



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

**HEART ATTACK PREDICTION AND PATIENT
RECORD MANAGEMENT SYSTEM**
A MINI PROJECT REPORT

Submitted by

UMESH SARATHY S K

231501177

VISHNU SELVAM A

231501186

KIRAN RAJAN S

231501507

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025



BONAFIDE CERTIFICATE

Certified that this project report **“HEART ATTACK PREDICTION AND PATIENT RECORD MANAGEMENT SYSTEM”** is the bonafide work of **“UMESH SARATHY S K (231501177) ,**
VISHNU SELVAM (231501186),
KIRAN RAJAN (231501507)” who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Patient Health Monitoring and Heart Attack Risk Prediction System

The increasing prevalence of cardiovascular diseases highlights the importance of real-time health monitoring and predictive analytics. This project aims to design and develop an intelligent system that manages patient health records, tracks vital signs, and predicts the likelihood of heart attacks based on key health parameters. The system integrates data storage, real-time analysis, and predictive algorithms to assist healthcare professionals in making informed decisions and improving patient outcomes.

Core Objectives:

The system will maintain a secure and structured repository of patient health data, including essential metrics like blood pressure (BP), blood sugar levels, heart rate, and other medical details. All records will be stored in a MySQL database to ensure scalability, reliability, and ease of access.

The system provides a robust interface for updating and reviewing patient health metrics regularly. It facilitates trend analysis, enabling patients and doctors to identify patterns and deviations in vital signs over time.

Leveraging advanced machine learning techniques, the system will analyze historical data and vital statistics to predict the probability of heart attacks. Algorithms such as logistic regression, random forests, or deep learning models will be trained using clinical datasets to achieve high accuracy in predictions.

The system will include a mechanism to trigger alerts for critical conditions, such as dangerously high blood pressure or sugar levels, and inform caregivers or medical professionals for immediate intervention.

Comprehensive reporting features will allow users to generate visual dashboards and reports on patient health metrics, enabling better decision-making through graphical insights such as line charts, bar graphs, and heatmaps.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 INTRODUCTION

1.2 OBJECTIVES

1.3 MODULES

2. SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

2.2 LANGUAGES

2.2.1 SQL

2.2.2 PYTHON

3. DATABASE STRUCTURE

3.1 ER DIAGRAM

3.2 NORMALIZATION

4. PROGRAM CODE

5. PROJECT SCREENSHOTS

6. CONCLUSION

7. REFERENCE

1. INTRODUCTION

1.1 Introduction

Cardiovascular diseases (CVDs) are one of the leading causes of mortality worldwide, with heart attacks being among the most prevalent and life-threatening conditions. Early detection and timely intervention can significantly reduce the risk of severe outcomes and improve survival rates. However, managing patient health records and identifying potential risks proactively remains a challenge in the current healthcare landscape. To address these challenges, this project introduces a **Patient Health Monitoring and Heart Attack Risk Prediction System**, a technology-driven solution aimed at streamlining patient data management while incorporating predictive analytics for better preventive care.

The proposed system is designed to maintain comprehensive patient health records, such as blood pressure (BP), blood sugar levels, and other critical metrics, in a structured and secure manner using a **MySQL database**. Beyond just record-keeping, the system incorporates a **predictive analytics module** that uses machine learning algorithms to assess the likelihood of heart attack risks based on these health metrics. By doing so, it empowers healthcare professionals and patients to make informed decisions and take preventive actions.

This system is particularly valuable in the context of modern healthcare, where data-driven insights play a pivotal role in improving patient outcomes. It not only ensures efficient data management but also provides real-time insights into patient health trends. The integration of predictive tools further enhances the system's ability to identify at-risk individuals early, supporting proactive and preventive healthcare strategies.

1.2 Objectives

Objectives of the System:

1. **Efficient Health Record Management:** To maintain structured, accurate, and up-to-date patient health data, accessible to healthcare professionals for monitoring and analysis.
2. **Risk Prediction:** To utilize advanced analytics for predicting the risk of heart attacks based on vital health parameters.
3. **User-Friendly Interface:** To offer a simple and intuitive platform for users, enabling seamless data entry, monitoring, and visualization of trends.

-
4. **Proactive Alerts:** To notify patients and healthcare providers about critical health conditions for timely intervention.

The integration of data management with predictive capabilities aligns with the broader goal of shifting from reactive to proactive healthcare. By addressing both the operational and analytical needs of patient monitoring, this system contributes to a more efficient, scalable, and patient-centric approach to healthcare delivery.

