INTELLIMED: A WEB BASED PLATFORM FOR INTERPRETABLE DISEASE PREDICTION USING EXPLAINABLE AI

**A PROJECT REPORT**

***Submitted by***

**ANANDHAKUMAR P (512220205001)**

**DIVAGAR D (512220205003)**

**GANESH S (512220205004)**

**KEERTHI T (512220205009)**

***in partial fulfillment for the award of the degree***

***Of***

**BACHELOR OF TECHNOLOGY**

***in***

**INFORMATION TECHNOLOGY**

**SKP ENGINEERING COLLEGE,**

**TIRUVANNAMALAI - 606611**

****

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY – 2024**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“INTELLIMED: A WEB BASED PLATFORM FOR INTERPRETABLE DISEASE PREDICTION USING EXPLAINABLE AI”** is the bonafide work of **“Anandhakumar P (512220205001), Divagar D (512220205003), Ganesh S (512220205004), Keerthi T (512220205009)”** who carried out the project work under my supervision.

----------------------------------------- -----------------------------------------

**SIGNATURE** **SIGNATURE**

**Dr. V. Raji, M.E., Ph.D.,**  **Mr. L. Sankaran, M.E.,**

HEAD OF THE DEPARTMENT, SUPERVISOR,

DEPARTMENT OF IT, DEPARTMENT OF IT ,

SKP ENGINEERING COLLEGE, SKP ENGINEERING COLLEGE,

TIRUVANNAMALAI. TIRUVANNAMALAI.

**Submitted for the Project Viva-Voce Examination held on………………………**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We are very thankful to the management of **SKP Engineering College,**

**Tiruvannamalai** for giving us this wonderful opportunity to study in this college and utilize all the facilities to the fullest.

Our sincere thanks to the lord blessing on us for the successful

completion of our project. We proudly thank **Mr. K. KARUNANITHI, B.E., MBA,** Chairman, **Mr. K. V. ARANGASAMY,** Joint Secretary and **Mr. R. SAKTHI KRISHNAN, M.L., M.B.A., Ph.D.,** CEO of our college for providing the facility to do the project. We are grateful for our Principal **Dr.S.BASKARAN, M.E, Ph.D.,** for his constant support and encouragement to our project.

We express our sincere thanks to **Dr. V. RAJI, M.E., Ph.D.,** Head of the Department of Information Technology and Our Class Advisor **Ms. M. SAMHITHA, M.E,** for her guidance, timely suggestions and word motivation.

It is a Great pleasure to express our gratitude and thanks towards our project Guide **Mr. L. Sankaran, M.E.,** for his uninterruptable suggestions and words of improvements regarding this project, which played a major role in guiding us in our track.

We also extend our thanks to our Class Advisor and every member of the faculty who provided valuable academic guidance, and their co-operation to do this successful project. Finally, we would like to thank our parents, relatives and friends for their encouragement and enthusiastic co-operation.

## ABSTRACT

Explainable AI (XAI) enhances understanding of complex AI models by offering interpretable explanations for predictions, addressing biases for fair decision-making. In this project we proposed a user-friendly system for predicting and interpreting health conditions using ML models, prioritizing transparency and interpretability with LIME explanations. It involves deploying a web application that predicts and interprets health conditions through trained ML models and LIME for local interpretability, alongside a user-friendly interface for input and result visualization. By leveraging XAI tools, this project promotes fair and unbiased decision-making in healthcare by revealing factors influencing predictions. The system facilitates clearer comprehension of AI decision-making processes, ensuring transparency and trustworthiness in health condition predictions while providing interpretable insights for users.

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE NO NO**

**ABSTRACT iv**

**LIST OF FIGURES vii**

**LIST OF ABBREVIATIONS viii**

## INTRODUCTION 1

|  |  |  |
| --- | --- | --- |
| 1.1 | GENERAL | 1 |
| 1.2 | OBJECTIVE | 2 |
| 1.3 | SUMMARY | 2 |

1. **LITERATURE SURVEY 3**
   1. [GENERAL 3](#_3znysh7)
   2. [LITERATURE REVIEW 3](#_2et92p0)
   3. [SUMMARY 5](#_tyjcwt)
2. **SYSTEM ANALYSIS 6**
   1. GENERAL 6
   2. [EXISTING SYSTEM](#_1t3h5sf) 6
      1. [Existing System Architecture](#_4d34og8) 7
      2. Limitations of Existing System 7
   3. [PROPOSED SYSTEM](#_2s8eyo1) 8
      1. [Proposed System Architecture](#_17dp8vu) 9
      2. [Advantages of Proposed System 9](#_3rdcrjn)
   4. [SUMMARY 1](#_1ci93xb)2
3. **SYSTEM DESIGN AND IMPLEMENTATION 13**
   1. [GENERAL 1](#_26in1rg)3
   2. [LIST OF MODULES 13](#_lnxbz9)
   3. [MODULE DESCRIPTION 1](#_35nkun2)3
      1. Data Acquisition and Preprocessing 13
      2. Model Training and Evaluation 14
      3. Prediction and Explanation Generation 15
      4. Application Development and Deployment 15
   4. [SUMMARY 16](#_3as4poj)
4. **SYSTEM REQUIREMENTS 17**
   1. [GENERAL 1](#_1ksv4uv)7
   2. [SYSTEM REQUIREMENTS 1](#_44sinio)7
      1. Hardware Requirements 17
      2. [Software Requirements 1](#_2jxsxqh)7
   3. TECHNICAL SPECIFICATIONS 18
      1. [Python 1](#_1pxezwc)8
      2. LIME 20

[5.4 SUMMARY 2](#_z337ya)1

**6 SYSTEM ARCHITECTURE** 22

6.1 GENERAL 22

6.2 ARCHITECTURE DIAGRAM 23

**7 SYSTEM IMPLEMENTATION** 24

7.1 CODING 24

7.2 OUTPUT 27

**8 SYSTEM SECURITY** 29

8.1 DEEP LEARNING MODEL AND DESIGN 29

8.2 DATA MANAGEMENT AND PRIVACY 29

8.3 USER INTERFACE AND EXPERIENCE 29

8.4 FEASIBILITY STUDY 30

**9 SYSTEM TESTING 31**

* 1. BLACK BOX TESTING 31

9.2 WHITE BOX TESTING 32

**10 CONCLUSION AND FUTURE ENHANCEMENTS 34**

[**APPENDIX 1 SCREENSHOTS 3**](#_3j2qqm3)**5**

[**APPENDIX 2 SAMPLE CODING 3**](#_1y810tw)**8**

[**REFERENCES**](#_4i7ojhp) **4**[**7**](#_4i7ojhp)

## LIST OF FIGURES

**FIGURE NO TITLE PAGE NO**

* 1. Existing System Architecture 7
  2. Proposed System Architecture 9
  3. Use Case Diagram 10
  4. Class Diagram 11
  5. Sequence Diagram 11
  6. Data Acquisition and Preprocessing 14
  7. Model Training and Evaluation 14
  8. Prediction and Explanation Generation 15
  9. Application Development and Deployment 16

A1.1 Lung Disease Prediction Form 23

A1.2 Lung Disease Prediction Result 23

A1.3 Diabetes Prediction Form 24

A1.4 Diabetes Prediction Result 24

A1.5 Heart Disease Prediction Form 25

A1.6 Heart Disease Prediction Result 25

A7.1 Output – 1 27

A7.2 Output – 2 28

A7.3 Output - 3 28

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| API | Application Programming Interface |
| UI | User Interface |
| XAI | Explainable AI |
| LIME | Local Interpretable Model-agnostic Explanation |
| ML | Machine Learning |
| HTML | Hyper Text Markup Language |
| PCA | Principle Component Analysis |
|  |  |
|  |  |
|  |  |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Machine learning and AI have revolutionized healthcare industries by enabling advanced disease prediction techniques. By analyzing vast amounts of patient data, including medical records, diagnostic images, genetic information, and lifestyle factors, machine learning algorithms can identify patterns and trends that may indicate the likelihood of various diseases. These algorithms can predict diseases such as cancer, diabetes, heart disease, and infectious diseases with high accuracy, allowing for early detection and intervention. Additionally, AI-driven predictive models can assist healthcare providers in personalized treatment planning and risk stratification, ultimately improving patient outcomes and optimizing resource allocation in healthcare systems. In today's world, AI models are being black box and their web applications pose challenges in healthcare due to their lack of transparency and interpretability. Users struggle to understand how these models make predictions, leading to distrust and missed opportunities for early diagnosis and appropriate treatment. To solve this problem Explainable AI came into play. Explainable AI (XAI) refers to a set of techniques and methodologies aimed at enhancing the transparency and interpretability of artificial intelligence (AI) models. In healthcare systems, XAI plays a crucial role in elucidating the decision-making process of complex machine learning algorithms utilized for tasks such as disease prediction, diagnosis, and treatment recommendation. By providing interpretable explanations for AI-generated predictions and recommendations, XAI enables healthcare professionals and patients to better understand the underlying factors influencing the outcomes. This transparency fosters trust in AI-driven healthcare systems, facilitates collaboration between humans and machines, and empowers stakeholders to make informed decisions regarding patient care. In this proposed system, we utilize machine learning, XAI technique called Local Interpretable Model-agnostic Explanations (LIME) and a Python Flask Web Application to provide transparent disease prediction system in healthcare industry. The ML model makes predictions, while LIME offers interpretable explanations. The Flask web app offers a user-friendly interface for accessing and understanding predictions, promoting trust and collaboration in healthcare decision-making.

* 1. **OBJECTIVE**

To create a user-centric platform aimed at transforming disease prediction into a transparent and understandable process. Leveraging machine learning models in such a way that the system will be equipped to predict different diseases based on patient data.

The core focus will be on integrating Explainable AI techniques, such as **LIME** (Local Interpretable Model-Agnostic Explanations) with the ML model to provide clear and interpretable insights into the model's decision-making process.

* 1. **SUMMARY**

This project introduces a user-friendly system employing ML models and LIME explanations to predict and interpret health conditions transparently. Through a web application, it enables users to input data and visualize results, promoting fair decision-making in healthcare by revealing factors influencing predictions. This approach enhances understanding and trust in AI-driven healthcare solutions.

## CHAPTER 2

## LITERATURE SURVEY

## GENERAL

Literature survey gives the overall description of the reference papers that has been referred to design the application, using which the problems of the existing applications and technologies is identified. The methods to overcome such limitations are also recognized.

