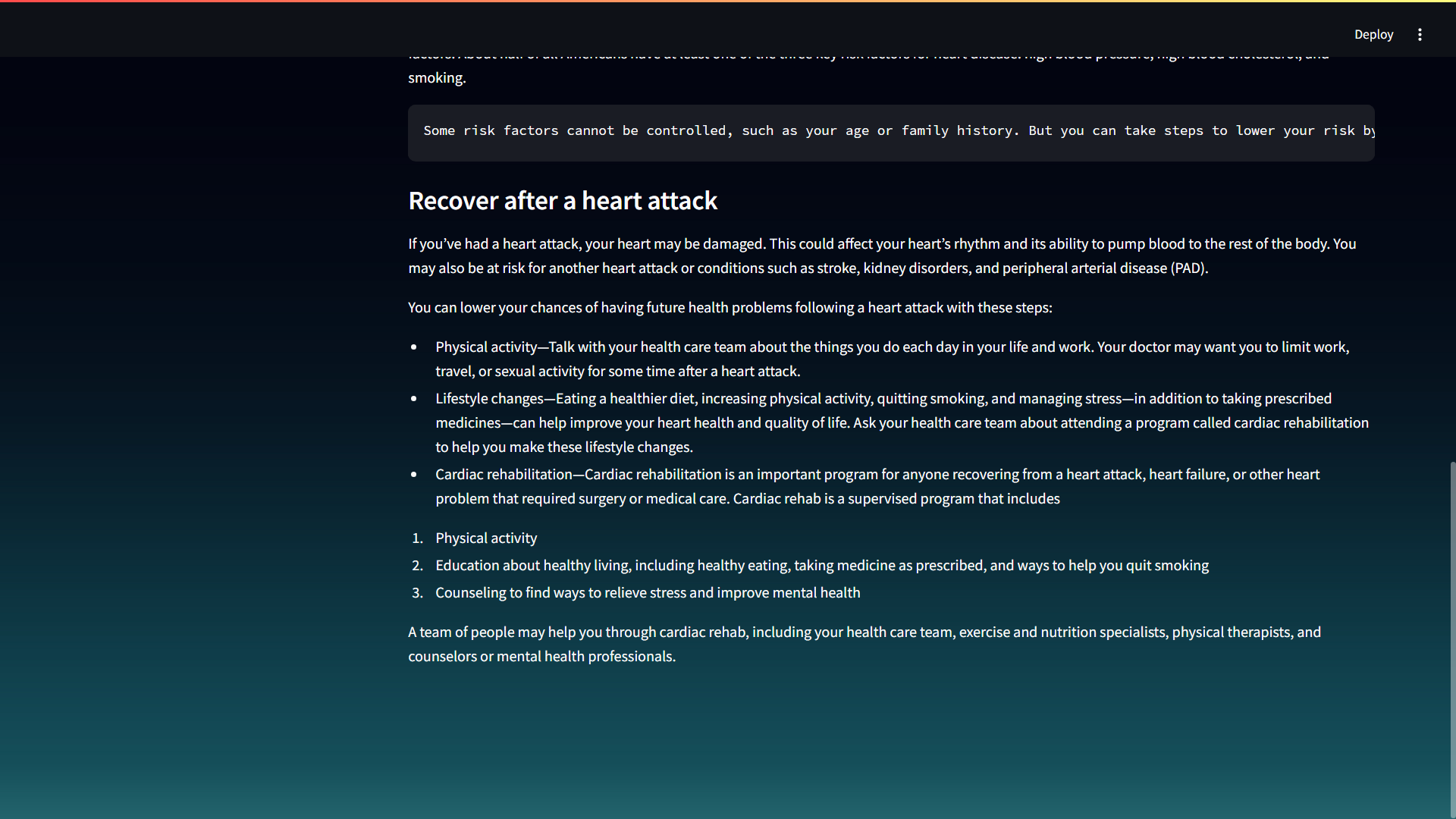
An Activity Diagram provides a visual representation of the flow of activities or operations in the system. It captures the sequence of steps and decisions that take place in various processes within the heart failure predictive model. The diagram helps to understand the flow of control from one activity to another, and how different system components interact during each phase of the process.

