Requirement and Prototype Document

**Developing an Expert system for diagnosing Heart Disease 1.0**

**Part 1: Introduction, Purpose, and Goal of the Requirement Document**

**1.1 Introduction**

The Expert System for Diagnosing Heart Disease is a project aimed at developing a web-based expert system that will assist doctors, healthcare professionals and patients. Particularly doctors and physicians, in diagnosing a wide set of heart diseases. This document models the functional requirements with use cases and UML diagrams along with the non functional requirements. This requirements document specifies the needs and expectations for this project.

**1.2 Purpose**

The purpose of this document is to establish a clear and comprehensive understanding of the project's goals,objectives, scope, and requirements. It will be a guideline for our team to ensure the successful creation and implementation of the expert system for diagnosing heart diseases.

**1.3 Goal of the Requirement Document**

The goal of this requirement document is to provide a detailed, comprehensive and organized set of guidelines and specifications that will enable the development of an accurate, efficient, and user-friendly web-based expert system for diagnosing heart diseases. This system should fulfill the client's needs accurately and robustly and adhere to industry best practices and standards.

**Part 2: Requirement Glossary**

**2.1 Glossary**

* **Expert System**: A computer system that emulates the decision-making ability of a human expert in a particular domain.
* **Heart Disease**: A term for various conditions that affect the heart, including coronary artery disease, cardiac arrest, Angina and arrhythmias.
* **Patient Details**: Information about the patient like personal information, medical history, and diagnostic data.
* **Doctor/Physician**: The user of the system who uses patient data and interacts with the system to diagnose and treat heart diseases.
* **Diagnosis**: The determination of the specific heart disease affecting a patient based on their symptoms and medical history.
* **Parameters**: Medical, physiological, and clinical data used for diagnosing heart diseases.

**Part 3: User Requirements**

**Functional Requirements:**

**Patient Registration:** This functionality enables individuals to create an account within the system. Patients are required to provide their information, including their name, contact details and relevant medical history.

**User Login (Patients and Doctors):** Registered users, both patients and doctors can securely access the system by logging in using their credentials. This ensures that each user has personalized access to the platform.

**Symptom Input (Patients):** Patients have the capability to input information about their symptoms, medical history, and lifestyle factors. This data plays a role in the process and helps doctors gain a better understanding of the patient’s health condition.

**Initial Diagnosis (Machine Learning Module):** This is a computerized system that has an artificial intelligence (AI)/machine learning component used to analyze inputs such as past patient medical history and current symptoms. It uses medical knowledge and algorithms to produce an initial diagnosis or assessment. Initially, this diagnosis serves just as a baseline for evaluation which may be useful to physicians for other assessments.

**Consultation Request (Patient):** If patients feel that they need an additional checkup, or consultation, they can lodge a consultation request. This might have been due to initial diagnostic or other consideration.

**Consultation Scheduling (Doctor):** Physicians can view requests for the services of the physician or the appointment time by using the system. This ensures that patients find medical advice and treatment promptly, while appointment schedules are seamless.

**Consultation (Doctor):** Afterwards, the doctor can discuss with the patient what he has submitted into the system to generate the doctor’s first diagnosis. Some examples include discussing symptoms, gathering data, and performing exams face-to-face or virtually through a telemedical system.

**Additional Tests (Doctor):** On not ruling out any hypothesis the doctor can also advise and order some diagnostic tests e.g., blood tests, imaging or ECG’S so as they might be able to arrive at a conclusive diagnosis. With this, the physician gets more information on which to base his/her evaluations.

**Final Diagnosis (Doctor)**: A doctor will be able to make a final diagnosis using the data given by a patient, preliminary diagnoses as well as results of further tests. Based on this complete diagnosis, these are the recommendations for treatment.

**User Interface (Web Interface):** Through its user interface, patients and, indeed, physicians, will be able to communicate with the system. This is a straightforward setting that enables patients to input their information, check preliminary diagnoses, as well as demand consultations on an inquiry basis, while clinicians make use of the platform to peruse patient data, conduct inquiries, in addition to make scheduled arrangements.

**System Administration (Administrator):** Therefore, a system should have a system administrator who supervises and guards the whole system. This entails such tasks as account management for users, maintenance of data storage, secure and law-abiding information preservation, keeping watch over the system’s overall availability and functionality.

To meet the users’ requirements, we need to define their needs and create a use case diagram that outlines the participants and important functional requirements.

**Use Case Diagram:**

A use case diagram provides a representation of how users (actors) interact with the system. In our case we will focus on an expert system designed for diagnosing heart disease and examine how different actors interact with it.

**Actors:**

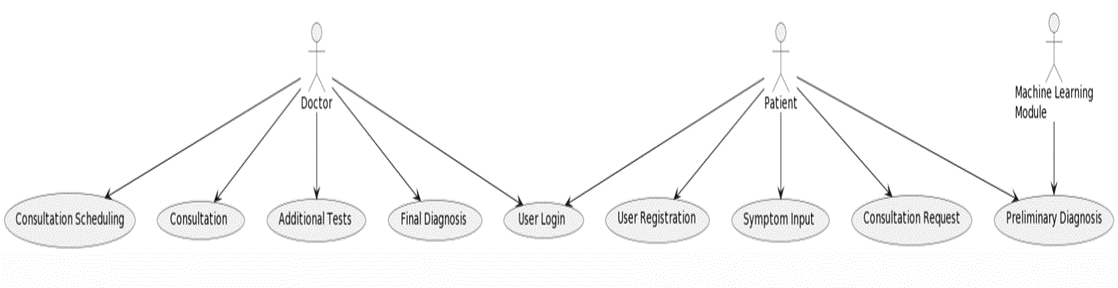
**1. Patient:** The individual seeking a diagnosis for heart disease.

