PURDUE UNIVERSITY | FORT WAYNE

Developing an Expert system for diagnosing Heart Disease 1.0

Final Project Documentation ACS560: Software Engineering

Team Members: Group 5

1.Navyaprabha Rajappa (Team Lead)

2.Atharva Atre

3.Sai Vikram Gurram

4.Chaitanya Nalluru

Table of content:

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.

Purpose:-

The purpose of designing an expert system for diagnosing heart disease (Heart Disease 1.0) is to provide a technology-based solution that can assist healthcare providers and patients in making early, accurate, and consistent diagnosis of heart disease. This system aims to reduce human errors, offer access to specialized medical knowledge, provide timely responses, support healthcare professionals, improve cost efficiency, educate patients, enable remote healthcare, and efficiently manage patient data, ultimately enhancing the quality of care and patient outcomes.

Goal:-

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.

Project 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.

Project Constraints:

Project constraints in the context of developing an Expert System for Diagnosing Heart Disease can include various limitations and factors that may impact the project's scope, schedule, and resources. Here are some potential project constraints:

Time Constraints:

- Deadline: The project may have a specific timeframe for completion, influenced by factors like regulatory requirements, client expectations, or industry standards.
- Development Time: The time required for developing, testing, and deploying the expert system may be limited.

Technical Constraints:

- Technology Stack: The project may be constrained by the choice of technologies, languages, or frameworks mandated by the client, organization, or existing infrastructure.
- Integration Limitations: Compatibility issues with existing healthcare systems or limitations in data exchange protocols may impose constraints.

Scope Constraints:

- Feature Set: The project scope may be limited by the defined features and functionalities, potentially affecting the comprehensiveness of the expert system.
- Data Availability: The quality and availability of medical data for training machine learning models or conducting accurate diagnoses may be a constraint.

Use case & Class Model:-

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.

User Requirements:-

A use case diagram visually represents the interactions between users (actors) and the system. For a heart disease diagnosis expert system, we will identify the actors and their interactions with the system.

Actors:

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

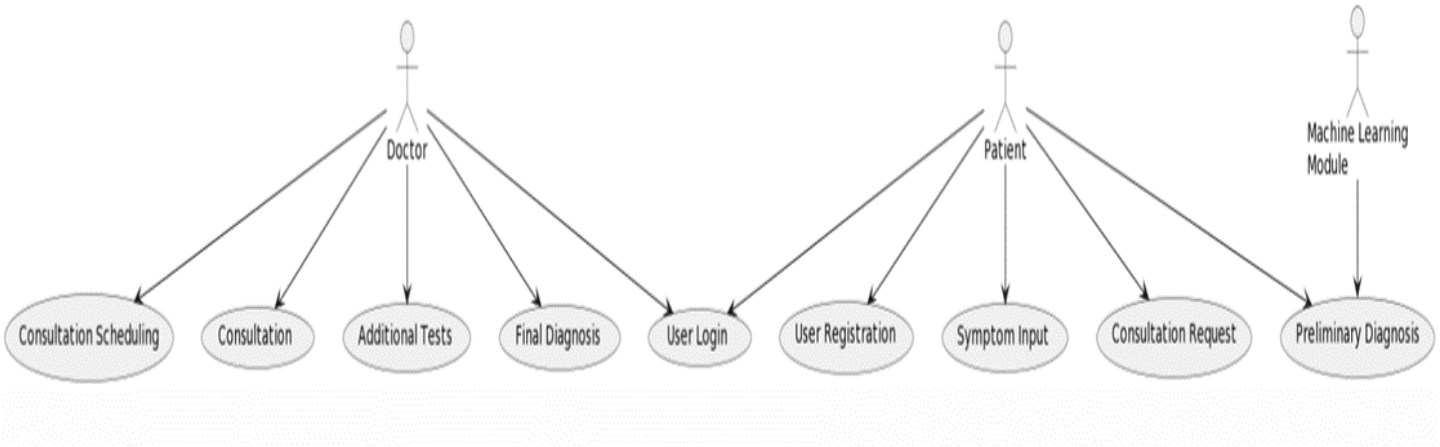
2. Doctor: A medical professional who interacts with the system to review and confirm diagnoses.

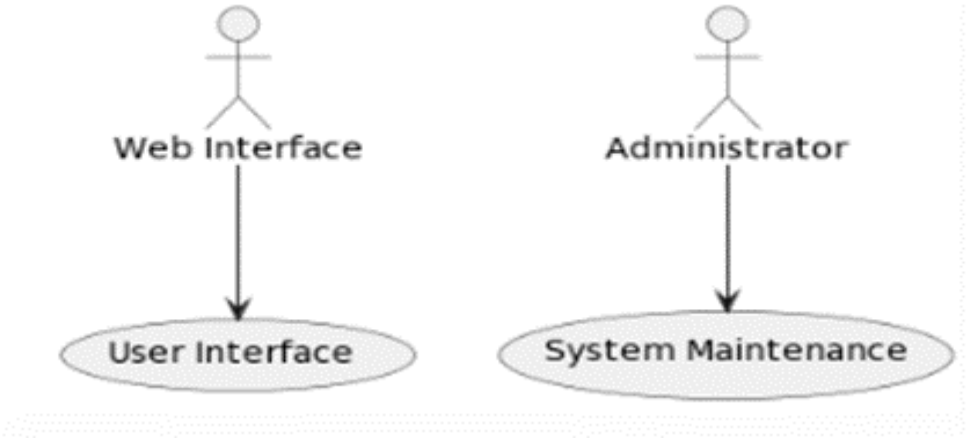
3. Machine Learning Module: Represents the AI/ML component responsible for making preliminary diagnoses based on input.

4. Web Interface: The user interface through which patients and doctors interact with the system.

5. Administrator: The person responsible for system maintenance and user management

Use Case Diagram:-





Design & Implementation Modules:-

In the context of the Expert System for Diagnosing Heart Disease, the design and implementation involve several important models to ensure effective development and testing.

1. Class Model:

The Class Model provides a blueprint for the system's structure by defining classes, their attributes, and methods. It captures the relationships between different entities in the system, such as patients, doctors, and the machine learning module. Attributes and methods specified in the class model correspond to the functionalities identified in the requirements. The class diagram depicts the static structure of the system, showcasing how different components collaborate and interact.

