Summer Term-2024

PROJECT REPORT

Submitted by

PRATHYUSHA MUDUNURI

USN: 22BCAR0364

KRISHNA JASANI

USN: 22BCAR0487

5th Semester- General

in partial fulfillment for the award of the

degree

of

BACHELOR OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER SCIENCE & IT

JAIN KNOWLEDGE CAMPUS JAYANAGAR 9TH BLOCK BANGALORE - 560069

JUNE - 2024

JGi JAIN DEEMED-TO-BE UNIVERSITY School Of Computer Science and IT

DEPARTMENT OF COMPUTER SCIENCE & IT

Jain Knowledge Campus
Jayanagar 9th Block Bangalore, 560069

This is to certify that the project entitled

"HEART ATTACK PREDICTION SYSTEM"

is the bonafide record of project work done by

Prathyusha Mudunuri	**USN: 22BCAR0364**
Krishna Jasani	**USN: 22BCAR0487**

BCA during the summer term
June-2024

Prof Jayashree M Kudari
Faculty Mentor
JAIN (Deemed-to-be University)

Dr. Suneetha K
Head, School of CS & IT
JAIN (Deemed-to-be University)

JAIN DEEMED-TO-BE UNIVERSITY | School Of Computer Science and IT

DECLARATION

We affirm that the project work titled **"Heart Attack Prediction System"**, being submitted in partial fulfillment for the award of BACHELOR OF COMPUTER APPLICATIONS is the original work carried out by us. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University/Institution.

PRATHYUSHA MUDUNURI

22BCAR0364

KRISHNA JASANI

22BCAR0487

School Of Computer Science and IT

CERTIFICATE

This is to certify that **PRATHYUSHA MUDUNURI** with USN: 22BCAR0364 and **KRISHNA JASANI** with USN: 22BCAR0487 of Bachelor of Computer Applications in the School of Computer Science and IT, has worked on the project entitled, **“Heart Attack Prediction System”** as part of summer term May-June, 2024 under my direct supervision. No part of this project was submitted for the award of any degree or diploma prior to any other university/institution till this date and is the authentic work of student.

Prof Jayashree M Kudari
Guide / Mentor
JAIN (Deemed-to-be University)

ACKNOWLEDGEMENT

We would like to acknowledge the following people, who have encouraged, guided and helped to accomplish our report to award my degree at the JAIN (Deemed to be University), Department of Computer Science and IT, School of Computer Science and IT:

1. Project mentor Prof Jayashree M Kudari for guiding us through pivotal moments for our study and professional career and for always being there to make sure that our progress was reviewed, documented and acknowledged. Her encouragement has been the greatest source of inspiration and confidence for carrying out my project work.

2. Faculty and staff members of **Department of Computer Science & IT** for sharing their expertise and for always showing their interests in our work.

3. We also would like to extend my thanks to our friends, for their efforts to make our report a more effective.

4. Finally, we would like to thank our families, to whom this work is dedicated, for their support and encouragement during these years.

PRATHYUSHA MUDUNURI

KRISHNA JASANI

ABSTRACT

The Heart Attack Prediction System project is an interdisciplinary initiative aimed at developing a reliable, user-friendly tool for predicting the likelihood of heart disease in patients. By harnessing the power of machine learning algorithms, specifically logistic regression, and integrating it with an intuitive graphical user interface (GUI), this project seeks to enhance preventive healthcare through accurate predictions and effective data visualization.

PROJECT SCOPE AND OBJECTIVES:

The primary goal of this project was to create a system capable of predicting heart disease risk based on various patient health metrics. The project encompassed several key objectives:

1. Developing an Accurate Prediction Model: Utilize logistic regression to analyze patient data and predict heart disease probability.
2. Creating a User-Friendly Interface: Design a GUI using Tkinter that allows users to input data and view predictions easily.
3. Implementing Data Visualization: Generate graphs and charts to help users understand their health data and prediction results.
4. Ensuring Secure Access: Develop a login system to restrict application access to authorized users only.

METHODOLOGY:

- Data Collection and Preprocessing: The project began with the collection of a comprehensive dataset, including demographic information (age, gender), clinical measurements (blood pressure, cholesterol levels), and lifestyle factors (smoking, physical activity). Data preprocessing involved handling missing values, standardizing data, and preparing it for model training.

- Model Training and Validation: Logistic regression was selected for its effectiveness in binary classification tasks. The model was trained using standardized patient data with an increased number of iterations and the 'sag' solver to enhance convergence and accuracy. Cross-validation techniques were employed to assess the model's performance and avoid overfitting.

- GUI Development: Tkinter, a Python library, was used to create the graphical interface. The GUI includes forms for data input, a login screen for user authentication, and dashboards for displaying prediction results and visualizations.

- Data Visualization: Matplotlib was used to create various graphs and charts, illustrating patient health metrics and prediction probabilities. These visualizations provide insights into the relationship between different health parameters and heart disease risk.

RESULTS:

The Heart Attack Prediction System successfully predicts heart disease risk with high accuracy, providing binary outputs that indicate the presence or absence of heart disease based on input parameters. The GUI simplifies data entry and result interpretation, making the system accessible to non-experts. Visualization tools enhance understanding by depicting key health indicators and their relationships visually.

CONCLUSION:

This project demonstrates the effective combination of machine learning and user-centric design in developing a practical healthcare tool. The Heart Attack Prediction System offers a promising solution for early detection and prevention of heart disease, empowering healthcare providers and patients with actionable insights.

FUTURE WORK:

To further improve the system, future efforts could focus on incorporating more advanced machine learning models, expanding the dataset for better generalizability, and adding features such as personalized health recommendations and real-time data analysis capabilities. Enhancing the authentication system and incorporating cloud-based services for data storage and processing could also be valuable enhancements.

Overall, this project highlights the potential of integrating data science and application development to create impactful solutions in the healthcare domain, potentially saving lives through early intervention and informed decision-making.

LIST OF TABLES

Table 1: Patient Data Fields and Descriptions

Field Name	Data Type	Description
Patient_ID	Integer	Unique identifier for each patient
Age	Integer	Age of the patient
Sex	Integer	Gender of the patient (1 = Male, 0 = Female)
Chest_Pain_Type	Integer	Type of chest pain (1-4 indicating severity)
Resting_BP	Integer	Resting blood pressure (in mm Hg)
Cholesterol	Integer	Serum cholesterol in mg/dl
Fasting_BS	Integer	Fasting blood sugar > 120 mg/dl (1 = True, 0 = False)
Rest_ECG	Integer	Resting electrocardiographic results (0-2)
Max_Heart_Rate	Integer	Maximum heart rate achieved
Exercise_Angina	Integer	Exercise-induced angina (1 = Yes, 0 = No)
Oldpeak	Float	ST depression induced by exercise relative to rest
Slope	Integer	Slope of the peak exercise ST segment (1-3)
CA	Integer	Number of major vessels (0-3) colored by fluoroscopy
Thal	Integer	Thalassemia (1 = Normal; 2 = Fixed Defect; 3 = Reversible Defect)
Target	Integer	Heart disease diagnosis (0 = No, 1 = Yes)

Table 2: Database Schema for Heart Attack Prediction System

Table Name	Field Name	Data Type	Description
patients	Patient_ID	Integer	Unique identifier for each patient
patients	Age	Integer	Age of the patient
patients	Sex	Integer	Gender of the patient (1 = Male, 0 = Female)
patients	Chest_Pain_Type	Integer	Type of chest pain (1-4 indicating severity)
patients	Resting_BP	Integer	Resting blood pressure (in mm Hg)
patients	Cholesterol	Integer	Serum cholesterol in mg/dl
patients	Fasting_BS	Integer	Fasting blood sugar > 120 mg/dl (1 = True, 0 = False)
patients	Rest_ECG	Integer	Resting electrocardiographic results (0-2)
patients	Max_Heart_Rate	Integer	Maximum heart rate achieved
patients	Exercise_Angina	Integer	Exercise-induced angina (1 = Yes, 0 = No)
patients	Oldpeak	Float	ST depression induced by exercise relative to rest
patients	Slope	Integer	Slope of the peak exercise ST segment (1-3)
patients	CA	Integer	Number of major vessels (0-3) colored by fluoroscopy
patients	Thal	Integer	Thalassemia (1 = Normal; 2 = Fixed Defect; 3 = Reversible Defect)
patients	Target	Integer	Heart disease diagnosis (0 = No, 1 = Yes)

