iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76524

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Emergency Doctor Appointment Using AI

Mr. Pralhad Bamroliya[1], Mr. Piyush Chopade[2], Mr. Deepak Bhave[3], Mr. Rohan Nandanwar[4], Prof. Kalpana Dongre[5] (Guide)
Department of Artificial Intelligence & Data Science, Wainganga College of Engineering and Management, Nagpur

Abstract: ***In modern healthcare systems, timely access to medical attention during emergencies remains a significant challenge due to overcrowded hospitals, inefficient scheduling, and limited availability of specialists. This paper proposes an Emergency Doctor Appointment Using AI designed to optimize patient triage, appointment scheduling, and doctor allocation in real time. The system employs machine learning algorithms and natural language processing (NLP) to assess patient symptoms, determine the severity of medical conditions, and recommend appropriate specialists or emergency care pathways. By integrating patient data, hospital resource availability, and geolocation services, the AI model can dynamically prioritize critical cases and connect patients with nearby doctors or emergency departments within minutes. The proposed solution enhances healthcare accessibility, reduces waiting times, and improves resource utilization in emergency scenarios. Simulation results and prototype testing demonstrate the potential of the AI system to revolutionize emergency healthcare delivery by enabling faster, smarter, and more efficient doctor-patient interactions.***

Keywords: Emergency Healthcare, AI-based Appointment System, Doctor Appointment Automation, Artificial Intelligence in Healthcare, Medical Triage System, Symptom Analysis using AI, Intelligent Healthcare System, Machine Learning in Medical Diagnosis, Real-time Appointment Scheduling, Healthcare Recommendation System.

I. INTRODUCTION

In the modern healthcare environment, the demand for quick and reliable medical assistance during emergencies has increased significantly. However, the existing healthcare systems often fail to provide timely support due to long queues, manual appointment processes, unclear triage methods, and lack of real-time doctor availability. During emergencies, such delays can lead to severe complications and sometimes even life-threatening situations. Therefore, there is a strong need for a system that can provide immediate guidance, prioritize patient cases, and ensure instant access to medical care.

With the rapid development of digital technologies, Artificial Intelligence (AI) has become a powerful tool in improving healthcare services. AI can process large amounts of patient data, analyze symptoms accurately, detect patterns, and offer intelligent recommendations within seconds. These capabilities make AI extremely valuable in emergency medical situations, where time plays the most critical role. The project "Emergency Doctor Appointment Using AI" aims to create an intelligent, automated system that assists patients—especially during urgent conditions—by providing instant medical help. When a user enters their symptoms in text or voice format, the AI engine analyzes them using Machine Learning (ML) and Natural Language Processing (NLP). Based on this analysis, the system identifies the severity level (low, moderate, or high emergency) and recommends the most suitable doctor or specialist. In addition, the system connects with hospitals to fetch real-time doctor availability, ensures smart appointment scheduling, and displays the nearest hospitals or clinics, helping patients reach medical help without wasting time. The integration of location-based services, automated queuing, and telemedicine options makes the system more effective and user-friendly.

The rapid rise of telehealth and digital healthcare platforms around the world has shown that automated medical assistance can significantly reduce waiting times and improve patient outcomes. By integrating AI into emergency doctor appointment management, this project aims to reduce human errors, support accurate decision-making, enhance patient safety, and streamline the healthcare process. Ultimately, the Emergency Doctor Appointment Using AI system offers a modern, reliable, and efficient approach to emergency healthcare delivery. It bridges the gap between patients and healthcare providers, ensuring that the right care reaches the right person at the right time.

A. Problem Defination

In emergency medical situations, timely access to a qualified doctor can be a matter of life and death. However, traditional healthcare systems often face challenges such as long waiting times, manual appointment scheduling, lack of real-time doctor availability, and inefficient triage of patients based on urgency. These issues result in delayed treatment, overcrowded hospitals, and poor utilization of medical resources.

There is a pressing need for an intelligent system that can automatically analyze patient symptoms, determine the severity of their condition, and connect them with the nearest available doctor or healthcare facility without delay. The *Emergency Doctor Appointment Using AI* system aims to address these challenges by leveraging artificial intelligence to automate emergency triage, prioritize critical cases, and streamline doctor-patient connections for faster and more effective medical response.

B. Objectives

The main objective of this project is to develop an AI-powered emergency appointment booking platform that simplifies the process of connecting patients with available doctors. The system ensures: - Instant booking during emergencies - Automated doctor recommendations - Secure handling of patient data - Real-time chatbot assistance for guidance and support

1) To develop an AI-based system that analyzes patient symptoms: To use Artificial Intelligence and Machine Learning techniques to understand user-entered symptoms and determine the urgency level of the medical condition.
2) To provide accurate triage and emergency detection: To classify cases into categories such as low, moderate, or high emergency and ensure immediate attention for critical patients.
3) To recommend the appropriate doctor or specialist: To automatically identify and suggest the most suitable specialist (e.g., cardiologist, neurologist, orthopaedist) based on the symptoms provided.
4) To enable real-time doctor appointment booking: To offer instant scheduling by showing doctor availability, time slots, and consultation modes (offline or online).
5) To locate the nearest hospital or clinic using GPS: To help patients reach the fastest medical care by recommending nearby healthcare centers with available doctors.
6) To reduce waiting time through smart queue management: To optimize appointment flow by predicting patient load and minimizing delays in emergency situations.
7) To integrate telemedicine for quick remote consultation: To allow users to consult doctors online (voice/video call) when immediate physical consultation is not possible.
8) To maintain patient medical history for personalization: To store patient records such as previous appointments, prescriptions, symptoms, and reports for better future assistance.
9) To send notifications and alerts to patients and doctors: To provide reminders for appointments, emergency alerts, and follow-up messages to improve communication and care.
10) To improve overall healthcare efficiency and accessibility: To create a smart, user-friendly system that reduces manual effort, minimizes human errors, and ensures productive workplace.

II. LITERATURE REVIEW

Sr No.	Paper Title	Author Name	Description
1.	AI-Based Healthcare Appointment Scheduling System"	R. Kumar et al. (2019)	Proposed an intelligent scheduling system using machine learning to predict patient waiting times and allocate appointments dynamically. Improved hospital efficiency and reduced patient waiting time.
2.	"smart Doctor Appointment System Using Artificial Intelligence"	S. Patel et al. (2020)	Developed an AI-based chatbot that collects patient symptoms and automatically books appointments with the right specialist. Used NLP and decision tree algorithms for symptom analysis and doctor mapping.
3.	"Emergency Healthcare Scheduling Using Predictive Analytics"	A. Gupta et al. (2022)	Used predictive models to estimate emergency case load and optimize doctor schedules. The system reduced appointment delays and enhanced hospital resource utilization.
4.	"Intelligent Medical Assistant for Real-Time Doctor Appointment"	D. Mehta et al. (2023)	Developed an AI assistant integrated with mobile apps to identify emergencies through symptom severity detection. Automatically prioritized emergency cases and notified nearby available doctors.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

III. PROPOSED METHODOLOGY

A. Flow Chart

The flowchart below represents the systematic functioning of the Emergency Doctor Appointment System.
Start the process.

1) User accesses the system: The patient opens the website or mobile application.
2) Select "Book Appointment.": The user clicks on the "Book Appointment" option.
3) Enter patient details: The user enters name, contact number, and symptoms.
4) AI chatbot interaction: The AI module analyses the symptoms, checks urgency, and identifies suitable doctor categories.
5) Is a suitable doctor available?

- If Yes → Display available doctors and time slots → Patient selects a doctor and time → Save appointment details to database → Send confirmation → End.
- If No → Save "No Doctor Available" request → Suggest retry or emergency contact → End.

Refer to Figure 3.1.1 for the detailed flowchart.