## LITERATURE REVIEW

## Ahmad Chaddad, Jihao Peng, Jian Xu, Ahmed Bouridane, “Survey of Explainable AI Techniques in Healthcare”, MDPI, 2023.

The paper discusses about provides a comprehensive overview of explainable AI (XAI) methods in healthcare. It discusses the increasing importance of interpretability in AI models, particularly in medical settings where decisions directly impact patient outcomes. The authors highlight various XAI techniques such as rule-based systems, model-agnostic methods, and post-hoc explanation approaches. They examine how these techniques address the need for transparency and trust in AI systems, emphasizing their potential to enhance clinical decision-making, patient understanding, and regulatory compliance. Additionally, the review discusses challenges and future directions in implementing XAI in healthcare, aiming to bridge the gap between AI capabilities and clinical practice.

## Deepti Sarawat, Prona Bhattacharyaa, Ashwin Verma, Vivek Kumar Prasad, Sudeep Tanwar, Gulshan Sharma, Pitshoou N. Bokoro, Ravi Sharma “Explainable AI for Healthcare 5.0: Opportunities and Challenges”, IEEE, 2022.

The research examines the extensive examination of research concerning explainable AI (XAI) in healthcare. It delves into diverse XAI techniques applicable in medical contexts, including rule-based systems, model-agnostic approaches, and post-hoc explanation methods. Moreover, it underscores the crucial significance of interpretability in healthcare AI models, highlighting their potential to optimize clinical decision-making, facilitate patient comprehension, and ensure adherence to regulatory standards. The review is anticipated to address obstacles and future trajectories in integrating XAI into healthcare, with the overarching goal of bridging the gap between AI capabilities and pragmatic clinical implementation, thereby fostering more transparent and effective healthcare practices.

## S. Khedkar, V. Subramanian, G. Shinde, and P. Gandhi, “Explainable AI in Healthcare”, SSRN Electronic Journal, 2019.

## The literature review in "Explainable AI in Healthcare" likely provides a comprehensive overview of existing research on explainable AI (XAI) within the healthcare sector. It likely explores various XAI techniques and their applications in medical contexts, such as rule-based systems, model-agnostic methods, and post-hoc explanation approaches. Furthermore, the review probably emphasizes the importance of interpretability in healthcare AI models, highlighting how it can enhance clinical decision-making, patient understanding, and regulatory compliance. Challenges and future directions in implementing XAI in healthcare are likely discussed, aiming to bridge the gap between AI capabilities and practical clinical application. Overall, the review serves to elucidate the role of XAI in healthcare and its potential impact on improving healthcare outcomes.

## 2.3 SUMMARY

This chapter gives a brief description about each paper referred and also explains the concept which is extracted from them and is implemented in the proposed system.

**CHAPTER 3**

**SYSTEM ANALYSIS**

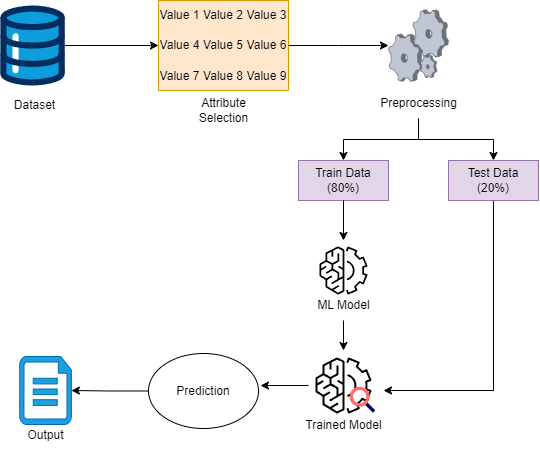
## GENERAL

System Analysis is important because it provides an avenue for solutions in the system through the various tasks involved in doing the analysis. This chapter explains about the existing system and how it works and also its disadvantages based on which the proposed system is designed.

## EXISTING SYSTEM

Current disease prediction systems typically present outputs as raw numerical values, lacking user-friendly formats for interpretation. This deficiency poses significant challenges for healthcare professionals in grasping prediction factors. The complexity of these numerical outputs hampers effective comprehension, thus limiting practicality in healthcare settings. To address these challenges, the proposed system prioritizes Explainable AI (XAI). By placing emphasis on interpretability, it endeavors to enhance user-friendliness and facilitate understanding of prediction processes for improved utility in healthcare contexts. Through techniques like LIME, the system aims to elucidate the rationale behind predictions, providing clear and accessible explanations for each output. This approach not only aids healthcare professionals in making informed decisions but also enhances collaboration between AI systems and practitioners. Ultimately, by promoting transparency and comprehension of model logic, the system seeks to bolster the efficacy and acceptance of predictive models in clinical practice.

## EXISTING SYSTEM ARCHITECTURE



**Figure 3.1 Existing system architecture**

## LIMITATIONS OF THE EXISTING SYSTEM

**Lack of Interpretability:** The existing system relies on presenting predictions in raw numerical formats, making it challenging for healthcare professionals and end-users to understand the factors contributing to the outcomes. This lack of interpretability hinders effective decision-making and may lead to mistrust in the system.

**Complexity:** The numerical output format adds complexity to the understanding of how the model reaches its predictions. Healthcare professionals may struggle to decipher the underlying logic or patterns within the data, reducing the system's utility in practical healthcare settings.

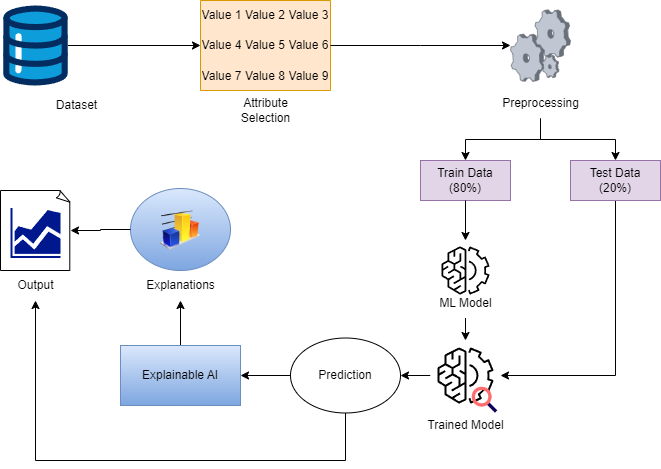
**Limited User-Friendliness:** The existing system's output format does not prioritize user-friendliness, which can impede its adoption and usage by healthcare practitioners. Without clear and intuitive presentation, users may find it cumbersome to navigate and interpret the predictions effectively.

**Ineffective Communication:** The numerical format may result in ineffective communication between AI models and healthcare practitioners. Misinterpretation or misunderstanding of the predictions due to their complexity can hinder collaboration and the exchange of valuable insights for better patient care.

## PROPOSED SYSTEM

This project integrates cutting-edge Explainable AI (XAI) techniques, notably LIME, to augment model interpretability and transparency. Through machine learning, it delivers precise disease predictions while circumventing the opaque nature of "black box" interpretations. LIME plays a pivotal role by facilitating clear visualization and explanation of model decisions for each prediction, thereby generating localized and easily understandable rationales. The user interface will showcase influential features that drive predictions, fostering collaborative comprehension between AI systems and healthcare practitioners. This approach aims to promote transparency and comprehension of the model's logic for both physicians and patients alike. By elucidating the reasoning behind predictions, this system empowers stakeholders to make informed decisions and build trust in AI-driven healthcare solutions, ultimately enhancing the efficacy and acceptance of predictive models in clinical practice.

## PROPOSED SYSTEM ARCHITECTURE



**Figure 3.2 Proposed system architecture**

## ADVANTAGES OF PROPOSED SYSTEM

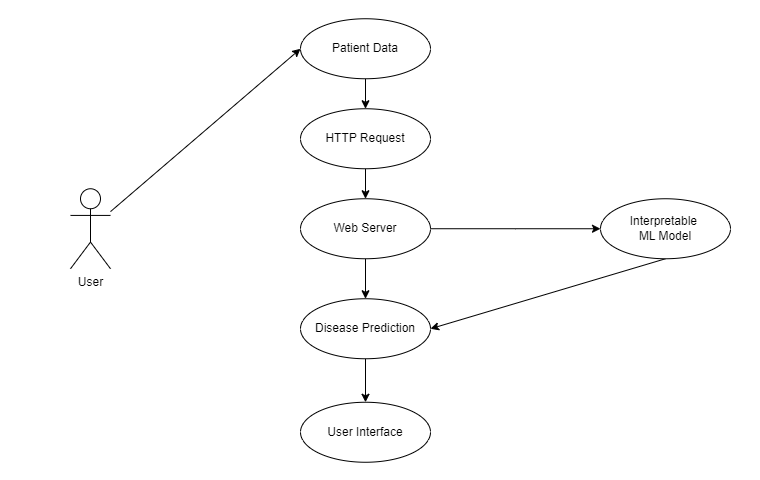
## Enhanced Transparency: The proposed system integrates Explainable AI techniques, such as LIME, to provide clear and interpretable insights into the prediction process. This transparency fosters trust among healthcare professionals and end-users by allowing them to understand how the model arrives at its predictions.

**Improved Interpretability:** By leveraging LIME, the proposed system offers human-interpretable explanations for individual predictions, enabling healthcare practitioners to grasp the influential features behind each prediction. This improved interpretability enhances decision-making and promotes a deeper understanding of the underlying data patterns.

**User-Friendly Interface:** The proposed system features an intuitive user interface with visualizations that highlight the influential features driving each prediction. This user-friendly design facilitates ease of use and enhances the system's accessibility for healthcare professionals, leading to more efficient and effective utilization.

**Collaborative Understanding:** Through clear and comprehensible insights provided by the proposed system, there is an opportunity for collaborative understanding between AI and healthcare practitioners. By fostering a shared understanding of the predictive models, the system promotes collaboration and facilitates the exchange of valuable insights for improved patient care.