LIST OF FIGURES

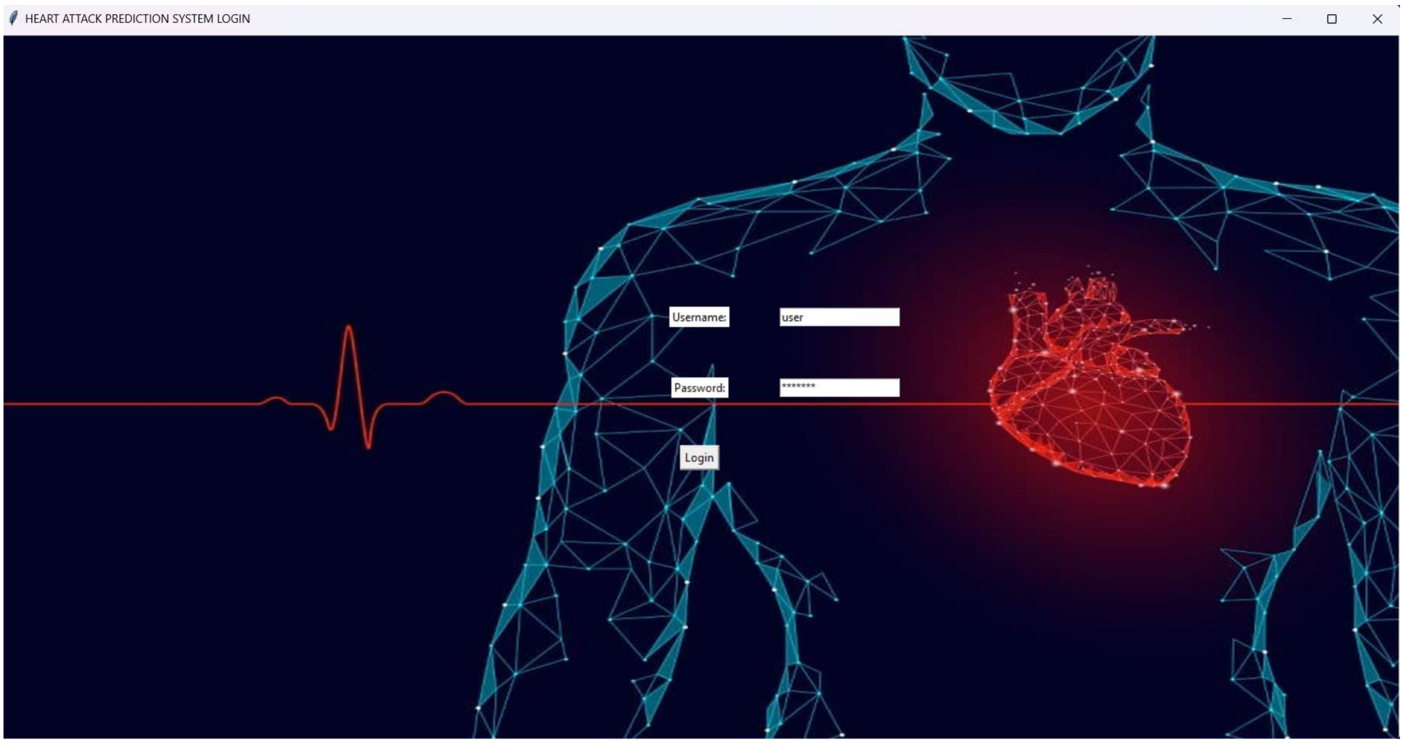


Figure 1 : Login Page

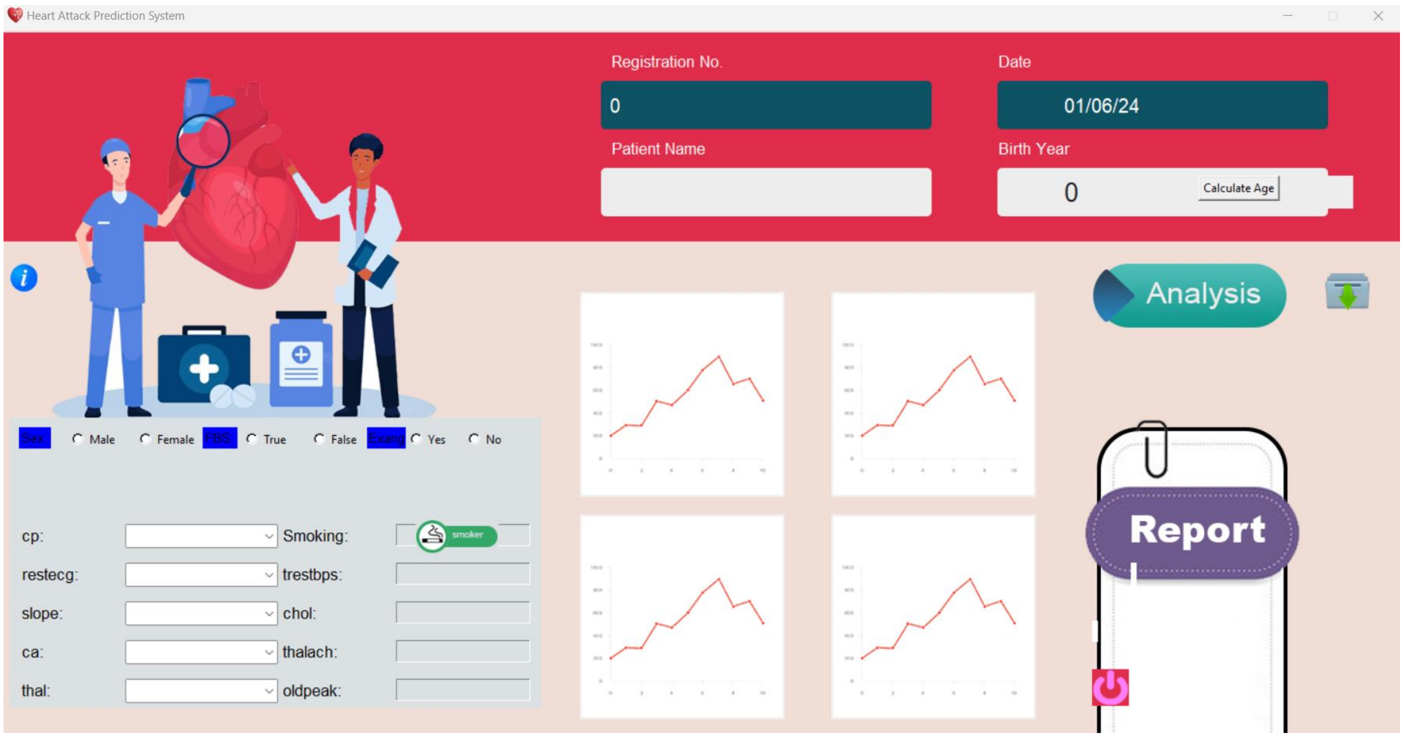
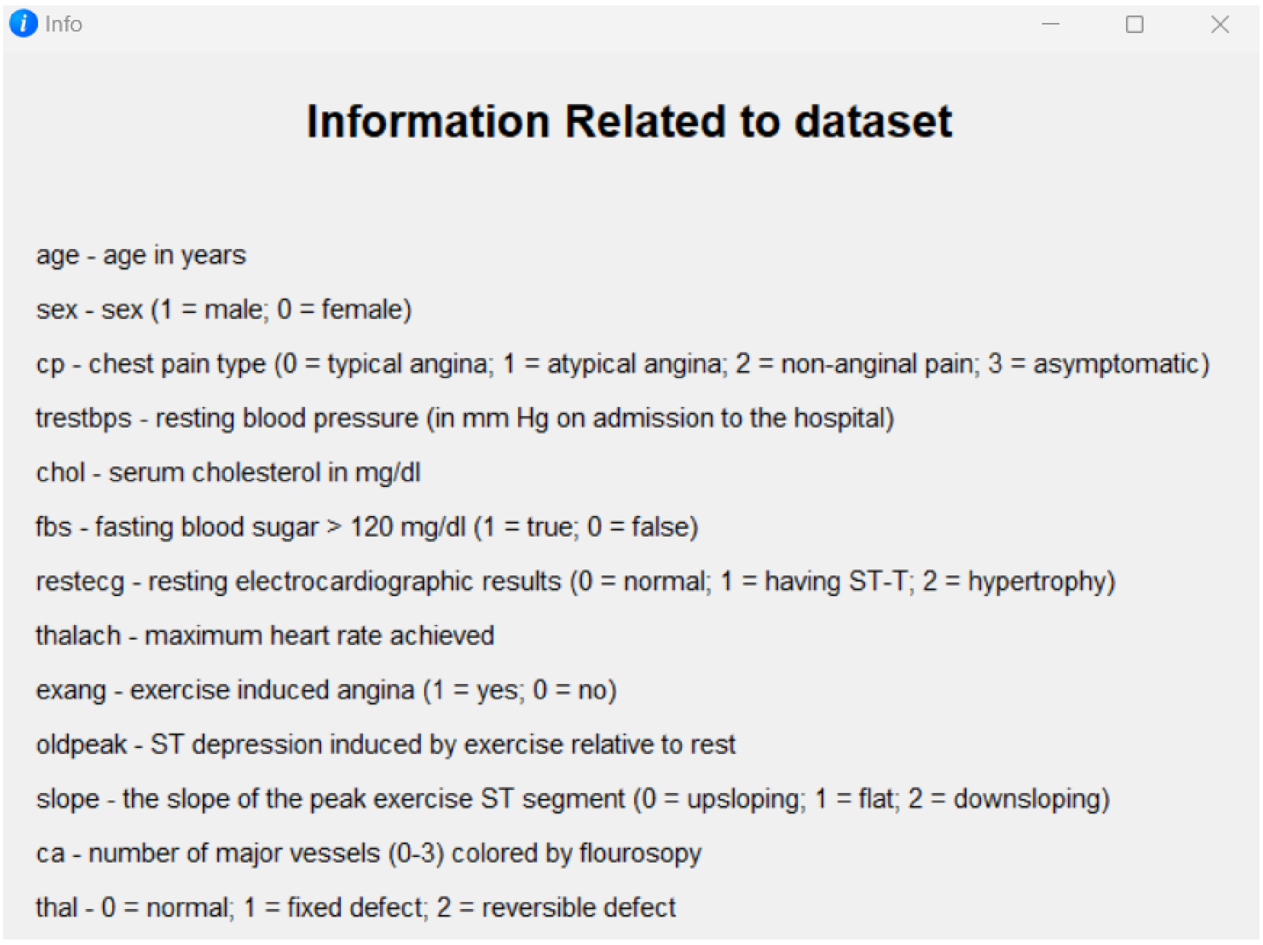


Figure 2: Main Window- Data Entry hub


Info

Information Related to dataset

age - age in years

sex - sex (1 = male; 0 = female)

cp - chest pain type (0 = typical angina; 1 = atypical angina; 2 = non-anginal pain; 3 = asymptomatic)

trestbps - resting blood pressure (in mm Hg on admission to the hospital)

chol - serum cholesterol in mg/dl

fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)

restecg - resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)

thalach - maximum heart rate achieved

exang - exercise induced angina (1 = yes; 0 = no)

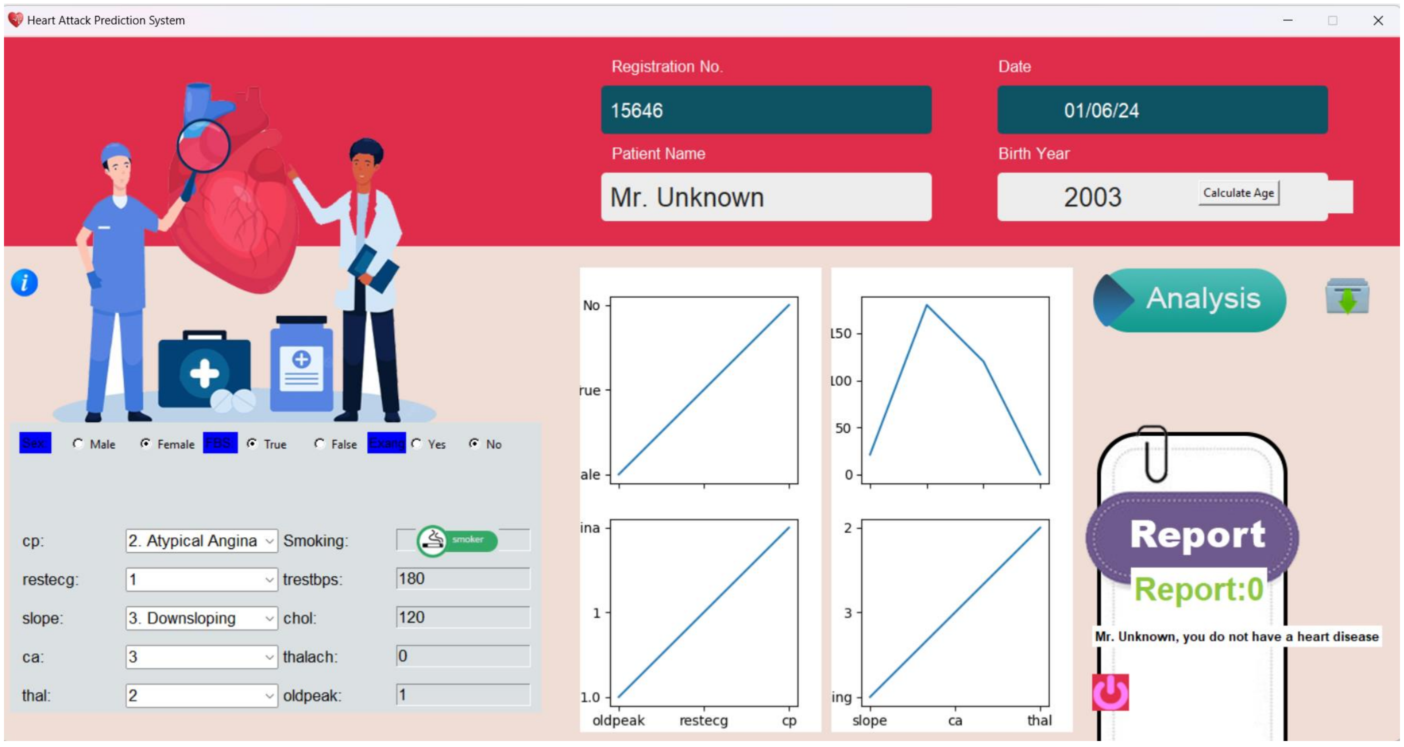
oldpeak - ST depression induced by exercise relative to rest

slope - the slope of the peak exercise ST segment (0 = upsloping; 1 = flat; 2 = downsloping)

ca - number of major vessels (0-3) colored by flourosopy

thal - 0 = normal; 1 = fixed defect; 2 = reversible defect

Figure 3: Information related to dataset


Heart Attack Prediction System

Registration No. 15646

Date 01/06/24

Patient Name Mr. Unknown

Birth Year 2003

Sex: Male [✓ Female] FBS: [✓ True] False Exang: Yes [✓ No]

cp:	2. Atypical Angina	Smoking:	smoker
restecg:	1	trestbps:	180
slope:	3. Downsloping	chol:	120
ca:	3	thalach:	0
thal:	2	oldpeak:	1

Report:0

Mr. Unknown, you do not have a heart disease

Figure 4: Insightful Reports Generated from Entered Data

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO THE PROJECT

1.1 Overview

Heart disease remains one of the leading causes of mortality worldwide, posing a significant public health challenge. Early detection and timely intervention are crucial in reducing the morbidity and mortality associated with heart attacks. The Heart Attack Prediction System project addresses this critical need by developing a software solution capable of predicting the risk of a heart attack based on various health parameters. This system leverages machine learning techniques to provide accurate and timely predictions, thereby assisting healthcare professionals in making informed decisions and encouraging patients to take proactive measures in managing their heart health.

1.2 The Need for Heart Attack Prediction

The modern lifestyle, characterized by sedentary behaviour, poor dietary habits, and high-stress levels, has significantly contributed to the increasing prevalence of heart disease. Traditional methods of diagnosing and predicting heart attacks often rely heavily on clinical expertise and may not always be accessible or timely, particularly in underserved or remote areas where medical facilities might be lacking. Furthermore, the inherent complexity of cardiovascular conditions, coupled with the multitude of risk factors involved, necessitates a more comprehensive and data-driven approach to accurately assess and predict heart attack risk. There is a pressing need for an advanced system that can meticulously analyse various medical parameters, including but not limited to genetic factors, lifestyle choices, and existing health conditions, to provide a reliable prediction of heart attack risk. This would enable early intervention, allow for more personalized treatment plans, and potentially save numerous lives by preventing heart attacks before they occur.

1.3 Solution Provided

The Heart Attack Prediction System offers a robust and comprehensive solution to the pressing problem of heart attack risk assessment by integrating advanced machine learning algorithms with an intuitive, user-friendly interface. This system is designed to accommodate a wide range of users, including healthcare professionals and patients, who can easily input key health parameters such as age, gender, blood pressure, cholesterol levels, smoking status, family medical history, and other relevant metrics. By processing this extensive set of data through a Logistic Regression model, the system is able to predict the likelihood of a heart attack with a high degree of accuracy.

Additionally, the system provides invaluable insights through sophisticated data visualization tools, making it easier for users to understand the underlying factors contributing to the predicted risk. This includes charts, graphs, and other visual aids that highlight trends and correlations, helping healthcare professionals identify critical risk factors and make informed decisions. By offering timely and accurate heart attack risk assessments, the system aims to improve patient outcomes through early intervention and personalized treatment plans. This proactive approach can reduce heart attack incidence and associated healthcare costs. Ultimately, the Heart Attack Prediction System seeks to enhance heart health management, promote healthier lifestyles, and save lives.

CHAPTER 2

PROBLEM DEFINITION

2.1 Literature Survey

Heart disease is a predominant cause of death globally, with millions of fatalities attributed to heart attacks each year. Numerous studies have been conducted to identify the risk factors and improve the early detection of heart disease. Research highlights the significance of factors such as age, gender, blood pressure, cholesterol levels, smoking habits, and family history in the prediction of heart attacks.

Recent advancements in machine learning and data analytics have shown promise in the field of medical diagnostics. Algorithms such as Logistic Regression, Decision Trees, Support Vector Machines, and Neural Networks have been successfully applied to predict various health conditions, including heart disease. Literature indicates that machine learning models, when trained on extensive patient datasets, can achieve high accuracy in predicting heart attack risks, thereby aiding in early diagnosis and preventive care.

Despite these advancements, there remains a gap in integrating these predictive models into user-friendly software solutions accessible to both healthcare providers and patients. This project aims to bridge this gap by developing a comprehensive Heart Attack Prediction System.

2.2 Problem Identification

While traditional diagnostic methods for heart disease are effective, they often require extensive medical examinations and specialized knowledge, which may not always be accessible to everyone, especially in underserved areas. The increasing prevalence of heart disease necessitates a scalable, efficient, and accurate prediction system that can assist in early diagnosis and intervention.

Current systems often lack the ability to provide real-time, user-friendly, and accessible predictions. Furthermore, many existing solutions do not effectively incorporate the diverse range of risk factors into a cohesive predictive model. There is a need for a system that not only predicts the risk of a heart attack with high accuracy but also presents the information in an understandable and actionable format.

2.3 Detailed Problem Statement

Business Problem: The primary business problem addressed by the Heart Attack Prediction System is the need for an accessible, accurate, and user-friendly tool that can predict the risk of heart attacks based on multiple health parameters. This system aims to aid healthcare providers in early diagnosis and intervention, thereby reducing the incidence of heart attacks and improving patient outcomes. It also aims to empower patients by providing them with insights into their heart health and encouraging proactive health management.

Software Requirements:

1. **Functional Requirements:**

- **Data Input Interface:** A user-friendly interface for inputting health parameters such as age, gender, blood pressure, cholesterol levels, smoking status, and family history.
- **Prediction Algorithm:** Integration of a Logistic Regression model to analyze input data and predict the likelihood of a heart attack.
- **Output and Visualization:** Display of prediction results and risk factors through intuitive visualizations and graphs.
- **User Management:** Functionality for healthcare providers and patients to create and manage user profiles.
- **Report Generation:** Ability to generate and export reports summarizing the prediction results and recommendations.

2. **Non-Functional Requirements:**

- **Performance:** The system should provide real-time predictions with minimal latency.
- **Scalability:** The system should be capable of handling a large number of user requests and data inputs.
- **Usability:** The interface should be intuitive and easy to navigate for users with varying levels of technical expertise.
- **Security:** The system must ensure the confidentiality and integrity of user data, adhering to relevant data protection regulations.
- **Reliability:** The system should maintain high availability and provide consistent prediction accuracy.

CHAPTER 3

PROJECT DESIGN

System Architecture:

The system architecture will consist of several interconnected and meticulously designed components to ensure the effectiveness and reliability of heart attack prediction:

- **Data Collection Module:** This component is responsible for gathering relevant patient data from various sources, including electronic health records (EHRs), patient demographics, comprehensive medical history, lifestyle information, and diagnostic test results such as blood tests, ECG readings, and imaging studies. This module ensures that a holistic set of data is collected, covering all necessary aspects to provide a robust foundation for analysis.

- **Preprocessing Module:** This module focuses on cleansing and preparing the collected data for subsequent analysis. Tasks include handling missing values through imputation techniques, normalization of data to ensure consistency, and encoding categorical variables. This step is crucial for transforming raw data into a format that is suitable for machine learning algorithms, thereby enhancing the quality and reliability of the predictions.

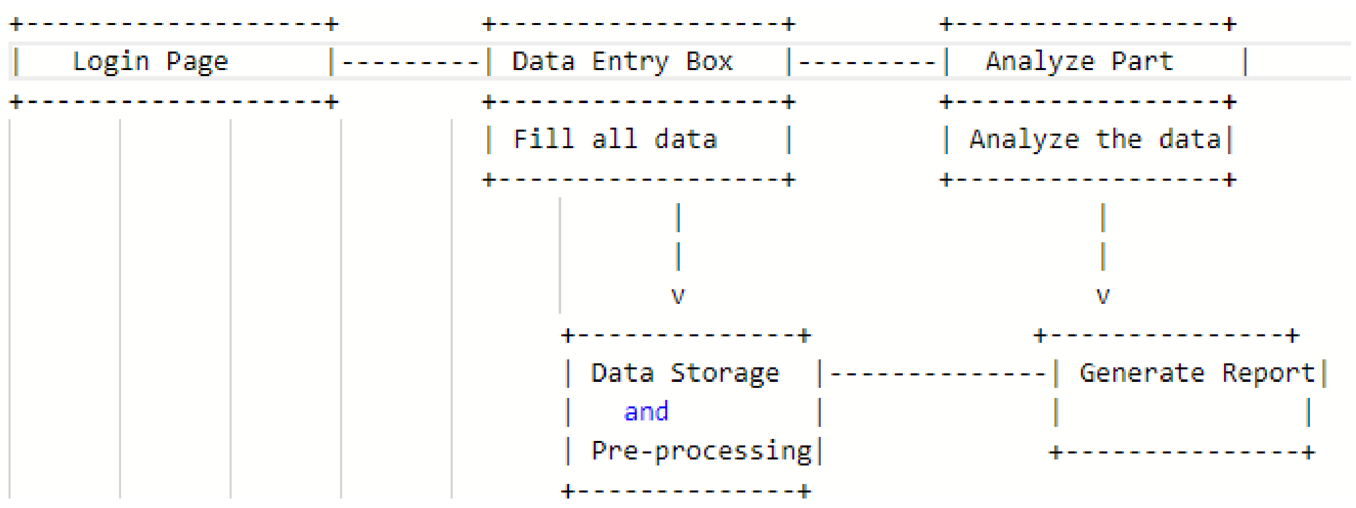
- **Feature Selection Module:** In this component, the system identifies the most relevant features for heart attack prediction through techniques such as correlation analysis, mutual information, and feature importance from models. By selecting key features and reducing dimensionality, this module aims to improve model performance, reduce overfitting, and enhance the interpretability of the predictive models.

- **Machine Learning Model:** This component involves utilizing various supervised learning algorithms to build predictive models. Techniques such as logistic regression, decision trees, random forests, support vector machines, or neural networks are employed to develop models that can accurately predict the likelihood of a heart attack. The selection of algorithms is based on their performance, interpretability, and suitability for the specific characteristics of the dataset.