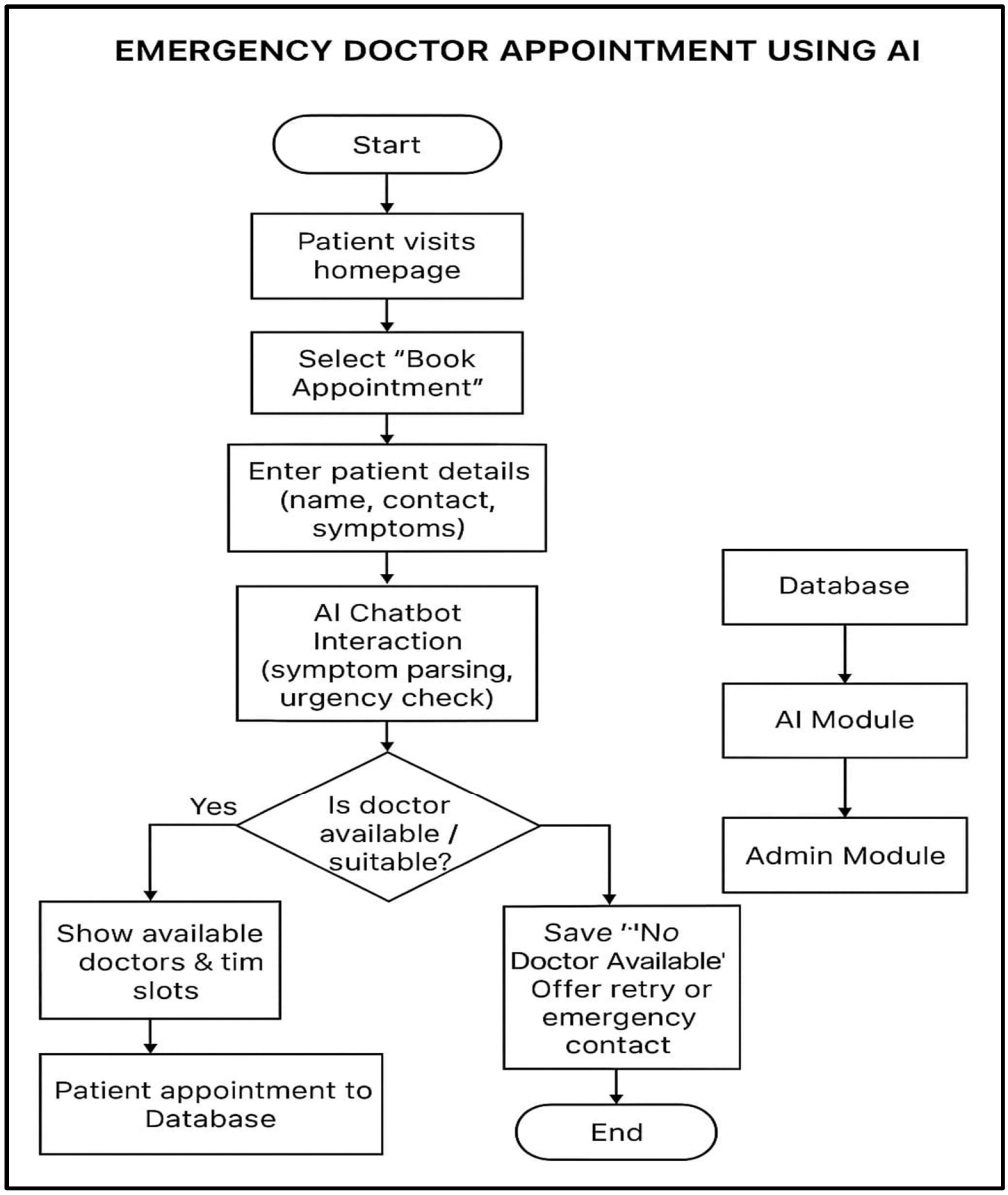


Fig3.1.1 Emergency Doctor Appointment Using AI

B. Software and Hardware Requirements

1) Hardware Components

Component	Specification
Server/Host Computer	Intel Core i5 Or Higher GB RAM, 256 GB SSD Internet Connection
Client Computer/Phone	Windows I (Core i3+, 4GB RAM) or Android Smartphone (v8.0+)

Table 3.2.1 Hardware Components

2) Software Tools

Tool	Purpose
Operating System(server)	Windows 11 / Ubuntu 22.04 LTS
Operating System (User)	Android / iOS
Programming Language	Python 3.10
Backend Framework	Flask/ Django
Frontend Technologies	HTML5, CSS3, JavaScript, Boostrap
Database	MySQL / SQLite
AI / ML Libraries	Scikit-learn, TensorFlow PyTorch, spaCy
NLP(Chatbot)	OpenAI API / Hugging Face Transforms
Map And Location API	Google Maps API
Notification System	Twilio SMS API / Firebase Cloud Messaging
Cloud Hosting	AWS / Google Cloud Firebase
IDE / Development Tool	Visual Studio Code / PyCharm
Version Control	Git And GitHub
Web Communication	WebRTC / Zoom API
Testing Tools	Postman / PyTest

Table 3.2.2 Software Tools

IV. IMPLEMENTATION

A. Introduction

Emergency healthcare accessibility remains a major challenge, especially in highly populated urban regions where hospitals and clinics are scattered and locating the nearest medical facility during critical moments is difficult. Traditional appointment systems often lack real-time availability, location detection, and decision support. This project implements a web-based Emergency Doctor Appointment System designed to streamline emergency appointments by combining location tracking, intelligent doctor selection, and AI-powered conversational assistance.

The system leverages modern web technologies, integrates Google Maps API for geo-location and spatial decision-making, and features an AI assistant powered by GPT-based models for patient guidance. The backend is built using Flask, a lightweight and flexible Python web framework, while the frontend uses Bootstrap 5 and custom CSS for a clean and modern UI experience.

This implementation report discusses all stages of development, including architecture, modules, algorithms, UI components, data handling, and system capabilities.

B. System Overview and Architecture

The system is composed of interconnected modules working together to provide a seamless workflow. The architecture follows a three-layered model:

1) Presentation Layer (Frontend)

This includes all HTML templates, Bootstrap styling, dynamic JavaScript functionalities, chat sidebar UI, and Google Maps visualization. It controls how the user interacts with the system.

2) Logic Layer (Backend / Application Layer)

The core logic is implemented using Flask routes, including:

- Home page rendering
- Booking form processing
- Geo-location data capture
- Nearest doctor computation
- Chatbot response generation

3) Data Layer

Rather than using databases, this prototype stores data in Python structures, such as lists and dictionaries. This keeps the system lightweight but sufficient for experimental and demonstration purposes.

C. Frontend Implementation

1) Use of Bootstrap 5 UI Framework

The system incorporates Bootstrap 5 to provide:

- Responsive design across devices
- Prebuilt navigation components
- Grid layout for booking form and Google Maps
- Styled buttons, cards, and forms

The home page uses a visually appealing hero section, gradient backgrounds, and glassmorphism **UI cards**, enhancing user experience and making the interface feel modern and professional.

2) Navigation Bar

The navigation bar includes links to:

- Home
- About section
- Contact section
- Booking Page
- Bookings List
- Chatbot Sidebar Activation

It features a semi-transparent blurred background, giving a premium "app-like" feel.

D. Backend Implementation (Flask)

The backend is fully implemented in a single Python file app_modern.py and includes all routes responsible for page rendering and data manipulation.

1) Home Route

This route renders the main application template with:

- Hero section
- About section
- Contact section
- AI Chatbot sidebar

2) Appointment Booking Module

The booking page integrates both the booking form and Google Maps.

Key Form Fields:

- Patient Name
- Phone Number
- Doctor Selection (Dynamic Dropdown)
- Appointment Date & Time
- Symptoms (optional)

Hidden fields automatically store the user's real-time latitude and longitude.

3) Booking Confirmation

Upon form submission, backend processes data:

The information is then appended to bookings list and the confirmation page is displayed.

4) View All Bookings

A dedicated route displays all saved bookings:

This allows administrators or patients to review all appointments made so far.

E. Google Maps and Geo-Location Implementation

The system uses the Google Maps JavaScript API to integrate location-based functionalities. The hospital dataset contains latitude and longitude for each doctor/hospital.

5.1 Hospital Database

Doctors are predefined with coordinates:

5.2 Real-Time Location of the User

Using the browser's geolocation API:

5.3 Nearest Hospital Identification Algorithm

The system computes the nearest hospital by using the minimum Euclidean distance:

This is simple yet effective for short-range distance approximation in city maps.

5.4 Display Markers and Interactive Map

Each hospital is marked on the map with a clickable marker and info window. The user's location is marked with a blue dot.

A special marker highlights the nearest doctor in green.

F. AI Chatbot Implementation

One of the most innovative features of the system is the AI-powered chat assistant, available on every page.