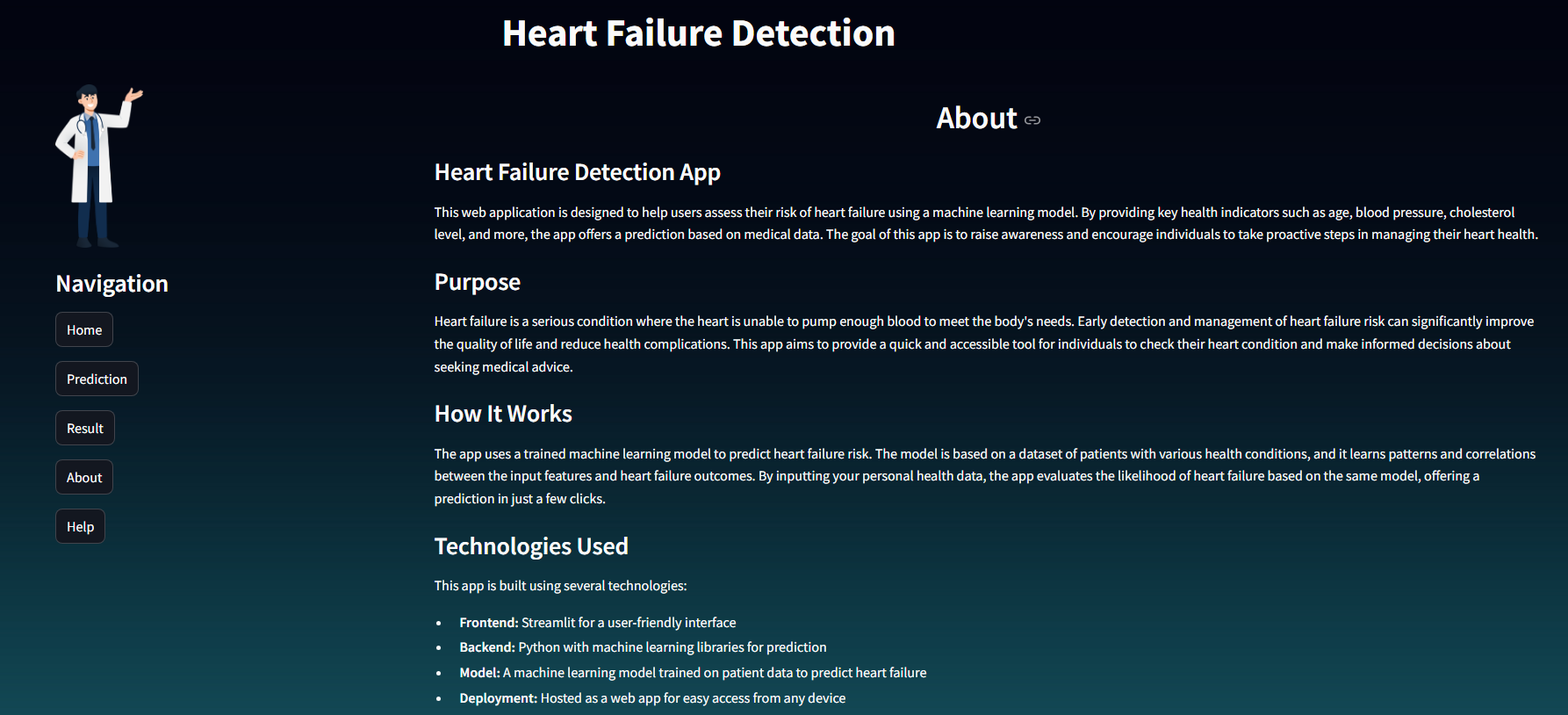


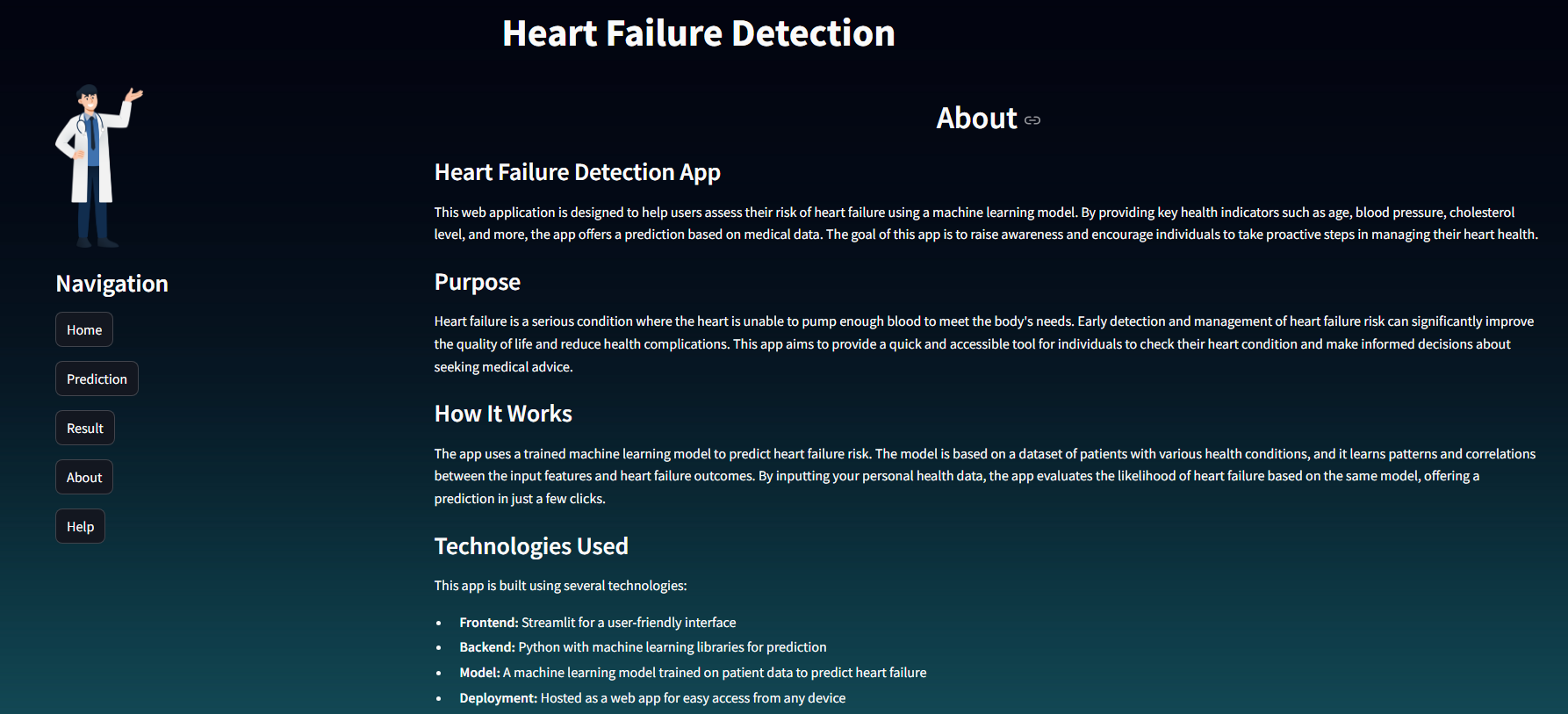
**Screenshot**

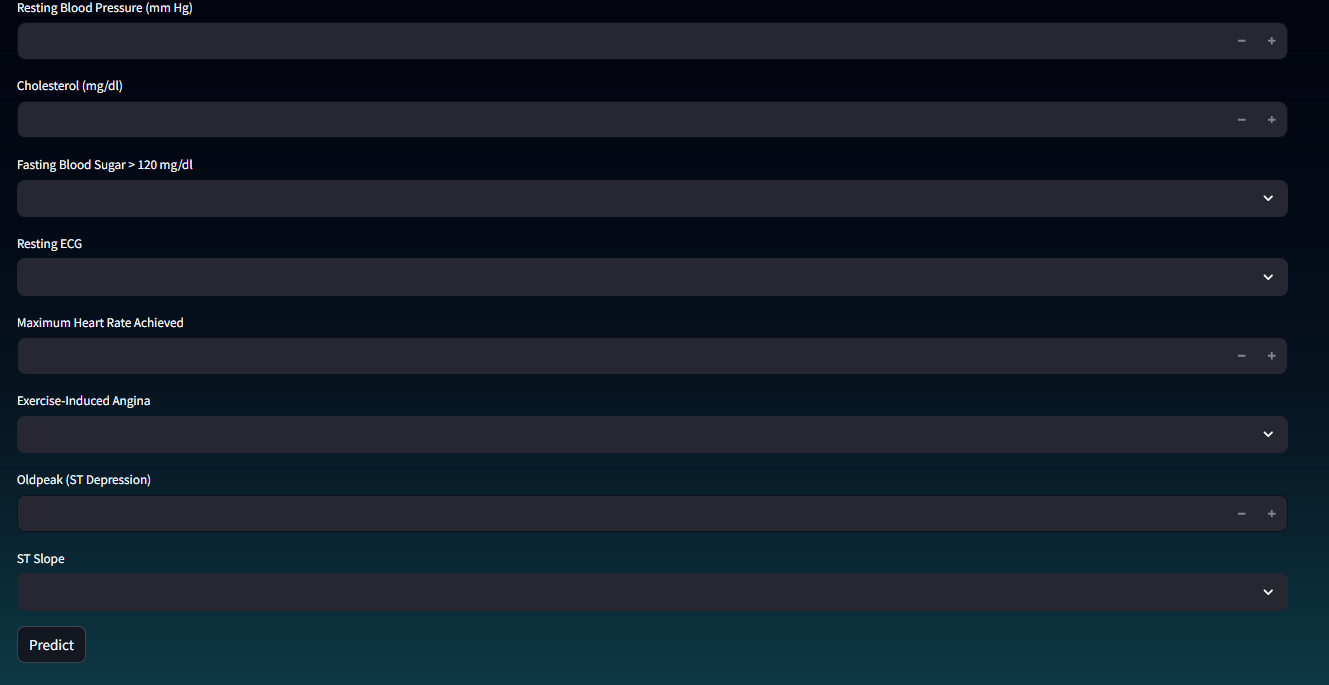
****

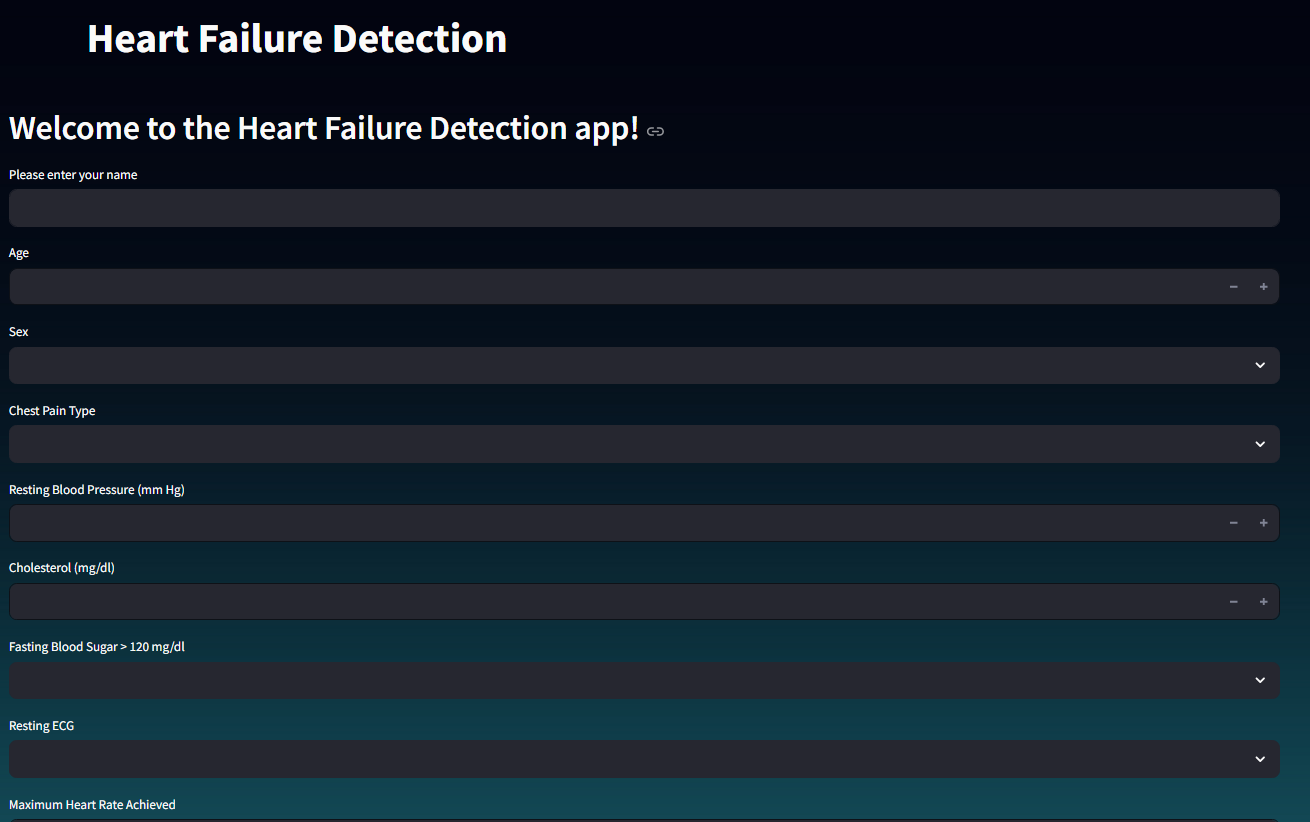
****

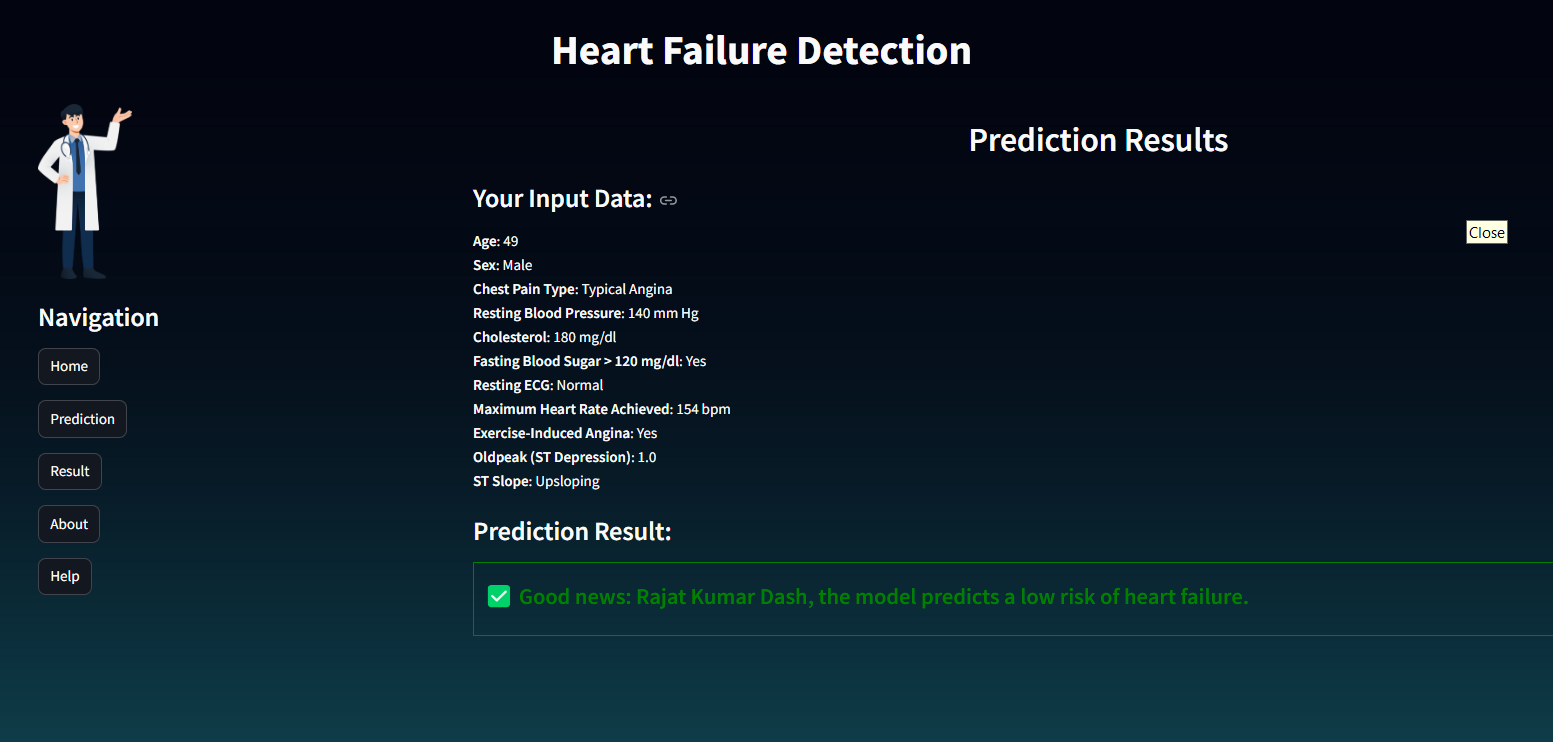
****









****

**CHAPTER - 5**

**CODING IMPLEMENTATION**

import streamlit as st

import pickle

import numpy as np

# Load the trained model

with open('a.pkl', 'rb') as model\_file:

    model = pickle.load(model\_file)

# Title of the web app

st.markdown("<h1 style='margin-left:30%'>Heart Failure Detection</h1>", unsafe\_allow\_html=True)

# Add background image using CSS

page\_bg\_img = '''

<style>

.stApp {

  background-image: url("https://rare-gallery.com/uploads/posts/5002028-dark-gradient-artist-artwork-digital-art-hd-4k-blur-simple-background.jpg");

