



PATIENT ALLOCATION SYSTEM FOR DOCTORS USING MACHINE LEARNING

A MINI PROJECT REPORT

Submitted by

KEVIN JOE CLIFFORD.J [REGISTER NO:211421104128]

KISHORE.K [REGISTER NO:211421104131]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

OCTOBER 2023

BONAFIDE CERTIFICATE

Certified that this project report “ **PATIENT ALLOCATION SYSTEM FOR DOCTORS USING MACHINE LEARNING** ” Is the bonafide work of “**KEVIN JOE CLIFFORD.J [REGISTER NO:211421104128] KISHORE.K [REGISTER NO:211421104131]**”who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA.,M.E.,Ph.D.
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mrs. P. Deepa
Associate Professor

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We KEVIN JOE CLIFFORD.J [REGISTER NO: 211421104128], KISHORE.K [REGISTER NO:211421104131] hereby declare that this project report titled **“PATIENT ALLOCATION SYSTEM FOR DOCTORS USING MACHINE LEARNING”**, under the guidance of Mrs. P. Deepa Is the original work done by us and we have not plagiarized or submitted to any Other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.**, for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. .JABASHEELA , M.E.,Ph.D.**, for the support extended throughout the project.

We would like to thank my **Project Guide Mrs. P. Deepa** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

KEVIN JOE CLIFFORD.J [REGISTER NO: 211421104128]
KISHORE.K [REGISTER NO: 211421104131]

ABSTRACT

The "Patient Allocation System For Doctors Using Machine Learning " is a web-based application designed to streamline the process of booking medical appointments with different doctors. This project combines web development and computer vision techniques to create a user-friendly interface for patients seeking medical care. The application features a responsive web form that allows users to input their personal information, including name, age, gender, email, phone number, preferred doctor, and appointment date. The interface is designed to be intuitive and accessible on various devices. Behind the scenes, the application utilizes OpenCV for real-time face detection from camera feeds to facilitate the appointment booking process. When a patient submits their information, the system detects and assigns them to an available doctor based on their choice or the doctor with the fewest patients, ensuring efficient allocation of medical resources. The project also incorporates data storage functionality using Excel sheets, where patient details are securely stored for future reference. This allows for easy tracking of appointments and patient records. Overall, the "Doctor Appointment Scheduler" provides a practical solution for managing medical appointments, optimizing doctor-patient allocation, and maintaining organized patient records. It showcases the integration of web development, computer vision, and data storage to improve the efficiency of healthcare appointment systems.

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1	ER Diagram	5
2	Data Directory	5
3	Data flow Diagram	5
4	Architecture Diagram	6
5	Output Screenshot -Appointment screen	26
6	Output Screenshot-Appointment successful	26
7	Output Screenshot-Doctor 1	27
8	Output Screenshot-Doctor 2	27
9	Output Screenshot-Doctor 3	27

LIST OF SYMBOLS, ABBREVIATIONS

Flask:	A Python web framework used for building web applications.
cv2:	An abbreviation for OpenCV, an open-source computer vision library often used for image and video processing.
XML:	Extensible Markup Language, a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable
URL:	Uniform Resource Locator, a reference or address used to access resources on the internet.
HTML:	HyperText Markup Language, a standard markup language used to create web pages.
CSS:	Cascading Style Sheets, a style sheet language used for describing the presentation of a document written in HTML.
POST:	One of the HTTP methods used to submit data to be processed to a specified resource.
GET:	Another HTTP method used to request data from a specified resource
Excel:	A spreadsheet application developed by Microsoft often used for data storage and manipulation.

Abbreviations in the HTML:

- **Doctor Specializations:** In your HTML form, you have options like "General Doctor," "Cardiologist," "Radiologist," and "Neurologist," which likely represent different medical specialties.
- **CSS Selectors:** In your CSS code, you've used various selectors like #body2, .container, .input-box, .gender-title, .gender-category, .button-container, etc., to style specific HTML elements.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF FIGURES	VI
	LIST OF SYMBOLS AND ABBREVIATION	VII
1.	INTRODUCTION	
1.1	Problem Definition	1
2.	LITERATURE SURVEY	2
3.	SYSTEM ANALYSIS	
3.1	Existing System	3
3.2	Proposed system	3
3.3	Hardware Environment	4
3.4	Software Environment	4
4.	SYSTEM DESIGN	
4.1	ER diagram	5
4.2	Data Directory	5
4.3	Data Flow Diagram	5
5.	SYSTEM ARCHITECTURE	
5.1	Architecture Overview	6
5.2	Module Design Specification	7
5.3	Algorithms	8
6.	SYSTEM IMPLEMENTATION	
6.1	Client-side coding	9
6.2	Server-side coding	14

CHAPTER NO.	TITLE	PAGE NO.
7.	SYSTEM TESTING	
7.1	Unit Testing	21
7.2	Integration Testing	21
7.3	Test Cases	22
8.	CONCLUSION	
8.1	Conclusion and Future Enhancements	25
	APPENDICES	
	A.1 Sample Screens	26
	REFERENCES	28

1. INTRODUCTION

In today's fast-paced world, access to healthcare is paramount, and the efficiency of medical services is a critical factor in ensuring the well-being of individuals and communities. The "Patient Allocation System For Doctors Using Machine Learning" is a pioneering web-based application that addresses the challenges associated with booking medical appointments by seamlessly integrating modern technology with healthcare management.

This project is a testament to the power of innovation in improving healthcare accessibility and efficiency. It combines elements of web development, computer vision, and data management to provide a comprehensive solution for both patients and healthcare providers.

The Need for Streamlined Appointment Booking:

Traditionally, scheduling a medical appointment has been a cumbersome and often time-consuming process, characterized by paperwork, long wait times, and uneven distribution of appointments among healthcare providers. Patients may face challenges in securing timely appointments, while doctors may experience an uneven patient load.