2. Use Case-Sequence Model:

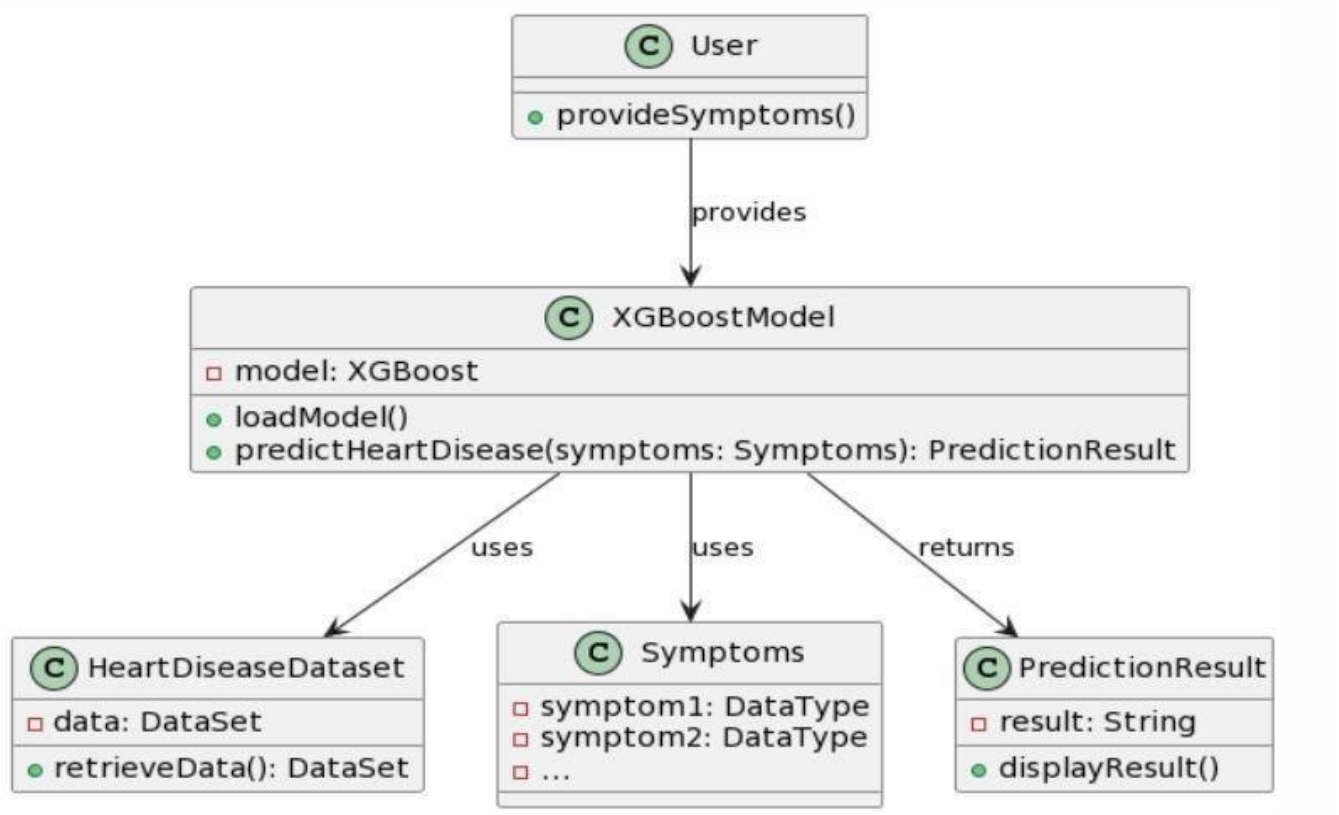
The Use Case-Sequence Model illustrates the dynamic behavior of the system by showcasing the sequence of interactions between different actors and the system. In the context of diagnosing heart disease, use cases like patient registration, consultation, and diagnosis are sequenced to demonstrate the flow of activities. This model helps in understanding how the system responds to user actions over time, providing insights into the user-system interactions during various scenarios.

3. State Model:

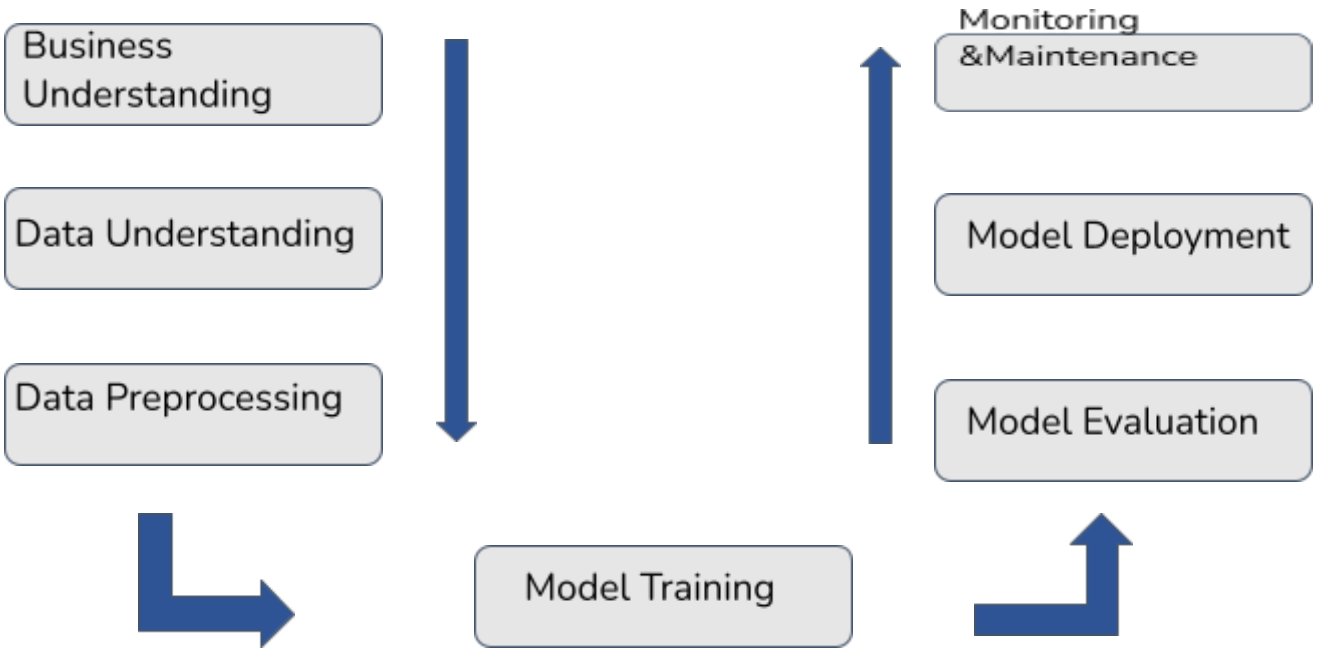
The State Model represents the different states that an object or a system can transition through during its lifecycle. In the case of the heart disease expert system, the state model could be applied to represent the various states of a patient's consultation process—starting from registration, progressing through diagnosis stages, to the final treatment recommendations. This model aids in visualizing and understanding the system's behavior as it undergoes different states in response to user actions.

These models collectively contribute to a comprehensive design and implementation strategy, ensuring that the system not only adheres to functional requirements but also provides a structured, efficient, and user-friendly experience. The class model defines the static structure, the use case-sequence model captures dynamic interactions, and the state model visualizes the system's evolving states, collectively shaping the foundation for the development and testing phases