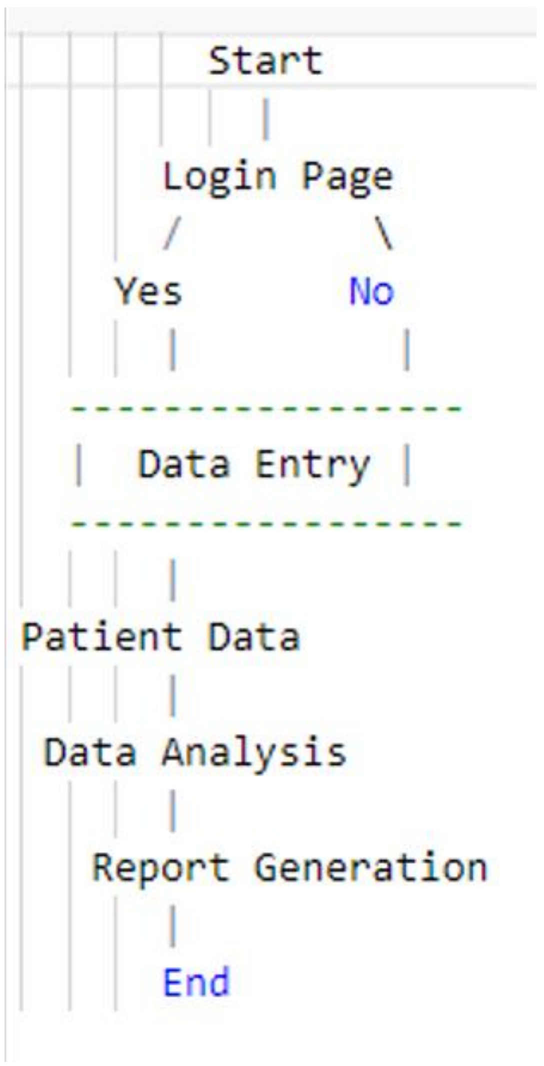
- **Prediction Engine:** The trained machine learning model is deployed within this component to predict the likelihood of a heart attack based on input data from new or existing patients. The prediction engine ensures that the model can process real-time data and provide timely predictions, which can be crucial for early intervention and treatment planning.

- **User Interface:** This component provides a user-friendly interface designed for healthcare professionals to input patient data and view prediction results. The interface is designed to be intuitive and accessible, featuring functionalities such as data entry forms, result visualization, and interactive dashboards. This enables clinicians to easily interpret the prediction outcomes and make informed decisions regarding patient care.

➢ Architectural Design of The Software:



```
+-------------------+         +------------------+        +-----------------+
|    Login Page     |---------| Data Entry Box   |---------|  Analyze Part    |
+-------------------+         +------------------+        +-----------------+
                              | Fill all data    |        | Analyze the data|
                              +------------------+        +-----------------+
                                        |                          |
                                        |                          |
                                        v                          v
                                   +-------------+             +---------------+
                                   | Data Storage |-------------| Generate Report|
                                   |   and        |             |               |
                                   | Pre-processing|            +---------------+
                                   +-------------+
```



```
        Start
          |
     Login Page
      /         \
    Yes          No
      |           |
  -----------------
  |  Data Entry  |
  -----------------
      |
Patient Data
      |
 Data Analysis
      |
   Report Generation
      |
     End
```

CHAPTER 4

HARDWARE SOFTWARE CONFIGURATION

▪ Technology Stack:

The proposed solution leverages a modern and comprehensive technology stack that is specifically tailored to meet the unique requirements of heart attack prediction. The choice of technologies encompasses both hardware and software components, ensuring optimal performance, scalability, and seamless compatibility with existing IT infrastructure in various healthcare settings. This approach ensures that the solution can be integrated smoothly into current systems while also being adaptable for future technological advancements.

▪ Hardware Configuration:

The hardware infrastructure comprises standard computing resources that are readily available in most healthcare facilities. This includes powerful servers or cloud-based platforms equipped with sufficient computational power and memory to handle data processing, model training, and prediction tasks efficiently. To further enhance the system’s capabilities, specialized hardware accelerators such as Graphics Processing Units (GPUs) may be employed to expedite computationally intensive operations, thereby significantly enhancing the speed and performance of the predictive model. This combination of standard and specialized hardware ensures that the solution can process large volumes of data quickly and accurately.

▪ Software Configuration:

The software ecosystem encompasses a diverse and carefully selected set of tools and frameworks specifically designed to facilitate various stages of the solution development lifecycle. This includes everything from data processing and analysis to model training and deployment:

1. **Programming Languages:** Python serves as the primary programming language due to its versatility, extensive library support, and widespread popularity in the data science and machine learning communities. Additionally, SQL is utilized for efficient database operations and data manipulation tasks, ensuring seamless interaction with the underlying data infrastructure.

2. **Data Processing and Analysis**: Pandas and NumPy are employed for sophisticated data manipulation and numerical operations, providing a robust foundation for data preprocessing. Meanwhile, scikit-learn offers a rich collection of machine learning algorithms and utilities for model training, evaluation, and validation, ensuring that the predictive models are both accurate and reliable.

3. **Deep Learning Frameworks:** TensorFlow or PyTorch may be employed for implementing deep

learning models, enabling the utilization of advanced neural networks for feature learning and predictive modeling. These frameworks provide the tools necessary for building complex models that can capture intricate patterns in the data.

4. **Visualization Libraries:** Matplotlib and Seaborn are utilized for comprehensive data visualization, facilitating the exploration of data patterns and model performance through intuitive visualizations such as histograms, scatter plots, and Receiver Operating Characteristic (ROC) curves. These visual tools are essential for understanding and interpreting the results of the predictive models.

5. **Development Environment:** Integrated Development Environments (IDEs) such as Jupyter Notebook or PyCharm are preferred for their interactive computing capabilities. These environments enable rapid prototyping, experimentation, and collaborative development, making it easier for data scientists and developers to work together efficiently.

6. **Database Management System (DBMS**): Depending on the scale and specific requirements of the project, relational databases like MySQL may be employed for data storage and retrieval. This ensures data integrity, scalability, and efficient management of the large volumes of data necessary for accurate heart attack prediction.

7. **Deployment and Hosting:** Containerization technologies such as Docker are utilized to facilitate the deployment of the solution in a reproducible and scalable manner. This ensures that the solution can be easily moved across different environments without compatibility issues. Additionally, cloud platforms like Amazon Web Services (AWS) or Microsoft Azure offer robust infrastructure for hosting, scaling, and managing the application, providing the necessary resources for high availability and performance.

By leveraging this comprehensive technology stack, the proposed solution aims to deliver a reliable and efficient system for heart attack prediction, capable of integrating seamlessly into existing healthcare IT environments and adapting to future technological advancements.