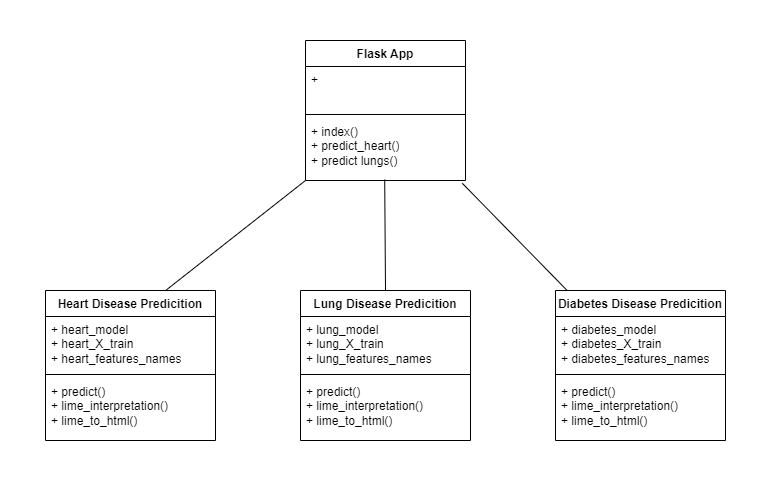
## USE CASE DIAGRAM FOR PROPOSED SYSTEM



**Figure 3.3 Use Case Diagram**

Describes how the user performs various functions from start to finish in the System.

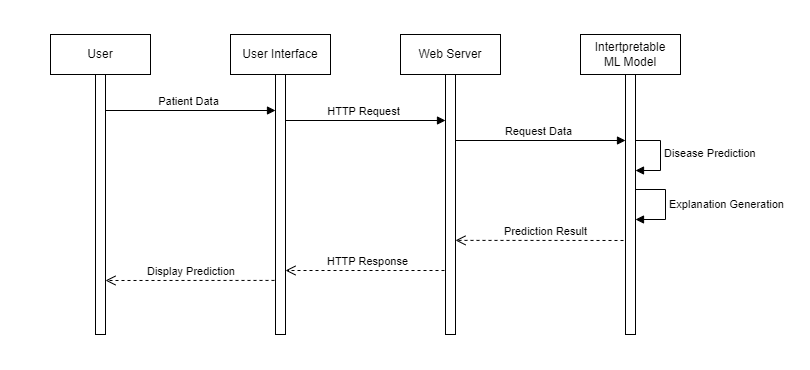
## CLASS DIAGRAM FOR PROPOSED SYSTEM



**Figure 3.4 Class Diagram**

Figure 3.4 depicts the Class Diagram which describes the user’s details and service requests as opposed to how the System responds to these requests with the help of the database.

## SEQUENCE DIAGRAM FOR PROPOSED SYSTEM



**Figure 3.5 Sequence diagram**

Figure 3.5 depicts the Sequence Diagram which describes the sequence of events occurring within the System with details as to how the System utilizes the database to deliver the various functionalities to the user.

## 3.4 SUMMARY

The UML Diagrams are an ideal representation of the design work that describes the working of the System in a general perspective. In that, the various categories of UML Diagrams describe the System in a unique manner such that all the aspects of the System are described, analyzed, rectified and finalized before it is translated in the form of working code. The purpose of representing the requirements in the form of such diagrams is to make sure that all the necessary changes can be made before the implementation process as doing so here will be effective in terms of cost and time.

## CHAPTER 4

## SYSTEM DESIGN AND IMPLEMENTATION

## GENERAL

This chapter describes the design phase of a project which is divided into various modules. It briefs about system design of each module along with its functionalities.

## LIST OF MODULES

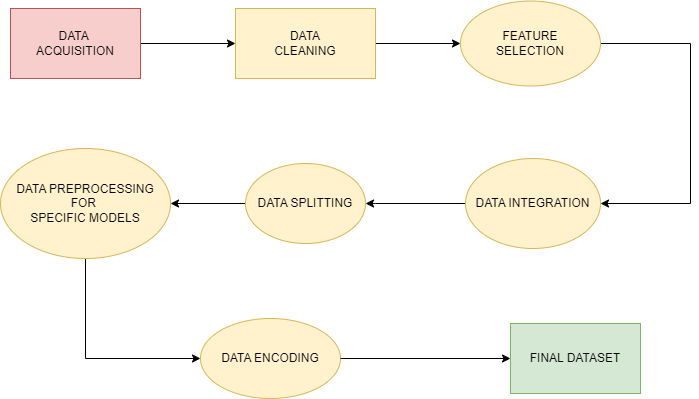
* + Data Acquisition and Preprocessing
  + Model Training and Evaluation
  + Prediction and Explanation Generation
  + Application Development and Deployment

## MODULE DESCRIPTION

The modules are performed using different techniques and they are described below with the help of illustrations.

## DATA ACQUISITION AND PREPROCESSING

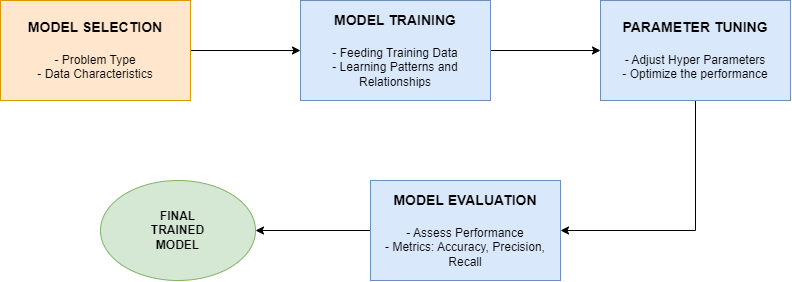
Data collection entails gathering relevant data from various sources such as databases, surveys, or APIs. Subsequently, data cleaning involves identifying and addressing missing values, inconsistencies, and errors in the data. Feature engineering selects and transforms the data into meaningful features for the chosen machine learning model. Finally, data splitting divides the prepared data into training and testing sets, ensuring the model doesn't overfit the training data.



**Fig 4.1 Data Acquisition and Preprocessing**

## MODEL TRAINING AND EVALUATION

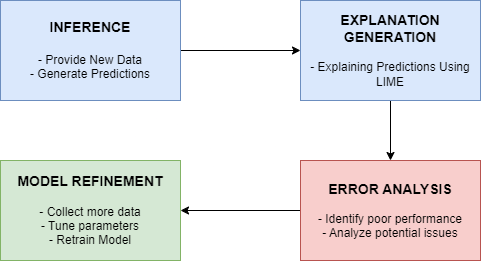
Model selection is critical, requiring the choice of a suitable machine learning model based on the problem and data characteristics. Subsequently, model training involves feeding the training data to the selected model, enabling it to learn patterns. Parameter tuning fine-tunes hyperparameters to optimize performance and prevent overfitting or underfitting. Finally, model evaluation rigorously assesses performance using diverse metrics such as accuracy, precision, and recall on the testing set to ensure effectiveness and reliability.



**Figure 4.2 Model Training and Evaluation**

## PREDICTION AND EXPLAINATION GENERATION

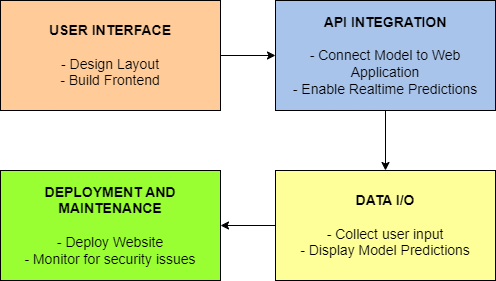
## Inference entails providing new data points to the trained model, prompting it to generate predictions based on learned knowledge. Explanation generation utilizes techniques like LIME importance to clarify prediction reasoning, bolstering transparency and trust. Error analysis examines instances of poor model performance, identifying potential issues and improvement opportunities. Model refinement, guided by evaluation and analysis, may involve collecting more data, fine-tuning parameters, or experimenting with different algorithms for enhancement.



**Figure 4.3 Prediction and Explanation Generation**

## APPLICATION DEVELOPMENT AND DEPLOYMENT

## User Interface (UI) Design focuses on crafting an intuitive and user-friendly interface for users to interact seamlessly with the model. API Integration seamlessly incorporates the trained model's functionalities into web applications, facilitating real-time predictions. Data Input and Output mechanisms enable users to conveniently input data and receive predictions, potentially accompanied by explanations. Deployment and Maintenance entail launching the web application on a server and ensuring its continuous operation and security measures are upheld.



**Figure 4.4 Application Development and Deployment**

## 4.4 SUMMARY

The various modules needed to design, develop and implement the proposed systems is summarized.

## CHAPTER 5

## SYSTEM REQUIREMENT

## GENERAL

This chapter explains about the software and hardware requirements and their specifications. The requirements listed here are used to satisfy system design and other implementation design.

## SYSTEM REQUIREMENTS

* + 1. **HARDWARE REQUIREMENTS**

The hardware components are used to develop the systems and to achieve the objectives of the system.

* + - * System : Intel i3 5th Generation or Higher
      * Hard Disk : 500 GB
      * Monitor : 14” LCD monitor display
      * Ram : 8 GB

## SOFTWARE REQUIREMENTS

The software requirements of the system can also be enlisted in terms of that is used to achieve the objectives of the System and those that will assist the former.

* + - * Operating system : Windows 10 or Higher
      * Programming Language : Python 3.12
      * Tool : Visual Studio Code , Anaconda ,

Jupyter Notebook

## TECHNICAL SPECIFICATION

## Python

Python is a general-purpose interpreted, interactive, object-oriented, and high level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is an interpreted high-level programming language for general-purpose programming. Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Under the Python release. Python v2.0 (October 2000), Python v2.6.x and 2.7.x versions, Python 3.0 (December 2008).

## Features of Python

Python's features include –

## 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.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

## It supports functional and structured programming methods as well as OOP.

## It can be used as a scripting language or can be compiled to byte-code for building large applications.

## It provides very high-level dynamic data types and supports dynamic type checking.

## IT supports automatic garbage collection.

## It can be very easily integrated with C, C++, Java, etc.

## LIME (Local Interpretable Model-agnostic Explanation)

LIME stands for "Local Interpretable Model-agnostic Explanations." It's a technique used in machine learning for explaining the predictions of complex models. LIME aims to provide human-interpretable explanations for individual predictions made by a black-box model, such as a deep neural network.

Here's how LIME typically works:

**Select Instance:** Choose a specific instance from the dataset for which you want an explanation of the model's prediction.

**Generate Perturbations:** Perturb the features of the selected instance by making small changes while keeping the target instance similar to the original one. This could involve adding noise or making slight alterations to the features.

**Model Prediction:** Use the black-box model to predict outcomes for the perturbed instances.

**Fit Interpretable Model:** For each perturbed instance, fit an interpretable model (such as linear regression or decision trees) to explain the predictions made by the black-box model.

**Weighting:** Weight the explanations based on the proximity of the perturbed instances to the original instance.

**Interpretation:** Use the fitted interpretable model to understand the contributions of different features to the prediction of the original instance.