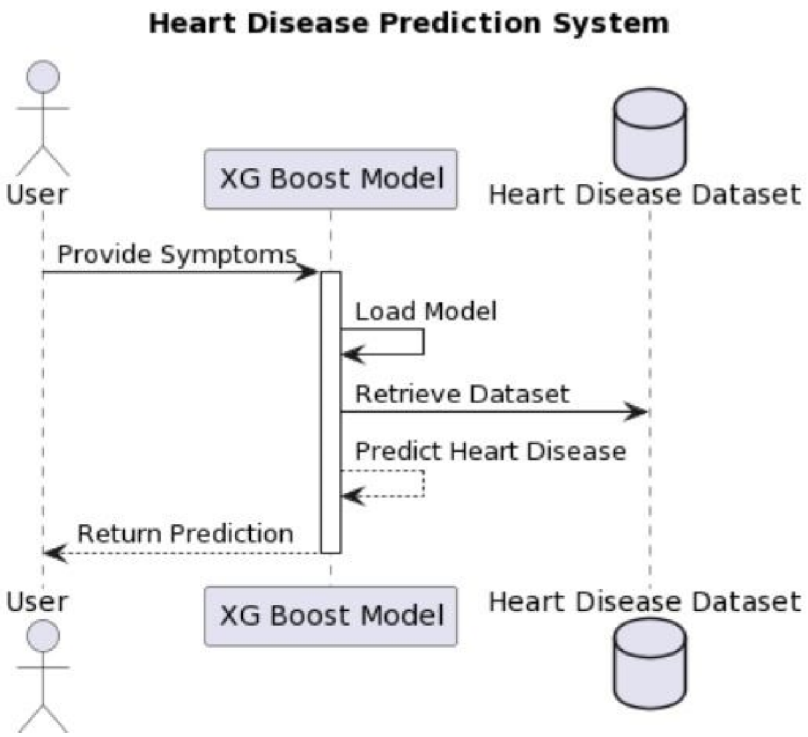
Class Diagram:-



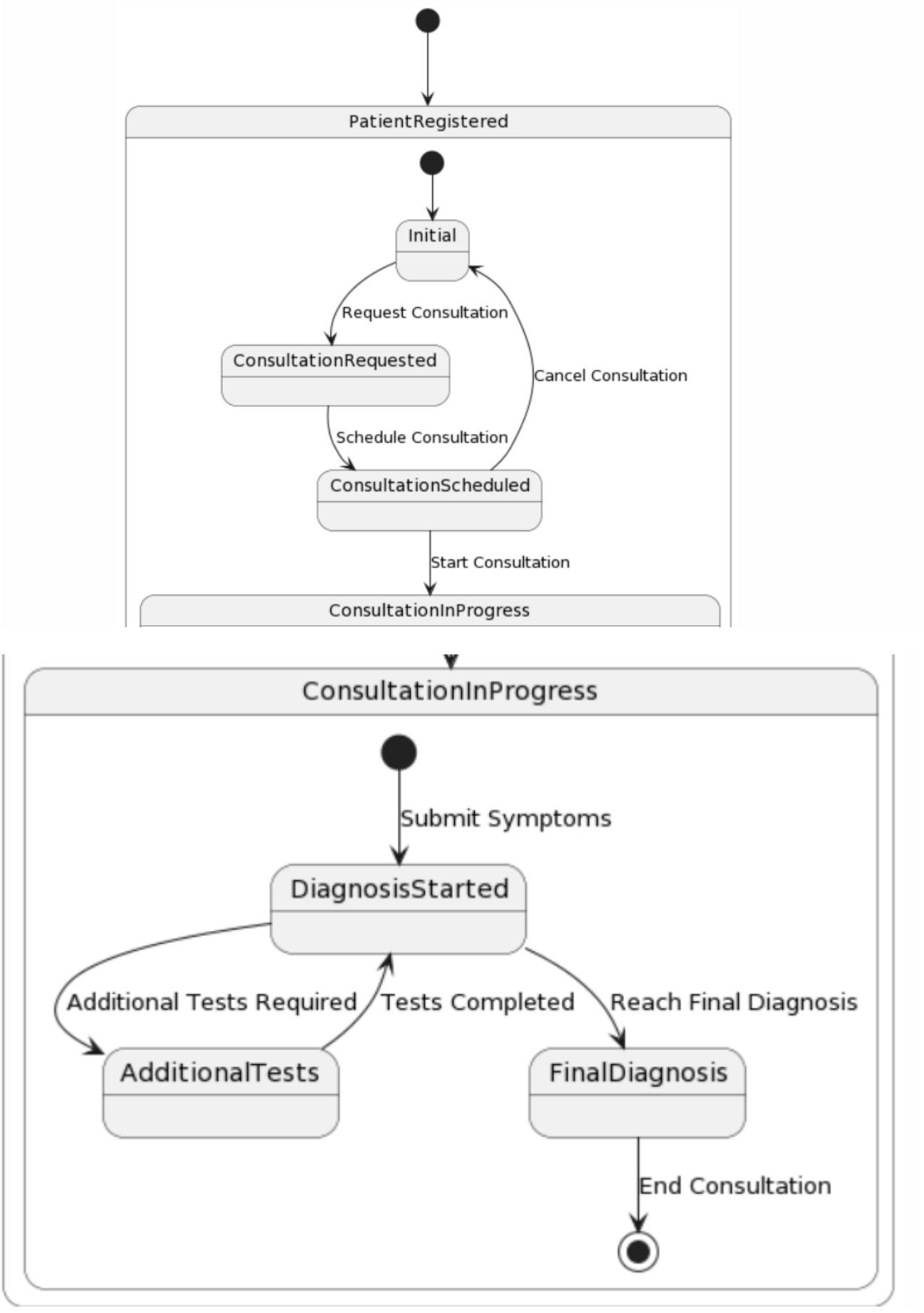
project design:-



Sequence model:-



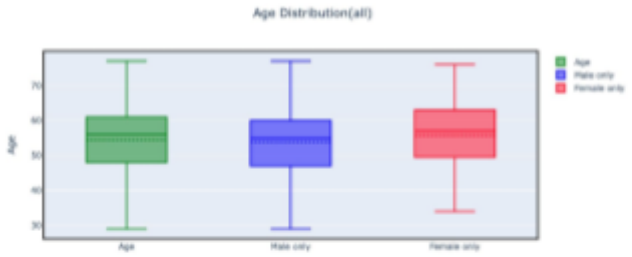
State Diagram

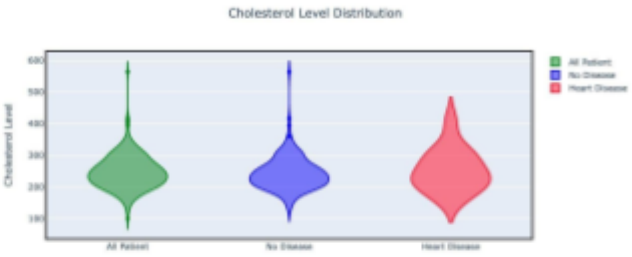

PatientRegistered
Initial
Request Consultation
ConsultationRequested
Cancel Consultation
Schedule Consultation
ConsultationScheduled
Start Consultation
ConsultationInProgress
ConsultationInProgress
Submit Symptoms
DiagnosisStarted
Additional Tests Required
Tests Completed
Reach Final Diagnosis
AdditionalTests
FinalDiagnosis
End Consultation

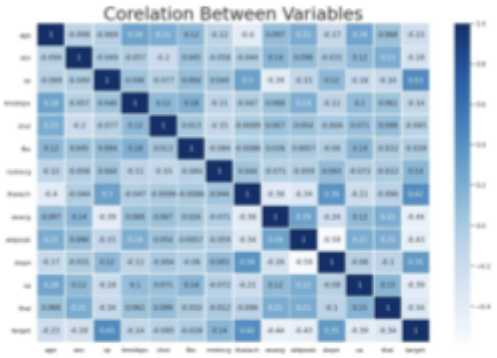
Model – Data:-

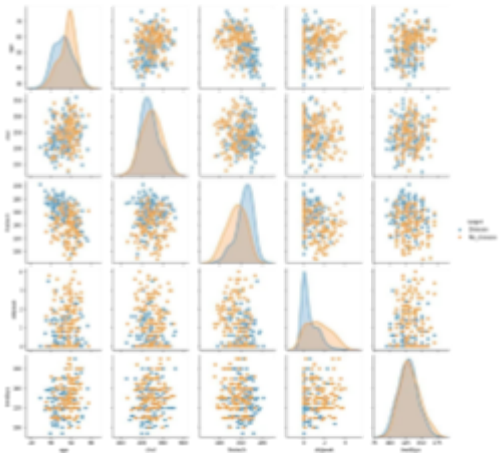
- Data is taken from actual ~1000 heart disease patients.
  Link to dataset - https://archive.ics.uci.edu/dataset/45/heart+disease
- All sensitive patient information like patient actual name, and social security numbers are removed from the dataset, and replaced with dummy values.
- This database contains 76 attributes, but after careful PCA analysis as well as our use case, we decided to go with a subset of 14 features.

Exploratory Data Analysis :-



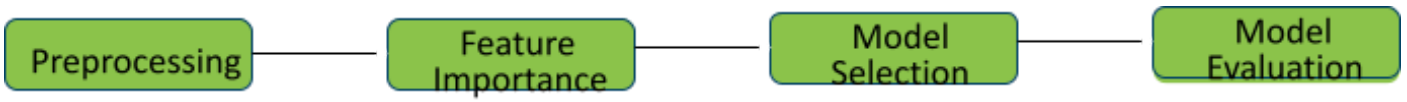






Model Selection

- We worked on a lot of data preprocessing, followed by feature ranking, feature selection and experimented with different models for our use case.



Model Name	Precision	Recall	F-1	Accuracy
Logistic Regression	0.83	0.85	0.9	0.9
SVM	0.7	0.76	0.73	0.78
Random Forest	0.88	0.85	0.87	0.85
XGBoost	0.92	0.95	0.93	0.92

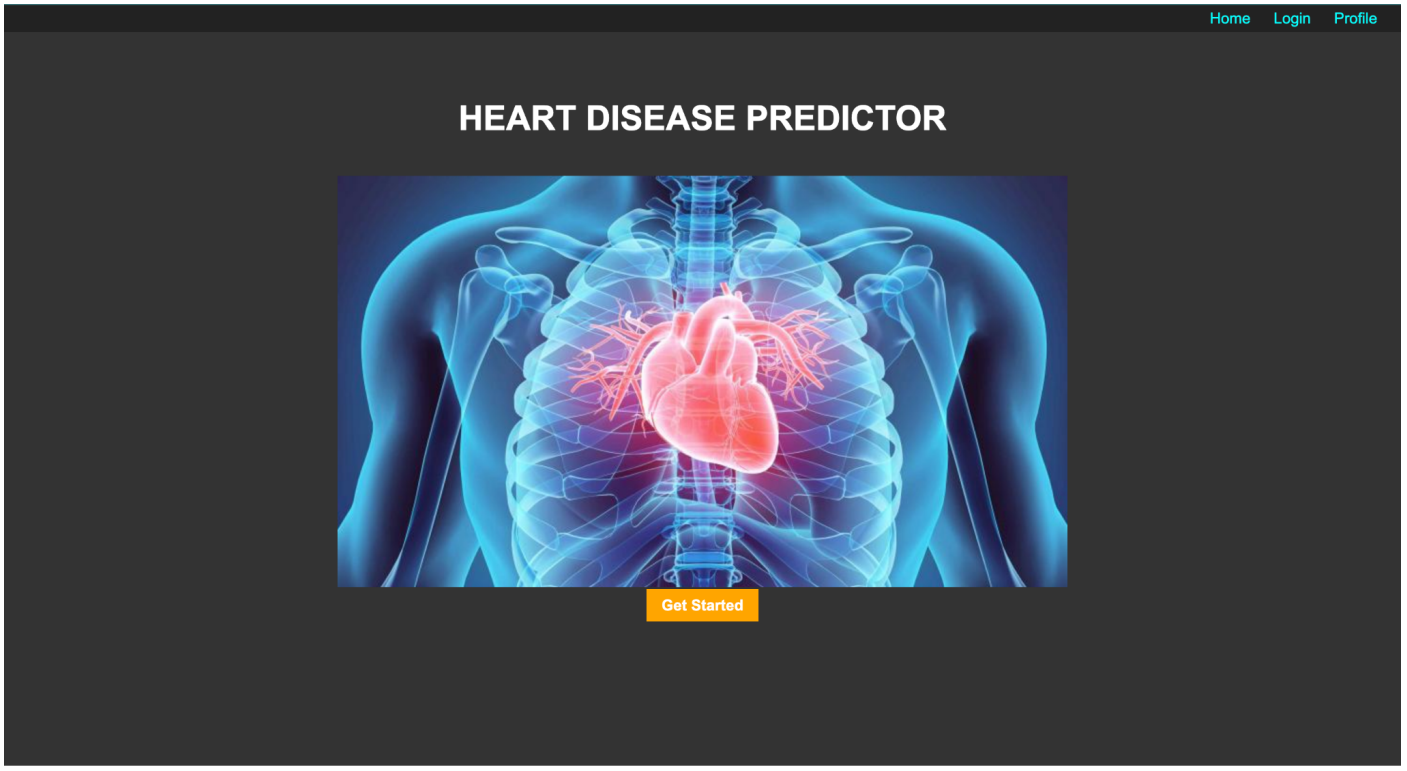
As we have received the best precision and accuracy in the XGboost model we have used this model for our project as it can be the one which provides the best diagnosis for patient so that our project is efficient to its maximum extent and as we are dealing with health related data we should be more careful with the accuracy and precision as any error in this can affect lives of people.

Back-end and Integration:-

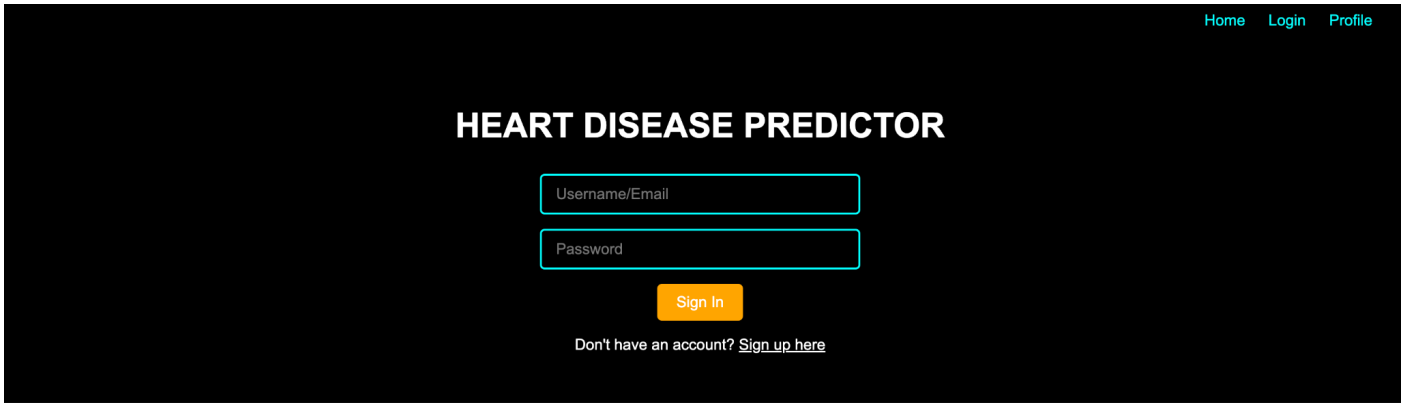
Flask API serves as the backend for our project, and when a user interacts with the front end by clicking on the predict button, the Flask API processes the request, runs a predictive model, and sends back the prediction percentage to be displayed on the front end. Here's a brief overview of the flow:

- User Interaction:
- User clicks the predict button on the front-end interface.
- Front-end Request:
- Front-end sends a request to the Flask API, indicating that a prediction is requested.
- Flask API Processing:
- Flask API receives the request.
- It triggers the backend process to run the predictive model using the provided input or parameters.

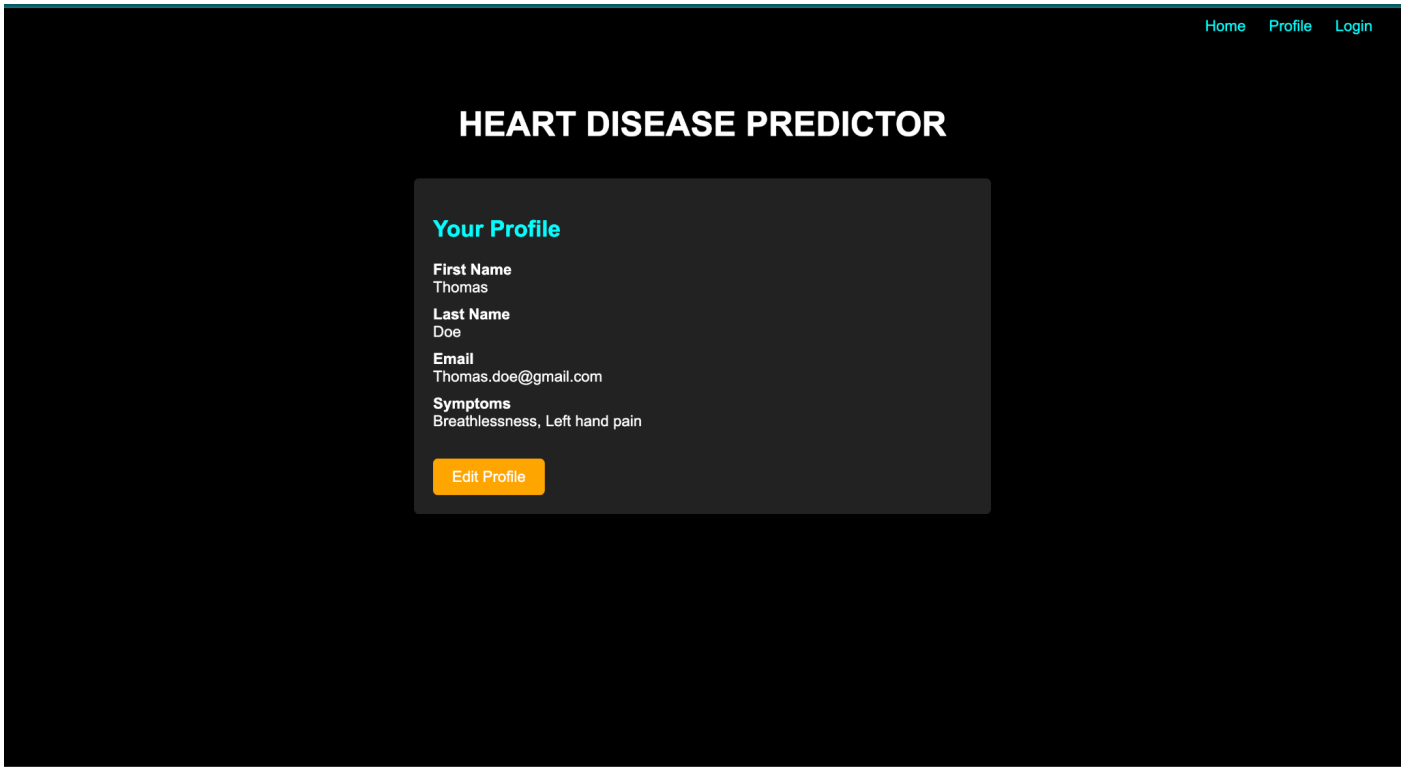
Demo ScreenShots :



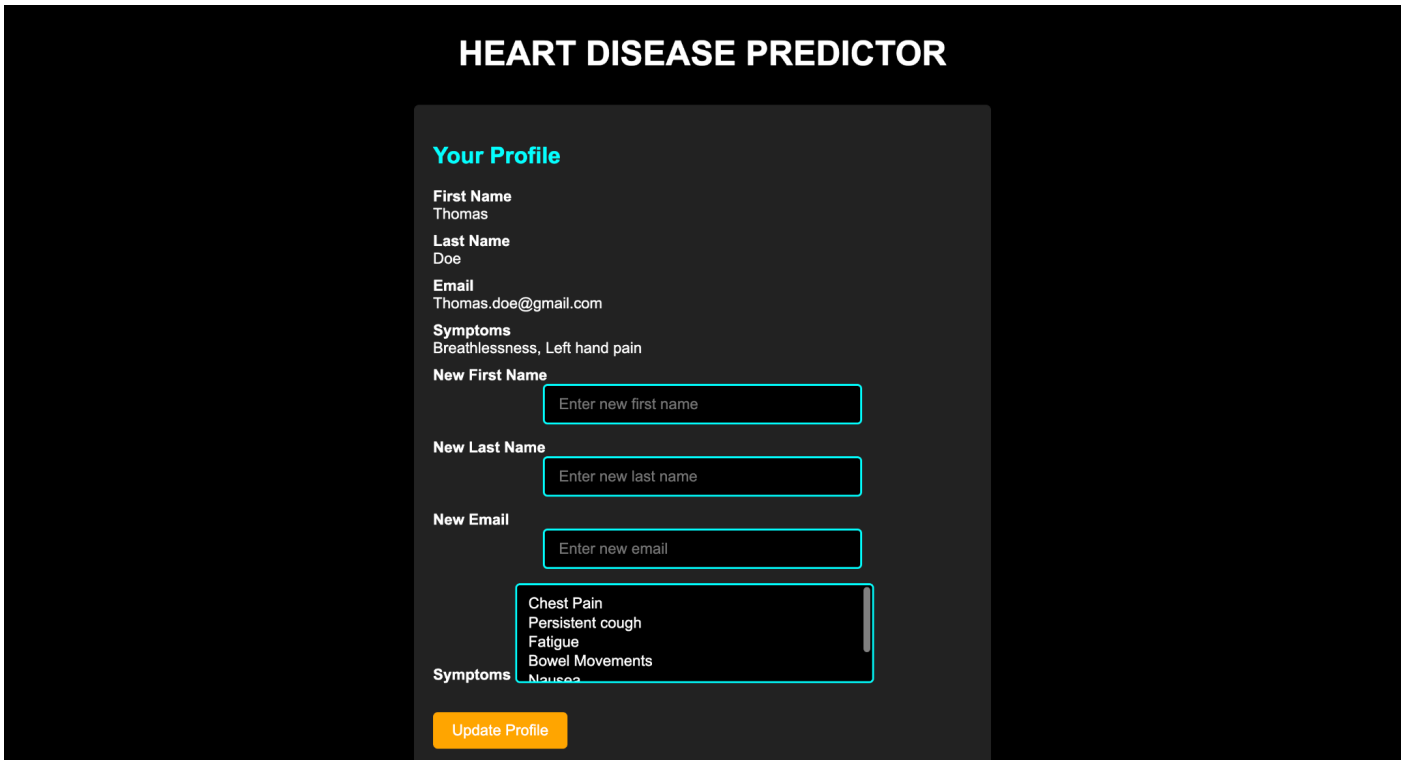
This is our home page. We can go to the login/signup page by clicking on the “Get Started” button.