CHAPTER 5

CODING

Main.py

```python
import sys
from re import I
from tkinter import *
from tkinter import messagebox
import tkinter
from tkinter import filedialog
from tkinter import PhotoImage
from tkinter.ttk import Combobox
import datetime
from datetime import date
import tkinter as tk
from tkinter import ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
import numpy as np
import matplotlib
import selection
matplotlib.use("TKAgg")
import os
from sklearn.linear_model import LogisticRegression
from backend import *
from mysql import *
from sklearn.preprocessing import StandardScaler


import backend
from PIL import Image, ImageTk

import my_module
result = my_module

# Create a connection to the MySQL database
import mysql.connector
from mysql.connector import Error

def create_connection():
    """Create a database connection to the MySQL database"""
    connection = None
    try:
        connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='varma123',
            database='heart_data'
        )
        if connection.is_connected():
            print("Connection to MySQL DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")
    return connection

# Create a connection to the MySQL database
connection = create_connection()

# Flag to check if login was successful
login_successful = False

def login():
    username = username_entry.get()
    password = password_entry.get()

    # Validate username and password
    if username == "user" and password == "pass123":
        # If credentials are correct, close login window and open main application window
        login_window.destroy()
        open_main_window()
    else:
        messagebox.showerror("Login Failed", "Invalid username or password")

def open_main_window():

    pass

# Create login window
login_window = Tk()
login_window.title("HEART ATTACK PREDICTION SYSTEM LOGIN")
login_window.geometry("1450x730+60+80")


# Load the background image
background_image = Image.open("Images/inside.png")
background_image = background_image.resize((1450, 730), Image.LANCZOS)
background_photo = ImageTk.PhotoImage(background_image)

# Create a canvas
canvas = tk.Canvas(login_window, width=1450, height=730)
canvas.pack(fill="both", expand=True)

# Set the background image on the canvas
canvas.create_image(0, 0, image=background_photo, anchor="nw")

# Username Label and Entry
Label(login_window, text="Username:", bg="white").place(relx=0.5, rely=0.4, anchor=CENTER)
username_entry = Entry(login_window)
```

```
username_entry.place(relx=0.6, rely=0.4, anchor=CENTER)

# Password Label and Entry
Label(login_window, text="Password:", bg="white").place(relx=0.5, rely=0.5, anchor=CENTER)
password_entry = Entry(login_window, show="*")
password_entry.place(relx=0.6, rely=0.5, anchor=CENTER)
# Login Button
login_button = Button(login_window, text="Login", command=login)
login_button.place(relx=0.5, rely=0.6, anchor=CENTER)

login_window.mainloop()

# Increase max_iter
model = LogisticRegression(max_iter=1000)
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train the model using Logistic Regression with increased max_iter and a different solver
model = LogisticRegression(max_iter=1000, solver='sag')
model.fit(X_scaled, Y)

background = "#f0ddd5"
frame = "#fefbfb"

root = Tk()
root.title("Heart Attack Prediction System")
root.geometry("1450x730+60+80")
root.resizable(False, False)
root.config(bg=background)

#############Analysis####################
def analysis():
    global prediction
    try:
        print("Starting analysis...")

        # Retrieve user inputs
        name = Name.get()
        D1 = Date.get()
        today = date.today()
        birth_year = DOB.get()

        if birth_year == 0:
            messagebox.showerror("Invalid input", "Please enter a valid birth year!")
            return

        A = today.year - birth_year

        # Retrieve radio button selections
        B = sex_var.get()
        if B == '':
            messagebox.showerror("missing", "Please select gender!")
            return

        F = fbs_var.get()
        if F == '':
            messagebox.showerror("missing", "Please select fbs!")
            return

        I = exang_var.get()
        if I == '':
            messagebox.showerror("missing", "Please select exang!")
            return

        # Retrieve combobox selections
        C = cp_combobox.get()
        if C == '':
            messagebox.showerror("missing", "Please select cp!")
            return

        G = restecg_combobox.get()
        if G == '':
            messagebox.showerror("missing", "Please select restecg!")
            return

        K = slope_combobox.get()
        if K == '':
            messagebox.showerror("missing", "Please select slope!")
            return

        L = ca_combobox.get()
        if L == '':
            messagebox.showerror("missing", "Please select ca!")
            return

        M = thal_combobox.get()
        if M == '':
            messagebox.showerror("missing", "Please select thal!")
            return

        # Retrieve numeric entries
        try:
            D = int(trestbps_entry.get())
            E = int(chol_entry.get())
            H = int(thalach_entry.get())
            J = float(oldpeak_entry.get())
        except ValueError:
            messagebox.showerror("missing data", "Few missing data entry!")
            return

        print("Retrieved values:")
        print("A is age:", A)
        print("B is gender:", B)
        print("C is cp:", C)
        print("D is trestbps:", D)
        print("E is chol:", E)
        print("F is fbs:", F)
        print("G is restecg:", G)
        print("H is thalach:", H)
        print("I is exang:", I)
        print("J is oldpeak:", J)
        print("K is slope:", K)
        print("L is ca:", L)
        print("M is thal:", M)

        # Call selection2() and selection3() functions
        P2 = selection2()
        Q2 = selection3()

        print("P2 (ca):", P2)
        print("Q2 (thal):", Q2)

    except Exception as e:
        # Catch any exceptions and print error message
        print("Error occurred during analysis:", e)


        ############first graph###############
        f = Figure(figsize=(5, 5), dpi=100)
        a = f.add_subplot(111)
        a.plot(["sex", "fbs", "exang"], [B, F, I])
        canvas = FigureCanvasTkAgg(f, master=root)
        canvas.get_tk_widget().pack(side=tkinter.BOTTOM, fill=tk.BOTH, expand=True)
        canvas._tkcanvas.place(width=250, height=250, x=600, y=240)

        ############second graph###############
        f2 = Figure(figsize=(5, 5), dpi=100)
        a2 = f2.add_subplot(111)
        a2.plot(["age", "trestbps", "chol", "thalach"], [A, D, E, H])
        canvas2 = FigureCanvasTkAgg(f2, master=root)
        canvas2.get_tk_widget().pack(side=tkinter.BOTTOM, fill=tk.BOTH, expand=True)
        canvas2._tkcanvas.place(width=250, height=250, x=860, y=240)

        ############third graph###############
        f3 = Figure(figsize=(5, 5), dpi=100)
        a3 = f3.add_subplot(111)
        a3.plot(["oldpeak", "restecg", "cp"], [J, G, C])
        canvas3 = FigureCanvasTkAgg(f3, master=root)
        canvas3.get_tk_widget().pack(side=tkinter.BOTTOM, fill=tk.BOTH, expand=True)
        canvas3._tkcanvas.place(width=250, height=250, x=600, y=470)

        ############fourth graph###############
        f4 = Figure(figsize=(5, 5), dpi=100)
        a4 = f4.add_subplot(111)
        a4.plot(["slope", "ca", "thal"], [K, L, M])
        canvas4 = FigureCanvasTkAgg(f4, master=root)
        canvas4.get_tk_widget().pack(side=tkinter.BOTTOM, fill=tk.BOTH, expand=True)
        canvas4._tkcanvas.place(width=250, height=250, x=860, y=470)

        # Convert categorical variables to numeric
        sex_mapping = {"Male": 1, "Female": 0}
        fbs_mapping = {"True": 1, "False": 0}
        exang_mapping = {"Yes": 1, "No": 0}
        cp_mapping = {"1. Typical Angina": 1, "2. Atypical Angina": 2, "3. Non-Anginal
Pain": 3, "4. Asymptomatic": 4}
        restecg_mapping = {"0": 0, "1": 1, "2": 2}
        slope_mapping = {"1. Upsloping": 1, "2. Flat": 2, "3. Downsloping": 3}
        ca_mapping = {"0": 0, "1": 1, "2": 2, "3": 3, "4": 4}
        thal_mapping = {"0": 0, "1": 1, "2": 2, "3": 3, "4": 4}

        B = sex_mapping.get(B)
        F = fbs_mapping.get(F)
        I = exang_mapping.get(I)
        C = cp_mapping.get(C)
        G = restecg_mapping.get(G)
        K = slope_mapping.get(K)
        L = ca_mapping.get(L)
        M = thal_mapping.get(M)

        #input data
        input_data=(A,B,C,D,E,F,G,H,I,J,K,L,M)
        input_data_as_numpy_array=np.asanyarray(input_data)

        #reshape the numpy array as we are predicting for only on instances
        input_data_reshape=input_data_as_numpy_array.reshape(1,-1)

        prediction=model.predict(input_data_reshape)
        print(prediction[0])

        if(prediction[0]==0):
            print('The person does not have a Heart disease')
            report.config(text=f"Report:{0}",fg="#8dc63f")
            report1.config(text=f"{name}, you do not have a heart disease")

        else:
            print('The person has Heart disease')
            report.config(text=f"Report:{1}",fg="#ed1c24")
            report1.config(text=f"{name}, you  have a heart disease")

        # Assuming selection() function performs the analysis
        try:
            B = selection()
            print(f"Selection result: {B}")
        except Exception as e:
            print("An error occurred during selection:", e)
            messagebox.showerror("NEW USER", " NEW USER ADDED")
            return

        print("Analysis completed successfully.")

    except Exception as e:
        print("An error occurred during analysis:", e)
        messagebox.showerror("Error", "An error occurred during analysis.")

##info window  (operated by info button)>>

def Info():
    Icon_window = Toplevel(root)
    Icon_window.title("Info")
    Icon_window.geometry("700x600+100+100")

    # icon_image
    icon_image = PhotoImage(file="Images/info.png")
    Icon_window.iconphoto(False, icon_image)

    # Heading
    Label(Icon_window, text="Information Related to dataset", font="robot 19
bold").pack(padx=20, pady=20)

    # info
    Label(Icon_window, text="age - age in years", font="arial 11").place(x=20, y=100)
    Label(Icon_window, text="sex - sex (1 = male; 0 = female)", font="arial
11").place(x=20, y=130)
    Label(Icon_window, text="cp - chest pain type (0 = typical angina; 1 = atypical
angina; 2 = non-anginal pain; 3 = asymptomatic)",
        font="arial 11").place(x=20, y=160)
    Label(Icon_window, text="trestbps - resting blood pressure (in mm Hg on admission to
the hospital)",
        font="arial 11").place(x=20, y=190)
    Label(Icon_window, text="chol - serum cholesterol in mg/dl", font="arial
11").place(x=20, y=220)
    Label(Icon_window, text="fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)",
font="arial 11").place(
        x=20, y=250)
    Label(Icon_window, text="restecg - resting electrocardiographic results (0 = normal; 1
= having ST-T; 2 = hypertrophy)",
        font="arial 11").place(x=20, y=280)
    Label(Icon_window, text="thalach - maximum heart rate achieved", font="arial
11").place(x=20, y=310)
    Label(Icon_window, text="exang - exercise induced angina (1 = yes; 0 = no)",
font="arial 11").place(x=20, y=340)
    Label(Icon_window, text="oldpeak - ST depression induced by exercise relative to
rest", font="arial 11").place(
        x=20, y=370)
    Label(Icon_window, text="slope - the slope of the peak exercise ST segment (0 =
upsloping; 1 = flat; 2 = downsloping)",
        font="arial 11").place(x=20, y=400)
    Label(Icon_window, text="ca - number of major vessels (0-3) colored by flourosopy",
font="arial 11").place(x=20, y=430)
    Label(Icon_window, text="thal - 0 = normal; 1 = fixed defect; 2 = reversible defect",
font="arial 11").place(x=20, y=460)

    Icon_window.mainloop()

######################### it is use for closing window ##############################

def logout():
    root.destroy()

#clear

oldpeak = 0.0
thalach = 0
chol = 0
trestbps = 0

def clear():
    Name.get('')
    DOB.get('')
    trestbps.get('')
    chol.get('')
    thalach.set('')

    #####save
    def save():
        B2=Name.get()
        C2=Date.get()
        D2=DOB.get()

        today=datetime.date.today()
        E2=today.year-DOB.get()
        def selection():
            pass

        try:
            F2=selection()
        except:
            messagebox.showerror("Missing Data","Please select gender!")

            def selection2():
                pass

        try:
            J2=selection2()
        except:
            messagebox.showerror("Missing Data","Please select fbs!")

            def selection3():
                pass

        try:
            M2=selection3()
        except:
            messagebox.showerror("Missing Data","Please select exang!")

            def selection4():
                pass

        try:
            G2=selection4()
        except:
            messagebox.showerror("Missing Data","Please select CP!")

        try:
            K2=restecg_combobox.get()
        except:
            messagebox.showerror("Missing Data","Please select restecg!")

            def selection5():
                pass

        try:
            O2=selection5()
        except:
            messagebox.showerror("Missing Data","Please select slope!")

        try:
            P2=ca_combobox.get()
        except:
            messagebox.showerror("Missing Data","Please select ca!")

        try:
            Q2=thal_combobox.get()
        except:
            messagebox.showerror("Missing Data","Please select thal!")

        H2=trestbps.get()
        I2=chol.get()
        L2=thalach.get()
        N2=float(oldpeak.get())

        print(B2)
        print(C2)
        print(D2)
        print(E2)
        print(F2)
        print(G2)
        print(H2)
        print(I2)
        print(J2)
        print(K2)
        print(L2)
        print(M2)
        print(N2)
        print(O2)
        print(P2)
        print(Q2)

def set_icon():
    image_icon = PhotoImage(file="Images/icon.png")
    root.iconphoto(False, image_icon)


root.after(100, set_icon)

# Header section
logo = PhotoImage(file="Images/header.png")
myimage = Label(image=logo, bg=background)
myimage.place(x=0, y=0)

# Frame
Heading_entry = Frame(root, width=800, height=190, bg="#df2d4b")
Heading_entry.place(x=600, y=20)
Label(Heading_entry, text="Registration No.", font="arial 13", bg="#df2d4b",
fg=frame).place(x=30, y=0)
Label(Heading_entry, text="Date", font="arial 13", bg="#df2d4b", fg=frame).place(x=430,
y=0)
Label(Heading_entry, text="Patient Name", font="arial 13", bg="#df2d4b",
fg=frame).place(x=30, y=90)
Label(Heading_entry, text="Birth Year", font="arial 13", bg="#df2d4b",
fg=frame).place(x=430, y=90)

Entry_image = PhotoImage(file="Images/Rounded Rectangle 1.png")
Entry_image2 = PhotoImage(file="Images/Rounded Rectangle 2.png")
Label(Heading_entry, image=Entry_image, bg="#df2d4b").place(x=20, y=30)
Label(Heading_entry, image=Entry_image, bg="#df2d4b").place(x=430, y=30)

Label(Heading_entry, image=Entry_image2, bg="#df2d4b").place(x=20, y=120)
Label(Heading_entry, image=Entry_image2, bg="#df2d4b").place(x=430, y=120)

Registration = tk.IntVar()
reg_entry = tk.Entry(Heading_entry, textvariable=Registration, width=30, font="arial 15",
bg="#0e5363", fg="white", bd=0)
reg_entry.place(x=30, y=45)

Date = StringVar()
today = date.today()
```

```
d1 = today.strftime("%d/%m/%y")
date_entry = Entry(Heading_entry, textvariable=Date, width=15, font='arial 15',
bg="#0e5363", fg="white", bd=0)
date_entry.place(x=500, y=45)
Date.set(d1)

Name = StringVar()
name_entry = Entry(Heading_entry, textvariable=Name, width=20, font="arial 20",
bg="#ededed", fg="#222222", bd=0)
name_entry.place(x=30, y=130)

# Define Date of Birth variable and Entry widget
DOB = IntVar()
# Define Date of Birth variable and Entry widget
DOB = IntVar()
dob_entry = Entry(Heading_entry, textvariable=DOB, width=20, font="arial 20",
bg="#ededed", fg="#222222", bd=0)
dob_entry.place(x=500, y=130)

# Function to calculate age and display it
def calculate_age():
    birth_year = DOB.get()
    if birth_year != 0:  # Ensure birth year is provided
        current_year = today.year
        age = current_year - birth_year
        Label(Heading_entry, text=f"Age: {age}", font="arial 13", bg="#df2d4b",
fg=frame).place(x=30, y=160)
    else:
        # Display error message if birth year is not provided
        Label(Heading_entry, text="Please enter Birth Year", font="arial 13",
bg="#df2d4b", fg="red").place(x=30, y=160)

# Button to trigger age calculation
calculate_button = Button(Heading_entry, text="Calculate Age", command=calculate_age)
calculate_button.place(x=640, y=130)

# Frame for radio buttons
Radio_frame = Frame(root, width=550, height=100, bg="#dbe0e3")
Radio_frame.place(x=10, y=400)

# Sex radio buttons
sex_var = StringVar()
sex_var.set(None)

# Adjusted positions and sizes for the sex, fbs, and exang buttons
Label(Radio_frame, text="Sex:", font="arial 10", bg="blue", fg="black").place(x=10, y=10)
Radiobutton(Radio_frame, text="Male", variable=sex_var, value="Male",
bg="#dbe0e3").place(x=60, y=10)
Radiobutton(Radio_frame, text="Female", variable=sex_var, value="Female",
bg="#dbe0e3").place(x=130, y=10)
```

```
# FBS radio buttons
fbs_var = StringVar()
fbs_var.set(None)

Label(Radio_frame, text="FBS:", font="arial 10", bg="blue", fg="black").place(x=200, y=10)
Radiobutton(Radio_frame, text="True", variable=fbs_var, value="True",
bg="#dbe0e3").place(x=240, y=10)
Radiobutton(Radio_frame, text="False", variable=fbs_var, value="False",
bg="#dbe0e3").place(x=310, y=10)

# Exang radio buttons
exang_var = StringVar()
exang_var.set(None)

Label(Radio_frame, text="Exang:", font="arial 10", bg="blue", fg="black",
padx=0).place(x=370, y=10)
Radiobutton(Radio_frame, text="Yes", variable=exang_var, value="Yes",
bg="#dbe0e3").place(x=410, y=10)
Radiobutton(Radio_frame, text="No", variable=exang_var, value="No",
bg="#dbe0e3").place(x=470, y=10)

# Detail entry frame
Detail_entry = Frame(root, width=550, height=200, bg="#dbe0e3")
Detail_entry.place(x=10, y=500)

# Labels and comboboxes for "cp", "restecg", "slope", "ca", and "thal"
Label(Detail_entry, text="cp:", font="arial 13", bg="#dbe0e3", fg="black").place(x=10,
y=10)
cp_combobox = Combobox(Detail_entry, values=['1. Typical Angina', '2. Atypical Angina',
'3. Non-Anginal Pain', '4. Asymptomatic'], font="Arial 13", state="readonly", width=15)
cp_combobox.place(x=120, y=10)

Label(Detail_entry, text="restecg:", font="arial 13", bg="#dbe0e3",
fg="black").place(x=10, y=50)
restecg_combobox = Combobox(Detail_entry, values=['0', '1', '2'], font="Arial 13",
state="readonly", width=15)
restecg_combobox.place(x=120, y=50)

Label(Detail_entry, text="slope:", font="arial 13", bg="#dbe0e3", fg="black").place(x=10,
y=90)
slope_combobox = Combobox(Detail_entry, values=['1. Upsloping', '2. Flat', '3.
Downsloping'], font="Arial 13", state="readonly", width=15)
slope_combobox.place(x=120, y=90)

Label(Detail_entry, text="ca:", font="arial 13", bg="#dbe0e3", fg="black").place(x=10,
y=130)
ca_combobox = Combobox(Detail_entry, values=['0', '1', '2', '3', '4'], font="Arial 13",
state="readonly", width=15)
ca_combobox.place(x=120, y=130)
```

```python
Label(Detail_entry, text="thal:", font="arial 13", bg="#dbe0e3", fg="black").place(x=10,
y=170)
thal_combobox = Combobox(Detail_entry, values=['0', '1', '2', '3', '4'], font="Arial 13",
state="readonly", width=15)
thal_combobox.place(x=120, y=170)

# Labels for smoking, trestbps, chol, thalach, and oldpeak
Label(Detail_entry, text="Smoking:", font="arial 13", bg="#dbe0e3",
fg="black").place(x=280, y=10)
smoking_entry = Entry(Detail_entry, font="Arial 13", bg="#dbe0e3", fg="black", width=15)
smoking_entry.place(x=400, y=10)

Label(Detail_entry, text="trestbps:", font="arial 13", bg="#dbe0e3",
fg="black").place(x=280, y=50)
trestbps_entry = Entry(Detail_entry, font="Arial 13", bg="#dbe0e3", fg="black", width=15)
trestbps_entry.place(x=400, y=50)

Label(Detail_entry, text="chol:", font="arial 13", bg="#dbe0e3", fg="black").place(x=280,
y=90)
chol_entry = Entry(Detail_entry, font="Arial 13", bg="#dbe0e3", fg="black", width=15)
chol_entry.place(x=400, y=90)

Label(Detail_entry, text="thalach:", font="arial 13", bg="#dbe0e3",
fg="black").place(x=280, y=130)
thalach_entry = Entry(Detail_entry, font= "Arial 13", bg="#dbe0e3", fg="black", width=15)
thalach_entry.place(x=400, y=130)
Label(Detail_entry, text="oldpeak:", font="arial 13", bg="#dbe0e3",
fg="black").place(x=280, y=170)
oldpeak_entry = Entry(Detail_entry, font="Arial 13", bg="#dbe0e3", fg="black", width=15)
oldpeak_entry.place(x=400, y=170)

square_report_image = PhotoImage(file="Images/Report.png")
report_background = Label(image=square_report_image, bg=background)
report_background.place(x=1120, y=400)

report = Label(root, font="arial 25 bold", bg="white", fg="#8dc63f")
report.place(x=1170, y=550)

report1 = Label(root, font="arial 10 bold", bg="white")
report1.place(x=1130, y=610)

#####################Graph#########################

# Load the image and scale it
graph_image = PhotoImage(file="Images/graph.png").zoom(1)

# Create labels and place them
labels = []
positions = [(600, 270), (860, 270), (600, 500), (860, 500)]

for x, y in positions:
    label = tk.Label(root, image=graph_image)
    label.place(x=x, y=y)
    labels.append(label)

####################Button###########################
analysis_button = PhotoImage(file="Images/Analysis.png")
Button(root, image=analysis_button, bd=0, bg=background, cursor='hand2',
command=analysis).place(x=1130, y=240)

###################info button#######################

info_button = PhotoImage(file="Images/info.png")
Button(root, image=info_button, bd=0, bg=background, cursor='hand2',
command=Info).place(x=10, y=240)

# Define the save function
def save(file_path, data):
    with open(file_path, 'w') as f:
        for item in data:
            f.write("%s\n" % item)

# Call the save function with 'file' and 'arr' arguments
save("example.txt", [1, 2, 3])

# Define a function to handle button click event
def on_save_button_click():
    # Get the file path from the user
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text
files", "*.txt"), ("All files", "*.*")])
    if file_path:
        # Assuming data is a list of strings
        data = [
            "G is restecg: 0",
            "H is thalach: 34",
            "I is exang: Yes",
            "J is oldpeak: 54.0",
            "K is slope: 1. Upsloping",
            "L is ca: 4",
            "M is thal: 2"
        ]

######################save button #####################

save_button = PhotoImage(file="Images/save.png")
Button(root, image=save_button, bd=0, bg=background,
cursor='hand2',command=np.save).place(x=1370, y=250)

###############smoking and non-smoking button#############
button_mode = True
choice = "smoking"
```