  background-size: cover;

  background-position: top;

}

</style>

'''

st.markdown(page\_bg\_img, unsafe\_allow\_html=True)

# Create two columns: one for the image and buttons, the other for dynamic content

col1, col2 = st.columns([1, 3])  # 30% left, 70% right

# Left column for the image and navigation

with col1:

    st.image("https://static.vecteezy.com/system/resources/thumbnails/021/360/193/small\_2x/doctor-character-illustration-free-png.png", width=100)

    st.markdown("<h3>Navigation</h3>", unsafe\_allow\_html=True)

    # Buttons for navigation with session state to manage content

    if 'page' not in st.session\_state:

        st.session\_state.page = 'home'  # Default page is 'home'

    if st.button("Home"):

        st.session\_state.page = 'home'

    if st.button("Prediction"):

        st.session\_state.page = 'pd'

    if st.button("Result"):

        st.session\_state.page = 'prediction'

    if st.button("About"):

        st.session\_state.page = 'about'

    if st.button("Help"):

        st.session\_state.page = 'help'

# Function to display user input and make predictions

def display\_prediction(input\_data):

    # Mapping user inputs to model-friendly format

    sex\_map = {"Male": 1, "Female": 0}

    fasting\_bs\_map = {"Yes": 1, "No": 0}

    exercise\_angina\_map = {"Yes": 1, "No": 0}

    chest\_pain\_map = {"Typical Angina": 0, "Atypical Angina": 1, "Non-Anginal Pain": 2, "Asymptomatic": 3}

    resting\_ecg\_map = {"Normal": 0, "ST-T Wave Abnormality": 1, "Left Ventricular Hypertrophy": 2}

    st\_slope\_map = {"Upsloping": 0, "Flat": 1, "Downsloping": 2}

    # Validate and provide fallbacks for missing inputs (if any)

    try:

        input\_vector = np.array([

            input\_data['age'],

            sex\_map[input\_data['sex']],

            chest\_pain\_map[input\_data['chest\_pain\_type']],

            input\_data['resting\_bp'],

            input\_data['cholesterol'],

            fasting\_bs\_map[input\_data['fasting\_bs']],

            resting\_ecg\_map[input\_data['resting\_ecg']],

            input\_data['max\_hr'],

            exercise\_angina\_map[input\_data['exercise\_angina']],

            input\_data['oldpeak'],

            st\_slope\_map[input\_data['st\_slope']]