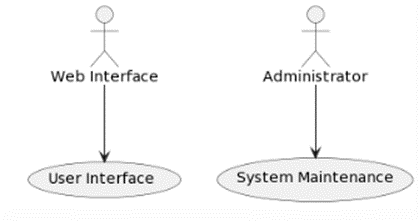
**2. Doctor:** A specialist who utilizes the system to review and validate diagnoses.

**3. Machine Learning Module:** This component of AI/ML is responsible for generating diagnoses based on input data.

**4. Web Interface:** The user interface utilized by physicians and patients to communicate with the system.

**5. Administrator:** The person responsible for managing user accounts and maintaining the system.



****

**Part 4: Prototype**

List of all the requirements for each page with screenshots

**1. Index Page:**

**Sign-Up Form:**

Fields: Username, password, email, and any other required information.

Validation for unique usernames and email addresses.

Submit button to create a new user account.

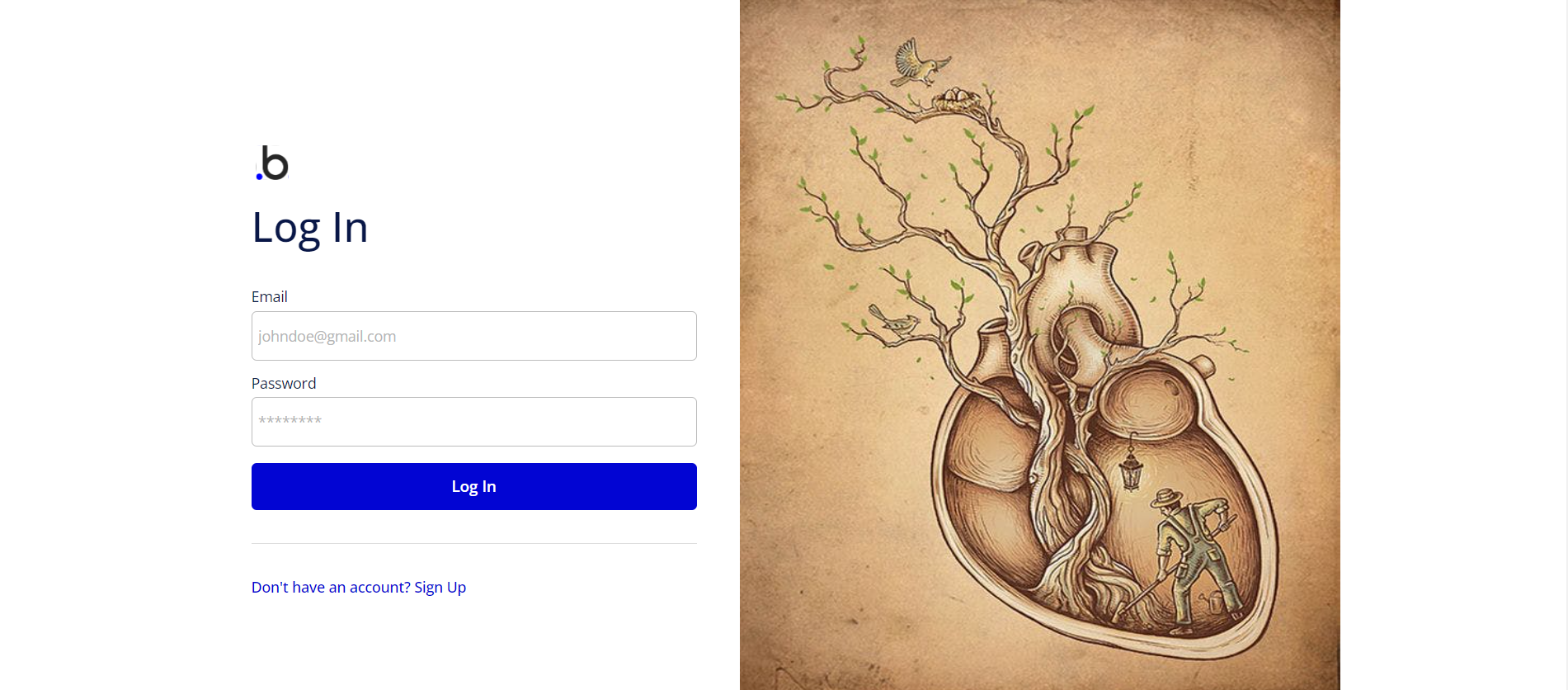
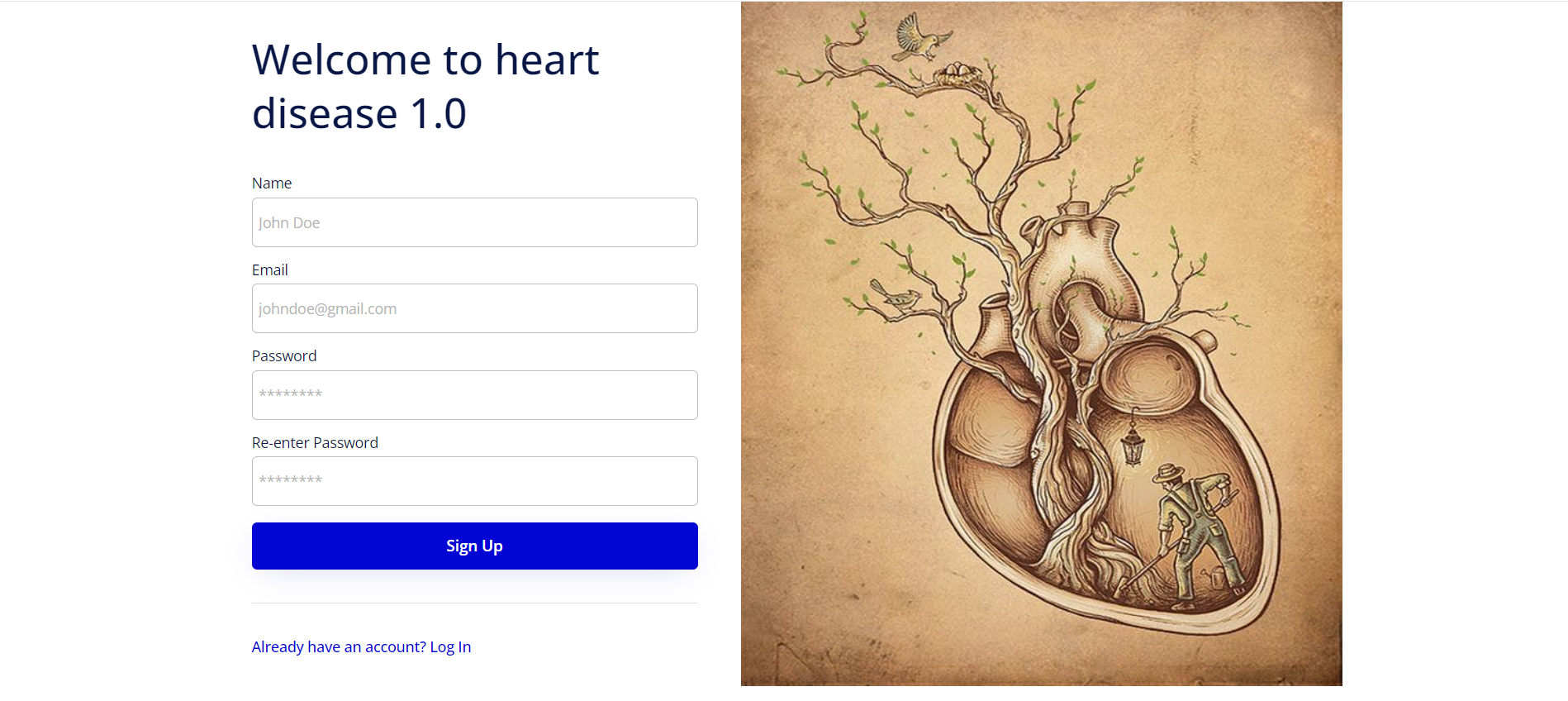
**Login Form:**

