A HEALTHCARE SYSTEM FOR DISEASE PREDICTION WITH CHATBOT ASSISTANCE

Pavan Badempet, Shiva Prasad Anagondi, Prashanth Cheerala

Department of Computer Science & Engineering

Guru Nanak Institutions Technical Campus, Ibrahimpatnam 501506, Telangana, India

This work is submitted to Jawaharlal Nehru Technological University Hyderabad, In partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering

**ABSTRACT:** This project aims to develop a healthcare system that can predict multiple diseases, including liver disease, diabetes and heart disease using machine learning techniques. These are the most common types of Diseases that plaque the Indians today. This system will use XGBoost (eXtreme Gradient Boosting) for these diseases while maintaining accuracy, speed, resource utilization, stability and reducing complexity. The system will also include a healthcare chatbot that can provide personalized health recommendations to users. The system will be developed using Python and its many Machine Learning Libraries & Streamlit Framework. Machine learning models will be trained on publicly available datasets to predict diseases accurately. The developed models will be evaluated using performance metric like accuracy of training and testing data. The healthcare chatbot will be developed with NLP and/or OpenAI API Integration Techniques to provide personalized health recommendations to users. The system will be deployed on a cloud-based platform to ensure scalability and availability. The Healthcare System has the potential to improve the accuracy of disease diagnosis, care, and early intervention, thereby reducing healthcare costs and improving patient outcomes. The healthcare chatbot can provide accessible and personalized health recommendations, leading to a healthier and happier life.

**DATA AVAILABILITY:** The data that support the findings of this study are openly available in A-Healthcare-System-using-Machine-Learning-Techniques-for-Disease-Prediction-with-Chatbot-Assistance at https://github.com/pavanbadempet/A-Healthcare-System-using-Machine-Learning-Techniques-for-Disease-Prediction-with-Chatbot-Assistance.

**TABLE OF CONTENTS**

**CONTENTS PAGE**

**LIST OF FIGURES……………………………………………..………………………….....i**

**LIST OF SNAPSHOTS……………………………………..……………………………….ii**

**LIST OF NOTATIONS………………………………………...………………….………...iii**

**LIST OF ABBREVIATIONS…………………………….…………………………………vi**

**CHAPTER 1: INTRODUCTION……………………….…………………………………..1**

1.1 AIM OF THE PROJECT………….…………….….……………………………...1

1.2 OBJECTIVE OF THE PROJECT………...…..…………………………………...2

1.3 SCOPE OF THE PROJECT……………..………………………………………...2

1.4 EXISTING SYSTEM………………………..…………………………………….2

1.4.1 EXISTING SYSTEM DISADVANTAGES………..……………………3

1.5 LITERATURE SURVEY…………………………………………..……………...3 1.6 PROPOSED SYSTEM………….…………………………………………………6

1.6.1 PROPOSED SYSTEM ADVANTAGES…………………….………….7

**CHAPTER 2: PROJECT DESCRIPTION…………………………………………………8**

2.1 METHODOLOGIES…….….……..……………………………………………….8

2.2 TECHNIQUE USED OR ALGORITHM USED…………..……...……………...11

**CHAPTER 3: REQUIREMENTS ENGINEERING……………...……………………...12**

3.1 HARDWARE REQUIREMENTS……………...………………………………..12

3.2 SOFTWARE REQUIREMENTS………………………………………………...12

3.3 FUNCTIONAL REQUIREMENTS……………………………………………...13

3.4 NON-FUNCTIONAL REQUIREMENTS……………………………………….13

**CHAPTER 4: DESIGN ENGINEERING…………………………………………………15**

4.1 SYSTEM ARCHITECTURE…………………………………………………….15

4.2 LEVEL 0 DATA FLOW DIAGRAM…………………………………….………16

4.3 LEVEL 1 DATA FLOW DIAGRAM………………………………………...…..17

4.4 LEVEL 2 DATA FLOW DIAGRAM………………………………………...…..18

4.5 E-R DIAGRAM…………………………………………………………….……19

4.6 USE CASE DIAGRAM.........................................................................................20

4.7 CLASS DIAGRAM...............................................................................................21

4.8 OBJECT DIAGRAM.............................................................................................22

4.9 SEQUENCE DIAGRAM……………………..………………………………….23

4.10 COLLABORATION DIAGRAM…………………………..…………………..24

4.11 STATE DIAGRAM……………………………………………………………..25

4.12 ACTIVITY DIAGRAM………………………………………………………...26

4.13 COMPONENT DIAGRAM…………………………………………………….27

4.14 DEPLOYMENT DIAGRAM…………………………………………………...28

**CHAPTER 5: DEVELOPMENT TOOLS……………….………….…………………….29**

5.1 PYTHON…………………………………………………………………………29

5.2 HISTORY OF PYTHON…………………………………………………………29

5.3 IMPORTANCE OF PYTHON………………....………....………………………29

5.4 FEATURES OF PYTHON……………….………………………………………30

5.5 LIBRARIES USED………………………………………………………………31

**CHAPTER 6: IMPLEMENTATION……………………………………………………...32**

6.1 SAMPLE SOURCE CODE……………………………………………………...32

**CHAPTER 7: SNAPSHOTS…………………………………………..…………………...49**

7.1 VARIOUS SNAPSHOTS…………………………….…………………………..49

**CHAPTER 8: SOFTWARE TESTING……………………………………….…………...54**

8.1 DEVELOPING METHODOLOGIES……………………...……………………54 8.2 TYPES OF TESTING……………………………………………………………54

8.3 BUILD THE TEST PLAN……………………………………………………….56

8.3.1 TEST CASES……………………………..……………………………56

**CHAPTER 9: CONCLUSION & FUTURE ENHANCEMENT……………...…………58**

9.1 CONCLUSION…...……………………………………………………………..58 9.2 FUTURE ENHANCEMENT……………………………………………………58

**CHAPTER 10: BIBLIOGRAPHY…………………………………………………….…..59**

10.1 REFERENCE PAPERS…...……………………………………………………59 10.2 REFERENCE BOOKS…………………………………………………………60

10.3 REFERENCE WEBSITES……………………………………………………..60

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1 | SYSTEM ARCHITECTURE | 15 |
| 4.2 | LEVEL 0 DATA FLOW DIAGRAM | 16 |
| 4.3 | LEVEL 1 DATA FLOW DIAGRAM | 17 |
| 4.4 | LEVEL 2 DATA FLOW DIAGRAM | 18 |
| 4.5 | E-R DIAGRAM | 19 |
| 4.6 | USE CASE DIAGRAM | 20 |
| 4.7 | CLASS DIAGRAM | 21 |
| 4.8 | OBJECT DIAGRAM | 22 |
| 4.9 | SEQUENCE DIAGRAM | 23 |
| 4.10 | COLLABORATION DIAGRAM | 24 |
| 4.11 | STATE DIAGRAM | 25 |
| 4.12 | ACTIVITY DIAGRAM | 26 |
| 4.13 | COMPONENT DIAGRAM | 27 |
| 4.14 | DEPLOYMENT DIAGRAM | 28 |

**LIST OF SNAPSHOTS**

|  |  |  |
| --- | --- | --- |
| **SNAPSHOT NO.** | **NAME OF THE SNAPSHOT** | **PAGE NO.** |
| 7.1 | THE DEFAULT INTERFACE | 49 |
| 7.2 | DIABETES POSTIVE PREDICTION | 50 |
| 7.3 | DIABETES NEGATIVE PREDICTION | 50 |
| 7.4 | HEART DISEASE POSITIVE PREDICTION | 51 |
| 7.5 | HEART DISEASE NEGATIVE PREDICTION | 51 |
| 7.6 | LIVER DISEASE POSITIVE PREDICTION | 52 |
| 7.7 | LIVER DISEASE NEGATIVE PREDICTION | 52 |
| 7.8 | HEALTHCARE CHATBOT INTERFACE | 53 |
| 7.9 | HEALTHCARE CHATBOT RESPONSE | 53 |

**LIST OF NOTATIONS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO.** | **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *- private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | NAME  CLASS A  CLASS B  CLASS B  CLASS A | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor |  | It aggregates several classes into a single class |
| 4. | Aggregation | CLASS B  CLASS A  CLASS B  CLASS A | Interaction between the system and external environment |
| 5. | Relation  (uses) | Uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use Case |  | Interact ion between the system and external environment. |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATIONS**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **ABBREVATION** | **EXPANSION** |
| 1**.** | ML | Machine Learning |
| 2. | GUI | Graphical User Interface |
| 3. | NLP | Natural Language Processing |
| 4. | XGBoost | eXtreme Gradient Boosting |
| 5. | AI | Artificial Intelligence |
| 6. | DL | Deep Learning |

**CHAPTER 1**

**INTRODUCTION**

In today's era, healthcare systems face numerous challenges in accurately diagnosing and predicting diseases, leading to increased healthcare costs and compromised patient outcomes. To address these challenges, this project aims to develop a comprehensive healthcare system that utilizes machine learning techniques to predict multiple diseases, including liver disease, diabetes, and heart disease. These diseases are prevalent in the Indian population, making them a primary focus of this endeavour.

To ensure the system's effectiveness, the project will employ XGBoost (eXtreme Gradient Boosting), a powerful machine learning algorithm known for its accuracy, speed, and resource utilization. By leveraging the capabilities of XGBoost, the system will deliver robust predictions while reducing complexity and maintaining stability.

Additionally, the developed healthcare system will incorporate a healthcare chatbot that offers personalized health recommendations to users. This chatbot will utilize Natural Language Processing (NLP) techniques and/or potentially integrate with OpenAI APIs to provide users with tailored advice and support.

The implementation of the healthcare system will be carried out using Python programming language, which offers a rich ecosystem of machine learning libraries. Furthermore, the Streamlit framework will be utilized to create a user-friendly interface for seamless interaction with the system.

To ensure the accuracy of disease predictions, the machine learning models will be trained on publicly available datasets that encompass a wide range of patient information. Performance evaluation of the models will be conducted using metrics such as accuracy on both training and testing data, ensuring their reliability and effectiveness.

**1.1 AIM OF THE PROJECT**

The ultimate goal of this healthcare system is to significantly improve disease diagnosis accuracy, promote early intervention, and enhance patient care. By doing so, it has the potential to reduce healthcare costs and, more importantly, improve patient outcomes. Moreover, the integration of a healthcare chatbot into the system will enable users to access personalized health recommendations conveniently, fostering a healthier and happier life.

**1.2 OBJECTIVE OF THE PROJECT**

The objectives of this project are to develop a healthcare system that leverages machine learning techniques to accurately predict prevalent diseases like liver disease, diabetes, and heart disease, with a specific focus on the Indian population. The system aims to utilize the XGBoost algorithm, renowned for its accuracy, speed, and efficiency, to ensure precise disease predictions while maintaining stability and reducing complexity. Additionally, the project aims to incorporate a healthcare chatbot into the system, utilizing techniques such as Natural Language Processing (NLP) and OpenAI API integration to provide personalized health recommendations to users. The implementation will be carried out using the Python programming language and various machine learning libraries, with the Streamlit framework facilitating a user-friendly interface. The machine learning models will be trained on publicly available datasets, enabling robust disease predictions, and their performance will be evaluated using metrics such as accuracy on both training and testing data. Deployment on a cloud-based platform will ensure scalability and availability of the healthcare system. Ultimately, the project seeks to enhance disease diagnosis accuracy, enable early intervention, reduce healthcare costs, and improve patient outcomes. The integration of a healthcare chatbot aims to offer accessible and tailored health recommendations, empowering users to lead healthier and happier lives.

**1.3 SCOPE OF THE PROJECT**

The scope of this project includes the development of a healthcare system that utilizes machine learning techniques to predict common diseases, such as liver disease, diabetes, and heart disease, with a focus on the Indian population. The project also involves integrating a healthcare chatbot to provide personalized health recommendations [11]. The system will be developed using Python, machine learning libraries, and the Streamlit framework. The machine learning models will be trained on publicly available datasets, and the system will be deployed on a cloud-based platform for scalability and availability.

**1.4 EXISTING SYSTEM**

In the case of the existing system, there are very few existing systems in the market that have the potential to evolve into a comprehensive All-in-One Healthcare System for Disease Prediction. However, even those existing systems that do exist face significant limitations. These systems often require a substantial amount of computing resources, yet they struggle to achieve high levels of accuracy in disease prediction.

In terms of disease prediction algorithms, the existing system relies on Decision Trees for liver and heart disease prediction [4], and K-Nearest Neighbours (KNN) for diabetes prediction. However, each of these algorithms has its own drawbacks and limitations, leading to suboptimal performance in disease prediction.

Moreover, the existing systems lack a well-designed user interface (UI) and struggle with issues related to scalability and ease of use. They also fall short in providing personalized recommendations to users, which limits their effectiveness in addressing individual healthcare needs [9].

Given these limitations, there is a clear opportunity to develop a more advanced and comprehensive healthcare system that surpasses the existing solutions in terms of accuracy, usability, scalability, and personalized recommendations.

### **1.4.1 EXISTING SYSTEM DISADVANTAGES**

* KNN can be computationally expensive and slow when dealing with large datasets, as it requires calculating the distance between the new input point and every point in the training dataset.
* Decision Trees are prone to overfitting, especially when dealing with noisy data or complex datasets with many features. This can lead to poor generalization and low accuracy on new data.
* Both KNN and Decision Trees can suffer from the curse of dimensionality, where the performance of the model decreases as the number of features increases. This can lead to increased computation time and poor accuracy.