        ]).reshape(1, -1)

    except KeyError as e:

        st.error(f"Missing or invalid input: {e}")

        return

    # Predict heart failure based on user input

    try:

        prediction = model.predict(input\_vector)

    except Exception as e:

        st.error(f"Error during prediction: {e}")

        return

    # Display the input data provided by the user

    st.subheader("Your Input Data:")

    st.markdown(f"""

    \*\*Age\*\*: {input\_data['age']}

    \*\*Sex\*\*: {input\_data['sex']}

    \*\*Chest Pain Type\*\*: {input\_data['chest\_pain\_type']}

    \*\*Resting Blood Pressure\*\*: {input\_data['resting\_bp']} mm Hg

    \*\*Cholesterol\*\*: {input\_data['cholesterol']} mg/dl

    \*\*Fasting Blood Sugar > 120 mg/dl\*\*: {input\_data['fasting\_bs']}

    \*\*Resting ECG\*\*: {input\_data['resting\_ecg']}

    \*\*Maximum Heart Rate Achieved\*\*: {input\_data['max\_hr']} bpm

    \*\*Exercise-Induced Angina\*\*: {input\_data['exercise\_angina']}

    \*\*Oldpeak (ST Depression)\*\*: {input\_data['oldpeak']}

    \*\*ST Slope\*\*: {input\_data['st\_slope']}

    """)

    # Display prediction result with styles inside a box

    st.subheader("Prediction Result:")

    if prediction[0] == 1:

        st.markdown(f'<div style="border: 2px solid red; padding: 10px;"><h4 style="color:red;">⚠️ Warning: {st.session\_state.user\_name}, the model predicts a high risk of heart failure.</h4></div>', unsafe\_allow\_html=True)

    else:

        st.markdown(f'<div style="border: 2px solid green; padding: 10px;"><h4 style="color:green;">✅ Good news: {st.session\_state.user\_name}, the model predicts a low risk of heart failure.</h4></div>', unsafe\_allow\_html=True)

# Right column for dynamic content based on the selected button

with col2:

    if st.session\_state.page == 'pd':

        st.markdown('<h2>Welcome to the Heart Failure Detection app!</h2>', unsafe\_allow\_html=True)

        # Ask for the user's name

        user\_name = st.text\_input("Please enter your name")

        # Input fields for user to provide attributes, with default blank or reasonable placeholders

        age = st.number\_input("Age", min\_value=5, max\_value=120, value=None)

        sex = st.selectbox("Sex", options=["", "Male", "Female"], index=0)

        chest\_pain\_type = st.selectbox("Chest Pain Type", options=["", "Typical Angina", "Atypical Angina", "Non-Anginal Pain", "Asymptomatic"], index=0)

        resting\_bp = st.number\_input("Resting Blood Pressure (mm Hg)", min\_value=50, max\_value=200, value=None)

        cholesterol = st.number\_input("Cholesterol (mg/dl)", min\_value=125, max\_value=400, value=None)

        fasting\_bs = st.selectbox("Fasting Blood Sugar > 120 mg/dl", options=["", "Yes", "No"], index=0)

        resting\_ecg = st.selectbox("Resting ECG", options=["", "Normal", "ST-T Wave Abnormality", "Left Ventricular Hypertrophy"], index=0)

        max\_hr = st.number\_input("Maximum Heart Rate Achieved", min\_value=60, max\_value=220, value=None)

        exercise\_angina = st.selectbox("Exercise-Induced Angina", options=["", "Yes", "No"], index=0)

        oldpeak = st.number\_input("Oldpeak (ST Depression)", min\_value=0.0, max\_value=10.0, step=0.1, value=None)

        st\_slope = st.selectbox("ST Slope", options=["", "Upsloping", "Flat", "Downsloping"], index=0)

        # Store the user input in session state

        if st.button("Predict"):

            if user\_name:

                st.session\_state.user\_name = user\_name  # Store the name in session state

            else:

                st.error("Please enter your name.")

            # Check if all inputs are valid before proceeding

            if (resting\_bp is not None and resting\_bp >= 50 and

                cholesterol is not None and cholesterol >= 100 and

                max\_hr is not None and max\_hr >= 60 and

                oldpeak is not None and oldpeak >= 0.0):

                st.session\_state.page = 'prediction'

                st.session\_state.input\_data = {

                    'age': age,

                    'sex': sex,

                    'chest\_pain\_type': chest\_pain\_type,

                    'resting\_bp': resting\_bp,

                    'cholesterol': cholesterol,

                    'fasting\_bs': fasting\_bs,

                    'resting\_ecg': resting\_ecg,

                    'max\_hr': max\_hr,

                    'exercise\_angina': exercise\_angina,

                    'oldpeak': oldpeak,

                    'st\_slope': st\_slope

                }

            else:

                st.error("Please provide valid inputs for all fields within the specified ranges.")

    elif st.session\_state.page == 'prediction':

        st.markdown('<h2 style="text-align: center;">Prediction Results</h2>', unsafe\_allow\_html=True)

        if 'input\_data' not in st.session\_state:

            st.error("No input data found! Please go to the Home page and provide the necessary inputs.")

        else:

            display\_prediction(st.session\_state.input\_data)

    elif st.session\_state.page=='home':

         st.header("The Heart Disease")

         st.write("""A heart attack, or myocardial infarction, occurs when a section of the heart muscle is deprived of oxygen-rich blood, leading to potential damage. In India, coronary artery disease (CAD) is the primary culprit, often stemming from lifestyle factors such as poor diet, lack of exercise, and increasing stress levels.