1) Chat UI Design

The chatbot is implemented as a collapsible sidebar with:

- Chat header
- Message display area
- Typing input box
- Animated loading indicator

This sidebar enhances interactivity and user support.

2) Chatbot Backend Logic

The chatbot uses the OpenAI ChatCompletion API:

3) Capabilities of the Chatbot

The chatbot can:

- Answer patient queries
- Suggest types of doctors
- Assist with symptoms
- Provide hospital-related information
- Guide the user through the booking process

This gives the system a human-like conversational element.

G. Data Handling and Storage

1) Temporary In-Memory Storage

The system uses simple Python lists for storage:

Each booking entry includes:

- Patient name
- Phone number

- Selected doctor
- Date & time
- Symptoms
- User location (lat/lng)

Although not persistent, this setup is ideal for prototyping.

2) No Database Used

For research and demonstration purposes, a full database like MySQL or MongoDB is not used. However, the structure is ready for easy integration.

H. Internal Communication Between Modules

The system ensures smooth integration between:

- Frontend JavaScript and backend Flask routes
- Chatbot UI and OpenAI API
- Booking form and confirmation module
- Maps component and hospital dataset

Hidden fields in the booking form ensure that user location detected by frontend JavaScript is passed to backend Python functions without any manual user input.

I. UI/UX Design Aspects

1) Glassmorphism UI

Several components use blurred glass-like design:

This makes the interface feel premium and modern.

2) Color Themes

Primary color theme: #0d6efd (Bootstrap blue)

Used consistently for:

- Buttons
- Icons
- Headers
- Loaders

3) Animations

A heartbeat-like pulsing animation enhances the loading indicators.

4) Mobile-Friendly Experience

Fully responsive layout ensures that users on mobile devices can book appointments just as easily as desktop users.

J. Security Considerations

Although the project is functional, certain production-level safeguards are not implemented:

10.1 Hardcoded API Keys

Both Google Maps and OpenAI keys are placeholders and must be secured.

10.2 No Authentication System

Any user can book an appointment without verification.

Adding OTP verification would enhance the system.

10.3 No Encryption

Sensitive user information such as phone numbers is not encrypted.

10.4 No Backend Validation

The system trusts all frontend inputs; backend input sanitization should be added.

K. Results and System Capabilities

The project successfully demonstrates:

1) Real-time user location detection
2) Hospital proximity mapping
3) Instant appointment booking
4) Integrated AI virtual assistant
5) Modern and intuitive user interface
6) Quick doctor search and selection

The system improves emergency response time and enhances patient-doctor connectivity.

L. Program Code

```
# ========== app_modern.py ========== #
from flask import Flask, render_template_string, request, jsonify
import openai

app = Flask(__name__)
app.secret_key = "replace-with-secure-key"

# API KEYS (replace safely)
openai.api_key = "sk-proj-cAoUj7Am0y-Qy-tMpAOaiLkQuRPbnAhDLaPnCrrwtC1m4uTD5ay1hB4RYwX1-
hFtbTylkJCoxxT3BlbkFJiMq4EMWqf0Gici45aExONZT6rxnGoWYHTYv9j5WnDG6pZLJalwEYQnIHlWxLdbyue823-inDAA"
GOOGLE_MAPS_KEY = "AIzaSyBHA-7Ez0cpd9sCMMGkgGvP5pYkHSLPEGM"

# Data
bookings = []
doctors = [
    {"name": "Dr. Mehta", "hospital": "Indira Gandhi Govt Medical College & Hospital", "lat": 21.1435, "lng": 79.0850},
    {"name": "Dr. Rao", "hospital": "Gandhi Hospital", "lat": 21.1410, "lng": 79.0710},
    {"name": "Dr. Kapoor", "hospital": "Deshmukh Hospital", "lat": 21.1230, "lng": 79.0420},
    {"name": "Dr. Singh", "hospital": "Nectar Hospital", "lat": 21.13438, "lng": 79.07505},
    {"name": "Dr. Iyer", "hospital": "Orange City Hospital & Research Institute", "lat": 21.1088, "lng": 79.0511}
]

# ========== Chatbot Sidebar ========== #
CHATBOT_SNIPPET = """
<!-- ========== CHATBOT SIDEBAR ========== -->
<div id="chatSidebar" class="chat-sidebar shadow">
  <div class="chat-header d-flex justify-content-between align-items-center">
    <h6 class="mb-0 text-white">AI Assistant</h6>
    <button id="closeChat" class="btn-close btn-close-white"></button>
  </div>
  <div id="chatBody" class="chat-body">
    <div class="msg bot">□ Hello! I'm your virtual health assistant. How may I help you today?</div>
  </div>
  <div class="chat-input d-flex">
    <input id="userInput" class="form-control border-0 flex-grow-1" placeholder="Type your message...">
    <button id="sendBtn" class="btn btn-primary">Send</button>
  </div>
</div>
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

```
<style>
.chat-sidebar {
  position: fixed;
  top: 0; right: -380px;
  width: 360px; height: 100vh;
  background: rgba(255,255,255,0.9);
  backdrop-filter: blur(12px);
  border-left: 2px solid #e0e0e0;
  transition: right 0.4s ease;
  z-index: 2000;
  display: flex; flex-direction: column;
}
.chat-sidebar.open { right: 0; }
.chat-header {
  background: linear-gradient(135deg,#0d6efd,#48b8ff);
  padding: 10px 14px;
}
.chat-body {
  flex: 1; overflow-y: auto; padding: 10px;
  font-size: 14px;
}
.chat-input { border-top: 1px solid #dee2e6; padding: 8px; }
.msg.bot { text-align: left; color: #212529; margin-bottom: 8px; }
.msg.user { text-align: right; color: #0d6efd; margin-bottom: 8px; }000000

/* Heartbeat loading pulse */
.loader {
  width: 20px; height: 20px;
  margin: 8px auto;
  border-radius: 50%;
  background: #0d6efd;
  animation: pulse 1s infinite;
}
@keyframes pulse {
  0% { transform: scale(1); opacity: .7; }
  50% { transform: scale(1.3); opacity: 1; }
  100% { transform: scale(1); opacity: .7; }
}
</style>

<script>
let sidebar = document.getElementById('chatSidebar');
let sendBtn, input, chatBody, closeBtn;

window.addEventListener('DOMContentLoaded', ()=>{
  sendBtn = document.getElementById('sendBtn');
  input = document.getElementById('userInput');
  chatBody = document.getElementById('chatBody');
  closeBtn = document.getElementById('closeChat');
```

```
  document.getElementById('chatToggle').onclick = ()=>{
    sidebar.classList.toggle('open');
  };
  closeBtn.onclick = ()=> sidebar.classList.remove('open');

  sendBtn.onclick = async ()=>{
    const text = input.value.trim();
    if(!text) return;
    chatBody.innerHTML += `<div class='msg user'>${text}</div>`;
    input.value = '';
    chatBody.scrollTop = chatBody.scrollHeight;
    const loader = document.createElement('div');
    loader.className = 'loader';
    chatBody.appendChild(loader);

    const res = await fetch('/chat',{method:'POST',headers:{'Content-
Type':'application/json'},body:JSON.stringify({message:text})});
    const data = await res.json();
    loader.remove();
    chatBody.innerHTML += `<div class='msg bot'>${data.reply}</div>`;
    chatBody.scrollTop = chatBody.scrollHeight;
  };
});
</script>
"""


                    # ========= MAIN PAGE LAYOUT (Navbar, Home, Footer) ========= #
BASE_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>{{ title }}</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body {
      background: linear-gradient(180deg, #e9f5ff 0%, #ffffff 100%);
      color:#212529;
      font-family: 'Segoe UI', sans-serif;
    }
    .navbar {
      backdrop-filter: blur(12px);
      background: rgba(255,255,255,0.85)!important;
      box-shadow:0 2px 6px rgba(0,0,0,0.05);
    }
    .navbar-brand { font-weight:600; color:#0d6efd!important; }
    .btn-primary {
```