This project thus represents a step forward in leveraging technology to combat life-threatening conditions, ensuring better access to preventive care and ultimately improving health outcomes at both individual and community levels.

1.3 Modules

This system is divided into several modules, each handling a specific part of the application functionality:

1.3.1 Frontend Module

Frontend: Tkinter for Desktop GUI

- The **Tkinter** module in Python is used to build a desktop application for interacting with the system. It allows users to input health data, view stored records, and check heart attack risk predictions.
- **Key Features of Tkinter:**
 - **Widgets:** Tkinter provides various widgets such as Label, Entry, Button, Frame, and Treeview for creating interactive elements like forms, tables, and buttons.
 - **Ease of Use:** Tkinter is easy to learn and integrates seamlessly with Python, making it suitable for lightweight applications.
 - **Event Handling:** Provides event-driven programming to handle user actions like button clicks.

1.3.2 Backend Module

The backend is designed to handle the application logic, store data, and run machine learning predictions. It integrates with the Tkinter frontend seamlessly.

Key Backend Features:

1. **Data Storage:** Use MySQL to store patient data such as blood pressure, sugar levels, and prediction results.
2. **Prediction:** Process data through a machine learning model to predict heart attack risk.
3. **APIs for Data Communication:** Use Python functions to serve as the interface between the Tkinter frontend and the database or prediction module.

Backend Libraries:

- **SQLite/MySQL:** For database storage.
- **Scikit-learn:** For machine learning models.
- **Pandas:** For data manipulation.
- **Joblib:** To load and use pre-trained machine learning models.

1.3.3 Database Module

Database: MySQL Server

- **Purpose:** Provides a robust and scalable solution for storing patient health data securely.
- **Features:**
 - Relational database structure to store patient metrics (e.g., BP, sugar levels) and prediction results.
 - SQL queries for data insertion, retrieval, and management.
 - Scalability to handle large volumes of data.
- **Advantages:**
 - Reliable and secure for multi-user environments.
 - Well-suited for healthcare applications requiring precise data management.

- Easy integration with Python through connectors like mysql-connector-python.

1.3.4 Models Module

Machine Learning: scikit-learn

Purpose: Predicts the risk of heart attack based on patient health metrics using a pre-trained machine learning model.

Features:

- Provides a robust suite of tools for model training, testing, and evaluation.
- Supports supervised learning algorithms like Logistic Regression, Decision Trees, and Random Forest, ideal for binary risk prediction (e.g., "High Risk" vs. "Low Risk").
- Handles data preprocessing and feature scaling to improve model accuracy.

Advantages:

- Open-source and well-documented.
- Easily integrates with Python for seamless prediction workflows.
- Widely used in healthcare predictive analytics.

2. SURVEY OF TECHNOLOGIES

2.1 Software Description

The **Patient Health Monitoring and Heart Attack Risk Prediction System** is a comprehensive software solution designed to monitor and maintain patient health data, including key metrics such as **blood pressure (BP)**, **sugar levels**, and more. This system uses machine learning to predict the risk of a heart attack based on the patient's health records and stores these records in a secure **MySQL database server** for future analysis and retrieval. The application provides an intuitive user interface for healthcare professionals or patients to input data, view predictions, and access historical records.

Key Features

1. Patient Data Management:

- The system allows healthcare professionals or patients to input vital health metrics, including **blood pressure**, **sugar levels**, and other relevant data.
- Data is stored securely in a **MySQL database** for easy retrieval and future analysis.

2. Heart Attack Risk Prediction:

- The system uses machine learning algorithms (via the **scikit-learn** library) to assess the patient's risk of a heart attack based on their health data.
- The prediction is made using a pre-trained model that takes **BP**, **sugar levels**, and other factors into account, providing a **low-risk** or **high-risk** result.

3. Data Storage:

- All health records, including input data and prediction results, are stored in a **MySQL** relational database.
- The database is designed to handle large amounts of patient data, providing efficient storage and retrieval options.

4. User-Friendly Interface:

- The system features a **desktop GUI** built using the **Tkinter** framework, providing an intuitive and easy-to-use interface for both healthcare providers and patients.
- Users can input health metrics, view real-time predictions, and access historical data from a simple, easy-to-navigate window.

5. Data Retrieval and Reporting:

- Patients and healthcare providers can view stored records, which include vital health statistics, prediction outcomes, and any changes to patient health over time.
- The system supports the retrieval of data through SQL queries and presents it in a structured, user-friendly format.