LIME provides explanations at a local level, meaning it explains the prediction for a single instance rather than the model's global behavior. It's agnostic to the underlying black-box model, which means it can be applied to any model without needing to know its internal workings. This makes it particularly useful for complex models where interpretability is challenging. LIME has found applications in various domains, including healthcare, finance, and natural language processing.

## SUMMARY

This chapter summarizes about the usage of technologies. It also explains about the associated entities which are used for the implementation.

**CHAPTER 6**

**SYSTEM ARCHITECTURE**

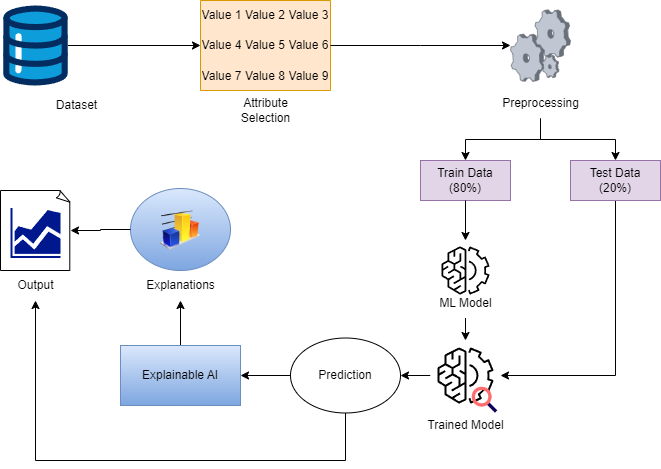
#### **6.1 GENERAL**

The architecture of "IntelliMed: A Web-Based Platform for Interpretable Disease Prediction Using Explainable AI" comprises several key layers that work in concert to provide a robust and user-friendly experience. At the forefront is the User Interface Layer, which presents the web interface allowing users to interact with the platform.

This layer seamlessly integrates with the Presentation Layer, responsible for rendering and managing the user interface elements to ensure a smooth user experience. Beneath these frontend layers lies the Application Layer, where the core logic of the platform resides. Here, user requests are processed, data is managed, and machine learning models for disease prediction are invoked. The Explainable AI Model Layer houses interpretable machine learning models chosen specifically for their ability to provide transparent and understandable predictions. These models are trained and updated within the Model Training and Update Layer, utilizing historical patient data to optimize their performance.

The Interpretation Generation Layer plays a crucial role in generating explanations for the predictions made by the AI models, enhancing transparency and trust in the platform. Security measures are enforced by the Security Layer, ensuring the confidentiality and integrity of patient data through user authentication, data encryption, and access control mechanisms. Scalability is addressed by the Scalability Layer, which ensures that the platform can handle increasing user loads and data volumes efficiently. Finally, the Monitoring and Logging Layer provides real-time monitoring of system performance, logs user activities, and detects any anomalies or security breaches, contributing to the overall reliability and stability of the platform.

**6.2 ARCHITECTURE DIAGRAM**

****

**Figure 6.2 Architecture Diagram**

**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

#### **7.1 CODING**

from flask import Flask, render\_template, request

import joblib

import numpy as np

from lime import lime\_tabular

import pandas as pd

app = Flask(\_name\_)

# Load the heart disease model

heart\_model\_filename = "logistic\_regression\_model.joblib"

diabetes\_model = joblib.load("diabetes.joblib")

diabetes\_X\_train = pd.read\_csv("diabetes\_train.csv")

diabetes\_feature\_names = [

'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age'

]

try:

heart\_model = joblib.load(heart\_model\_filename)

except Exception as e:

print(f"Error loading the heart disease model: {e}")

raise

heart\_feature\_names = [

'Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',

'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina',

'Oldpeak', 'ST\_Slope'

]

# Load training data for heart disease from CSV

try:

heart\_X\_train = pd.read\_csv("heart\_X\_train.csv")

except FileNotFoundError:

print("heart\_X\_train.csv not found. Please check the file path.")

raise

# Define Lime interpretation function for heart disease

def heart\_lime\_interpretation(features):

heart\_explainer = lime\_tabular.LimeTabularExplainer(

heart\_X\_train.values, # Use heart\_X\_train as a pandas DataFrame

feature\_names=heart\_feature\_names,

class\_names=['No Heart Disease', 'Heart Disease'],

discretize\_continuous=True

)

# Convert the input features to a NumPy array

heart\_features\_array = np.array(features, dtype=float)

heart\_explanation = heart\_explainer.explain\_instance(

heart\_features\_array,

heart\_model.predict\_proba,

num\_features=len(heart\_features\_array)

)

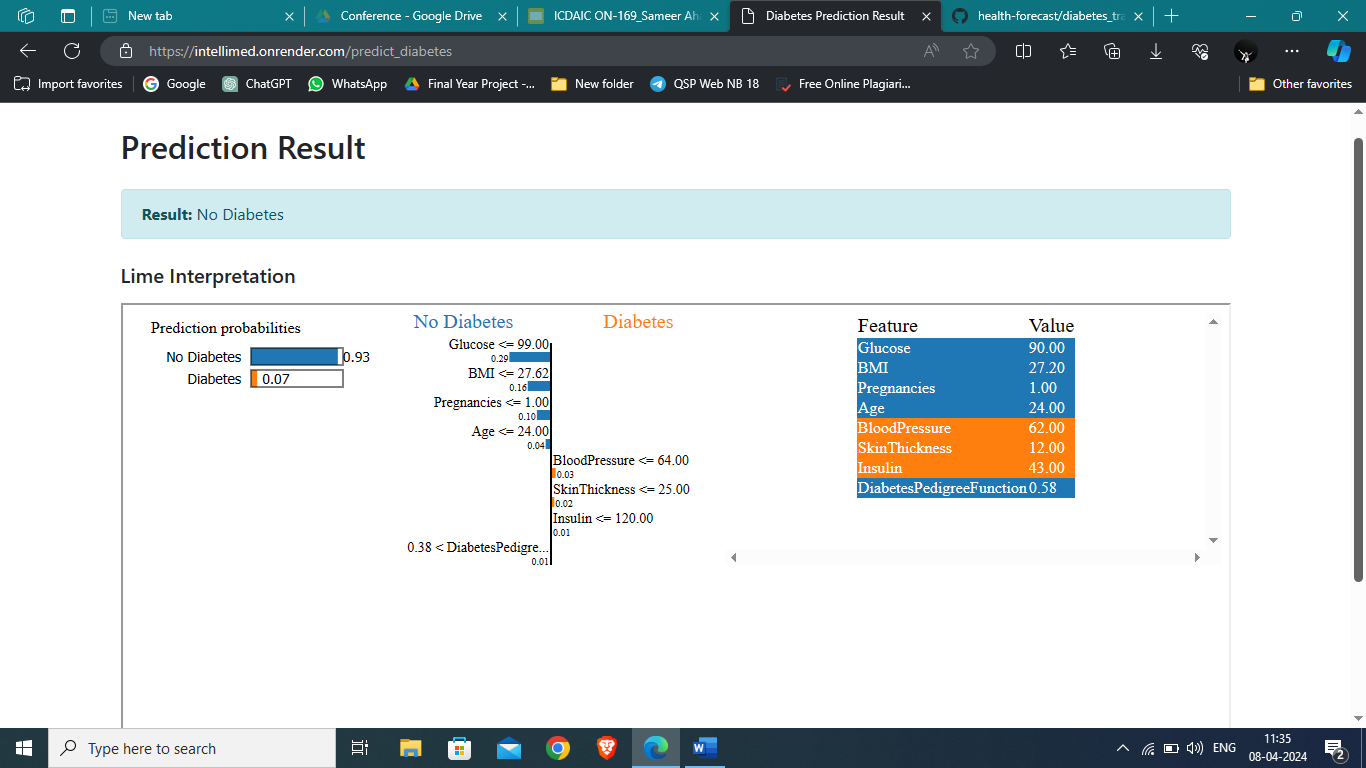
return heart\_explainer, heart\_explanation

# Function to convert Heart Lime explanation to HTML format

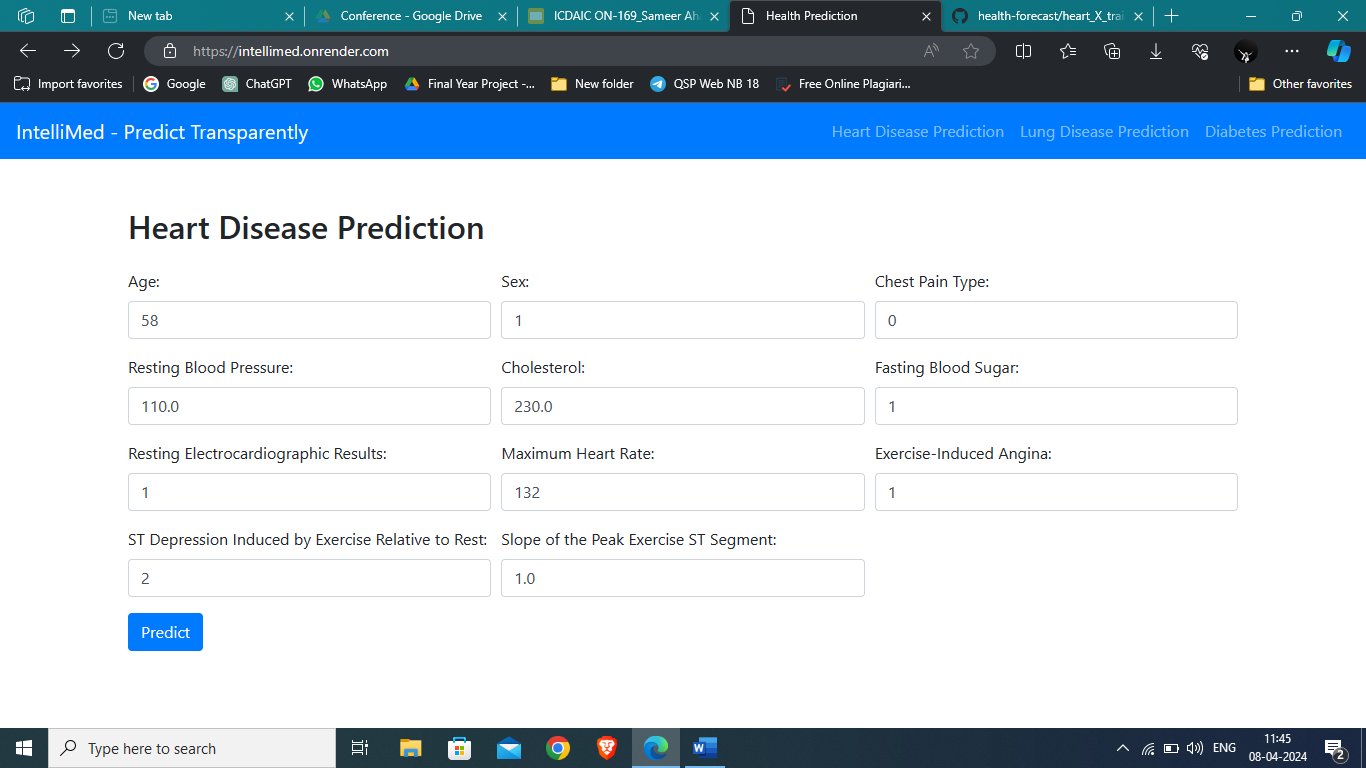
def heart\_lime\_to\_html(heart\_explanation):

return heart\_explanation.as\_html()