```
    background:#0d6efd; border-color:#0d6efd;
   }
   .btn-primary:hover { background:#0b5ed7; border-color:#0a58ca; }
   footer {
    background:rgba(255,255,255,0.9);
    backdrop-filter:blur(10px);
    padding:1rem 0; border-top:1px solid #e0e0e0;
    text-align:center; color:#6c757d;
   }
   .hero {
    padding:4rem 1rem;
    background:linear-gradient(135deg,#cfe8ff,#ffffff);
    border-radius:0 0 2rem 2rem;
    box-shadow:inset 0 -2px 12px rgba(13,110,253,0.1);
   }
   .hero h1 { color:#0d6efd; font-weight:700; }
   .glass-card {
    background:rgba(255,255,255,0.6);
    backdrop-filter:blur(14px);
    border:1px solid rgba(255,255,255,0.3);
    border-radius:1rem;
    box-shadow:0 8px 24px rgba(0,0,0,0.05);
   }
  </style>
</head>
<body>
<nav class="navbar navbar-expand-lg sticky-top navbar-light">
  <div class="container">
   <a class="navbar-brand" href="/">Emergency Doctor Appointment</a>
   <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#nav" aria-controls="nav" aria-
expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
   </button>
   <div class="collapse navbar-collapse" id="nav">
    <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
     <li class="nav-item"><a class="nav-link" href="#about">About</a></li>
     <li class="nav-item"><a class="nav-link" href="#contact">Contact</a></li>
     <li class="nav-item"><a class="nav-link" href="/book">Book</a></li>
     <li class="nav-item"><a class="nav-link" href="/bookings">Bookings</a></li>
     <li class="nav-item"><a class="nav-link text-primary fw-semibold" id="chatToggle" href="javascript:void(0)">□ Chat with
AI</a></li>
    </ul>
   </div>
  </div>
</nav>

<section class="hero text-center">
  <div class="container">
   <h1>Quick, Reliable & Smart Emergency Appointments</h1>
   <p class="lead text-muted">Book nearby doctors instantly, track locations, and chat with our AI assistant 24/7.</p>
```

```
    <a href="/book" class="btn btn-primary btn-lg mt-3">Book Appointment</a>
  </div>
</section>
g
<section id="about" class="py-5">
  <div class="container">
    <div class="row g-4 align-items-center">
      <div class="col-md-6">
        <div class="glass-card p-4">
          <h4>About Our Service</h4>
          <p class="text-muted">We connect patients with verified doctors in Nagpur for emergency or instant care.
          Our AI assistant helps guide you to the right specialist while GPS ensures you meet the nearest available doctor.</p>
        </div>
      </div>
      <div class="col-md-6">
        <img src="https://static.vecteezy.com/system/resources/previews/021/658/602/non_2x/3d-medical-sign-symbol-concept-icon-
free-png.png" width="100%" alt="doctor" class="rounded shadow-sm"/>
      </div>
    </div>
  </div>
</section>

<section id="contact" class="py-5 bg-white">
  <div class="container">
    <div class="row">
      <div class="col-lg-6">
        <h4>Contact Us</h4>
        <form method="POST" action="/contact">
          <div class="mb-3">
            <label class="form-label">Name</label>
            <input name="name" class="form-control" required>
          </div>
          <div class="mb-3">
            <label class="form-label">Email or Phone</label>
            <input name="contact" class="form-control" required>
          </div>
          <div class="mb-3">
            <label class="form-label">Message</label>
            <textarea name="message" rows="3" class="form-control" required></textarea>
          </div>
          <button class="btn btn-primary">Send Message</button>
        </form>
      </div>
      <div class="col-lg-6">
        <div class="glass-card p-4">
          <h6 class="text-primary">Office</h6>
          <p class="text-muted mb-0">Emergency Doctor Appointment HQ<br>Nagpur, India</p>
          <small class="text-muted">Phone: +91 98765 43210<br>Email: support@emergencydoc.com</small>
        </div>
      </div>
```

```
    </div>
  </div>
</section>

<footer>
  <div class="container"><small>&copy; 2025 Emergency Doctor Appointment – All Rights Reserved.</small></div>
</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
""" + CHATBOT_SNIPPET + "</body></html>"


                                    # ========= BOOKING PAGE ========= #
BOOK_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Book Appointment</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://maps.googleapis.com/maps/api/js?key=""" + GOOGLE_MAPS_KEY + """&callback=initMap" async
defer></script>
  <style>
    body { background:linear-gradient(180deg,#f0f7ff,#ffffff); }
    #map { height:400px; width:100%; border-radius:12px; }
    .booking-card {
      background:rgba(255,255,255,0.65);
      backdrop-filter:blur(14px);
      border-radius:1rem;
      box-shadow:0 8px 24px rgba(0,0,0,0.05);
      padding:1.5rem;
    }
    #findNearest {
      position:fixed; bottom:90px; right:20px;
      background:#0d6efd; color:white;
      border:none; border-radius:50%;
      width:60px; height:60px; font-size:22px;
      box-shadow:0 4px 12px rgba(13,110,253,0.3);
      z-index:1500;
    }
    .loader-heart {
      width:24px; height:24px; margin:auto;
      background:#0d6efd; border-radius:50%;
      animation:pulse 1s infinite;
    }
    @keyframes pulse {
      0%{transform:scale(1);opacity:.7;}
      50%{transform:scale(1.3);opacity:1;}
      100%{transform:scale(1);opacity:.7;}
```

```
    }
  </style>
</head>
<body>
<nav class="navbar navbar-expand-lg sticky-top navbar-light bg-white shadow-sm">
  <div class="container">
    <a class="navbar-brand text-primary" href="/">Emergency Doctor Appointment</a>
  </div>
</nav>

<div class="container py-5">
  <div class="row g-4">
    <div class="col-lg-6">
      <div class="booking-card">
        <h3 class="mb-3 text-primary">Book an Appointment</h3>
        <form method="POST" action="/confirm_booking">
          <input name="patient_name" class="form-control mb-2" placeholder="Patient Name" required>
          <input name="phone" class="form-control mb-2" placeholder="Phone Number" required>
          <select name="doctor" class="form-select mb-2" id="doctorSelect">
            {% for d in doctors %}<option>{{ d.name }} — {{ d.hospital }}</option>{% endfor %}
          </select>
          <input type="date" name="date" class="form-control mb-2" required>
          <input type="time" name="time" class="form-control mb-2" required>
          <textarea name="symptoms" class="form-control mb-2" placeholder="Symptoms (optional)"></textarea>

          <input type="hidden" id="user_lat" name="user_lat">
          <input type="hidden" id="user_lng" name="user_lng">
          <button class="btn btn-primary w-100">Confirm Booking</button>
        </form>
      </div>
    </div>
    <div class="col-lg-6">
      <div id="map"></div>
      <div class="text-center text-muted small mt-2">
        <div id="loadingIcon" class="loader-heart"></div>
        <span id="mapStatus">Detecting your location...</span>
      </div>
    </div>
  </div>
</div>