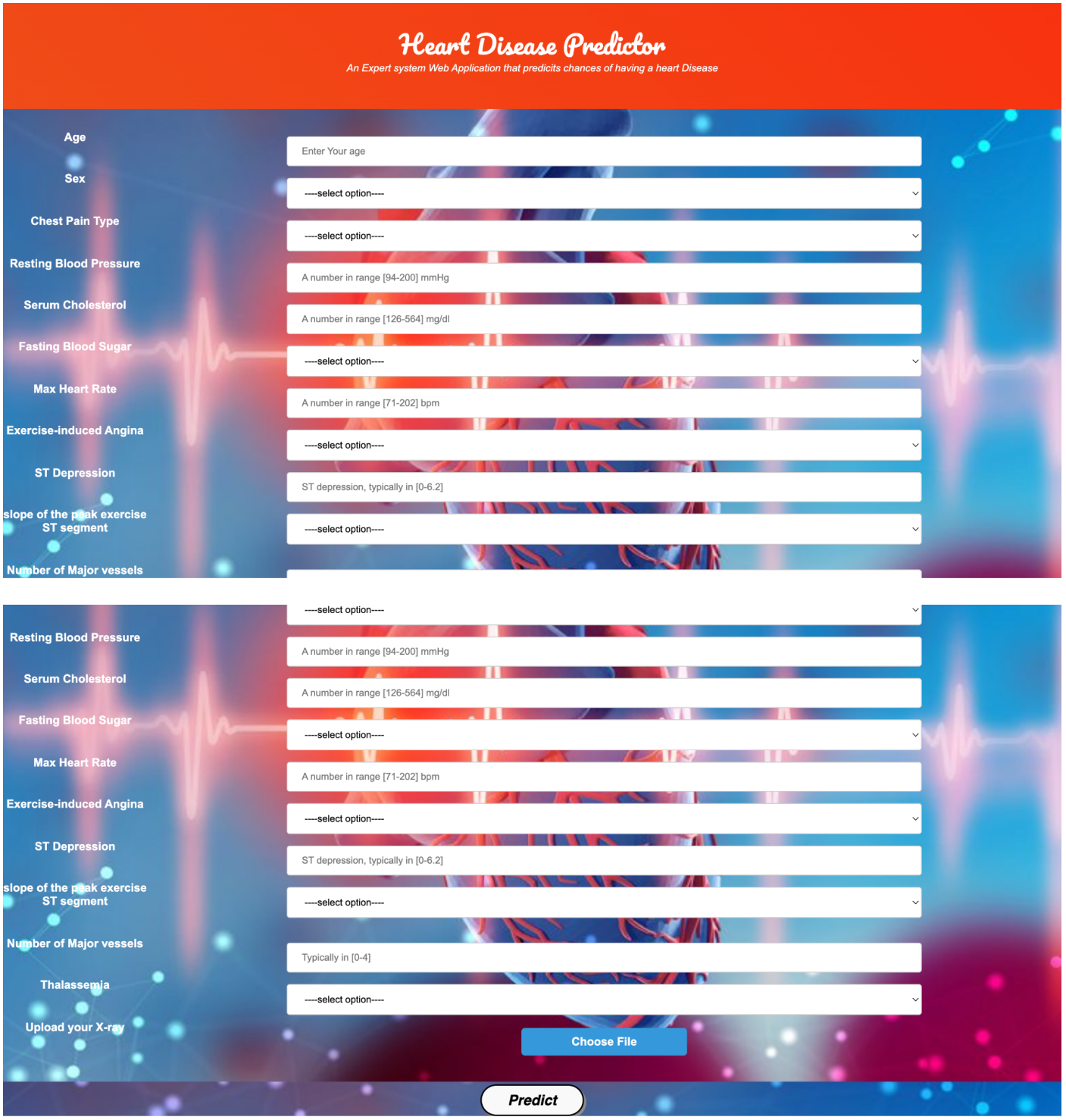
This is our portal for entering username/email and password. If the user is not registered, they can signup with us



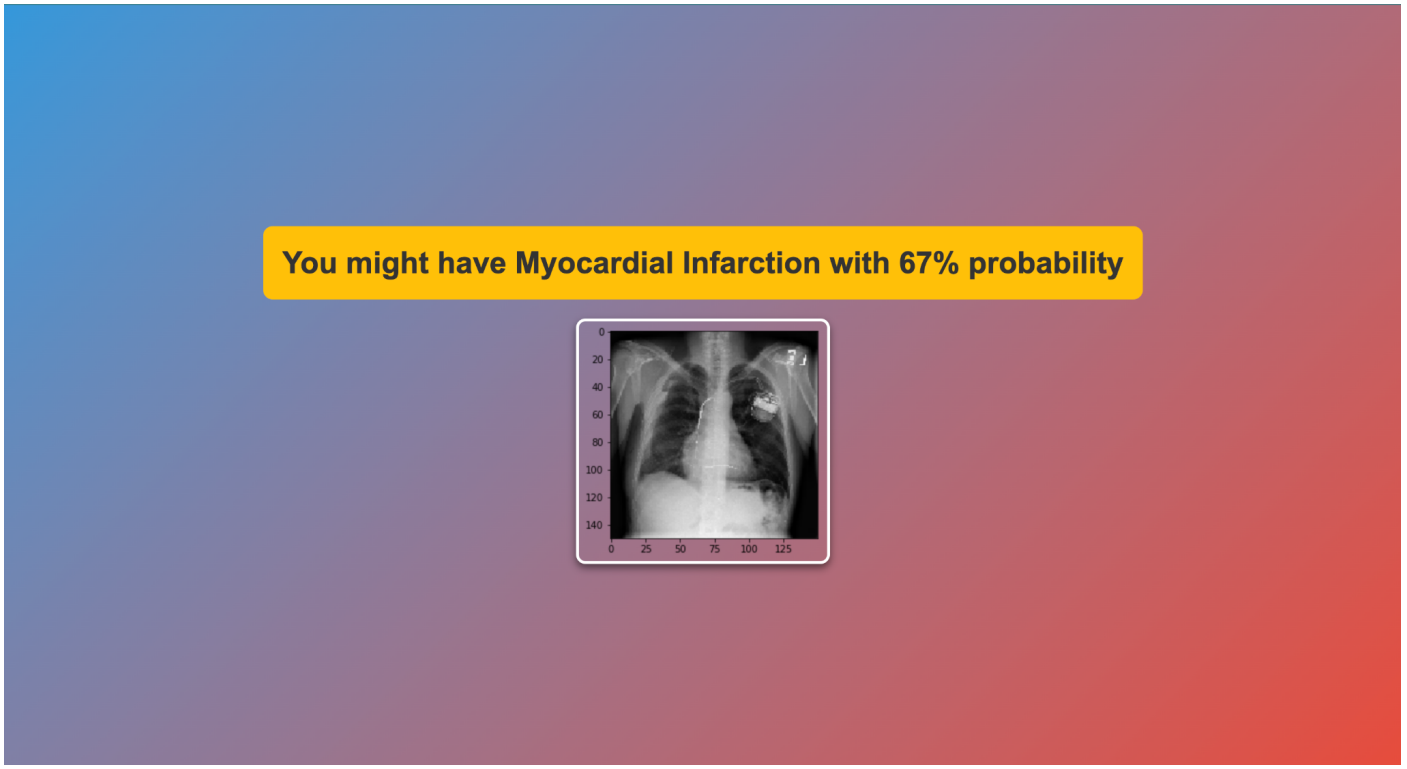
This is the user/patient profile when they have signed in to our system.