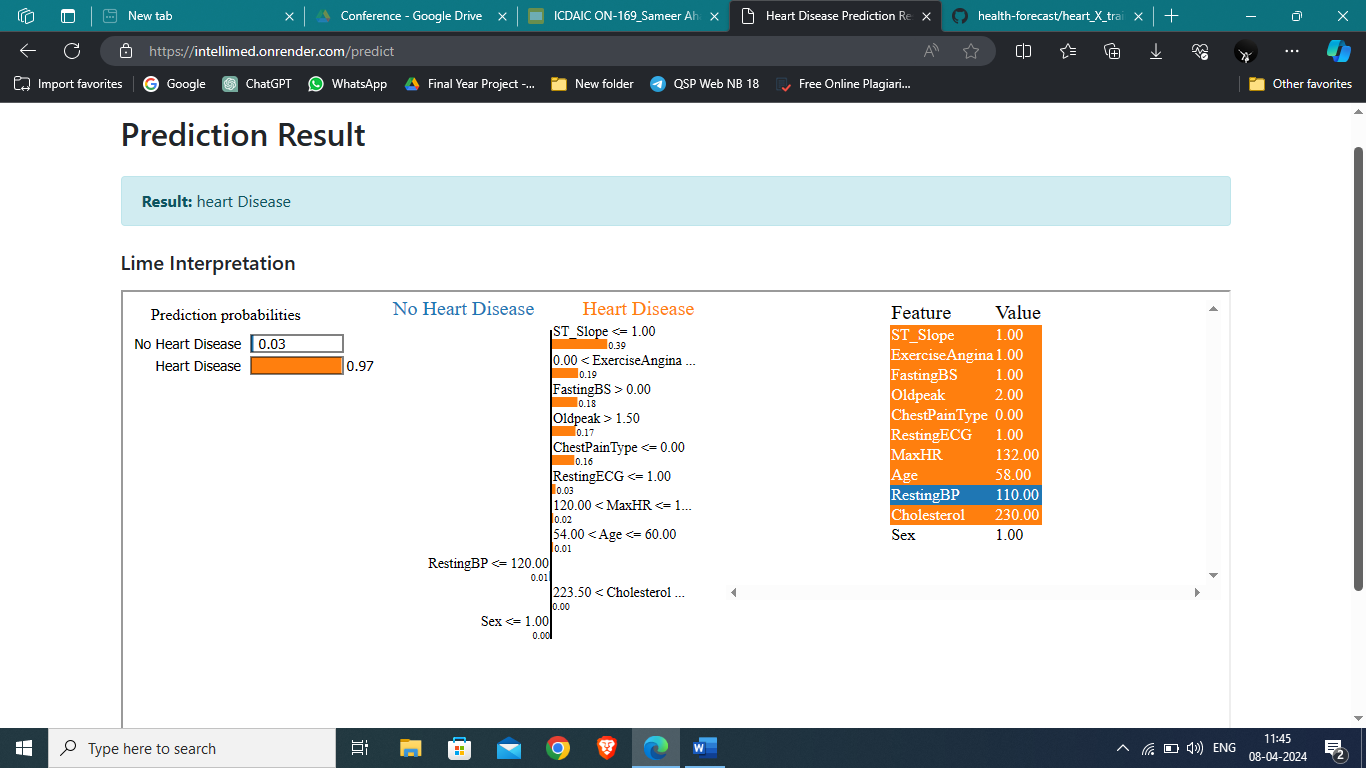
**7.2 OUTPUT**

****

A7.1 – OUTPUT-1



A7.2 – OUTPUT - 2



A7.3 OUTPUT - 3

**CHAPTER 8**

**SYSTEM SECURITY**

#### **8.1 DEEP LEARNING MODEL DESIGN**

This involves selecting and implementing a suitable deep learning architecture for image recognition and classification. Considerations include model accuracy, complexity, and training time. Models like ResNet, VGG, and Inception are popular choices for image processing tasks. This topic also encompasses data augmentation and preprocessing techniques to improve model robustness.

#### **8.2 DATA MANAGEMENT AND PRIVACY**

Handling user data requires careful consideration of privacy laws and best practices. This includes data collection, storage, and sharing. Compliance with regulations like GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) is crucial. This topic also covers anonymization and user consent to ensure data is used ethically and legally**.**

#### **8.3 USER INTERFACE AND EXPERIENCE**

A user-friendly interface is essential for a successful system. This topic involves designing intuitive user workflows, clear navigation, and accessibility features. The use of front-end technologies like HTML, CSS, and JavaScript ensures a responsive and engaging user experience. User feedback and usability testing are crucial to refine the interface and meet user needs.

**8.4 FEASIBILITY STUDY**

Conducting a feasibility study for "IntelliMed: A Web-Based Platform for Interpretable Disease Prediction Using Explainable AI" is crucial to assess its viability. This study will examine various aspects such as technical, economic, and operational feasibility to determine the project's feasibility. Firstly, it will evaluate the availability of necessary technology and expertise to develop and maintain the platform. Secondly, it will analyze the potential market demand and economic viability, considering factors like revenue generation and cost-effectiveness. Operational feasibility will assess the practicality of integrating the platform into existing healthcare systems and workflows. Furthermore, legal and regulatory compliance, including privacy laws like GDPR and healthcare regulations like HIPAA, will be scrutinized. The study will also consider potential risks and challenges and propose mitigation strategies to ensure the successful implementation of the project.

**CHAPTER 9**

### **SYSTEM TESTING**

#### **9.1 BLACK BOX TESTING**

Black box testing is a software testing technique where the internal workings of a system are not examined; instead, the focus is on testing the system's functionality based on its inputs and expected outputs. It is a common approach for validating whether a system meets specified requirements without delving into the underlying code or internal structures. For a medicinal plant identification system using image processing and deep learning, black box testing involves examining the system's behavior in response to various user interactions and ensuring it operates as intended.

1. **Functional Testing:**

This involves testing the core functionalities of the system, such as image upload, plant identification, and results retrieval. Test cases are designed to validate that the system correctly identifies medicinal plants from images and provides the expected results to the user.

1. **Boundary Testing:**

This involves testing the system's behavior at the boundaries of acceptable input values. In this context, boundary testing examines factors like image size, format, and resolution to ensure the system handles these variations properly**.**

1. **Error Handling:**

This involves testing how the system handles invalid or erroneous inputs. The goal is to ensure that the system provides appropriate error messages and does not crash or behave unexpectedly.

1. **User Interface Testing:**

This focuses on the usability and responsiveness of the system's user interface. It assesses whether users can easily navigate the system, upload images, and view results without issues.

1. **Security Testing:**

This involves testing the system's security features, such as authentication, authorization, and data encryption, to ensure user data is protected**.**

#### **9.2 WHITE BOX TESTING**

White box testing, also known as clear box or glass box testing, is a software testing approach where the tester examines the internal structure, logic, and code of a system. In a medicinal plant identification system using image processing and deep learning, white box testing focuses on the underlying algorithms, deep learning models, and codebase to ensure robustness and accuracy.

1. **Code Coverage Testing:**

This aspect ensures that the code is thoroughly tested, with a high percentage of code coverage. Testers create test cases to cover as much of the codebase as possible, including all branches and conditional statements. This helps identify untested or unreachable code sections, reducing the risk of hidden bugs**.**

1. **Path Testing:**

Path testing involves examining all possible execution paths in the code to ensure that every logical route is tested. This type of testing is particularly useful in complex algorithms, such as those used in image processing and deep learning, to ensure they function as expected in various scenarios.

1. **Unit Testing:**

White box testing typically includes unit tests that validate individual functions or components within the system. For this project, unit testing could focus on specific image processing functions, deep learning model methods, and database interactions, ensuring each component behaves as intended**.**

1. **Deep Learning Model Testing:**

This aspect involves testing the internal workings of the deep learning model, including verifying the model's training process, weights, and accuracy. It may involve analyzing the model's structure, activation functions, and layer outputs to ensure the model is learning and performing as expected.

1. **Integration Testing:**

White box testing extends to integration testing, where different components of the system are tested together. This ensures that data flows correctly between modules and that the integrated system functions as intended. In this project, integration tests could cover interactions between the user interface, back-end server, and deep learning model.

1. **Error Handling Testing:**

This involves testing how the system handles exceptions and errors at the code level. White box testing focuses on validating that the system gracefully handles errors and provides appropriate responses without crashing or causing unexpected behavior.