Fields: Username and password.

Validation for correct login credentials.

"Forgot password" option for account recovery.

Submit button for user login.



**2. After Login Page:**

**User Profile:**

Display user's name, profile picture, and basic information.

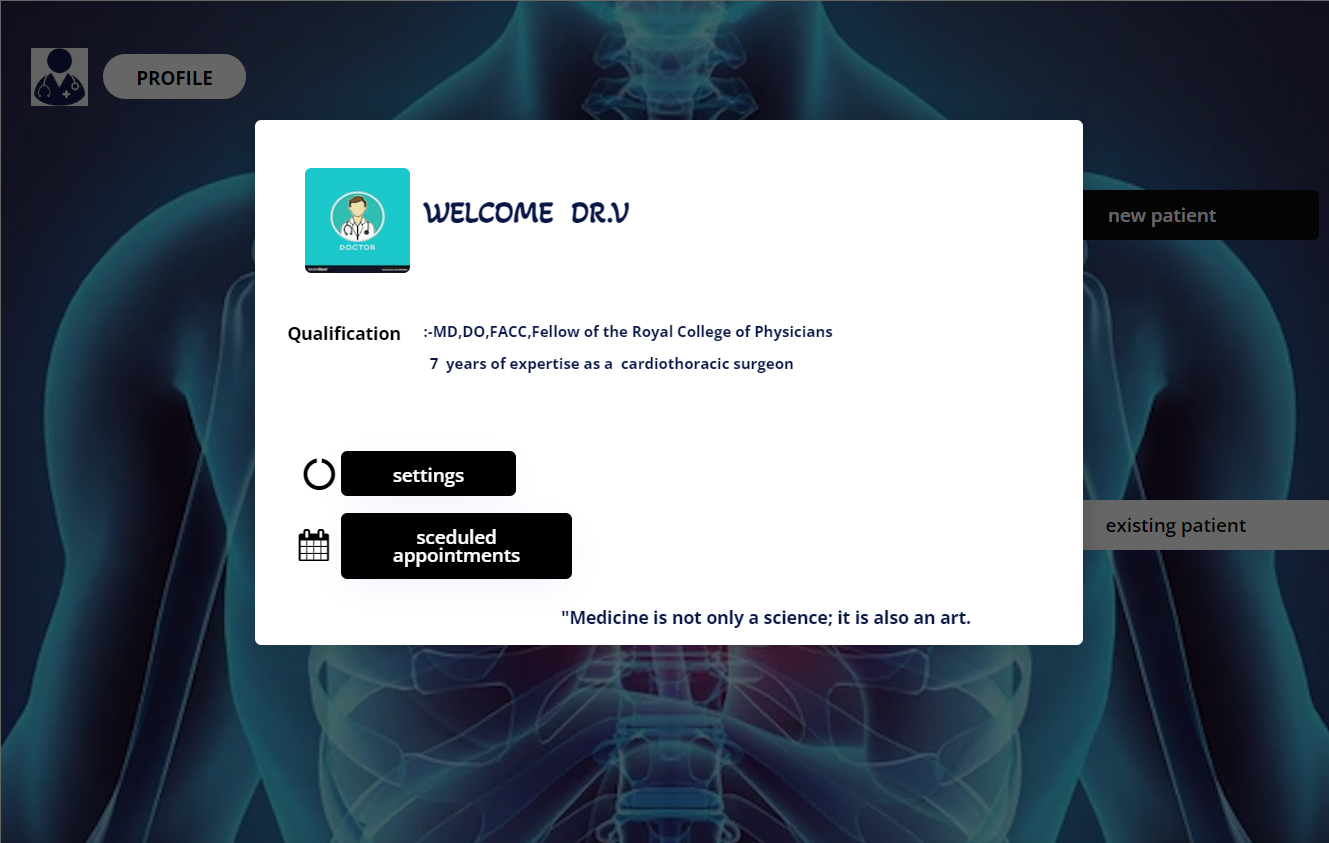
Option to edit profile details.