Key Features of the Doctor Appointment Scheduler:

User-Friendly Interface: The heart of this project is an intuitive and responsive web form. Patients can easily enter their personal information, specify their preferred doctor, choose an appointment date, and complete the booking process with just a few clicks.

Efficient Doctor Assignment: Behind the scenes, cutting-edge computer vision techniques powered by OpenCV are employed to enhance the appointment booking process. The system not only ensures the security and identity of patients but also intelligently assigns patients to doctors. Patients can choose their preferred doctor, but in cases of no preference, the system dynamically allocates appointments to ensure an equitable distribution among healthcare providers.

Data Management: To maintain organized patient records, the "Doctor Appointment Scheduler" incorporates data storage capabilities using Excel sheets. This feature simplifies the tracking of appointments and offers valuable insights for optimizing medical services.

1.1 Problem definition

In the realm of healthcare, the efficient management of patient appointments and doctor allocations is a persistent challenge. Patients often face hurdles when attempting to schedule appointments, and doctors may experience an uneven distribution of patients across their schedules. This inefficiency can lead to longer wait times for patients, overburdened doctors, and an overall suboptimal healthcare experience.

The core problem addressed by the "Doctor Appointment Scheduler" project is the need for a streamlined and equitable appointment booking system that optimizes the allocation of patients to healthcare providers. The key issues encompassed by this problem include:

Appointment Accessibility: Patients encounter difficulties in securing timely appointments, which can lead to delays in medical care and potential health risks.

Doctor Workload: Healthcare providers may experience variations in patient loads, with some doctors being overbooked while others have available time slots.

Patient Data Management: The manual recording and organization of patient data can be error-prone and time-consuming, leading to inefficiencies in maintaining accurate records.

Choice and Allocation: Ensuring that patients can choose their preferred doctor while maintaining a fair distribution of appointments among healthcare providers is a complex task.

Optimization: Maximizing the utilization of healthcare resources, such as doctor availability, is crucial for efficient healthcare delivery.

By addressing these challenges, the project intends to revolutionize the way patients book medical appointments, making it a more efficient and equitable process that benefits both patients and healthcare providers. It seeks to pave the way for a future where technology and healthcare seamlessly merge to enhance the overall healthcare experience.

2. LITERATURE SURVEY

Introduction:

The integration of computer vision technology into healthcare appointment systems is a burgeoning field with the potential to enhance security, efficiency, and user experience. This literature survey explores existing research and developments in this area, providing insights into key findings, trends, and challenges.

1. Computer Vision Applications in Healthcare:

Summary: Computer vision plays a pivotal role in healthcare, enabling applications such as patient identification, monitoring, and diagnosis.

Relevance: Understanding the broader context of computer vision in healthcare is essential to appreciate its role in appointment systems.

2. Facial Recognition in Healthcare:

Summary: Facial recognition technology is increasingly used in healthcare settings for patient identification and authentication.

Relevance: Examining the application of facial recognition provides insight into the specific technology employed in appointment systems.

3. Real-Time Face Detection for Appointment Booking:

Summary: Real-time face detection is utilized to enhance the security and efficiency of appointment scheduling and patient management.

Relevance: Investigating the effectiveness of face detection techniques is crucial for evaluating their impact on appointment systems.

4. Security and Data Privacy:

Summary: The integration of computer vision introduces security and data privacy considerations, including compliance with regulations like HIPAA.

Relevance: Understanding the regulatory landscape and security measures is essential for responsible implementation.

5. User Experience and Acceptance:

Summary: User perception and acceptance of computer vision-based appointment systems influence their adoption and success.

Relevance: Analyzing user feedback provides insights into the usability and user satisfaction aspects of these systems.

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Existing System Overview:

- **Electronic Health Record (EHR) Systems:** Many healthcare facilities use EHR systems with basic patient allocation features for appointment scheduling and patient data management.
- **Hospital Information Systems (HIS):** Comprehensive software used by hospitals, including advanced patient allocation capabilities for scheduling, resource management, and billing.
- **Clinical Decision Support Systems (CDSS):** Assist healthcare professionals in making clinical decisions, often including patient prioritization based on medical urgency.
- **Emergency Department Information Systems (EDIS):** Tailored for emergency departments to manage patient triage and allocation during emergencies.
- **Bed Management Systems:** Focus on optimizing bed allocation within hospitals, tracking available beds and coordinating patient assignments and transfers.

3.2 PROPOSED SYSTEM

Proposed System Overview:

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

- Security of data.
- Ensure data accuracy's.
- Proper control of the higher officials.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive.
- Minimum time required

3.3 Hardware Requirements:

1. **Computer:**
You'll need a computer (PC or laptop) capable of running Python and OpenCV. The specific hardware requirements depend on the complexity of the image processing and video streaming tasks you plan to perform. For basic functionality, a modern computer with a webcam should suffice.
2. **Webcam:**
For face detection, you'll need an integrated or external webcam. Ensure that the webcam is functional and correctly connected to your computer.
3. **Internet Connection:**
If you intend to use video streaming from IP cameras, you'll need a stable internet connection to access the camera's feed.

3.4 Software Requirements:

1. **Python:**
You must have Python installed on your computer. The code appears to be written in Python 3. Ensure that you have Python 3.x installed.
2. **OpenCV:**
The code relies on OpenCV for image processing and video capture. You should install OpenCV for Python.
3. **OpenPyXL:**
This library is used for interacting with Excel files.
4. **Flask:**
If you plan to run the Flask web application at the end of your code, you'll need Flask.
5. **Web Browser:**
To interact with the Flask web application, you'll need a web browser like Chrome, Firefox, or Safari.
6. **Web Camera URL:**
In your code, you've specified URLs for video streams from IP cameras. Ensure that these URLs are valid and accessible on your network.
7. **HTML Templates:**
If you have HTML templates (e.g., 'index.html') for the Flask application, make sure they are available and correctly referenced in your Flask code.
8. **Operating System:** The code is written in a platform-independent way and should work on Windows, macOS, or Linux.

4.SYSTEM DESIGN

4.1. ER diagram

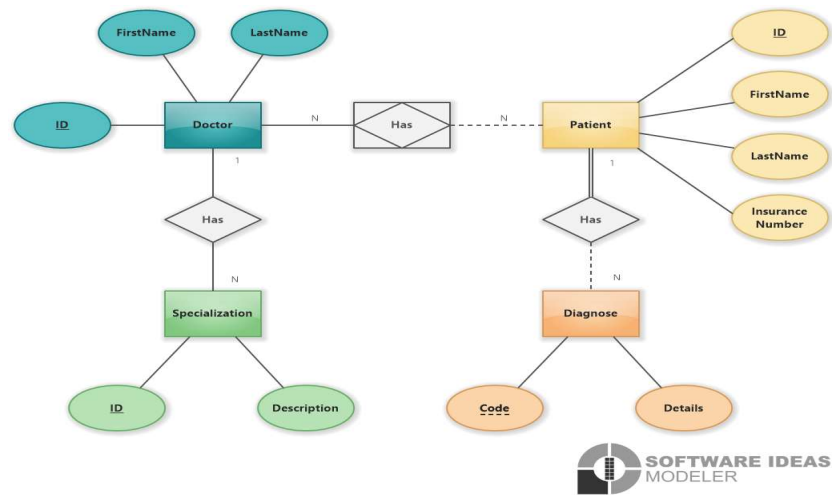


Figure 1: ER Diagram

4.2 Data dictionary

The screenshot shows a spreadsheet application with a data dictionary table. The table has columns for Name, Age, Sex, Phone, and Email. The data is as follows:

	A	B	C	D	E	F	G
1	Name	Age	Sex	Phone	Nur	Email	
2	Kevin Joe	55	male	94447327	:	joekevin40@gmail.com	
3	Kevin Joe	55	male	94447327	:	joekevin40@gmail.com	
4	Kevin Joe	55	male	94447327	:	joekevin40@gmail.com	

