# Progressive Education Society’s

**MODERN COLLEGE OF COMMERCE & COMPUTER**

**STUDIES NIGDI PUNE- 44**



A PROJECT REPORT ON

"**VITALVISION**"

## SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF

***BBA(CA)***

**Savitribai Phule University of Pune**

**SUBMITTED BY**

**“Palavi Raut”**

## GUIDED BY

**Dr. Prasanna Chavan**

### ACADEMIC YEAR 2024 -25 (VI)



# MODERN COLLEGE OF COMMERCE AND COMPUTER STUDIES

**NIGDI, PUNE – 411044**

**DEPARTMENT OF BBA (CA)**

**2024-25**

# CERTIFICATE

This is to certify that the project report titled **“VITALVISION”** has been successfully completed by **Palavi Raut**, a student of the sixth semester of the **BBA (CA) degree program**. This report has been prepared in fulfilments of the requirements for the award of the **BBA (CA) degree** and has been submitted to the **Department of Commerce, Savitribai Phule University**. The project work was carried out during the academic year **2024-25**, as per the prescribed curriculum.

**Project Guide Principal**

**Internal Examiner External Examiner**

## DECLARATION

I hereby declare that this project work entitled "**VITALVISION**" submitted at PES's Modern College of Commerce and Computer Studies Nigdi, (Affiliated to SPPU) is a record of the original work done by me under the supervision and guidance of **Dr. Prasanna Chavan ,** Department of **Bachelor of Business Administration (Computer Application.)**

This work has not been previously submitted to any other University/College for any examination.

I have taken due care to acknowledge all sources of information, including books, journals, websites, and other materials, used in the preparation of this project.

**Name of Student:**

**Signatures:**

**Date:**

**Place:**

## ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my guide, **Dr. Prasanna Chavan ,** for providing me with the valuable opportunity to undertake this project on

**“VITALVISION”.** Their continuous support, encouragement, and expert guidance have been instrumental in shaping my understanding of the subject matter. Throughout the project, their insights and constructive feedback helped me gain deeper knowledge and develop essential skills. I am sincerely thankful for their patience, time, and efforts in mentoring me, which have contributed significantly to the successful completion of this project.

I would like to express my deepest appreciation towards **Dr. Sadashiv Shirgave Sir**, Principal of Modern College of Commerce and Computer Studies, Nigdi, Pune – 411044, for his invaluable support and encouragement.

Lastly, I extend my sincere thanks to all the staff members, my family, and friends who have supported and helped me directly in making this project a success

**Name: Palavi Raut**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.NO** | **CONTENT** | **PAGENO** |
| **1** | **INTRODUCTION** | 6-8 |
|  | 1.1 Abstract | 6 |
|  | 1.2 Introduction | 6 |
|  | 1.3 Need for System | 6 |
|  | 1.4 Scope of Work | 7 |
|  | 1.4 Operating Environment-Hardware and Software | 7-8 |
| **2** | **PROPOSED SYSTEM** | 9-10 |
|  | 2.1 Proposed System | 9 |
|  | 2.2 Objectives of System | 9 |
|  | 2.3 User Requirements | 9 - 10 |
| **3** | **SYSTEM ANALYSIS AND DESIGN** | 11-14 |
|  | 3.1 Entity Relation Diagram | 11 |
|  | 3.2 Class Diagram | 12 |
|  | 3.3 Object Diagram | 13 |
|  | 3.4 Activity Diagram | 14 |
|  | 3.5 Use Case Diagram | 14 |
| **4** | **USER MANUAL** |  |
|  | 4.1 User Manual | 17 |
|  | 4.2 Operations Manual/Menu Explanation | 18 |
|  | 4.3 Program Specifications/Flow Charts | 19-20 |
| **5** | **ADVANTAGES AND LIMITATIONS** | 21 |
| **6** | **CONCLUSION** | 22 |
| **7** | **ANNEXURE** | 23 |
|  | **ANNEXURE 1:** User Interface Screens | 23 |
|  | **ANNEXURE 2:** Output Reports with Data | 24-30 |
|  | **ANNEXURE 3:** Sample Program Code | 31-49 |
| **8** | **FUTURE SCOPE** | 50 |
| **9** | **BIBLIOGRAPHY AND REFERENCES** |  |

**INTRODUCTION**

**1.1 Abstract**

Healthcare accessibility and management remain significant challenges, especially in a vast and diverse country like India. Patients often struggle with timely consultations, misplaced medical records, and long travel distances for basic healthcare services. Additionally, healthcare providers face inefficiencies due to a lack of digital record-keeping tools and limited opportunities for medical interns.

VitalVision is a comprehensive web platform designed to address these issues by integrating AI-powered diagnostics, online consultations, structured medical record management, and QR code-enabled patient profiles. This project aims to bridge the gaps in healthcare accessibility and efficiency by offering an all-in-one digital solution.

**1.2 Introduction**

India's healthcare landscape faces numerous challenges, ranging from limited accessibility to inefficient patient data management. Patients frequently rely on unverified online sources for medical advice, leading to misdiagnoses and delayed treatments. Many individuals misplace their medical records, resulting in repeated tests and unnecessary expenses. Furthermore, rural patients often travel long distances for consultations, which could otherwise be conducted virtually. Healthcare providers, on the other hand, lack digital tools to streamline patient data management, leading to fragmented medical histories and inefficient practices. Medical interns also struggle to gain hands-on experience due to limited patient interaction opportunities. VitalVision is designed to mitigate these challenges by leveraging modern technology to enhance healthcare accessibility, diagnosis accuracy, and record management efficiency.

**1.3 Need for System**

The need for VitalVision arises from the growing demand for a structured and efficient healthcare management system. The traditional healthcare model has several inefficiencies, such as:

* Patients facing difficulties in booking timely appointments.
* Misplacement of crucial health records leading to redundant tests and increased costs.
* Challenges for chronic patients in maintaining a continuous medical history.
* Limited opportunities for medical interns to interact with patients and gain practical exposure.
* Lack of digital tools for healthcare providers to maintain patient records and streamline treatment plans. By addressing these critical pain points, VitalVision provides an integrated platform that enhances healthcare accessibility and streamlines patient management through digital solutions.

**1.4 Scope of Work**

VitalVision aims to provide a centralized platform that benefits both patients and healthcare providers. The primary features and scope of the project include:

* **Online Consultations:** Patients, especially from rural areas, can consult with healthcare providers remotely.
* **Medical Record Management:** Patients will have unique profile cards with QR codes, ensuring easy access to medical histories.
* **AI-Assisted Diagnosis:** AI tools will help analyze patient data for early detection of diseases.
* **Support for Medical Interns:** The platform offers structured engagement for PG medical interns to gain hands-on experience.
* **Secure and Scalable System:** The platform will be developed using scalable technologies like React.js for frontend, Node.js and Flask for backend, and MongoDB for data management.

**1.5 Operating Environment - Hardware and Software**

To ensure the efficient functioning of the VitalVision platform, the following hardware and software requirements must be met:

**Hardware Requirements**

* **Client Side:**
  + Minimum 4GB RAM, recommended 8GB+ RAM
  + Minimum 2GHz processor, recommended i5/i7
  + Internet connection for online consultations
* **Server Side:**
  + Cloud hosting or dedicated server with scalable resources
  + Minimum 8GB RAM, recommended 16GB+ RAM
  + Database storage for medical records

**Software Requirements**

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** Node.js, Express.js, Flask (Python)
* **Database:** MongoDB
* **Additional Tools:** AI/ML models for diagnosis
* **Operating System:** Windows 10
* **Development Tools:** VS Code, MySQL Workbench (for additional data analysis)

VitalVision is designed to be a scalable and user-friendly platform that enhances healthcare accessibility, record management, and medical diagnostics. The system will provide an efficient and structured approach to handling healthcare challenges while integrating modern technological advancements.

**PROPOSED SYSTEM**

**2.1 Proposed System**

The **VitalVision** platform is a healthcare accessibility and management system designed to bridge the gap between patients, healthcare providers, and medical interns. It leverages modern technology to offer online consultations, AI-assisted diagnosis, and digital medical record management to enhance patient care.

**Key Features:**

1. **Online Consultations** - Patients, especially those in rural areas, can connect with doctors remotely.
2. **Medical Record Management** - Patients receive unique profile cards with QR codes linked to their health history.
3. **AI-Powered Diagnosis** - AI/ML models analyze patient symptoms for early disease detection.
4. **Healthcare Provider Support** - A structured platform for medical interns to engage with real patient cases.
5. **All-in-One System** - Combines consultations, diagnostics, and record management in a seamless interface.

**2.2 Objectives of the System**

1. **Improve Healthcare Accessibility** - Enable remote consultations, reducing the need for travel.
2. **Enhance Diagnosis Efficiency** - Implement AI to assist in early disease detection.
3. **Streamline Patient Record Management** - Provide a digital repository for medical histories to avoid loss of records.
4. **Facilitate Medical Intern Training** - Offer hands-on experience in a controlled environment.
5. **Reduce Healthcare Costs** - Minimize redundant medical tests by centralizing records.
6. **Ensure Data Security & Privacy** - Secure patient data with encryption and access control measures.

**2.3 User Requirements**

**Patient Requirements:**

* Easy-to-use web platform for booking consultations.
* Secure access to personal medical records via QR codes.
* AI-powered preliminary diagnostics for better health awareness.

**Healthcare Provider Requirements:**

* A structured patient management system.
* Secure access to patient histories for better diagnosis.
* Dashboard for consultation scheduling and tracking.

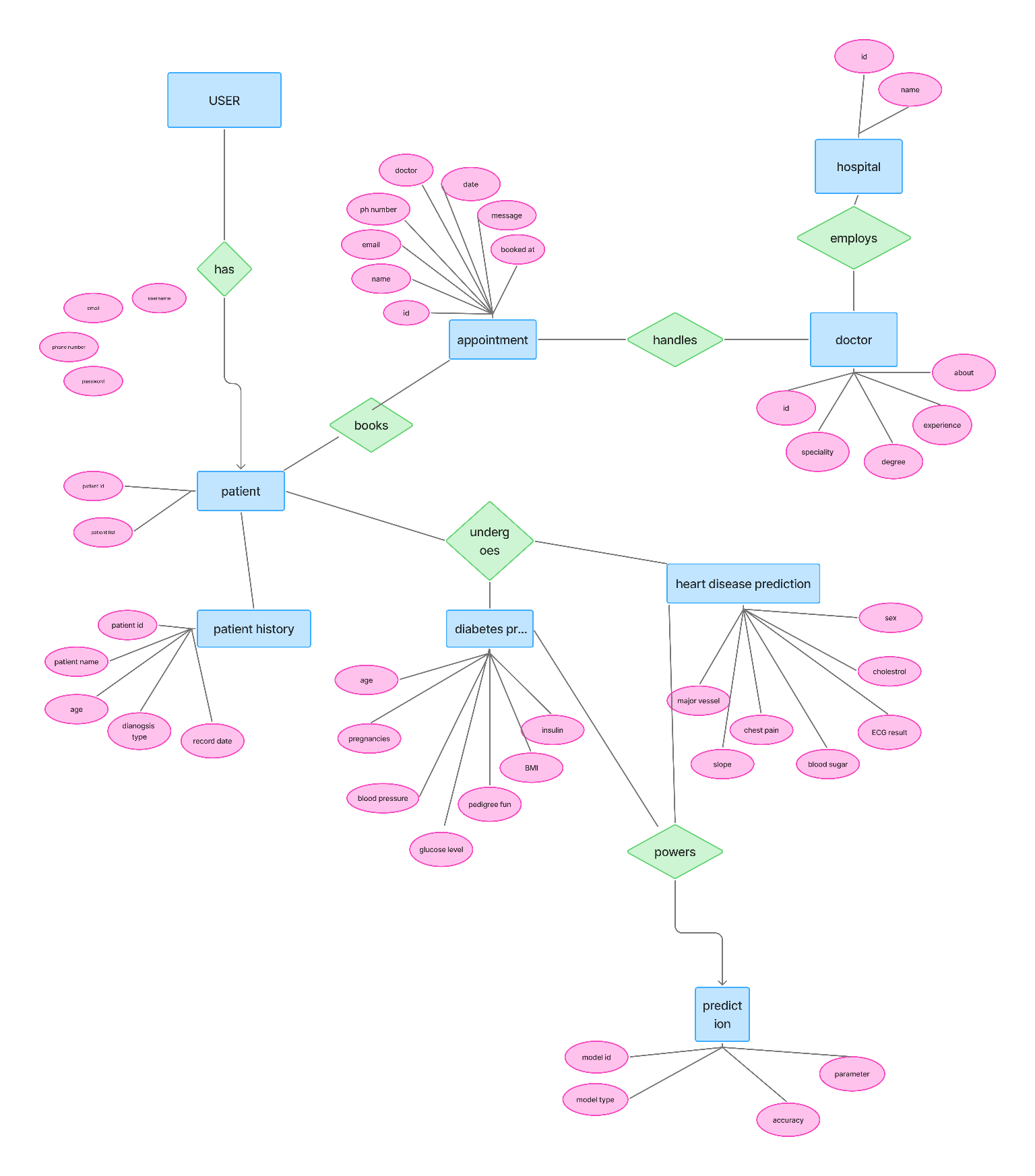
**Medical Intern Requirements:**

* Platform for virtual patient interactions.
* AI tools for guided learning in diagnostics.
* Secure logging and monitoring of consultation cases.