**1.5 LITERATURE SURVEY**

**Title:** Implementation and Use of Disease Diagnosis Systems for Electronic Medical Records Based on Machine Learning: A Complete Review

**Author:** Jahanzaib Latif, Chuangbai Xiao, Shanshan Tu, Sadaqat Ur Rehman, Azhar Imran, Anas Bilal

**Year:** 2021

**Description:** Electronic health records are used to extract patient's information instantly and remotely, which can help to keep track of patients' due dates for checkups, immunizations, and to monitor health performance. The Health Insurance Portability and Accountability Act (HIPAA) in the USA protects the patient data confidentiality, but it can be used if data is re-identified using `HIPAA Safe Harbor' technique. Usually, this re-identification is performed manually, which is very laborious and time captivating exertion. Various techniques have been proposed for automatic extraction of useful information, and accurate diagnosis of diseases. Most of these methods are based on Machine Learning and Deep Learning Methods, while the auxiliary diagnosis is performed using Rule-based methods. This review focuses on recently published papers, which are categorized into Rule-Based Methods, Machine Learning (ML) Methods, and Deep Learning (DL) Methods.

**Title:** Deployment of Containerized Deep Learning Applications in the Cloud

**Author:** Rim Doukha, Sidi Ahmed Mahmoudi, Mostapha Zbakh, Pierre Manneback

**Year:** 2021

**Description:** During the last years, the use of Cloud computing environment has increased as a result of the various services offered by Cloud providers (Amazon Web Services, Google Cloud, Microsoft Azure, etc.). Many companies are moving their data and applications to the Cloud in order to tackle the complex configuration effort, for having more flexibility, maintenance, and resource availability. However, it is important to mention the challenges that developers may face when using a Cloud solution such as the variation of applications requirements (in terms of computation, memory and energy consumption) over time, which makes the deployment and migration a hard process. In fact, the deployment will not depend only on the application, but it will also rely on the related services and hardware for the proper functioning of the application. In this paper, we propose a Cloud infrastructure for automatic deployment of applications using the services of Kubernetes, Docker, Ansible and Slurm. Our architecture includes a script to deploy the application depending of its requirement needs. Experiments are conducted with the analysis and the deployment of Deep Learning (DL) applications and more particularly images classification and object localization.

**Title:** Disease Prediction using Machine Learning Algorithms

**Author:** Sneha Grampurohit; Chetan Sagarnal

**Year:** 2020

**Description:** The development and exploitation of several prominent Data mining techniques in numerous real-world application areas (e.g. Industry, Healthcare and Bio science) has led to the utilization of such techniques in machine learning environments, in order to extract useful pieces of information of the specified data in healthcare communities, biomedical fields etc. The accurate analysis of medical database benefits in early disease prediction, patient care and community services. The techniques of machine learning have been successfully employed in assorted applications including Disease prediction. The aim of developing classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage. A Sample data of 4920 patients' records diagnosed with 41 diseases was selected for analysis. A dependent variable was composed of 41 diseases. 95 of 132 independent variables(symptoms) closely related to diseases were selected and optimized. This research work carried out demonstrates the disease prediction system developed using Machine learning algorithms such as Decision Tree classifier, Random forest classifier, and Naïve Bayes classifier.

**Title:** Comparison of Gradient Boosting and Extreme Boosting Ensemble Methods for Webpage Classification

**Author:** J Dutta, Yong Woon Kim, Dalia Dominic

**Year:** 2020

**Description:** Web page classification is an important task in various areas like web content filtering, contextual advertising and maintaining or expanding web directories etc. Machine Learning methods have been found to perform well to classify web pages, and ensemble models have been used to improve the results obtained from single classifiers. The Gradient Boosting and Extreme Boosting ensemble models are used in this work for binary classification. The dataset containing URLs of web pages have been collected manually. The comparison between the two boosting algorithms validated the improvement in accuracy and speed obtained through Extreme boosting. Extreme boosting has been found to be around ten times faster than Gradient boosting and also shows improvement in accuracy. The effect of three preprocessing techniques; lemmatization, stop words removal and regular expressions shows that these preprocessing techniques improves the accuracy of the results but not significantly.

**Title:** An Improved XGBoost Model Based on Spark for Credit Card Fraud Prediction

**Author:** Hongwei Chen, He Ai; Zhihui Yang, Weiwei Yang, Zhiwei Ye, Dawei Dong

**Year:** 2020

**Description:** Credit card fraud causes huge economic losses for many financial institutions. Given the imbalance of dataset and the huge amount of data in the field of credit card fraud, an improved XGBoost model based on Spark is proposed. In this project, the Smote algorithm was used to to balance the training set. And the XGBoost classifier based on Spark was used as the fraud detection mechanism. Finally, the test sets were classified in parallel. In the model comparison experiment, the model proposed in this project is compared with logistic regression model, decision tree model, random forest model, and original XGBoost model. The experimental results show that in the three metrics of Recall, Fl-Score, and AUC, the model proposed in this project is the best, which is 9.1%, 1.4%, and 1.2% ahead of the model ranked second respectively. In the speedup experiment, the speedup on the dataset of 70,000, 140,000, and 280,000 samples are 2.06, 3.28, and 3.75 respectively.

## 1.6 PROPOSED SYSTEM

The proposed system aims to overcome the limitations of existing systems by introducing advanced features and robust algorithms for disease prediction. In the proposed system, XGBoost (eXtreme Gradient Boosting), an Ensemble Boosting Technique, is utilized for accurate prediction of liver disease, heart disease, and diabetes [1].

By incorporating XGBoost, the proposed system ensures superior performance in disease prediction compared to traditional algorithms. XGBoost is known for its ability to handle complex and noisy medical datasets, resulting in more accurate and interpretable results.

Furthermore, the proposed system includes a healthcare chatbot that provides personalized health recommendations to users. This chatbot leverages the power of the XGBoost models to offer tailored advice and support based on individual healthcare needs.

To enhance usability and convenience, the proposed system incorporates a user-friendly interface (UI) that provides easy access to all the tools and functionalities in one place. This centralized UI allows users to interact with the disease prediction models and healthcare chatbot seamlessly.

The proposed system aims to be powerful, efficient, and comprehensive, delivering accurate disease predictions and personalized health recommendations. By leveraging advanced algorithms and providing a user-friendly interface, it seeks to provide a holistic solution for disease prediction and management, ultimately improving healthcare outcomes for individuals.

### **1.6.1 PROPOSED SYSTEM ADVANTAGES: -**

* XGBoost uses optimized algorithms and parallel processing techniques to handle large datasets more efficiently.
* XGBoost addresses overfitting issue by using regularization techniques, such as L1 and L2 regularization, to prevent overfitting and improve the generalization of the model.
* XGBoost uses feature selection and feature engineering techniques to reduce the number of features and improve the performance of the model.

**CHAPTER 2**

**PROJECT DESCRIPTION**

This project involves the development of a healthcare system that utilizes XGBoost (eXtreme Gradient Boosting), an Ensemble Boosting Technique, for accurate disease prediction [4]. The system focuses on predicting liver disease, heart disease, and diabetes. Additionally, a healthcare chatbot is integrated into the system to provide personalized health recommendations. The project aims to overcome the limitations of existing systems by employing robust algorithms, offering a user-friendly interface, and delivering accurate and interpretable results.

**2.1 METHODOLOGIES**

**2.1.1 MODULES**

* Data Collection
* Data Preprocessing
* Model Training
* Model Evaluation & Hyperparameter Tuning
* Web Application
* Containerization & Deployment on Cloud

**1. Data Collection**

The Data Collection module is a crucial component of the healthcare system for disease prediction. It involves the systematic gathering and acquisition of relevant data from various sources. In this project, the data collection process aims to obtain comprehensive datasets that encompass the necessary features for accurate disease prediction. The sources include publicly available healthcare databases, research studies, clinical records, and other reliable sources of medical information. Kaggle is a platform that hosts a variety of datasets that can be used for different projects. To ensure that the data is up to date, it is important to constantly revise the dataset and make minimal to no changes to the code. This ensures that the analysis is based on the most recent data available.

**2. Data Preprocessing**

The Data Preprocessing module plays a vital role in preparing the collected healthcare data for further analysis and model training. It involves a series of data transformation and normalization techniques to enhance the quality and usability of the data. The preprocessing steps typically include data cleaning, where missing values are handled through imputation or removal, outliers are identified and treated, and any inconsistencies or errors in the data are resolved. Feature scaling and normalization methods are applied to ensure that all features are on a similar scale, preventing certain features from dominating the analysis due to their larger magnitude. Categorical variables are encoded into numerical representations suitable for machine learning algorithms. Feature selection or dimensionality reduction techniques may also be employed to eliminate redundant or irrelevant features, reducing model complexity and improving computational efficiency. The preprocessing module aims to transform the raw data into a clean, structured, and standardized format that is suitable for training accurate and robust disease prediction models.

**3. Model Training**

The Model Training module is responsible for developing accurate disease prediction models using machine learning techniques. It involves training the models on the preprocessed healthcare data to learn patterns, relationships, and predictive features associated with different diseases. Various machine learning algorithms, such as XGBoost (eXtreme Gradient Boosting), can be employed for this purpose [5]. The training process involves splitting the dataset into training and validation subsets to assess model performance and prevent overfitting. The models are trained using optimization techniques, such as gradient descent, to iteratively adjust model parameters and minimize the prediction errors [4]. Hyperparameter tuning is also performed to find the optimal configuration that yields the best model performance. Model evaluation metrics, such as accuracy, precision, recall, and F1 score, are used to assess the model's performance on the training and validation datasets. The trained disease prediction models aim to provide accurate and reliable predictions, enabling early detection and intervention for improved healthcare outcomes.

**4. Model Evaluation & Hyperparameter Tuning**

The Model Evaluation & Hyperparameter Tuning module is an essential component of the healthcare system, focusing on assessing and fine-tuning the performance of the trained disease prediction models. This module involves evaluating the models using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score. The models are tested on separate testing datasets to assess their generalization capability and their ability to make accurate predictions on unseen data. Additionally, hyperparameter tuning techniques are employed to optimize the models' performance by finding the best combination of hyperparameters. Hyperparameters control the behavior and complexity of the models, and their values can significantly impact the prediction accuracy. Techniques like grid search, random search, or Bayesian optimization can be used to systematically explore the hyperparameter space and find the optimal configuration.

**5. Web Application**

The Web Application module serves as a user-friendly interface for disease prediction, leveraging the Streamlit framework. It comprises multiple pages for predicting diabetes, heart disease, liver disease, and interacting with a healthcare chatbot. Users input relevant medical data, triggering pre-trained machine learning models to make predictions. The application displays the results and diagnoses, indicating whether the person has the respective disease or not. The chatbot page enables users to ask healthcare-related queries, utilizing the OpenAI API to generate responses. The Web Application module provides an accessible platform for disease prediction and healthcare interactions, enhancing user experience and facilitating informed decision-making.

**6. Containerization & Deployment on Cloud**

The Containerization & Deployment on Cloud module employs Docker and Google Cloud Platform for the containerization and cloud deployment of the web application. Docker facilitates the creation of a container image containing all necessary dependencies, libraries, and configurations, while GCP serves as the cloud platform for hosting the containerized application [2]. By leveraging Docker and Google Cloud Platform, the module enables seamless deployment, scalability, and secure access to the web application, empowering users to utilize the disease prediction and healthcare chatbot functionalities effortlessly.

**2.2 TECHNIQUE USED OR ALGORITHM USED**

**2.2.1 EXISTING SYSTEM ALGORITHM**

**Decision Tree**

A decision tree is a machine learning Algorithm that uses a hierarchical structure of Branching decisions to model the Relationship between the input features and The output label. [8] Each internal node of the Tree represents a decision on a particular Feature, and each leaf node represents a class Label or a numerical value. The goal of the Decision tree is to recursively split the data Into subsets that are as homogeneous as Possible with respect to the target variable.

**k-Nearest Neighbours**

k-Nearest Neighbours (KNN) works by Finding the K nearest data points to a new Input data point in the training dataset, and then predicting the output value based on the Majority class or average value of the k-Nearest neighbours.

**2.2.2 PROPOSED SYSTEM ALGORITHM**

**XGBoost**

XGBoost is a powerful ensemble learning algorithm that is used for both classification and regression tasks [6]. It uses a gradient boosting framework to iteratively train multiple weak models (decision trees) on the data, where each subsequent model tries to correct the errors of the previous models [5].

XGBoost is known for its speed and scalability, and it has become a popular algorithm for winning machine learning competitions on Kaggle. It achieves this speed by implementing parallel processing and using hardware optimization techniques such as cache-aware computing and out-of-core computing. These Optimizations allow XGBoost to handle large Datasets with millions of features and millions of Instances in a relatively short amount of time.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

Requirements engineering is a critical phase in software development where the needs, expectations, and constraints are considered, analysed, and documented to define the desired system behaviour. It involves gathering, prioritizing, and validating requirements to ensure that the final software solution meets the intended objectives and user.