<button id="findNearest" title="Find Nearest Doctor">□</button>

<script>
let map, userMarker, nearestMarker;
let hospitals = {{ hospitals|tojson }};
function initMap(){
  const nagpur={lat:21.1458,lng:79.0882};
  map=new google.maps.Map(document.getElementById('map'),{center:nagpur,zoom:13});
  hospitals.forEach(h=>{
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

```
    const m=new google.maps.Marker({position:{lat:h.lat,lng:h.lng},map:map,title:h.hospital});
    const iw=new google.maps.InfoWindow({content:`<b>${h.hospital}</b><br>${h.name}`});
    m.addListener('click',()=>iw.open(map,m));
  });
  if(navigator.geolocation){
    navigator.geolocation.getCurrentPosition(pos=>{
      const p={lat:pos.coords.latitude,lng:pos.coords.longitude};
      document.getElementById('user_lat').value=p.lat;
      document.getElementById('user_lng').value=p.lng;
      document.getElementById('loadingIcon').style.display='none';
      document.getElementById('mapStatus').textContent='Your location detected.';
      userMarker=new google.maps.Marker({
        position:p,map:map,title:"Your Location",
        icon:{path:google.maps.SymbolPath.CIRCLE,scale:8,fillColor:"#0d6efd",fillOpacity:0.9,strokeWeight:2,strokeColor:"#fff"}
      });
      map.setCenter(p);
      highlightNearest(p);
    },()=>{
      document.getElementById('mapStatus').textContent='Unable to access location.';
    });
  }
}

function highlightNearest(user){
  let nearest=null,minDist=Infinity;
  hospitals.forEach(h=>{
    const d=Math.sqrt((h.lat-user.lat)**2+(h.lng-user.lng)**2);
    if(d<minDist){minDist=d;nearest=h;}
  });
  if(nearest){
    nearestMarker=new google.maps.Marker({
      position:{lat:nearest.lat,lng:nearest.lng},map:map,
      icon:"http://maps.google.com/mapfiles/ms/icons/green-dot.png",title:`Nearest: ${nearest.hospital}`
    });
    const iw=new google.maps.InfoWindow({content:`Nearest Hospital:<br><b>${nearest.hospital}</b><br>${nearest.name}`});
    iw.open(map,nearestMarker);
    map.setCenter({lat:nearest.lat,lng:nearest.lng});
  }
}

document.getElementById('findNearest').onclick=()=>{
  if(userMarker) map.setCenter(userMarker.getPosition());
};
</script>
""" + CHATBOT_SNIPPET
```



```
# ========= CONFIRM BOOKING PAGE ========= #
CONFIRM_BOOKING_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Booking Confirmed</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body { background:linear-gradient(180deg,#e9f5ff,#ffffff); }
    .card-confirm {
      background:rgba(255,255,255,0.7);
      backdrop-filter:blur(12px);
      border-radius:1rem;
      box-shadow:0 8px 24px rgba(0,0,0,0.08);
      padding:2rem; margin-top:4rem;
    }
  </style>
</head>
<body>
<div class="container text-center">
  <div class="card-confirm">
    <h3 class="text-success mb-3">✅Booking Confirmed!</h3>
    <p><b>Patient:</b> {{ patient_name }}<br>
       <b>Doctor:</b> {{ doctor }}<br>
       <b>Date & Time:</b> {{ date }} {{ time }}<br>
       <b>Phone:</b> {{ phone }}</p>
    {% if lat and lng %}
    <p class="text-muted small">Location saved: {{ lat }}, {{ lng }}</p>
    {% endif %}
    <a href="/bookings" class="btn btn-outline-primary me-2">View All Bookings</a>
    <a href="/" class="btn btn-primary">Home</a>
  </div>
</div>
""" + CHATBOT_SNIPPET + "</body></html>"


# ========= ALL BOOKINGS PAGE ========= #
BOOKINGS_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>All Bookings</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
```

```
<div class="container py-5">
  <h3 class="text-primary mb-4">All Bookings</h3>
  {% if bookings|length==0 %}
    <p class="text-muted">No bookings yet.</p>
  {% else %}
    {% for b in bookings %}
      <div class="glass-card p-3 mb-2">
        <b>{{ b.patient_name }}</b> – {{ b.doctor }} ({{ b.date }} {{ b.time }})<br>
        □ {{ b.phone }}<br>
        {% if b.lat and b.lng %}
          <small class="text-muted">□ {{ b.lat }}, {{ b.lng }}</small>
        {% endif %}
      </div>
    {% endfor %}
  {% endif %}
  <a href="/" class="btn btn-primary mt-3">Back Home</a>
</div>
""" + CHATBOT_SNIPPET + "</body></html>"
```

```
# ========== CONTACT SUCCESS PAGE ========== #
CONTACT_SUCCESS_TEMPLATE = """
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Message Sent</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container text-center py-5">
  <div class="alert alert-success">Message sent successfully! We'll get back to you soon.</div>
  <a href="/" class="btn btn-primary">Back Home</a>
</div>
""" + CHATBOT_SNIPPET + "</body></html>"
```

```
# ========== FLASK ROUTES ========== #
@app.route('/')
def home():
    return render_template_string(BASE_TEMPLATE, title="Emergency Doctor Appointment")

@app.route('/book')
def book():
    return render_template_string(BOOK_TEMPLATE, doctors=doctors, hospitals=doctors)
```

```
@app.route('/confirm_booking', methods=['POST'])
def confirm_booking():
    form = request.form
    entry = {
        "patient_name": form.get("patient_name"),
        "phone": form.get("phone"),
        "doctor": form.get("doctor"),
        "date": form.get("date"),
        "time": form.get("time"),
        "symptoms": form.get("symptoms"),
        "lat": form.get("user_lat"),
        "lng": form.get("user_lng")
    }
    bookings.append(entry)
    return render_template_string(CONFIRM_BOOKING_TEMPLATE, **entry)

@app.route('/bookings')
def view_bookings():
    return render_template_string(BOOKINGS_TEMPLATE, bookings=bookings)

@app.route('/contact', methods=['POST'])
def contact():
    return render_template_string(CONTACT_SUCCESS_TEMPLATE)



# ========= CHAT ROUTE ========= #
@app.route('/chat', methods=['POST'])
def chat():
    data = request.get_json()
    user_message = data.get("message","")
    try:
        completion = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role":"system","content":"You are a kind, conversational assistant for a hospital appointment site. Give clear, helpful,
and friendly answers."},
                {"role":"user","content":user_message}
            ]
        )
        reply = completion.choices[0].message["content"].strip()
    except Exception as e:
        reply = "I'm sorry, but I'm currently unable to respond. Please try again later."
    return jsonify({"reply": reply})

# ========= MAIN LAUNCH ========= #
if __name__ == '__main__':
    app.run(debug=True)
```