6. Security:

- The system ensures that all sensitive health data is securely stored in a **MySQL database**, ensuring confidentiality and preventing unauthorized access.

2.2 Languages

The core languages used in the development of this application are SQL and Python, each playing a crucial role in different parts of the system.

2.2.1 SQL

SQL (Structured Query Language) is a standard programming language used for managing and manipulating relational databases. In this project, SQL is primarily used in the following ways:

- **Database Creation and Management:** SQL commands are used to create the book_recommendation database and the books table, as defined in the schema.sql script. SQL statements like CREATE TABLE, INSERT, SELECT, and UPDATE define the database structure, manage entries, and modify data as needed.
- **Data Insertion:** SQL is used to insert book data from CSV files into the database. This is done using the import_csv_to_db.py script, where SQL INSERT statements add each book's details (title, author, genres, etc.) to the books table.
- **Data Querying:** SQL SELECT statements retrieve book records from the database based on user search and sorting preferences. These queries, generated dynamically in the backend, filter and sort data based on fields like title, author, and average rating, providing only the relevant records to the user.

Advantages of SQL:

- **Structured Data Management:** SQL is ideal for managing structured data with defined relationships, which makes it well-suited for this project where each book record has specific fields.
- **Efficient Querying:** SQL provides powerful capabilities for filtering, sorting, and aggregating data, allowing for quick response times in applications with large datasets.

2.2.2 Python

Python is an interpreted, high-level, and versatile programming language, known for its readability and robust libraries. Python is the primary language for the backend in this project,

performing tasks like handling HTTP requests, processing data, and interacting with the database.

- **MySQL Connector/Python Library:** Python's `mysql.connector` library allows secure and efficient interactions with the MySQL database. The `get_db_connection` function in `database.py` uses this library to establish a connection with the database and execute queries.

Advantages of Python:

- **Rapid Development:** Python's simplicity and extensive libraries reduce development time, allowing for quick implementation and testing.
- **Strong Database Integration:** Python libraries like `mysql.connector` enable seamless communication with databases, making it efficient to handle and manage large datasets.
- **Versatile Frameworks:** Flask, a lightweight framework, provides the flexibility needed for a customized backend, while other Python libraries support everything from data handling to web development.