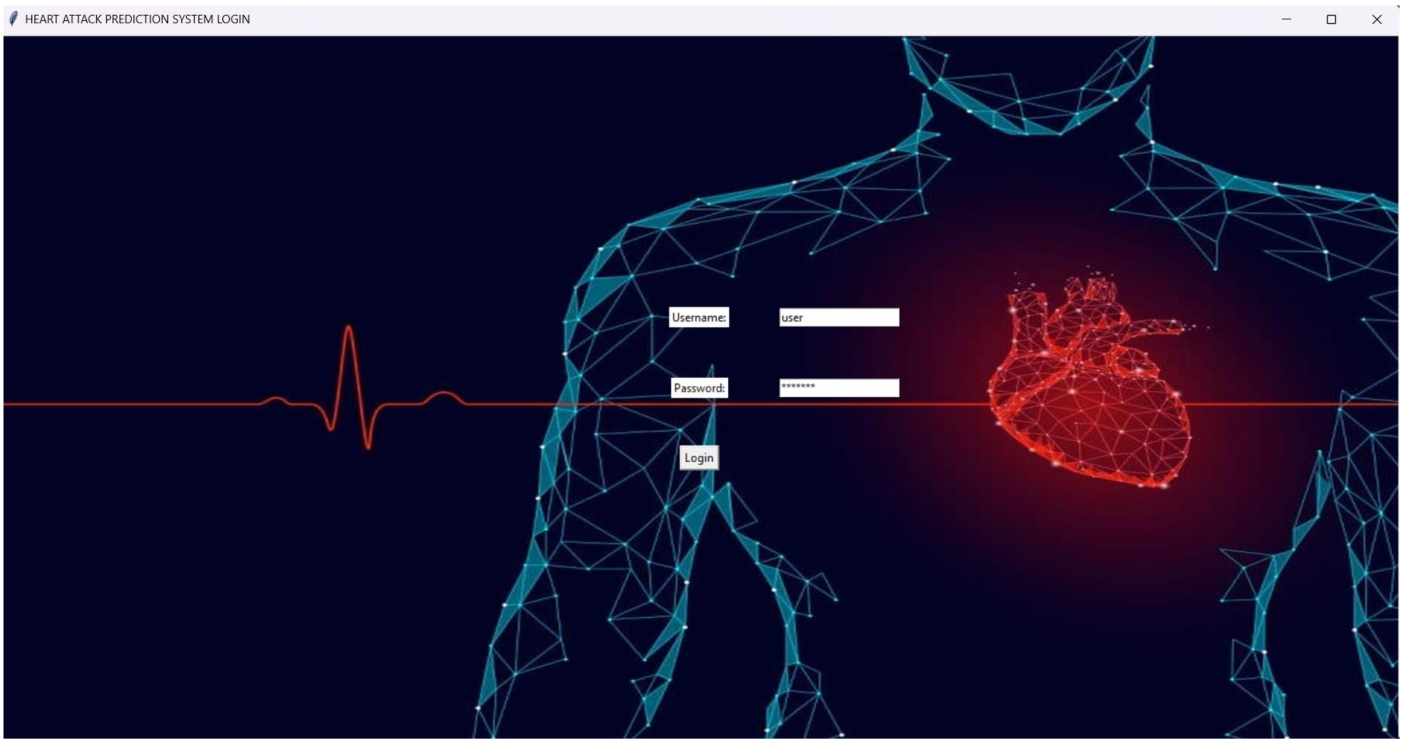
```python
def changemode():
    global button_mode
    global choice

    if button_mode:
        choice = "non_smoking"
        mode.config(image=non_smoking_icon, activebackground="white")
        button_mode = False
    else:
        choice = "smoking"
        mode.config(image=smoking_icon, activebackground="white")
        button_mode = True
        print(choice)

smoking_icon = PhotoImage(file="Images/smoker.png")
non_smoking_icon = PhotoImage(file="Images/non-smoker.png")
mode = Button(root, image=smoking_icon, bg="#dbe0e3", bd=0, cursor="hand2",
command=changemode)
mode.place(x=430, y=505)

######################logout button###########################

logout_icon = PhotoImage(file="Images/logout.png")
logout_button = Button(root, image=logout_icon, bg="#df2d4b", cursor="hand2", bd=0,
command=logout)
logout_button.place(x=1130, y=660)
print
root.mainloop()
```