Figure 2: Data Dictionary

4.3 Data flow diagram

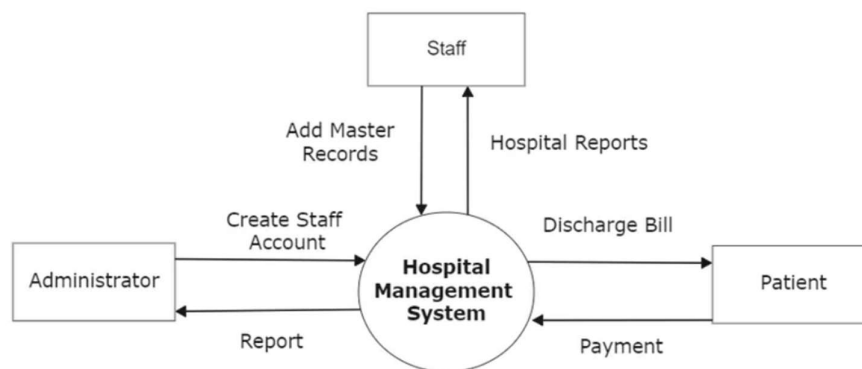


Figure 3: Data Flow Diagram

5.SYSTEM ARCHITECTURE

5.1 Architecture Overview:

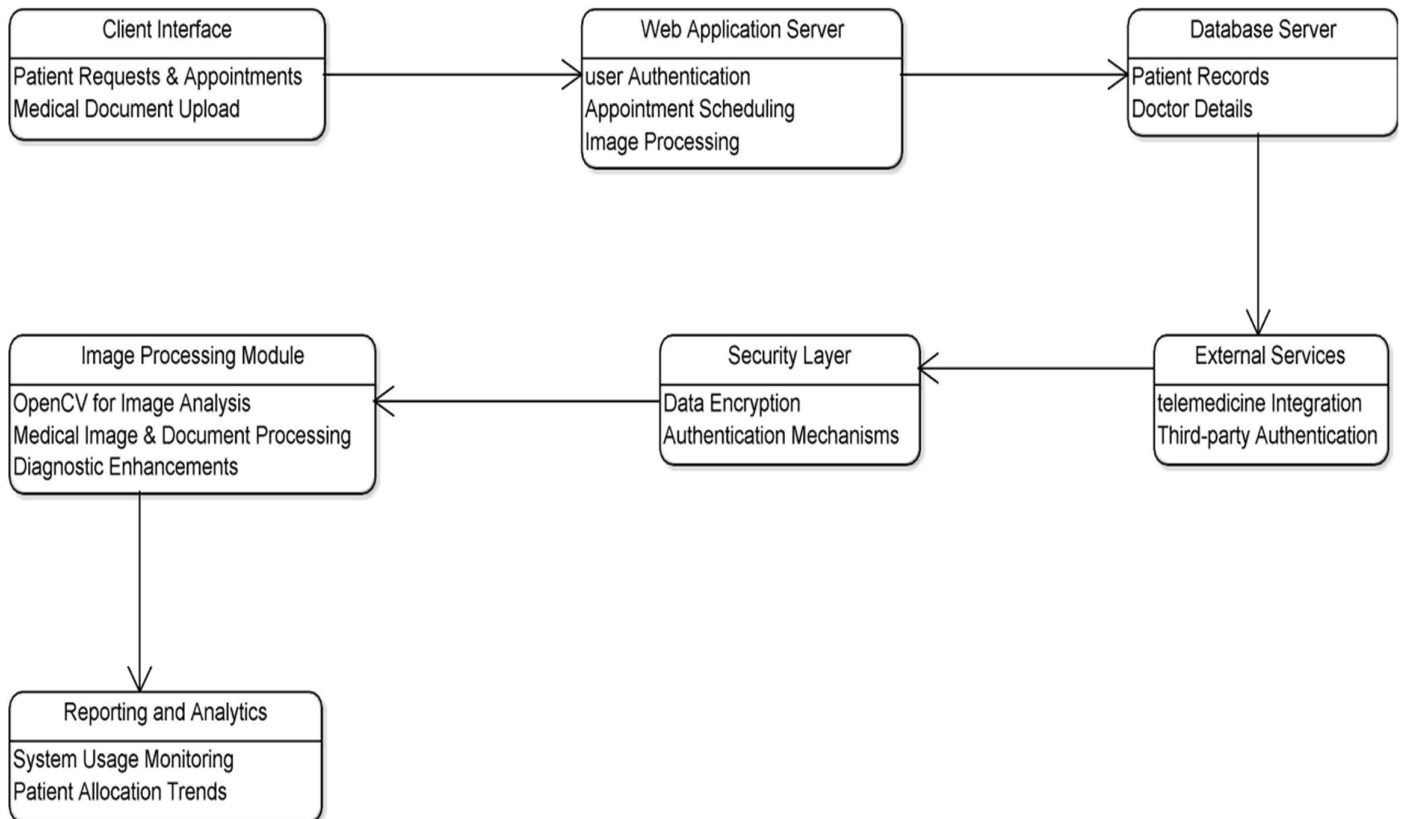


Figure 4: Architecture Diagram

5.2 Module Design Specification

1. User Authentication Module:

- *Description:* This module handles user registration and login. It ensures secure access to the system, distinguishing between doctors and patients.
- *Functionality:* Allows users to create accounts, login securely, and reset passwords when necessary.

2. Appointment Scheduling Module:

- *Description:* This module enables patients to request appointments and doctors to manage their schedules.
- *Functionality:* Patients can view available slots, request appointments, and receive confirmation. Doctors can accept, reject, or reschedule appointments.

3. Patient Allocation Module:

- *Description:* This module automates the allocation of patients to available doctors based on various criteria such as specialization, location, and availability.
- *Functionality:* Efficiently matches patients with doctors, reducing wait times and optimizing resource allocation.

4. Electronic Health Records (EHR) Module:

- *Description:* The EHR module centralizes patient medical records, ensuring secure storage and accessibility.
- *Functionality:* Allows authorized healthcare providers to view and update patient records, upload medical documents, and maintain comprehensive medical histories.

5. Image Processing Module (OpenCV Integration):

- *Description:* Integrated with OpenCV, this module analyzes medical images and documents uploaded by patients for diagnostic purposes.
- *Functionality:* Provides image analysis capabilities, assisting doctors in interpreting X-rays, scans, and other medical images.

6. Telemedicine Integration Module:

- *Description:* This module facilitates remote consultations between doctors and patients through telemedicine services.
- *Functionality:* Supports video conferencing, secure messaging, and document sharing, enhancing accessibility to healthcare services.

7. Reporting and Analytics Module:

- *Description:* This module generates reports and provides data analytics to assist healthcare providers in monitoring system usage and allocation trends.
- *Functionality:* Offers insights into patient allocation patterns, appointment data, and system performance.

5.3 Algorithms:

1. Data Preparation:

- Maintain lists of available doctors and patients with their respective information.

2. Patient Request:

- Gather patient information, including medical condition and preferences (if any).

3. Doctor Selection:

- Match patient's medical condition with doctors' specializations.
- Consider doctor availability and schedule.
- Check for any specific doctor preferences from the patient.
- Rank doctors based on suitability for the patient's needs using a scoring system or criteria.

4. Appointment Booking:

- Book an appointment with the selected doctor at an agreed-upon time.
- Update the doctor's schedule to reflect the booked appointment.

5. Confirmation and Notification:

- Send appointment details and confirmations to both the patient and the doctor.
- Notify the patient if their preferred doctor is unavailable and provide details of the assigned doctor.

6. Feedback and Follow-up:

- After the appointment, collect feedback from the patient.
- Use this feedback to improve the allocation process and enhance the patient experience.

6. SYSTEM IMPLEMENTATION

6.1 Client-side Coding (Sample - HTML/CSS/JavaScript):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Doctor Appointment Scheduler</title>
</head>

<body id="body2">

  <div class="container">
    <form action="#" method="POST">

      <h2>Book an Appointment</h2>
      <div class="content">

        <div class="input-box">
          <label for="name">Your Name</label>
          <input type="text" id="name" name="name" required>
        </div>

        <div class="input-box">
          <label for="email">Email</label>
          <input type="email" id="email" name="email" required>
        </div>

        <div class="input-box">
          <label for="phone">Phone Number</label>
          <input type="tel" id="phone" name="phone" required>
        </div>

        <div class="input-box">
          <label for="age">Age</label>
          <input type="text" id="age" name="age" required>
        </div>

        <div class="input-box">
          <label for="doctor">Select a Doctor</label>
          <select id="doctor" name="doctor" required>
            <option id="options" value="General Doctor">General Doctor</option>
            <option id="options" value="Cardiologist">Cardiologist</option>
            <option id="options" value="Radiologist">Radiologist</option>
            <option id="options" value="Neurologist">Neurologist</option>
          </select>
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

```

        </select>
    </div>

    <div class="input-box">
        <label for="date">Select a Date</label>
        <input type="date" id="date" name="date" required>
    </div>
    <span class="gender-title">Gender</span>
    <div class="gender-category">

        <input type="radio" id="male" name="gender" value="male" required>
        <label for="male">Male</label>

        <input type="radio" id="female" name="gender" value="female" required>
        <label for="female">Female</label>

        <input type="radio" id="other" name="gender" value="other" required>
        <label for="other">Other</label>
    </div>
    <div class="button-container">
        <button type="submit">Book Appointment</button>
    </div>
</form>
</div>