**3.1 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system does and not how it should be implemented.

* Processor : Core i3 Processor
* Ram : 4GB DDR3 RAM
* Hard Disk : 500 GB

**3.2 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Operating System : Linux 3.0/ Windows 7
* Front End : HTML, CSS, Streamlit, Python
* Containerization : Docker
* Back End : GCP, Python
* Version Control : GitHub

**3.3 FUNCTIONAL REQUIREMENTS:**

Functional requirements define the specific functionalities and features that a software system or application must possess to fulfill the needs of its users. These requirements describe what the system should do in terms of its behavior, operations, and interactions with users and other software components. They typically include details about user interactions, data processing, input/output operations, system responses, and any specific functional capabilities or constraints that need to be met. Functional requirements provide a clear understanding of the expected system behavior and serve as a basis for designing, implementing, and testing the software.

**3.4 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements, also known as quality attributes or system qualities, describe the overall characteristics and qualities that a software system must possess, beyond its specific functionalities. These requirements focus on the system's performance, reliability, usability, security, scalability, maintainability, and other aspects that contribute to its overall effectiveness and user satisfaction.

**Reliability**

Ensure consistent and error-free system performance under normal and exceptional conditions, minimizing failures and downtime.

**Performance**

Deliver fast response times, handle large workloads efficiently, and scale effectively as user demand grows.

**Usability**

Provide a user-friendly interface, intuitive navigation, and support for users with varying technical expertise.

**Security**

Safeguard user data through robust authentication, encryption, and measures against unauthorized access and threats.

**Scalability**

Design the system to handle increasing workloads, data volumes, and user bases without compromising performance.

**Maintainability**

Enable easy updates, bug fixes, and enhancements through well-documented code, standards, and modularity

**CHAPTER 4**

**DESIGN ENGINEERING**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.1 SYSTEM ARCHITECTURE:**

A screenshot of a computer

Description automatically generated with medium confidence

Fig 4.1: System Architecture

## EXPLANATION

The system architecture of this project is designed as a web-based application that utilizes machine learning models for disease prediction. The application is built using the Streamlit framework for the frontend, which allows users to input their health data. The backend consists of the loaded machine learning models for diabetes, heart disease, and liver disease, and a Healthcare Chatbot which are used to process the input data and generate predictions or answers. The application provides an intuitive and user-friendly interface for users to input their health parameters, and the system processes the data using the appropriate model to provide accurate disease predictions. This architecture ensures a seamless and efficient user experience in predicting multiple diseases based on individual health data by using a very powerful algorithm and containerization.

## 4.2 LEVEL 0 DATA FLOW DIAGRAM

**Level 0:**

For collected data create a dataframe in .csv

Search for available dataset for Diabetes, Heart & Liver Diseases

Collect Data from UCI/Kaggle

User

Fig 4.2: Level 0 Data Flow Diagram

**4.3 LEVEL 1 DATA FLOW DIAGRAM**

**Level 1:4.4 LEVEL 2 DATA FLOW DIAGRAM**

Fig 4.3: Level 1 Data Flow Diagram

Train using XGBoost(eXtreme Gradient Boosting), Validate & Tune Hyperparameters & save like Model.pkl

Data Pre-processing

Disease Datasets

User

**Level 2:**

User Health data input

Load TrainedModel.pkl & Apply converted data.

Disease Prediction

Float/int, Array conversion

Fig 4.4: Level 2 Data Flow Diagram

**EXPLANATION**

A data flow diagram (DFD) is a graphical representation of the “flow” of the data through an information system, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFD’s can also be used for the visualization of data processing (structured design). A DFD shows what kind of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

## 4.5 E-R DIAGRAM

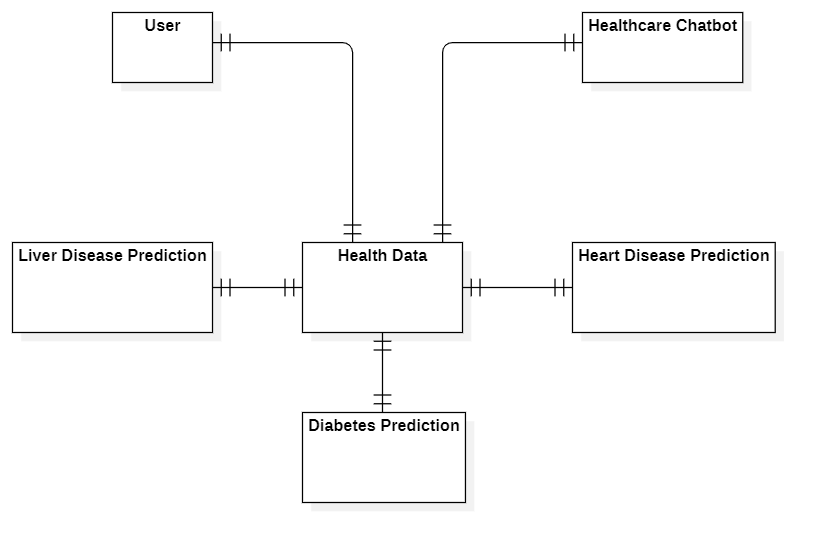


Fig 4.5: E-R Diagram

## EXPLANATION

An ER (Entity Relationship) diagram is a graphical representation of entities and their relationships to each other. It is used to model and design the database of a system, and it is a common tool in the field of database design. ER diagrams are used to depict the relationships between entities and the attributes that define them. ER diagrams are a useful tool for visualizing and communicating the structure of a database, and they can be used to identify potential issues or inconsistencies in the design before implementing it in the database.