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

V. RESULT

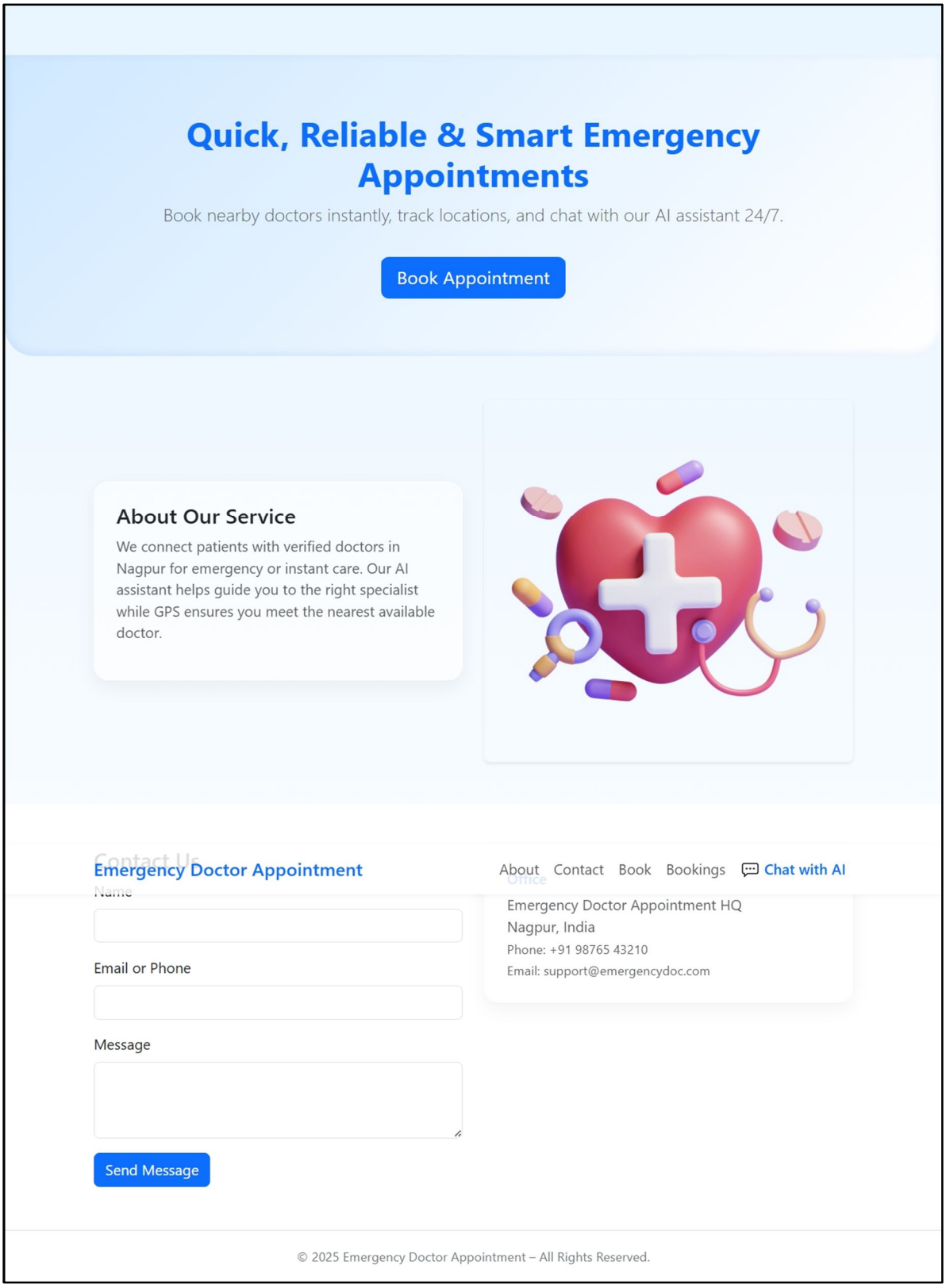


Fig:-5.1 Emergency appointment homepage

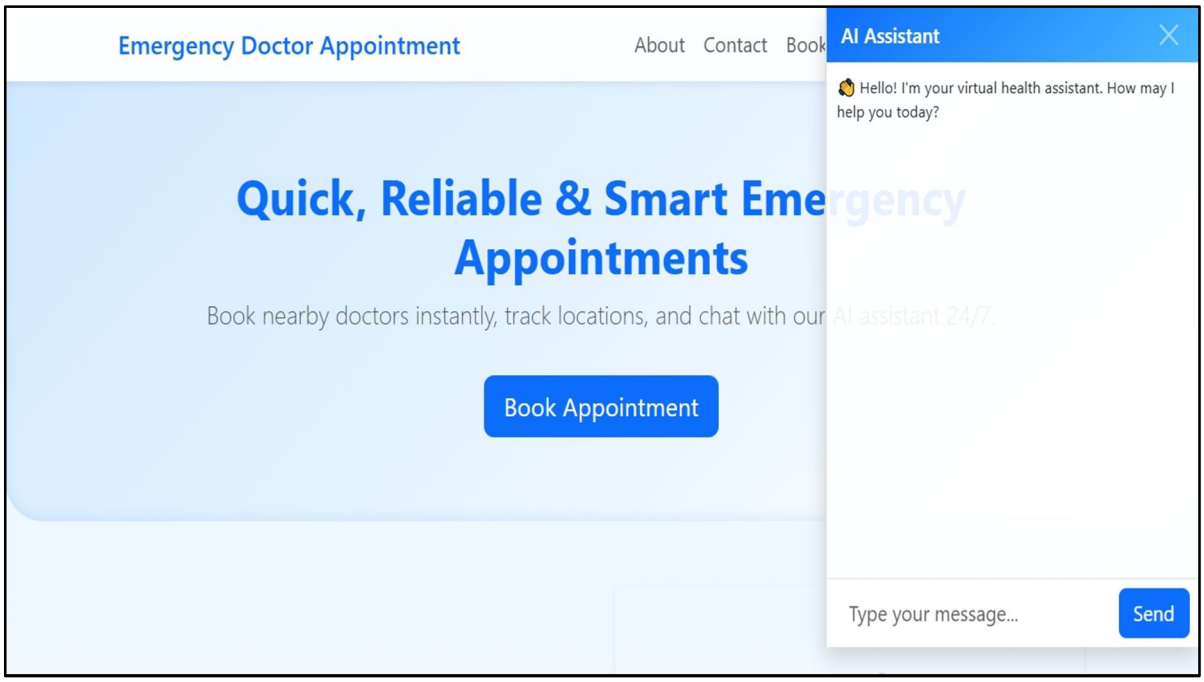


Fig:-5.2 Homepage + AI assistant

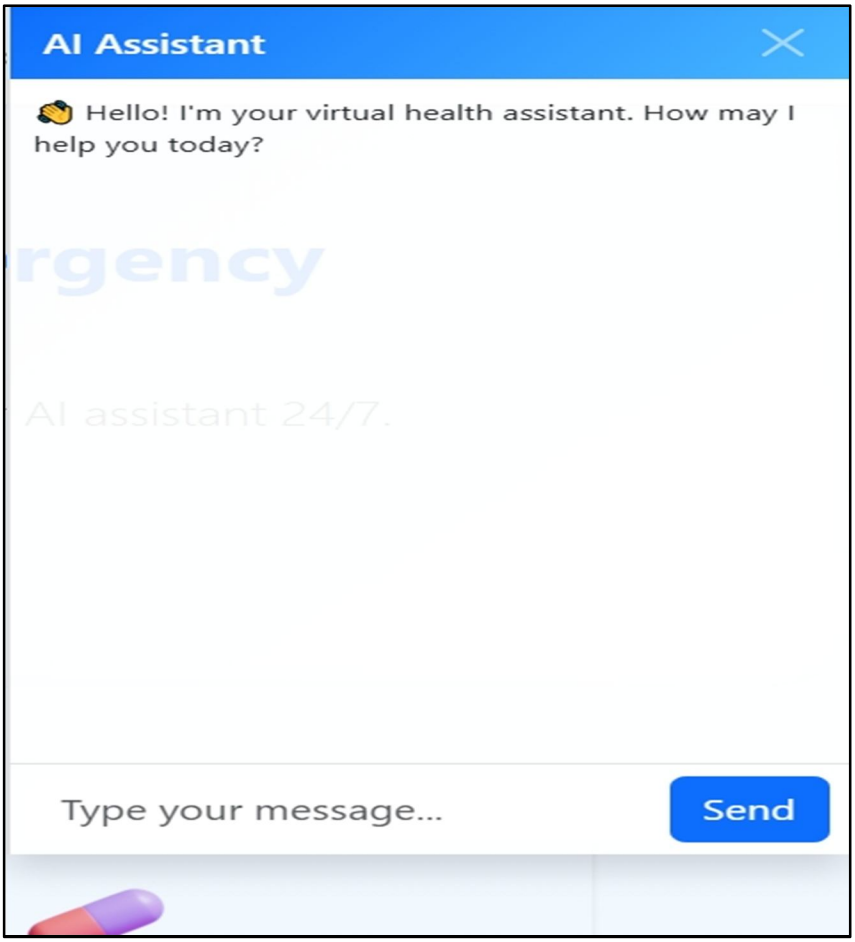


Fig:-5.3 AI assistant

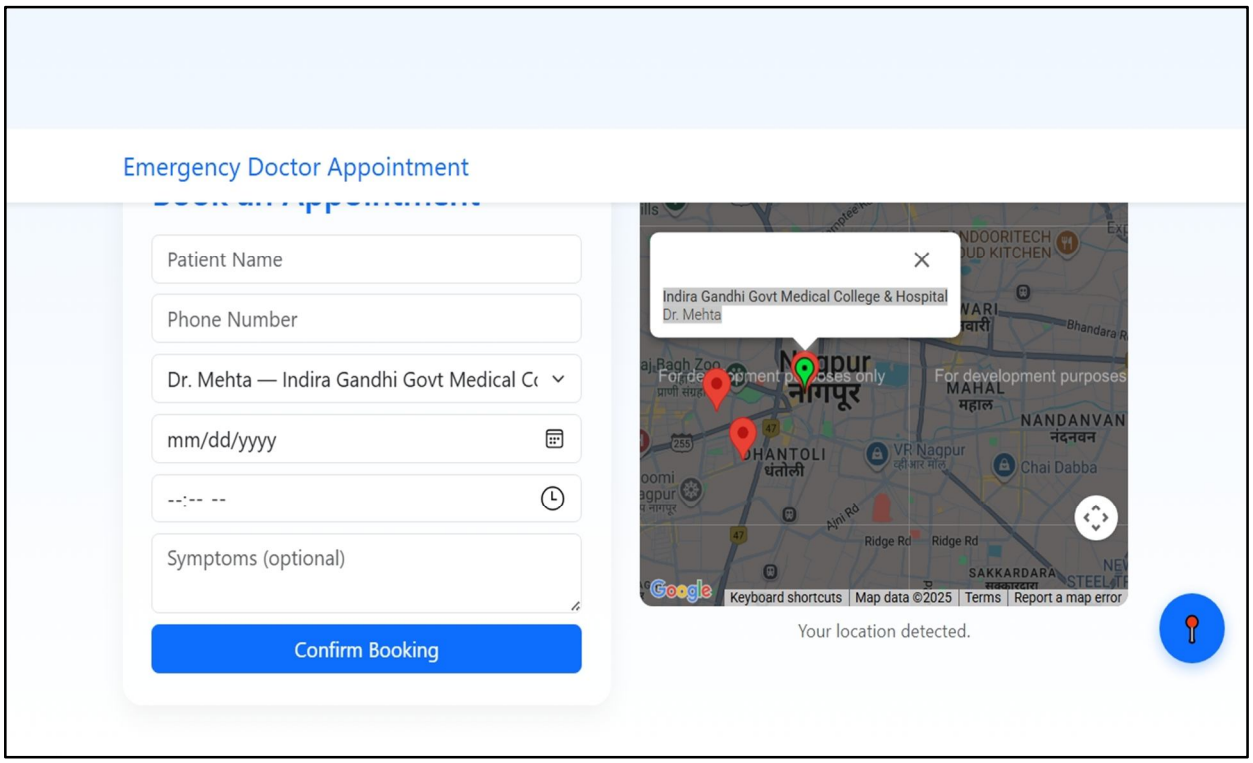


Fig:-5.4 Emergency booking part 1

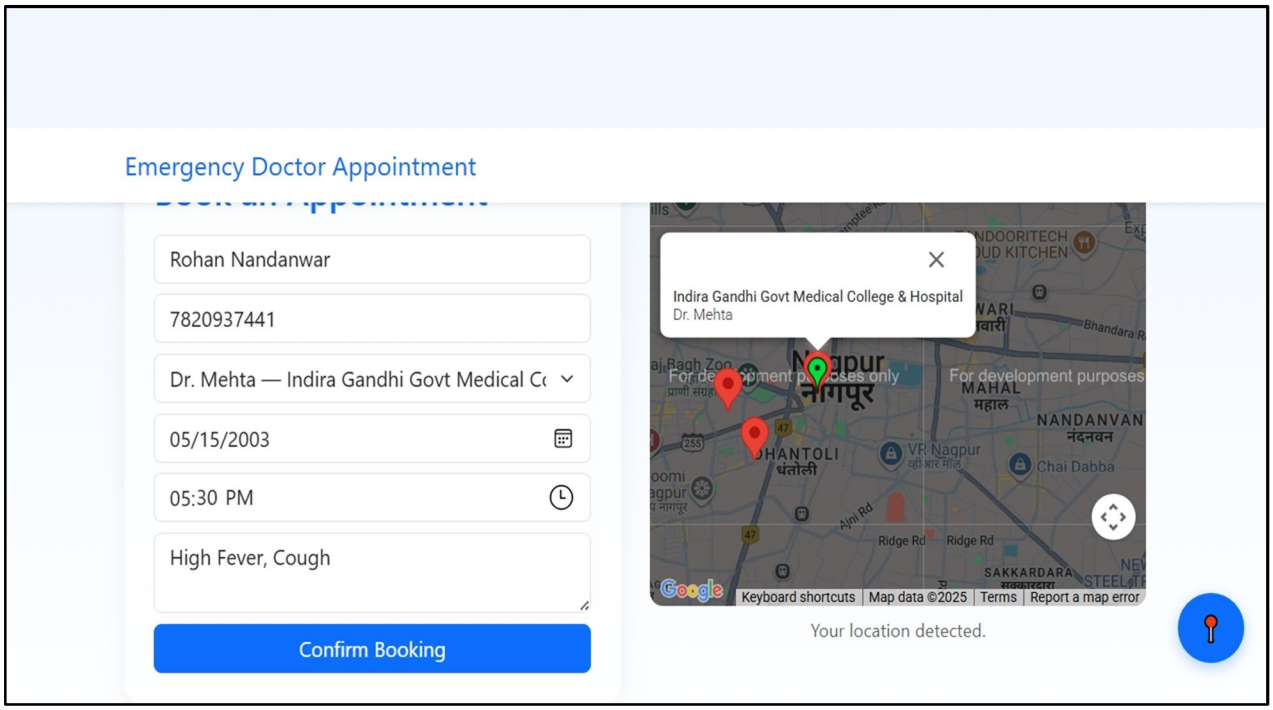


Fig:-5.5 Emergency booking part 2

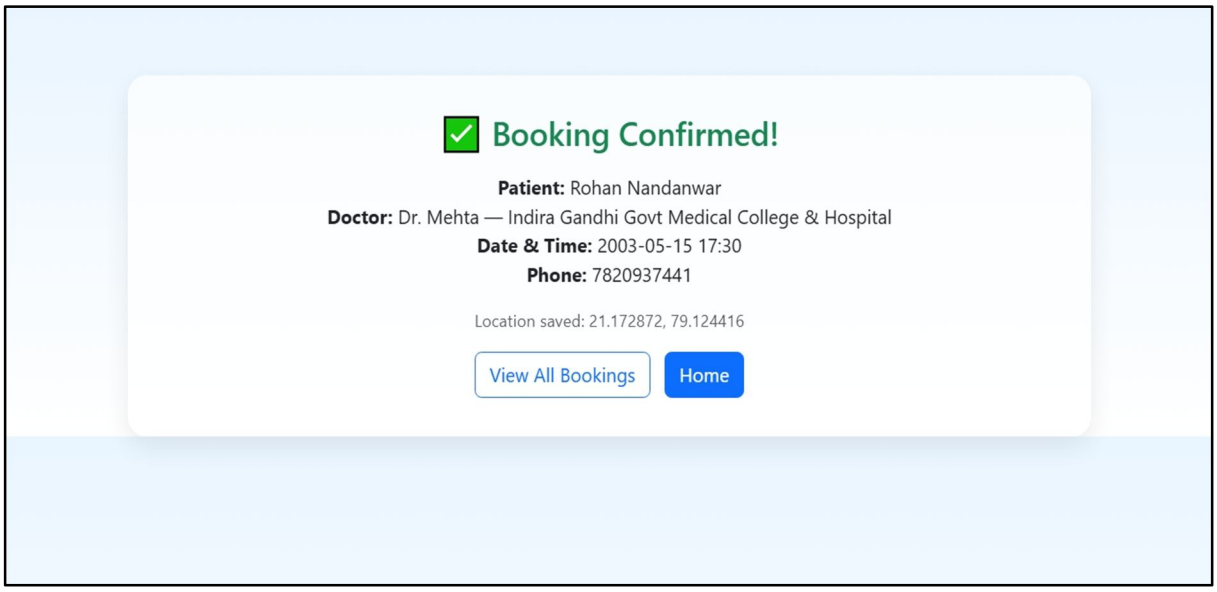


Fig:-5.6 Emergency booking confirmation

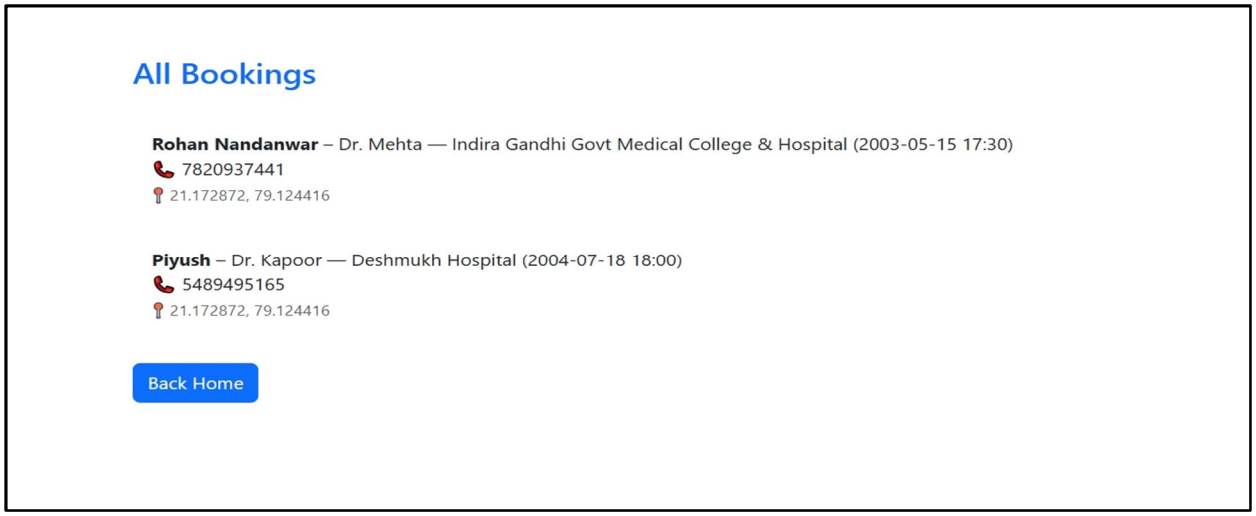


Fig:-5.7 List of emergency booking

VI. FUTURE SCOPE

1) Integration with Wearable Devices: In the future, the system can be integrated with smart health devices and wearables (like smartwatches, heart-rate monitors, or oxygen sensors) to automatically collect real-time health data and detect emergencies instantly.
2) Voice-Based Appointment Booking: The chatbot can be enhanced to support voice recognition and multilingual interaction, allowing users to book emergency appointments hands-free — useful for elderly or physically weak patients.
3) AI-Driven Diagnosis Suggestions: Advanced AI models can be implemented to provide preliminary diagnosis or treatment advice based on symptoms and past medical records before consulting the doctor.
4) Integration with Hospital Management Systems (HMS): The project can be expanded to integrate with existing hospital databases for automatic doctor allocation, record updates, and billing systems for a complete end-to-end healthcare solution.
5) Emergency Vehicle and Location Tracking: The system can connect with ambulance services and GPS modules to dispatch emergency vehicles automatically when the patient's condition is critical.
6) Predictive Analytics for Healthcare Resources: Using AI, hospitals can forecast patient flow, doctor workload, and emergency case surges to optimize staffing and resource planning.