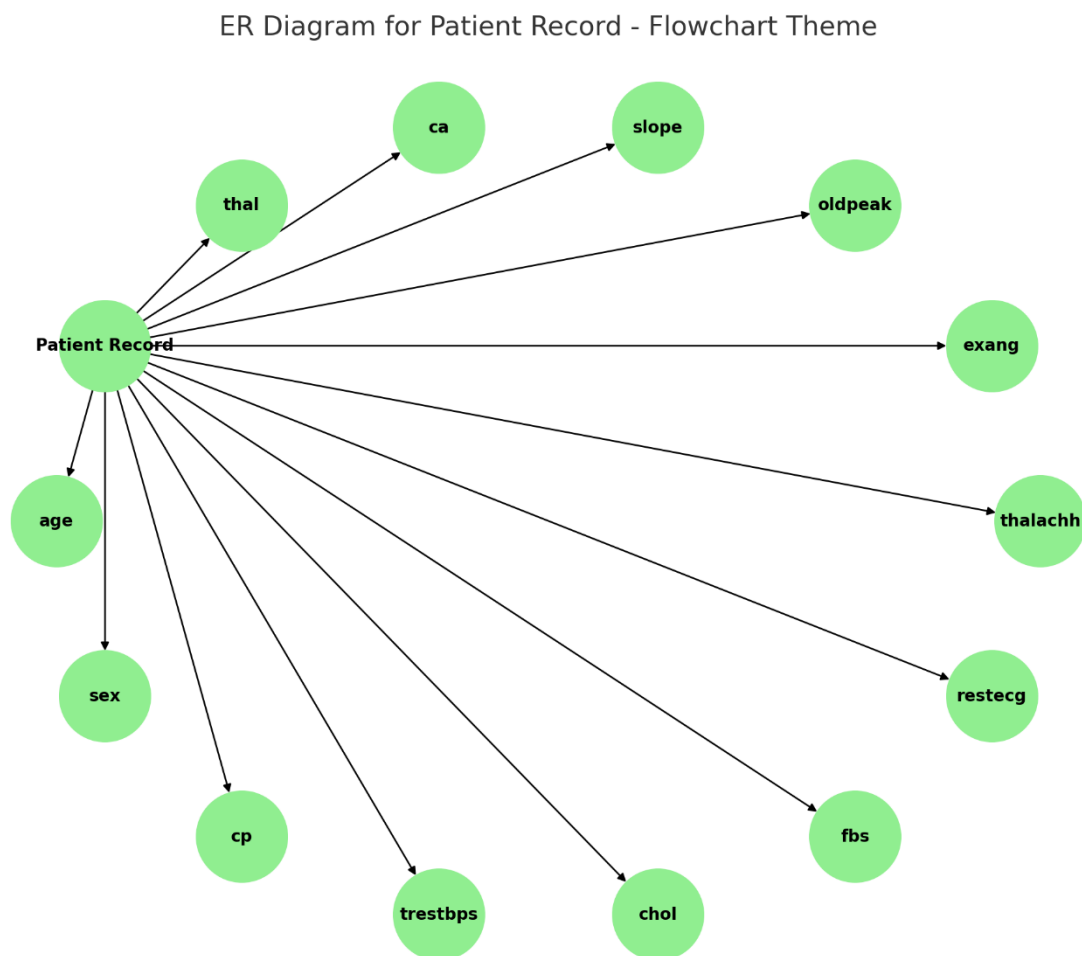
3. DATABASE STRUCTURE

3.4 Er Diagram

An Entity-Relationship (ER) diagram provides a visual representation of the data model for Predicting patient record and management system

"Patient Record" is the central entity.

All the listed columns such as **age**, **sex**, **cp**, **trestbps**, **chol**, and others are attributes connected to this central entity, representing various features of the patient record.



3.5 Normalization

Normalization is the process of organizing the database to reduce redundancy and dependency, ensuring efficient data storage and retrieval.

1. **1NF (First Normal Form)**: Ensures that all data is stored in tables with atomic values. Each field in the books table, like Title, Author, Genres, etc., holds a single value per row. In cases where books have multiple genres, genres could either be stored as a comma-separated string or normalized further by creating a separate **Genre** table.

2. **2NF (Second Normal Form):** Achieved by ensuring that each non-primary key attribute is fully dependent on the primary key. Since each book is uniquely identified by BookID or Title (if unique), there are no partial dependencies in the books table.
3. **3NF (Third Normal Form):** Achieved by ensuring that non-key attributes are not dependent on other non-key attributes. In this case, all book information like Description, Genres, Avg_Rating, etc., directly relates to the BookID and is independent of each other, satisfying 3NF.

4. PROGRAM CODE

4.1 Overview of Code Structure

The **Book Recommendation System** codebase is structured into several key files, each handling a specific function within the application. The main components include the backend server code, database connection scripts, models, data import scripts, and frontend files.

Below is a breakdown of each file and its role in the project:

1. **app.py**: This is the main application file that configures the Flask server and defines routes for serving book data.
2. **database.py**: Contains the logic for establishing a connection to the MySQL database.
3. **import_csv_to_db.py**: This script imports data from a CSV file into the database, making it accessible to the application.
4. **models.py**: Defines the Book model to represent books and organize book-related data.
5. **schema.sql**: A SQL script for creating the database schema, defining tables and fields necessary for storing book data.
6. **index.html**: The main frontend file, displaying the book list, search, and sort options to the user.

4.2 Code Walkthrough

Here is a detailed explanation of each file, with key code snippets and functionality descriptions.

4.2.1 login.py

```
import customtkinter as ctk
import mysql.connector
from mysql.connector import Error
import subprocess

# Connect to the MySQL database
def create_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="sivakumar#",
        database="heartattackpred"
```

)

app = ctk.CTk()

app.geometry("300x205")

Userlabel = ctk.CTkLabel(app, text="Username",

font=("Arial", 15))

Userlabel.place(x=25,y=50)

Userlabelsemi = ctk.CTkLabel(app, text=":",

font=("Arial", 15))

Userlabelsemi.place(x=100,y=50)

UserEntry = ctk.CTkEntry(app)

UserEntry.place(x=140,y=50)

Passlabel = ctk.CTkLabel(app, text="Password",

font=("Arial", 15))

Passlabel.place(x=25,y=90)

Passlabelsemi = ctk.CTkLabel(app, text=":",

font=("Arial", 15))

Passlabelsemi.place(x=100,y=90)

PassEntry = ctk.CTkEntry(app)