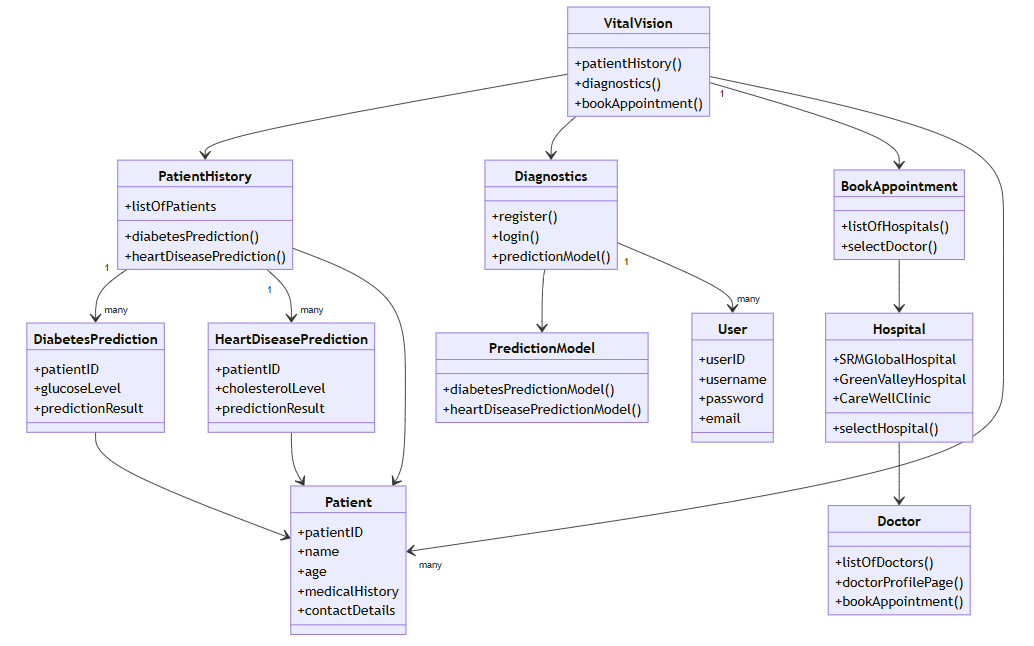
With this structured approach, **VitalVision** ensures seamless healthcare accessibility, better diagnostics, and optimized patient record management.

**SYSTEM ANALYSIS AND DESIGN**

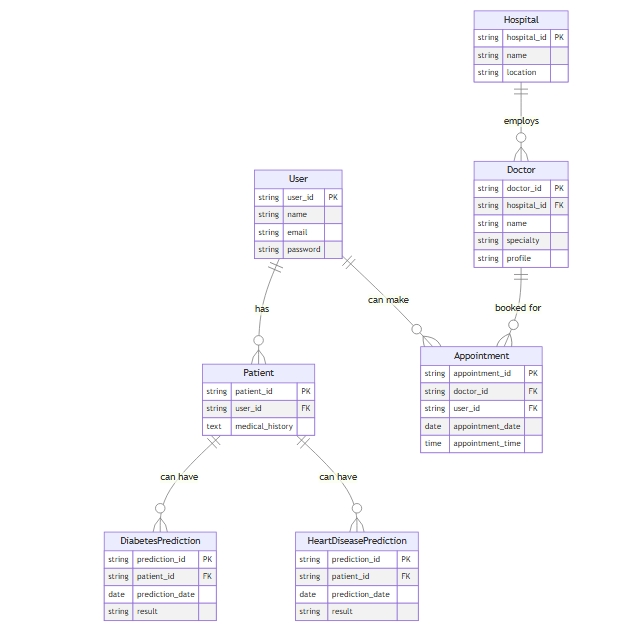
**3.1 Entity Relation Diagram**



**3.2 Class Diagram**



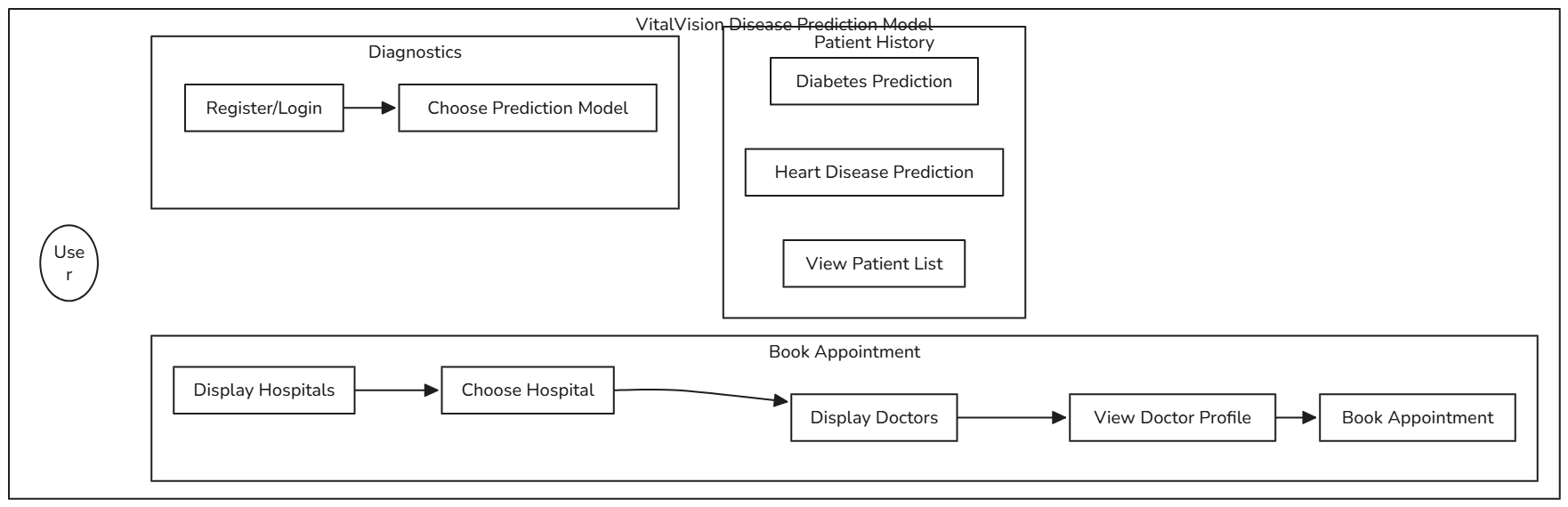
**3.3 Object diagram**

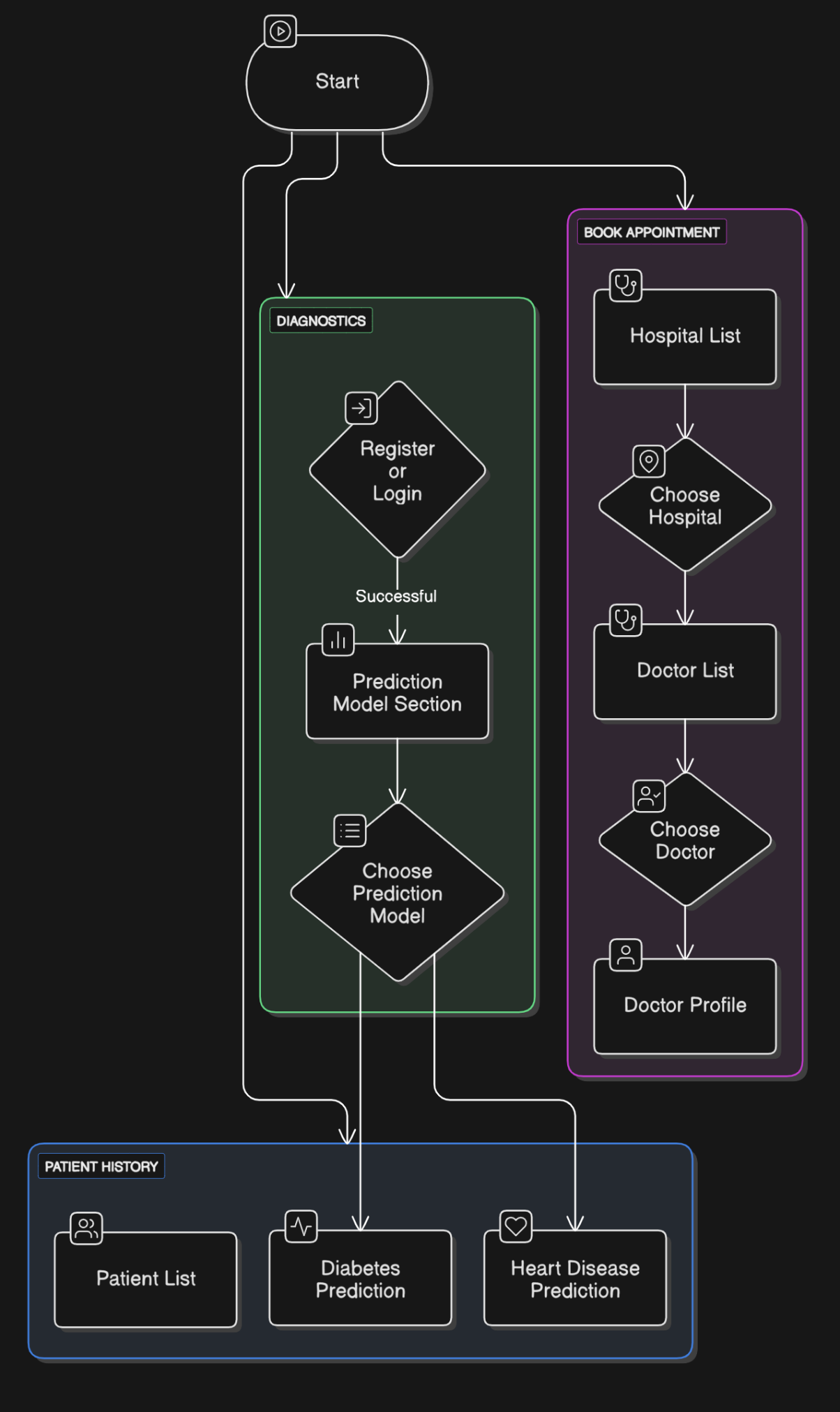


**3.4 Activity diagram**

****

**3.5 Use case diagram**



**3.6 Component diagram**

**3.7 Deployment diagram** 

**USER MANUAL**

The user manual provides a step-by-step guide for using the **VitalVision** platform, ensuring that both patients and healthcare providers can navigate and utilize its features efficiently.

**4.1 User Manual**

VitalVision is a web-based healthcare management platform that helps patients schedule appointments, manage health records, and receive AI-assisted diagnoses. Healthcare providers can manage patient records and offer virtual consultations.