## 4.6 USE CASE DIAGRAM

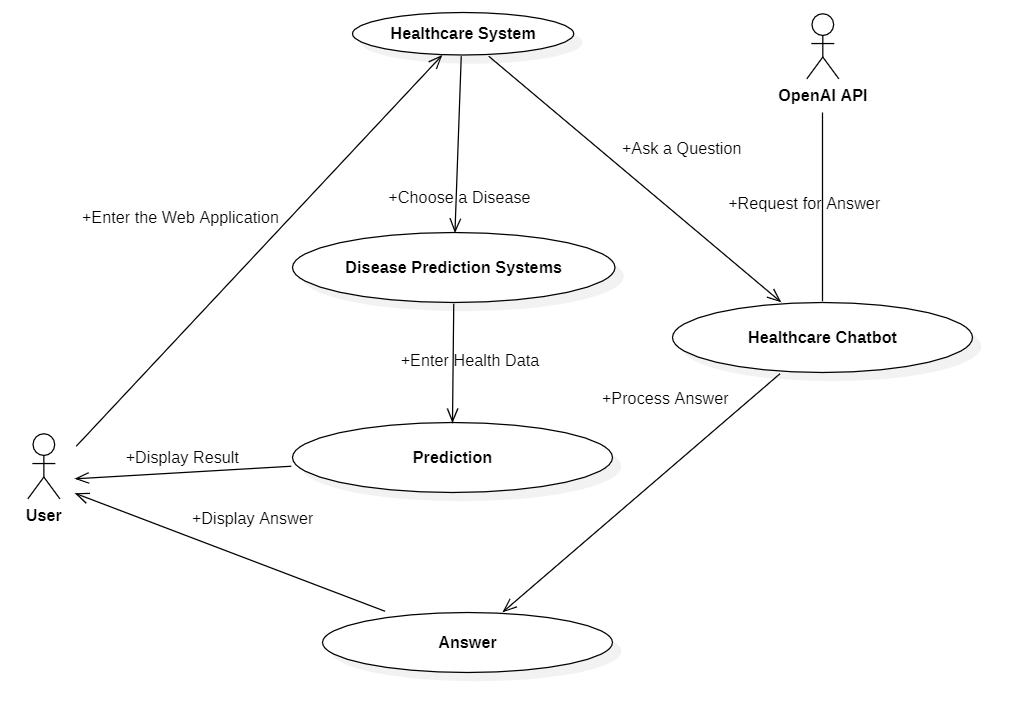


Fig 4.6: Use Case Diagram

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

## 4.7 CLASS DIAGRAM

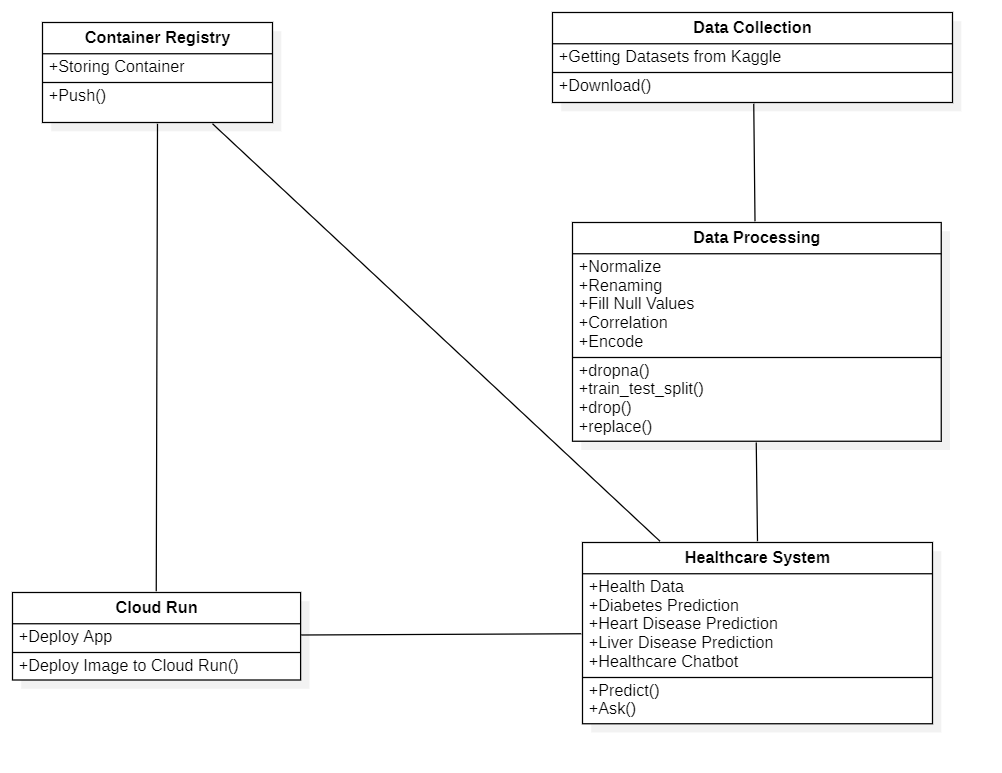


Fig 4.7: Class Diagram

## EXPLANATION

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

## 4.8 OBJECT DIAGRAM

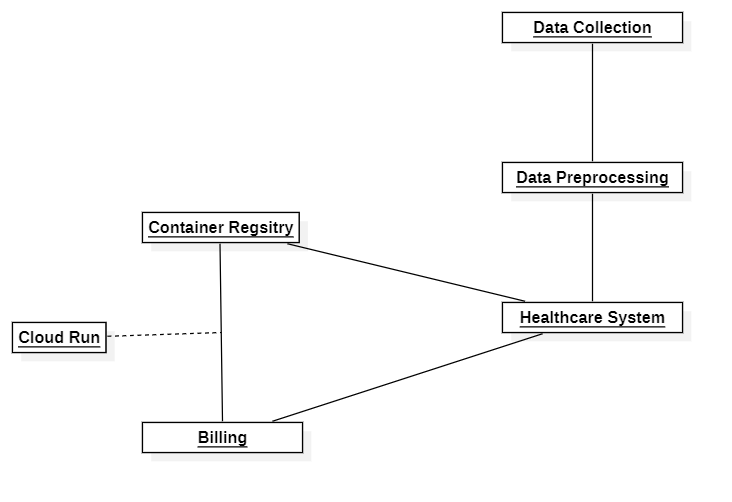


Fig 4.8: Object Diagram

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.9 SEQUENCE DIAGRAM**

A screenshot of a computer

Description automatically generated with medium confidence

Fig 4.9: Sequence Diagram

**EXPLANATION:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

## 4.10 COLLABORATION DIAGRAM

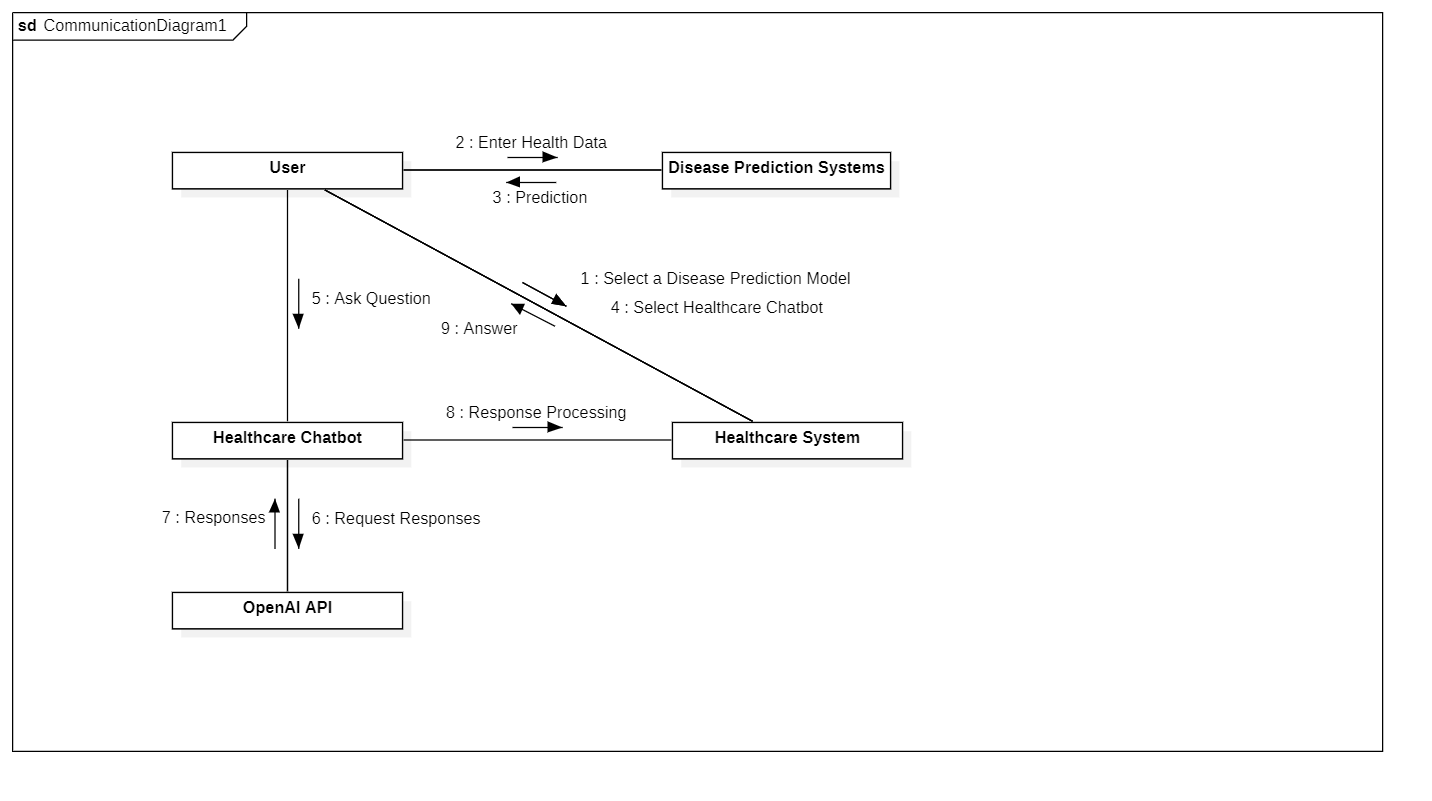


Fig 4.10: Collaboration Diagram

**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

## 4.11 STATE DIAGRAM

A screen shot of a computer screen

Description automatically generated with low confidence

Fig 4.11: State Diagram

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration, and concurrency. State diagrams require that the system described is composed of a finite number of states, sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

## 4.12 ACTIVITY DIAGRAM

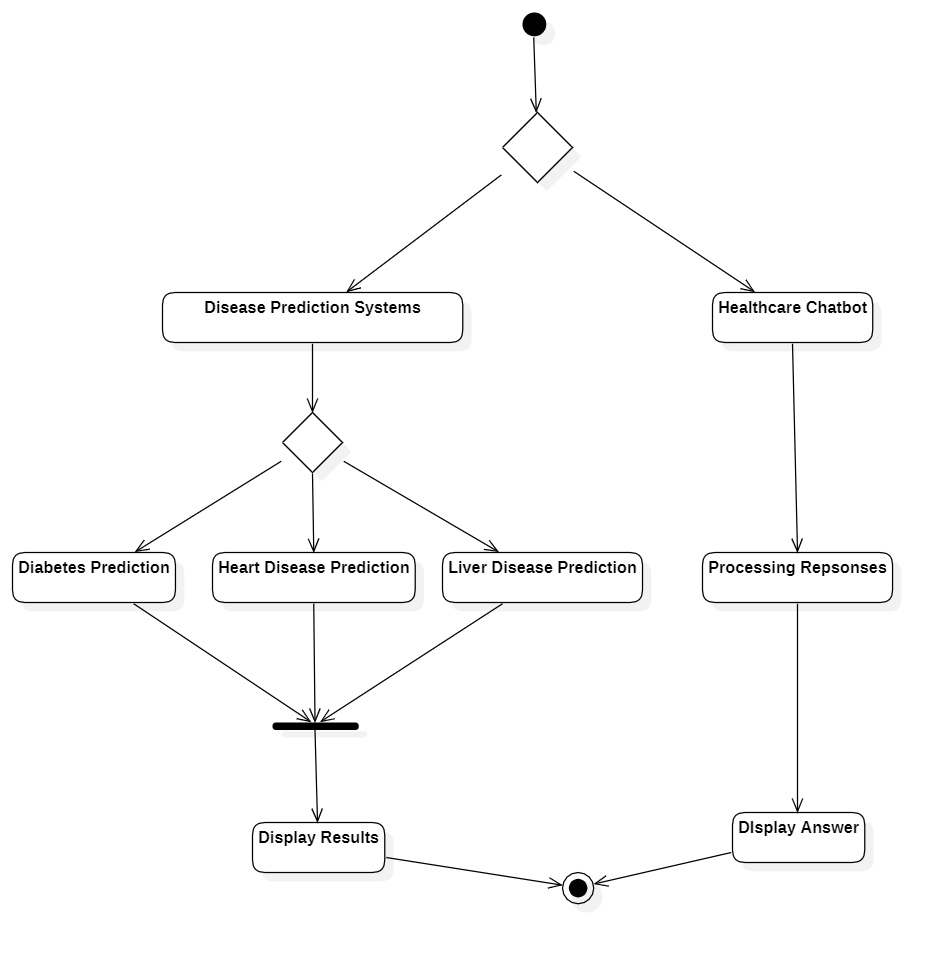


Fig 4.12: Activity Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

## 4.13 COMPONENT DIAGRAM

A picture containing text, screenshot, font, line

Description automatically generated

Fig 4.13: Component Diagram

## EXPLANATION

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicate dependencies.

## 4.14 DEPLOYMENT DIAGRAM

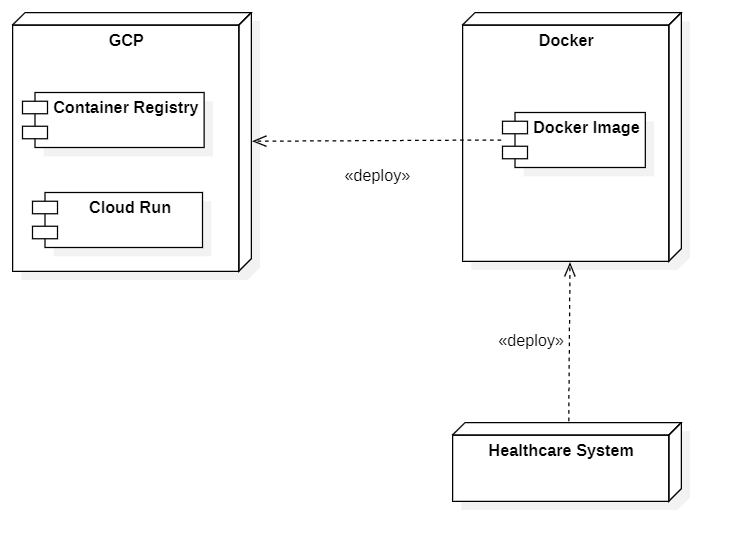


Fig 4.14 Deployment Diagram

**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

# **CHAPTER 5**

# **DEVELOPMENT TOOLS**

This chapter is about the software language and the tools used in the development of the project. The platform used here is PYTHON.

## 5.1 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.2 HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

### **5.3 IMPORTANCE OF PYTHON**

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
* **Python is Interactive** − You can sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## 5.4 FEATURES OF PYTHON

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

**5.5 LIBRARIES USED IN PYTHON:**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as data frames.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.

A close-up of a logo

Description automatically generated with low confidence

Figure 5.6: Logos of NumPy, Pandas, Matplotlib, Scikit-learn

# **CHAPTER 6**

# **IMPLEMENTATION**

The Implementation includes the sample source codes of Diabetes Model, Heart Disease Model, Liver Disease Model along with the web app code.

## 6.1 SAMPLE SOURCE CODE

#Diabetes

#importing dependences

import xgboost as xgb

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

# Data Collection and Analysis

# loading the diabetes dataset to a pandas DataFrame

# https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/code

diabetes\_dataset = pd.read\_csv(r"C:\Users\pavan\Desktop\A Healthcare System using Machine Learning Techniques for Disease Prediction with Chatbot Assistance\Datasets\diabetes.csv")

# printing the first 5 rows of the dataset

diabetes\_dataset.head()

# number of rows and Columns in this dataset

diabetes\_dataset.info()

diabetes\_dataset = diabetes\_dataset.astype({'age':'int'})

#Replacing with numeric

diabetes\_dataset['gender'].replace(['Other', 'Male','Female'],[0,1,2], inplace=True)

diabetes\_dataset['smoking\_history'].replace(['never', 'No Info', 'current', 'former', 'ever', 'not current'],[0,1,2,3,4,5,], inplace=True)

#Removed Other because the values were very less

diabetes\_dataset = diabetes\_dataset[diabetes\_dataset['gender'] != 0]

diabetes\_dataset = diabetes\_dataset.reset\_index(drop=True)

# getting the statistical measures of the data

diabetes\_dataset.describe()

#Train Test Split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(diabetes\_dataset.drop("diabetes", axis=1), diabetes\_dataset["diabetes"], test\_size=0.3)

#Model Training

classifier = xgb.XGBClassifier()

classifier.fit(X\_train, Y\_train)

#Accuracy

print('Training accuracy {:.4f}'.format(classifier.score(X\_train,Y\_train)))

print('Testing accuracy {:.4f}'.format(classifier.score(X\_test,Y\_test)))

#Making a Predictive System

input\_data = (1,89,66,23,94,28.1,0.167,21)

# changing the input\_data to numpy array

input\_data\_as\_numpy\_array = np.asarray(input\_data)

# reshape the array as we are predicting for one instance

input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)

prediction = classifier.predict(input\_data\_reshaped)

print(prediction)

if (prediction[0] == 0):

print('The person is not diabetic')

else:

print('The person is diabetic')

# Saving the trained model

import pickle

filename = 'diabetes\_model.pkl'

pickle.dump(classifier, open(filename, 'wb'))

#Heart Disease

#Similar Preprocessing Steps

#GridSearchCV for finding the best hyperparameters

# Define the parameter grid

param\_grid = {

'n\_estimators': [100, 150, 200],

'learning\_rate': [0.05, 0.1, 0.15],

'max\_depth': [5, 6, 7],

'subsample': [0.4, 0.5, 0.6],

'colsample\_bytree': [0.4, 0.5, 0.6],

'gamma': [0, 0.01, 0.1],

'reg\_alpha': [0, 0.01, 0.1],

'reg\_lambda': [0.5, 0.6, 0.7]

}

# Define the best accuracy and corresponding hyperparameters

best\_accuracy = 0

best\_params = {}

# Iterate over all parameter combinations

for n\_estimators in param\_grid['n\_estimators']:

for learning\_rate in param\_grid['learning\_rate']:

for max\_depth in param\_grid['max\_depth']:

for subsample in param\_grid['subsample']:

for colsample\_bytree in param\_grid['colsample\_bytree']:

for gamma in param\_grid['gamma']:

for reg\_alpha in param\_grid['reg\_alpha']:

for reg\_lambda in param\_grid['reg\_lambda']:

# Create the XGBoost classifier with the current parameter values

classifier = xgb.XGBClassifier(

n\_estimators=n\_estimators,

learning\_rate=learning\_rate,

max\_depth=max\_depth,

subsample=subsample,

colsample\_bytree=colsample\_bytree,

gamma=gamma,

reg\_alpha=reg\_alpha,

reg\_lambda=reg\_lambda,

random\_state=42

)

# Fit the classifier to the training data

classifier.fit(X\_train, Y\_train)

# Make predictions on the test set

predictions = classifier.predict(X\_test)

# Calculate the accuracy score

accuracy = accuracy\_score(Y\_test, predictions)

# Check if this combination of parameters achieved a better accuracy

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_params = {

'n\_estimators': n\_estimators,

'learning\_rate': learning\_rate,

'max\_depth': max\_depth,

'subsample': subsample,

'colsample\_bytree': colsample\_bytree,

'gamma': gamma,

'reg\_alpha': reg\_alpha,

'reg\_lambda': reg\_lambda

}

# Print the best hyperparameters and their corresponding accuracy score

print("Best Hyperparameters: ", best\_params)

print("Accuracy Score: ", best\_accuracy)

#Bulding on the best parameters

classifier = xgb.XGBClassifier(\*\*best\_params, random\_state=42)

classifier.fit(X\_train, Y\_train)

train\_accuracy = classifier.score(X\_train, Y\_train)

print('Training accuracy: {:.4f}'.format(train\_accuracy))

test\_accuracy = classifier.score(X\_test, Y\_test)

print('Testing accuracy: {:.4f}'.format(test\_accuracy))

#Liver Disease

# More comprehensive Preprocessing

# Finding Correlations between features to have only independent features

liver\_corr = liver\_data.corr()

plt.figure(figsize=(30, 30))

sns.heatmap(liver\_corr, cbar = True, square = True, annot=True, fmt= '.2f',annot\_kws={'size': 15},cmap= 'coolwarm')

plt.title('Correlation between features')

# Direct\_Bilirubin & Total\_Bilirubin 0.87

# Alamine\_Aminotransferase & Aspartate\_Aminotransferase 0.79

# Albumin and Total Protein 0.78

# albumin\_and\_globulin\_ratio & albumin 0.69

# dropping highly correlated features as features should be independent

liver\_data.drop(['Direct\_Bilirubin', 'Aspartate\_Aminotransferase', 'Total\_Protiens', 'Albumin'], axis=1, inplace=True)