7) Mobile Application Development: Developing a dedicated mobile app for both patients and doctors would make the system more accessible, providing real-time notifications and updates about appointments or emergencies.

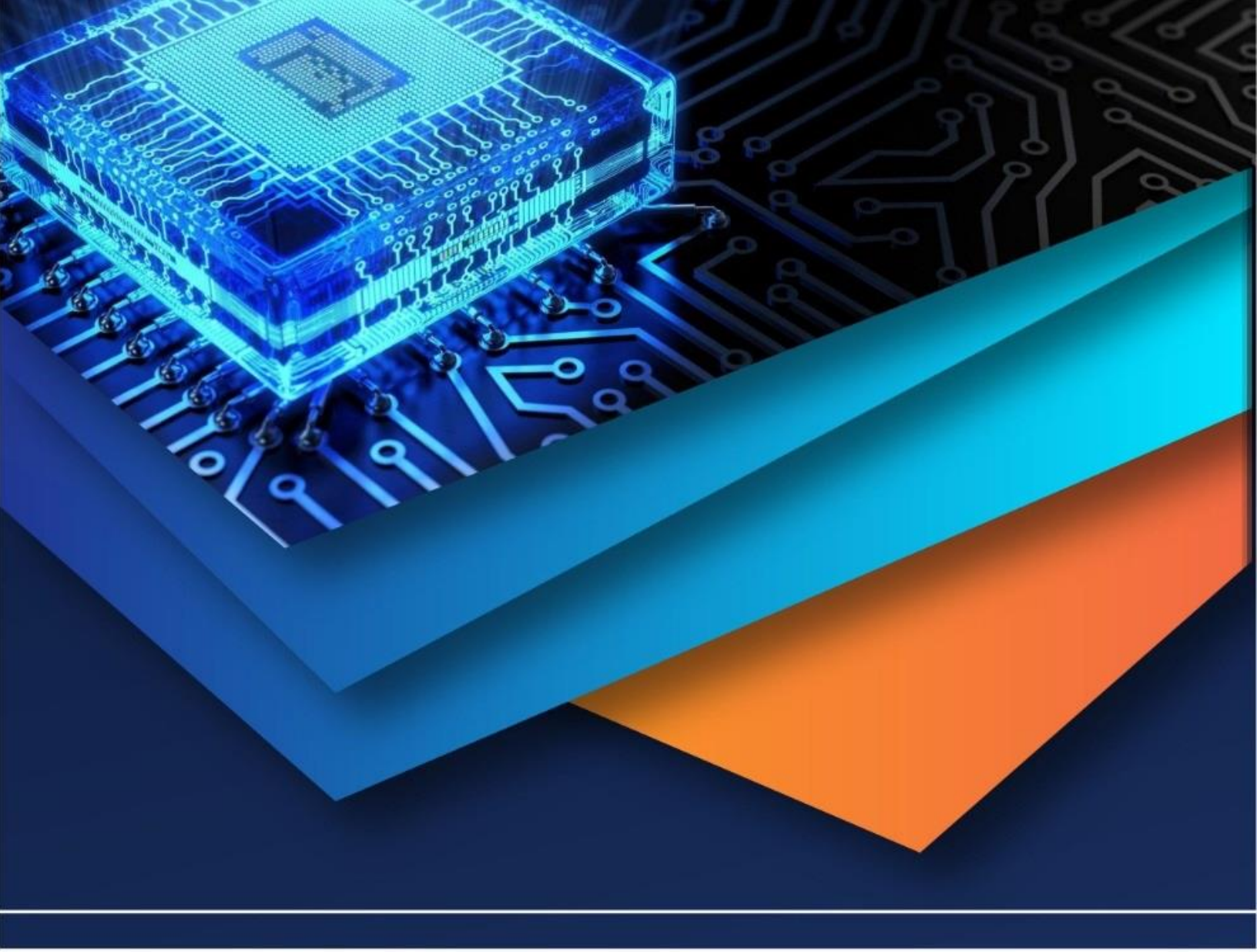
8) Data Security and Blockchain Integration: Future versions can include blockchain technology for secure and transparent storage of patient medical data and appointment records.

VII. ACKNOWLEDGMENT

The attainment and final outcome of this project required a lot of supervision and assistance from many people and I am extremely privileged to have got this all along the completion of our project. Whatever I have done is only due to such guidance and assistance and I would not forget to express thanks to them. My earnest gratitude goes to my Guide, Asst. Prof. Kalpana Dongre, Assistant Professor, Department of Artificial Intelligence & Data Science, for her support, guidance, inspiration and encouragement throughout the period this work was carried out. Her willingness for meeting at all times, her lucrative comments, her concern and support even with practical things have been very helpful. I express my gratitude to our H.O.D. Dr. Aparna Gale, for his encouragement and guidance to complete our project work. My sincere thanks to the Director Dr. Sangita Deshmukh, for providing me necessary facilities to carry out the work. I would also like to thank all instructors and professors, and members of the department of Artificial Intelligence & Data Science for their warm help in various ways for the completion of this thesis.
Last, but not the least, I would like to a vote of appreciation for my Parents and fellow friends for their cooperation.

REFERENCES