Logout button.

Buttons:

"New Customer": Redirects to the New Patient Page.

"Existing Customer": Redirects to the Existing Patient Page.



**3. New Patient Page:**

**Input Fields**:

Name: Text input.

Age: Numeric input.

Family Health History: Text input or a form for family medical history.

Symptoms: Text input or checkboxes for common symptoms.

Surgery History: Text input for any past surgeries.

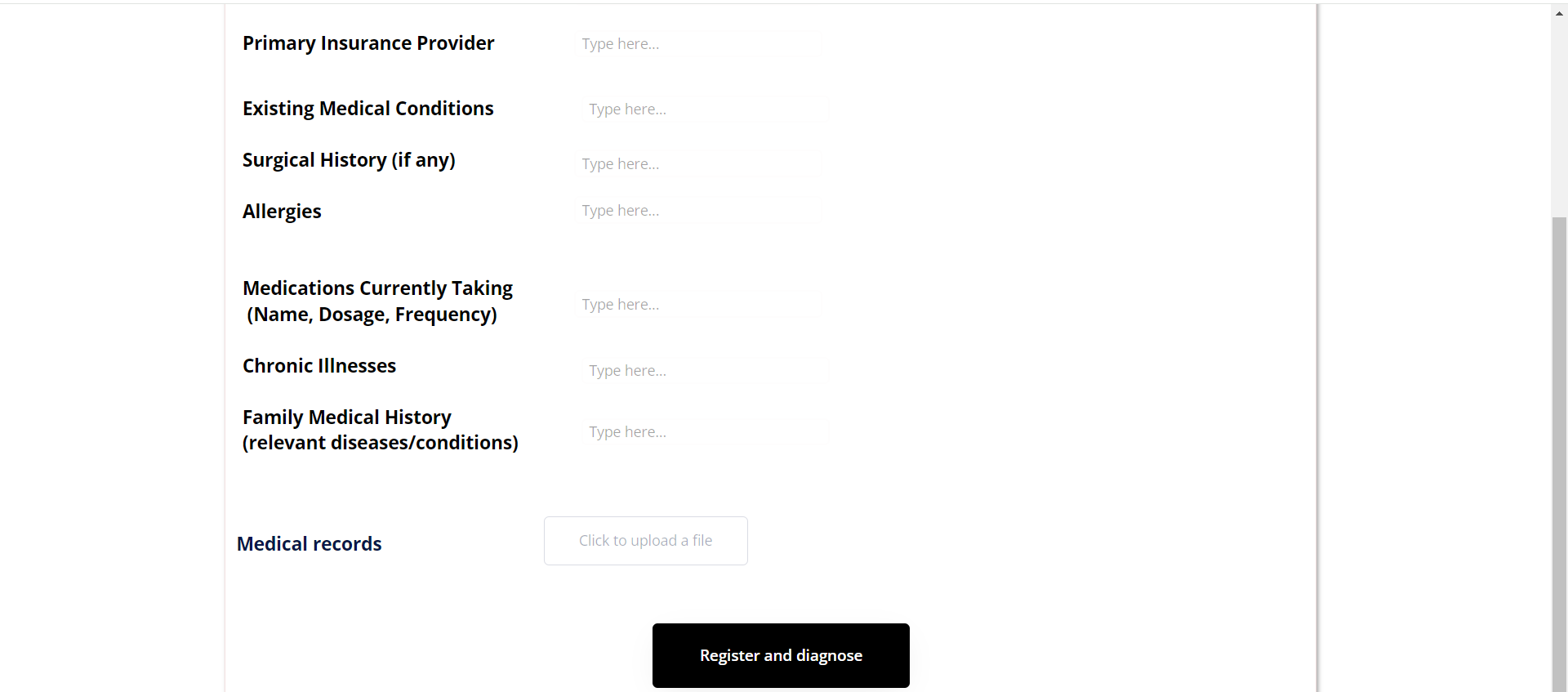
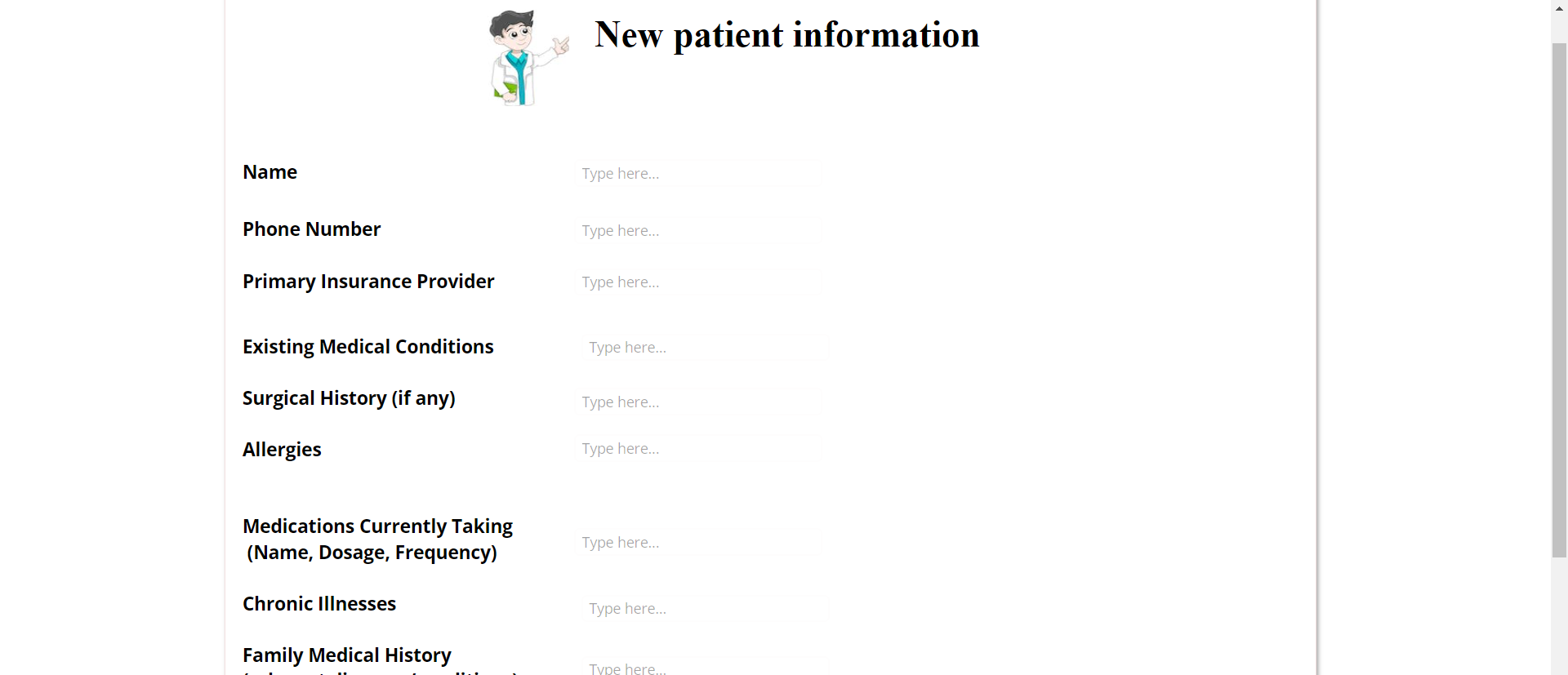
Medical Records: Option to upload and attach relevant medical documents (e.g., PDF, images).