#Data Analysis shows skewness

plt.figure(figsize=(15, 12))

features = ['Albumin\_and\_Globulin\_Ratio', 'Total\_Bilirubin', 'Alkaline\_Phosphotase', 'Alamine\_Aminotransferase']

num\_rows = (len(features) - 1) // 2 + 1

num\_cols = 2

for i, feature in enumerate(features):

plt.subplot(num\_rows, num\_cols, i+1)

sns.distplot(liver\_data[features])

plt.title('Distribution plot for field: ' + feature)

plt.xlabel('')

plt.tight\_layout(pad=0.4, w\_pad=0.5, h\_pad=1.0)

plt.show()

skewness = liver\_data.skew().abs() # Absolute skewness values

print(skewness)

#These features with high skewness

skewed = ['Albumin\_and\_Globulin\_Ratio','Total\_Bilirubin', 'Alkaline\_Phosphotase', 'Alamine\_Aminotransferase']

# Apply log1p transformation on dataframe - just selected values to reduce skewness

for i in skewed:

liver\_data[i] = liver\_data[i].apply('log1p')

#Scaling & Encoding

from sklearn.preprocessing import LabelEncoder, RobustScaler

# gender contains string values Male, Female; these will be converted into 0, 1, as ML algorithms like just numerical values

le = LabelEncoder()

liver\_data['Gender'] = le.fit\_transform(liver\_data['Gender'])

liver\_data.Gender.head()

# Get all attribute names except 'Dataset'

attributes = [col for col in liver\_data.columns if col != 'Dataset']

rs = RobustScaler()

liver\_data[attributes] = rs.fit\_transform(liver\_data[attributes])

# Web Application

import pickle

import requests

import streamlit as st

from streamlit\_option\_menu import option\_menu

import openai

import pandas as pd

import numpy as np

st.set\_page\_config(

page\_title="Healthcare System",

page\_icon=":health\_worker:",

layout="centered",

initial\_sidebar\_state="expanded"

)

hide\_streamlit\_style = """

<style>

#MainMenu {visibility: hidden;}

footer {visibility: hidden;}

</style>

"""

st.markdown(hide\_streamlit\_style, unsafe\_allow\_html=True)

# loading the saved models

diabetes\_model = pickle.load(open('Diabetes Model.pkl', 'rb'))

heart\_disease\_model = pickle.load(open("Heart Disease Model.pkl",'rb'))

liver\_model = pickle.load(open("Liver Disease Model.pkl", 'rb'))

scaler = pickle.load(open("Scaler.pkl", 'rb'))

# sidebar for navigation

with st.sidebar:

selected = option\_menu('Healthcare System',

['Diabetes Prediction',

'Heart Disease Prediction',

'Liver Disease Prediction',

'Healthcare Chatbot'],

icons=['droplet-fill','heart','person'],

default\_index=0)

if selected == 'Diabetes Prediction':

# Page title

st.title('Diabetes Prediction')

st.markdown("Note: 1: Gender (Female: 0, Male: 1)")

st.markdown("Note: 2: Smoking History (never: 0, No Info: 1, current: 2, former:3, ever:4, not current: 5)")

st.markdown("Note: 3: Heart Disease (No: 0 , Yes: 1)")

# Input fields

col1, col2, col3 = st.columns(3)

with col1:

gender = st.text\_input('Gender ')

gender = float(gender) if gender else 0.0

with col2:

age = st.text\_input('Age')

age = float(age) if age else 0.0

with col3:

hypertension = st.text\_input('Hypertension Value')

hypertension = float(hypertension) if hypertension else 0.0

with col1:

heart\_disease = st.text\_input('Heart Disease')

heart\_disease = float(heart\_disease) if heart\_disease else 0.0

with col2:

smoking\_history = st.text\_input('Smoking\_history Level')

smoking\_history = float(smoking\_history) if smoking\_history else 0.0

with col3:

BMI = st.text\_input('BMI value')

BMI = float(BMI) if BMI else 0.0

with col1:

HbA1c\_level = st.text\_input('HbA1c\_level value')

HbA1c\_level = float(HbA1c\_level) if HbA1c\_level else 0.0

with col2:

blood\_glucose\_level = st.text\_input('Blood Glucose Level')

blood\_glucose\_level = float(blood\_glucose\_level) if blood\_glucose\_level else 0.0

# Perform prediction

diab\_diagnosis = ''

if st.button('Diabetes Test Result'):

input\_data = np.array([[gender, age, hypertension, heart\_disease, smoking\_history, BMI, HbA1c\_level, blood\_glucose\_level]], dtype=object)

input\_data = input\_data.astype(float)

diab\_prediction = diabetes\_model.predict(input\_data)

if diab\_prediction[0] == 1:

diab\_diagnosis = 'The person is predicted to have diabetes.'

st.error(diab\_diagnosis)

else:

diab\_diagnosis = 'The person is predicted to be healthy.'

st.success(diab\_diagnosis)

# Heart Disease Prediction Page

if (selected == 'Heart Disease Prediction'):

# page title

st.title('Heart Disease Prediction')

st.markdown("Note: 1: Gender (Female: 0, Male: 1)")

st.markdown("Note: 2: Thal (Normal: 0, Fixed Defect: 1, Reversible Defect: 2)")

col1, col2, col3 = st.columns(3)

col4, col5, col6 = st.columns(3)

col7, col8, col9 = st.columns(3)

col10, col11, col12, col13 = st.columns(4)

# Input fields

with col1:

age = st.text\_input('Age')

age = float(age) if age else 0.0

with col2:

gender = st.text\_input('Gender')

gender = float(gender) if gender else 0.0

with col3:

cp = st.text\_input('Chest Pain types')

cp = float(cp) if cp else 0.0

with col4:

trestbps = st.text\_input('Resting Blood Pressure')

trestbps = float(trestbps) if trestbps else 0.0

with col5:

chol = st.text\_input('Serum Cholestoral')

chol = float(chol) if chol else 0.0

with col6:

fbs = st.text\_input('Fasting Blood Sugar')

fbs = float(fbs) if fbs else 0.0

with col7:

restecg = st.text\_input('Resting Electrocardiographic results')

restecg = float(restecg) if restecg else 0.0

with col8:

thalach = st.text\_input('Maximum Heart Rate achieved')

thalach = float(thalach) if thalach else 0.0

with col9:

exang = st.text\_input('Exercise Induced Angina')

exang = float(exang) if exang else 0.0

with col10:

oldpeak = st.text\_input('ST depression induced by exercise')

oldpeak = float(oldpeak) if oldpeak else 0.0

with col11:

slope = st.text\_input('Slope of the peak exercise ST segment')

slope = float(slope) if slope else 0.0

with col12:

ca = st.text\_input('Major vessels colored by fluoroscopy')

ca = float(ca) if ca else 0.0

with col13:

thal = st.text\_input('Thal Value')

thal = float(thal) if thal else 0.0

# Perform prediction

heart\_diagnosis = ''

if st.button('Heart Disease Test Result'):

input\_data = np.array([[age, gender, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]], dtype=object)

input\_data = input\_data.astype(float)

heart\_prediction = heart\_disease\_model.predict(input\_data)

if heart\_prediction[0] == 1:

heart\_diagnosis = 'The person is predicted to have heart disease.'

st.error(heart\_diagnosis)

else:

heart\_diagnosis = 'The person is predicted to be healthy.'

st.success(heart\_diagnosis)

# Liver Prediction Page

if (selected == "Liver Disease Prediction"):

# page title

st.title("Liver Disease Prediction")

st.markdown("Note: 1: Gender (Female: 0, Male: 1)")

col1, col2, col3 = st.columns(3)

col4, col5, col6 = st.columns(3)

with col1:

Age = st.text\_input('Age')

with col2:

Gender = st.text\_input('Gender')

with col3:

Total\_Bilirubin = st.text\_input('Total Bilirubin')

with col4:

Alkaline\_Phosphotase = st.text\_input('Alkaline Phosphotase')

with col5:

Alamine\_Aminotransferase = st.text\_input('Alamine Aminotransferase')

with col6:

Albumin\_and\_Globulin\_Ratio = st.text\_input('Albumin and Globulin Ratio')

# code for Prediction

liver\_diagnosis = ''

# creating a button for Prediction

if st.button("Liver Test Result"):

def preprocess\_input(data):

# Apply log1p transformation

skewed = ['Total\_Bilirubin', 'Alkaline\_Phosphotase', 'Alamine\_Aminotransferase','Albumin\_and\_Globulin\_Ratio']

data[skewed] = np.log1p(data[skewed])

# Scale the data using the loaded scaler

attributes = [col for col in data.columns]

data[attributes] = scaler.transform(data[attributes])

return data

input\_data = [Age,Gender,Total\_Bilirubin,Alkaline\_Phosphotase,Alamine\_Aminotransferase,Albumin\_and\_Globulin\_Ratio]

column\_names = ['Age', 'Gender', 'Total\_Bilirubin', 'Alkaline\_Phosphotase','Alamine\_Aminotransferase', 'Albumin\_and\_Globulin\_Ratio']

# Convert the user's input into a pandas DataFrame

user\_data = pd.DataFrame([input\_data], columns=column\_names)

user\_data[column\_names] = user\_data[column\_names].apply(pd.to\_numeric, errors='coerce')

# Preprocess the user's input data

preprocessed\_data = preprocess\_input(user\_data)

prediction = liver\_model.predict(preprocessed\_data)

if (prediction[0] == 0):

liver\_diagnosis = "The person does not have a Liver disease"

st.success(liver\_diagnosis)

else:

liver\_diagnosis = "The Person has Liver Disease"

st.error(liver\_diagnosis)

#Chatbot

if (selected == 'Healthcare Chatbot'):

# Define the GPT API endpoint

API\_ENDPOINT = "https://api.pawan.krd/v1/completions"

# Define your OpenAI API key

API\_KEY = "pk-eAWvHQfEkRiWCiCNMDLnOGdfpqgxCQzbPtPrBvtdbmHmFktW"

# Function to interact with the GPT API

def generate\_response(prompt):

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_KEY}"

}

data = {

"prompt": prompt,

"max\_tokens": 1000 # Adjust the max tokens as needed

}

response = requests.post(API\_ENDPOINT, headers=headers, json=data)

response\_json = response.json()

return response\_json["choices"][0]["text"].strip()

# Function to simulate bot typing effect

def simulate\_typing():

st.text("Bot is typing...")

# Main code

st.title("Healthcare Chatbot")

st.markdown("Welcome to the Healthcare Chatbot! How can I assist you today?")

# User input

user\_input = st.text\_input("User:")

# Generate bot response

if user\_input:

bot\_response = generate\_response(user\_input)

bot\_response\_html = f'<div style="overflow-wrap: break-word; height: auto; padding: 10px;">{bot\_response}</div>'

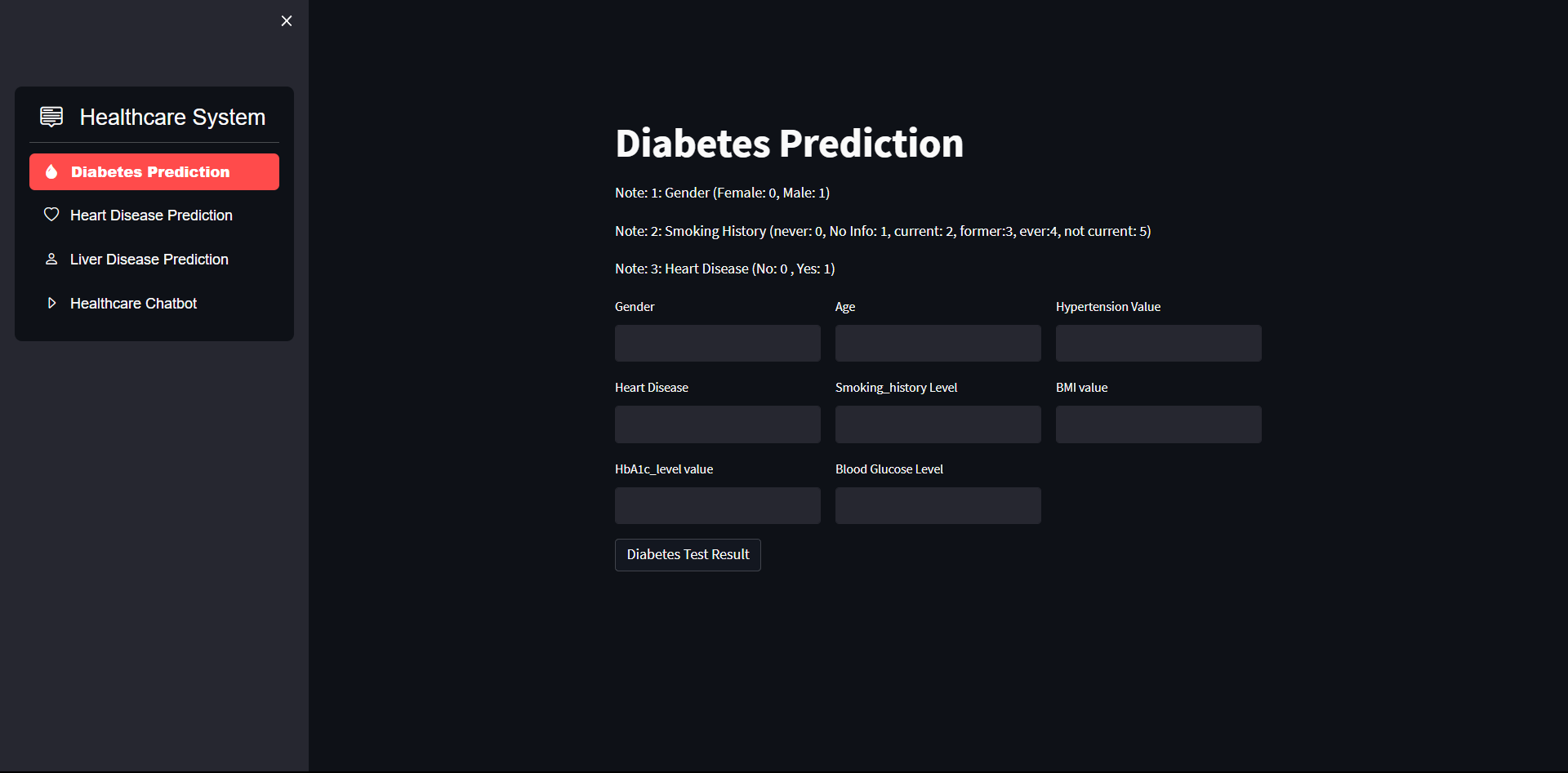
st.markdown(bot\_response\_html, unsafe\_allow\_html=True)