**Steps for Patients**

1. **Registration & Login**
   * Visit the homepage and click on **"Register"** if you are a new user.
   * Enter details like name, contact, email, and password.
   * Click **"Submit"** to create an account.
   * If already registered, click **"Login"** and enter your credentials.
2. **Booking an Appointment**
   * Go to the **"Book Appointment"** section.
   * Select a doctor based on specialization.
   * Choose a date and time slot.
   * Click **"Confirm Appointment"** to finalize the booking.
3. **Accessing Medical Records**
   * View and upload health reports in the **"My Health Records"** section.
   * Download past reports using the **QR code feature** for easy sharing with doctors.
4. **AI-Assisted Diagnosis**
   * Input symptoms into the AI tool.
   * Receive preliminary suggestions for potential health conditions.
   * Book a consultation for further medical guidance.

**Steps for Healthcare Providers**

1. **Doctor Registration & Profile Setup**
   * Register using medical credentials.
   * Upload certifications and practice details.
   * Set availability for patient appointments.
2. **Managing Patient Records**
   * View assigned patients and their uploaded medical history.
   * Update treatment plans, prescriptions, and diagnosis reports.
3. **Providing Virtual Consultations**
   * Accept or decline appointment requests.
   * Conduct online video/audio consultations.
   * Provide digital prescriptions post-consultation.

**4.2 Operations Manual / Menu Explanation**

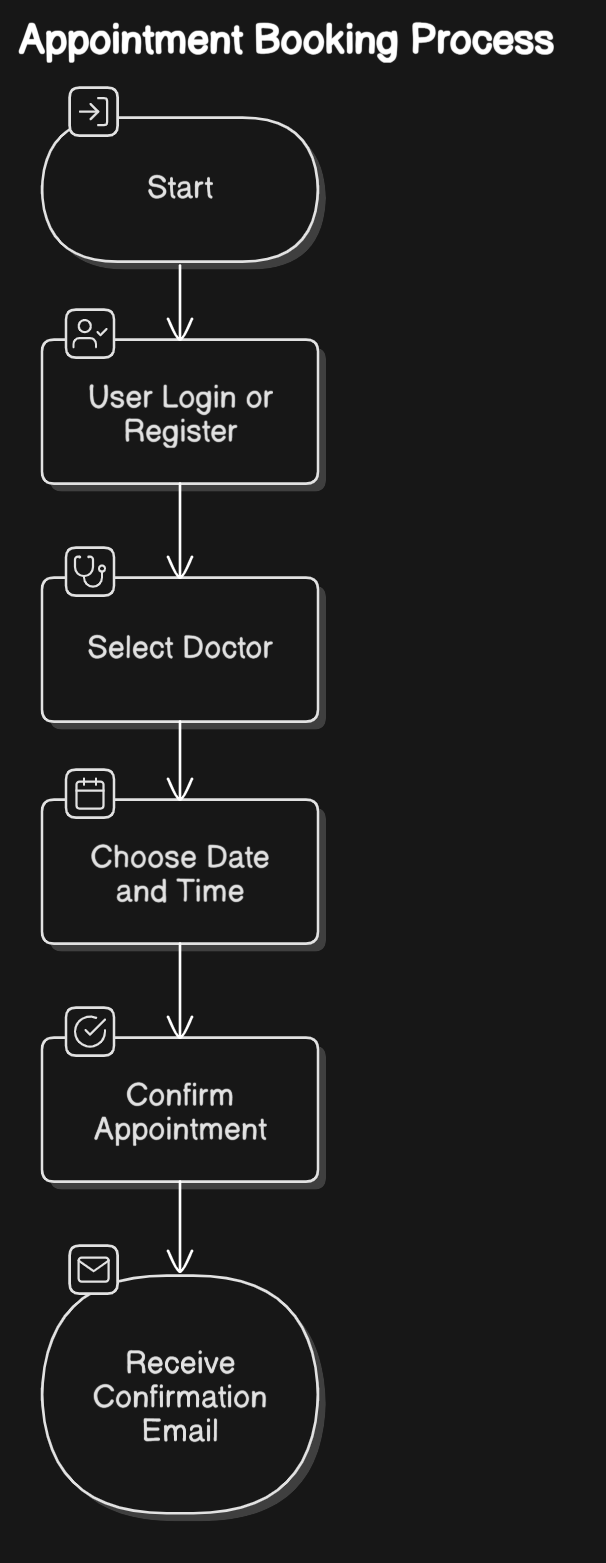
The **VitalVision** platform consists of the following key menus:

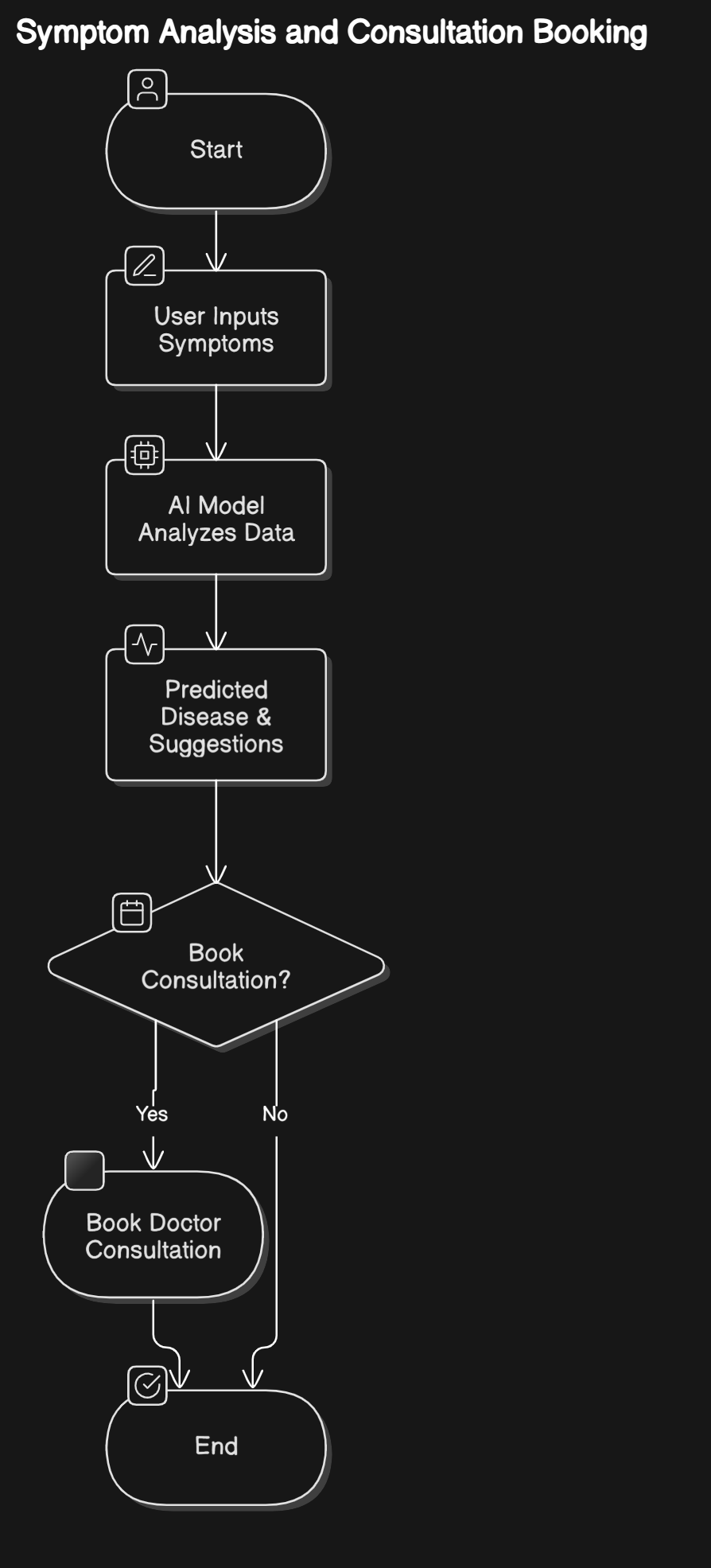
1. **Home Page:** Provides an overview of services, including appointment booking, record management, and AI-assisted diagnosis.
2. **Register/Login Page:** Allows users to create an account and sign in.
3. **Dashboard:** Displays personalized user options based on role (patient/doctor).
4. **Book Appointment:** Patients can schedule consultations with doctors.
5. **My Health Records:** A section for storing, viewing, and sharing medical documents.
6. **AI Diagnosis:** Allows users to enter symptoms and receive AI-generated health insights.
7. **Contact Us:** A feedback and support section for queries and assistance.

**4.3 Program Specifications / Flow Charts**

**Program Specifications**

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** Node.js, Flask (for AI models)
* **Database:** MongoDB for storing user and medical records
* **APIs:** Integration with third-party services for AI

****



**ADVANTAGES AND LIMITATIONS**

**Advantages**

✅ **Remote Healthcare Accessibility:** Patients can book virtual consultations, reducing the need for travel.  
✅ **Efficient Health Record Management:** QR-based medical history prevents data loss.  
✅ **AI-Assisted Diagnosis:** Provides early detection of diseases, reducing treatment delays.  
✅ **Streamlined Appointment Process:** Easy scheduling and tracking of doctor visits.  
✅ **Enhanced Learning for Medical Interns:** A platform for gaining real-world consultation experience.

**Limitations**

⚠️ **Internet Dependency:** Requires a stable internet connection for accessing services.  
⚠️ **Limited Physical Examination:** Virtual consultations cannot replace physical checkups.  
⚠️ **Data Security Risks:** Sensitive medical data needs robust security measures.  
⚠️ **Initial Adoption Barrier:** Patients and doctors unfamiliar with technology may face challenges.

**CONCLUSION**

VitalVision is a comprehensive digital healthcare platform that aims to bridge the gap between patients and healthcare providers by integrating AI-driven diagnostics, online consultations, and an advanced medical record management system. It effectively addresses major challenges in the Indian healthcare system, such as delayed diagnoses, fragmented medical records, and accessibility issues for both patients and medical professionals.

By leveraging AI and machine learning, the platform not only helps in early disease detection but also enhances the accuracy of diagnoses, reducing reliance on self-diagnosis from unverified sources. The QR-code-based health record management system ensures that patient histories are easily accessible, reducing unnecessary medical tests and improving continuity of care.

For healthcare providers, especially PG medical interns, VitalVision serves as a learning and experience-building platform, enabling them to interact with real patients, enhancing their diagnostic skills and medical expertise. The platform's seamless UI/UX, efficient database management, and robust backend infrastructure make it scalable, secure, and user-friendly.

With the rapid advancement of technology in healthcare, VitalVision has the potential to revolutionize digital health services, making quality healthcare more affordable, efficient, and accessible to even the most remote regions. Its impact extends beyond urban hospitals, reaching rural communities where medical facilities are limited, ultimately contributing to a healthier society with better patient outcomes.

As technology continues to evolve, VitalVision will further expand its features, integrating IoT-based wearable health monitoring, blockchain for secure data storage, and advanced AI models for predictive healthcare analytics. This continuous evolution will ensure that the platform remains at the forefront of digital healthcare transformation, helping countless individuals and medical professionals improve their health management and service delivery.