```

```

<style>
*{
padding: 0;
margin: 0;
font-family: sans-serif;
box-sizing: border-box;
}

#body2{
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
background-color:#0CBABA ; /*linear-gradient(#380036,#0CBABA);*/
}

.container{
max-width: 650px;
padding: 28px;
margin: 0 28px;
border-radius: 10px;
overflow: hidden;
background: rgba(0,0,0,0.2);

```

```
box-shadow: 0 15px 20px rgba(0,0,0,0.6);
/* background-image: linear-gradient(rgba(4,9,30,0.7),rgba(4,9,30,0.7)),url("main 1b.png"); */
}

h2{
  font-size: 26px;
  font-weight: bold;
  text-align: left;
  color: #41303ffb;
  padding-bottom: 8px;
  border-bottom: 1px solid silver;
}

.content{
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  padding: 20px 0px;
}

.input-box{
  display: flex;
  flex-wrap: wrap;
  width: 50%;
  padding-bottom: 15px;
}

.input-box:nth-child(2n){
  justify-content: end;
}

.input-box label, .gender-title{
  width: 95%;
  color: #fff;
  font-weight: bold;
  margin: 5px 0px;
}

.gender-title{
  font-size: 16px;
}

.input-box input, select{
  background-color: #fff;
  height: 40px;
  width: 95%;
  padding: 0 10px;
  border-radius: 0px;
```

```
border: 1px solid #fff;
outline: none;
}
```

```
#options{
  font-size: 15px;
}
```

```
.input-box input:hover{
  border-color: black;
}
select{
  background-color: #fff;
}
select:hover{
  border-color: black;
}
```

```
#options:hover{
  cursor: pointer;
}
```

```
.gender-category label{
  padding: 0 20px 0 5px;
  font-size: 14px;
}
```

```
.gender-category{
  color: #fff;
}
```

```
.gender-category input{
  color: black;
}
```

```
.gender-category label, .gender-category input{
  cursor: pointer;
}
```

```
.button-container{
  margin: 15px;
  display: flex;
  justify-content: center;
  align-items: center;
}
.button-container button{
  width: 100%;
```

```
padding: 10px;
font-size: 20px;
border: 1px solid black;
/* color: #fff; */
border-radius: 5px;
background-color: transparent;
transition: 0.3s;
}
button:hover{
  cursor: pointer;
  border: 1px solid #fff;
  background: #fff;

}
@media(max-width: 600px)
{
  .container{
    min-width: 280px;
  }
  .content{
    max-height: 380px;
    overflow: auto;
  }
  .input-box{
    margin-bottom: 12px;
    width: 100%;
  }
  .input-box:nth-child(2n){
    justify-content: space-between;

  }
  .gender-category{
    display: flex;
    justify-content: space-between;
    width: 60%;
  }
  .content::-webkit-scrollbar{
    width:3;
  }
}

</style>

</body>
</html>
```

6.2 Server-side Coding (Sample - Python using Flask):

MAIN.PY

```
import cv
import openpyxl

def detect_and_return_face(a):

    # Load the pre-trained Haar Cascade face detector
    face_cascade = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_frontalface_default.xml')
    flag=False
    # Initialize the camera (0 is typically the default webcam)
    cap = cv.VideoCapture(a)
    if not cap.isOpened():
        return False
    t= 1000
    while True:
        # Read a frame from the camera
        ret, frame = cap.read()

        if not ret:
            flag = False
            break

        # Convert the frame to grayscale for face detection
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

        # Detect faces in the frame# Detect faces with adjusted parameters
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(50, 50))

        # Check if at least one face is detected
        if len(faces) > 0:
            for (x, y, w, h) in faces:
                cv.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

        # Display the frame with detected faces
        cv.imshow('Face Detection', frame)
```



```
# Release the camera and close all OpenCV windows
```

```
cap.release()
```

```
cv.destroyAllWindows()
```

```
print("Face detected!")
```

```
flag = True
```

```
break
```

```
# Display the frame
```

```
cv.imshow('Face Detection', frame)
```

```
# Exit the loop when the 'q' key is pressed
```

```
if cv.waitKey(1) & 0xFF == ord('q'):
```

```
    break
```

```
t=t-1
```

```
if t==0:
```

```
    flag = False
```

```
    print("I")
```

```
    break
```

```
# Release the camera and close all OpenCV windows if no face is detected
```

```
cap.release()
```

```
cv.destroyAllWindows()
```

```
return flag
```

```
def create_excel_sheet(filename, sheetname):
```

```
    workbook = openpyxl.Workbook()
```

```
    sheet = workbook.active
```

```
    sheet.title = sheetname
```

```
# Labels (headers) for each column
```

```
labels = ["Name", "Age", "Sex", "Phone Number", "Email"]
```

```
sheet.append(labels)
```

```
workbook.save(filename)
```

```
print("creation successful")
```

```
def insert_patient_info(filename, sheetname, data):
```

try:

```
workbook = openpyxl.load_workbook(filename)
```

```
sheet = workbook[sheetname]
```