        - \*\*Left Ventricular Hypertrophy (LVH):\*\* This shows that the heart's left ventricle is enlarged, often due to high blood pressure or other heart conditions.''')

        # ST Slope Information

        st.subheader('''ST Slope:''')

        st.markdown('''

        - \*\*Upsloping:\*\* The ST segment rises upward in an ECG tracing, and this pattern can be normal or indicative of early-stage ischemia.

        - \*\*Flat:\*\* A flat ST segment could be a sign of heart disease or ischemia.

        - \*\*Downsloping:\*\* A downsloping ST segment is often a concerning sign and could indicate more severe heart issues like ischemia or a blockage in the heart arteries.''')

        # Additional Help

        st.subheader('Additional Help:')

        st.markdown('''

        If you are unsure about any of the inputs, refer to the following:

        - \*\*Age:\*\* Enter your age in years.

        - \*\*Sex:\*\* Choose your biological sex (Male/Female).

        - \*\*Resting Blood Pressure:\*\* Enter your resting blood pressure in mm Hg.

        - \*\*Cholesterol Level:\*\* Enter your cholesterol level in mg/dL.

        - \*\*Fasting Blood Sugar:\*\* This is your blood sugar level after fasting (1 = fasting blood sugar > 120 mg/dL, 0 = otherwise).

        - \*\*Maximum Heart Rate (MaxHR):\*\* This is the highest heart rate achieved during exercise.

        - \*\*Exercise-Induced Angina:\*\* Choose if you experience chest pain during physical activity (Y/N).

        - \*\*Oldpeak:\*\* This measures the ST depression induced by exercise relative to rest (enter the value in millimeters).''')

**CHAPTER - 6**

**TESTING**

**6.1 Unit Testing**

Unit testing involves verifying that individual component of the application function correctly. This section focuses on example test cases for core functions in both the model and API.

**1. Test Cases for Core Functions**

Begin by identifying the critical functions within the application that require testing. These may include data preprocessing functions, prediction algorithms, and API endpoints.

**2. Example Test Cases**

* **Input Validation**: Create test cases to ensure that the application properly validates input data types and ranges. This includes checking for invalid inputs and ensuring that appropriate errors are raised when invalid data is submitted.
* **Prediction Function**: Develop test cases to verify that the prediction function returns expected results based on given inputs. These tests can ensure that the outputs align with predefined expected values for a variety of test scenarios.

**3. Running Unit Tests**

Describe the process of executing unit tests, including how to use a testing framework like pytest. Mention how to interpret the test results and address any failures or issues that arise.

**6.2 Integration Testing**

Integration testing evaluates how different components of the application work together. This section focuses on end-to-end testing of the frontend-backend communication.

**1. Test Flows for End-to-End Testing**

Define a comprehensive test flow that simulates a user inputting data through the frontend and receiving a prediction from the backend. This flow should capture all user interactions and corresponding system responses.

**2. Specific Testing Frameworks**

Highlight the use of testing frameworks like pytest for conducting integration tests. Describe how these frameworks can facilitate the testing of both unit and integration aspects, providing robust test coverage.

**6.3 User Acceptance Testing (UAT)**

User Acceptance Testing focuses on validating the application against user requirements and expectations. This section summarizes user feedback and subsequent improvements made.

**1. Feedback Received from Users**

Outline the methods employed to gather user feedback, such as surveys or interviews. Describe how users interacted with the application and the insights gained from their experiences.

**2. Improvements Made**

List any enhancements implemented based on user feedback. These may include changes to the user interface, added features, or adjustments to existing functionalities to improve user experience.

**3. UAT Checklists**

Provide a detailed UAT checklist that was used to evaluate the application against specific requirements. This checklist should include key features, criteria for acceptance, and pass/fail outcomes for each item assessed.

**4. Result Summaries**

Summarize the results of the UAT process, highlighting significant insights and patterns that emerged from user interactions with the application. Discuss any areas for improvement that were identified.

**6.4 Performance Testing**

Performance testing assesses the application's behavior under various conditions to ensure it meets performance standards. This section presents findings from performance testing efforts.

**1. Performance Under Different Conditions**

Define various testing scenarios, such as simulating different user loads or varying the size of input data. Describe how these scenarios were structured to evaluate the application's performance.

**2. Graphs/Tables**

Include graphical representations or tables that illustrate key performance metrics, such as response times observed under different conditions. Summarize how response times varied with changes in user load or input size.