PassEntry.place(x=140,y=90)

def validate_login():

try:

```
username = UserEntry.get()
password = PassEntry.get()
# Create a connection to the database
db_connection = create_connection()
cursor = db_connection.cursor()

# SQL query to check if the user exists and the
password matches
cursor.execute("SELECT * FROM authentication
WHERE username = %s AND password = %s",
(username, password))
user = cursor.fetchone()

if user:
    print("Login successful!")
    app.destroy()
    command = r"python
C:\ProgrammingProjects\ML-Project\menu.py"
    subprocess.run(command)

else:
    print("Invalid username or password.")
    Label = ctk.CTkLabel(app,text="Invalid Login")
    Label.place(x=100,y=170)

except Error as e:
    print(f'Error: {e}')
```

```
LoginButton = ctk.CTkButton(app,text =  
"Login",command=validate_login)  
LoginButton.place(x = 85,y=140)  
  
app.mainloop()
```

Description: app.py serves as the main entry point for the application, handling HTTP requests, managing API endpoints, and rendering the frontend template.

4.2.2 menu.py

```
import customtkinter as ctk  
import subprocess  
  
app = ctk.CTk()  
app.geometry("300x205")  
  
def PredictNew():  
    app.destroy()  
    command = r"python  
C:\ProgrammingProjects\ML-  
Project\gui.py"  
    subprocess.run(command)  
  
PredictNewButton =  
ctk.CTkButton(app,text =  
"Predict New  
Record",command=PredictNew)
```

```
PredictNewButton.place(x =  
85,y=50)  
  
ViewTableButton =  
ctk.CTkButton(app,text = "View  
Predictions",)  
ViewTableButton.place(x =  
85,y=105)  
  
app.mainloop()
```

Description: database.py provides a function to connect to the MySQL database, which is used by the Flask app to retrieve and manage book data.

4.2.3 main.py

```
import customtkinter as ctk  
import joblib  
import pandas as pd  
import mysql.connector  
from mysql.connector import Error  
# Load the saved model  
model = joblib.load("heartAttackModel.pkl")  
print("Model loaded successfully!")  
  
app = ctk.CTk()  
app.geometry("700x720")  
  
#Patient ID  
PatientIDlabel = ctk.CTkLabel(app, text="PatientID", font=("Arial", 16))
```

```
PatientIDlabel.place(x=100,y=10)
```

```
PatientIDlabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
```

```
PatientIDlabelsemi.place(x=300,y=10)
```

```
PatientIDEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
```

```
PatientIDEntry.place(x=400,y=10)
```

```
#Age
```

```
Agelabel = ctk.CTkLabel(app, text="Age", font=("Arial", 16))
```

```
Agelabel.place(x=100,y=50)
```

```
Agelabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
```

```
Agelabelsemi.place(x=300,y=50)
```

```
AgeEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
```

```
AgeEntry.place(x=400,y=50)
```

```
#Gender
```

```
Genderlabel = ctk.CTkLabel(app, text="Gender", font=("Arial", 16))
```

```
Genderlabel.place(x=100,y=90)
```

```
Genderlabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
```

```
Genderlabelsemi.place(x=300,y=90)
```

```
GenderEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
```

```
GenderEntry.place(x=400,y=90)
```

```
#cp
```

```
CLabel = ctk.CTkLabel(app, text="Chest Pain Type", font=("Arial", 16))
CLabel.place(x=100,y=130)
```

```
CLabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
CLabelsemi.place(x=300,y=130)
```

```
CPEnt = ctk.CTkEntry(app, placeholder_text="Enter text here")
CPEnt.place(x=400,y=130)
```

```
#trestbps
TrestBPSLabel = ctk.CTkLabel(app, text="Resting BP", font=("Arial", 16))
TrestBPSLabel.place(x=100,y=170)
```

```
TrestBPSLabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
TrestBPSLabelsemi.place(x=300,y=170)
```

```
TrestBPSEnt = ctk.CTkEntry(app, placeholder_text="Enter text here")
TrestBPSEnt.place(x=400,y=170)
```

```
#chol
chollabel = ctk.CTkLabel(app, text="Serum Cholesterol", font=("Arial", 16))
chollabel.place(x=100,y=210)
```

```
chollabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
chollabelsemi.place(x=300,y=210)
```

```
cholEnt = ctk.CTkEntry(app, placeholder_text="Enter text here")
cholEnt.place(x=400,y=210)
```

#fbs

fbslabel = ctk.CTkLabel(app, text="Fasting Blood Sugar", font=("Arial", 16))

fbslabel.place(x=100,y=250)

fbslabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))

fbslabelsemi.place(x=300,y=250)

fbsEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")

fbsEntry.place(x=400,y=250)

#restecg

RestECGlabel = ctk.CTkLabel(app, text="Resting ECG", font=("Arial", 16))