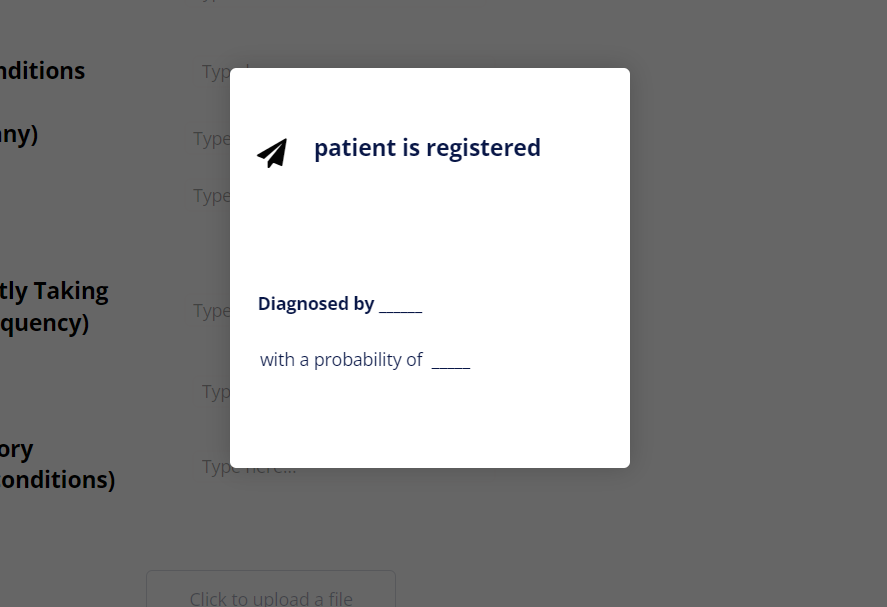
Diagnosis Button:

Performs the diagnosis based on the provided information.

Displays the diagnosis result, which can include the likelihood of heart disease and recommended actions.

Option to save the patient's record for future reference.





**4. Existing Patient Page:**

**Patient Information:**

Display the patient's name, age, family health history, symptoms, surgery history, and other relevant details.

Medical records and diagnostic history should be readily available.