**ANNEXURE**

**ANNEXURE 1:** User Interface Screens

Home Page

Overview of the platform

Navigation to different sections (Appointments, Medical Records, Consultations)

User Registration & Login

Registration with personal details

Patient Dashboard

View & book doctor appointments

Access and upload medical history (health records)

Doctor Dashboard

Intern Dashboard

Appointment Booking Page

AI-Assisted Diagnosis Page

Patients input symptoms

AI predicts possible health issues and suggests specialists

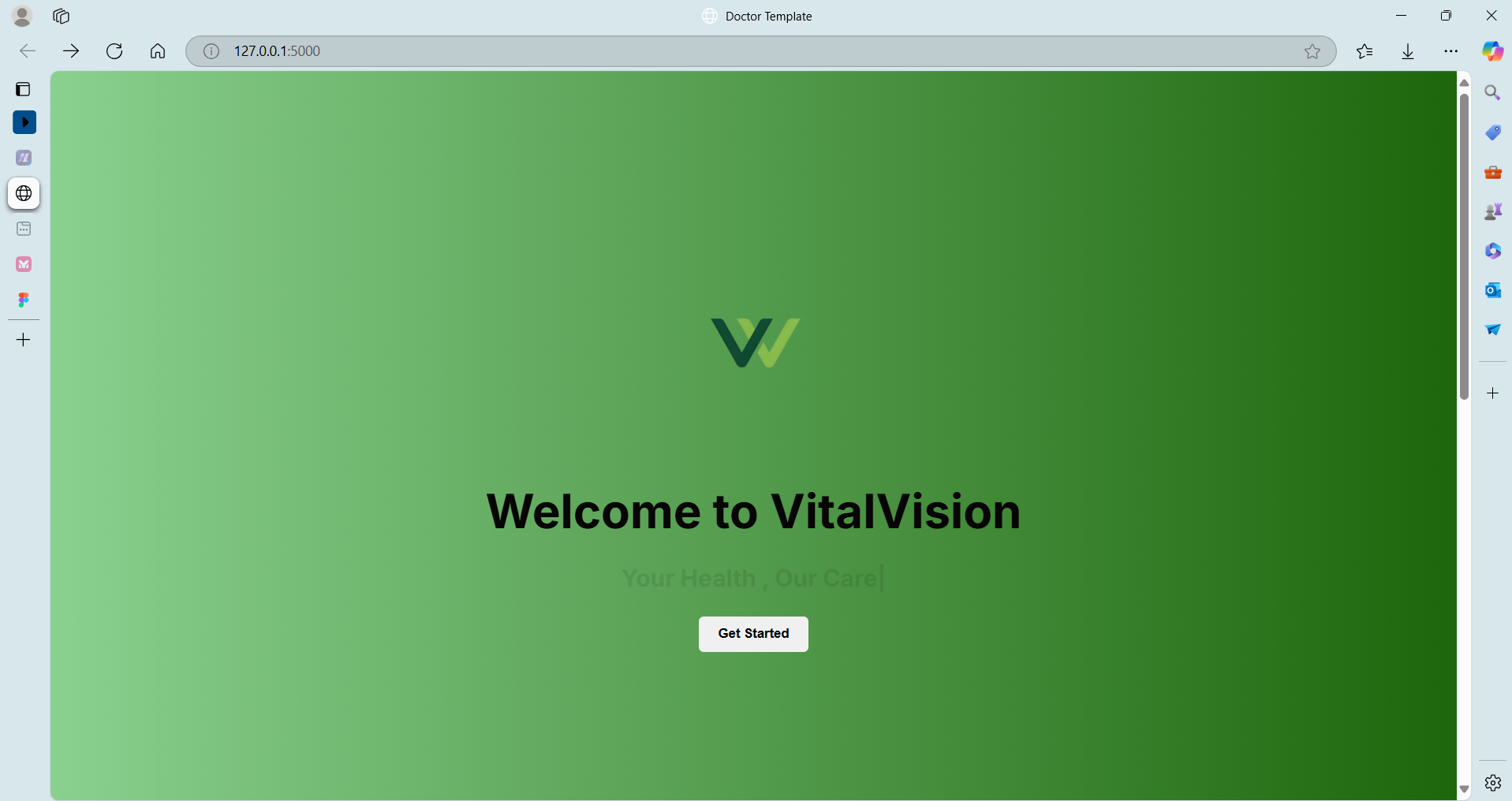
Consultation Page (Virtual Consultation)

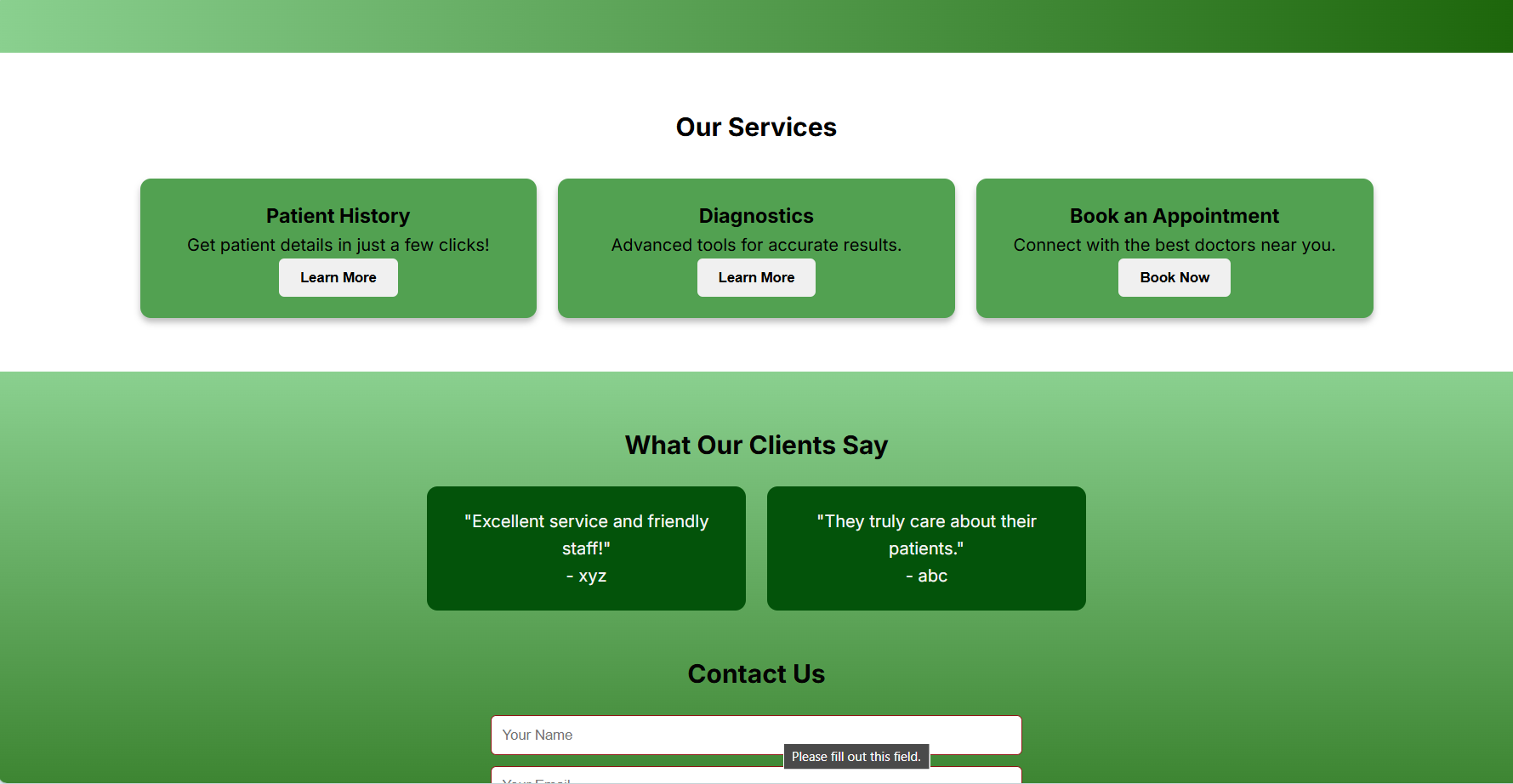
Video call or chat-based consultation with doctors