# **CHAPTER 7**

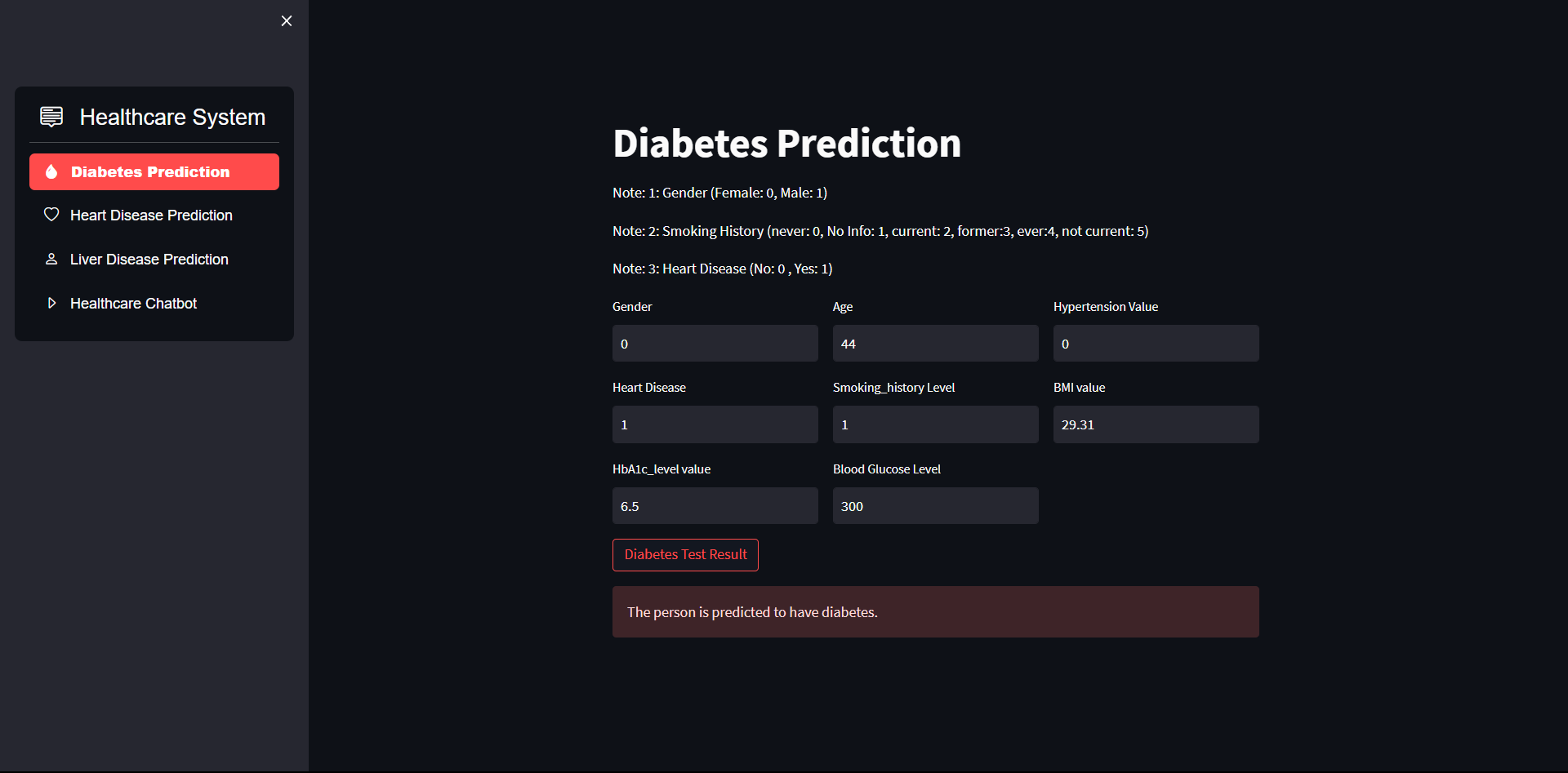
# **SNAPSHOTS**

The project is developed using Python and its many machine learning modules. While the model was developed using Python the Web App Service is developed using Streamlit framework.