Edit Button:

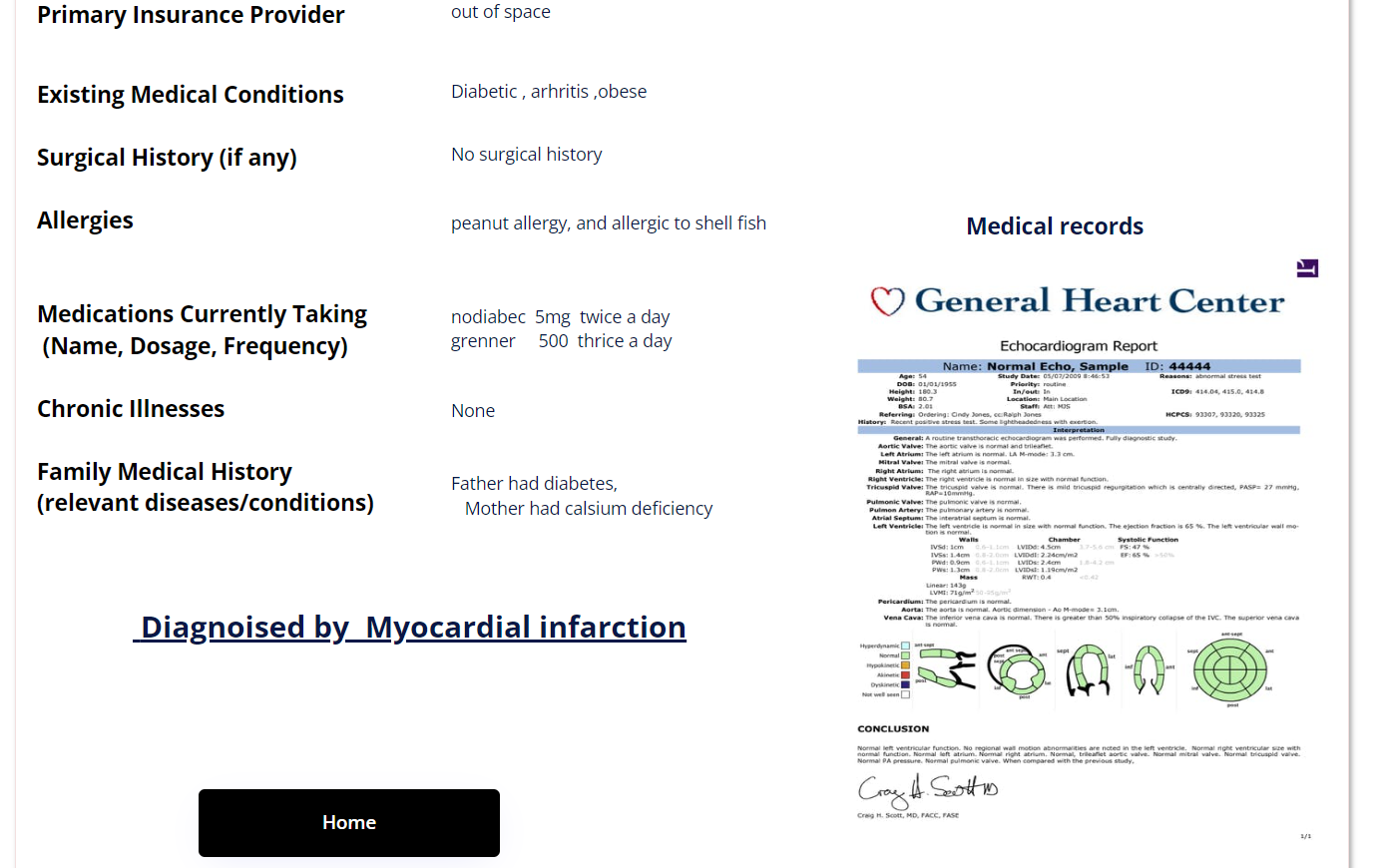
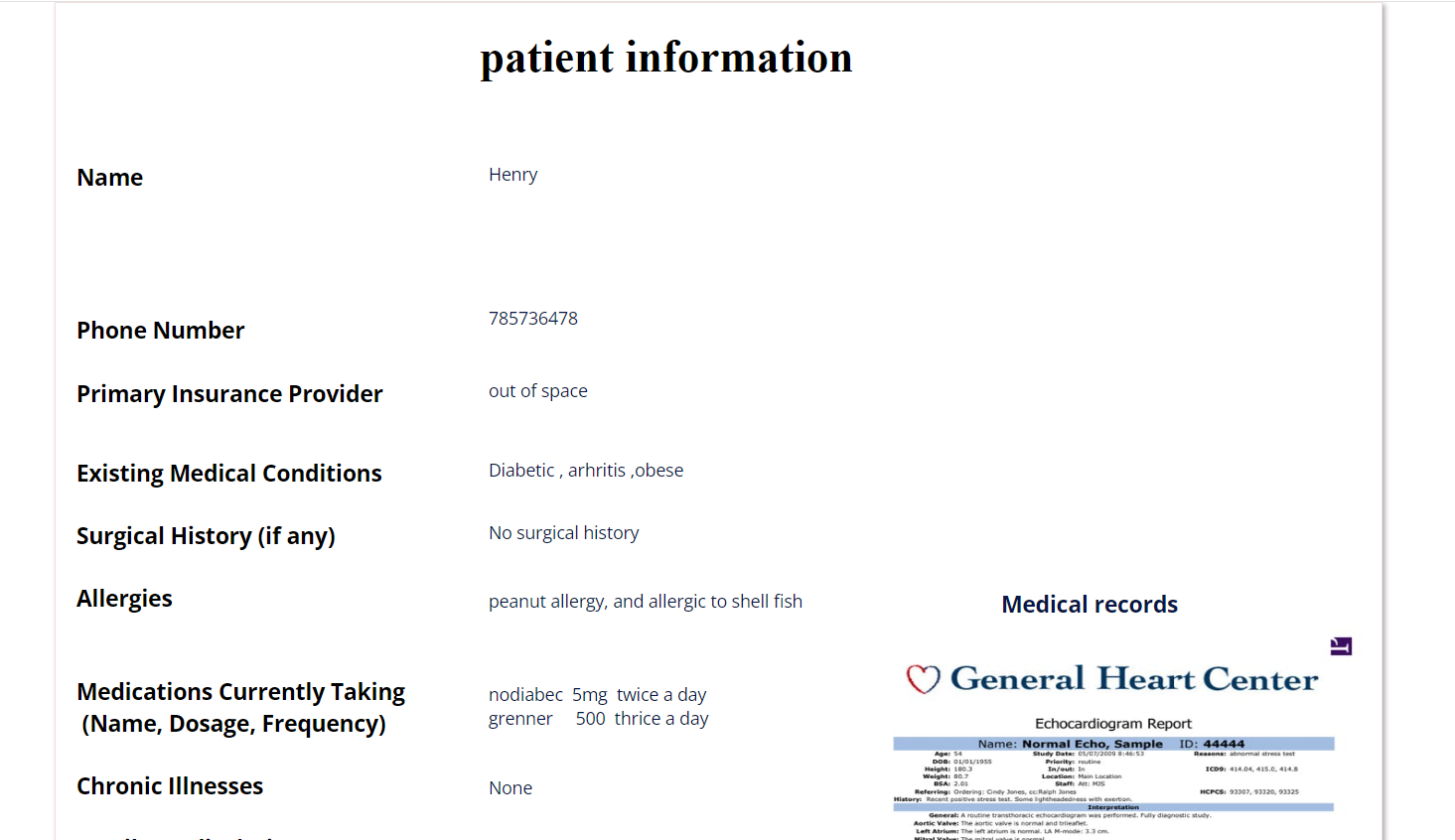
Allows updating patient information, if needed.