Monitor system activity and analytics

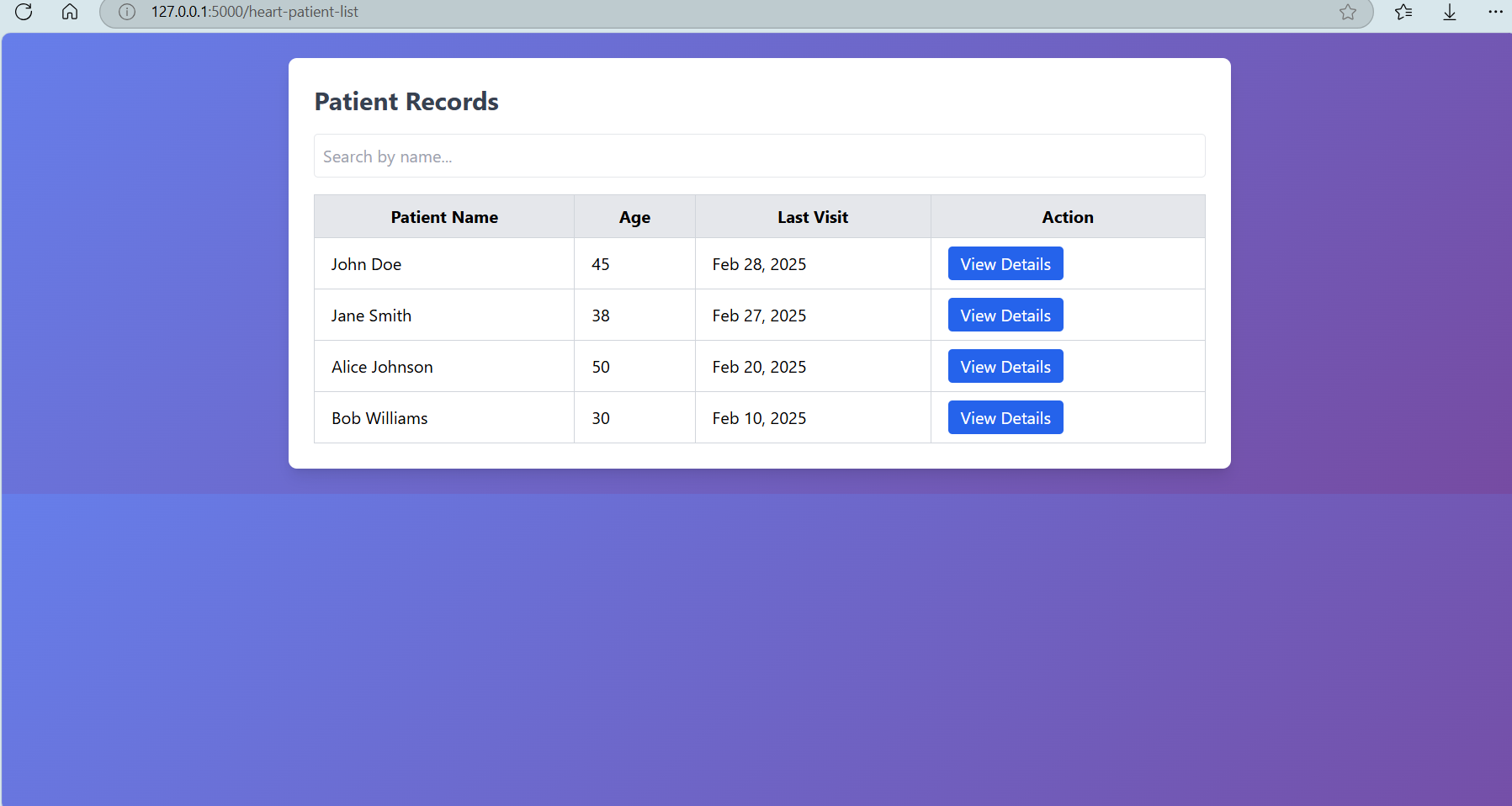
Contact Us

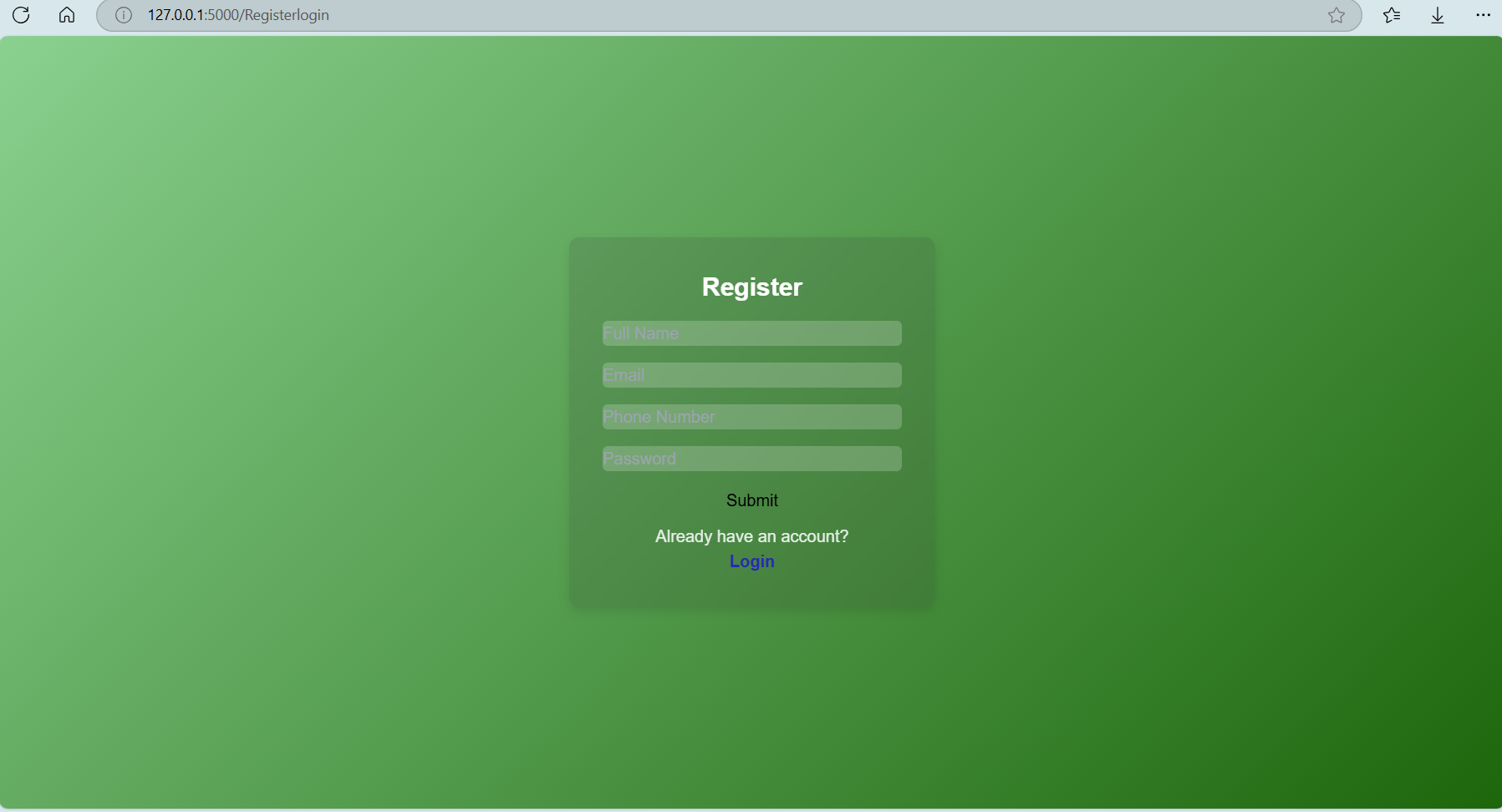
**ANNEXURE 2:** Output Reports with Data