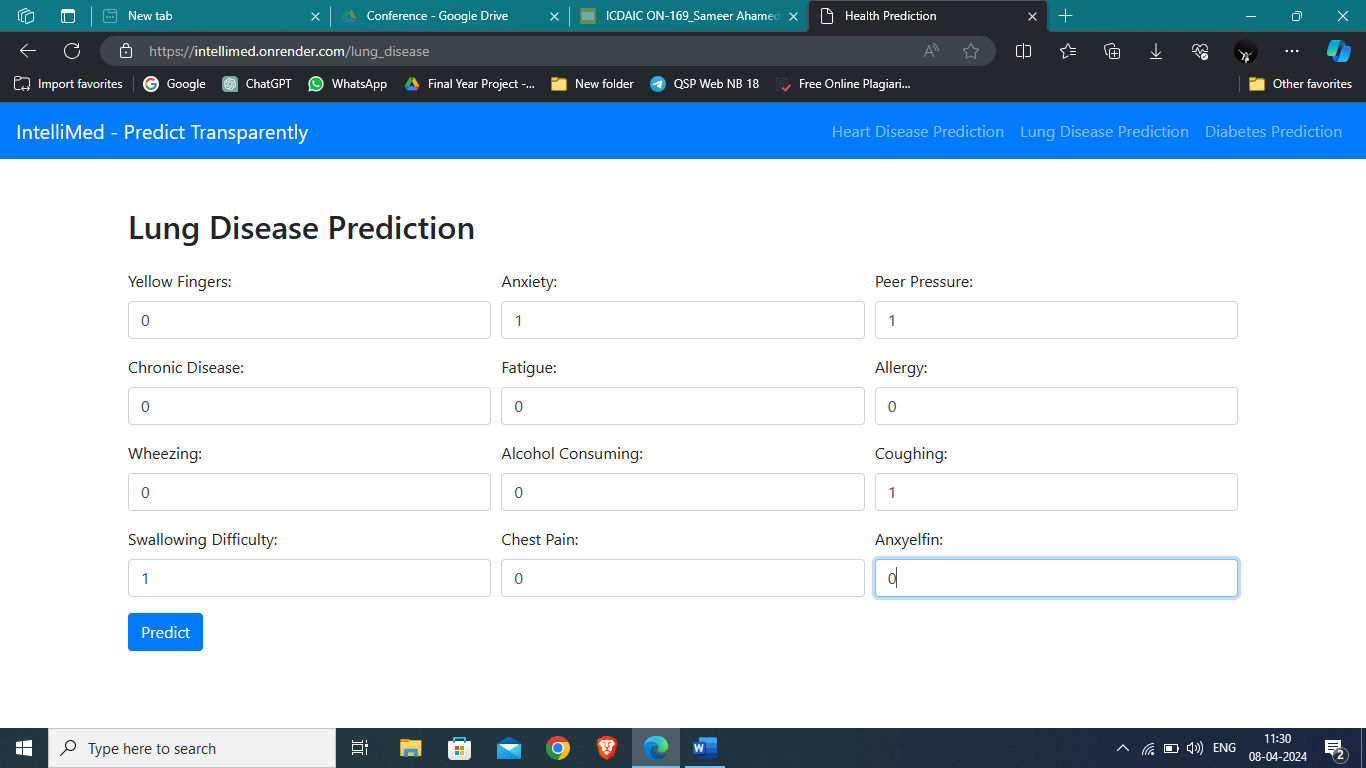
## CHAPTER 10

**CONCLUSION AND FUTURE ENHANCEMENT**

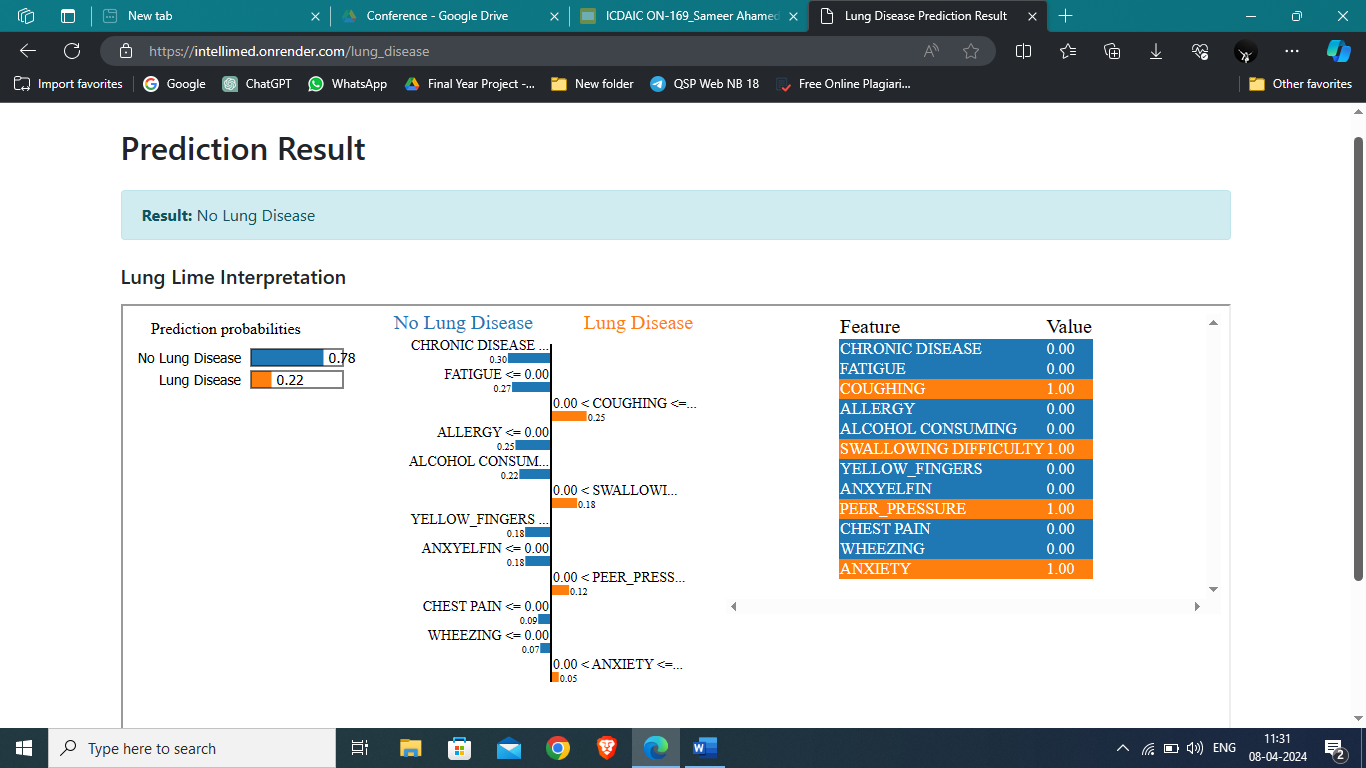
## In this project, we've developed an interpretable disease prediction model using explainable AI techniques. Leveraging advanced machine learning algorithms and interpretable models like decision trees and rule-based systems, we've achieved high transparency and interpretability in disease prediction. This not only ensures accurate predictions but also provides insights into influencing factors, fostering trust and understanding among users and healthcare professionals. Our model showcases explainable AI's potential in healthcare, where interpretability and transparency are paramount. To enhance our interpretable disease prediction model further, we aim to integrate larger and more diverse datasets for increased robustness. Exploring advanced techniques such as counterfactual explanations and model-agnostic methods will augment interpretability. Real-time data integration and continuous learning mechanisms will enable adaptation to evolving healthcare landscapes. Collaborating with healthcare stakeholders for deployment in clinical settings and conducting rigorous validation studies are crucial for realizing the model's potential impact on improving patient outcomes and healthcare decision-making.