Previous Diagnoses:

A history of past diagnoses for reference.

Back to Profile Button:

Navigates back to the user's profile.



**Part 5: Non-Functional Requirements**

**5.1 Definition of Non-Functional Requirements**

Non-functional requirements specify the attributes of the system that are not directly related to its functionality. They describe how the system should perform rather than what it should do in terms of speed, reliability, security, and usability etc.

**5.2 Non-Functional Requirements**

**5.2.1 Performance**

* **Requirement**: The system should respond to user queries and provide diagnoses within 5 seconds of user interaction.
* **Justification**: Quick responses are crucial for the system's usability and to ensure efficient diagnosis.

**5.2.2 Reliability**

* **Requirement**: The system should have a 99% uptime, with minimal downtime for maintenance.
* **Justification**: Healthcare professionals and users rely on this system for critical patient assessments, making high reliability essential.

**5.2.3 Security**

* **Requirement**: All patient data should be encrypted and compliant with health data privacy regulations
* **Justification**: Protecting patient data and complying with legal requirements is a top priority.

**5.2.4 Usability**

* **Requirement**: The system should have an intuitive web user interface and provide clear, easy-to-understand diagnostic results.
* **Justification**: Healthcare professionals with varying levels of technical expertise should be able to use the system effectively.

**5.2.5 Scalability**

* **Requirement**: The system should be able to handle a growing number of patient data records and concurrent users.
* **Justification**: To accommodate the increasing usage and data volume over time.

**Part 6: Technical Requirements**

### **6.1.1 Database**

* Database system (e.g., SQL, NoSQL) or cloud storage (e.g., AWS S3, Google Cloud Storage) for storing raw and processed data.
* Considerations for data backup and security.

### **6.1.2 Data Pre-processing:**

* Tools to handle missing values, outliers, and normalization (e.g., Pandas, NumPy).
* Feature selection and engineering capabilities.

### **6.1.3 Development Environment:**

* An integrated development environment (IDE): Visual Studio Code.
* Cloud platforms for scalability, if required, such as Google Colab for GPU/TPU support.

### **6.1.4 Model Training and Evaluation:**

* Hardware: Access to high-performance GPUs or TPUs for faster model training, as deep learning is involved.
* Software: ML libraries like Scikit-learn, TensorFlow/ PyTorch.
* Model evaluation metrics like accuracy, precision, recall, ROC, and AUC.

### **6.1.5 Model Deployment:**

* Web-server or cloud platform for deployment (e.g., Flask, Django, AWS Lambda, Google Cloud Functions).
* Tools for model serialization and version control: Joblib.

### **6.1.6 Monitoring & Maintenance:**

* Tools for monitoring the model's performance in real-time.
* Logging tools for error tracking and debugging.

### **6.1.7 User Interface:**

This expert system will be developed as a full-fledged web application. The system must be accessible via web browser, making it platform-independent and accessible from any location with an internet connection.

## **2. Choosing AI/ML Technologies:**

### **6.2.1 Data Analysis & Visualization:**

* Pandas: For data manipulation.
* Matplotlib and Seaborn: For data visualization.
* Scikit-learn: For initial data analysis and simple machine learning models.

### **6.2.2 Machine Learning Frameworks:**

* Scikit-learn: For traditional algorithms like Random Forest, SVM, Gradient Boosting, etc.
* TensorFlow and Keras: If you're leaning towards deep learning or need a more customizable environment.
* PyTorch: An alternative to TensorFlow for deep learning.

### **6.2.3 Model Deployment & Serving:**

* Flask or Django: For building a web API.
* Streamlit or Dash: For creating interactive web applications with less effort.
* Docker: For containerization to ensure consistency in different environments.
* Cloud services like AWS SageMaker, Google AI Platform, or Azure Machine Learning for scalable deployments.

### **6.2.4 Model Optimization & AutoML:**

* Optuna or Hyperopt: For hyperparameter optimization.
* H2O.ai or Google Cloud AutoML: If you want to try automated machine learning solutions.
* Version Control & Collaboration:

**6.2.5 Version Control:**

* Git: For version control.
* GitHub or GitLab: For code repositories and collaboration.