except FileNotFoundError:

```
create_excel_sheet(filename, sheetname)
```

```
workbook = openpyxl.load_workbook(filename)
```

```
sheet = workbook[sheetname]
```

```
sheet.append(data)
```

```
workbook.save(filename)
```

```
print("Patient information saved successfully!")
```

```
def patient_details():
```

```
    name = str(input("Enter name:"))
```

```
    age = int(input("Enter age:"))
```

```
    sex = input("enter sex:")
```

```
    p_no= input("Enter nobile number:")
```

```
    e_mail = input("Enter the email:")
```

```
    details = [name,age,sex,p_no,e_mail]
```

```
    return details
```

```
    # main block
```

```
if __name__ == "__main__":
```

```
    doctor1_1 = []
```

```
    temp_doctor1 = []
```

```
    doctor2_1 = []
```

```
    temp_doctor2 = []
```

```
    doctor3_1 = []
```

```
    temp_doctor3 = []
```

```
    dict = {}
```

```
    create_excel_sheet("doctor_1.xlsx","patients")
```

```
    create_excel_sheet("doctor_2.xlsx","patients")
```

```
    create_excel_sheet("doctor_3.xlsx","patients")
```

```
    camera1_url = 'http://192.168.108.8:4747/video'
```

```
    camera2_url = 'http://192.168.108.52:4747/video'
```

```
    camera3_url = 'http://192.168.1.158:4747/video'
```

```

while True:
    print("Enter the choice:")
    print("1.Out pasien")
    print("2.personal appointment")
    print("3.exit")
    inp = int(input())
    details=[]
    temp=[]
    doctors_1 = []

    if inp == 1:
        details=patient_details()
        try:
            doctor_1 = detect_and_return_face(camera1_url )
            if doctor_1:

                doctors_1.append(doctor1_1)
        except:
            print("cam o is invalid")
        try:
            doctor_2 = detect_and_return_face(camera2_url)
            if doctor_2:

                doctors_1.append(doctor2_1)

        except:
            print("cam 1 is invalid")
        try:
            doctor_3 = detect_and_return_face(2)
            if doctor_3:
                doctors_1.append(doctor3_1)

        except:
            print("cam 2 is invalid")
    print(doctor_1,doctor_2,doctor_3)

```

```

print(doctors_1)
if doctors_1 == []:
    print("No doctor is available!!!!")

else:
    print(doctor1_1)
    print(temp_doctor1)

# doctors_1 = [item for item in my_list if item != False]
# doctors = {key: value for key, value in doctors.items() if value != False}
lis=[]
doctors_1[(lis:= [len(doctors_1[i]) for i in range(len(doctors_1))]).index(min(lis))].append(details[0])
print(doctor1_1)
print(doctor2_1)
print(doctor3_1)
print(temp_doctor1)
print(temp_doctor2)
print(temp_doctor3)
if doctor1_1 != temp_doctor1:
    insert_patient_info("doctor_1.xlsx", "patients", details)
    temp = temp_doctor1
    doctor1_1 = temp

if doctor2_1 != temp_doctor2:
    insert_patient_info("doctor_2.xlsx", "patients", details)
    temp1=temp_doctor2
    doctor2_1 = temp1

if doctor3_1 != temp_doctor3:
    insert_patient_info("doctor_3.xlsx", "patients", details)
    temp2 = temp_doctor3
    doctor3_1 = temp2

elif inp ==2:
    detail=patient_details()
    try:
        doctor_1 = detect_and_return_face(camera1_url )

```

```

if doctor_1:

    doctors_1.append("doctor1")
except:
    print("cam o is invalid")
try:
    doctor_2 = detect_and_return_face(camera2_url)
    if doctor_2:

        doctors_1.append("doctor2")

except:
    print("cam 1 is invalid")
try:
    doctor_3 = detect_and_return_face(2)
    if doctor_3:
        doctors_1.append("doctor3")

except:
    print("cam 2 is invalid")
if doctors_1 == []:
    print("No doctor is available!!!!")

else:
    print(doctor1_1)
    print(temp_doctor1)
    print("doctors are detected ")
    print("choose an option")
    for k in doctors_1:
        print(k)

inp = int(input())
if inp== 1:
    insert_patient_info("doctor_1.xlsx", "patients", details)
elif inp== 2:
    insert_patient_info("doctor_2.xlsx", "patients", details)

```

```
        else:
            insert_patient_info("doctor_3.xlsx", "patients", details)
    else:
        break
```