## APPENDIX 1

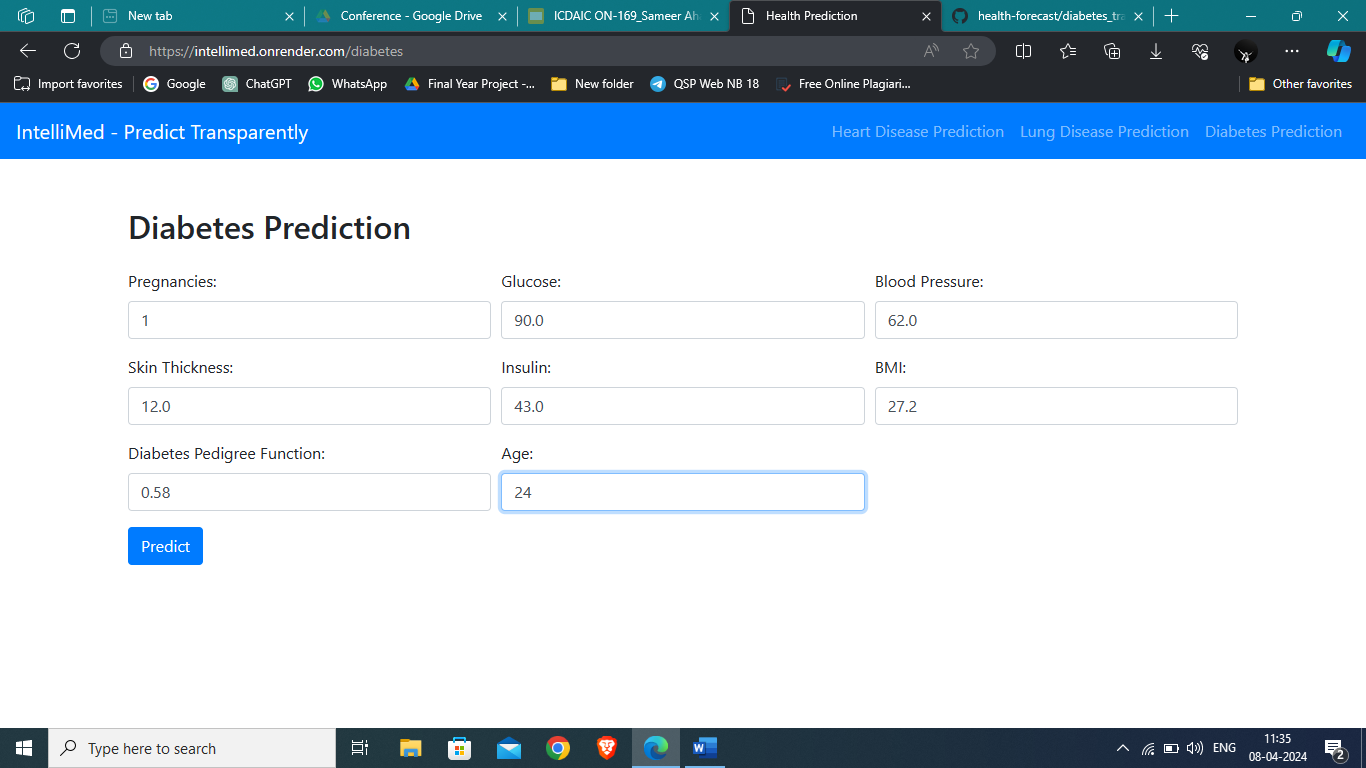
## SCREENSHOTS



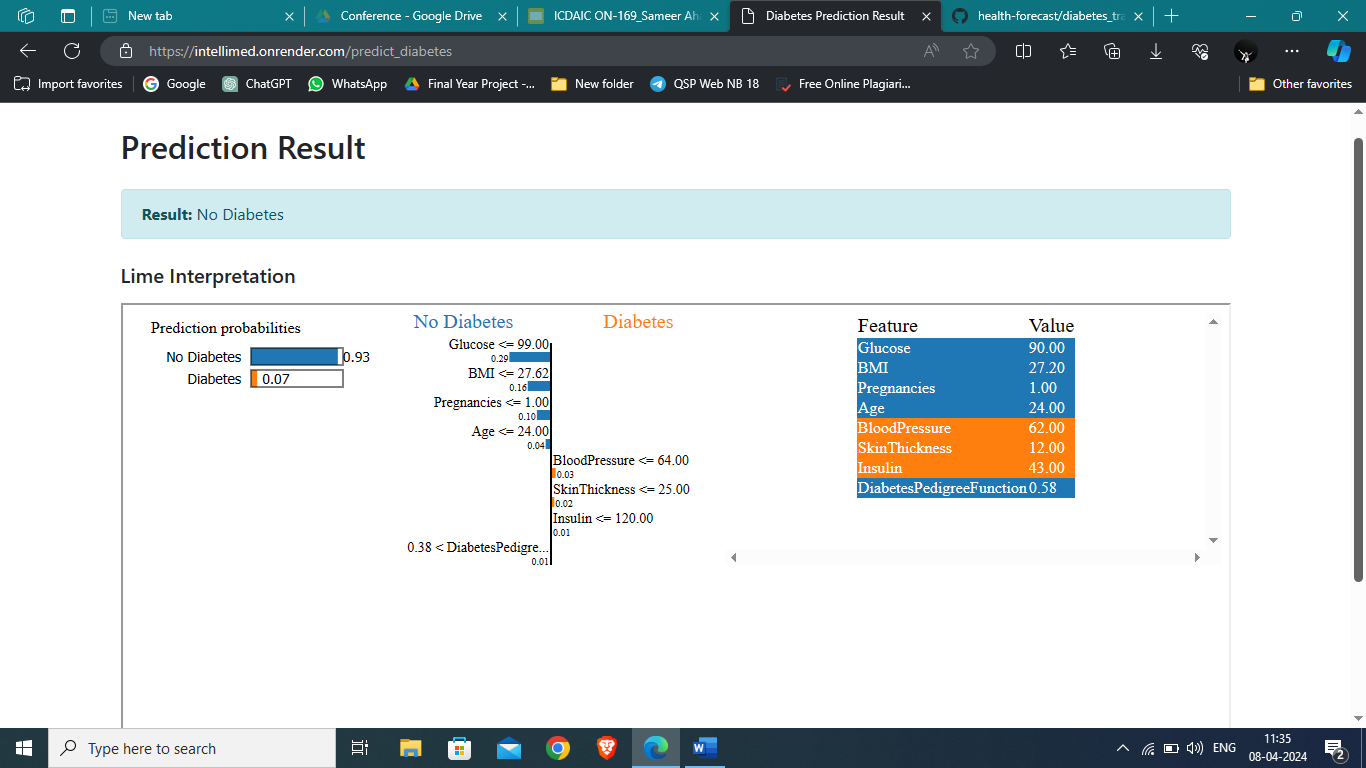
**Figure A1.1 Lung Disease Prediction Form**



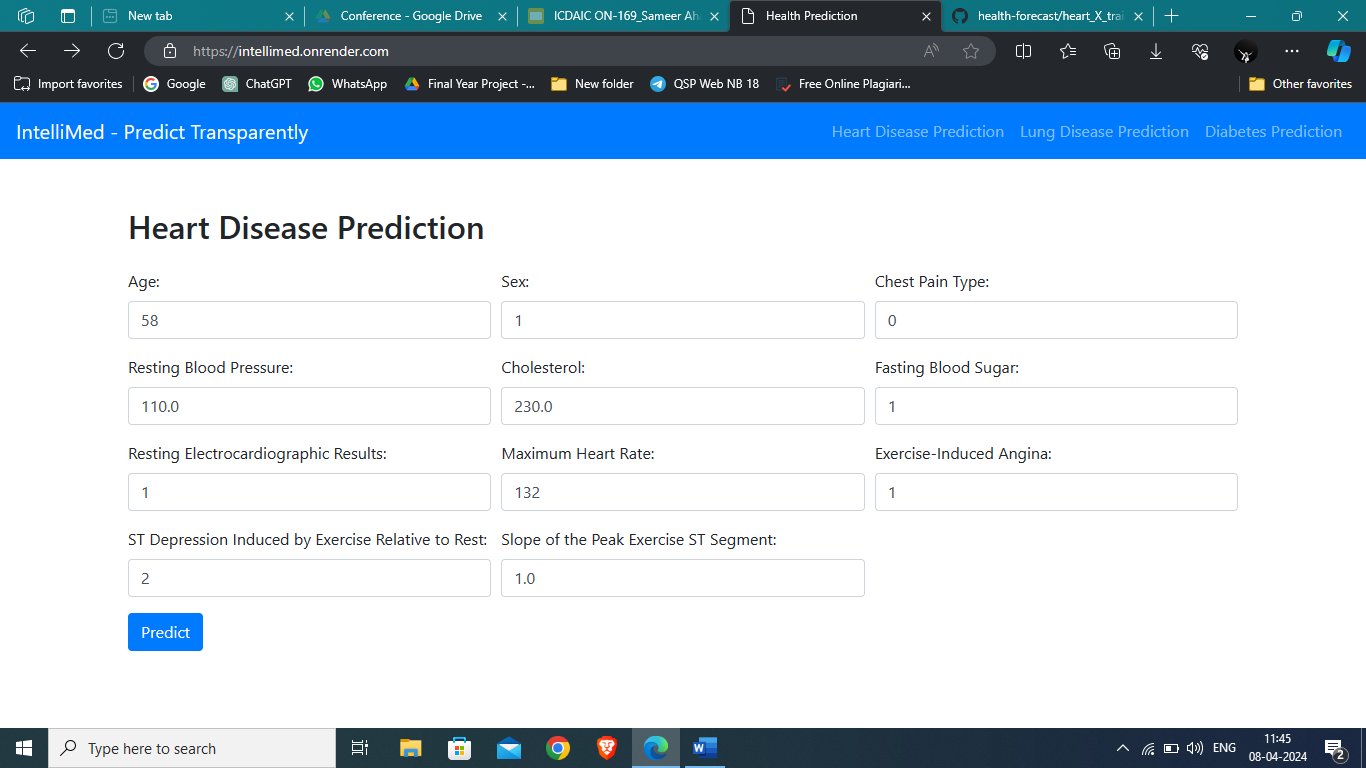
**Figure A1.2 Lung Disease Prediction Result**



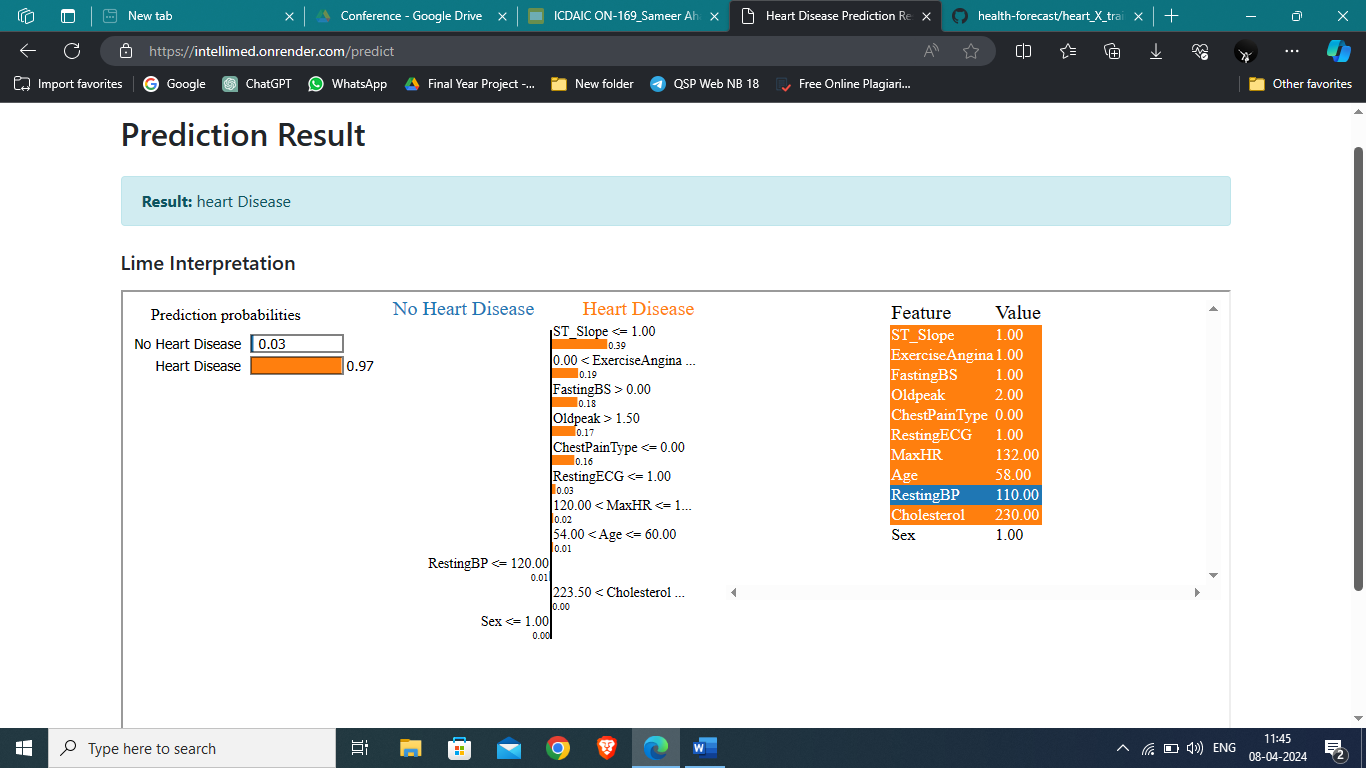
**Figure A1.3 Diabetes Prediction Form**



**Figure A1.4 Diabetes Prediction Result**



**Figure A1.5 Heart Disease Prediction Form**



**Figure A1.6 Heart Disease Prediction Result**

## APPENDIX 2

## SAMPLE CODING

# Import additional libraries

from flask import Flask, render\_template, request

import joblib

import numpy as np

from lime import lime\_tabular

import pandas as pd

app = Flask(\_\_name\_\_)

# Load the heart disease model

heart\_model\_filename = "logistic\_regression\_model.joblib"

diabetes\_model = joblib.load("diabetes.joblib")

diabetes\_X\_train = pd.read\_csv("diabetes\_train.csv")

diabetes\_feature\_names = [

'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age'