**6.2.6 Continuous Integration & Continuous Deployment (CI/CD):**

Tools like Jenkins, GitHub Actions, or GitLab CI for automating testing and deployment processes.

**6.2.7 Databases:**

UCI Heart Disease Dataset:<https://archive.ics.uci.edu/dataset/45/heart+disease>

* Originates from the University of California, Irvine.
* It has data from four major heart disease research institutions.
* Contains 76 attributes, but many researchers typically use a subset of 14.
* It's one of the most commonly used datasets for heart disease prediction.

Framingham Heart Study : <https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset>

* A long-term, ongoing cardiovascular study on residents of the city of Framingham, Massachusetts.
* Provides a wealth of data related to heart diseases and has been pivotal in the identification of risk factors for cardiovascular diseases.

Cleveland Heart Disease Dataset:

* Part of the UCI dataset but specifically from the Cleveland Clinic Foundation.
* Often used in its own right because of the quality of data it offers.

STATLOG (Heart) Dataset: <https://archive.ics.uci.edu/dataset/145/statlog+heart>

* Also from the UCI repository.
* Contains data like age, sex, chest pain type, resting blood pressure, etc., for predicting the absence or presence of heart disease.

Cardiovascular Disease Dataset: <https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset>

* Available on Kaggle.
* Consists of 70,000 records of patients with 11 features each.
* Features include age, gender, systolic blood pressure, cholesterol, etc.

MIT-BIH Arrhythmia Dataset: <https://physionet.org/content/mitdb/1.0.0/>

* Specifically for detecting arrhythmias, not general heart disease, but relevant for cardiovascular health.
* Hosted on PhysioNet and has 48 half-hour excerpts of two-channel ambulatory ECG recordings.

BIDMC Congestive Heart Failure Database: <https://physionet.org/content/chfdb/1.0.0/>

* Another dataset on PhysioNet.
* Contains data specifically related to congestive heart failure.

PTB Diagnostic ECG Database:<https://physionet.org/content/ptbdb/1.0.0/>

* For diagnostic electrocardiography.
* Has 549 records from 290 subjects.
* Each record includes a 15 signal, 12 lead ECG.

**Part 7: Project Risks**

**7.1 Identified Risks**

* **Data Security Breach**: Risk of unauthorized access to patient data, leading to privacy violations and legal consequences.
* **Inaccurate Diagnoses**: The system may provide incorrect diagnoses, leading to incorrect treatment recommendations and patient harm.
* **Integration Challenges**: Difficulty integrating the system with existing healthcare IT infrastructure and electronic health records (EHR) systems.
* **Regulatory Compliance**: Risk of non-compliance with healthcare data regulations, resulting in fines and legal issues.
* **Scalability Issues**: As the user base grows, the system may struggle to handle the increasing load and data volume.

**7.2 Risk Mitigation**

* **Data Security**: Implement strong encryption and access control measures, regularly update security protocols, and conduct security audits.
* **Accuracy Improvement**: Continuously update the system with the latest medical research and involve medical experts in refining the diagnosis algorithms.
* **Integration Planning**: Engage IT experts and healthcare providers to plan and execute a seamless integration process.
* **Regulatory Compliance**: Regularly review and update the system to ensure compliance with healthcare data regulations.
* **Scalability Planning**: Develop scalable architecture and monitoring mechanisms to identify and address scalability issues in real-time.

**Part 8: Project Management**

Phase 1: Project Initiation (By September 20th)

· Define Project Scope and Objectives

· Identify Client and Team Members

· Set Up Project Environment and Tools

· Develop a Project Charter

· Conduct Kick-off Meeting

Phase 2: Requirements and Planning (By October 20th)

· Gather User Requirements

· Create a Use Case Diagram

· Define Functional and Non-functional Requirements

· Identify Technical Requirements and Choose AI/ML Technologies

· Develop a High-Level Project Plan

· Allocate Team members for particular task

· Develop a Risk Management Plan

· Review and Approve Project Plan

Phase 3: Data Collection and Preparation (By October 27th)

· Identify Data Sources

· Collect and Curate Medical Data

· Preprocess and Clean Data

· Implement Data Security and Privacy Measures

· Ensure Data Quality Assurance

Phase 4: AI/ML Model Development (By November 3rd)

· Design the AI/ML Model Architecture

· Perform Data Splitting (Training, Validation, Testing)

· Train the AI/ML Model

· Evaluate and Fine-Tune the Model

· Implement Explainable AI/ML Features

· Develop the User Interface for Data Input

Phase 5: Integration and Testing (By November 10th)

· Integrate the AI/ML Model with the User Interface

· Conduct Functional Testing

· Implement Security Measures

· Perform Integration Testing

· Set Up Automated Testing

· Conduct User Acceptance Testing

Phase 6: Documentation (By November 17th)

· Develop User and Technical Documentation

· · Review and Approve Documentation

Phase 7: Deployment (By November 20th)

· Deploy the Expert System

· Gather Feedback and Make Necessary Revisions

Phase 8: System Validation (By November 24th)

· Conduct System Validation and Performance Testing

· Address Performance Issues

Phase 10: Project Closure (By December 1)

· Conduct Post-Implementation Review

· Prepare Final Project Report

This project schedule is a general outline, and we may adjust and tweak it based on the specific needs and constraints of our project. In Agile methodology it is important to continuously review and update the schedule as the project progresses. We as a team and client will involve in the planning and execution phases to ensure the successful delivery of the project.