RestECGlabel.place(x=100,y=290)

RestECGlabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))

RestECGlabelsemi.place(x=300,y=290)

RestECGEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")

RestECGEntry.place(x=400,y=290)

#thalachh

thalachhlabel = ctk.CTkLabel(app, text="Maximum heart rate", font=("Arial", 16))

thalachhlabel.place(x=100,y=330)

thalachhlabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))

thalachhlabelsemi.place(x=300,y=330)

thalachhEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")

thalachhEntry.place(x=400,y=330)

#exang

exanglabel = ctk.CTkLabel(app, text="Exercise-induced angina", font=("Arial", 16))

exanglabel.place(x=100,y=370)

exanglabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))

exanglabelsemi.place(x=300,y=370)

exangEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")

exangEntry.place(x=400,y=370)

#oldpeak

oldpeaklabel = ctk.CTkLabel(app, text="ST depression", font=("Arial", 16))

oldpeaklabel.place(x=100,y=410)

oldpeaklabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))

oldpeaklabelsemi.place(x=300,y=410)

oldpeakEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")

oldpeakEntry.place(x=400,y=410)

#slope

slopelabel = ctk.CTkLabel(app, text="Slope of the peak exercise\nST segment",
font=("Arial", 16))

slopelabel.place(x=100,y=450)

slopelabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 11))

slopelabelsemi.place(x=300,y=450)

```

slopeEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
slopeEntry.place(x=400,y=450)

#ca
calabel = ctk.CTkLabel(app, text="Number of major vessels\n(0-3) colored by fluoroscopy",
font=("Arial", 16))
calabel.place(x=100,y=510)

calabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 11))
calabelsemi.place(x=300,y=510)

caEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
caEntry.place(x=400,y=510)

#thal
thallabel = ctk.CTkLabel(app, text="Thalassemia type", font=("Arial", 16))
thallabel.place(x=100,y=550)

thallabelsemi = ctk.CTkLabel(app, text=":", font=("Arial", 16))
thallabelsemi.place(x=300,y=550)

thalEntry = ctk.CTkEntry(app, placeholder_text="Enter text here")
thalEntry.place(x=400,y=550)

Entries =
[AgeEntry,GenderEntry,CPEntry,TrestBPSEntry,cholEntry,fbsEntry,RestECGEntry,thalachhE
ntry,exangEntry,oldpeakEntry,slopeEntry,caEntry,thalEntry]

# Values = [[]]

Columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalachh', 'exang', 'oldpeak', 'slope',
'ca', 'thal']

```

```
Values = [["Umesh",67,0,2,152,277,0,1,172,0,0.0,2,1,2]]

#prediction
def predfunc():
    # for i in Entries:
    #     text = i.get()
    #     Values[0].append(text)
    X_Df = pd.DataFrame(Values, columns=Columns)
    print(X_Df)
    prediction = model.predict(X_Df)
    if prediction == [1]:
        Text = "High Risk"
    else:
        Text = "Low Risk"
    predLabel = ctk.CTkLabel(app, text=Text, font=("Arial", 16,),text_color="green")
    predLabel.place(x=280,y=650)
    print(prediction)
    # model = joblib.load("heartAttackModel.pkl")
    # model.predict()

def saveInDB():
    # Connect to the MySQL database
    db_connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="sivakumar#",
        database="heartattackpred"
    )
    cursor = db_connection.cursor()
    values=""
```

```

for i in Values[0]:
    if type(i) == str:
        values += ""

        txt = str(i)

        values+=txt

        values += ",'"

    else:
        txt = str(i)+","
        values+=txt

values = values[0:-1]

command = f"""\INSERT INTO heartattackpred.patientrecords(PatientName,age, sex, cp,
trestbps, chol, fbs, restecg, thalachh, exang, oldpeak, slope, ca, thal) Values({values})"""

print(command)

cursor.execute(command)

db_connection.commit()

cursor.close()

db_connection.close()

predButton = ctk.CTkButton(app, text="Predict", command=predfunc)
predButton.place(x=140,y=610)

SaveButton = ctk.CTkButton(app,text="Save in DB",command=saveInDB)
SaveButton.place(x=340,y=610)

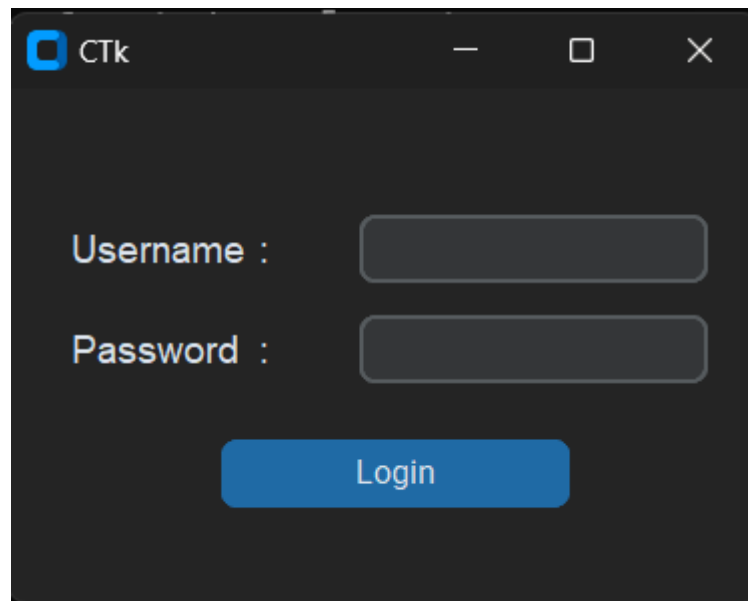
app.mainloop()

```

Description: import_csv_to_db.py is a utility script to import book data from a CSV file into the database, useful for initializing the database with data.

5.PROJECT SCREENSHOTS

LOGIN PAGE

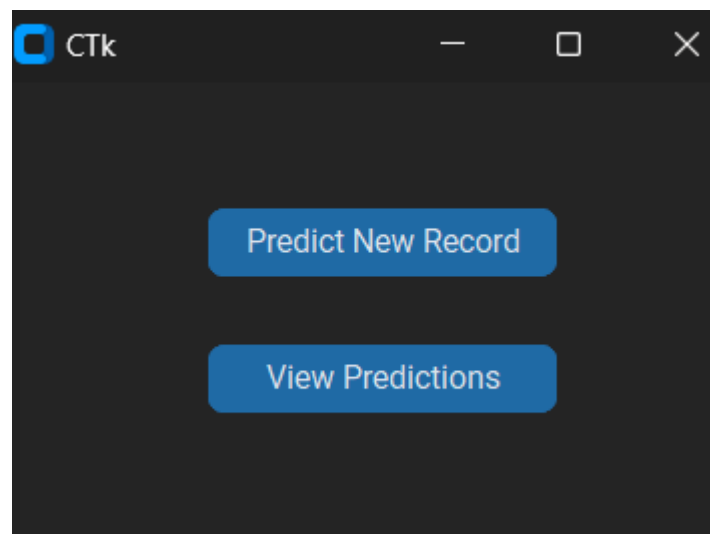


A screenshot of a login window titled 'CTk'. It features a dark background with two input fields: 'Username :' and 'Password :'. Below these fields is a blue 'Login' button.

AUTHENTICATION TABLE

	username	password
▶	admin	pass
•	NULL	NULL

MENU PAGE



A screenshot of a menu window titled 'CTk'. It features a dark background with two blue buttons: 'Predict New Record' and 'View Predictions'.

PREDICT NEW RECORD

CTk

PatientID	:	<input type="text" value="Enter text here"/>
Age	:	<input type="text" value="Enter text here"/>
Gender	:	<input type="text" value="Enter text here"/>
Chest Pain Type	:	<input type="text" value="Enter text here"/>
Resting BP	:	<input type="text" value="Enter text here"/>
Serum Cholesterol	:	<input type="text" value="Enter text here"/>
Fasting Blood Sugar	:	<input type="text" value="Enter text here"/>
Resting ECG	:	<input type="text" value="Enter text here"/>
Maximum heart rate	:	<input type="text" value="Enter text here"/>
Exercise-induced angina	:	<input type="text" value="Enter text here"/>
ST depression	:	<input type="text" value="Enter text here"/>
Slope of the peak exercise ST segment	:	<input type="text" value="Enter text here"/>
Number of major vessels (0-3) colored by fluoroscopy	:	<input type="text" value="Enter text here"/>
Thalassemia type	:	<input type="text" value="Enter text here"/>