]

try:

heart\_model = joblib.load(heart\_model\_filename)

except Exception as e:

print(f"Error loading the heart disease model: {e}")

raise

heart\_feature\_names = [

'Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',

'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina',

'Oldpeak', 'ST\_Slope'

]

# Load training data for heart disease from CSV

try:

heart\_X\_train = pd.read\_csv("heart\_X\_train.csv")

except FileNotFoundError:

print("heart\_X\_train.csv not found. Please check the file path.")

raise

# Define Lime interpretation function for heart disease

def heart\_lime\_interpretation(features):

heart\_explainer = lime\_tabular.LimeTabularExplainer(

heart\_X\_train.values, # Use heart\_X\_train as a pandas DataFrame

feature\_names=heart\_feature\_names,

class\_names=['No Heart Disease', 'Heart Disease'],

discretize\_continuous=True

)

# Convert the input features to a NumPy array

heart\_features\_array = np.array(features, dtype=float)

heart\_explanation = heart\_explainer.explain\_instance(

heart\_features\_array,

heart\_model.predict\_proba,

num\_features=len(heart\_features\_array)

)

return heart\_explainer, heart\_explanation

# Function to convert Heart Lime explanation to HTML format

def heart\_lime\_to\_html(heart\_explanation):

return heart\_explanation.as\_html()

lung\_model = joblib.load("lungs\_model.joblib")

lung\_feature\_names = ['YELLOW\_FINGERS',

'ANXIETY',

'PEER\_PRESSURE',

'CHRONIC DISEASE',

'FATIGUE ',

'ALLERGY ',

'WHEEZING',

'ALCOHOL CONSUMING',

'COUGHING',

'SWALLOWING DIFFICULTY',

'CHEST PAIN',

'ANXYELFIN']

# Load training data for lung disease from CSV

try:

lung\_X\_train = pd.read\_csv("lungs\_train.csv")

except FileNotFoundError:

print("lung\_X\_train.csv not found. Please check the file path.")

raise

# Define Lime interpretation function for lung disease

def lung\_lime\_interpretation(features):

lung\_explainer = lime\_tabular.LimeTabularExplainer(

lung\_X\_train.values, # Use lung\_X\_train as a pandas DataFrame

feature\_names=lung\_feature\_names,

class\_names=['No Lung Disease', 'Lung Disease'],

discretize\_continuous=True

)

# Convert the input features to a NumPy array

lung\_features\_array = np.array(features, dtype=float)

lung\_explanation = lung\_explainer.explain\_instance(

lung\_features\_array,

lung\_model.predict\_proba,

num\_features=len(lung\_features\_array)

)

return lung\_explainer, lung\_explanation

# Function to convert Lung Lime explanation to HTML format

def lung\_lime\_to\_html(lung\_explanation):

return lung\_explanation.as\_html()

def diabetes\_lime\_interpretation(features):

diabetes\_explainer = lime\_tabular.LimeTabularExplainer(

diabetes\_X\_train.values, # Use diabetes\_X\_train as a pandas DataFrame

feature\_names=diabetes\_feature\_names,

class\_names=['No Diabetes', 'Diabetes'],

discretize\_continuous=True

)

# Convert the input features to a NumPy array

diabetes\_features\_array = np.array(features, dtype=float)

diabetes\_explanation = diabetes\_explainer.explain\_instance(

diabetes\_features\_array,

diabetes\_model.predict\_proba,

num\_features=len(diabetes\_features\_array)

)

return diabetes\_explainer, diabetes\_explanation

# Function to convert Diabetes Lime explanation to HTML format

def diabetes\_lime\_to\_html(diabetes\_explanation):

return diabetes\_explanation.as\_html()

# Render the main page with the input form

@app.route('/')

def index():

return render\_template('index.html')

# Predict heart disease based on input and display the result with Lime interpretation

@app.route('/predict', methods=['POST','GET'])

def predict():

if request.method == 'POST':

# Get input values from the form

features = [

float(request.form['age']),

float(request.form['sex']),

float(request.form['chest\_pain\_type']),

float(request.form['resting\_bp']),

float(request.form['cholesterol']),

float(request.form['fasting\_bs']),

float(request.form['resting\_ecg']),

float(request.form['max\_hr']),

float(request.form['exercise\_angina']),

float(request.form['oldpeak']),

float(request.form['st\_slope'])

]

if not hasattr(heart\_model, 'predict\_proba') or not callable(getattr(heart\_model, 'predict\_proba', None)):

raise AttributeError("The loaded heart disease model does not have a 'predict\_proba' method.")

# Make a heart disease prediction

heart\_raw\_prediction = heart\_model.predict\_proba([features])[0]

heart\_probability\_positive\_class = heart\_raw\_prediction[1] # Assuming 1 corresponds to the positive class

# Get Heart Lime interpretation

heart\_explainer, heart\_lime\_explanation = heart\_lime\_interpretation(features)

# Convert Heart Lime explanation to HTML format

heart\_lime\_html = heart\_lime\_to\_html(heart\_lime\_explanation)

# Save Heart Lime explanation to HTML file

heart\_html\_filename = 'static/heart\_lime\_visualization.html'

with open(heart\_html\_filename, 'w', encoding='utf-8') as file:

file.write(heart\_lime\_html)

# Display the result in a popup

heart\_result = "Heart Disease" if heart\_probability\_positive\_class >= 0.5 else "No Heart Disease"

return render\_template('result.html', result=heart\_result, lime\_visualization\_path=heart\_html\_filename)

return render\_template('index.html')

@app.route('/lung\_disease', methods=['GET', 'POST'])

def lung\_disease():

if request.method == 'POST':

# Get input values from the form for lung disease prediction

lung\_features = [

float(request.form['YELLOW\_FINGERS']),

float(request.form['ANXIETY']),

float(request.form['PEER\_PRESSURE']),

float(request.form['CHRONIC\_DISEASE']),

float(request.form['FATIGUE']),

float(request.form['ALLERGY']),

float(request.form['WHEEZING']),

float(request.form['ALCOHOL\_CONSUMING']),

float(request.form['COUGHING']),

float(request.form['SWALLOWING\_DIFFICULTY']),

float(request.form['CHEST\_PAIN']),

float(request.form['ANXYELFIN'])

]

if not hasattr(lung\_model, 'predict\_proba') or not callable(getattr(lung\_model, 'predict\_proba', None)):

raise AttributeError("The loaded lung disease model does not have a 'predict\_proba' method.")

# Make a lung disease prediction

lung\_raw\_prediction = lung\_model.predict\_proba([lung\_features])[0]

lung\_probability\_positive\_class = lung\_raw\_prediction[1] # Assuming 1 corresponds to the positive class

# Get Lung Lime interpretation

lung\_explainer, lung\_lime\_explanation = lung\_lime\_interpretation(lung\_features)

# Convert Lung Lime explanation to HTML format

lung\_lime\_html = lung\_lime\_to\_html(lung\_lime\_explanation)

# Save Lung Lime explanation to HTML file

lung\_html\_filename = 'static/lung\_lime\_visualization.html'

with open(lung\_html\_filename, 'w', encoding='utf-8') as file:

file.write(lung\_lime\_html)

# Display the result in a popup

lung\_result = "Lung Disease" if lung\_probability\_positive\_class >= 0.5 else "No Lung Disease"

return render\_template('lung\_result.html', lung\_result=lung\_result, lung\_lime\_visualization\_path=lung\_html\_filename)

return render\_template('lung\_index.html')

@app.route('/predict\_diabetes', methods=['POST','GET'])

def predict\_diabetes():

# Get input values from the form for diabetes prediction

if request.method == 'POST':

diabetes\_features = [

float(request.form['pregnancies']),

float(request.form['glucose']),

float(request.form['blood\_pressure']),

float(request.form['skin\_thickness']),

float(request.form['insulin']),

float(request.form['bmi']),

float(request.form['diabetes\_pedigree\_function']),

float(request.form['age'])

]

if not hasattr(diabetes\_model, 'predict\_proba') or not callable(getattr(diabetes\_model, 'predict\_proba', None)):

raise AttributeError("The loaded diabetes model does not have a 'predict\_proba' method.")

# Make a diabetes prediction

diabetes\_raw\_prediction = diabetes\_model.predict\_proba([diabetes\_features])[0]

diabetes\_probability\_positive\_class = diabetes\_raw\_prediction[1] # Assuming 1 corresponds to the positive class

# Get Diabetes Lime interpretation

diabetes\_explainer, diabetes\_lime\_explanation = diabetes\_lime\_interpretation(diabetes\_features)

# Convert Diabetes Lime explanation to HTML format

diabetes\_lime\_html = diabetes\_lime\_to\_html(diabetes\_lime\_explanation)

# Save Diabetes Lime explanation to HTML file

diabetes\_html\_filename = 'static/diabetes\_lime\_visualization.html'

with open(diabetes\_html\_filename, 'w', encoding='utf-8') as file:

file.write(diabetes\_lime\_html)

# Display the result in a popup

diabetes\_result = "Diabetes" if diabetes\_probability\_positive\_class >= 0.5 else "No Diabetes"

return render\_template('diabetes\_result.html', diabetes\_result=diabetes\_result, diabetes\_lime\_visualization\_path=diabetes\_html\_filename)

@app.route('/diabetes', methods=['POST','GET'])

def diabetes\_form():

return render\_template('diabetes\_index.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

## REFERENCES

* 1. Ahmad Chaddad, Jihao Peng, Jian Xu, Ahmed Bouridane (2023) “Survey of Explainable AI Techniques in Healthcare”, MDPI .
  2. Deepti Sarawat, Prona Bhattacharyaa, Ashwin Verma, Vivek Kumar Prasad, Sudeep Tanwar, Gulshan Sharma, Pitshoou N. Bokoro, Ravi Sharma (2022) “Explainable AI for Healthcare 5.0: Opportunities and Challenges”, IEEE .
  3. S. Khedkar, V. Subramanian, G. Shinde, and P. Gandhi (2019) “Explainable AI in Healthcare”, SSRN Electronic Journal .
  4. M. T. Ribeiro and C. Guestrin, “Anchors : High-Precision Model-Agnostic Explanations”.
  5. Urja Pawar, Donna O’ Shea, Susan Rea, Ruairi O’ Reilly (2020) “Explainable AI in Healthcare”, International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA) .
  6. “Smart healthcare: making medical care more intelligent” (2019) GlobalHealth Journal, vol. 3, no. 3, pp. 62–65 .