backend.py

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

heart_data = pd.read_csv('heart.csv')

heart_data.isnull().sum()

# statistical measures about the data
heart_data.describe()

# checking the distribution of Target Variable
heart_data['target'].value_counts()

X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y,
random_state=2)

model = LogisticRegression()

model.fit(X_train, Y_train)

# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

RESULTS AND SCREENSHOTS

Login Page

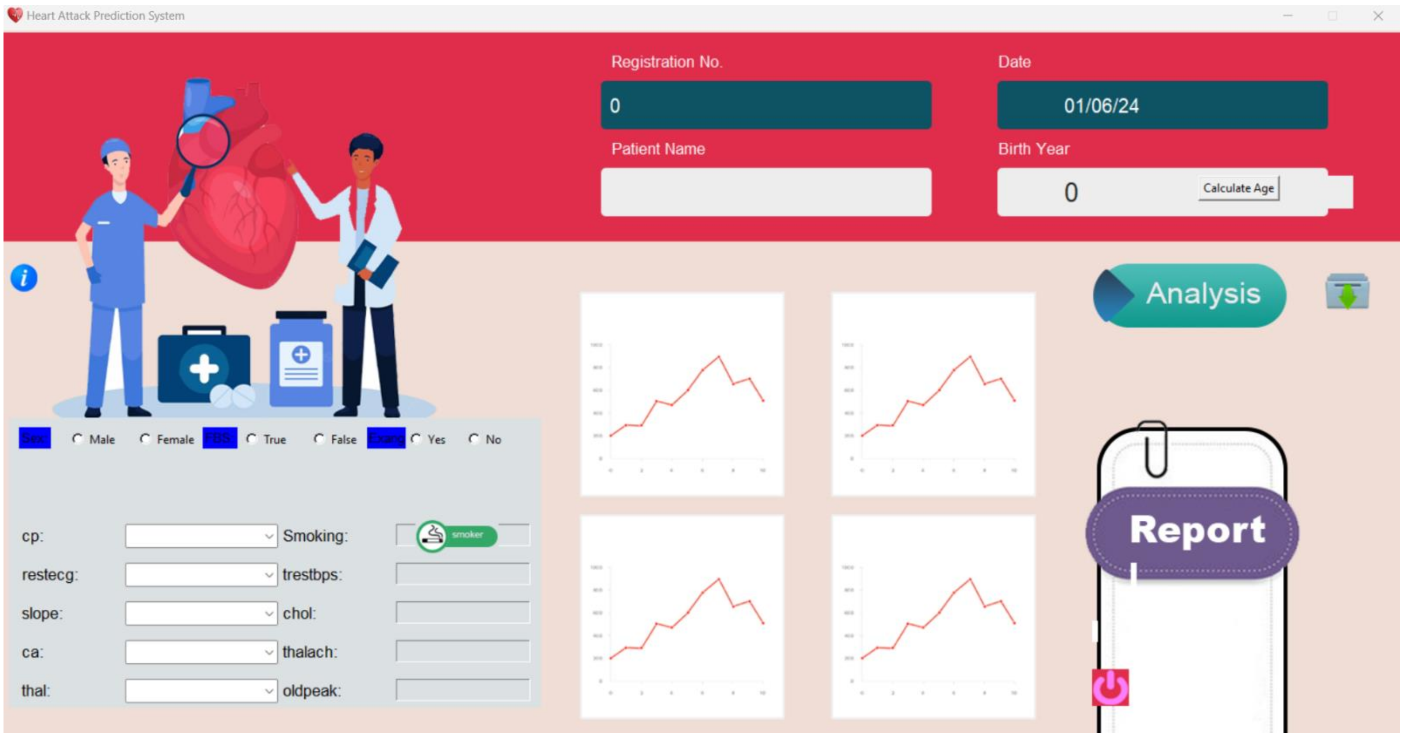


HEART ATTACK PREDICTION SYSTEM LOGIN

Username: user

Password: *******

Main Window



Heart Attack Prediction System

Registration No. 0

Date 01/06/24

Patient Name

Birth Year 0

Sex: ○ Male ○ Female FBS: ○ True ○ False Exang ○ Yes ○ No

cp:

Smoking:

restecg:

trestbps:

slope:

chol:

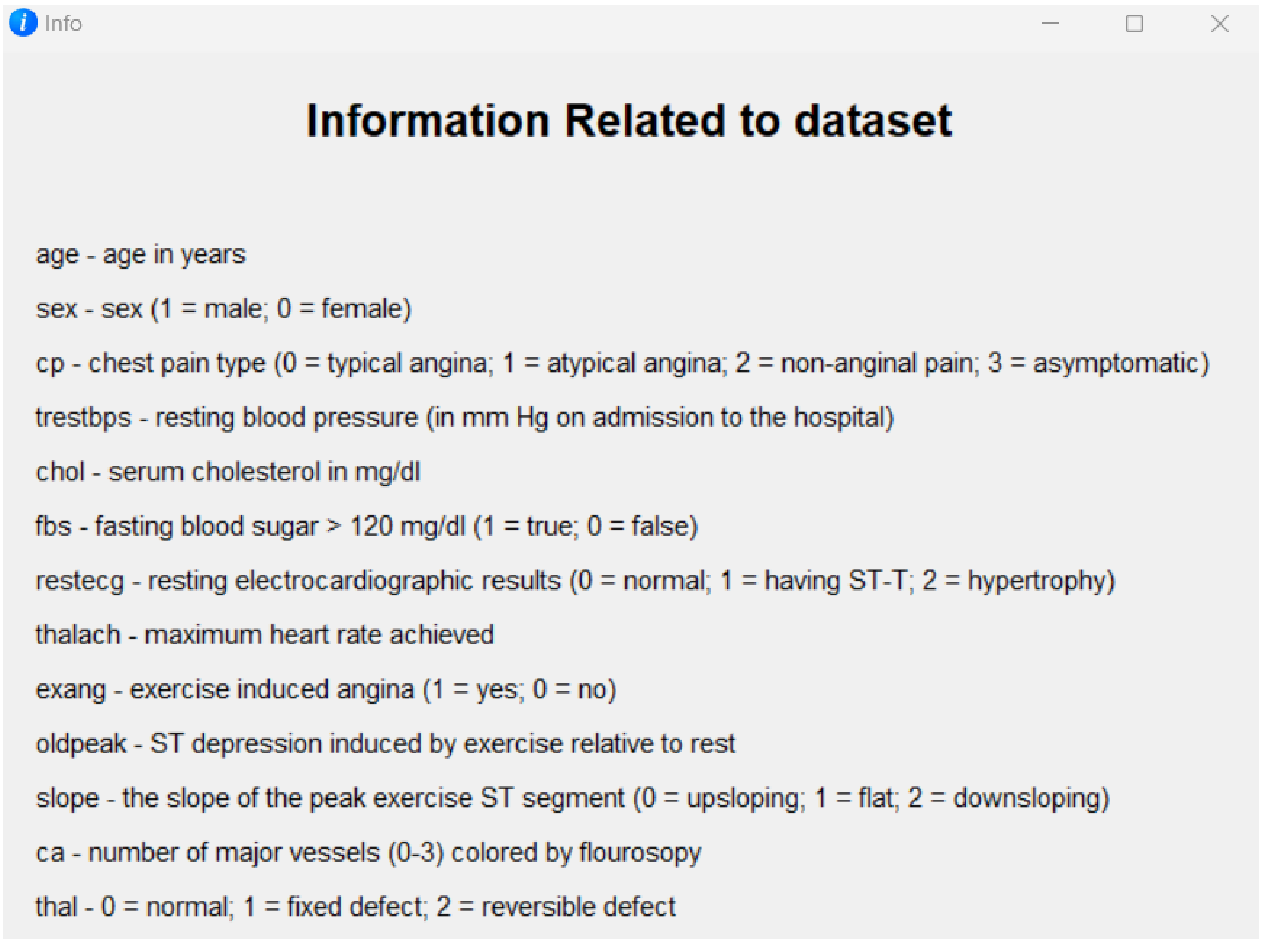
ca:

thalach:

thal:

oldpeak:

Information Related to dataset



Info

Information Related to dataset

age - age in years

sex - sex (1 = male; 0 = female)

cp - chest pain type (0 = typical angina; 1 = atypical angina; 2 = non-anginal pain; 3 = asymptomatic)

trestbps - resting blood pressure (in mm Hg on admission to the hospital)

chol - serum cholesterol in mg/dl

fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)

restecg - resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)

thalach - maximum heart rate achieved

exang - exercise induced angina (1 = yes; 0 = no)