**3. Analysis of Memory Usage**

Discuss the importance of memory profiling during performance testing. Describe how monitoring memory usage can provide insights into the application's efficiency and help identify potential memory leaks or bottlenecks.

**4. Response Time Analysis**

Compare expected response times to actual results gathered during testing. Analyze any discrepancies, discussing possible reasons for slower-than-expected performance and potential areas for optimization.

**CONCLUSION**

In this project, we developed an AI-powered heart failure prediction application aimed at enhancing patient care and supporting healthcare providers in making informed decisions. The primary purpose of this project was to leverage machine learning techniques to analyze patient data and provide accurate predictions regarding the risk of heart failure, ultimately contributing to early intervention and improved patient outcomes.

**Achievements**

Throughout the development process, we successfully implemented a robust machine learning model using various algorithms, including logistic regression and decision trees. Our application allows users to input critical health parameters, which are processed through the trained model to generate real-time risk assessments. The frontend was designed with user experience in mind, incorporating intuitive navigation and a clear presentation of results, including risk levels and recommendations. The thorough testing phases—comprising unit, integration, and user acceptance testing—ensured the reliability and accuracy of our application.

**Benefits for Users and Healthcare Providers**

The heart failure prediction application offers significant benefits for both patients and healthcare providers. For users, particularly those at risk of heart failure, the application provides a valuable tool for self-assessment, empowering them to seek timely medical advice based on their risk level. For healthcare providers, the application serves as a decision-support tool, facilitating early identification of high-risk patients and enabling more personalized treatment plans. By integrating this technology into clinical workflows, healthcare professionals can enhance their ability to manage patient care proactively, ultimately leading to better health outcomes and reduced healthcare costs.

**Future Possibilities**

Looking ahead, there are several promising avenues for future work. One potential direction is to further train and refine the machine learning model with additional data, which could improve its accuracy and reliability. This might involve incorporating larger datasets, exploring advanced algorithms, or employing ensemble methods to enhance predictive performance. Moreover, expanding the application to include predictions for other cardiovascular diseases could significantly broaden its utility and impact in the healthcare domain.

Additionally, implementing features such as real-time data integration from wearable devices or electronic health records could provide a more comprehensive view of a patient’s health status, allowing for continuous monitoring and intervention when necessary. Collaboration with healthcare professionals for feedback and insights will be essential in evolving the application to meet clinical needs effectively.

In summary, this project not only demonstrates the feasibility of using machine learning for health risk predictions but also paves the way for future innovations in patient care and healthcare technology.

**LIMITATIONS**

While the heart failure prediction application represents a significant advancement in leveraging machine learning for healthcare, it is important to acknowledge certain limitations that may impact its overall effectiveness and applicability.

**Data Limitations**

One of the primary limitations faced in the development of the predictive model is the reliance on limited datasets. The model's performance is heavily dependent on the quality and quantity of the training data. In many cases, healthcare datasets may be small, incomplete, or biased, leading to potential overfitting and undergeneralization of the model. If the training data does not adequately represent the diverse population it aims to serve, the model may fail to perform effectively for certain demographic groups or patients with unique health profiles. Additionally, if critical variables that influence heart failure risk are not included in the dataset, the model may miss important predictors, ultimately affecting its predictive accuracy.

**Accuracy Constraints**

The accuracy of the predictions generated by the application is another area that requires ongoing improvement. While the initial model may provide reasonably accurate predictions, there is always room for enhancement. Factors such as the selection of algorithms, feature engineering, and hyperparameter tuning play a crucial role in determining the model’s performance. Continuous evaluation and updates to the model are necessary to adapt to new data and trends in heart failure risk factors. Moreover, establishing thresholds for risk categorization can be subjective, and determining the most clinically relevant thresholds is essential for ensuring that the predictions align with healthcare practices.

**Deployment Limitations**

When it comes to deploying the application in real-world settings, several challenges arise. Integrating real-time data from various sources, such as electronic health records (EHR) and wearable devices, poses technical difficulties. Ensuring seamless data flow and synchronization while maintaining patient privacy and compliance with healthcare regulations (such as HIPAA in the U.S.) is critical yet challenging. Additionally, healthcare providers may face resistance in adopting new technologies due to existing workflows and the need for training. Ensuring that the application is user-friendly and integrates smoothly into clinical practices is essential for widespread adoption and effectiveness.

**REFERENCES**

1. **Articles and Datasets Used**
   * UCI Machine Learning Repository: Heart Disease Dataset. Available at: [UCI Heart Disease Dataset](https://archive.ics.uci.edu/ml/datasets/heart+Disease)
   * “Heart Failure Prediction Using Machine Learning: A Review.” IEEE Access, 2020
   * ps://ijaem.net/issue\_dcp/Cardiovascular%20Disease%20Predic on%20System.pdf
2. **Documentation for Python and Streamlit**
   * o Python Documentation: Python.org
   * o Streamlit Documentation: Streamlit.io
   * o NumPy: https://numpy.org
   * o Matplotlib: https://matplotlib.org
   * o SK- Learn: https://scikit-learn.org