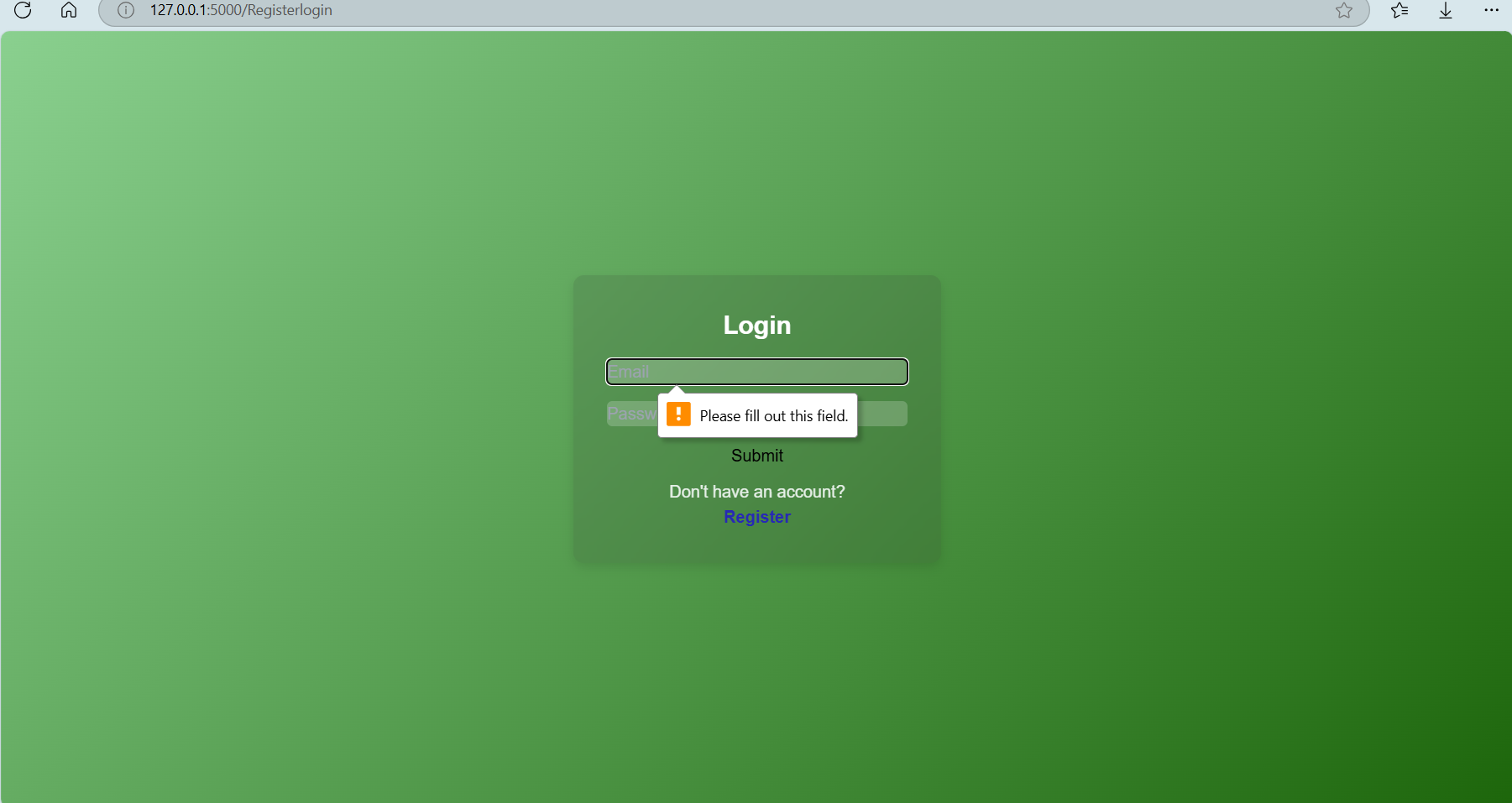


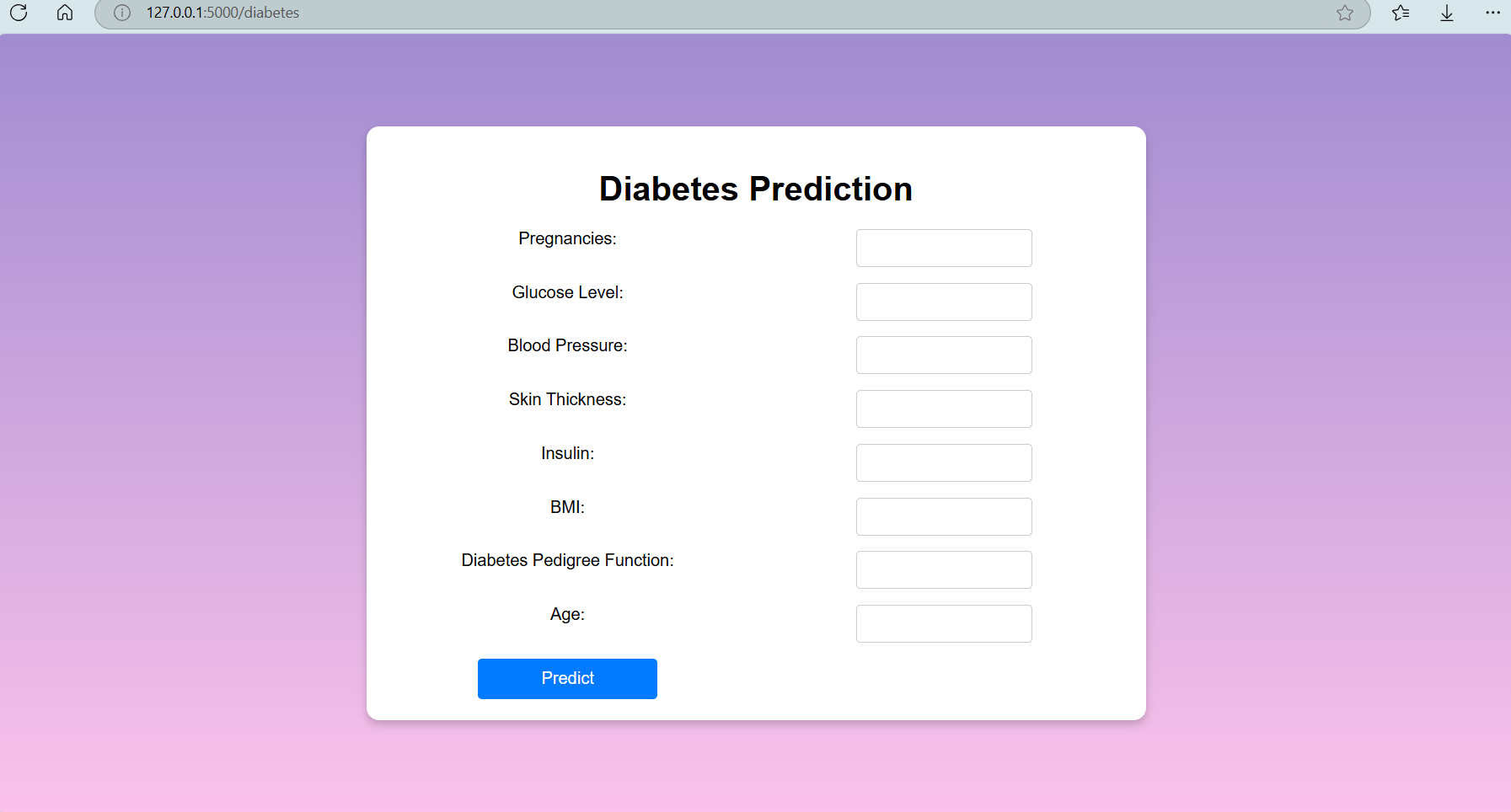


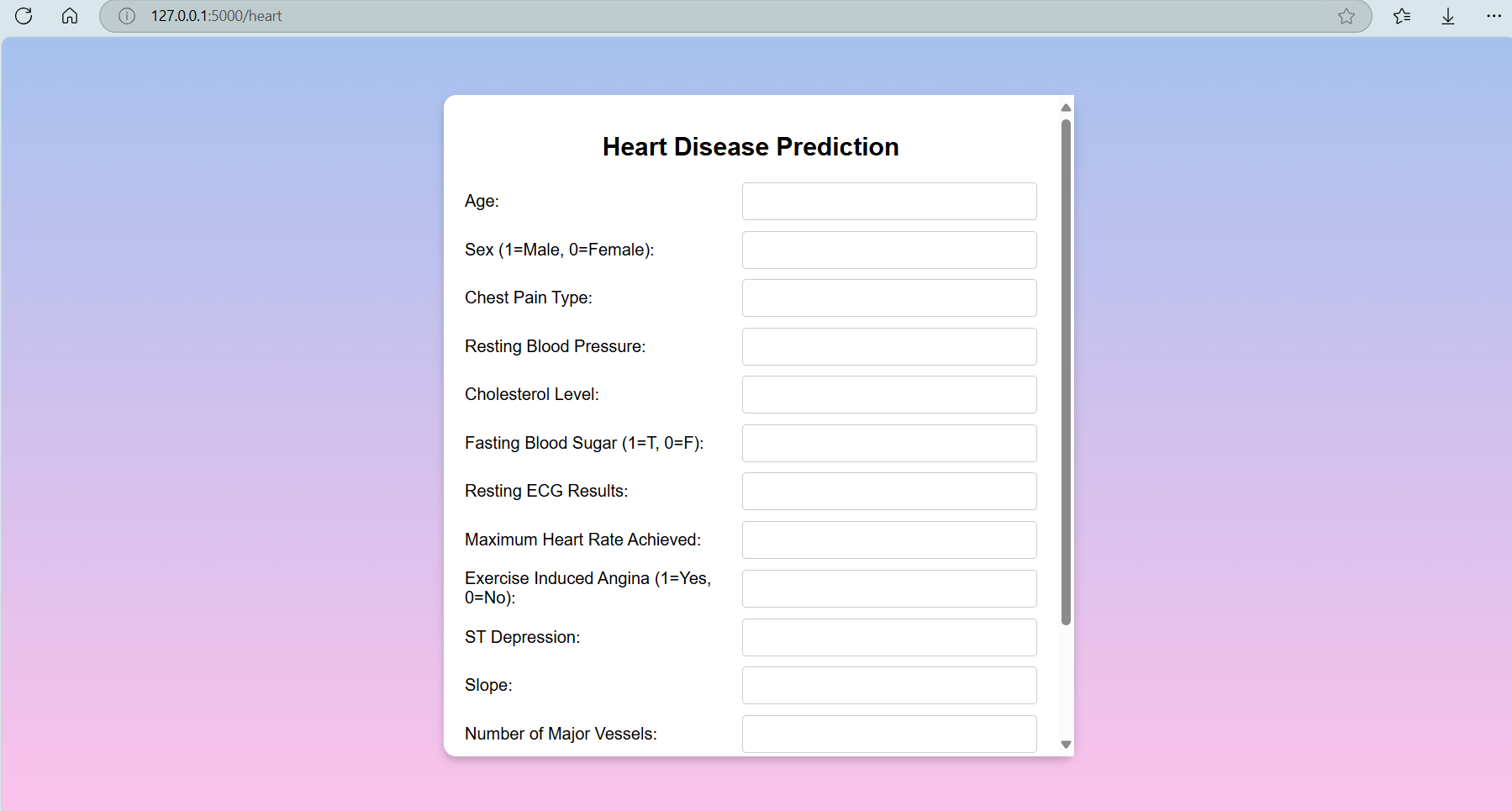


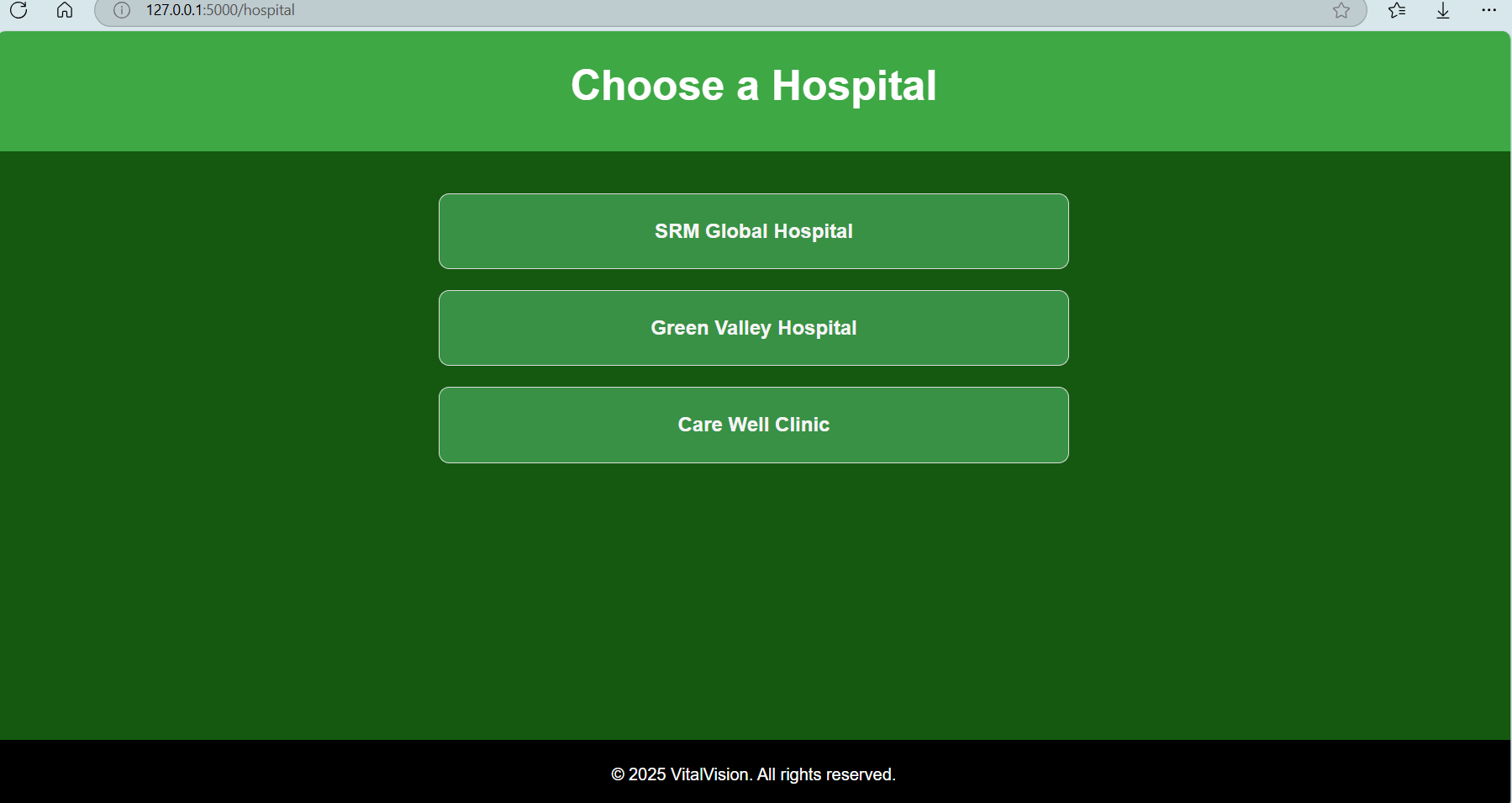




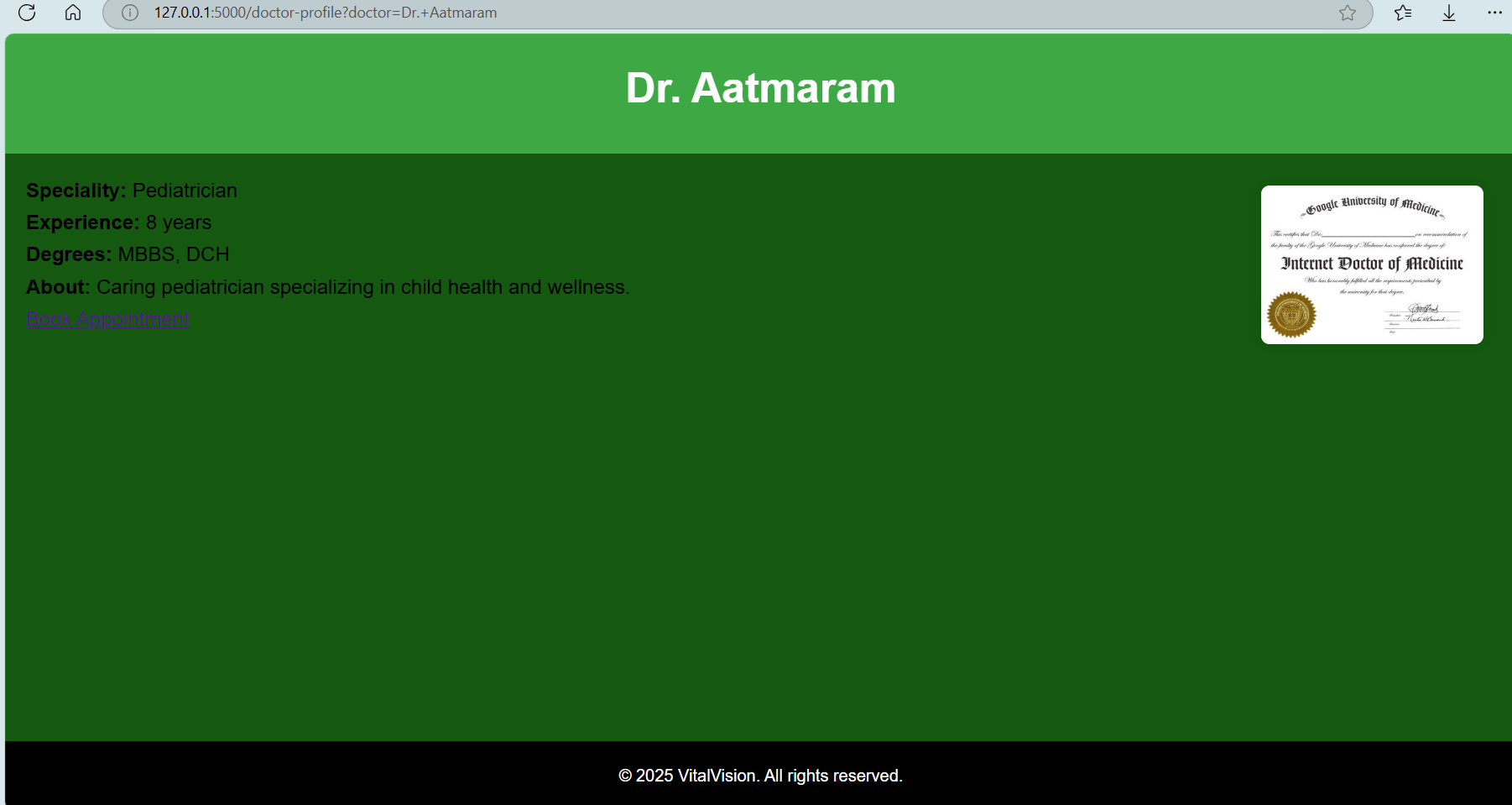


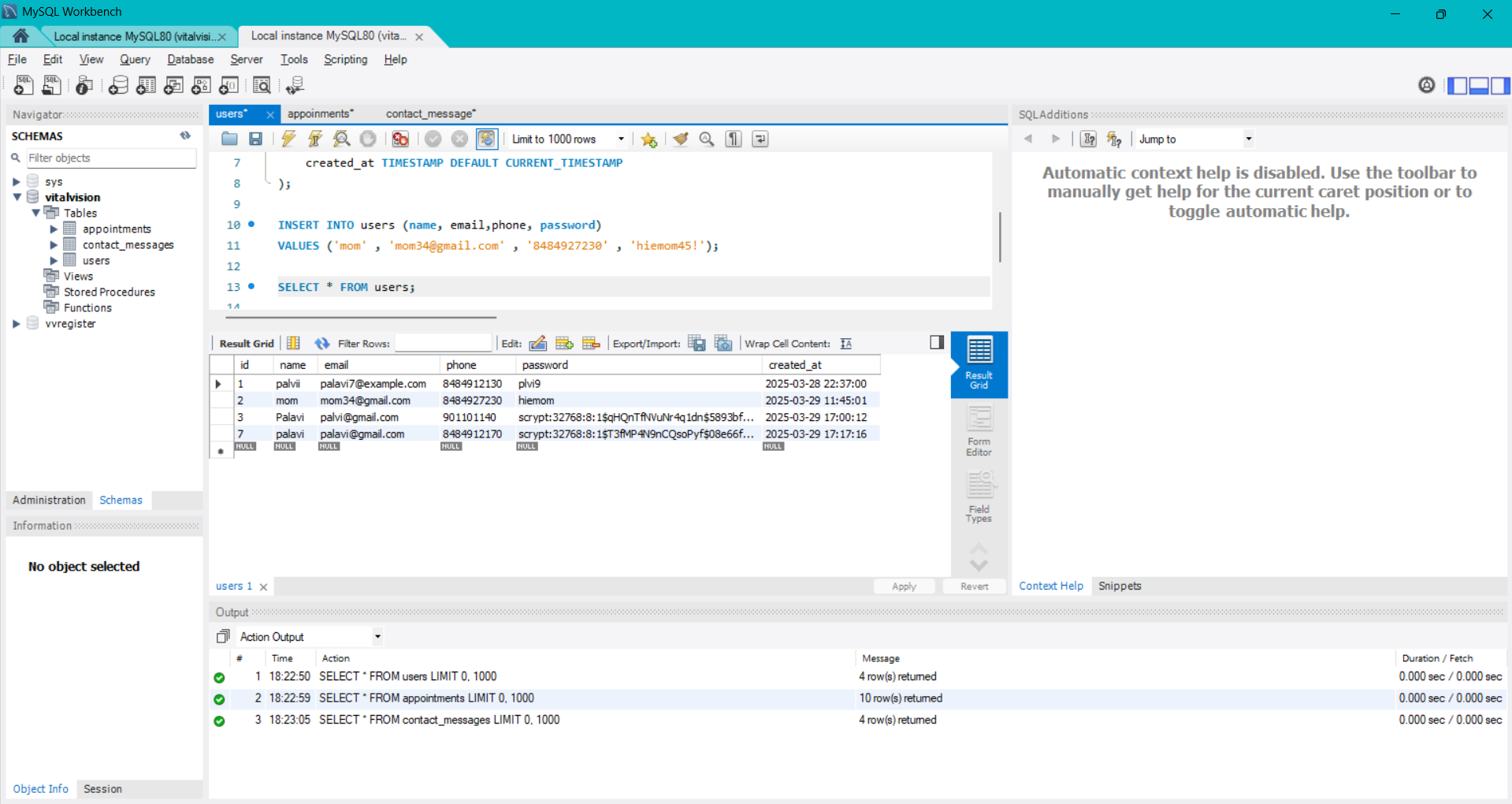


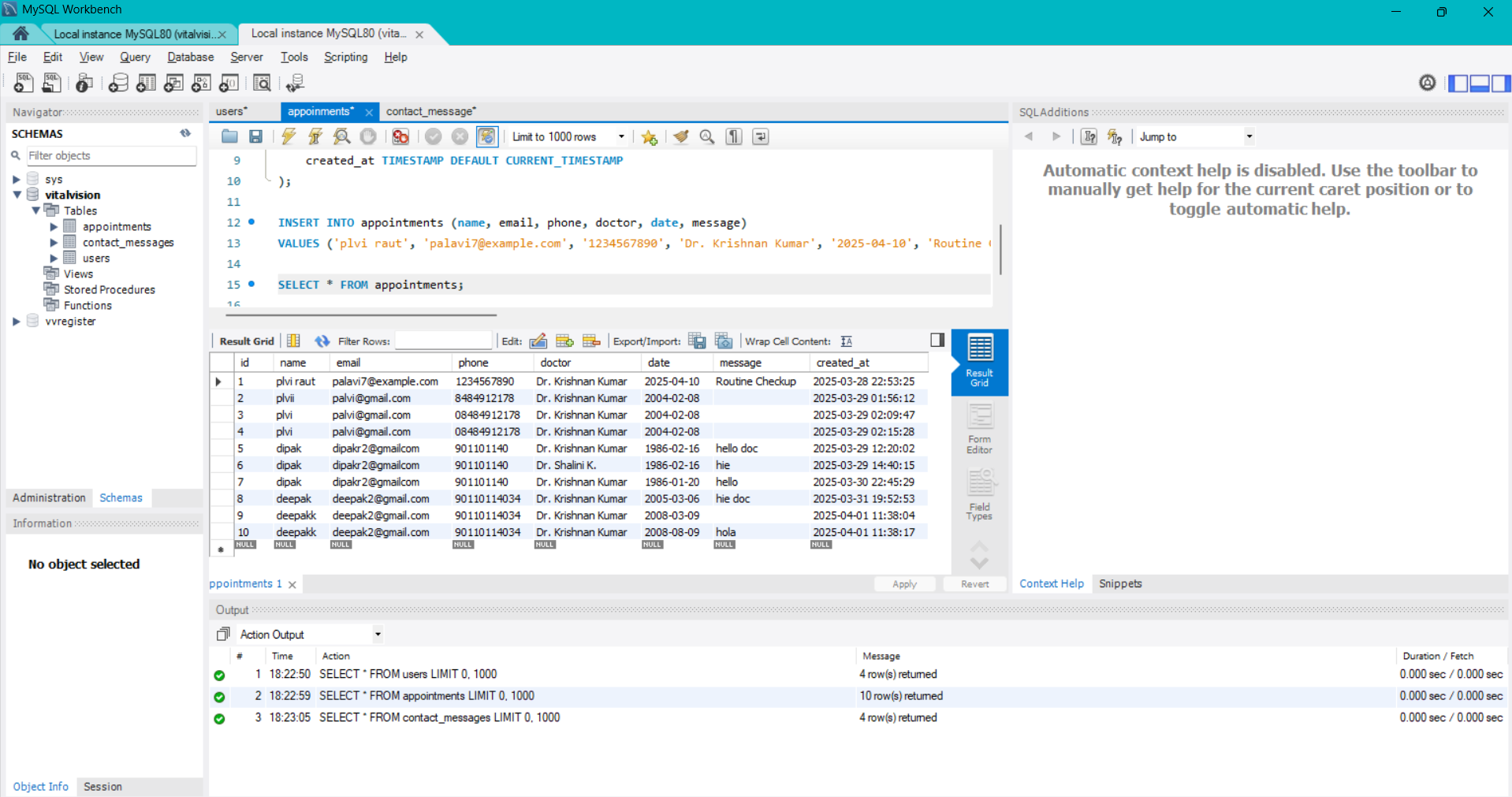


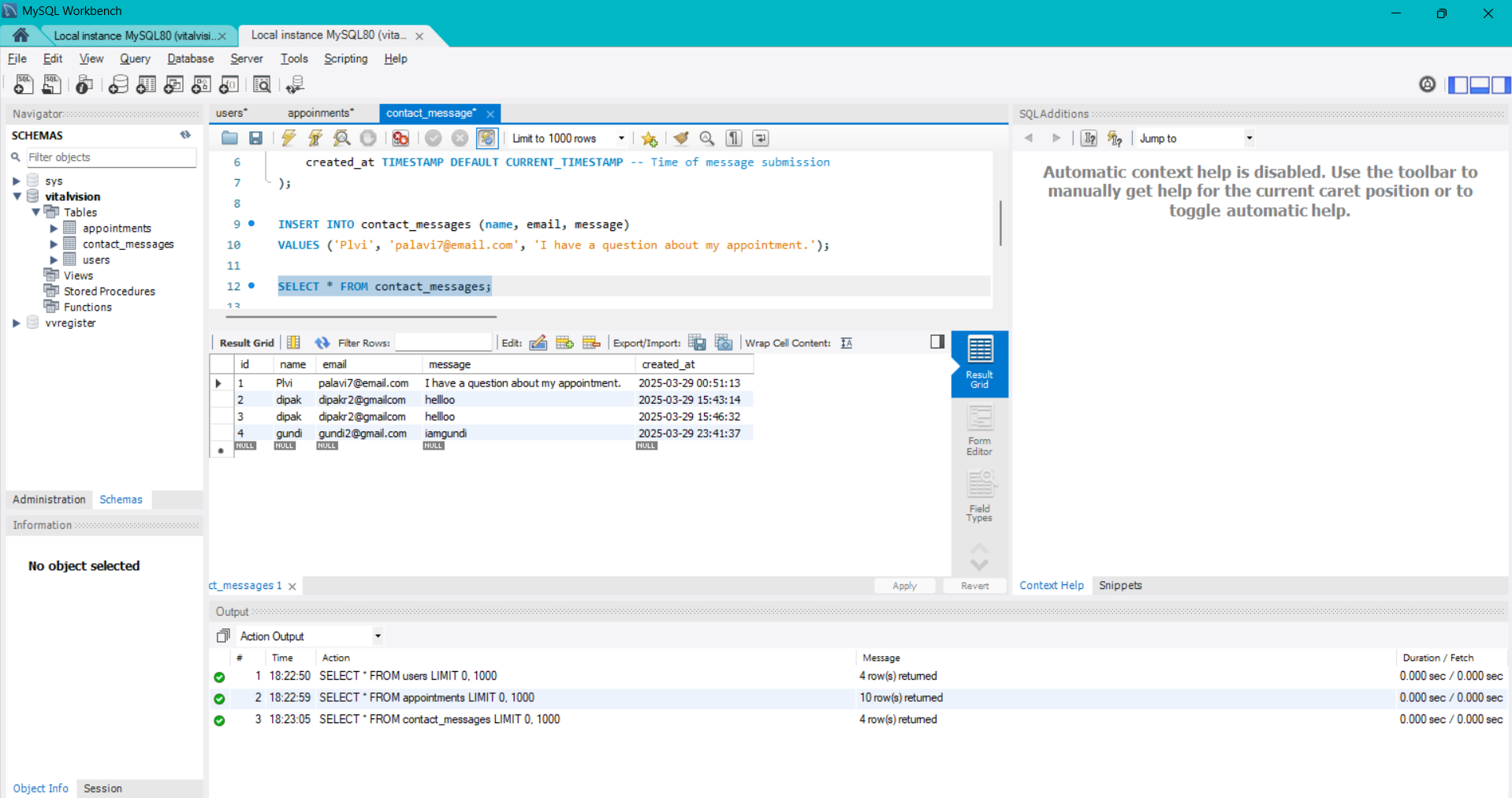












**ANNEXURE 3:** Sample Program Code

HOME.HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Doctor Template</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/vv.css') }}">

<script src="script.js"></script>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap" rel="stylesheet">

</head>

<body>

<!-- Display User's Name if Logged In -->

{% if user\_name %}

<div class="welcome-message">

<h2>Welcome, {{ user\_name }}! 🎉</h2>

</div>

{% endif %}

<!-- Hero Section -->

<section class="hero">

<div class="container">

<div class="hero-text">

<img src="{{ url\_for('static', filename='images/vv\_OG\_logo.png') }}" alt="Profile Picture" class="profile-image" height="250" width="250">

<h1>Welcome to VitalVision</h1>

<p><span id="typing"></span><span id="cursor">Your Health , Our Care|</span></p>

<a href="#services" class="cta-link">

<button class="cta-button">Get Started</button>

</a>

</div>

</div>

</section>

<!-- Services Section -->

<section class="services" id="services">

<div class="container">

<h2>Our Services</h2>

<div class="service-cards">

<div class="card">

<h3>Patient History</h3>

<p>Get patient details in just a few clicks!</p>

<a href="{{ url\_for('patient\_index') }}" class="cta-link">

<button class="cta-button">Learn More</button>

</a>

</div>

<div class="card">

<h3>Diagnostics</h3>

<p>Advanced tools for accurate results.</p>

<a href="{{ url\_for('Registerlogin') }}"class="cta-link">

<button class="cta-button" >Learn More</button>

</a>

</div>

<div class="card">

<h3>Book an Appointment</h3>