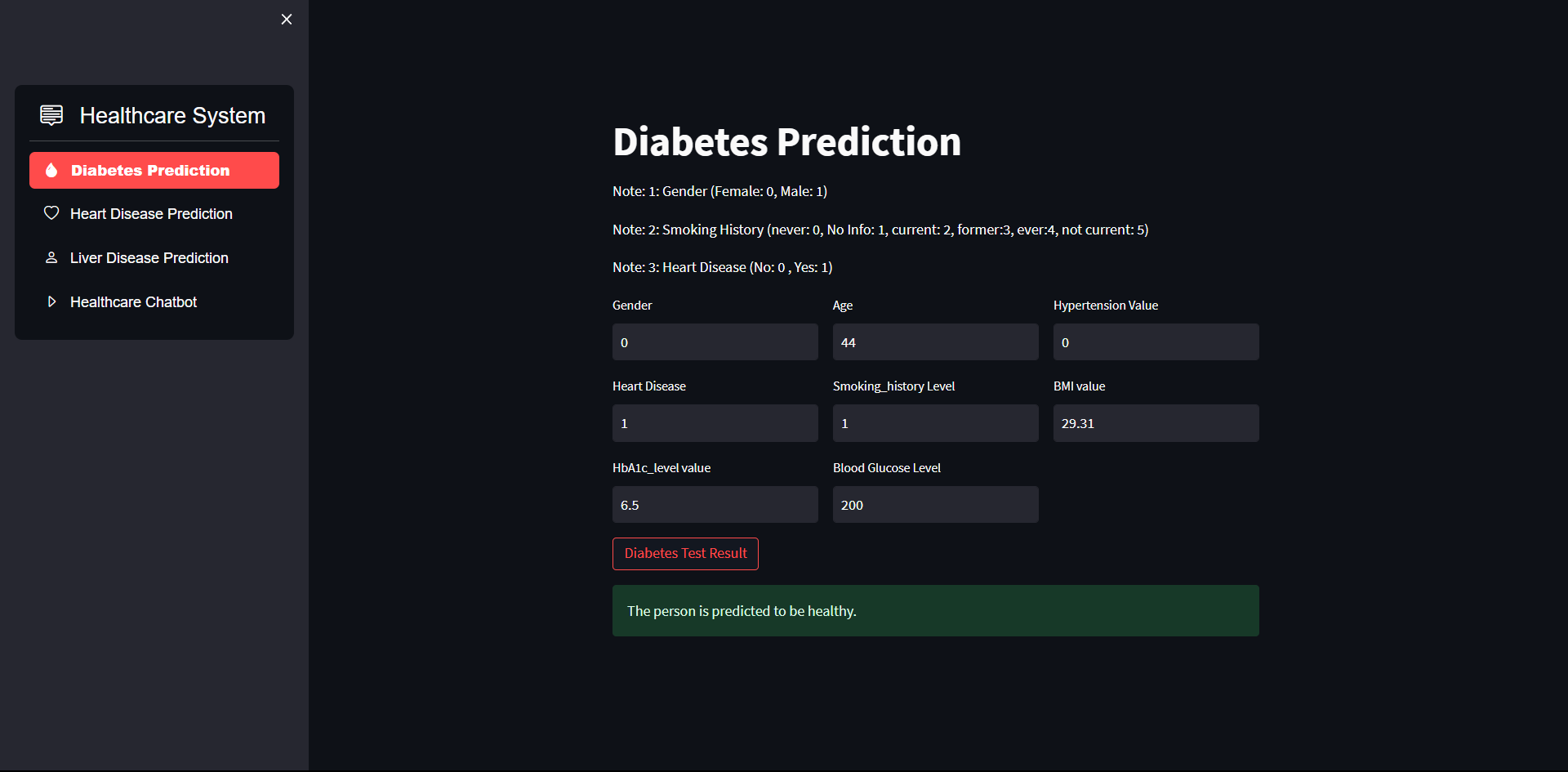
## 7.1 VARIOUS SNAPSHOTS



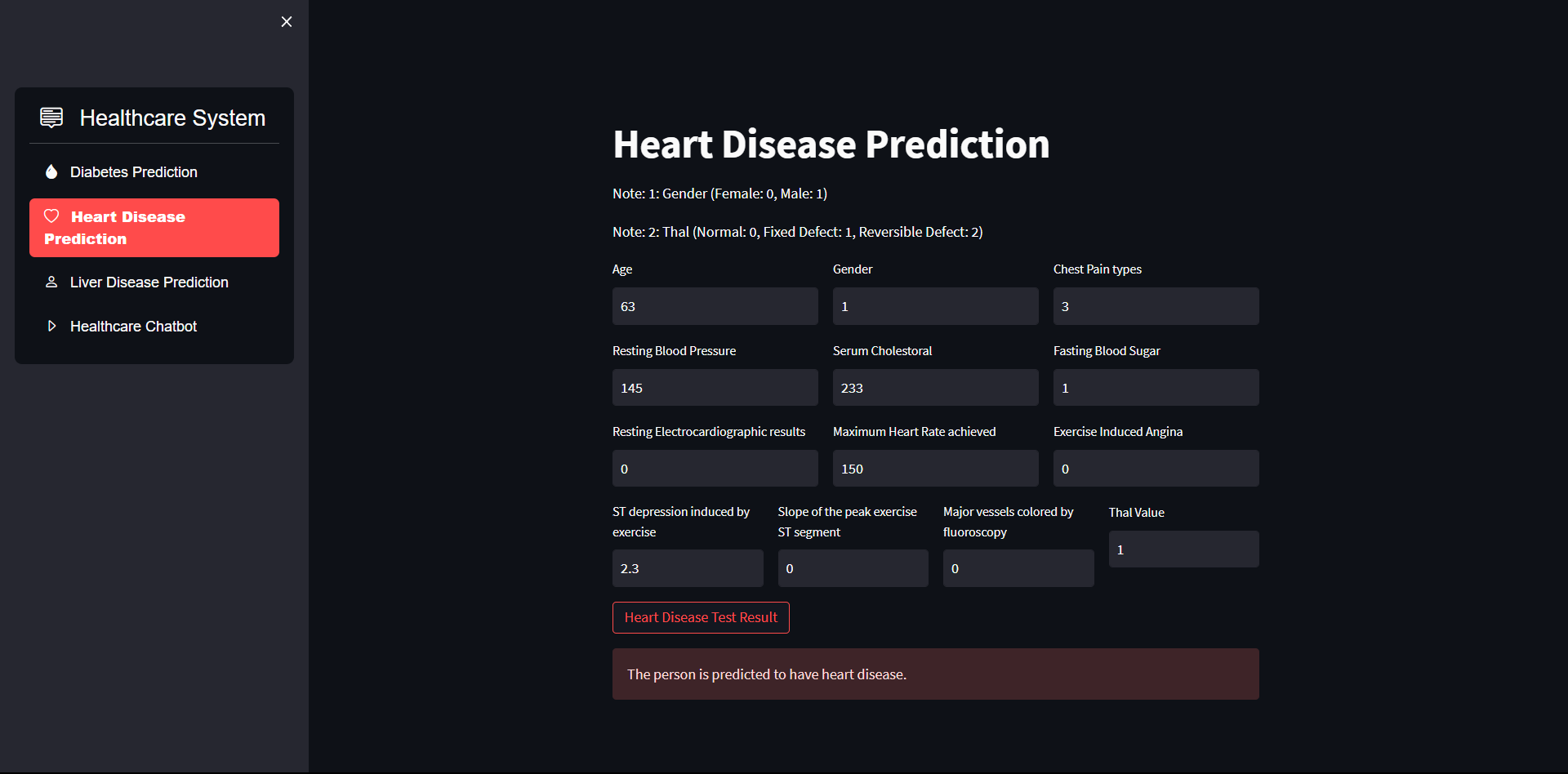
Snapshot 7.1: The Default Interface



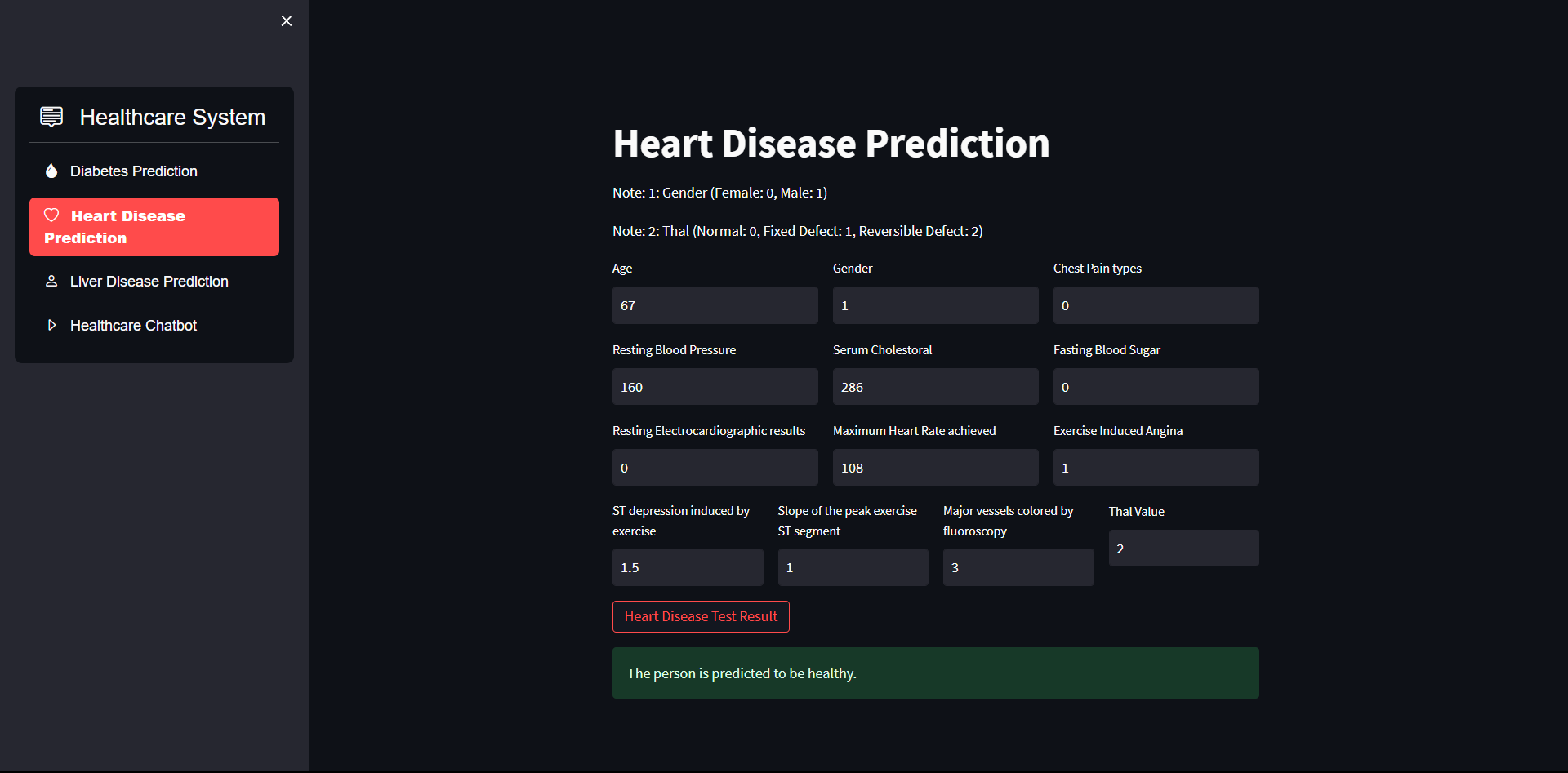
Snapshot 7.2: Diabetes Positive Prediction



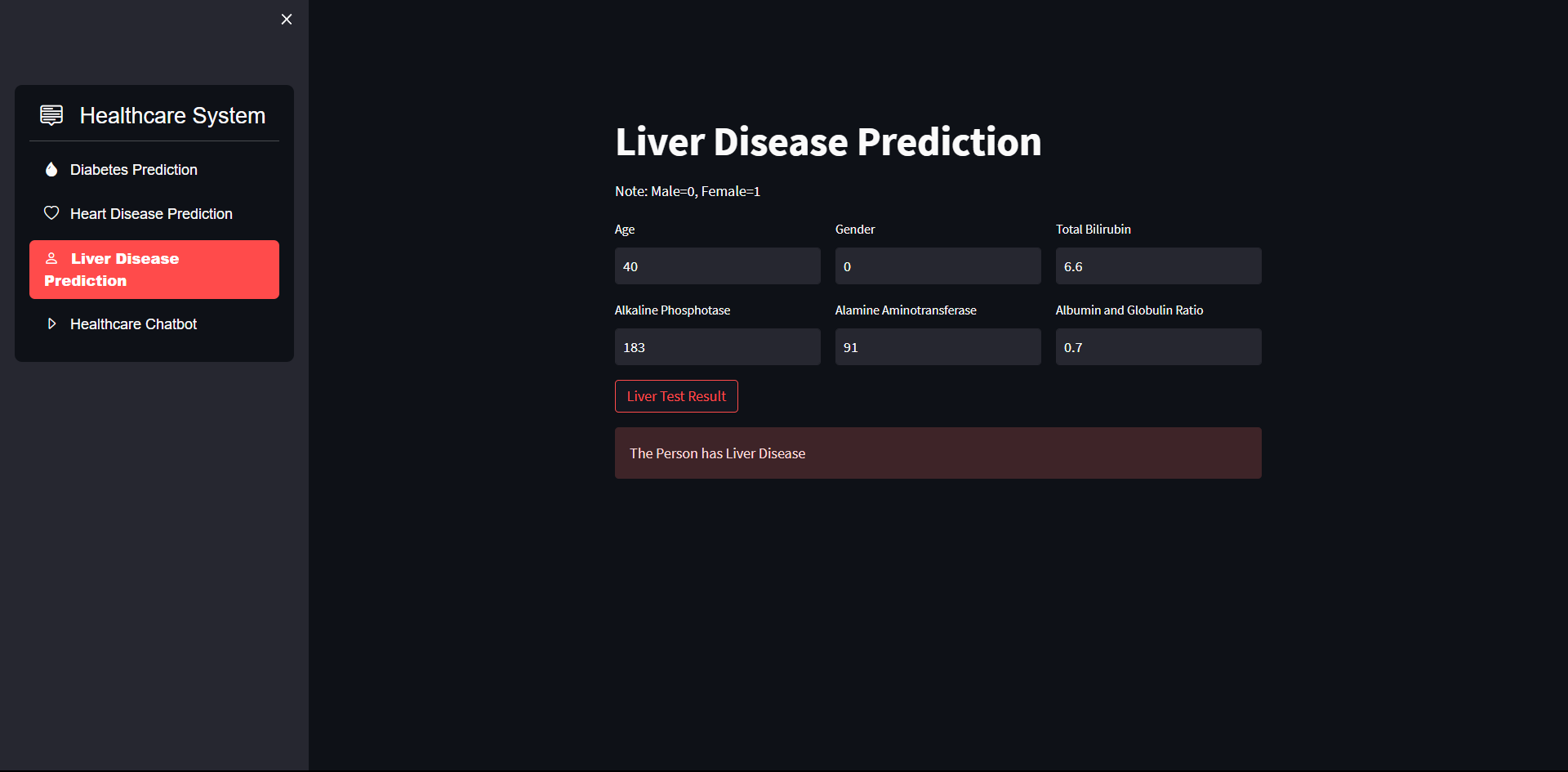
Snapshot 7.3: Diabetes Negative Prediction



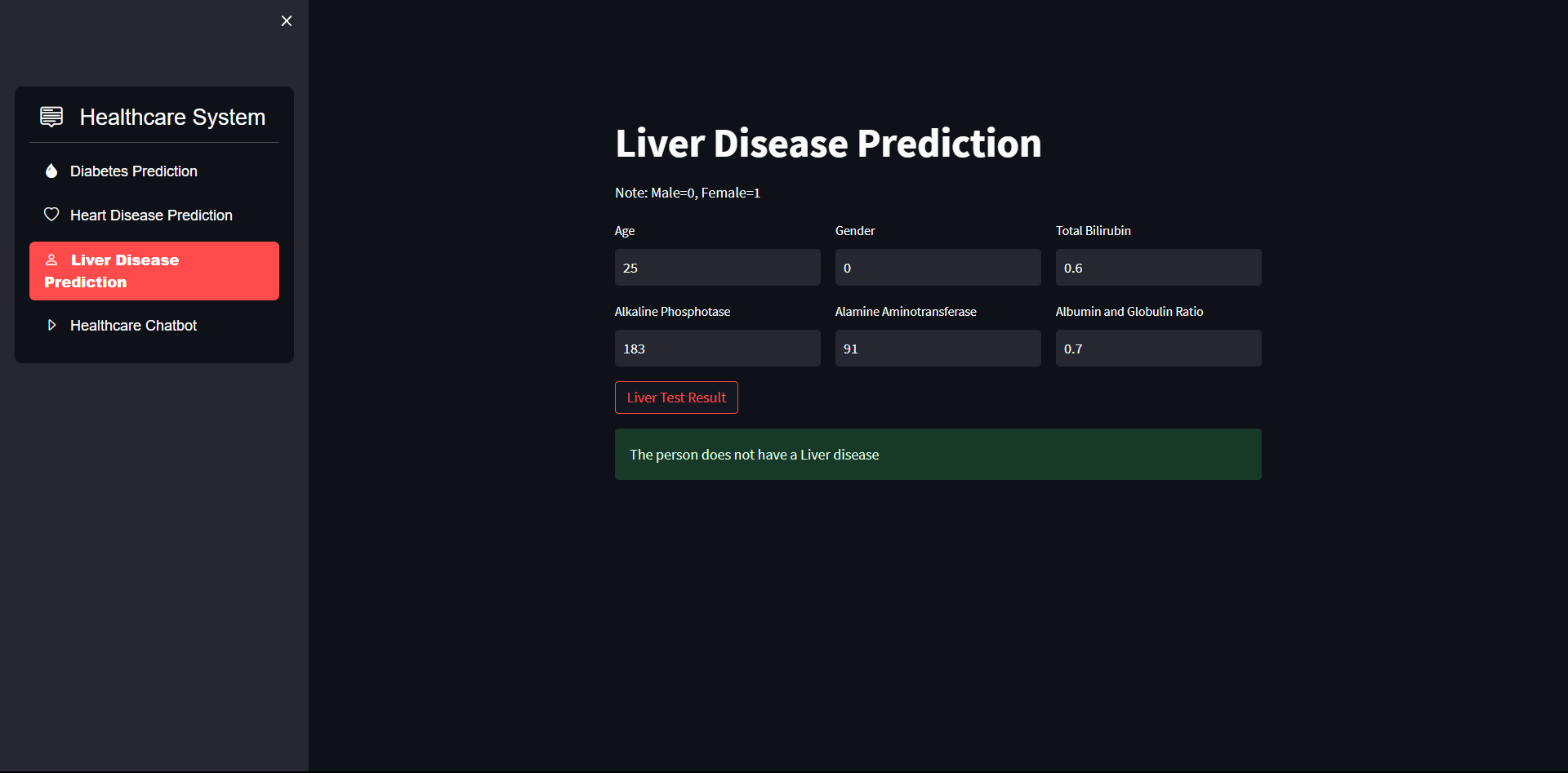
Snapshot 7.4: Heart Disease Positive Prediction



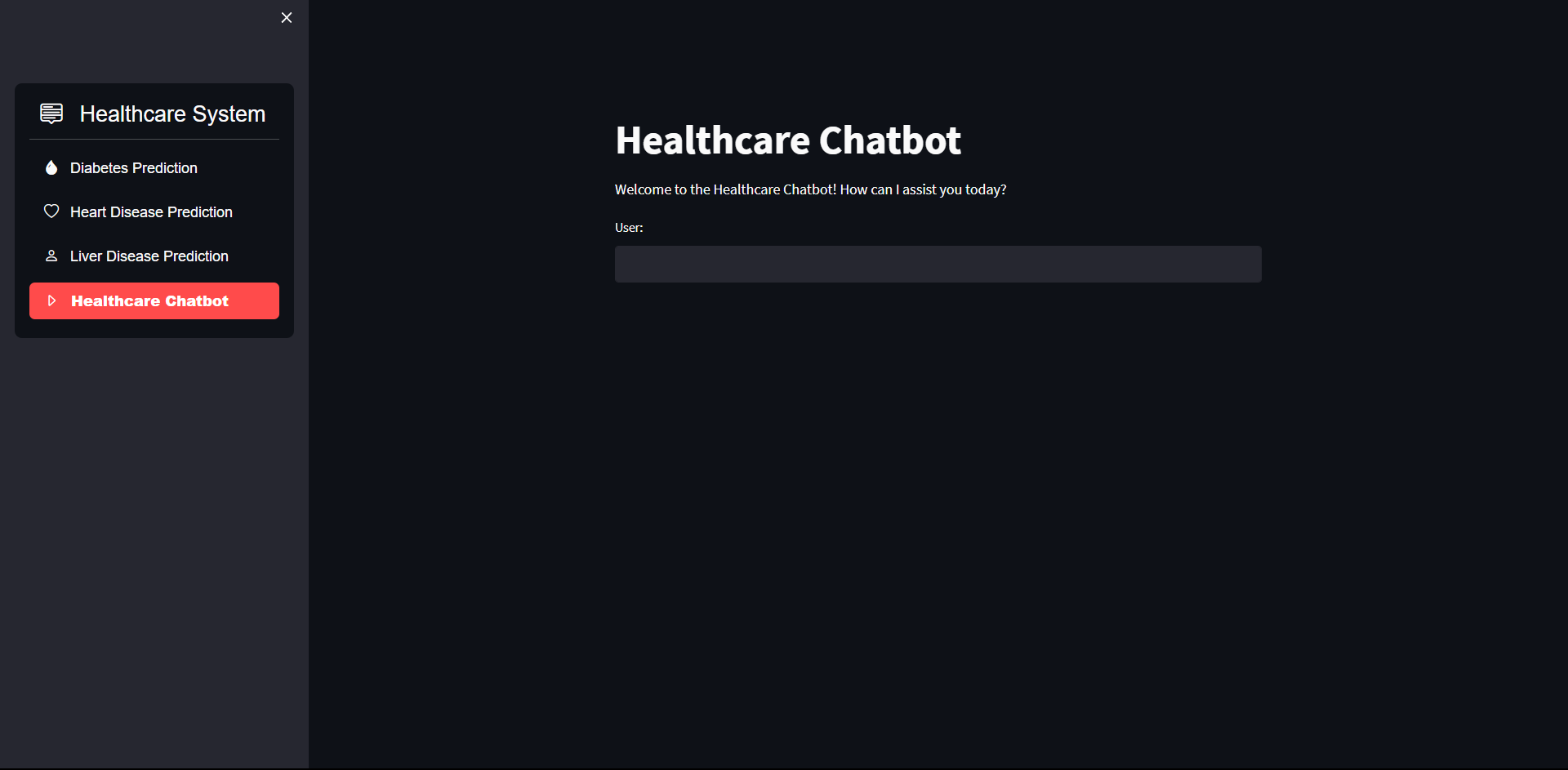
Snapshot 7.5: Heart Disease Negative Prediction