There is an option to edit the user profile.



This is the form the user has to fill up in order to get a diagnosis. The inputs from this form are fed to the model for prediction.



The model output is displayed above with the X-ray image used for the prediction as well.

Testing Report(use case)

Below are the overall test cases written for our project. We would like to explain a few of them, so that the reader can understand the context.Also while submitting we are going to submit the Test Case excel sheet for future references.

The first table presents a sign-up use case with a specific test case (TC028) where the user attempts to sign up with blank fields. The expected result is an error message indicating that the Name, Email, and Password fields cannot be empty, and the actual result confirms the system's correct handling of empty fields, resulting in a "Pass."

The second table addresses age input validation in a patient details use case, featuring two test cases (TC029 and TC030). TC029 verifies the system's response to entering alphanumeric characters as age, prompting a numeric input requirement, which is successfully confirmed as expected. TC030 ensures the system correctly processes and accepts valid numeric age entries, passing the test with the expected outcome.

Use Case: Sign Up							
Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC028	Entry of Blank Name, Email, and Password	User is on the sign-up page.	1. Leave the name, email, and password fields blank. 2. Click on the "Sign Up" button.	Blank name, email, and password fields.	System displays an error message indicating that the Name, Email, and Password fields cannot be empty.	Error message displayed for empty fields.	Pass
Use Case: Validate Patient's Age Input							
Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC029	Entry of Alphanumeric Characters in Age	User is entering patient details.	1. Enter a combination of alphanumeric characters as the age.	Alphanumeric characters entered as age.	System recognizes the input as invalid and does not let the user input other than numbers.	Invalid input recognized with a prompt to enter numbers only.	Pass
TC030	Successful Entry of Numeric Age	User is entering patient details.	1. Enter a valid numeric age.	Valid numeric age entered.	The system accepts the numeric age without errors.	Numeric age entered without errors.	Pass

The Below table outlines three email validation test cases conducted during system testing for a sign-up functionality. The tests ensure that the system correctly handles scenarios where the user attempts to sign up with a blank email field, enters random text without a valid email format, and successfully enters a valid email address. Through these tests, it is confirmed that the system prompts appropriate error messages for invalid inputs and accepts valid email entries as intended.

Use Case: Email Validation							
Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC031	Entry of Blank Email	User is on the sign-up page.	1. Leave the email field blank. 2. Click on the "Sign Up" button.	Blank email field.	System displays an error message indicating that the Email field cannot be empty.	Error message displayed for empty email field.	Pass
TC032	Entry of Random Text in Email	User is on the sign-up page.	1. Enter random text without '@' and '.' in the email field. 2. Click on the "Sign Up" button.	Random text entered in the email field.	System displays an error message instructing the user to enter a valid email address with '@' and '.'.	Error message displayed for invalid email format.	Pass
TC033	Entry of Valid Email	User is on the sign-up page.	1. Enter a valid email address. 2. Click on the "Sign Up" button.	Valid email address entered.	The system accepts the valid email without errors.	Valid email entered without errors.	Pass

Similarly we have developed 5 test cases for each use case please find them in below screenshots:

Profile_data							
Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC001	Patient Registration	The system is running and accessible.	1. Navigate to the patient registration page. 2. Enter valid patient details (name, age, gender, medical history). 3. Click on the "Submit" button.	Patient details with no errors.	Patient registration is successful, and data is saved in the database.	Patient registration is successful, and data is saved in the database.	Pass
TC002	Login as Doctor	A registered doctor account exists.	1. Navigate to the login page. 2. Enter valid doctor credentials. 3. Click on the "Login" button.	Valid doctor credentials.	Doctor is successfully logged in, and the dashboard is displayed.	Doctor is successfully logged in, and the dashboard is displayed.	Pass
TC003	Patient Data Entry	Doctor is logged in and viewing the patient's profile.	1. Select a patient from the list. 2. Enter relevant medical and diagnostic data for the patient. 3. Save the data.	Valid patient data.	Patient data is saved successfully.	Patient data is saved successfully.	Pass
TC004	Diagnose Heart Disease	Patient data is available in the system.	1. Select a patient for diagnosis. 2. Analyze the patient's symptoms and medical history. 3. Click on the "Diagnose" button.	Patient's specific heart disease is identified based on symptoms and medical history.	The system accurately diagnoses the heart disease.	The system accurately diagnoses the heart disease.	Pass
TC005	View Patient History	Patient data and diagnoses are available in the system.	1. Select a patient. 2. Navigate to the patient history section.	Patient's historical data and diagnoses are displayed.	The system correctly displays the patient's history.	The system correctly displays the patient's history.	Pass
TC006	System Availability	The system is running.	1. Access the system at different times of the day.	N/A	The system is accessible and responsive.	The system is available without downtime.	Pass

Use Case: Login/Sign up							
Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC007	Login with correct username and password	User account exists.	1. Navigate to the login page. 2. Enter valid username and password. 3. Click on the "Login" button.	Valid username and valid password.	Successful login.	Successful login.	Pass
TC008	Verify case sensitivity for credentials	User account exists.	1. Navigate to the login page. 2. Enter jumbled case sensitivity username/password. 3. Click on the "Login" button.	Jumbled case sensitivity username/password.	Successful login.	Successful login.	Pass
TC009	Login with invalid or non-existent username	User account does not exist.	1. Navigate to the login page. 2. Enter a non-registered username. 3. Click on the "Login" button.	Non-existent usernames are handled properly with a failed login attempt and an error message.	Failed login with error message.	Failed login with error message.	Pass
TC010	Forgot Password	User account exists.	1. Click on the "Forgot Password" link. 2. Enter a legitimate email for password recovery.	Email with a reset link is sent.	Email with reset link received.	Email with reset link received.	Pass
TC011	Prevent creation of an account with an existing email	User account with existing email exists.	1. Navigate to the sign-up page. 2. Enter an existing email and a new password. 3. Click on the "Sign Up" button.	Existing account creation is prevented with a relevant message.	Account creation prevented with a relevant message.	Account creation prevented with a relevant message.	Pass

Use Case: Validate Patient's Age Input

Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC012	Successful Entry of Numeric Age	User is entering patient details.	1. Enter a valid numeric age.	The system accepts the numeric age without errors.	Numeric age entered without errors.	Numeric age entered without errors.	Pass
TC013	Entry of Non-Numeric Characters in Age	User is entering patient details.	1. Enter non-numeric characters as the age.	The system displays an error message instructing the user to enter numbers only.	Error message displayed for non-numeric characters.	Error message displayed for non-numeric characters.	Pass
TC014	Entry of Alphanumeric Characters in Age	User is entering patient details.	1. Enter a combination of alphanumeric characters as the age.	The system recognizes the input as invalid and prompts the user to enter numbers only.	Invalid input recognized with a prompt to enter numbers only.	Invalid input recognized with a prompt to enter numbers only.	Pass
TC015	Entry of Blank Age	User is entering patient details.	1. Leave the age field blank.	The system displays an error message indicating that the age field cannot be empty.	Error message displayed for blank age field.	Error message displayed for blank age field.	Pass
TC016	Entry of Negative Age	User is entering patient details.	1. Enter a negative numeric value as the age.	The system recognizes the input as invalid and displays an error	Invalid input recognized with an error message.	Invalid input recognized with an	Pass

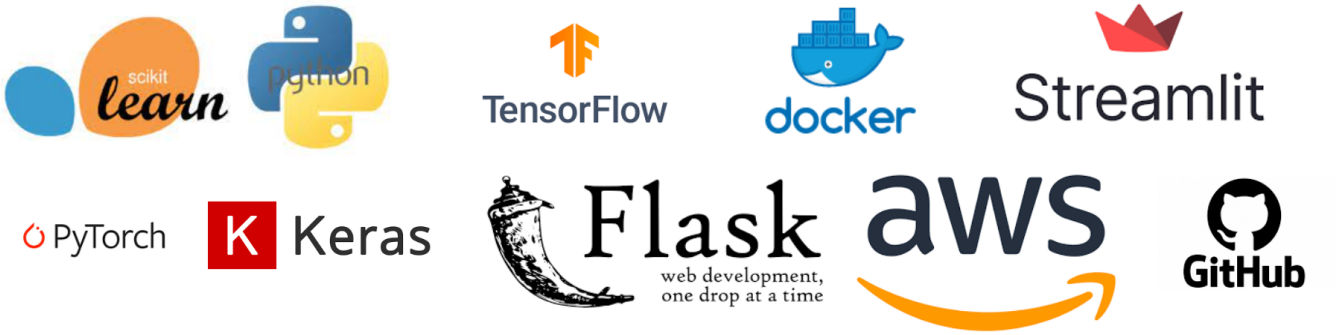
Use Case: Obtain Preliminary Diagnosis

Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC017	Successful Symptom Input and Diagnosis	Patient details and symptoms are entered.	1. Enter valid symptoms for diagnosis. 2. Click on the "Diagnose" button.	System successfully processes symptoms, generates a preliminary diagnosis, and displays it.	Preliminary diagnosis displayed accurately.	Preliminary diagnosis displayed accurately.	Pass
TC018	Insufficient Symptoms for Diagnosis	Patient details are entered.	1. Enter insufficient symptoms for diagnosis. 2. Click on the "Diagnose" button.	System recognizes lack of information, prompts user for more details, and does not	Prompt for more details displayed.	Prompt for more details displayed.	Pass
TC019	Technical Error during Diagnosis	Patient details and symptoms are entered.	1. Simulate a technical error during diagnosis. 2. Click on the "Diagnose" button.	System displays an error message and prompts the user to try the diagnosis again.	Error message displayed with prompt to try diagnosis again.	Error message displayed with prompt to try diagnosis again.	Pass
TC020	User Acknowledges Diagnosis	Patient details and symptoms are entered.	1. Receive and acknowledge the preliminary diagnosis.	System records acknowledgment, and additional instructions or prompts are provided.	Acknowledgment recorded with additional instructions.	Acknowledgment recorded with additional instructions.	Pass
TC021	User Seeks Further Information	Patient details and symptoms are entered.	1. Request more information or clarification regarding the preliminary diagnosis.	System provides additional details or recommendations based on the user's request.	Additional details provided in response to user query.	Additional details provided in response to user query.	Pass

Use Case: Validate X-ray Report Format

Test Case ID	Test Case	Pre-condition	Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
TC022	Successful Upload of Supported Image (JPG)	X-ray report upload functionality is accessible.	1. Upload a valid X-ray report in JPG format.	System accepts the JPG file without errors.	JPG file uploaded without errors.	JPG file uploaded without errors.	Pass
TC023	Successful Upload of Supported Image (PNG)	X-ray report upload functionality is accessible.	1. Upload a valid X-ray report in PNG format.	System accepts the PNG file without errors.	PNG file uploaded without errors.	PNG file uploaded without errors.	Pass
TC024	Successful Upload of Supported Document (PDF)	X-ray report upload functionality is accessible.	1. Upload a valid X-ray report in PDF format.	System accepts the PDF file without errors.	PDF file uploaded without errors.	PDF file uploaded without errors.	Pass
TC025	Attempted Upload of Unsupported Document (DOC)	X-ray report upload functionality is accessible.	1. Attempt to upload an X-ray report in DOC format.	System displays an error message, instructing the user to upload in a supported image or document format.	Error message displayed for unsupported document format.	Error message displayed for unsupported document format.	Pass
TC026	Attempted Upload of Unsupported Image (GIF)	X-ray report upload functionality is accessible.	1. Attempt to upload an X-ray report in GIF format.	System displays an error message, instructing the user to upload in a supported image format.	Error message displayed for unsupported image format.	Error message displayed for unsupported image format.	Pass
TC027	Attempted Upload of Empty File	X-ray report upload functionality is accessible.	1. Attempt to upload an empty X-ray report.	System displays an error message indicating that the file cannot be empty.	Error message displayed for empty file.	Error message displayed for empty file.	Pass

Tools & Read Me(file)



Tech Stack Used -

Development Environment - VS Code and Google Colab

Programming Language - Python

ML Libraries - Pytorch, Scikitlearn, pandas

Web technologies - html, css, javascript

Version Control - Github

API wrapper - Flask Framework

ReadMe File (Step - by - Step guide for deployment)

After a meeting with our client (Dr. Hajiarbabi), it was concluded that he wanted our AI model results only, and would have a better frontend and backend implementation for the next semester team.

However, we still created a working prototype, exceeding our clients expectations.

Here is a guide on how to run our project -

1. Conda activate mldl (our conda environment for this project)
2. Python refactored_model_serving.py (starts the flask backend)
3. Goto link - http://127.0.0.1:5002/ in your local host browser
4. Our home page should be visible there

As instructed by our client, we have also shared our entire dataset (45+ GB) used for training our models as well as all our EDA, data preprocessing , and model evaluation scripts with him for future development.

We have deployed our project here:
https://github.com/rnavyaprabha/Diagnosing-Heart-Disease

References:

- https://link.springer.com/article/10.1007/s42979-023-01818-w
- https://archive.ics.uci.edu/dataset/45/heart+disease
- https://physionet.org/content/mitdb/1.0.0/
- https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset
- https://physicsworld.com/a/deep-learning-model-uses-chest-x-rays-to-detect-heart-disease/

Team roles and Responsibilities:

Navyaprabha Rajappa**(Team lead and scam master)**
Worked on front end(Home,login,signup and profile), test cases, documentation, database and Backend, Deployment, contributed slides, scheduled client meetings .

Chaithanya Nalluru :
Worked on front end input page, prediction page, contributed in making slides,attended client meetings

Atharva Atre:
Worked on AI model development and Integration of model results with the front end and helped in documentation, slides.

Sai vikram gurram:
Worked on the Prototype(Bubble.io),Design, state diagrams and contributed in documentation, slides.