<p>Connect with the best doctors near you.</p>

<a href="{{ url\_for('hospital') }}" class="cta-link">

<button class="cta-button">Book Now</button>

</a>

</div>

</div>

</div>

</section>

<!-- Unified Testimonials & Contact Us Section -->

<section class="testimonial-contact">

<div class="container">

<!-- Testimonials -->

<h2>What Our Clients Say</h2>

<div class="testimonial-slider">

<div class="testimonial">

<p>"Excellent service and friendly staff!"</p>

<span>- xyz</span>

</div>

<div class="testimonial">

<p>"They truly care about their patients."</p>

<span>- abc</span>

</div>

</div>

<!--notification--!>

{% with messages = get\_flashed\_messages(with\_categories=true) %}

{% if messages %}

{% for category, message in messages %}

<div class="alert alert-{{ category }}">

{{ message }}

</div>

{% endfor %}

{% endif %}

{% endwith %}

<!-- Contact Us -->

<div class="contact">

<h2>Contact Us</h2>

<form action="/contact" method="post">

<input type="text" name="name" placeholder="Your Name" required>

<input type="email" name="email" placeholder="Your Email" required>

<textarea name="message" placeholder="Your Message"></textarea>

<button type="submit">Send Message</button>

</form>

</div>

</div>

</section>

<!-- Footer -->

<footer>

<div class="container">

<p>&copy; 2025 . VitalVision. All rights reserved.</p>

</div>

</footer>

<script src="script.js"></script>

</body>

</html>

VVAPP.PY

from flask import Flask, render\_template, request, redirect, url\_for, flash, session

import mysql.connector

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

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

from sklearn import svm

from sklearn.linear\_model import LogisticRegression

from werkzeug.security import generate\_password\_hash, check\_password\_hash

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

app.secret\_key = 'your\_secret\_key\_here'