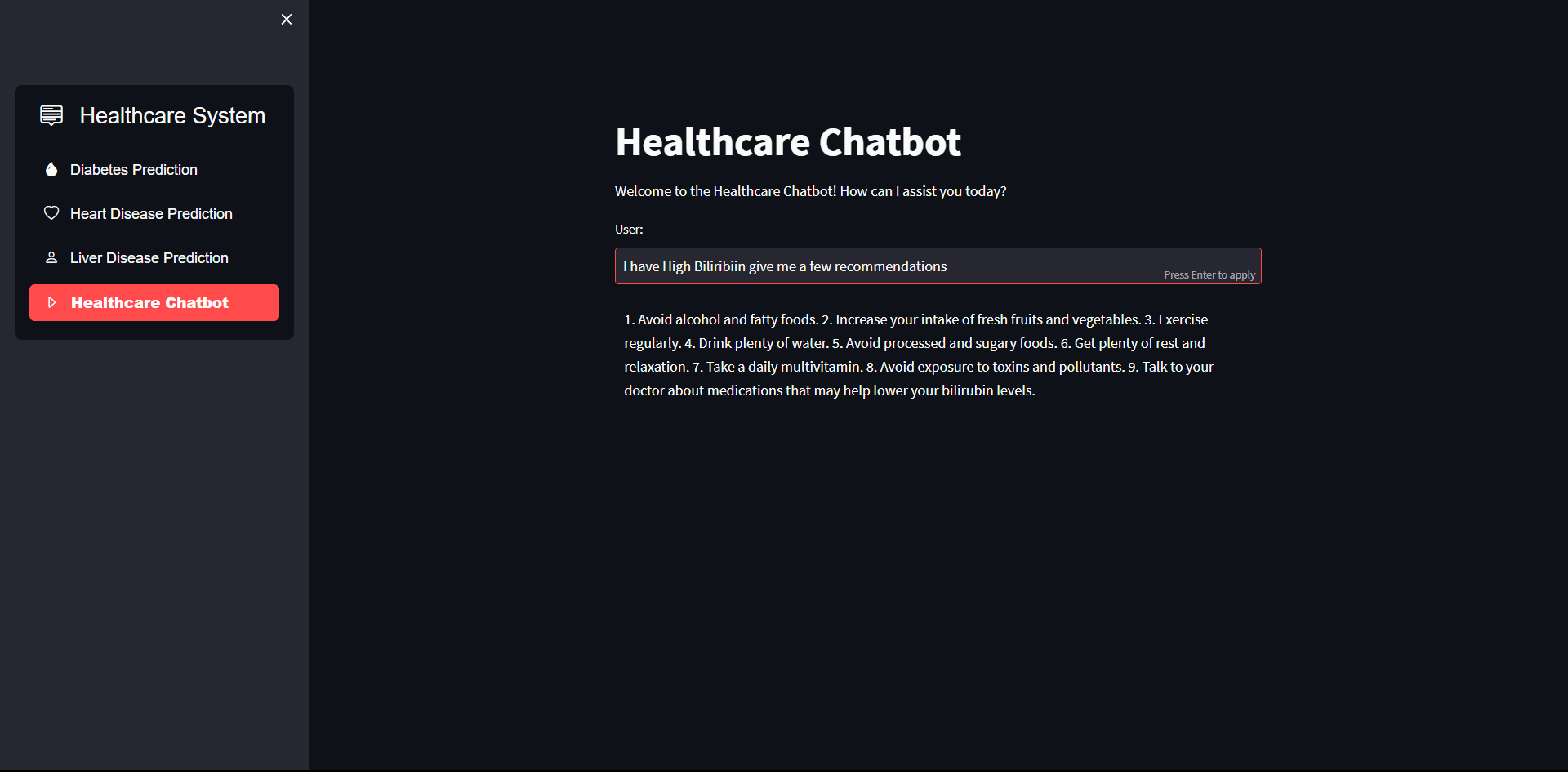
Snapshot 7.6: Liver Disease Positive Prediction



Snapshot 7.7: Liver Disease Negative Prediction



Snapshot 7.8: Healthcare Chatbot Interface



Snapshot 7.9: Healthcare Chabot Response

# **CHAPTER 8**

# **SOFTWARE TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

## 8.1 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

## 8.2 TYPES OF TESTING

### **UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### **FUNCTIONAL TESTING**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

### **SYSTEM TESTING**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configurationoriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **PERFORMANCE TESTING**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

### **INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

### **ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**ACCEPTANCE TESTING FOR DATA SYNCHRONIZATION:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need.
* The Status of Nodes information is done automatically in the Cache Updating process.

## 8.3 BUILD THE TEST PLAN

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

## TEST CASES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Test Description** | **Test Date** | **Expected Result** | **Actual Result** | **Remarks** |
| 1. | Diabetes Model Testing | 01/04/2023 | High Accuracy | Accuracy: 96.33% | Pass |
| 2. | Heart Disease Model Testing | 19/04/2023 | High Accuracy | Accuracy: 79.21% | Pass |
| 3. | Liver Disease Model Testing | 06/05/2023 | High Accuracy | Accuracy: 76.20% | Pass |
| 4. | Healthcare System Web Application Integration Testing | 13/05/2023 | High Accuracy | Diabetes Model: 96.33%  Heart Disease Model: 76.21%  Liver Disease Model: 76.20% | Pass |
| 5. | Integrating OpenAI GPT-3 Model into the Healthcare System using OpenAI API | 29/05/2023 | Successful Integration | Successful Integration | Pass |
| 6. | Training on Updated Datasets & Tuning | 02/06/2023 | Higher Accuracy | Diabetes Model: 97.20%  Heart Disease Model: 83.61%  Liver Disease Model: 81.25% | Pass |
| 7. | Evaluating system's performance under various load conditions. | 03/06/2023 | Good Performance | High Performance | Pass |
| 8. | Containerization of Web Application | 08/06/2022 | Composing Up  Images and  Pushing to Docker  Registry. | Successful  Composing Up of  Images and  Pushing to Docker  Registry with  Logs | Pass |
| 9. | Deployment on Cloud | 12/11/2022 | Registry Creation  and deploying of  Web App. | Successful Registry Creation and deployment of container. | Pass |
| 10. | Verifying the system's compatibility with different browsers, operating systems, or devices. | 14/11/2022 | Consistent Access the system using different browsers, operating systems, or devices. | System functions correctly and displays properly across various platforms, maintaining consistent behaviour and user experience. | Pass |

# **CHAPTER 9**

# **CONCLUSION & FUTURE ENHANCEMENT**

This chapter includes the Conclusion & Future Enhancement. With Conclusion defining the final inference from the model and Future Enhancement includes all the possible Future Enhancements that could be added to the current model.

## 9.1 CONCLUSION

In conclusion, the development of a Machine Learning-Based Healthcare System with Chatbot Assistance for Disease Prediction has the potential to significantly improve the accuracy of disease diagnosis, care and early intervention, as well as provide personalized health recommendations to users.

The use of XGBoost algorithm, NLP and/or OpenAI API Integration techniques will ensure accuracy, speed, resource utilization, stability, and reduced complexity.

The healthcare chatbot will allow for easy and personalized interaction with people, leading to a healthier and happier life.

## 9.2 FUTURE ENHANCEMENT

In future enhancements, there are several aspects to consider. Firstly, the utilization of XGBoost, a powerful machine learning algorithm, can provide advantages over neural networks in terms of interpretability, training speed, and handling small to medium-sized datasets. XGBoost's ability to handle feature importance and missing values makes it suitable for healthcare applications. Additionally, incorporating XGBoost into smart watches can enable real-time monitoring of health parameters and provide personalized health recommendations, enhancing the overall user experience and promoting proactive health management. Another area of improvement lies in patients' records management, where the implementation of a robust database system can efficiently store, retrieve, and update patient data, ensuring data integrity and accessibility for healthcare professionals. Lastly, enhancing the chatbot's interactivity by integrating natural language processing techniques can improve the accuracy and relevance of responses, providing more personalized and comprehensive healthcare support. These future enhancements can elevate the functionality, usability, and effectiveness of the healthcare system, leading to better patient outcomes and user satisfaction.

**CHAPTER 10**

# **BIBLIOGRAPHY**

The Bibliography includes the reference papers, reference books and Reference Websites we have referred to make this project a reality. Thereby acknowledging them.

## 10.1 REFERENCE PAPERS

1. Jahanzaib Latif, Chuangbai Xiao, Shanshan Tu, Sadaqat Ur Rehman, Azhar Imran, Anas Bilal, " Implementation and Use of Disease Diagnosis Systems for Electronic Medical Records Based on Machine Learning: A Complete Review", IEEE, 2021
2. Rim Doukha, Sidi Ahmed Mahmoudi, Mostapha Zbakh, Pierre Manneback, " Deployment of Containerized Deep Learning Applications in the Cloud", IEEE, 2021
3. Pham, T., Tran, T., Phung, D., Venkatesh, S., & Berk, M., "Predicting mental health disorders with heterogeneous data.", Journal of Biomedical Informatics, 2021
4. Sneha Grampurohit; Chetan Sagarnal, " Disease Prediction using Machine Learning Algorithms", IEEE, 2020
5. J Dutta, Yong Woon Kim, Dalia Dominic, " Comparison of Gradient Boosting and Extreme Boosting Ensemble Methods for Webpage Classification", 2020
6. Hongwei Chen, He Ai; Zhihui Yang, Weiwei Yang, Zhiwei Ye, Dawei Dong, “An Improved XGBoost Model Based on Spark for Credit Card Fraud Prediction", IEEE, 2020
7. Li, Y., Yu, Y., Duan, Y., & Tao, Z., "Using deep learning for tuberculosis detection in chest X-rays.", International Journal of Environmental Research and Public Health, 2020
8. Jain, R., & Jain, A., "Diagnosis of heart disease using machine learning techniques: A survey.", Artificial Intelligence Review, 2020
9. Behera, H. S., Hota, C., & Mohanty, S. N., "A survey on machine learning techniques for disease diagnosis.", Journal of Medical Systems, 2019
10. Jagannatha, A. N., & Yu, H., "Bidirectional RNN for medical event detection in electronic health records.", Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019
11. Rajkomar, A., Dean, J., & Kohane, I., "Machine learning in medicine.", New England Journal of Medicine, 380(14), 2018
12. Choudhury, P. R., & Kumar, A., "Predicting diabetes using machine learning techniques." Journal of Medical Systems, 2017
13. Rohan, B., & Leander, P., "Predicting Heart disease using KNN and filters" Journal of Medical Systems, 2017
14. Choi, E., Schuetz, A., Stewart, W. F., & Sun, J., "Using recurrent neural network models for early detection of heart failure onset.", Journal of the American Medical Informatics Association, 2016

## 10.2 REFERENCE BOOKS

* Software Engineering: A Practitioner's Approach, Authors: Roger S. Pressman, Bruce Maxim, Year: 2019
* Learning UML 2.0: A Pragmatic Introduction to UML, Author: Russ Miles, Year: 2007
* Machine Learning in Healthcare Fundamentals and Recent Applications, [Bikesh Kumar Singh](https://www.routledge.com/search?author=Bikesh%20Kumar%20Singh), [G.R. Sinha](https://www.routledge.com/search?author=G.R.%20Sinha), Year: 2016

## 10.3 REFERENCE WEBSITES

* https://www.foreseemed.com/blog/machine-learning-in-healthcare
* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6616181/
* <https://emeritus.org/blog/ai-ml-machine-learning-in-healthcare/>
* <https://xgboost.readthedocs.io/en/stable/>
* https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/