APP.PY

```
from flask import Flask, request, render_template
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def index():

    if request.method == 'POST':
        # Get data from the HTML form
        name = request.form['name']
        email = request.form['email']
        phone = request.form['phone']
        age = request.form['age']
        doctor = request.form['doctor']
        date = request.form['date']
        gender = request.form['gender']

        # You can now process the form data as needed
        # For example, you can print it to the console

        print(f'Name: {name}, Email: {email}, Phone: {phone}, Age: {age}, Doctor: {doctor}, Date: {date},
Gender: {gender}')

        # You can also store the data in a database or perform other actions here
        return render_template('index.html')

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

7. SYSTEM TESTING

7.1 Unit Testing:

Each module is considered independently. it focuses on each unit of software as implemented in the source code. it is white box testing.

White Box Testing:

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions or loops. all logical parts of the software checked once. errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decisions on their true and the false side are exercised , all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

7.2 Integration Testing:

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. modules are integrated by using the top down approach.

7.3 Test Cases:

Test Case 1: Successful Allocation

<ul style="list-style-type: none">• Objective:	<ul style="list-style-type: none">• Verify that the system successfully allocates a patient to an available doctor
<ul style="list-style-type: none">• Preconditions:	<ul style="list-style-type: none">• The system is operational.• The doctor and patient profiles are set up.• The patient has submitted an appointment request
<ul style="list-style-type: none">• Test Data:	<ul style="list-style-type: none">• Doctor specialization matches the patient's medical condition.• Doctor is available.• Patient preferences match the doctor's availability.
<ul style="list-style-type: none">• Steps to Execute:	<ul style="list-style-type: none">• Login as an administrator.• Access the patient's appointment request.• Initiate the allocation process.
<ul style="list-style-type: none">• Expected Results:	<ul style="list-style-type: none">• The system successfully assigns the patient to an available doctor with matching specialization and preferences.• The patient receives an appointment confirmation.

Test Case 2: Handling Invalid Patient Request

<ul style="list-style-type: none">• Objective:	<ul style="list-style-type: none">• Verify that the system handles an invalid patient request.
<ul style="list-style-type: none">• Preconditions:	<ul style="list-style-type: none">• The system is operational.• The patient's appointment request is incomplete or contains incorrect information.
<ul style="list-style-type: none">• Test Data:	<ul style="list-style-type: none">• Invalid patient information.
<ul style="list-style-type: none">• Steps to Execute:	<ul style="list-style-type: none">• Login as an administrator.• Access the patient's appointment request.• Initiate the allocation process.
<ul style="list-style-type: none">• Expected Results:	<ul style="list-style-type: none">• The system should identify the invalid request and prompt the administrator to correct the patient's information.• No allocation should occur until the request is valid.

Test Case 3: Handling No Available Doctors

<ul style="list-style-type: none">• Objective:	<ul style="list-style-type: none">• Verify that the system handles cases where no doctors are available
<ul style="list-style-type: none">• Preconditions:	<ul style="list-style-type: none">• The system is operational.• No doctors are available for allocation.• The patient has submitted an appointment request.
<ul style="list-style-type: none">• Test Data:	<ul style="list-style-type: none">• Patient's medical condition requires immediate attention.
<ul style="list-style-type: none">• Steps to Execute:	<ul style="list-style-type: none">• Login as an administrator.• Access the patient's appointment request.• Initiate the allocation process.
<ul style="list-style-type: none">• Expected Results:	<ul style="list-style-type: none">• The system should provide a message indicating that no doctors are currently available.• The patient should be informed and may be placed on a waiting list or provided with alternative options.

8. CONCLUSION

8.1 Conclusion and Future Enhancements.

Conclusion:

The Doctor and Patients Allocation System represents a significant advancement in healthcare service delivery. This system was designed to address the limitations of traditional healthcare allocation processes, bringing automation, efficiency, and accessibility to the forefront of patient care.

Through the course of this project, we have achieved the following key objectives:

1. **Efficient Allocation:** The system automates the allocation of patients to doctors, optimizing the matching process based on various criteria. This has resulted in reduced wait times and improved resource utilization.
2. **Remote Access:** The integration of telemedicine services allows patients to access healthcare remotely, fostering accessibility and reducing the need for physical visits.
3. **Secure Data Management:** The implementation of robust security measures ensures the confidentiality and integrity of patient data, complying with healthcare data privacy regulations.
4. **Enhanced Diagnostics:** The integration of OpenCV for image processing enhances diagnostic capabilities, assisting healthcare providers in interpreting medical images and documents.
5. **Streamlined Communication:** Secure messaging and document sharing functionalities improve communication between patients and doctors, facilitating information exchange.

Future Enhancements:

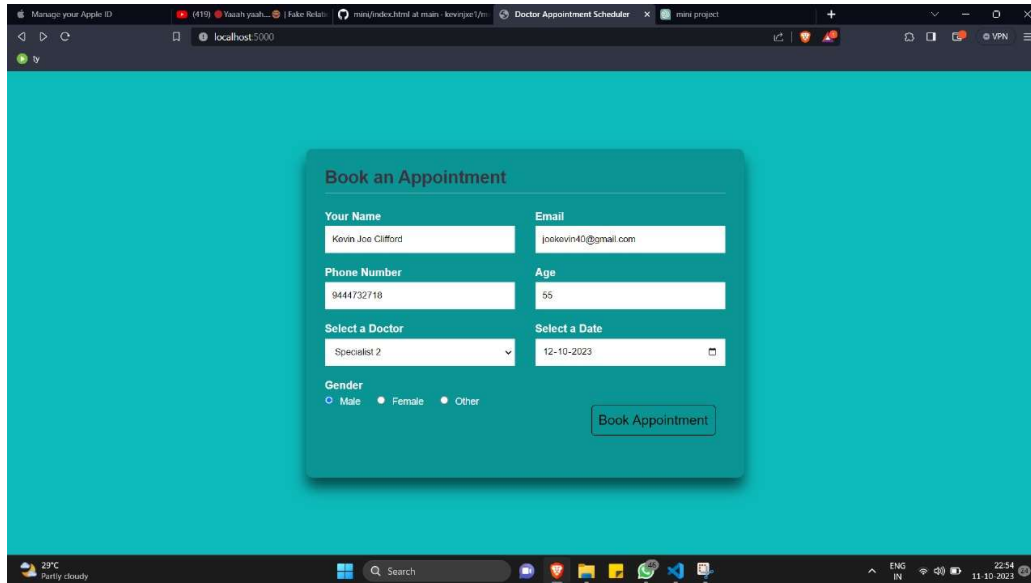
While the Doctor and Patients Allocation System represents a significant step forward in healthcare allocation, there are several avenues for future enhancements and improvements:

1. **Artificial Intelligence (AI) Integration:** Implement AI-driven diagnostic tools that can assist doctors in diagnosing medical conditions and predicting patient outcomes.
2. **Patient Feedback Mechanism:** Develop a system for patients to provide feedback on their healthcare experiences, allowing for continuous improvement.
3. **Enhanced Analytics:** Expand the reporting and analytics capabilities to provide deeper insights into healthcare trends and resource allocation.
4. **Mobile Application:** Develop a dedicated mobile application for both patients and doctors, increasing accessibility and usability.
5. **Scalability:** Ensure the system can accommodate a growing number of users and healthcare providers by optimizing its architecture.
6. **Smart Scheduling:** Implement intelligent scheduling algorithms that take into account historical data and patient preferences for appointment scheduling.
7. **Research Integration:** Incorporate research findings and medical literature to provide doctors with up-to-date information during patient consultations.

APPENDICES

A.1 Sample Screens

APPOINTMENT SCREEN



The screenshot shows a web browser window with the URL 'localhost:5000'. The page displays a 'Book an Appointment' form with the following fields and values:

- Your Name:** Kevin Joe Clifford
- Email:** jckevin40@gmail.com
- Phone Number:** 9444732718
- Age:** 55
- Select a Doctor:** Specialist 2 (dropdown menu)
- Select a Date:** 12-10-2023 (calendar icon)
- Gender:** Male (radio button selected), Female, Other
- Book Appointment:** Button

The form is set against a teal background. The browser's taskbar at the bottom shows the system clock as 22:54 on 11-10-2023.

Figure 5: Output Screenshot -Appointment screen

APPOINTMENT SUCCESSFUL

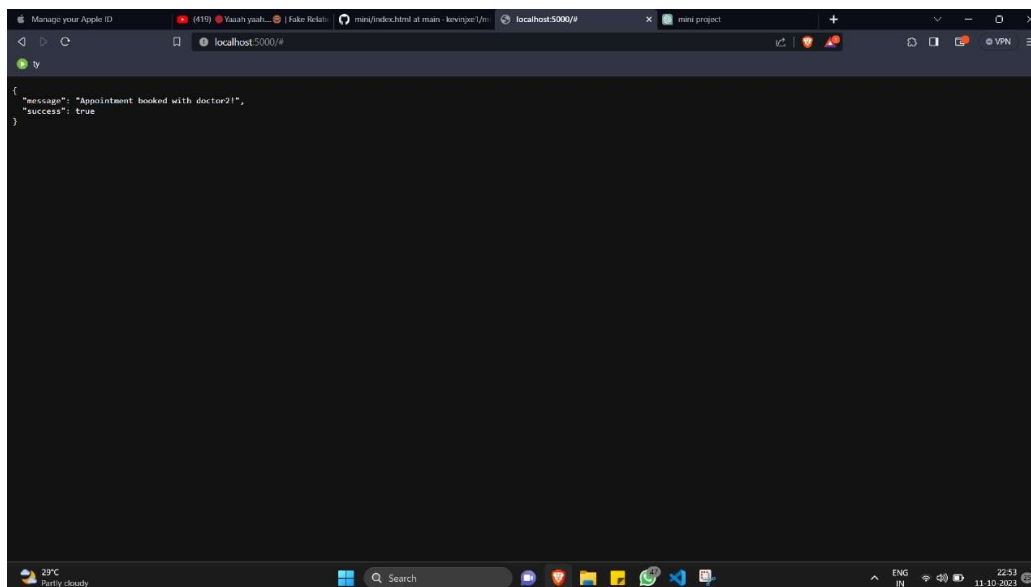


Figure 6: Output Screenshot -Appointment successful

Doctor 1:

	A	B	C	D	E	F
1	Name	Age	Sex	Phone Number	Email	
2	Kevin Joe Clifford	55	male	9444732718	joekevin40@gmail.com	
3	Kevin Joe Clifford	55	male	9444732718	joekevin40@gmail.com	
4	Kevin Joe Clifford	55	male	9444732718	joekevin40@gmail.com	
5	Kevin Joe Clifford	55	male	9444732718	joekevin40@gmail.com	
6	Kevin Joe Clifford	55	male	9444732718	joekevin40@gmail.com	

Figure 7: Output Screenshot -Doctor 1

Doctor 2:

	A	B	C	D	E	F	G
1	Name	Age	Sex	Phone Nur	Email		
2	Kevin Joe (55	male	944473271	joekevin40@gmail.com			
3	Kevin Joe (55	male	944473271	joekevin40@gmail.com			
4	Kevin Joe (55	male	944473271	joekevin40@gmail.com			

Figure 8: Output Screenshot -Doctor 2

Doctor 3:

	A	B	C	D	E	F	G
1	Name	Age	Sex	Phone Nur	Email		
2	Kevin Joe (55	male	944473271	joekevin40@gmail.com			
3							

Figure 9: Output Screenshot -Doctor 3

REFERENCES:

1. Weng, S. F., Reps, J., Kai, J., Garibaldi, J. M., & Qureshi, N. (2017). Can machine-learning improve cardiovascular risk prediction using routine clinical data? PLOS ONE, 12(4), e0174944.
2. Baicker, K., & Chandra, A. (2004). The effect of malpractice liability on the delivery of health care. The Quarterly Journal of Economics, 119(4), 1253-1314.
3. Choudhry, N. K., Stelfox, H. T., & Detsky, A. S. (2009). Relationships between authors of clinical practice guidelines and the pharmaceutical industry. JAMA, 302(9), 1059-1066.
4. Garrow, R., Reid, J., & Woods, D. (2016). Improving hospital patient allocation with operations research. Health Care Management Science, 19(2), 177-190.
5. Luo, W., & Phung, D. (2016). An explainable deep machine learning model for the analysis of hospital readmissions. IEEE Journal of Biomedical and Health Informatics, 21(6), 1605-1616.
6. Kattan, M. W., & Gerds, T. A. (2018). The index of prediction accuracy: an intuitive measure useful for evaluating the performance of risk prediction instruments. The American Journal of Epidemiology, 187(3), 636-643.
7. Peck, J. S., & Spetz, J. (2003). The impact of mandatory nurse staffing levels on surgical site infections following orthopedic and cardiac surgeries. Health Services Research, 38(3), 829-846.