[1] TensorFlow Documentation. (n.d.). Retrieved from https://www.tensorflow.org/ This source provides detailed information about AI model building, training, and deployment, which was useful for developing the symptom analysis module in the project.
[2] Scikit-learn Documentation. (n.d.). Retrieved from https://scikit-learn.org/stable/ Used as a reference for implementing machine learning algorithms for patient data classification and urgency prediction.
[3] Dialogflow Documentation. (n.d.). Retrieved from https://cloud.google.com/dialogflow/docs Helpful for designing the AI chatbot interaction system used to collect and analyze patient symptoms.
[4] MySQL Documentation. (n.d.). Retrieved from https://dev.mysql.com/doc/ Provided guidance on creating and managing the project database for storing patient, doctor, and appointment information.
[5] Google Maps Platform Documentation. (n.d.). Retrieved from https://developers.google.com/maps/documentation used for integrating location-based services, enabling users to find the nearest available emergency doctor or hospital and visualize routes using the map API.
[6] AI-Driven Doctor Scheduling System Research Paper. (2025). Retrieved from https://irjaeh.com/index.php/journal/article/view/924 This paper supported the conceptual design of the AI scheduling model, including location-based hospital search, specialist recommendation, and optimized appointment allocation.
[7] Outpatient Scheduling Using Deep Reinforcement Learning. (2024). Retrieved from https://link.springer.com/article/10.1007/s10791-024-09474-1 This study supported the appointment-optimization logic for efficient doctor allocation, especially during emergency or high-demand periods.
[8] Google Firebase Documentation. (n.d.). Retrieved fromhttps://firebase.google.com/docs Useful for integrating authentication, cloud storage, and real-time database components support secure patient login and appointment data synchronization.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY