Project Report

on

HEART DISEASE PREDICTION USING MACHINE LEARNING

by

ROSHAN V.C (RA2231241020040)

Submitted to the

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

BCA

Under the guidance of

Dr.E.SRIMATHI

Associate Professor Program Coordinator

Submitted in partial fulfillment of the requirement

for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Ramapuram,Chennai

2025

FACULTY OF SCIENCE & HUMANITIES

Ramapuram Campus

Department of Computer Science & Applications

(BCA)

BONAFIDE CERTIFICATE

Certified that this project report titled **"HEART DISEASE PREDICTION USING MACHINE LEARNING"** is the bonafide work of **ROSHAN V.C (RA2231241020040)** who carried out the **"UCA20D10J- PROJECT WORK"** done under my supervision.

Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of Internal Guide **Signature of Head of the Department**

Signature of Internal Examiner **Signature of External Examiner**

COMPLETION CERTIFICATE

Innovation is Our Business
STAIR Progressive Minds Private Limited
ISO Certified 9001:2015 & 27001:2022

+91 81481 64014
info@the-stair.com
www.the-stair.com

Acknowledgment of Project Completion – Machine Learning

Dear **V C Roshan**

Greetings from **STAIR Progressive Minds Private Limited**!

We are delighted to congratulate you on completing your final year project guidance with us, focusing on **Machine Learning**. It has been our pleasure to mentor and support you in gaining **practical industry exposure**, working with **state-of-the-art technologies** and **enhancing your analytical and technical skills** in a real-world setting.

Program Summary:

- **Specialization:** Machine Learning
- **Duration:** 3 months
- **Mode:** Hybrid
- **Commencement Date:** 14.12.2024
- **Mentorship:** Expert guidance from seasoned industry professionals
- **Certification:** Awarded upon successful completion

Your commitment and enthusiasm throughout the program have been truly commendable. We trust that your acquired skills and insights will contribute significantly to your future academic and professional success.

We wish you the very best in all your future endeavors!

Bhagirathi Sarath
Program Manager

No: 2, Unit F1, First Floor, Sriram Nagar, Navalur, Chennai - 600130, India.

COMPANY PROFILE

ABSTRACT

Heart Disease Prediction Using Machine Learning

Cardiovascular diseases (CVDs) are a leading cause of global mortality, underscoring the need for early detection and risk assessment. Cardio-AI is an AI-powered clinical decision support system designed to predict heart disease risk using a pre-trained Random Forest model with 89.5% accuracy. Developed with Python and Streamlit, it analyzes key health metrics—age, gender, BMI, total cholesterol, triglycerides, HDL, and LDL—to provide real-time probability scores and risk categorization (High/Low).

The system features a modern glassmorphism UI, offering seamless navigation across its core modules: Home, Risk Assessment, Patient Profile, and Model Information. Users can securely register, manage health profiles, track historical data, generate detailed PDF reports, and visualize risk trends with interactive Plotly charts. A MySQL database ensures secure user authentication and health data storage, managed by a custom DatabaseManager class for robust data persistence.

Cardio-AI is designed with a strong focus on privacy, usability, and clinical relevance, providing actionable insights for individuals and healthcare professionals to support preventive cardiovascular care. By leveraging machine learning and interactive analytics, it bridges the gap between technology and healthcare, empowering users with AI-driven early warning tools.

Future enhancements may include real-time wearable data integration, expanded predictive capabilities for other chronic conditions, and broader clinical validation to improve its accuracy and adoption in real-world medical settings.

SRM INSTITUTE OF SCIENCE AD TECHNOLOGY, RAMAPURAM

FACULTY OF SCIENCE AND HUMANITIES

Chennai , Ramapuram - 600 089

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

BCA

A Final Project Report

Acadamic Year:2024 – 2025 **Batch:2022-2025**

Project Title

HEART DISEASE PREDICTION USING MACHINE LEARNING

United Nations 17 Sustainable Development Goals:

Project Work towards 17 SDGs is mapped as follows:

Target 3.4: "By 2030, reduce by one third premature mortality from non-communicable diseases through prevention and treatment and promote mental health and well-being."

Target 4.7: "By 2030, ensure that all learners acquire the knowledge and skills needed to promote sustainable development, including, among others, through education for sustainable development and sustainable lifestyles, human rights, gender equality, promotion of a culture of peace and non-violence, global citizenship, and appreciation of cultural diversity and of culture's contribution to sustainable development."

Target 9.5: "Enhance scientific research, upgrade the technological capabilities of industrial sectors in all countries, in particular developing countries, including, by 2030, encouraging innovation and substantially increasing the number of research and development workers per 1 million people and public and private research and development spending."

Target 10.2: "By 2030, empower and promote the social, economic, and political inclusion of all, irrespective of age, sex, disability, race, ethnicity, origin, religion, or economic or other status."

ACKNOWLEDGEMENT

I extend my sincere gratitude to the Chancellor **Dr. T.R.PACHAMUTHU** and to Chairman **Dr. R. SHIVAKUMAR** of SRM Institute of Science and Technology, Ramapuram and Trichy campuses for providing me the opportunity to pursue the BCA degree at our esteemed University.

I express my sincere gratitude to **Dr.S.Thirumagan, DEAN(S&H),** SRM IST, Ramapuram for his support and encouragement for the successful completion of the project.

I am thankful to **Dr. J. Dhillipan M.Sc., MBA., M.Phil., Ph.D., HOD and Vice Principal-Admin (S&H)**, SRM IST, Ramapuram, for his support and encouragement for the successful completion of the project.

I record my sincere thanks to **Dr.S. Uma Shankari MCA., M.Phil., Ph.D., NET., SET., MSc., Associate Professor Program Coordinator, Department of Computer Science & Applications (BCA),** SRM IST, Ramapuram for her continuous support and keen interest to make this project a successful one.

I find no word to express profound gratitude to my guide **Dr.E.Srimathi**, Department of Computer Science & Applications (BCA), SRM IST Ramapuram.

I thank the almighty who has made this possible. Finally, I thank my beloved family member and friend for their motivation, encouragement and cooperation in all aspect which led me to the completion of this project.

ROSHAN.V.C

RA2231241020040

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

1. INTRODUCTION

1.1 PROJECT INTRODUCTION

Cardiovascular diseases (CVDs) are the leading cause of death globally, often progressing silently until severe complications arise. Early detection and risk assessment play a crucial role in preventing heart disease and improving long-term health outcomes. Cardio-AI is an AI-powered cardiovascular risk assessment system designed to predict heart disease risk in real time using machine learning techniques. It analyzes key health metrics—age, gender, BMI, total cholesterol, triglycerides, HDL, and LDL—to generate a probability score and categorize users into high-risk or low-risk groups.

Built using Python and Streamlit, Cardio-AI provides a modern, user-friendly glassmorphism UI for seamless navigation. At its core, the system integrates a pre-trained Random Forest model with 89.5% accuracy, ensuring reliable risk predictions. Users can register securely, store and manage their health records, track historical trends, and generate comprehensive PDF reports for medical consultation. A MySQL database ensures secure authentication and health data storage, maintaining patient privacy and data integrity.

The system features interactive Plotly charts to help users visualize their cardiovascular risk progression over time. Additionally, the Model Information section provides insights into the predictive mechanics, helping users understand the basis of AI-driven recommendations. Cardio-AI bridges the gap between technology and preventive healthcare, providing individuals with actionable insights to make informed lifestyle and medical decisions.

With a focus on privacy, usability, and clinical relevance, Cardio-AI empowers individuals to take control of their cardiovascular health. Future enhancements may include real-time wearable device integration, expansion to other chronic disease predictions, and broader clinical validation to improve real-world applicability. By leveraging AI-driven risk assessment, Cardio-AI aims to transform cardiovascular health management and support early intervention strategies.

CHAPTER-2

2. WORKING ENVIRONMENT

The working environment of Cardio-AI includes the hardware and software required for efficient development and execution. Functional requirements cover key features like cardiovascular risk assessment, user authentication, patient profile management, interactive visualizations, and PDF report generation. Non-functional requirements ensure performance, security, scalability, and data integrity, making the system reliable and user-friendly. Together, these elements enable Cardio-AI to deliver accurate, AI-driven cardiovascular risk predictions for better preventive healthcare.

2.1 HARDWARE REQUIREMENT

- **Processor**: Minimum **1.5 GHz**
- **RAM**: Minimum **4 GB**
- **Storage**: Minimum **20 GB free space**
- **Network**: Stable **internet connection** for accessing web-based features
- **Graphics**: Integrated GPU for **smooth UI rendering**

2.2 SOFTWARE REQUIREMENT

- **Operating System**: Windows, macOS, or Linux
- **Front-end Framework**: **Streamlit**
- **Back-end**: **Python** (Handles logic, database operations, and AI model execution)
- **Database**: **MySQL** (For user authentication and health data storage)
- **Libraries for Machine Learning & Data Processing**:
    - **Scikit-learn** (For Random Forest model implementation)
    - **Pandas & NumPy** (For data handling and processing)
    - **Matplotlib & Seaborn** (For static data visualization)
    - **Plotly** (For interactive visualizations of risk trends)
    - **ReportLab** (For PDF report generation)

2.3 SYSTEM SOFTWARE

- **Operating System:** Windows 11, macOS, or Linux
- **Integrated Development Environment (IDE):**
- Jupyter Notebook (For development and testing)
- VS Code (For code implementation)
- **Machine Learning Model:** Pre-trained Random Forest Model (89.5% accuracy)
- **Web Hosting:** Local deployment via Streamlit / Cloud-based deployment in future versions

CHAPTER-3

3.SYSTEM ANALYSIS

3.1 FEASIBILITY STUDY

The development of Cardio-AI is highly feasible due to advancements in machine learning, cloud-based deployment, and secure database management. The system utilizes a pre-trained Random Forest model (89.5% accuracy) to assess heart disease risk based on health parameters such as age, gender, BMI, cholesterol levels, and triglycerides. With Streamlit for UI, MySQL for secure data storage, and Python-based backend processing, the project ensures both technical feasibility and practical usability. The increasing adoption of AI in healthcare further supports the economic and operational feasibility of this project, making it a valuable preventive healthcare tool.

3.2 EXISTING SYSTEM

Existing cardiovascular risk assessment methods rely on manual evaluation by doctors, standardized scoring systems like Framingham Risk Score, and generalized medical guidelines. Many existing solutions lack personalized AI-driven insights, require frequent clinical visits, and do not offer real-time digital health tracking. Some risk prediction models are embedded in hospital information systems, making them inaccessible for personal use. The absence of interactive visualizations and historical trend tracking further limits user engagement and proactive health management.

3.3 DRAWBACKS OF EXISTING SYSTEM

Existing risk assessment methods have limited accuracy, relying on generalized scoring rather than AI-driven precision. They lack personalization, failing to adapt to individual health trends over time. With no real-time monitoring, users depend on periodic check-ups instead of continuous risk tracking. Additionally, fragmented patient data makes comprehensive risk assessment challenging.

3.4 PROPOSED SYSTEM

Cardio-AI introduces an AI-powered, web-based cardiovascular risk assessment tool that allows users to predict heart disease risk in real-time. The system leverages machine learning to analyze key health metrics and generate risk scores, classifying users as high or low risk. With secure MySQL-based data storage, users can track their historical risk trends, visualize data through interactive Plotly charts, and generate detailed PDF reports for medical consultations. Unlike traditional methods, Cardio-AI ensures secure, scalable, and user-friendly risk prediction accessible from anywhere.

3.5 BENEFITS OF PROPOSED SYSTEM

- AI-Driven Accuracy – Utilizes a pre-trained Random Forest model (89.5% accuracy) for precise predictions.
- Personalized Insights – Tracks individual health metrics over time for more tailored risk assessments.
- Interactive Visualizations – Uses Plotly charts for clear trend analysis and better decision-making.
- Data Security & Privacy – MySQL database ensures secure authentication and encrypted data storage.
- Seamless PDF Reports – Users can export detailed cardiovascular risk reports for doctor consultations.

3.6 SCOPE OF THE PROJECT

This project aims to develop a real-time cardiovascular risk prediction system that can be used by individuals for preventive healthcare. The system will enable users to assess, monitor, and track their heart disease risk over time, promoting early detection and lifestyle adjustments. Future expansions could include integration with wearable health devices, prediction models for other chronic diseases, and broader clinical validation to enhance its adoption in digital healthcare ecosystems.

3.7 BENEFITS TO THE SOCIETY

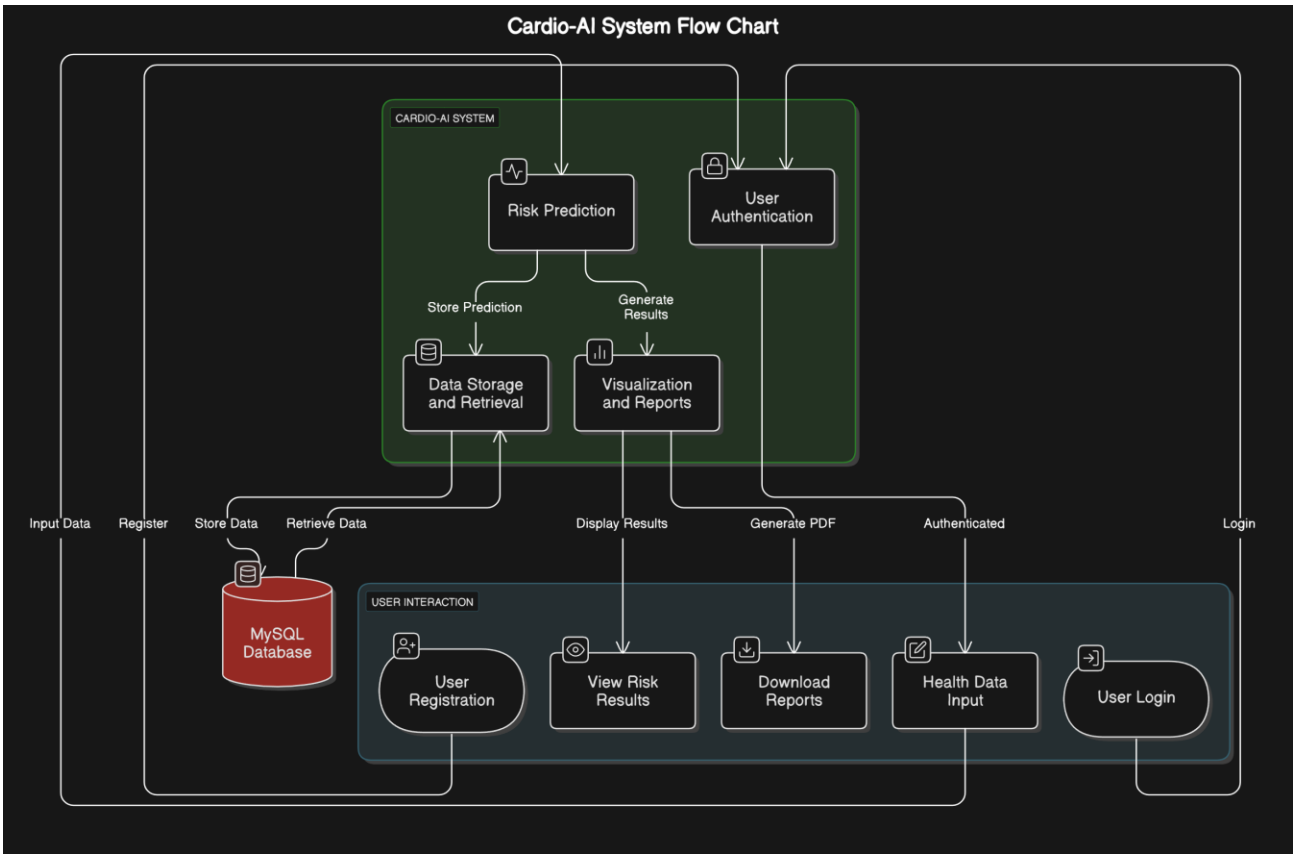
- Early Detection – Helps individuals take proactive measures to reduce heart disease risks.
- Improved Healthcare Accessibility – Offers a user-friendly, AI-powered tool for cardiovascular risk assessment.
- Better Health Outcomes – Encourages preventive care, reducing hospital visits and medical costs.
- Empowers Users – Provides actionable insights for lifestyle modifications and better heart health management.

CHAPTER-4

4. SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

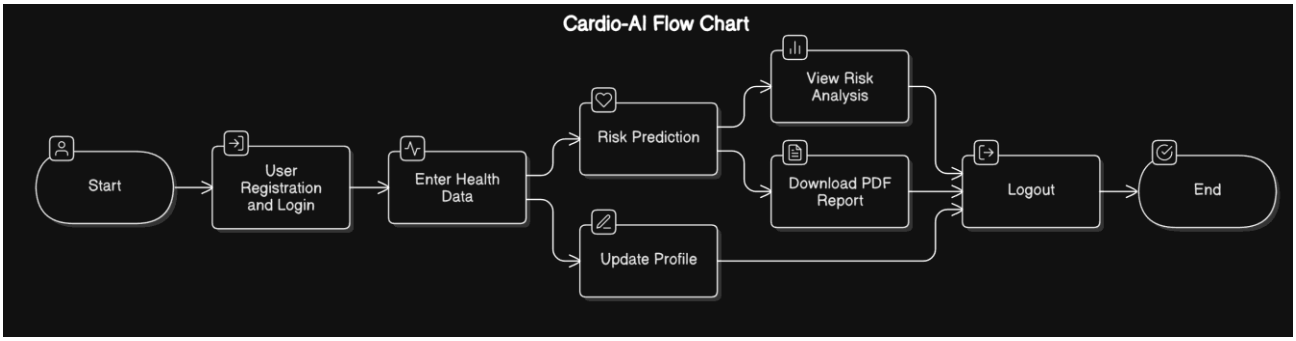
The Cardio-AI Data Flow Diagram (DFD) visualizes the movement of user health data through the system. Users input key metrics (age, BMI, cholesterol, etc.), which are processed by the Random Forest Model to predict heart disease risk (High/Low). This data is securely stored in a MySQL database for future tracking and analysis. Users can view interactive visualizations, receive personalized health tips, and download PDF reports. The DFD provides a high-level overview of these interactions, ensuring a structured and efficient data flow within Cardio-AI.



4.1.1: Representation of data flow diagram

4.2 USE CASE DIAGRAM

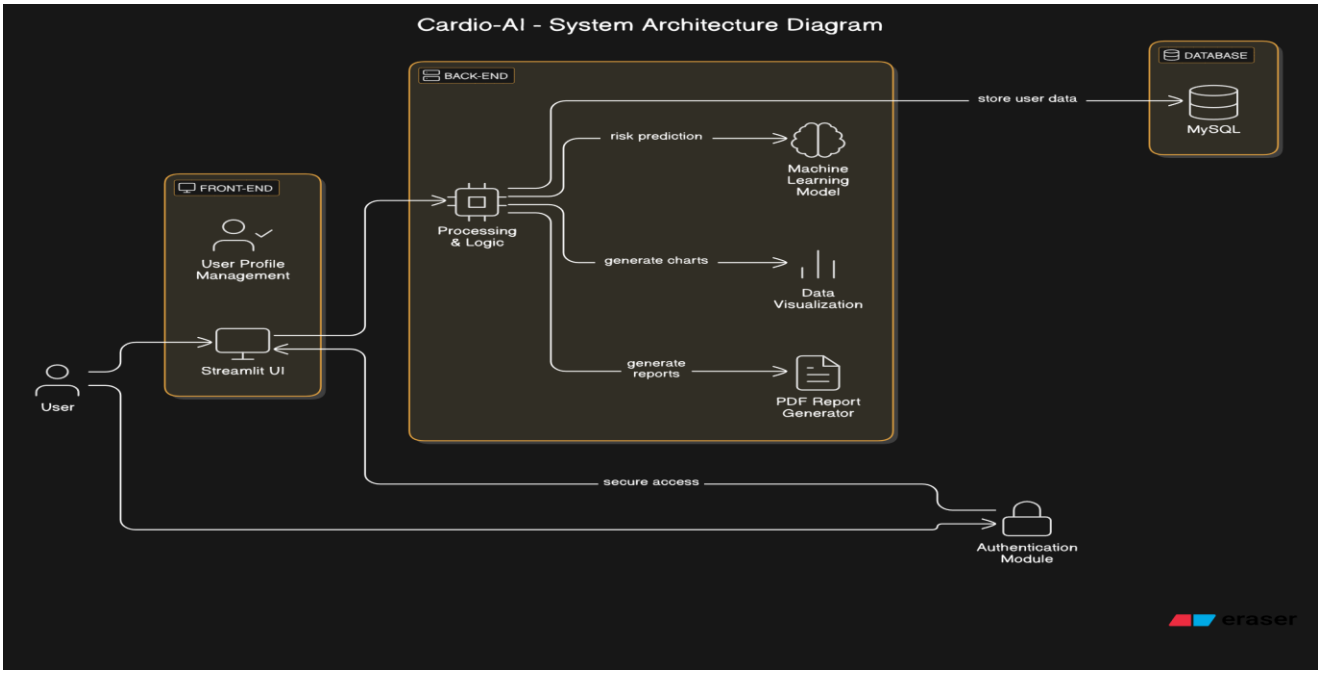
The Cardio-AI Use Case Diagram illustrates how the Patient interacts with the system. It begins with User Registration & Login, where the user creates an account and logs in securely. After logging in, the user enters their health data, including metrics like age, BMI, cholesterol levels, and blood pressure, through the Enter Health Data use case. This data is processed to generate a Risk Prediction, which assesses the likelihood of cardiovascular disease. Based on the prediction, users can View Risk Analysis, which provides interactive visualizations and trends of their health metrics. Users can also Download PDF Report, offering a detailed assessment of their risk. The Update Profile feature allows users to edit and update their health information for accurate tracking, while the Logout use case ensures secure system exit. The diagram demonstrates how each use case is interconnected, with Risk Prediction dependent on the health data entered, and View Risk Analysis and Download PDF Report relying on the prediction results. This provides a clear understanding of the user flow and system functionality.



4.2.1: Representation of use case diagram

4.3 ARCHITECTURE DIAGRAM

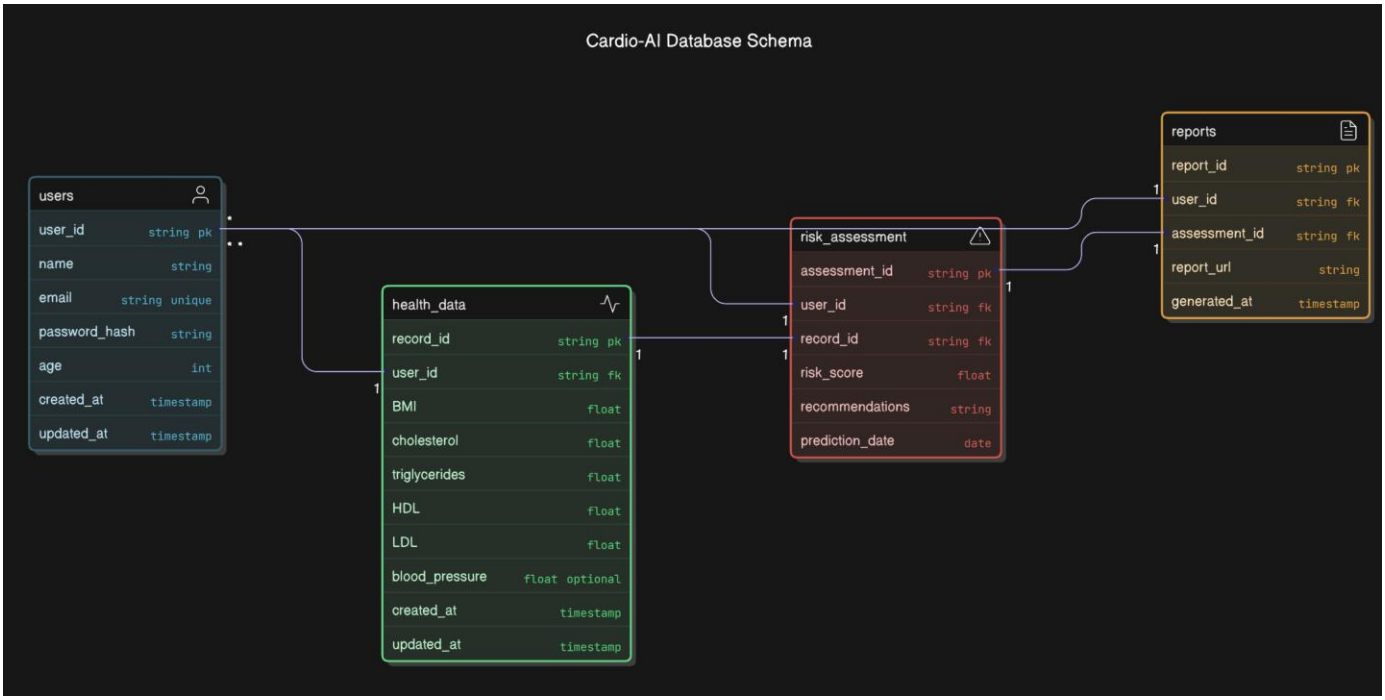
The architecture diagram of Cardio-AI illustrates the system's components and their interactions. The Front-End (Streamlit UI) allows users to register, log in, and input health data, which is securely stored in a MySQL database. The Machine Learning Model (Random Forest) processes this data to predict cardiovascular risk, providing real-time results. The Back-End (Python & Flask API) handles user authentication, data processing, and communication between the front-end and the database. Users can view risk visualizations, download PDF reports, and access educational resources. The system ensures security through encryption and regular model updates to enhance prediction accuracy.



4.3.1: Representation of architecture diagram

4.4 DATABASE DIAGRAM

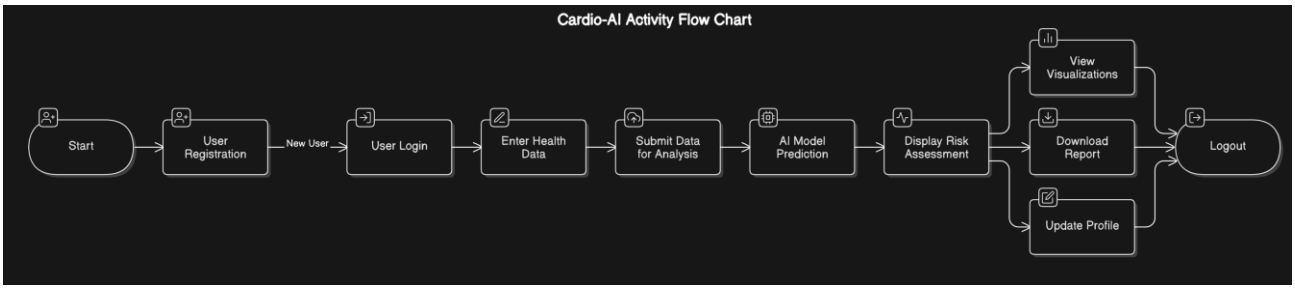
The Cardio-AI Database Schema Diagram illustrates the structured storage of user health data and risk assessments. The Users table stores patient information, linking to the Health Data table, which records key medical metrics like BMI, cholesterol, and triglycerides. The Risk Assessment table processes this data to generate a risk score and recommendations, while the Reports table stores generated PDF reports for users. Relationships ensure data integrity, with one-to-many connections between users and their health records, assessments, and reports. This schema enables efficient data retrieval, secure storage, and seamless heart disease risk analysis.



4.4.1: Representation of database diagram

4.5 ACTIVITY DIAGRAM

The Activity Diagram for Cardio-AI illustrates the step-by-step flow of user interactions within the system. It begins with User Registration & Login, allowing users to create an account or log in securely. Once authenticated, they Enter Health Data, which is then submitted for analysis by the AI model. The Random Forest model processes the data and provides a risk assessment (High/Low). Users can then view visualizations of their health trends, download a PDF report, or update their profile for future assessments. The process ends when the user logs out. This structured workflow ensures a seamless and interactive experience for cardiovascular risk assessment.



4.4.2: Representation of activity diagram

CHAPTER-5

5. PROJECT DESCRIPTION

5.1 OBJECTIVE

- Early detection of heart disease using machine learning.
- Providing real-time cardiovascular risk assessment.
- Assisting patients in understanding their heart health.
- Offering personalized health insights based on key health metrics.
- Generating predictive reports for further medical analysis.

5.2 MODULE DESCRIPTION

Data collection:

Heart disease dataset sourced from kaggle is used in CSV format.

Data Preprocessing:

The preprocessing step ensures the dataset is clean and ready for modeling. Missing values are handled efficiently by using df.dropna() to remove any incomplete entries. Categorical attributes like Gender are converted into numerical values through Label Encoding, facilitating their use in machine learning models. Health metrics such as Age, BMI, and Cholesterol are normalized to standardize the data and improve the model's performance.

Feature selection:

Crucial health indicators are selected for the model to focus on relevant attributes for accurate predictions. Key features such as Age, Gender, BMI, Cholesterol, Triglycerides, HDL, and LDL are chosen as they play a significant role in predicting heart disease risk. These features help to streamline the model, ensuring it is trained with the most impactful health data.

Building and Traning Model:

The project uses a pre-trained Random Forest model, achieving an accuracy of 89.5%. The dataset is split into training (x_train, y_train) and testing sets (x_test, y_test) to evaluate the model's performance. The model is then trained using the model.fit(x_train, y_train) function, allowing it to learn from the data and make predictions on new input.

Visualization:

Visualization is an essential part of the system, providing intuitive, interactive charts and graphs. Plotly and Matplotlib libraries are used to create visual representations of risk factors, trends, and predictions. These visual tools enable users to understand complex health data and gain insights into cardiovascular risk over time.

Algorithm Comparison:

To ensure the highest level of prediction accuracy, different machine learning algorithms are compared. By evaluating models based on performance metrics like accuracy and precision, the best-performing algorithm is selected, optimizing the system for accurate cardiovascular risk assessment.

5.3 IMPLEMENTATION

The system is built using Python and Streamlit, which provide a responsive, user-friendly web interface. It integrates securely with a MySQL database for user authentication and management of health records. The system dynamically displays machine learning predictions alongside interactive visual tools, creating a seamless experience for users.

Literature Survey:

1. **Predicting Heart Disease Using Machine Learning:**
   In this study, researchers applied machine learning techniques, such as Random Forest and Logistic Regression, to predict the likelihood of heart disease based on various health attributes like age, cholesterol levels, and BMI. The results showed that machine learning models can effectively identify high-risk individuals and assist doctors in early diagnosis, thus improving health outcomes.

2. **A Heart Disease Prediction Model Using SVM and Random Forest:**
   The authors of this paper used Support Vector Machines (SVM) and Random Forest algorithms to predict heart disease. Their system used a set of features, including patient demographics, blood pressure, cholesterol levels, and exercise habits. They demonstrated the efficacy of Random Forest models in predicting heart disease with high accuracy, similar to your approach.

3. **Improved Cardiovascular Risk Prediction Using Deep Learning:**
   A study by Lee et al. explored the use of deep learning algorithms, such as artificial neural networks (ANNs), to predict cardiovascular risk. They found that deep learning models could provide more nuanced insights into cardiovascular health compared to traditional machine learning models, highlighting the potential for future improvements in your risk prediction system.

4. **Predicting Cardiovascular Diseases with Decision Trees and Ensemble Methods:**
   This research applied Decision Trees and ensemble methods like Random Forest to predict cardiovascular diseases. By selecting important features, such as cholesterol, age, and family history, the authors were able to create models that provided reliable predictions, emphasizing the importance of feature selection, a key part of your system's development.

5. **Heart Disease Prediction using Naive Bayes and KNN Algorithms:**
   In a study by Kaur et al., Naive Bayes and K-Nearest Neighbors (KNN) were used to predict heart disease. The research showed that although Naive Bayes was efficient for data with multiple attributes, KNN performed better when predicting disease risk for individuals with more complex profiles, similar to your future model comparison approach.

6. **Personalized Health Risk Assessment Using Random Forest and Clustering Techniques:**
   This research focused on clustering techniques to categorize patients into different cardiovascular risk groups. The study used Random Forest as a predictive model to evaluate patient risk and cluster them accordingly. Their approach aligns with your intention to categorize risk levels using machine learning models.

7. **A Predictive Model for Cardiovascular Risk: Exploring Ensemble Learning Approaches:**

   Saeed et al. proposed a hybrid ensemble learning method combining multiple algorithms for better prediction of heart disease risk. By using both supervised and unsupervised techniques, they achieved more accurate results. This approach could offer future directions for enhancing your system's accuracy by integrating diverse learning models.

8. **Cardiovascular Disease Prediction Using Feature Selection and Machine Learning:**

   In their study, Nguyen and colleagues explored various machine learning techniques combined with feature selection methods for heart disease prediction. Their research emphasized the significance of selecting the right features to improve model performance. This aligns with your feature selection step to identify the most relevant health metrics.

9. **Using Machine Learning for Predicting Coronary Artery Disease:**

   An approach by Smith et al. used machine learning algorithms to predict the likelihood of coronary artery disease based on a variety of factors, including heart rate, cholesterol levels, and patient history. This research is highly relevant to your project's focus on predicting heart disease and suggests the potential for incorporating additional predictive features.

10. **A Comprehensive Model for Early Detection of Cardiovascular Diseases:**

    This study focused on the use of machine learning to create an early detection system for cardiovascular diseases. They employed techniques such as decision trees, Random Forest, and logistic regression. Their results showed that combining different algorithms enhanced prediction accuracy, which could be a useful insight for future improvements in your project's algorithm comparison phase.

5.4 USER AUTHENTICATION:

User authentication ensures that sensitive health data is protected. A secure login system is implemented using a username-password mechanism. This system guarantees encrypted storage of user credentials, adhering to best practices in security to safeguard patient privacy and data integrity.

5.5 DATA STORAGE & SECURITY:

All patient health data is stored securely in a MySQL database. The system ensures encryption of sensitive metrics to protect user privacy and safeguard health information. Regular backups are implemented to prevent data loss, and the system complies with healthcare regulations and privacy laws to maintain a secure data environment.

5.6 RISK PREDICTION & VISUALIZATION:

The heart disease prediction model analyzes key health parameters, such as Age, BMI, and Cholesterol, to assess cardiovascular risk. Personalized health insights and recommendations are provided based on the user's data. Results are visualized in an interactive format, including charts and graphs, which provide a clear picture of the user's cardiovascular health over time.

5.7 PDF REPORT GENERATION:

The system generates detailed health reports in PDF format, summarizing the user's cardiovascular risk profile, key risk factors, and preventive measures. These reports include interactive graphs and comparative analysis for easy review by healthcare professionals. Users are able to download and share these reports for further consultation or medical evaluation.

CHAPTER-6

6. SYSTEM TESTING

6.1 TESTING DEFINITION

System testing involves validating and verifying the functionality, security, and performance of the Cardio-AI system. It ensures that the heart disease prediction model and all associated features work as expected under real-world conditions. The primary goal is to identify and resolve any defects or performance bottlenecks in the system before deployment. This includes testing the user authentication process, data security measures, model predictions, and interactive visualizations.

6.2 TESTING OBJECTIVE

The main objectives of system testing for Cardio-AI are to ensure the heart disease prediction model delivers accurate and reliable results, confirm all features (authentication, data entry, prediction, visualization, and report generation) function correctly, validate the secure handling of sensitive health data in compliance with privacy regulations, ensure the user interface is intuitive and provides an optimal experience, and test the system's responsiveness and performance under varying conditions, ensuring it can handle multiple users simultaneously without issues.

6.3 TYPES OF TESTING

To ensure the Cardio-AI system works optimally, the following types of testing will be employed:

- **UNIT TESTING**

This involves testing individual components of the system, such as the heart disease prediction model, the database interaction functions, and the user authentication process, to ensure that each part works correctly in isolation before integration. This step ensures that all components are error-free and perform their tasks as expected when isolated.

➢ INTEGRATION TESTING

After the components have been individually verified, integration testing ensures that they function seamlessly when combined. This involves testing the communication and data flow between the machine learning model, user interface, and database to ensure that the entire system works harmoniously. For example, verifying that the model accurately processes inputs from the user interface and stores results in the database without data loss or errors.

➢ FUNCTIONAL TESTING

This focuses on ensuring that the core functionalities of the system, such as heart disease risk prediction, the accuracy of output predictions based on user inputs (like age, BMI, cholesterol levels), and the generation of detailed reports, are working as intended. Functional testing also includes verifying the correct generation and downloading of health reports, as well as checking if all features are working according to requirements.

- **Functional testing is centered on the following items:**

  - **Valid Input**: Ensure accurate heart disease risk prediction from valid health data (e.g., age, BMI, cholesterol levels).
  - **Invalid Input**: Reject invalid or incomplete inputs with error messages guiding users to correct them.
  - **Functions**: Test key features like user authentication, data entry, risk prediction, visualization, and PDF report generation.
  - **Output**: Verify that predicted risk levels, recommendations, and reports match expected results.
  - **Systems/Procedures**: Ensure smooth integration of the machine learning model and MySQL database for error-free operation.

SYSTEM TESTING

System Testing for Cardio-AI ensures the integrated system meets functional requirements, testing the web interface, machine learning model, database, and user features for smooth integration and accurate results. For example, it verifies that health data input leads to correct predictions and report generation.

WHITE BOX TESTING

White Box Testing focuses on the internal workings, such as validating the accuracy of the Random Forest model, data flow from input to prediction, and database interactions, ensuring the model processes data correctly and optimally.

BLOCK BOX TESTING

Black Box Testing validates the system from the user's perspective, ensuring correct risk prediction and report generation based on valid inputs (age, BMI, cholesterol) without needing to understand the internal processes.

6.4 TEST CASES

- **Unit Testing**: Individual components like the prediction model and user authentication will be tested to ensure they perform as expected in isolation.
- **Integration Testing**: Testing will focus on verifying that various components such as the machine learning model, user interface, and database work together without issues. For example, testing that the model generates predictions correctly after receiving data from the user interface and that the results are stored in the database without error.
- **User Acceptance Testing (UAT)**: This critical phase will involve end-users (patients) testing the system to ensure it meets their expectations and functional requirements. UAT will ensure that the system is intuitive, easy to use, and provides accurate and useful health insights based on user input.

CHAPTER-7

7. CONCLUSION

7.1 SUMMARY

The Cardio-AI system provides an innovative approach to cardiovascular health monitoring and early detection of heart disease. By leveraging machine learning, specifically a pre-trained Random Forest model with 89.5% accuracy, it analyzes key health indicators such as age, BMI, cholesterol levels, triglycerides, and others to assess the likelihood of heart disease. Built using Python and Streamlit, the platform offers a modern glassmorphism-styled user interface that allows patients to securely input their health data and receive real-time risk predictions. The system ensures robust data security with encrypted storage, regular backups, and adherence to healthcare privacy regulations, protecting sensitive patient information. Additionally, it includes dynamic visualizations powered by Plotly to present health trends and predictions in an easy-to-understand format. PDF report generation allows users to download detailed health reports, which can be shared with healthcare professionals for further consultation. Overall, Cardio-AI is a comprehensive, user-friendly, and secure solution designed to empower individuals with actionable insights into their heart health.

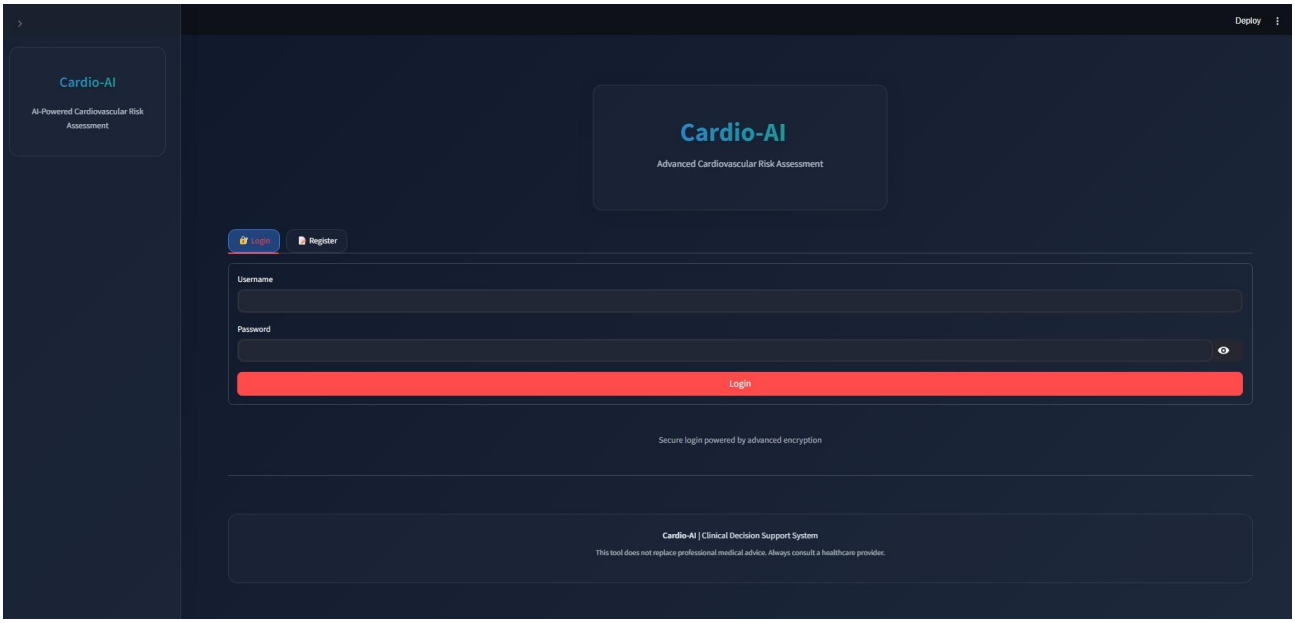
7.2 FUTURE ENHANCEMENTS

Future enhancements for Cardio-AI will focus on incorporating additional health metrics such as blood pressure, smoking status, and family history, which will provide a more comprehensive cardiovascular risk assessment. Integration with real-time data from wearable devices will allow continuous monitoring of users' health, enabling dynamic updates to risk predictions. A user feedback mechanism will be implemented to gather insights for improving the accuracy and usability of the system, while customizable report generation and interactive health tips will offer personalized recommendations. Additionally, the system will undergo broader clinical validation to ensure its applicability across diverse populations. Plans to develop a mobile app will make the platform more accessible, and enhanced data visualization tools will help users track their health trends over time, further empowering them in managing their cardiovascular health effectively.

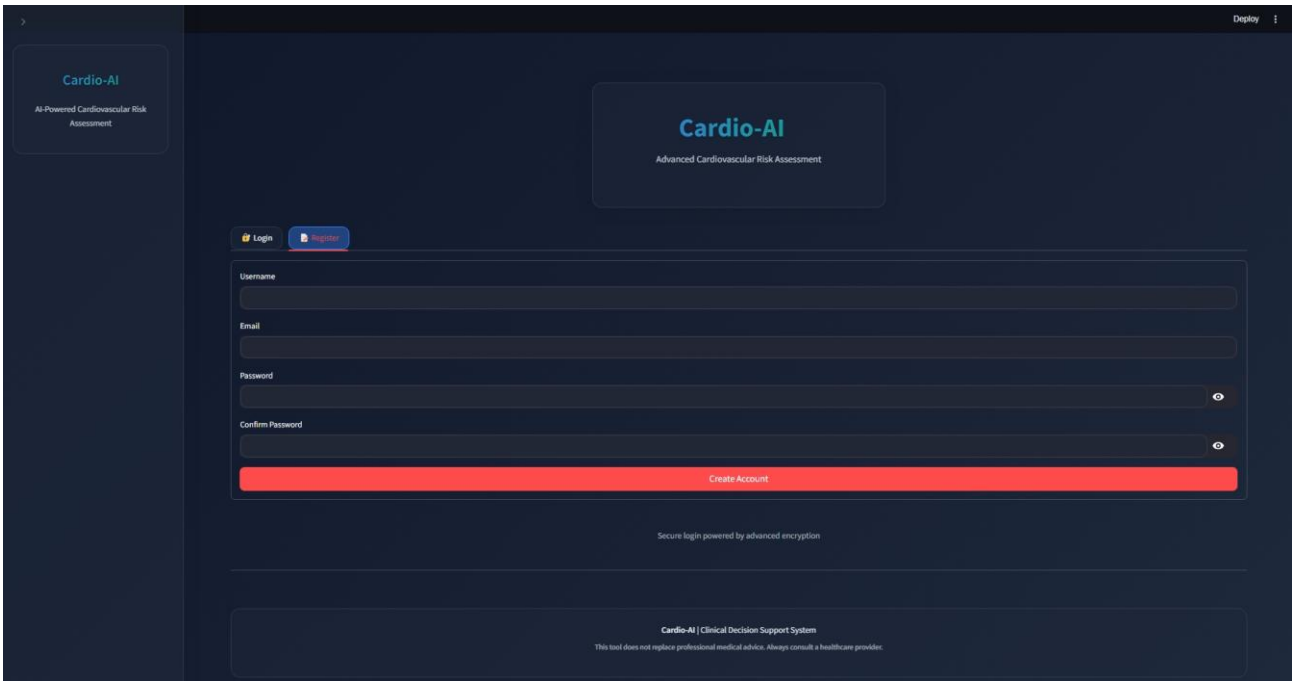
CHAPTER-8

8. APPENDIX

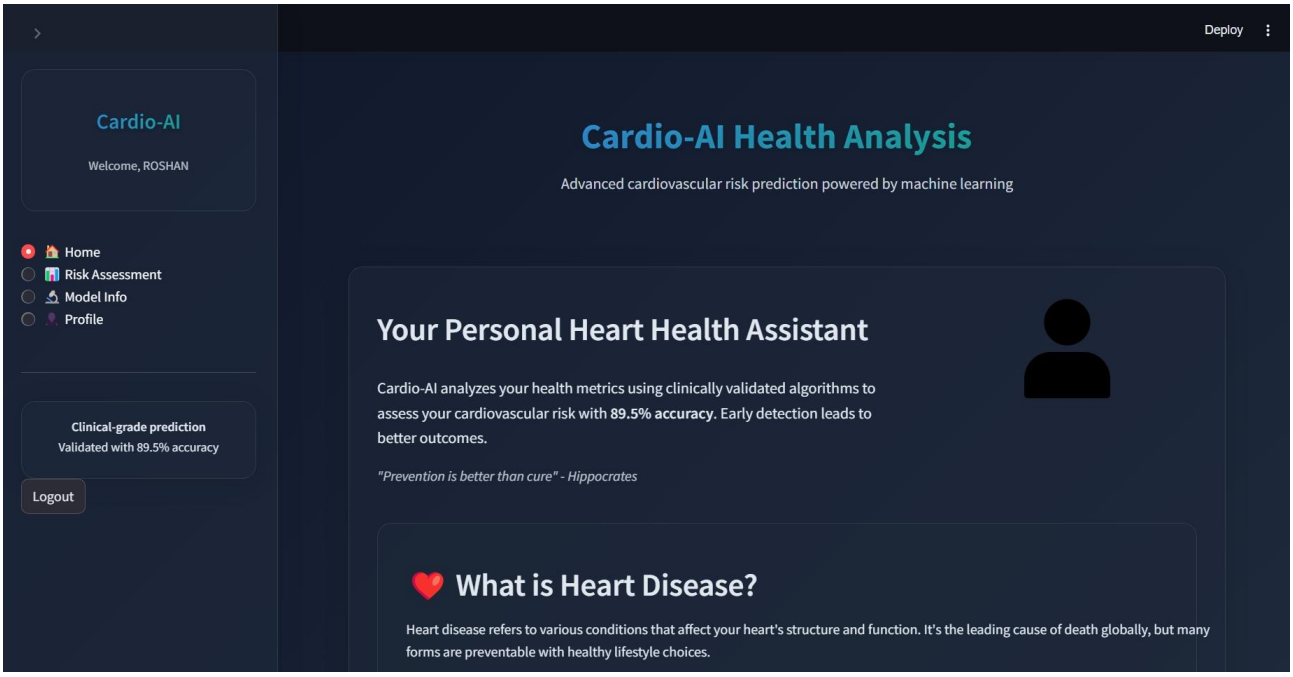
8.1 SCREENSHOTS



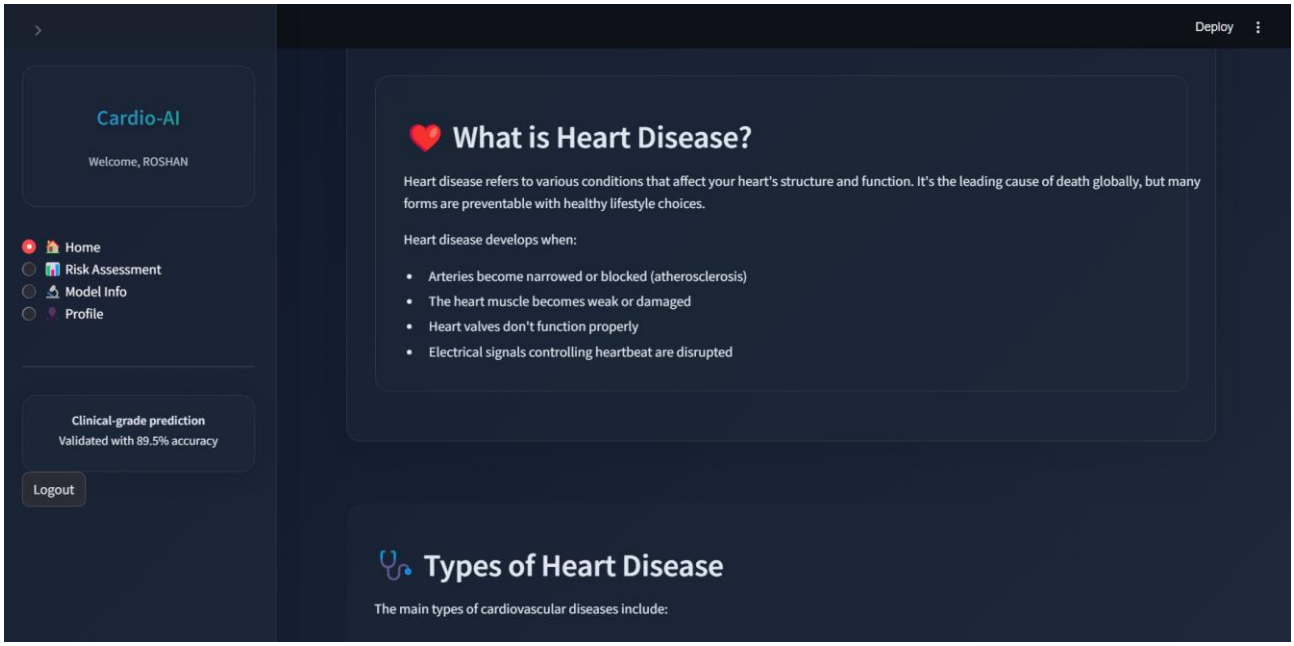
Implementation of login page in website



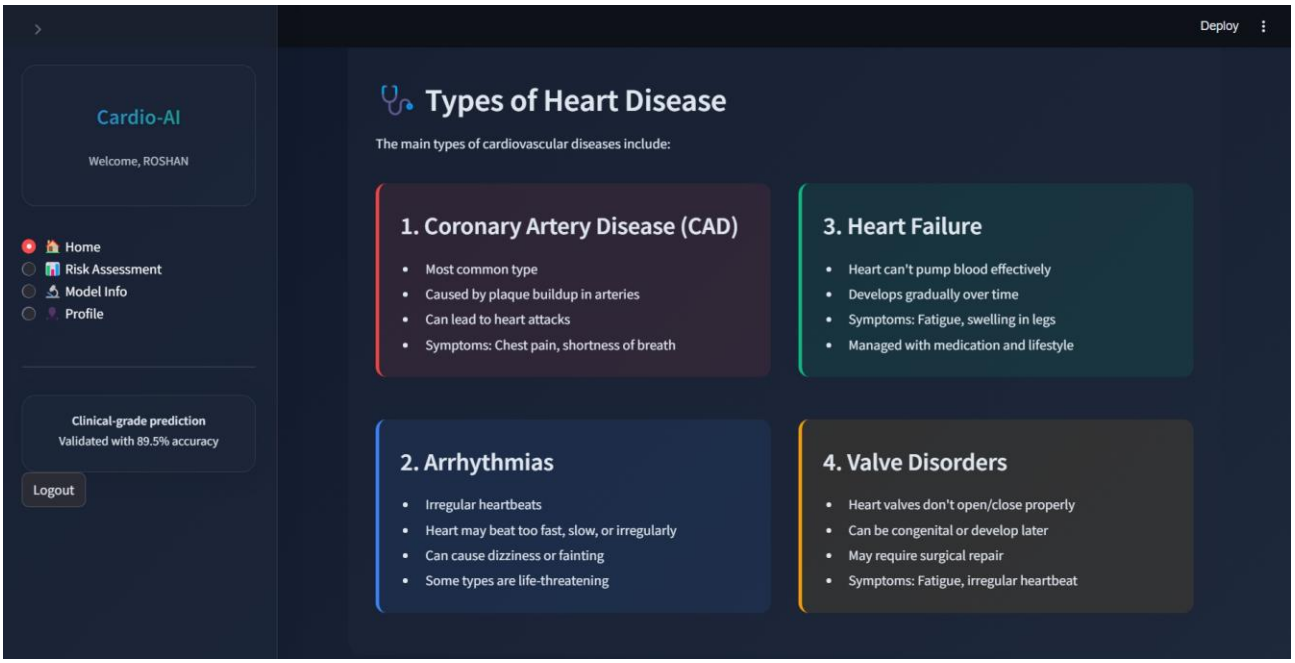
Implementation of registration page in website



Implementation of home page in website

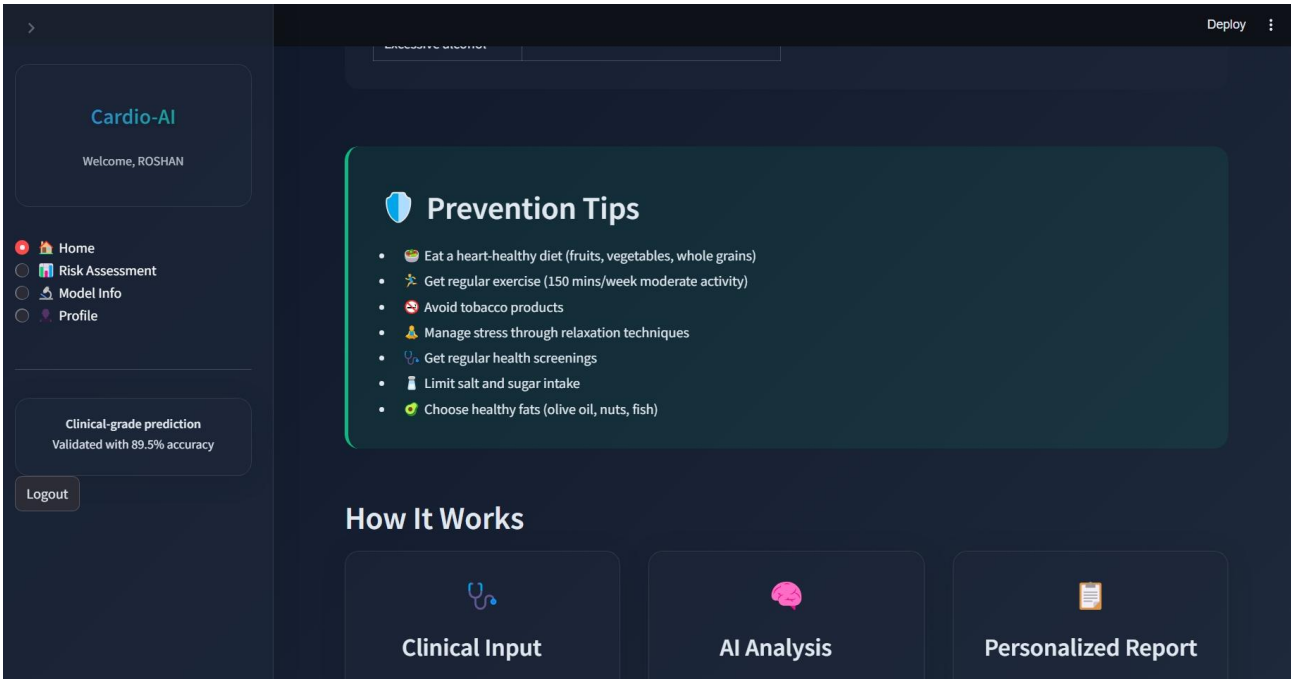


Implementation of risk Home page in website

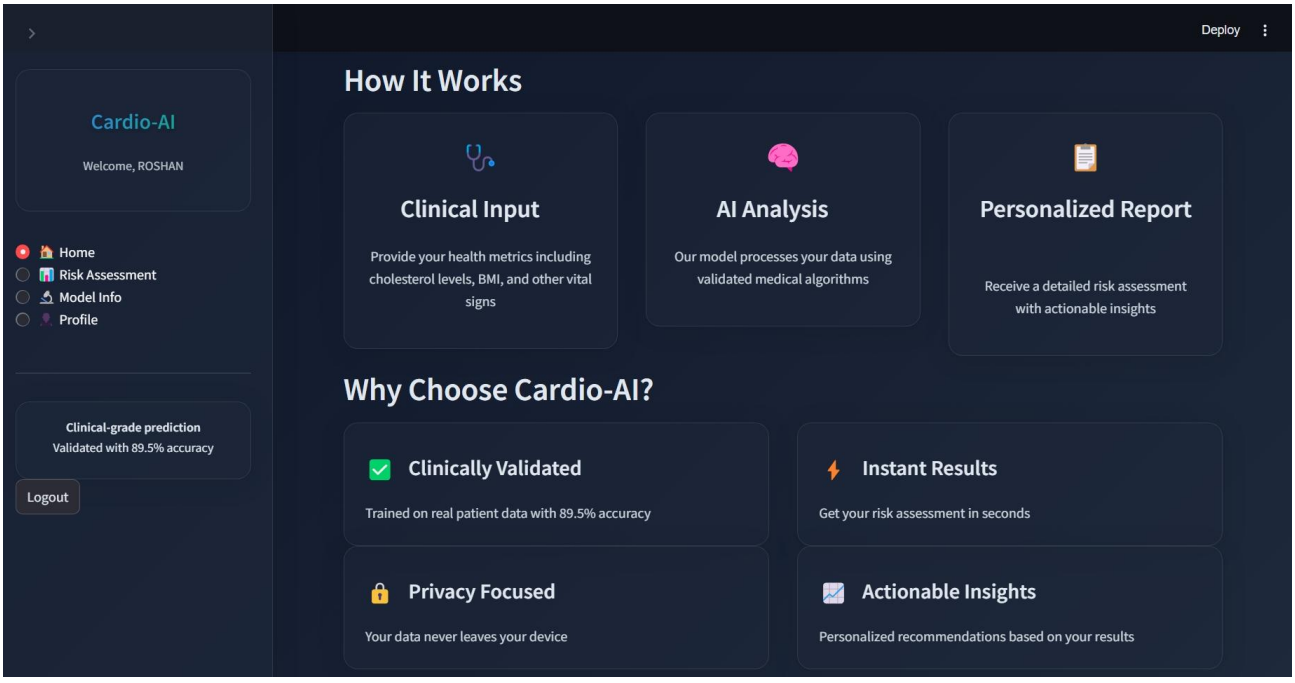


Implementation of home page in website

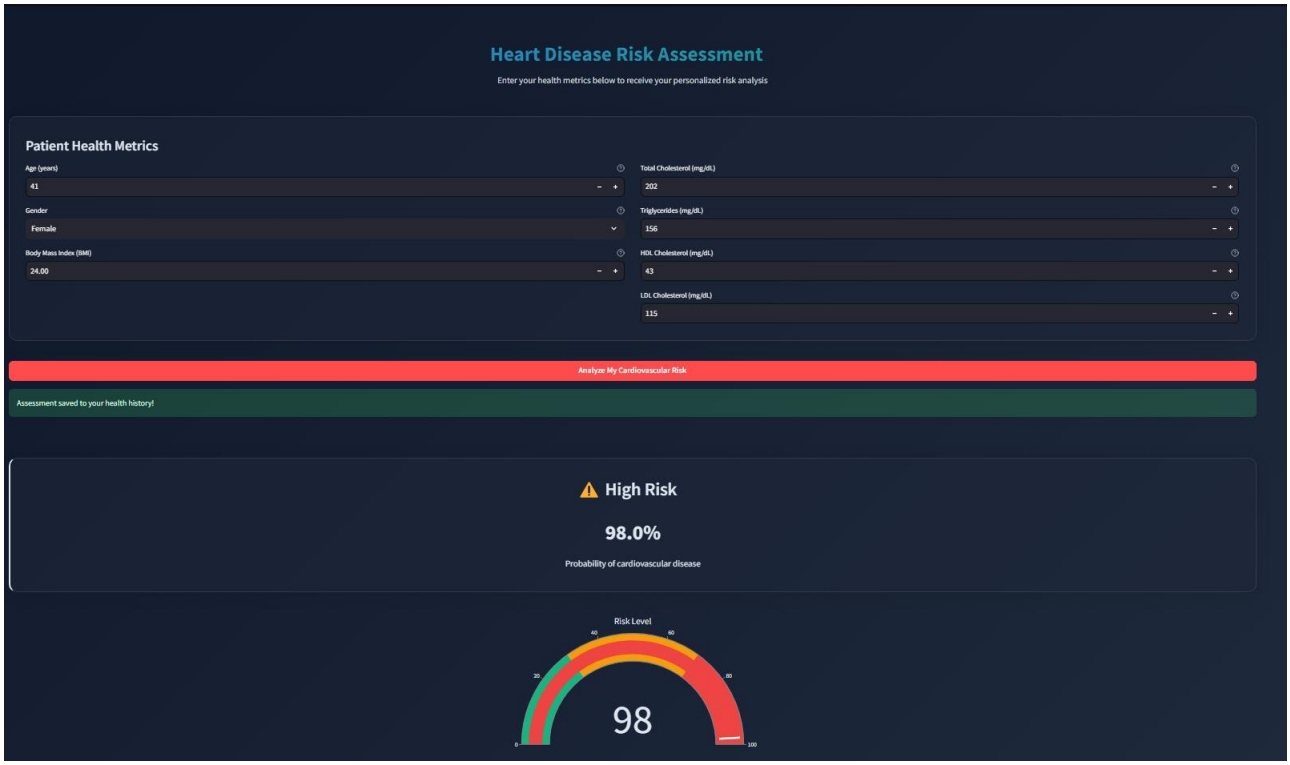
Implementation of risk home page in websiv



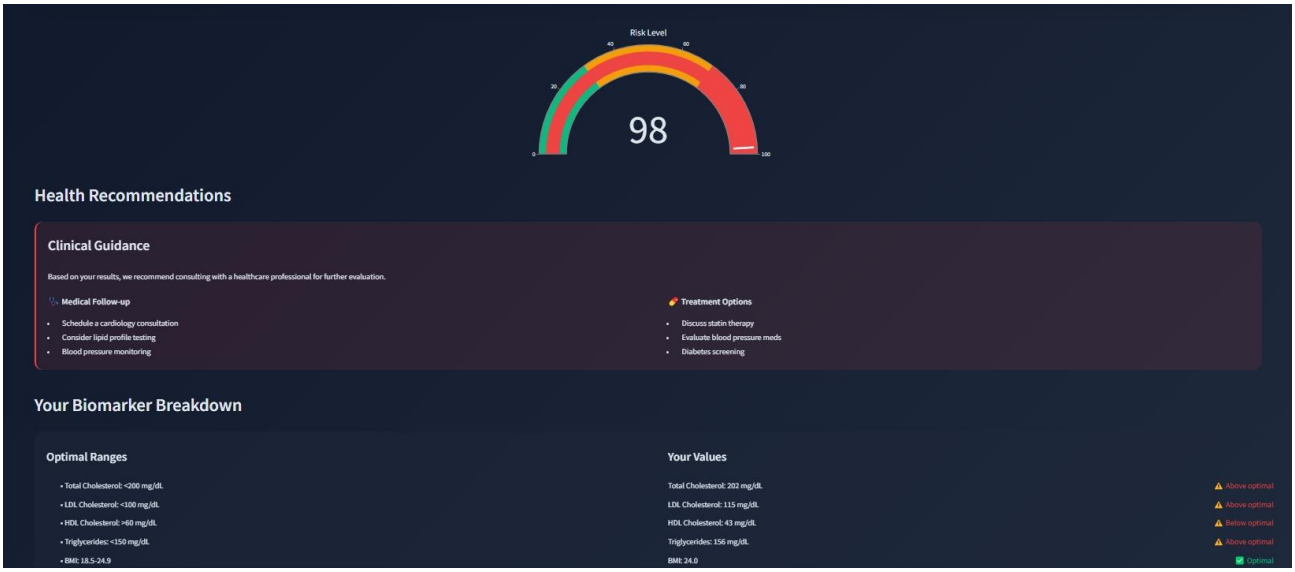
Implementation of home page in website



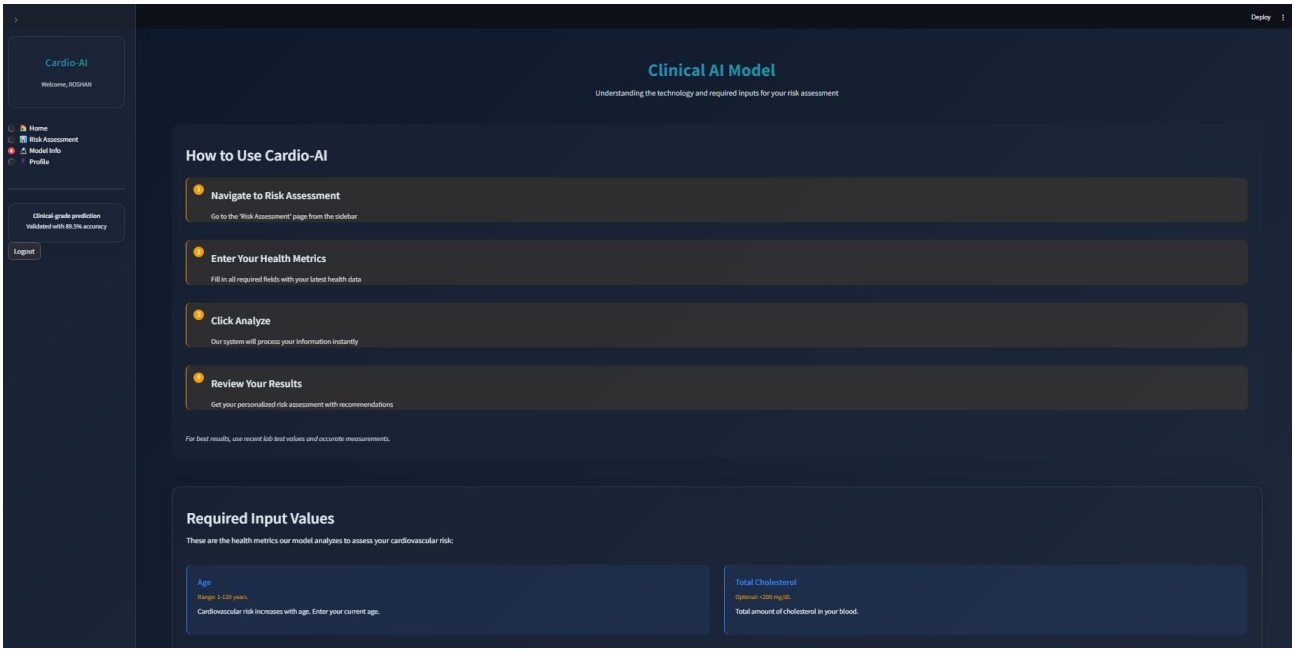
Implementation of home page in website



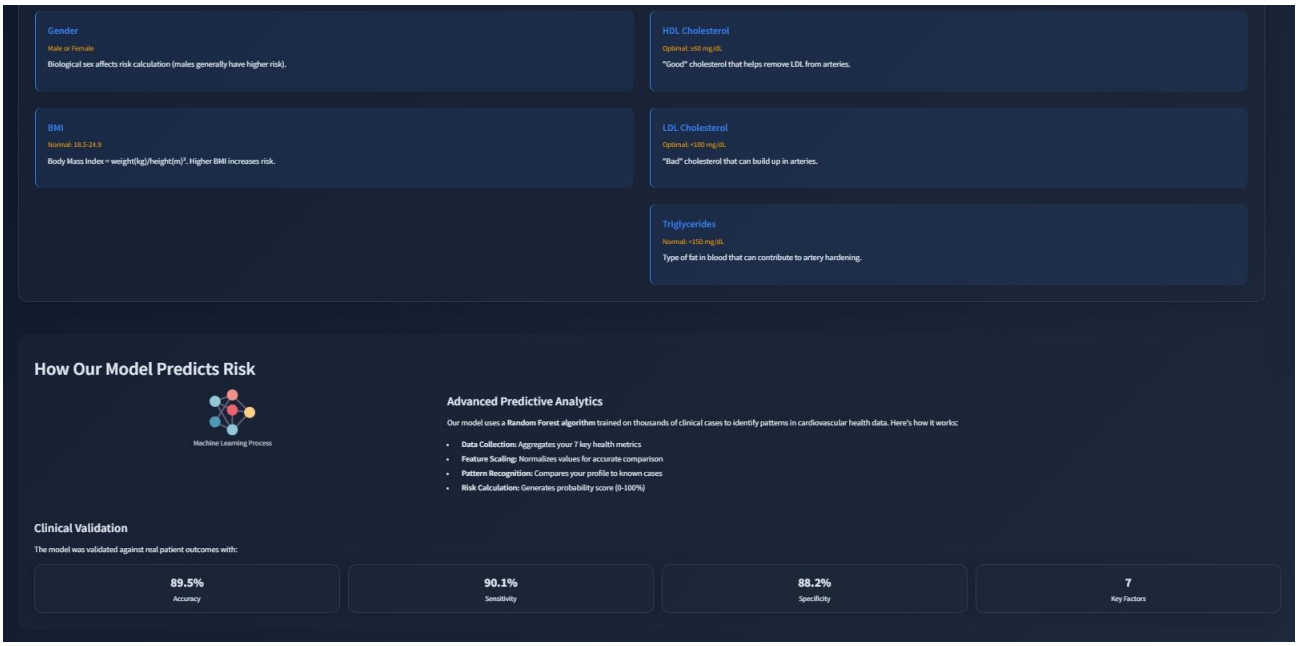
Implementation of registration page in website



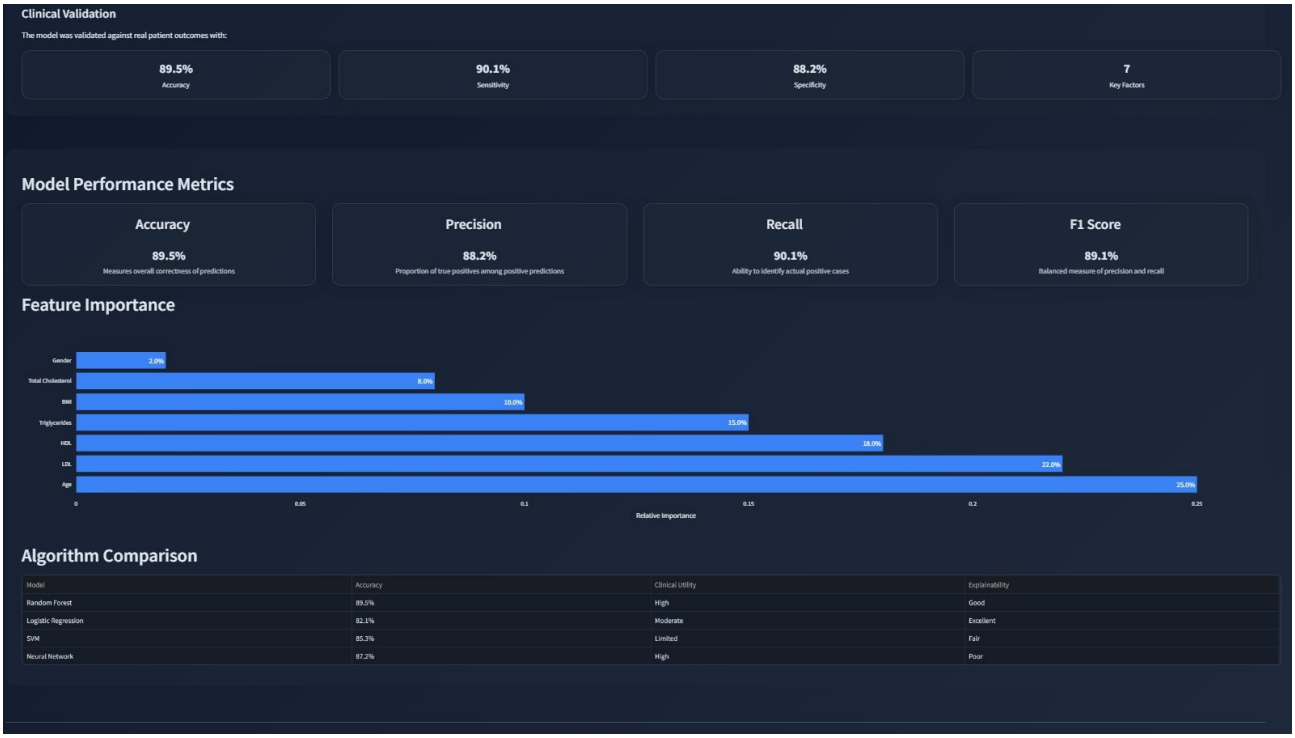
Implementation of risk assessment page in website



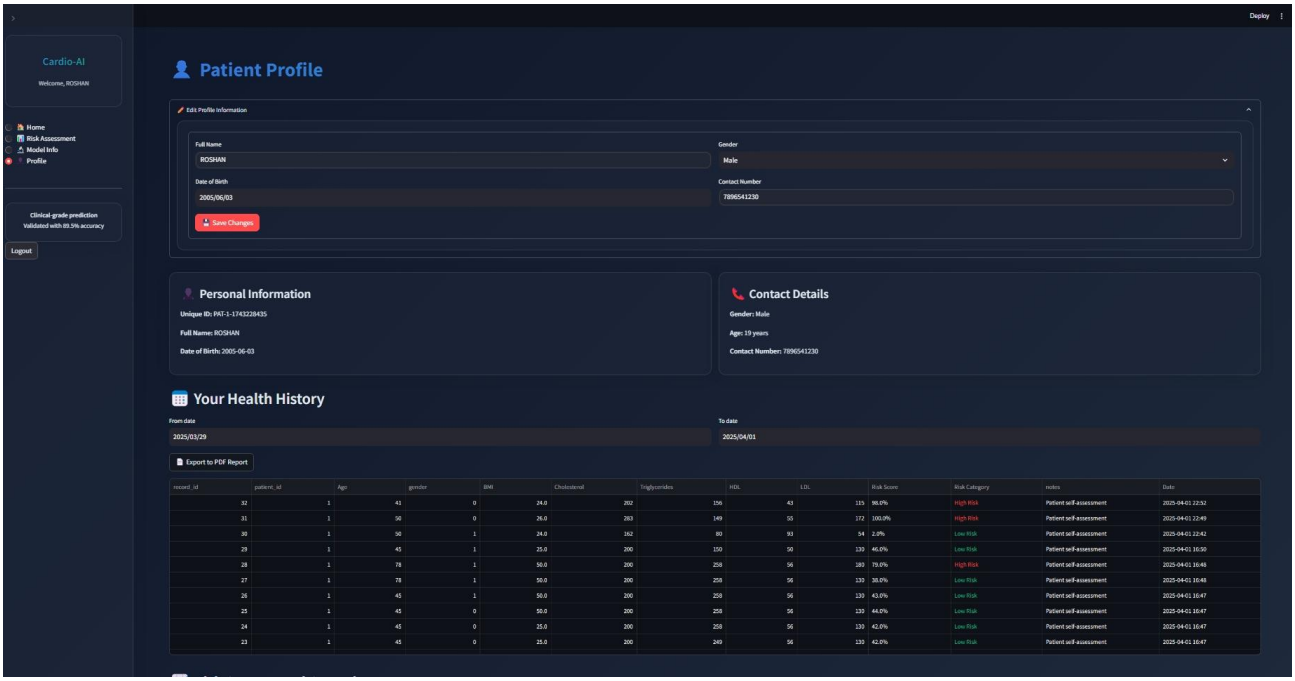
Implementation of model info page in website



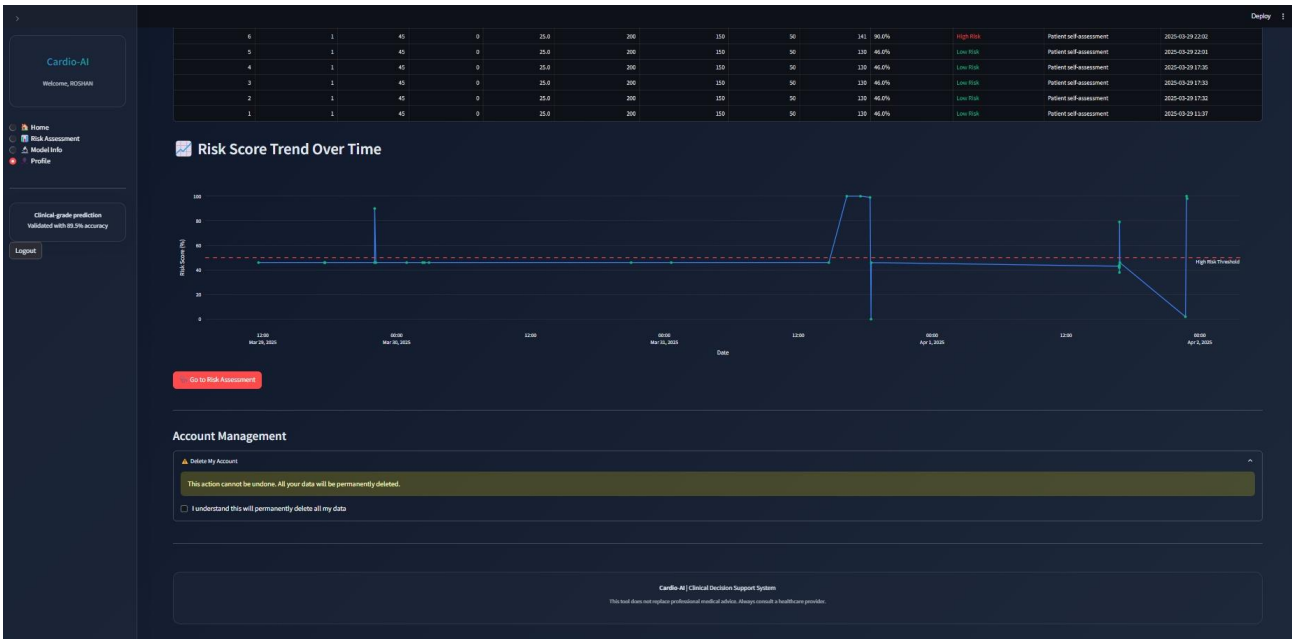
Implementation of model-info page in website



Implementation of model info page in website



Implementation of profile page in website



Implementation of profile page in website

8.2 CODING

: STREAMLIT CODE(app.py) :

```
import streamlit as st

import numpy as np

import pandas as pd

import joblib

import plotly.graph_objects as go

from streamlit_extras.let_it_rain import rain

from streamlit_extras.stylable_container import stylable_container

import time

import os

import hashlib
```

```
from database import DatabaseManager
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import Paragraph, SimpleDocTemplate, Table, TableStyle
from reportlab.lib import colors
import io
import datetime
import random

# Initialize database
db = DatabaseManager()


# --- Initialize Session State ---
if "current_page" not in st.session_state:
    st.session_state.current_page = "home"
if "authenticated" not in st.session_state:
    st.session_state.authenticated = False
if "user_id" not in st.session_state:
    st.session_state.user_id = None
if "patient_id" not in st.session_state:
    st.session_state.patient_id = None
if "username" not in st.session_state:
```

```
    st.session_state.username = None
if "is_admin" not in st.session_state:
    st.session_state.is_admin = False
if "unique_id" not in st.session_state:
    st.session_state.unique_id = None
# --- Load Model & Scaler ---
try:
    best_model = joblib.load(r"C:\Users\gamin\Documents\Finalyearproj\best_model (2).pkl")
    scaler = joblib.load(r"C:\Users\gamin\Documents\Finalyearproj\scaler (1).pkl")
except FileNotFoundError:
    st.error("Model or scaler file not found. Please ensure the files are in the correct directory.")
    st.stop()
feature_names = ['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL']
def predict_heart_disease(input_data):
    input_df = pd.DataFrame([input_data], columns=feature_names)
    input_scaled = scaler.transform(input_df)
    return best_model.predict_proba(input_scaled)[0]
def generate_pdf(patient_data, records):
    buffer = io.BytesIO()
    doc = SimpleDocTemplate(buffer, pagesize=letter)
    styles = getSampleStyleSheet()
    story = []
    # Title
    title = f"<h1>Cardio-AI Health Report for {patient_data['full_name']}</h1>"
```

```
story.append(Paragraph(title, styles['Title']))
# Patient Info
patient_info = f"""
<h2>Patient Information</h2>
<p><b>Unique ID:</b> {patient_data['unique_id']}</p>
<p><b>Date of Birth:</b> {patient_data['date_of_birth']}</p>
<p><b>Gender:</b> {patient_data['gender']}</p>
<p><b>Contact:</b> {patient_data['contact_number']}</p>
"""
story.append(Paragraph(patient_info, styles['Normal']))
# Health Records
story.append(Paragraph("<h2>Health History</h2>", styles['Heading2']))
# Prepare table data
table_data = [['Date', 'Age', 'BMI', 'Chol', 'HDL', 'LDL', 'TG', 'Risk Score', 'Category']]
for record in records:
    date = record['created_at'].strftime('%Y-%m-%d')
    table_data.append([
        date,
        str(record['age']),
        f"{record['bmi']:.1f}",
        str(record['chol']),
        str(record['hdl']),
        str(record['ldl']),
        str(record['tg']),
```

```
            f"{record['risk_score']:.1f}%",
            record['risk_category']
        ])
    # Create table
    t = Table(table_data)
    t.setStyle(TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
        ('FONTSIZE', (0, 0), (-1, 0), 10),
        ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
    ]))
    story.append(t)
    # Generate PDF
    doc.build(story)
    buffer.seek(0)
    return buffer
# --- Modern Glassmorphism Theme ---
def get_theme_config():
    return {
        "bg": "linear-gradient(135deg, #0F172A 0%, #1E293B 100%)",  # Gradient background
```

```
        "secondary_bg": "rgba(30, 41, 59, 0.7)",  # Semi-transparent
        "text": "#E2E8F0",      # Off-white text
        "primary": "#3B82F6",   # Blue
        "secondary": "#10B981", # Teal
        "accent": "#F59E0B",    # Amber
        "danger": "#EF4444",    # Red
        "card": "rgba(30, 41, 59, 0.5)",      # Glass card background
        "border": "1px solid rgba(255, 255, 255, 0.1)",  # Subtle border
        "hover": "0 8px 32px rgba(59, 130, 246, 0.2)",  # Glass hover effect
        "highlight": "0 0 0 2px rgba(59, 130, 246, 0.3)",  # Focus highlight
        "blur": "blur(16px)",   # Glass blur effect
        "shadow": "0 4px 30px rgba(0, 0, 0, 0.1)"  # Glass shadow
    }
# --- App Config ---
st.set_page_config(
    page_title="Cardio-AI",
    layout="wide",
    initial_sidebar_state="expanded",
    page_icon="🩺"
)



# --- Theme (Default to Dark) ---
theme_config = get_theme_config()
```

```
# --- Glassmorphism CSS with Modern Effects ---
st.markdown(f"""
<style>
    :root {{
        --primary: {theme_config["primary"]};
        --secondary: {theme_config["secondary"]};
        --bg: {theme_config["bg"]};
        --secondary-bg: {theme_config["secondary_bg"]};
        --text: {theme_config["text"]};
        --accent: {theme_config["accent"]};
        --danger: {theme_config["danger"]};
        --card: {theme_config["card"]};
        --border: {theme_config["border"]};
        --hover: {theme_config["hover"]};
        --highlight: {theme_config["highlight"]};
        --blur: {theme_config["blur"]};
        --shadow: {theme_config["shadow"]};
    }}
    /* Base styles with glassmorphism */
    .stApp {{
        background: var(--bg) !important;
        color: var(--text) !important;
        font-family: 'Inter', sans-serif;
    }}
```

```
[data-testid="stSidebar"] {{
    background: var(--secondary-bg) !important;
    backdrop-filter: var(--blur);
    -webkit-backdrop-filter: var(--blur);
    box-shadow: var(--shadow);
    border-right: var(--border) !important;
}}
/* Glassmorphism cards */
.glass-card {{
    background: var(--card);
    backdrop-filter: var(--blur);
    -webkit-backdrop-filter: var(--blur);
    border-radius: 16px;
    border: var(--border);
    box-shadow: var(--shadow);
    transition: all 0.3s ease;
}}
.glass-card:hover {{
    box-shadow: var(--hover);
    transform: translateY(-5px);
    border-color: rgba(255, 255, 255, 0.2);
}}
```

```
/* Button effects */
.stButton>button {{
    transition: all 0.2s ease !important;
}}
.stButton>button:hover {{
    transform: translateY(-1px) !important;
    box-shadow: var(--hover) !important;
}}
/* Input fields with glass effect */
.stTextInput>div>div>input,
.stNumberInput>div>div>input,
.stSelectbox>div>div>select {{
    background: rgba(30, 41, 59, 0.5) !important;
    backdrop-filter: var(--blur);
    -webkit-backdrop-filter: var(--blur);
    border-radius: 12px !important;
    border: var(--border) !important;
}}
.stTextInput>div>div>input:focus,
.stNumberInput>div>div>input:focus,
.stSelectbox>div>div>select:focus {{
    box-shadow: var(--highlight) !important;
```

```
    border-color: var(--primary) !important;
}}



/* Risk cards */
.risk-card {{
    border-left: 4px solid;
    transition: all 0.3s ease;
    backdrop-filter: var(--blur);
    -webkit-backdrop-filter: var(--blur);
}}
.high-risk {{
    border-color: var(--danger) !important;
    background: linear-gradient(
        to right,
        rgba(239, 68, 68, 0.1),
        rgba(239, 68, 68, 0.05)
    ) !important;
}}
.low-risk {{
    border-color: var(--secondary) !important;
    background: linear-gradient(
        to right,
        rgba(16, 185, 129, 0.1),
```

```
        rgba(16, 185, 129, 0.05)
    ) !important;
}}



/* Animations */
@keyframes fadeIn {{
    0% {{ opacity: 0; transform: translateY(10px); }}
    100% {{ opacity: 1; transform: translateY(0); }}
}}
@keyframes float {{
    0% {{ transform: translateY(0px); }}
    50% {{ transform: translateY(-5px); }}
    100% {{ transform: translateY(0px); }}
}}
.float {{
    animation: float 6s ease-in-out infinite;
}}
/* Gradient text for headings */
.gradient-text {{
    background: linear-gradient(90deg, var(--primary), var(--secondary));
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
```

```
    background-clip: text;
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}}
/* Glass button */
.glass-button {{
    background: rgba(59, 130, 246, 0.2) !important;
    backdrop-filter: var(--blur);
    -webkit-backdrop-filter: var(--blur);
    border-radius: 12px !important;
    border: 1px solid rgba(255, 255, 255, 0.1) !important;
    box-shadow: var(--shadow);
    transition: all 0.3s ease !important;
}}
.glass-button:hover {{
    background: rgba(59, 130, 246, 0.4) !important;
    transform: translateY(-2px) !important;
    box-shadow: var(--hover) !important;
}}
/* Modern tabs */
.stTabs [data-baseweb="tab-list"] {{
    gap: 10px;
}}
.stTabs [data-baseweb="tab"] {{
    background: rgba(30, 41, 59, 0.5) !important;
```

```
        backdrop-filter: var(--blur);
        -webkit-backdrop-filter: var(--blur);
        border-radius: 12px !important;
        border: var(--border) !important;
        padding: 8px 16px;
        transition: all 0.3s ease;
    }}
    .stTabs [data-baseweb="tab"]:hover {{
        background: rgba(59, 130, 246, 0.2) !important;
    }}

    .stTabs [aria-selected="true"] {{
        background: rgba(59, 130, 246, 0.4) !important;
        border-color: var(--primary) !important;
    }}
    /* Page entrance animation */
    .page-entrance {{
        animation: fadeIn 0.5s ease-out;
    }}
</style>
""", unsafe_allow_html=True)
def login_page():
    st.markdown("""
    <div class="page-entrance">
```

```
        <div style="
            max-width: 500px;
            margin: 2rem auto;
            padding: 2rem;
            background: rgba(30, 41, 59, 0.5);
            backdrop-filter: blur(16px);
            -webkit-backdrop-filter: blur(16px);
            border-radius: 16px;
            border: 1px solid rgba(255, 255, 255, 0.1);
            box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
        ">
            <div style="text-align: center; margin-bottom: 2rem;" class="float">
                <h1 class="gradient-text">Cardio-AI</h1>
                <p style="color: var(--text); opacity: 0.8;">Advanced Cardiovascular Risk Assessment</p>
            </div>
    """, unsafe_allow_html=True)
    tab1, tab2 = st.tabs([" 🔐 Login", " 📝 Register"])
    with tab1:
        with st.form("login_form"):
            username = st.text_input("Username", key="login_username")
            password = st.text_input("Password", type="password", key="login_password")
            submit = st.form_submit_button("Login", use_container_width=True, type="primary")
            if submit:
                user = db.get_user_by_username(username)
```

```
            if user and hashlib.sha256(password.encode()).hexdigest() == user['password_hash']:
                st.session_state['authenticated'] = True
                st.session_state['user_id'] = user['user_id']
                st.session_state['username'] = user['username']
                st.session_state['is_admin'] = user.get('is_admin', False)
                # Load patient data if exists
                patient = db.get_patient_by_user(user['user_id'])
                if patient:
                    st.session_state['patient_id'] = patient['patient_id']
                    st.session_state['unique_id'] = patient['unique_id']
                st.rerun()
            else:
                st.error("Invalid username or password")

    with tab2:
        with st.form("register_form"):
            new_username = st.text_input("Username", key="register_username")
            email = st.text_input("Email", key="register_email")
            new_password = st.text_input("Password", type="password", key="register_password")
            confirm_password = st.text_input("Confirm Password", type="password",
key="register_confirm_password")
            register = st.form_submit_button("Create Account", use_container_width=True,
type="primary")
            if register:
                # Validate all fields are filled
```

```
            if not new_username:
                st.error("Username is required")
            elif not email:
                st.error("Email is required")
            elif not new_password:
                st.error("Password is required")
            elif not confirm_password:
                st.error("Please confirm your password")
            elif new_password != confirm_password:
                st.error("Passwords do not match")
            else:
                # Validate email format
                if "@" not in email or "." not in email:
                    st.error("Please enter a valid email address")
                else:
                    hashed_password = hashlib.sha256(new_password.encode()).hexdigest()
                    user_id = db.create_user(new_username, email, hashed_password)
                    if user_id:
                        st.success("Account created successfully! Please log in.")
                    else:
                        st.error("Username or email already exists")
    st.markdown("""
        </div>
        <div style="text-align: center; margin-top: 2rem; color: var(--text); opacity: 0.7; font-size:
0.9rem;">
```

```
            <p>Secure login powered by advanced encryption</p>
        </div>
    </div>
    """, unsafe_allow_html=True
def user_management_page():
    st.markdown("""
    <div class="page-entrance">
        <h1 class="gradient-text" style="margin-bottom: 1.5rem;">👥 User Management</h1>
    """, unsafe_allow_html=True)
    if not st.session_state.get('is_admin'):
        st.error("You don't have permission to access this page")
        return
    # Get all users from the database
    users = db.get_all_users()
    if users:
        st.markdown("### User List")
        # Search functionality
        search_query = st.text_input("Search users", placeholder="Enter username or email")

        if search_query:
            users = [user for user in users
                    if search_query.lower() in user['username'].lower()
                    or search_query.lower() in user['email'].lower()]
        if not users:
```

```
            st.warning("No users found matching your search")
            return
        # Display users in a table with delete buttons
        for user in users:
            with st.container():
                cols = st.columns([3, 2, 1, 1])
                with cols[0]:
                    st.markdown(f"**{user['username']}** ({user['email']})")
                with cols[1]:
                    st.markdown(f"User ID: {user['user_id']}")
                with cols[2]:
                    st.markdown("**Admin**" if user.get('is_admin') else "🔵 User")
                with cols[3]:
                    if user['user_id'] != st.session_state.user_id:  # Prevent self-deletion
                        if st.button("Delete", key=f"delete_{user['user_id']}"):
                            if db.delete_user(user['user_id']):
                                st.success(f"User {user['username']} deleted successfully!")
                                st.rerun()
                            else:
                                st.error("Failed to delete user")
                    else:
                        st.warning("Current user")
    else:
        st.info("No users found in the database")
```

```
def patient_profile_page():
    st.markdown("""
    <div class="page-entrance">
        <h1 class="gradient-text" style="margin-bottom: 1.5rem;">👤 Patient Profile</h1>
    """, unsafe_allow_html=True)
    # Check if patient exists
    patient = db.get_patient_by_user(st.session_state['user_id'])
    if patient:
        st.session_state['patient_id'] = patient['patient_id']
        st.session_state['unique_id'] = patient['unique_id']
        # Edit Profile Section
        with st.expander("✏️ Edit Profile Information", expanded=False):
            with stylable_container(
                key="edit_profile_form",
                css_styles="""
                {
                    background: rgba(30, 41, 59, 0.5);
                    backdrop-filter: blur(16px);
                    -webkit-backdrop-filter: blur(16px);
                    border-radius: 16px;
                    padding: 1.5rem;
                    border: 1px solid rgba(255, 255, 255, 0.1);
                }
                """
```

```
        ):
            with st.form("edit_profile_form"):
                cols = st.columns(2)
                with cols[0]:
                    full_name = st.text_input("Full Name", value=patient['full_name'],
key="edit_full_name")
                    date_of_birth = st.date_input("Date of Birth",
                                                  value=patient['date_of_birth'],
                                                  max_value=datetime.date.today(),
                                                  key="edit_dob")
                with cols[1]:
                    gender = st.selectbox("Gender",
                                          ["Male", "Female", "Other"],
                                          index=["Male", "Female", "Other"].index(patient['gender']),
                                          key="edit_gender")
                    contact_number = st.text_input("Contact Number",
                                                   value=patient['contact_number'],
                                                   key="edit_contact")
                if st.form_submit_button("💾 Save Changes", type="primary"):
                    if db.update_patient(
                        patient['patient_id'],
                        full_name,
                        date_of_birth,
                        gender,
                        contact_number
```

```
            ):
                st.success("Profile updated successfully!")
                st.rerun()
            else:
                st.error("Failed to update profile")
# View Profile Section
cols = st.columns(2)
with cols[0]:
    st.markdown(f"""
    <div class="glass-card" style="padding: 1.5rem; margin: 1rem 0;">
        <h3>👤 Personal Information</h3>
        <p><strong>Unique ID:</strong> {patient['unique_id']}</p>
        <p><strong>Full Name:</strong> {patient['full_name']}</p>
        <p><strong>Date of Birth:</strong> {patient['date_of_birth']}</p>
    </div>
    """, unsafe_allow_html=True)

with cols[1]:
    age = (datetime.date.today() - patient['date_of_birth']).days // 365
    st.markdown(f"""
    <div class="glass-card" style="padding: 1.5rem; margin: 1rem 0;">
        <h3>📞 Contact Details</h3>
        <p><strong>Gender:</strong> {patient['gender']}</p>
        <p><strong>Age:</strong> {age} years</p>
```

```
        <p><strong>Contact Number:</strong> {patient['contact_number']}</p>
    </div>
    """, unsafe_allow_html=True)

# Health Records Section
st.markdown("## 📅 Your Health History")
records = db.get_patient_records(patient['patient_id'])
if records:
    # Date range selector
    record_dates = [r['created_at'].date() for r in records]
    min_date = min(record_dates)
    max_date = max(record_dates)
    col1, col2 = st.columns(2)
    with col1:
        start_date = st.date_input("From date",
                                   value=min_date,
                                   min_value=datetime.date(min_date.year - 5, 1, 1),
                                   max_value=max_date,
                                   key="start_date_filter")
    with col2:
        end_date = st.date_input("To date",
                                 value=max_date,
                                 min_value=min_date,
                                 max_value=datetime.date.today(),
```

```
                    key="end_date_filter")
# Filter records
filtered_records = [
    r for r in records
    if start_date <= r['created_at'].date() <= end_date
]
if filtered_records:
    # PDF Export
    if st.button(" 📄 Export to PDF Report", key="pdf_export"):
        pdf_buffer = generate_pdf(patient, filtered_records)
        st.download_button(
            label="⬇️ Download PDF Report",
            data=pdf_buffer,
            file_name=f"cardio_ai_report_{patient['unique_id']}_{datetime.date.today()}.pdf",
            mime="application/pdf",
            key="pdf_download"
        )
    # Enhanced Data Display
    df = pd.DataFrame(filtered_records)
    df['created_at'] = df['created_at'].dt.strftime('%Y-%m-%d %H:%M')
    df['risk_score'] = df['risk_score'].apply(lambda x: f"{x:.1f}%")
    # Style function for risk categories
    def color_risk(val):
        if val == 'High Risk':
```

```
        return 'color: #EF4444; font-weight: bold;'
    else:
        return 'color: #10B981; font-weight: bold;'
styled_df = df.style.applymap(color_risk, subset=['risk_category'])
st.dataframe(
    styled_df,
    use_container_width=True,
    hide_index=True,
    column_config={
        "created_at": "Date",
        "age": "Age",
        "bmi": st.column_config.NumberColumn("BMI", format="%.1f"),
        "chol": "Cholesterol",
        "hdl": "HDL",
        "ldl": "LDL",
        "tg": "Triglycerides",
        "risk_score": "Risk Score",
        "risk_category": "Risk Category"
    }
)
# Risk Trend Visualization
st.markdown("## 📈 Risk Score Trend Over Time")
trend_df = pd.DataFrame(filtered_records)
trend_df['created_at'] = pd.to_datetime(trend_df['created_at'])
```

```
trend_df = trend_df.sort_values('created_at')
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=trend_df['created_at'],
    y=trend_df['risk_score'],
    mode='lines+markers',
    name='Risk Score',
    line=dict(color=theme_config["primary"], width=2),
    marker=dict(size=6, color=theme_config["secondary"])
))
fig.add_hline(
    y=50,
    line_dash="dash",
    line_color=theme_config["danger"],
    annotation_text="High Risk Threshold",
    annotation_position="bottom right"
)
fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Risk Score (%)",
    hovermode="x unified",
    plot_bgcolor='rgba(0,0,0,0)',
    paper_bgcolor='rgba(0,0,0,0)',
    font=dict(color=theme_config["text"]),
```

```
            margin=dict(l=50, r=50, b=50, t=50)
        )
        st.plotly_chart(fig, use_container_width=True)
    else:
        st.warning(f"⚠️ No records found between {start_date} and {end_date}")
else:
    st.info("ℹ️ No health records found. Complete a risk assessment to get started.")
if st.button("🩺 Go to Risk Assessment", type="primary", key="profile_to_risk"):
    st.session_state.current_page = "risk"



# Account Deletion Section
st.markdown("---")
st.markdown("### Account Management")
with st.expander("⚠️ Delete My Account", expanded=False):
    st.warning("This action cannot be undone. All your data will be permanently deleted.")
    confirm = st.checkbox("I understand this will permanently delete all my data")
    if confirm and st.button("Confirm Account Deletion", type="primary"):
        if db.delete_user(st.session_state.user_id):
            st.success("Your account has been deleted successfully.")
            # Clear session and return to login
            for key in list(st.session_state.keys()):
                del st.session_state[key]
            st.rerun()
```

```
            else:
                st.error("Failed to delete account. Please try again.")
    else:
        # Patient registration form
        st.subheader("Complete Your Patient Profile")
        with stylable_container(
            key="patient_form",
            css_styles="""
            {
                background: rgba(30, 41, 59, 0.5);
                backdrop-filter: blur(16px);
                -webkit-backdrop-filter: blur(16px);
                border-radius: 16px;
                padding: 1.5rem;
                border: 1px solid rgba(255, 255, 255, 0.1);
            }
            """
        ):
            with st.form("patient_form"):
                full_name = st.text_input("Full Name", key="patient_full_name")
                date_of_birth = st.date_input("Date of Birth",
                                              max_value=datetime.date.today(),
                                              key="patient_dob")
                gender = st.selectbox("Gender", ["Male", "Female", "Other"], key="patient_gender")
```

```
            contact_number = st.text_input("Contact Number", key="patient_contact")
            submit = st.form_submit_button("💾 Save Profile", type="primary")
            if submit:
                unique_id = db.create_patient(
                    st.session_state['user_id'],
                    full_name,
                    date_of_birth,
                    gender,
                    contact_number
                )
                if unique_id:
                    # Refresh patient data
                    patient = db.get_patient_by_user(st.session_state['user_id'])
                    if patient:
                        st.session_state['patient_id'] = patient['patient_id']
                        st.session_state['unique_id'] = patient['unique_id']
                        st.success("✅ Profile saved successfully!")
                        st.rerun()
                    else:
                        st.error("Failed to load patient profile after creation")
def home_page():
    st.markdown("""
    <div class="page-entrance">
        <div style="text-align: center; margin-bottom: 3rem;">
```

```
        <h1 class="gradient-text" style="font-size: 2.5rem;">Cardio-AI Health Analysis</h1>
        <p style="font-size: 1.1rem; opacity: 0.9;">
        Advanced cardiovascular risk prediction powered by machine learning
        </p>
    </div>
""", unsafe_allow_html=True)
# Hero Section
with stylable_container(
    key="hero",
    css_styles=f"""
    {{
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 2rem 0;
        border: 1px solid rgba(255, 255, 255, 0.1);
        box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
    }}
    """
):
    col1, col2 = st.columns([2, 1], gap="medium")
    with col1:
```

```
        st.markdown("""
        <div>
            <h2 style='margin-bottom: 1rem;'>Your Personal Heart Health Assistant</h2>
            <p style='font-size: 1.1rem; line-height: 1.6;'>
        Cardio-AI analyzes your health metrics using clinically validated algorithms to assess your cardiovascular risk with <strong>89.5% accuracy</strong>. Early detection leads to better outcomes.
            </p>
            <p style='font-size: 1rem; font-style: italic; opacity: 0.8;'>
            "Prevention is better than cure" - Hippocrates
            </p>
        </div>
        """, unsafe_allow_html=True)
    with col2:
        st.markdown("""
        <div class="float" style="text-align: center;">
            <img src="https://cdn-icons-png.flaticon.com/512/3059/3059518.png" width="120">
        </div>
        """, unsafe_allow_html=True)
    with stylable_container(
    key="heart_disease_intro",
    css_styles="""
    {
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
```

```
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
        border: 1px solid rgba(255, 255, 255, 0.1);
    }
    """
):
    st.markdown("""
    ## ❤️ What is Heart Disease?
    Heart disease refers to various conditions that affect your heart's structure and function.
    It's the leading cause of death globally, but many forms are preventable with healthy lifestyle choices.
    Heart disease develops when:
    - Arteries become narrowed or blocked (atherosclerosis)
    - The heart muscle becomes weak or damaged
    - Heart valves don't function properly
    - Electrical signals controlling heartbeat are disrupted
    """)

# ==========================================
# TYPES OF HEART DISEASE SECTION
```

```
# ============================================

with stylable_container(
    key="types_container",
    css_styles="""
    {
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
    }
    """
):
    st.markdown("""
    ## 🩺 Types of Heart Disease
    The main types of cardiovascular diseases include:
    """)
    # Create columns for the types
    col1, col2 = st.columns(2, gap="medium")
    with col1:
        with stylable_container(
            key="cad_card",
            css_styles="""
```

```
    {
        background: rgba(239, 68, 68, 0.1);
        border-radius: 12px;
        padding: 1.5rem;
        margin: 1rem 0;
        border-left: 4px solid #EF4444;
    }
    """
):
    st.markdown("""
    ### 1. Coronary Artery Disease (CAD)
    - Most common type
    - Caused by plaque buildup in arteries
    - Can lead to heart attacks
    - Symptoms: Chest pain, shortness of breath
    """)
with stylable_container(
    key="arrhythmia_card",
    css_styles="""
    {
        background: rgba(59, 130, 246, 0.1);
        border-radius: 12px;
        padding: 1.5rem;
        margin: 1rem 0;
```

```
            border-left: 4px solid #3B82F6;
        }
        """
    ):
        st.markdown("""
        ### 2. Arrhythmias
        - Irregular heartbeats
        - Heart may beat too fast, slow, or irregularly
        - Can cause dizziness or fainting
        - Some types are life-threatening
        """)
with col2:
    with stylable_container(
        key="hf_card",
        css_styles="""
        {
            background: rgba(16, 185, 129, 0.1);
            border-radius: 12px;
            padding: 1.5rem;
            margin: 1rem 0;
            border-left: 4px solid #10B981;
        }
        """
    ):
```

```
    st.markdown("""
    ### 3. Heart Failure
    - Heart can't pump blood effectively
    - Develops gradually over time
    - Symptoms: Fatigue, swelling in legs
    - Managed with medication and lifestyle
    """)
with stylable_container(
    key="valve_card",
    css_styles="""
    {
        background: rgba(245, 158, 11, 0.1);
        border-radius: 12px;
        padding: 1.5rem;
        margin: 1rem 0;
        border-left: 4px solid #F59E0B;
    }
    """
):
    st.markdown("""
    ### 4. Valve Disorders
    - Heart valves don't open/close properly
    - Can be congenital or develop later
    - May require surgical repair
```

```
            - Symptoms: Fatigue, irregular heartbeat
            """)




# ================================================
# RISK FACTORS SECTION
# ================================================
with stylable_container(
    key="risk_factors",
    css_styles="""
    {
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
    }
    """
):
```

```
    st.markdown("""
    ## ⚠️ Major Risk Factors
    | Controllable Factors | Uncontrollable Factors |
    |----------------------|------------------------|
    | High blood pressure | Age (risk increases after 45) |
    | High cholesterol | Family history of heart disease |
    | Smoking | Gender (men at higher risk) |
    | Diabetes | Race (some ethnic groups at higher risk) |
    | Obesity | Previous heart attack |
    | Physical inactivity |  |
    | Poor diet |  |
    | Excessive alcohol |  |
    """)
# ==========================================
# PREVENTION SECTION
# ==========================================
```

```
with stylable_container(
    key="prevention",
    css_styles="""
    {
        background: rgba(16, 185, 129, 0.1);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
        border-left: 4px solid #10B981;
    }
    """
):
    st.markdown("""
    ## 🛡️ Prevention Tips
    - 🥗 Eat a heart-healthy diet (fruits, vegetables, whole grains)
    - ⛹️ Get regular exercise (150 mins/week moderate activity)
    - 🚭 Avoid tobacco products
    - 🧘 Manage stress through relaxation techniques
    - 🩺 Get regular health screenings
    - 🧂 Limit salt and sugar intake
    - 🥑 Choose healthy fats (olive oil, nuts, fish)
    """)
```

```
    st.markdown("## How It Works")

    cols = st.columns(3, gap="medium")

    steps = [

        ("🩺", "Clinical Input", "Provide your health metrics including cholesterol levels, BMI, and other vital signs"),

        ("🧠", "AI Analysis", "Our model processes your data using validated medical algorithms"),

        ("📋", "Personalized Report", "Receive a detailed risk assessment with actionable insights")

    ]

    for i, (icon, title, desc) in enumerate(steps):

        with cols[i]:

            st.markdown(f"""

            <div class="glass-card" style="padding: 1.5rem; text-align: center; height: 100%;">

                <div style="font-size: 2rem; margin-bottom: 0.5rem;">{icon}</div>

                <h3 style='margin-bottom: 0.5rem;'>{title}</h3>

                <p style='color: var(--text); opacity: 0.9;'>{desc}</p>

            </div>

            """, unsafe_allow_html=True)

    # Key Features

    st.markdown("## Why Choose Cardio-AI?")

    features = [

        ("✅", "Clinically Validated", "Trained on real patient data with 89.5% accuracy"),

        ("⚡", "Instant Results", "Get your risk assessment in seconds"),

        ("🔒", "Privacy Focused", "Your data never leaves your device"),

        ("📈", "Actionable Insights", "Personalized recommendations based on your results")
```

```
    ]
    for i in range(0, len(features), 2):
        cols = st.columns(2, gap="medium")
        for j in range(2):
            if i+j < len(features):
                with cols[j]:
                    st.markdown(f"""
                    <div class="glass-card" style="padding: 1.5rem;">
                        <div style="display: flex; align-items: center; gap: 1rem; margin-bottom: 0.5rem;">
                            <span style="font-size: 1.5rem;">{features[i+j][0]}</span>
                            <h4 style="margin: 0;">{features[i+j][1]}</h4>
                        </div>
                        <p style="opacity: 0.9; margin: 0;">{features[i+j][2]}</p>
                    </div>
                    """, unsafe_allow_html=True)
def risk_assessment_page():
    st.markdown("""
    <div class="page-entrance">
        <div style="text-align: center; margin-bottom: 2rem;">
            <h1 class="gradient-text" style="font-size: 2.5rem;">Heart Disease Risk Assessment</h1>
            <p style="font-size: 1.1rem; opacity: 0.9;">
            Enter your health metrics below to receive your personalized risk analysis
            </p>
        </div>
```

```
""", unsafe_allow_html=True)

# First check if patient profile exists
if not st.session_state.get('patient_id'):
    st.error("⚠️ Please complete your patient profile before performing risk assessments")
    if st.button("Go to Profile Page"):
        st.session_state.current_page = "profile"
        st.rerun()
    return
# Patient Health Metrics Section
with stylable_container(
    key="input_form",
    css_styles=f"""
    {{
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
        border: 1px solid rgba(255, 255, 255, 0.1);
        box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
    }}
    """
):
```

```
    st.markdown("### Patient Health Metrics")

    cols = st.columns(2, gap="medium")

    with cols[0]:

        age = st.number_input("Age (years)", min_value=1, max_value=120, value=45, help="Enter
your current age")

        if age < 1:

            st.error("Age must be at least 1.")

            st.stop()

        sex = st.selectbox("Gender", ["Female", "Male"], help="Biological sex affects
cardiovascular risk")

        gender = 1 if sex == "Male" else 0

        bmi = st.number_input("Body Mass Index (BMI)", min_value=10.0, max_value=50.0,
value=25.0, help="Weight (kg) / height (m)²")

        if bmi < 10.0:

            st.error("BMI must be at least 10.")

            st.stop()

    with cols[1]:

        chol = st.number_input("Total Cholesterol (mg/dL)", min_value=100, max_value=300,
value=200, help="Desirable: <200 mg/dL")

        if chol < 100:

            st.error("Total Cholesterol must be at least 100 mg/dL.")

            st.stop()

        tg = st.number_input("Triglycerides (mg/dL)", min_value=50, max_value=500, value=150,
help="Desirable: <150 mg/dL")

        if tg < 50:

            st.error("Triglycerides must be at least 50 mg/dL.")

            st.stop()
```

```
            hdl = st.number_input("HDL Cholesterol (mg/dL)", min_value=20, max_value=100,
value=50, help="Desirable: >60 mg/dL")

            if hdl < 20:

                st.error("HDL Cholesterol must be at least 20 mg/dL.")

                st.stop()

            ldl = st.number_input("LDL Cholesterol (mg/dL)", min_value=50, max_value=250,
value=130, help="Optimal: <100 mg/dL")

            if ldl < 50:

                st.error("LDL Cholesterol must be at least 50 mg/dL.")

                st.stop()

    if st.button("**Analyze My Cardiovascular Risk**", use_container_width=True, type="primary",
key="analyze_button"):

        # Double-check patient_id exists

        if not st.session_state.get('patient_id'):

            st.error("Patient profile not found. Please complete your profile first.")

            st.session_state.current_page = "profile"

            st.rerun()

            return

        with st.spinner('Processing your health metrics...'):

            time.sleep(1)

            try:

                input_data = {

                    'age': float(age),

                    'gender': gender,

                    'bmi': float(bmi),

                    'chol': float(chol),
```

```
    'tg': float(tg),
    'hdl': float(hdl),
    'ldl': float(ldl)
}
prediction = predict_heart_disease(list(input_data.values()))
high_risk = prediction[1] * 100
risk_category = "High Risk" if high_risk > 50 else "Low Risk"
# Save to database
record_id = db.save_health_record(
    st.session_state['patient_id'],
    input_data,
    high_risk,
    risk_category,
    notes="Patient self-assessment"
)
if record_id:
    st.success("Assessment saved to your health history!")
else:
    st.error("Failed to save assessment. Please try again.")
    return
# Visual feedback
if high_risk > 50:
    rain(emoji="⚠️", font_size=50, falling_speed=3, animation_length=5)
else:
```

```
    rain(emoji="✅", font_size=20, falling_speed=3, animation_length=1)
# Risk Card
risk_class = "high-risk" if high_risk > 50 else "low-risk"
risk_emoji = "⚠️ High Risk" if high_risk > 50 else "✅ Low Risk"
risk_color = "danger" if high_risk > 50 else "secondary"
st.markdown(f"""
<div class='risk-card glass-card' style="text-align: center; padding: 1.5rem; margin: 2rem
0;">
    <h2 style='margin-bottom: 0.5rem;'>{risk_emoji}</h2>
    <div style='font-size: 2.5rem; font-weight: 700; margin: 1rem 0;'>
        {high_risk:.1f}%
    </div>
    <p style='font-size: 1.1rem;'>
    Probability of cardiovascular disease
    </p>
</div>
""", unsafe_allow_html=True)
# Gauge Chart
fig = go.Figure(go.Indicator(
    mode = "gauge+number",
    value = high_risk,
    domain = {'x': [0, 1], 'y': [0, 1]},
    title = {'text': "Risk Level", 'font': {'size': 18}},
    gauge = {
```

```
        'axis': {'range': [0, 100], 'tickwidth': 1, 'tickcolor': "darkgray"},
        'bar': {'color': theme_config[risk_color]},
        'bgcolor': "white",
        'borderwidth': 2,
        'bordercolor': "gray",
        'steps': [
            {'range': [0, 30], 'color': theme_config["secondary"]},
            {'range': [30, 70], 'color': theme_config["accent"]},
            {'range': [70, 100], 'color': theme_config["danger"]}
        ],
        'threshold': {
            'line': {'color': "white", 'width': 4},
            'thickness': 0.75,
            'value': high_risk
        }
    }
))
fig.update_layout(
    height=300,
    margin=dict(t=50, b=30),
    font=dict(color=theme_config["text"]),
    paper_bgcolor='rgba(0,0,0,0)'
)
st.plotly_chart(fig, use_container_width=True)
```

```
# Recommendations Section
st.markdown("## Health Recommendations")
if high_risk > 50:
    with stylable_container(
        key="warning_box",
        css_styles=f"""
        {{
            background: rgba(239, 68, 68, 0.1);
            backdrop-filter: blur(16px);
            -webkit-backdrop-filter: blur(16px);
            border-radius: 16px;
            padding: 1.5rem;
            margin: 1rem 0;
            border-left: 4px solid {theme_config["danger"]};
        }}
        """
    ):
        st.markdown("""
        <div>
            <h3 style='margin-bottom: 1rem;'>Clinical Guidance</h3>
            <p style='margin-bottom: 1rem;'>
            Based on your results, we recommend consulting with a healthcare professional for further evaluation.
            </p>
            <div style='display: grid; grid-template-columns: repeat(2, 1fr); gap: 1.5rem;'>
```

```
            <div>
                <h4 style='font-size: 1.1rem;'>🩺 Medical Follow-up</h4>
                <ul>
                    <li>Schedule a cardiology consultation</li>
                    <li>Consider lipid profile testing</li>
                    <li>Blood pressure monitoring</li>
                </ul>
            </div>
            <div>
                <h4 style='font-size: 1.1rem;'>💊 Treatment Options</h4>
                <ul>
                    <li>Discuss statin therapy</li>
                    <li>Evaluate blood pressure meds</li>
                    <li>Diabetes screening</li>
                </ul>
            </div>
        </div>
    </div>
    """, unsafe_allow_html=True)
else:
    with stylable_container(
        key="success_box",
        css_styles=f"""
        {{
```

```
        background: rgba(16, 185, 129, 0.1);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 1.5rem;
        margin: 1rem 0;
        border-left: 4px solid {theme_config["secondary"]};
    }}
    """
):
    st.markdown("""
    <div>
        <h3 style='margin-bottom: 1rem;'>Preventive Measures</h3>
        <p style='margin-bottom: 1rem;'>
        Maintain your heart health with these evidence-based recommendations:
        </p>
        <div style='display: grid; grid-template-columns: repeat(2, 1fr); gap: 1.5rem;'>
            <div>
                <h4 style='font-size: 1.1rem;'>🍏 Nutrition</h4>
                <ul>
                    <li>Increase fiber intake</li>
                    <li>Choose healthy fats</li>
                    <li>Limit processed foods</li>
                </ul>
```

```
            </div>
            <div>
                <h4 style='font-size: 1.1rem;'>🏋️ Activity</h4>
                <ul>
                    <li>150 min/week moderate exercise</li>
                    <li>Strength training 2x/week</li>
                    <li>Reduce sedentary time</li>
                </ul>
            </div>
        </div>
    </div>
    """, unsafe_allow_html=True)
# Biomarker Analysis
st.markdown("## Your Biomarker Breakdown")
with stylable_container(
    key="biomarker_analysis",
    css_styles=f"""
    {{
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 1.5rem;
        margin: 1rem 0;
```

```
    }}
    """
):
    st.markdown(f"""
    <div>
        <div style='display: grid; grid-template-columns: repeat(2, 1fr); gap: 2rem;'>
            <div>
                <h4 style='margin-bottom: 1rem;'>Optimal Ranges</h4>
                <ul style='list-style-type: none; padding: 0;'>
                    <li style='margin-bottom: 0.75rem;'>• Total Cholesterol: <200 mg/dL</li>
                    <li style='margin-bottom: 0.75rem;'>• LDL Cholesterol: <100 mg/dL</li>
                    <li style='margin-bottom: 0.75rem;'>• HDL Cholesterol: >60 mg/dL</li>
                    <li style='margin-bottom: 0.75rem;'>• Triglycerides: <150 mg/dL</li>
                    <li style='margin-bottom: 0.75rem;'>• BMI: 18.5-24.9</li>
                </ul>
            </div>
            <div>
                <h4 style='margin-bottom: 1rem;'>Your Values</h4>
                <div style='display: grid; gap: 0.75rem;'>
                    <div style='display: flex; justify-content: space-between;'>
                        <span>Total Cholesterol: {chol} mg/dL</span>
                        <span style="color: {'var(--danger)' if chol > 200 else 'var(--secondary)'}">
                            {"⚠️ Above optimal" if chol > 200 else "✅ Optimal"}
                        </span>
```

```
</div>
<div style='display: flex; justify-content: space-between;'>
    <span>LDL Cholesterol: {ldl} mg/dL</span>
    <span style="color: {'var(--danger)' if ldl > 100 else 'var(--secondary)'}">
        {"⚠️ Above optimal" if ldl > 100 else "✅ Optimal"}
    </span>
</div>
<div style='display: flex; justify-content: space-between;'>
    <span>HDL Cholesterol: {hdl} mg/dL</span>
    <span style="color: {'var(--danger)' if hdl < 60 else 'var(--secondary)'}">
        {"⚠️ Below optimal" if hdl < 60 else "✅ Optimal"}
    </span>
</div>
<div style='display: flex; justify-content: space-between;'>
    <span>Triglycerides: {tg} mg/dL</span>
    <span style="color: {'var(--danger)' if tg > 150 else 'var(--secondary)'}">
        {"⚠️ Above optimal" if tg > 150 else "✅ Optimal"}
    </span>
</div>
<div style='display: flex; justify-content: space-between;'>
    <span>BMI: {bmi}</span>
    <span style="color: {'var(--danger)' if bmi > 25 else 'var(--secondary)'}">
        {"⚠️ Above optimal" if bmi > 25 else "✅ Optimal"}
    </span>
```

```
                    </div>
                </div>
            </div>
        </div>
        """, unsafe_allow_html=True)
    except ValueError:
        st.error("Please enter valid numerical values for all fields.")
def model_info_page():
    st.markdown("""
    <div class="page-entrance">
        <div style="text-align: center; margin-bottom: 2rem;">
            <h1 class="gradient-text" style="font-size: 2.5rem;">Clinical AI Model</h1>
            <p style="font-size: 1.1rem; opacity: 0.9;">
            Understanding the technology and required inputs for your risk assessment
            </p>
        </div>
    """, unsafe_allow_html=True)
    # How to Use Section
    with stylable_container(
        key="how_to_predict",
        css_styles=f"""
        {{
            background: rgba(30, 41, 59, 0.5);
```

```
            backdrop-filter: blur(16px);
            -webkit-backdrop-filter: blur(16px);
            border-radius: 16px;
            padding: 2rem;
            margin: 1.5rem 0;
        }}
        """
    ):
        st.markdown("## How to Use Cardio-AI")

        steps = [
            ("1", "Navigate to Risk Assessment", "Go to the 'Risk Assessment' page from the sidebar"),
            ("2", "Enter Your Health Metrics", "Fill in all required fields with your latest health data"),
            ("3", "Click Analyze", "Our system will process your information instantly"),
            ("4", "Review Your Results", "Get your personalized risk assessment with recommendations")
        ]
        for num, title, desc in steps:
            with stylable_container(
                key=f"step_{num}",
                css_styles=f"""
                {{
                    background: rgba(245, 158, 11, 0.1);
                    backdrop-filter: blur(16px);
                    -webkit-backdrop-filter: blur(16px);
```

```
        border-radius: 12px;
        padding: 1rem;
        margin: 0.75rem 0;
        border-left: 3px solid {theme_config["accent"]};
    }}
    """
):
    st.markdown(f"""
    <div>
        <div style="display: flex; gap: 1rem; align-items: flex-start;">
            <div style="
                background: {theme_config["accent"]};
                color: white;
                width: 24px;
                height: 24px;
                border-radius: 50%;
                display: flex;
                align-items: center;
                justify-content: center;
                flex-shrink: 0;
            ">{num}</div>
            <div>
                <h4 style="margin: 0 0 0.25rem 0;">{title}</h4>
                <p style="margin: 0; opacity: 0.9;">{desc}</p>
```

```
                </div>
            </div>
        </div>
        """, unsafe_allow_html=True)
    st.markdown("""
    <div style="margin-top: 1.5rem;">
        <p style="font-style: italic; opacity: 0.8;">
        For best results, use recent lab test values and accurate measurements.
        </p>
    </div>
    """, unsafe_allow_html=True)
     # New Input Values Section with Glassmorphism and Neon Effects
with stylable_container(
    key="input_values_section",
    css_styles="""
    {
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
        border: 1px solid rgba(255, 255, 255, 0.1);
        box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
```

```
    transition: all 0.3s ease;
}
.input-value-card {
    background: rgba(59, 130, 246, 0.1);
    border-radius: 12px;
    padding: 1.5rem;
    margin: 1rem 0;
    border-left: 3px solid var(--primary);
    transition: all 0.3s ease;
}
.input-value-card:hover {
    transform: translateY(-3px);
    box-shadow: 0 0 15px rgba(59, 130, 246, 0.5);
    border-left: 3px solid var(--accent);
}
.input-value-title {
    font-size: 1.2rem;
    font-weight: 600;
    margin-bottom: 0.5rem;
    color: var(--primary);
}
.input-value-range {
    font-size: 0.9rem;
    color: var(--accent);
```

```
        margin-bottom: 0.5rem;
    }
    """
):
    st.markdown("## Required Input Values")
    st.markdown("""
    <p style="margin-bottom: 1.5rem; font-size: 1.1rem;">
    These are the health metrics our model analyzes to assess your cardiovascular risk:
    </p>
    """, unsafe_allow_html=True)
    # Input Value Cards
    cols = st.columns(2, gap="medium")
    with cols[0]:
        st.markdown("""
        <div class="input-value-card">
            <div class="input-value-title">Age</div>
            <div class="input-value-range">Range: 1-120 years</div>
            <p>Cardiovascular risk increases with age. Enter your current age.</p>
        </div>
        """, unsafe_allow_html=True)
        st.markdown("""
        <div class="input-value-card">
            <div class="input-value-title">Gender</div>
            <div class="input-value-range">Male or Female</div>
```

```
        <p>Biological sex affects risk calculation (males generally have higher risk).</p>
    </div>
    """, unsafe_allow_html=True)

    st.markdown("""
    <div class="input-value-card">
        <div class="input-value-title">BMI</div>
        <div class="input-value-range">Normal: 18.5-24.9</div>
        <p>Body Mass Index = weight(kg)/height(m)². Higher BMI increases risk.</p>
    </div>
    """, unsafe_allow_html=True)
with cols[1]:
    st.markdown("""
    <div class="input-value-card">
        <div class="input-value-title">Total Cholesterol</div>
        <div class="input-value-range">Optimal: <200 mg/dL</div>
        <p>Total amount of cholesterol in your blood.</p>
    </div>
    """, unsafe_allow_html=True)
    st.markdown("""
    <div class="input-value-card">
        <div class="input-value-title">HDL Cholesterol</div>
        <div class="input-value-range">Optimal: ≥60 mg/dL</div>
        <p>"Good" cholesterol that helps remove LDL from arteries.</p>
```

```
        </div>
        """, unsafe_allow_html=True)
        st.markdown("""
        <div class="input-value-card">
            <div class="input-value-title">LDL Cholesterol</div>
            <div class="input-value-range">Optimal: <100 mg/dL</div>
            <p>"Bad" cholesterol that can build up in arteries.</p>
        </div>
        """, unsafe_allow_html=True)
        st.markdown("""
        <div class="input-value-card">
            <div class="input-value-title">Triglycerides</div>
            <div class="input-value-range">Normal: <150 mg/dL</div>
            <p>Type of fat in blood that can contribute to artery hardening.</p>
        </div>
        """, unsafe_allow_html=True)
# How the Model Works Section
with stylable_container(
    key="model_mechanics",
    css_styles=f"""
    {{
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
```

```
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
    }}
    """
):

    st.markdown("## How Our Model Predicts Risk")
    cols = st.columns([1, 2], gap="medium")
    with cols[0]:
        st.markdown("""
        <div style="text-align: center;" class="float">
            <img src="https://cdn-icons-png.flaticon.com/512/2103/2103633.png" width="100">
            <p style="font-size: 0.9rem; opacity: 0.8;">Machine Learning Process</p>
        </div>
        """, unsafe_allow_html=True)
    with cols[1]:
        st.markdown("""
        <div>
            <h4 style="margin-top: 0;">Advanced Predictive Analytics</h4>
            <p>
            Our model uses a <strong>Random Forest algorithm</strong> trained on thousands of clinical cases to identify patterns
            in cardiovascular health data. Here's how it works:
            </p>
```

```
            <ul>
                <li><strong>Data Collection:</strong> Aggregates your 7 key health metrics</li>
                <li><strong>Feature Scaling:</strong> Normalizes values for accurate comparison</li>
                <li><strong>Pattern Recognition:</strong> Compares your profile to known cases</li>
                <li><strong>Risk Calculation:</strong> Generates probability score (0-100%)</li>
            </ul>
        </div>
        """, unsafe_allow_html=True)

    st.markdown("""
    <div style="margin-top: 1.5rem;">
        <h4>Clinical Validation</h4>
        <p>
        The model was validated against real patient outcomes with:
        </p>
        <div style="display: grid; grid-template-columns: repeat(4, 1fr); gap: 1rem; margin: 1rem 0;">
            <div class="glass-card" style="padding: 1rem; text-align: center;">
                <div style="font-size: 1.5rem; font-weight: 700;">89.5%</div>
                <div style="font-size: 0.9rem;">Accuracy</div>
            </div>
            <div class="glass-card" style="padding: 1rem; text-align: center;">
                <div style="font-size: 1.5rem; font-weight: 700;">90.1%</div>
                <div style="font-size: 0.9rem;">Sensitivity</div>
```

```
            </div>
            <div class="glass-card" style="padding: 1rem; text-align: center;">
                <div style="font-size: 1.5rem; font-weight: 700;">88.2%</div>
                <div style="font-size: 0.9rem;">Specificity</div>
            </div>
            <div class="glass-card" style="padding: 1rem; text-align: center;">
                <div style="font-size: 1.5rem; font-weight: 700;">7</div>
                <div style="font-size: 0.9rem;">Key Factors</div>
            </div>
        </div>
    </div>
    """, unsafe_allow_html=True)

# Model Performance Section
with stylable_container(
    key="model_performance",
    css_styles=f"""
    {{
        background: rgba(30, 41, 59, 0.5);
        backdrop-filter: blur(16px);
        -webkit-backdrop-filter: blur(16px);
        border-radius: 16px;
        padding: 2rem;
        margin: 1.5rem 0;
```

```
    }}
    """
):
    st.markdown("## Model Performance Metrics")
    metrics = [
        ("Accuracy", "89.5%", "Measures overall correctness of predictions"),
        ("Precision", "88.2%", "Proportion of true positives among positive predictions"),
        ("Recall", "90.1%", "Ability to identify actual positive cases"),
        ("F1 Score", "89.1%", "Balanced measure of precision and recall")
    ]
    cols = st.columns(4, gap="medium")
    for i, (name, value, desc) in enumerate(metrics):
        with cols[i]:
            st.markdown(f"""
            <div class="glass-card" style="padding: 1rem; text-align: center; height: 100%;">
                <h3 style='margin-bottom: 0.5rem;'>{name}</h3>
                <div style='font-size: 1.5rem; font-weight: 700;'>
                    {value}
                </div>
                <div style='font-size: 0.9rem; opacity: 0.8;'>
                    {desc}
                </div>
            </div>
            """, unsafe_allow_html=True)
```

```
# Feature Importance
st.markdown("## Feature Importance")

features = ['Age', 'LDL', 'HDL', 'Triglycerides', 'BMI', 'Total Cholesterol', 'Gender']
importance = [0.25, 0.22, 0.18, 0.15, 0.10, 0.08, 0.02]
fig = go.Figure(go.Bar(
    x=importance,
    y=features,
    orientation='h',
    marker_color=theme_config["primary"],
    text=[f"{imp*100:.1f}%" for imp in importance],
    textposition='auto',
    textfont=dict(size=14)
))
fig.update_layout(
    height=400,
    xaxis_title="Relative Importance",
    plot_bgcolor='rgba(0,0,0,0)',
    paper_bgcolor='rgba(0,0,0,0)',
    font=dict(color=theme_config["text"]),
    margin=dict(l=100, r=50, b=50, t=50)
)
st.plotly_chart(fig, use_container_width=True)
# Model Comparison
```

```
        st.markdown("## Algorithm Comparison")
        comparison_data = {
            "Model": ["Random Forest", "Logistic Regression", "SVM", "Neural Network"],
            "Accuracy": ["89.5%", "82.1%", "85.3%", "87.2%"],
            "Clinical Utility": ["High", "Moderate", "Limited", "High"],
            "Explainability": ["Good", "Excellent", "Fair", "Poor"]
        }
        st.dataframe(
            pd.DataFrame(comparison_data).style.set_properties(**{
                'background-color': 'rgba(30, 41, 59, 0.5)',
                'color': theme_config["text"],
                'border': '1px solid rgba(255, 255, 255, 0.1)'
            }),
            use_container_width=True,
            hide_index=True
        )
        # [Previous imports remain exactly the same...]
# Add this new function after model_info_page()
def admin_dashboard():
    st.markdown("""
    <div class="page-entrance">
        <h1 class="gradient-text" style="margin-bottom: 1.5rem;">🛠️ Admin Dashboard</h1>
    """, unsafe_allow_html=True)
    # Security check
```

```
if not st.session_state.get('is_admin'):
    st.error(" Unauthorized access")
    return
# System Statistics
st.markdown("### System Overview")
cols = st.columns(4)
with cols[0]:
    st.metric("Total Users", len(db.get_all_users()))
with cols[1]:
    st.metric("Active Today", "N/A")  # Add actual metric
with cols[2]:
    st.metric("Risk Assessments", "N/A")
with cols[3]:
    if st.button("Create Backup"):
        if db.backup_database("backup.sql"):
            st.success("Backup created!")
        else:
            st.error("Backup failed")
# User Management
st.markdown("### User Accounts")
users = db.get_all_users()
for user in users:
    with st.container():
        cols = st.columns([3, 2, 1, 1, 1])
```

```
with cols[0]:
    st.write(f"**{user['username']}**")
with cols[1]:
    st.write(user['email'])
with cols[2]:
    admin_status = st.checkbox(
        "Admin",
        value=user.get('is_admin', False),
        key=f"admin_{user['user_id']}",
        disabled=(user['user_id'] == st.session_state.user_id)
    )
    if admin_status != user.get('is_admin', False):
        db.set_user_as_admin(user['user_id'], admin_status)
        st.rerun()
with cols[3]:
    if user['user_id'] != st.session_state.user_id:
        if st.button("Reset Password", key=f"pwd_{user['user_id']}"):
            # Implement password reset logic
            st.warning("Feature coming soon")
with cols[4]:
    if user['user_id'] != st.session_state.user_id:
        if st.button("Delete", key=f"del_{user['user_id']}"):
            if db.delete_user(user['user_id']):
                st.success("User deleted")
```

```
                    st.rerun()
                else:
                    st.error("Deletion failed")
    # Recent Activity
    st.markdown("### Recent Activity")
    stats = db.get_admin_stats()
    if stats:
        st.dataframe(pd.DataFrame(stats['recent_activity']))
    else:
        st.warning("No recent activity data available")
# --- Sidebar ---
with st.sidebar:
    if st.session_state.authenticated:
        st.markdown(f"""
        <div style='
            text-align: center;
            margin-bottom: 2rem;
            padding: 1.5rem;
            background: rgba(30, 41, 59, 0.5);
            backdrop-filter: blur(16px);
            -webkit-backdrop-filter: blur(16px);
            border-radius: 16px;
            border: 1px solid rgba(255, 255, 255, 0.1);
        ' class="page-entrance">
```

```
        <h1 style='margin-bottom: 0.5rem; font-size: 1.5rem;' class="gradient-text">Cardio-AI</h1>
        <p style='font-size: 0.9rem; color: var(--text); opacity: 0.8;'>
        Welcome, {st.session_state.username}
        </p>
    </div>
    """, unsafe_allow_html=True)
    # Dynamic navigation options based on user role
    if st.session_state.get('is_admin'):
        nav_options = ["🏠 Home", "📊 Risk Assessment", "🔬 Model Info", "👤 Profile", "👥 User Management"]
    else:
        nav_options = ["🏠 Home", "📊 Risk Assessment", "🔬 Model Info", "👤 Profile"]
    nav_option = st.radio(
        "Navigation",
        nav_options,
        index=nav_options.index("🏠 Home") if st.session_state.current_page == "home"
        else nav_options.index("📊 Risk Assessment") if st.session_state.current_page == "risk"
        else nav_options.index("🔬 Model Info") if st.session_state.current_page == "model"
        else nav_options.index("👤 Profile") if st.session_state.current_page == "profile"
        else nav_options.index("👥 User Management"),
        label_visibility="collapsed",
        key="nav_radio"
    )
    # Update current page based on selection
```

```
    if nav_option == "🏠 Home":
        st.session_state.current_page = "home"
    elif nav_option == "📊 Risk Assessment":
        st.session_state.current_page = "risk"
    elif nav_option == "🔬 Model Info":
        st.session_state.current_page = "model"
    elif nav_option == "👤 Profile":
        st.session_state.current_page = "profile"
    elif nav_option == "👥 User Management":
        st.session_state.current_page = "user_management"
    st.markdown("---")
    st.markdown("""
    <div class="glass-card" style="padding: 1rem; text-align: center;">
        <p style='font-size: 0.9rem; margin-bottom: 0.5rem;'>
        <strong>Clinical-grade prediction</strong><br>
        Validated with 89.5% accuracy
        </p>
    </div>
    """, unsafe_allow_html=True)
    if st.button("Logout", key="logout_button"):
        for key in list(st.session_state.keys()):
            del st.session_state[key]
        st.rerun()
else:
```

```
        st.markdown(f"""
        <div style='
            text-align: center;
            margin-bottom: 2rem;
            padding: 1.5rem;
            background: rgba(30, 41, 59, 0.5);
            backdrop-filter: blur(16px);
            -webkit-backdrop-filter: blur(16px);
            border-radius: 16px;
            border: 1px solid rgba(255, 255, 255, 0.1);
        ' class="page-entrance">
            <h1 style='margin-bottom: 0.5rem; font-size: 1.5rem;' class="gradient-text">Cardio-AI</h1>
            <p style='font-size: 0.9rem; color: var(--text); opacity: 0.8;'>
            AI-Powered Cardiovascular Risk Assessment
            </p>
        </div>
        """, unsafe_allow_html=True)
# Update the main app logic section to:
# --- Main App Logic ---
if not st.session_state.authenticated:
    login_page()
else:
    if st.session_state.current_page == "profile":
        patient_profile_page()
```

```
    elif st.session_state.current_page == "home":
        home_page()
    elif st.session_state.current_page == "risk":
        risk_assessment_page()
    elif st.session_state.current_page == "model":
        model_info_page()
    elif st.session_state.current_page == "user_management":
        admin_dashboard()  # Changed from user_management_page()
# --- Footer ---
st.markdown("---")
st.markdown(f"""
<div style='
    text-align: center;
    padding: 1.5rem;
    background: rgba(30, 41, 59, 0.5);
    backdrop-filter: blur(16px);
    -webkit-backdrop-filter: blur(16px);
    border-radius: 16px;
    border: 1px solid rgba(255, 255, 255, 0.1);
    margin-top: 2rem;
'>
    <p style='margin-bottom: 0.5rem; font-size: 0.9rem;'>
    <strong>Cardio-AI</strong> | Clinical Decision Support System
    </p>
```

```
    <p style='font-size: 0.8rem; opacity: 0.7;'>
    This tool does not replace professional medical advice. Always consult a healthcare provider.
    </p>
</div>
""", unsafe_allow_html=True)
```

: *DATABASE CODE*(database.py) :

```
# ============================================
# DATABASE CONNECTION AND MANAGEMENT CLASS
# ============================================
import mysql.connector
from mysql.connector import Error
import os
from dotenv import load_dotenv
import streamlit as st
import time
import hashlib
# Load environment variables
load_dotenv()
class DatabaseManager:
    # ============================================
    # INITIALIZATION
    # ============================================
    def __init__(self):
        self.host = os.getenv('DB_HOST', 'localhost')
        self.database = os.getenv('DB_NAME', 'cardio_ai')
        self.user = os.getenv('DB_USER', 'root')
        self.password = os.getenv('DB_PASSWORD', 'VcR@2005')
    # ============================================
    # DATABASE CONNECTION METHODS
    # ============================================
    def create_connection(self):
        """Create and return a database connection"""
        connection = None
        try:
```

```
        connection = mysql.connector.connect(
            host=self.host,
            user=self.user,
            passwd=self.password,
            database=self.database
        )
        return connection
    except Error as e:
        st.error(f"Error connecting to MySQL: {e}")
        return None
# ================================================
# USER MANAGEMENT METHODS
# ================================================
def create_user(self, username, email, password_hash):
    """Create a new user in the database"""
    connection = self.create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            query = """
            INSERT INTO users (username, email, password_hash)
            VALUES (%s, %s, %s)
            """
            cursor.execute(query, (username, email, password_hash))
            connection.commit()
            return cursor.lastrowid
        except Error as e:
            st.error(f"Error creating user: {e}")
            return None
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
def get_user_by_username(self, username):
    """Retrieve a user by username"""
    connection = self.create_connection()
    if connection:
        try:
            cursor = connection.cursor(dictionary=True)
            query = "SELECT * FROM users WHERE username = %s"
            cursor.execute(query, (username,))
            return cursor.fetchone()
        except Error as e:
            st.error(f"Error fetching user: {e}")
            return None
        finally:
            if connection.is_connected():
                cursor.close()
```

```
            connection.close()
def get_user_by_id(self, user_id):
    """Retrieve a user by user ID"""
    connection = self.create_connection()
    if connection:
        try:
            cursor = connection.cursor(dictionary=True)
            query = "SELECT * FROM users WHERE user_id = %s"
            cursor.execute(query, (user_id,))
            return cursor.fetchone()
        except Error as e:
            st.error(f"Error fetching user: {e}")
            return None
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
def delete_user(self, user_id):
    """Delete a user and all associated data from the database"""
    connection = self.create_connection()
    if connection:
        try:
            cursor = connection.cursor()

            # First get patient_id if exists
            cursor.execute(
                "SELECT patient_id FROM patients WHERE user_id = %s",
                (user_id,)
            )
            patient_id = cursor.fetchone()

            if patient_id:
                patient_id = patient_id[0]
                # Delete health records
                cursor.execute(
                    "DELETE FROM health_records WHERE patient_id = %s",
                    (patient_id,)
                )
                # Delete patient
                cursor.execute(
                    "DELETE FROM patients WHERE user_id = %s",
                    (user_id,)
                )

            # Finally delete the user
            cursor.execute(
                "DELETE FROM users WHERE user_id = %s",
                (user_id,)
```

```
            )
            connection.commit()
            return True
        except Error as e:
            st.error(f"Error deleting user: {e}")
            connection.rollback()
            return False
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
    # ============================================
    # PATIENT MANAGEMENT METHODS
    # ============================================
    def create_patient(self, user_id, full_name, date_of_birth, gender, contact_number):
        """Create a new patient profile"""
        connection = self.create_connection()
        if connection:
            try:
                cursor = connection.cursor()
                # Generate unique ID
                unique_id = f"PAT-{user_id}-{int(time.time())}"
                query = """
                INSERT INTO patients (user_id, unique_id, full_name, date_of_birth, gender,
contact_number)
                VALUES (%s, %s, %s, %s, %s, %s)
                """
                cursor.execute(query, (user_id, unique_id, full_name, date_of_birth, gender,
contact_number))
                connection.commit()
                return unique_id
            except Error as e:
                st.error(f"Error creating patient: {e}")
                return None
            finally:
                if connection.is_connected():
                    cursor.close()
                    connection.close()
    def get_patient_by_user(self, user_id):
        """Get patient profile by user ID"""
        connection = self.create_connection()
        if connection:
            try:
                cursor = connection.cursor(dictionary=True)
                query = "SELECT * FROM patients WHERE user_id = %s"
                cursor.execute(query, (user_id,))
                return cursor.fetchone()
            except Error as e:
```

```
            st.error(f"Error fetching patient: {e}")
            return None
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
    def update_patient(self, patient_id, full_name, date_of_birth, gender, contact_number):
        """Update patient profile information in the database"""
        connection = self.create_connection()
        if connection:
            try:
                cursor = connection.cursor()
                query = """
                UPDATE patients
                SET full_name = %s,
                    date_of_birth = %s,
                    gender = %s,
                    contact_number = %s
                WHERE patient_id = %s
                """
                cursor.execute(query, (
                    full_name,
                    date_of_birth,
                    gender,
                    contact_number,
                    patient_id
                ))
                connection.commit()
                return True
            except Error as e:
                st.error(f"Error updating patient: {e}")
                return False
            finally:
                if connection.is_connected():
                    cursor.close()
                    connection.close()
    # ================================================
    # HEALTH RECORD METHODS
    # ================================================
    def save_health_record(self, patient_id, input_data, risk_score, risk_category,
notes=None):
        """Save a health record to the database"""
        connection = self.create_connection()
        if connection:
            try:
                cursor = connection.cursor()
                query = """
                INSERT INTO health_records
```

```
            (patient_id, age, gender, bmi, chol, tg, hdl, ldl, risk_score, risk_category, notes)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(query, (
                patient_id,
                input_data['age'],
                input_data['gender'],
                input_data['bmi'],
                input_data['chol'],
                input_data['tg'],
                input_data['hdl'],
                input_data['ldl'],
                risk_score,
                risk_category,
                notes
            ))
            connection.commit()
            return cursor.lastrowid
        except Error as e:
            st.error(f"Error saving health record: {e}")
            return None
        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
    def get_patient_records(self, patient_id):
        """Get all health records for a patient"""
        connection = self.create_connection()
        if connection:
            try:
                cursor = connection.cursor(dictionary=True)
                query = """
                SELECT * FROM health_records
                WHERE patient_id = %s
                ORDER BY created_at DESC
                """
                cursor.execute(query, (patient_id,))
                return cursor.fetchall()
            except Error as e:
                st.error(f"Error fetching records: {e}")
                return None
            finally:
                if connection.is_connected():
                    cursor.close()
            connection.close()
```

8.3 DATA DICTIONARY

The data dictionary provides descriptions for all health-related data fields and their possible values:

- Age: Numeric value representing the user's age (1–120).
- Gender: Categorical variable (Male = 1, Female = 0).
- BMI: Body Mass Index, numeric value (10.0–50.0).
- Cholesterol Level: Numeric value representing cholesterol levels (100–300).
- Triglycerides: Numeric value for triglycerides level (50–500).
- HDL Cholesterol: Numeric value for High-Density Lipoprotein cholesterol (20–100).
- LDL Cholesterol: Numeric value for Low-Density Lipoprotein cholesterol (50–250).

This dictionary helps in understanding the data used in the model, ensuring consistency in data entry and interpretation.

CHAPTER-9

9 BIBLIOGRAPHY AND REFERENCES

- **Breiman, L. (2001). "Random Forests." Machine Learning Journal.**

  Fundamental reference for the Random Forest algorithm, which is used in Cardio-AI for heart disease prediction.

- **Framingham Heart Study (2020).**

  A widely recognized study on cardiovascular disease risk factors, serving as the basis for selecting key health parameters in the model.

- **Dhar, P., et al. (2019). "Machine Learning Approaches in Heart Disease Prediction." IEEE Access.**

  Research exploring different ML techniques for heart disease prediction, aiding in model selection and optimization.

- **Müller, A. C., & Guido, S. (2016). "Introduction to Machine Learning with Python." O'Reilly Media.**

  A comprehensive guide on implementing machine learning models, including Scikit-learn, which is used in Cardio-AI.

- **American Heart Association (2022). "Cardiovascular Risk Assessment and Prevention."**

  Provides clinical guidelines on heart disease risk factors, helping validate prediction results and recommendations.

- **HIPAA (Health Insurance Portability and Accountability Act) & GDPR (General Data Protection Regulation).**

  Ensures compliance with data privacy laws for securely handling patient health data within the system.

- **Python, Scikit-learn, MySQL, Streamlit (2024).**

  Core technologies used in the development of Cardio-AI for data processing, model deployment, database management, and user interface.