PREDICTING

CTk

PatientID	:	<div>Enter text here</div>
Age	:	<div>Enter text here</div>
Gender	:	<div>Enter text here</div>
Chest Pain Type	:	<div>Enter text here</div>
Resting BP	:	<div>Enter text here</div>
Serum Cholesterol	:	<div>Enter text here</div>
Fasting Blood Sugar	:	<div>Enter text here</div>
Resting ECG	:	<div>Enter text here</div>
Maximum heart rate	:	<div>Enter text here</div>
Exercise-induced angina	:	<div>Enter text here</div>
ST depression	:	<div>Enter text here</div>
Slope of the peak exercise ST segment	:	<div>Enter text here</div>
Number of major vessels (0-3) colored by fluoroscopy	:	<div>Enter text here</div>
Thalassemia type	:	<div>Enter text here</div>

Predict

Save in DB

High Risk

SAVING IN DB

CTk

PatientID

:

Enter text here

Age

:

Enter text here

Gender

:

Enter text here

Chest Pain Type

:

Enter text here

Resting BP

:

Enter text here

Serum Cholesterol

:

Enter text here

Fasting Blood Sugar

:

Enter text here

Resting ECG

:

Enter text here

Maximum heart rate

:

Enter text here

Exercise-induced angina

:

Enter text here

ST depression

:

Enter text here

Slope of the peak exercise
ST segment

:

Enter text here

Number of major vessels
(0-3) colored by fluoroscopy

:

Enter text here

Thalassemia type

:

Enter text here

Predict

Save in DB

Saved in DB

PATIENT RECORD TABLE

	PatientID	PatientName	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	t
▶	2	Umesh	67	0	0000000002	000000152	0000000277	0000000000	0000000001	000000172	0000000000	0000000000	0000000002	0000000001	0
	3	Umesh	67	0	0000000002	000000152	0000000277	0000000000	0000000001	000000172	0000000000	0000000000	0000000002	0000000001	0
	4	Umesh	67	0	0000000002	000000152	0000000277	0000000000	0000000001	000000172	0000000000	0000000000	0000000002	0000000001	0

6. CONCLUSION

The **Patient Health Monitoring and Heart Attack Risk Prediction System** serves as a vital tool in modern healthcare by combining efficient data management with predictive analytics. The system enables healthcare professionals to seamlessly record, monitor, and analyze key health parameters like blood pressure (BP) and blood sugar levels. This data not only aids in tracking patient health trends but also forms the basis for predicting the risk of heart attacks using machine learning algorithms.

The integration of a **MySQL database server** ensures robust, scalable, and secure storage of sensitive patient information, making it accessible for retrieval and analysis. Leveraging the **scikit-learn library**, the system provides advanced predictive capabilities that enhance clinical decision-making by identifying patients at higher risk, allowing for timely preventive measures. The intuitive **Tkinter-based GUI** ensures that users, even those with minimal technical expertise, can navigate the application effortlessly.

This project demonstrates the potential of combining technology with healthcare to achieve better outcomes. By automating record-keeping, improving data accuracy, and providing actionable insights, the system reduces manual effort and enhances the efficiency of healthcare providers. Furthermore, it emphasizes the importance of early detection and preventive care in reducing cardiovascular risks.

In conclusion, this system not only fulfills the immediate objective of maintaining patient records but also lays a strong foundation for future enhancements, such as integrating IoT devices for real-time data capture or incorporating more complex predictive models for broader healthcare applications. It is a step forward in harnessing technology to create smarter, data-driven, and patient-focused healthcare systems.

7. REFERENCES

1. MySQL Documentation

- "MySQL 8.0 Reference Manual." Oracle Corporation, <https://dev.mysql.com/doc/>.

-
- Comprehensive documentation for understanding and implementing database management using MySQL.

2. Python Official Documentation

- "Python 3.11 Documentation." Python Software Foundation, <https://docs.python.org/3/>.
- Provides detailed insights into Python programming and libraries used in the project.

3.scikit-learn Documentation

- "scikit-learn: Machine Learning in Python." <https://scikit-learn.org/stable/>.
- Key resource for understanding machine learning algorithms and their application in heart attack risk prediction.

4.Tkinter GUI Development

- "Tkinter – Python Interface to Tcl/Tk." Python Software Foundation, <https://docs.python.org/3/library/tkinter.html>.
- Official guide for developing user-friendly interfaces using Tkinter.

5.Healthcare and Risk Prediction Studies

- Pal, R., and Banerjee, S. "Predictive Analytics in Healthcare: Applications and Challenges." *Journal of Medical Systems*, 2020.
- Discusses the role of machine learning in predicting health risks, including cardiovascular diseases.

6.World Health Organization (WHO)

- "Cardiovascular Diseases (CVDs)." World Health Organization, [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)).

-
- Resource for understanding cardiovascular health and the global significance of preventive measures.