# Database Connection

db = mysql.connector.connect(

    host="localhost",

    user="root",

    password="SPqlrautmy283^",

    database="vitalvision"

)

cursor = db.cursor()

print("Database connected successfully!")  # Debugging line

# Load and preprocess Diabetes Model

diabetes\_dataset = pd.read\_csv("data/diabetes.csv")

X\_diabetes = diabetes\_dataset.drop(columns='Outcome', axis=1)

Y\_diabetes = diabetes\_dataset['Outcome']

scaler\_diabetes = StandardScaler()

X\_diabetes = scaler\_diabetes.fit\_transform(X\_diabetes)

X\_train\_diabetes, X\_test\_diabetes, Y\_train\_diabetes, Y\_test\_diabetes = train\_test\_split(

    X\_diabetes, Y\_diabetes, test\_size=0.2, stratify=Y\_diabetes, random\_state=2

)

classifier\_diabetes = svm.SVC(kernel='linear')

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

# Load and preprocess Heart Disease Model

heart\_data = pd.read\_csv("data/heart\_dataset.csv")

X\_heart = heart\_data.drop(columns='target', axis=1)

Y\_heart = heart\_data['target']

X\_train\_heart, X\_test\_heart, Y\_train\_heart, Y\_test\_heart = train\_test\_split(

    X\_heart, Y\_heart, test\_size=0.2, stratify=Y\_heart, random\_state=2

)

model\_heart = LogisticRegression(max\_iter=1000)

model\_heart.fit(X\_train\_heart, Y\_train\_heart)

# Routes for Main Pages

@app.route('/')

def home():

    return render\_template('main/home.html')

@app.route('/patient')

def patient\_index():

    return render\_template('main/patient\_index.html')

@app.route('/Registerlogin')

def Registerlogin():

    return render\_template('main/Registerlogin.html')

@app.route('/hospital')

def hospital():

    return render\_template('main/hospital.html')

# Doctors List Page

@app.route('/doctor')

def doctor\_page():

    hospital\_name = request.args.get('hospital')

    return render\_template('main/doctor.html', hospital=hospital\_name) if hospital\_name else ("Hospital not found", 404)

# Doctor Profile Page

@app.route('/doctor-profile')

def doctor\_profile():

    doctor\_name = request.args.get('doctor', 'Unknown Doctor')

    # Dictionary of doctors with details

    doctors = {

        "Dr. Krishnan Kumar": {

            "speciality": "Cardiologist",

            "experience": "15 years",

            "degrees": "MBBS, MD (Cardiology)",

            "about": "Experienced cardiologist with a passion for patient care and treatments.",

            "image": "images/dr\_krishnan.jpg"

        },

        "Dr. Shalini K.": {

            "speciality": "Neurologist",

            "experience": "10 years",

            "degrees": "MBBS, MD (Neurology)",

            "about": "Expert in treating neurological disorders with innovative techniques.",

            "image": "images/dr\_shalini.jpg"

        },

        "Dr. Aatmaram": {

            "speciality": "Pediatrician",

            "experience": "8 years",

            "degrees": "MBBS, DCH",

            "about": "Caring pediatrician specializing in child health and wellness.",

            "image": "images/dr\_aatmaram.jpg"

        },

        "Dr. Govinda": {

            "speciality": "Dentist",

            "experience": "12 years",

            "degrees": "BDS, MDS",

            "about": "Skilled dentist with expertise in dental surgery and oral health care.",

            "image": "images/dr\_govinda.jpg"

        }

    }

    # Check if doctor exists in the dictionary

    if doctor\_name in doctors:

        return render\_template('main/doctor-profile.html', doctor=doctors[doctor\_name], doctor\_name=doctor\_name)

    else:

        return "Doctor not found", 404

# Patient and Health Prediction Routes

@app.route('/index')

def index():

    return render\_template('index.html')

@app.route('/patient-history')

def patient\_history():

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

@app.route('/heart-patient-list')

def heart\_patient\_list():

    return render\_template('main/heart-patient-list.html')

@app.route('/diabetes-patient-list')

def diabetes\_patient\_list():

    return render\_template('main/diabetes-patient-list.html')

@app.route('/diabetes')

def diabetes\_page():

    return render\_template('diabetes.html')

@app.route('/heart')

def heart\_page():

    return render\_template('heart.html')

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

def predict\_diabetes():

    try:    input\_data = [float(request.form[key]) for key in ['pregnancies', 'glucose', 'blood\_pressure', 'skin\_thickness', 'insulin', 'bmi', 'diabetes\_pedigree\_function', 'age']]

        std\_data = scaler\_diabetes.transform(np.asarray(input\_data).reshape(1, -1))

        prediction = classifier\_diabetes.predict(std\_data)

        result = "The person is diabetic." if prediction[0] == 1 else "The person is not diabetic."

    except ValueError as e:

        result = f"Invalid input data: {e}"

    except Exception as e:

        result = f"Error in processing: {e}"

    return render\_template('diabetes.html', prediction\_text=result)

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

def predict\_heart():

    try:

        input\_data = [float(request.form[key]) for key in ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']]

        prediction = model\_heart.predict(np.asarray(input\_data).reshape(1, -1))

        result = "The person has heart disease." if prediction[0] == 1 else "The person does not have heart disease."

    except ValueError as e:

        result = f"Invalid input data: {e}"

    except Exception as e:

        result = f"Error in processing: {e}"

    return render\_template('heart.html', prediction\_text=result)

# User Authentication Routes

# Ensure Database is Connected

if db.is\_connected():

    print("database connected successfully!")

# Route to serve the Register/Login Page

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

def register\_login\_page():

    return render\_template('registerlogin.html')

# Register Route

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

def register():

    name = request.form.get('name', '').strip()

    email = request.form.get('email', '').strip()

    phone = request.form.get('phone', '').strip()

    password = request.form.get('password', '').strip()

    if not all([name, email, phone, password]):

        return "<script>alert('❌ All fields are required!'); window.location.href='/registerlogin';</script>"

    hashed\_password = generate\_password\_hash(password)

    try:

        cursor.execute("INSERT INTO users (name, email, phone, password) VALUES (%s, %s, %s, %s)", (name, email, phone, hashed\_password))

        db.commit()

        return "<script>alert('✅ Registration successful! Login now.'); window.location.href='/registerlogin';</script>"

    except mysql.connector.Error as e:

        return f"<script>alert('❌ Database Error: {str(e)}'); window.location.href='/registerlogin';</script>"

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

def login():

    email = request.form.get('email', '').strip()

    password = request.form.get('password', '').strip()

    if not all([email, password]):

        return "<script>alert('❌ Both email and password are required!'); window.location.href='/registerlogin';</script>"

    try:

        cursor.execute("SELECT id, name, password FROM users WHERE email = %s", (email,))

        user = cursor.fetchone()

        if user and check\_password\_hash(user[2], password):

            session['user\_id'] = user[0]

            session['user\_name'] = user[1]

            return "<script>alert('✅ Login successful! Redirecting...'); window.location.href='/index';</script>"

        return "<script>alert('❌ Invalid credentials! Try again.'); window.location.href='/registerlogin';</script>"

    except Exception as e:

        return f"<script>alert('❌ Database Error: {str(e)}'); window.location.href='/registerlogin';</script>"

# Dashboard Route

@app.route('/dashboard')

def dashboard():

    if 'user\_id' in session:

        return f"<h1>Welcome {session['user\_name']}! 🎉</h1><br><a href='/logout'>Logout</a>"

    else:

        return redirect(url\_for('register\_login\_page'))

# Logout Route

@app.route('/logout')

def logout():

    session.clear()

    return redirect(url\_for('register\_login\_page'))

# Register Login Page

@app.route('/registerlogin')

def register\_login():

    return render\_template('registerlogin.html', message=request.args.get('message', ''))

# Contact Form Route

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

def contact():

    # Debugging: Print database connection and cursor

    print(f"🔍 Debug: Cursor Object → {cursor}")

    print(f"🔍 Debug: Database Connection Status → {db.is\_connected()}")

    if request.method == 'POST':

        name = request.form.get('name', '').strip()

        email = request.form.get('email', '').strip()

        message = request.form.get('message', '').strip()

         print(f"✅ Received Contact Form Data: {name}, {email}, {message}")

  if not all([name, email, message]):

            print("⚠️ Error: Missing form fields!")

            flash("All fields are required!", "error")

            return redirect(url\_for('contact'))

        try:

            query = "INSERT INTO contact\_messages (name, email, message) VALUES (%s, %s, %s)"

            values = (name, email, message

            print(f"🛠 Executing Query: {query} | Values: {values}"

            # Check database connection

            if db.is\_connected():

                print("✅ Database is connected inside contact function!")

            else    print(" Database connection lost!")

            cursor.execute(query, values)

            db.commit()

            print("✅ Message Sent Successfully!")

            flash("Message sent successfully!", "success")

        except Exception as e:

            db.rollback()

            print(f" Database Error: {str(e)}")

            flash(f"Error: {str(e)}", "error")

      return redirect(url\_for('contact'))

    return render\_template('main/home.html')

# Appointment Booking Route

@app.route('/book-appointment/<doctor\_name>', methods=['GET', 'POST'])

def book\_appointment\_page(doctor\_name):

    if request.method == 'POST':

        name = request.form.get('name', '').strip()

        email = request.form.get('email', '').strip()

        phone = request.form.get('phone', '').strip()

        doctor = request.form.get('doctor', doctor\_name)

        date = request.form.get('date', '').strip()

        message = request.form.get('message', '').strip()

        print(f"📌 Received Data: Name={name}, Email={email}, Phone={phone}, Doctor={doctor}, Date={date}, Message={message}")

        # Check for missing fields

        if not all([name, email, phone, date]):

            print("⚠️ Error: Missing form fields!")

            flash("All fields are required!", "error")

            return redirect(url\_for('book\_appointment\_page', doctor\_name=doctor\_name))

        try:

            query = "INSERT INTO appointments (name, email, phone, doctor, date, message) VALUES (%s, %s, %s, %s, %s, %s)"

            values = (name, email, phone, doctor, date, message)

            # Print query before execution

            print(f" Executing Query: {query} | Values: {values}")

            cursor.execute(query, values)

            db.commit()

            print("✅ Data successfully inserted into database!")

            flash("Appointment booked successfully!", "success")

            return redirect(url\_for('book\_appointment\_page', doctor\_name=doctor\_name))

        except Exception as e:

            db.rollback()

            print(f" Database Error: {str(e)}")

            flash("Error booking appointment. Please try again.", "error")

    return render\_template('main/book-appointment.html', doctor\_name=doctor\_name)

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

    app.run(debug=True)

**FUTURE SCOPE**

🔹 Mobile Application Development

Expanding VitalVision into a mobile app for iOS & Android to improve accessibility.

🔹 Integration with Wearable Devices

Syncing vital health data from smartwatches and fitness trackers to monitor patient health in real time.

🔹 Telemedicine Expansion

Adding video call consultations with instant prescriptions for enhanced remote care.

🔹 Blockchain-Based Data Security

Using blockchain technology to ensure secure, tamper-proof medical records.

🔹 AI-Driven Personalized Treatment Plans

Advanced AI algorithms that recommend customized treatment plans based on genetic history and lifestyle data.50