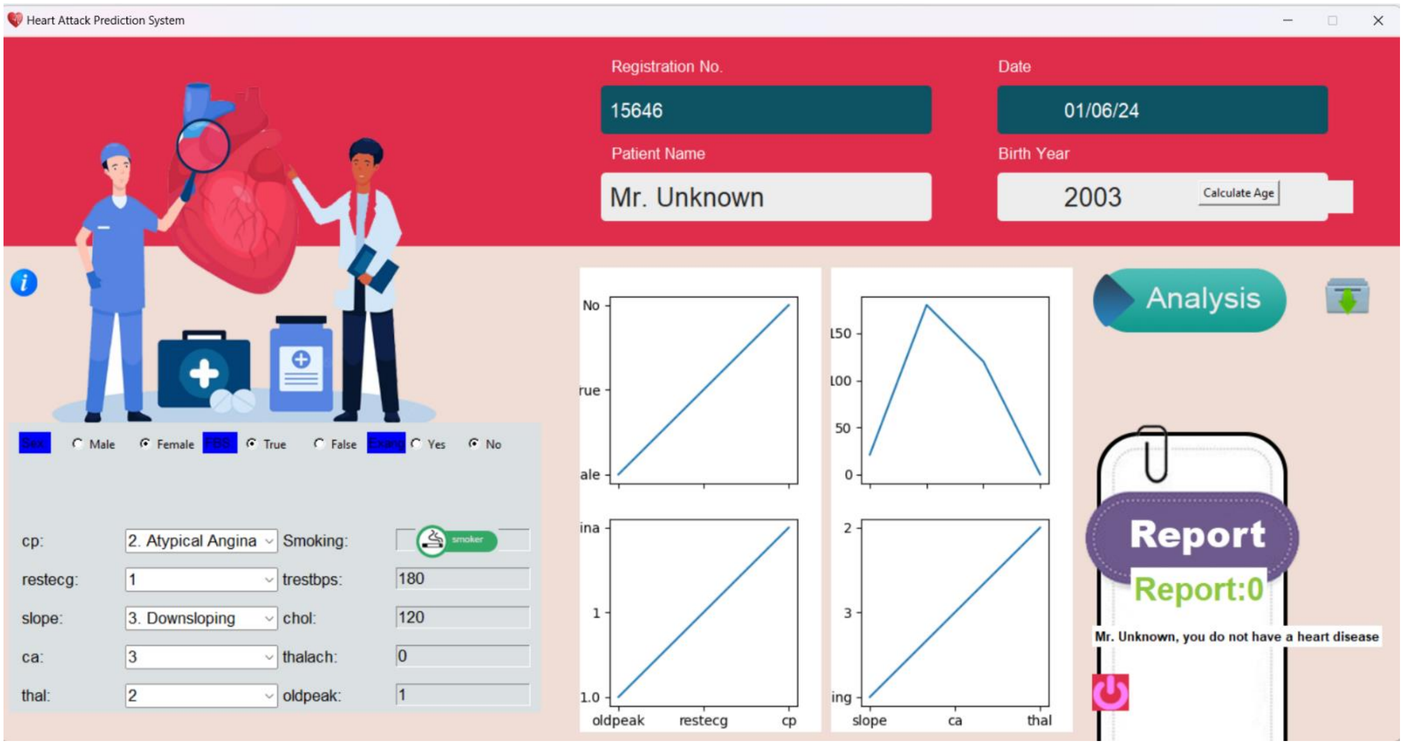
oldpeak - ST depression induced by exercise relative to rest

slope - the slope of the peak exercise ST segment (0 = upsloping; 1 = flat; 2 = downsloping)

ca - number of major vessels (0-3) colored by flourosopy

thal - 0 = normal; 1 = fixed defect; 2 = reversible defect

Report Generation after entering the data (GRAPHS & REPORT)



Heart Attack Prediction System

Registration No. 15646

Date 01/06/24

Patient Name Mr. Unknown

Birth Year 2003

Sex: Male [✓ Female]

FBS: [✓ True] False

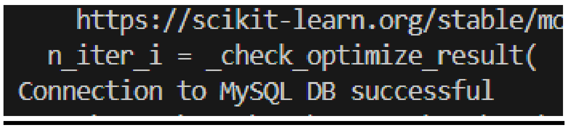
Exang: Yes [✓ No]

cp:	2. Atypical Angina	Smoking:	smoker
restecg:	1	trestbps:	180
slope:	3. Downsloping	chol:	120
ca:	3	thalach:	0
thal:	2	oldpeak:	1

Report:0

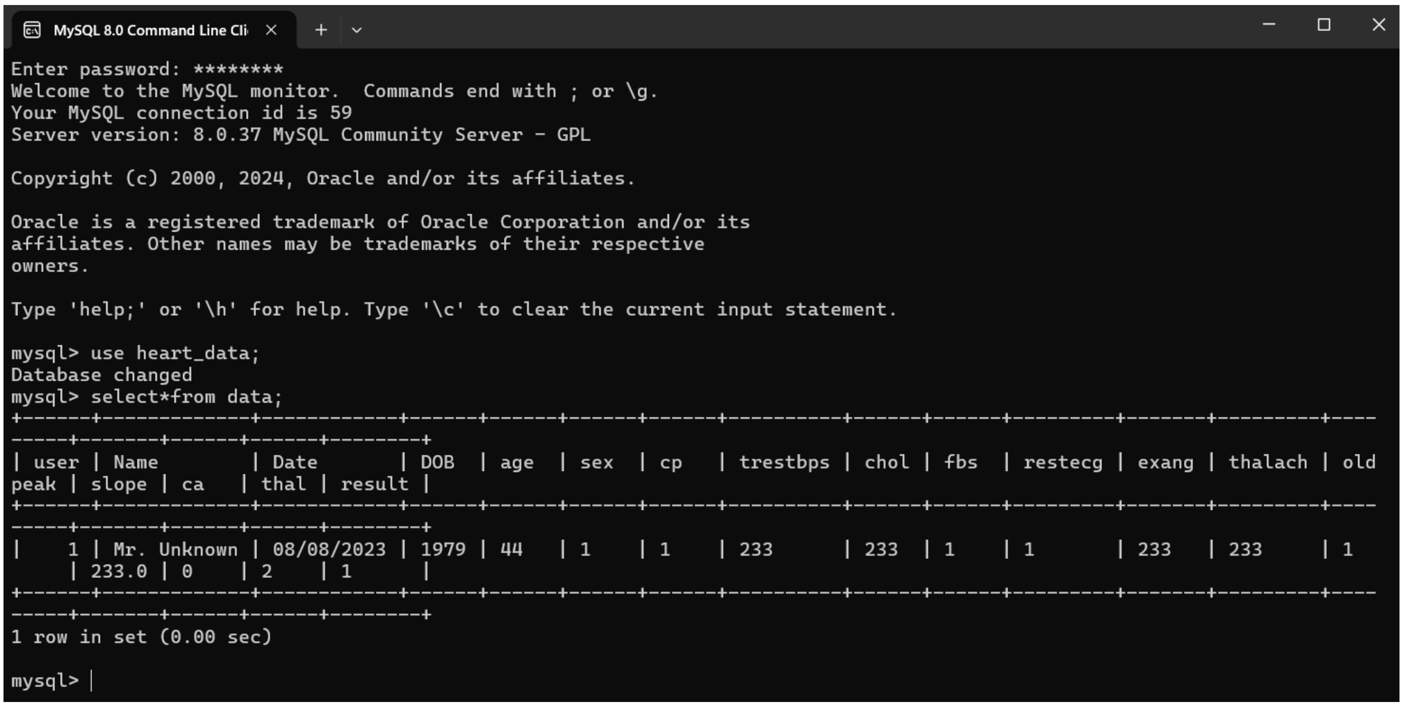
Mr. Unknown, you do not have a heart disease

DataBase is successfully connected



```
    https://scikit-learn.org/stable/mo
  n_iter_i = _check_optimize_result(
Connection to MySQL DB successful
```

Data that is saved in MySQL database



```
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 8.0.37 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use heart_data;
Database changed
mysql> select*from data;
+------+-------------+------------+------+------+------+------+----------+------+------+---------+-------+---------+----
-----+-------+------+------+--------+
| user | Name        | Date       | DOB  | age  | sex  | cp   | trestbps | chol | fbs  | restecg | exang | thalach | old
peak | slope | ca   | thal | result |
+------+-------------+------------+------+------+------+------+----------+------+------+---------+-------+---------+----
-----+-------+------+------+--------+
|    1 | Mr. Unknown | 08/08/2023 | 1979 | 44   | 1    | 1    | 233      | 233  | 1    | 1       | 233   | 233     | 1
     | 233.0 | 0    | 2    | 1      |
+------+-------------+------------+------+------+------+------+----------+------+------+---------+-------+---------+----
-----+-------+------+------+--------+
1 row in set (0.00 sec)

mysql> |
```

CONCLUSTION AND FUTURE ENHANCEMENT

Conclusion:

In conclusion, the heart attack prediction system represents a significant advancement in healthcare technology, offering a proactive approach to identifying individuals at risk of heart attack. By leveraging the power of machine learning algorithms and predictive modeling techniques, this innovative system provides healthcare professionals with valuable insights that can facilitate early intervention and personalized treatment strategies, ultimately improving patient outcomes and saving lives. The implementation of the heart attack prediction system has demonstrated promising results, with accurate predictions achieved through rigorous data preprocessing, feature engineering, and model development. Extensive data preprocessing ensures the quality and consistency of the input data, while sophisticated feature engineering techniques help in extracting the most relevant attributes that contribute to the prediction process. Furthermore, the careful development and fine-tuning of predictive models, including the selection of appropriate machine learning algorithms and optimization of hyperparameters, have led to high-performance metrics. The evaluation of these models, through various validation techniques and performance metrics such as accuracy, precision, recall, and the area under the ROC curve (AUC), has shown their effectiveness in reliably identifying individuals at risk of heart attack. This high level of accuracy and reliability in predictions underscores the system's potential as a critical tool in the arsenal of modern healthcare, aiming to reduce the incidence of heart attacks and improve the overall quality of patient care.

Future Enhancement:

Moving forward, further enhancements and refinements can be made to the heart attack prediction system to enhance its utility and performance. Future enhancements may include:

- **Integration of Additional Data Sources:** Incorporating additional data sources such as genetic information, lifestyle factors, and environmental variables can enrich the predictive capabilities of the system and provide a more comprehensive understanding of heart attack risk factors.

- **Fine-Tuning of Models:** Continuously fine-tuning the machine learning models based on feedback from real-world data can improve their accuracy and robustness, ensuring reliable predictions across diverse patient populations and healthcare settings.

- **Real-Time Monitoring and Alerts:** Implementing real-time monitoring capabilities and automated alerting systems can enable timely intervention by notifying healthcare providers of individuals at imminent risk of heart attack, allowing for prompt medical attention and preventive measures.

- **Patient-Centric Approach:** Adopting a patient-centric approach by tailoring the prediction models to individual patient profiles and preferences can enhance patient engagement and adherence to preventive strategies, leading to better health outcomes and quality of life.

- **Collaborative Research and Validation:** Partnering with healthcare institutions, research organizations, and regulatory bodies for large-scale validation studies and clinical trials can validate the system's efficacy and reliability, facilitating widespread adoption in clinical practice.

REFERENCES

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://archive.ics.uci.edu/dataset/45/heart+disease
- https://www.researchgate.net/publication/368848738_Heart_Disease_Prediction_Using_Logistic_Regression#:~:text=The%20main%20objective%20is%20to,more%20sets%20of%20independent%20variables
- https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
- https://towardsdatascience.com/logistic-regression-for-binary-classification-56a2402e62e6
- https://www.coursera.org/
- https://www.analyticsvidhya.com/blog/2022/02/heart-disease-prediction-using-machine-learning/#:~:text=Which%20algorithm%20is%20best%20for,decision%20trees%2C%20and%20random%20forests
- https://www.youtube.com/