PATIENTS BOOKING

MINI PROJECT

M.MARIA PRISCILLA-230701180

V.MAHALAKSHMI - 230701171

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE,THANDALAM,CHENNAI.

AN AUTONOMOUS INSTITUTE

CHENNAI

NOVEMBER 2024,

BONAFIDE CERTIFICATE

Certified that this project 'PATIENTS BOOKING''bonafide work of **"MARIA PRISCLLA.M , V.MAHALAKSHMI"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE SIGNATURE

Mrs. K.Maheshmeena
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College (Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Hospital Management System (HMS) website is designed to streamline the patient experience by providing an intuitive platform for booking treatment slots. This innovative system offers a user-friendly interface that allows patients to easily schedule appointments with their preferred healthcare providers. By integrating advanced features such as real-time slot availability,secure patient data management, the hospital management system enhances operational efficiency and patient satisfaction. Patients can view available time slots, select their desired appointment times, and receive instant confirmation. The system also supports healthcare professionals by optimizing their schedules and reducing administrative burdens. With robust security measures in place, patient confidentiality and data protection are ensured, making the hospital management system a comprehensive solution for modern healthcare needs.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGENO
	ABSTRACT	**iii**
	LIST OF FIGURES	**v**
	LIST OF ABBREVIATIONS	**vi**
1.	**INTRODUCTION**	**1**
	1.1 OBJECTIVES	1
	1.2 MODULES	1
2.	**SURVEY OF TECHNOLOGIES**	**3**
	2.1 SOFTWARE DESCRIPTION	3
	2.2 LANGUAGES	3
	2.2.1 SQL	3
	2.2.2 PHP	3
	2.2.3 HTML	4
	2.2.4 CSS	4
	2.2.5 JAVASCRIPT	4
3.	**REQUIREMENTS AND ANALYSIS**	**6**
	3.1 REQUIREMENT SPECIFICATION	6
	3.2 HARDWARE AND SOFTWARE REQUIREMENTS	8
	3.3 ARCHITECTURE DIAGRAM	9
	3.4 ER DIAGRAM	10
	3.5 NORMALIZATION	11
4.	**PROGRAM CODE**	**13**
5.	**RESULTS AND DISCUSSION**	**21**
6.	**CONCLUSION**	**28**
7.	**REFERENCES**	**29**

LIST OF FIGURES

S NO. NO.	TITLE	PAGE
3.1	SYSTEM ARCHITECTURE	9
3.2	ER DIAGRAM	10
5.1	CREATING DATABASE	22
5.2	IMPORTING DATABASE THROUGH XAMPP	22
5.3	LIST OF TABLES PRESENT IN DB	22
5.4	PATIENTS TABLE	23
5.5	TEST TABLE	23
5.6	TRIGR TABLE	23
5.7	USER TABLE	23
5.8	HOME PAGE	24
5.9	LOGIN PAGE	25
5.10	SIGN UP PAGE	25
5.11	BOOKING PAGE	26
5.12	BOOKING CONFIRMED	26
5.13	LIST OF APPOINTMENTS	27

LIST OF ABBREVIATIONS

DBMS	Database Management System
SQL	Structured Query Language
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
HTTP	HyperText Transfer Protocol
ER	Entity Relationship
NF	Normal Form

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES

The objective of this Hospital Management System project is to streamline and enhance the efficiency of hospital operations through the implementation of a comprehensive web-based platform. Utilizing HTML, CSS, and SQL, this system aims to facilitate seamless appointment booking, provide an accessible and organized list of doctors, and ensure secure patient data management. The platform includes user-friendly login and signup pages to ensure secure access for patients and medical staff. By integrating these functionalities, the system seeks to improve patient experience, optimize resource management, and uphold high standards of data security and privacy in medical records handling.

1.2 MODULES

A Hospital Management System (HMS) typically comprises several key modules to ensure comprehensive functionality and efficient management of Hospital records. Here are the main modules used in a Hospital Management System:

1.2.1 User Authentication Module

- **Login Page:** Secure login for patients, doctors, and administrators.
- **Signup Page:** Registration for new users with proper validation.

1.2.2. User Management Module

- Handles user registration and profile management.
- Enables users to update personal information and view their medical history and appointments.

1.2.3.Appointment Booking Module

- **Appointment Scheduling:** Patients can book appointments with available doctors.
- **Appointment History:** Patients can view their past and upcoming appointments.

1.2.5. Patient Data Management Module

- **Patient Profiles:** Secure storage and management of patient personal and medical information.
- **Medical Records:** Patients and doctors can view and update medical history, and recent appointments.

1.2.6. Dashboard Module

- **Patient Dashboard:** Overview of appointments, medical records, and personal information.
- **Doctor Dashboard:** Overview of scheduled appointments, patient profiles.

1.2.7. Search and Filter Module

- **Doctor Search:** Search functionality for patients to find doctors based on specialty of service.
- **Appointment Search:** Search functionality for patients and doctors to find specific appointments.

1.2.8. Security Module

- **Data Encryption:** Ensure all sensitive data is encrypted in transit and at rest.
- **Access Control:** Implement role-based access control to restrict access to sensitive information and functionalities.

CHAPTER 2

SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

2.1.1.Database: XAMPP (MySQL)

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting primarily of the Apache HTTP Server, MariaDB (formerly MySQL), and interpreters for scripts written in the PHP and Perl programming languages. It is widely used for local development of web applications and testing purposes.

Key Features:

- Apache Web Server: Provides the HTTP server to run web applications locally.
- MariaDB (MySQL): Database management system for storing and managing data.
- PHP and Perl Support:Enables server-side scripting and development.
- Cross-Platform Compatibility:Available for Windows, Linux, and macOS.
- Ease of Use: Simple installation and configuration, ideal for development and testing.

2.2 LANGUAGES

2.2.1 SQL(Structured Query Language)

SQL is a domain-specific language used in programming for managing data within a relational database management system (RDBMS). It is crucial for querying, inserting, updating, and deleting data, as well as managing database schemas and access control.

Key Features:

- **Data Manipulation:** Commands such as SELECT, INSERT, UPDATE, DELETE
- **Data Definition:**Commands like CREATE, ALTER, DROP
- **Data Control:** GRANT, REVOKE for managing access
- **Transaction Control:**COMMIT, ROLLBACK to manage transactions

2.2.2 PHP (Hypertext Preprocessor)

PHP is a popular open-source scripting language that is especially suited for web development and can be embedded into HTML. It is known for its ease of use, speed, and flexibility.

Key Features:

- Server-side Scripting: Processes code on the server before sending it to the client's browser.
- Database Integration: Seamlessly integrates with databases like MySQL, PostgreSQL, and others.
- Rich Ecosystem: Extensive libraries and frameworks such as Laravel and Symfony.
- Cross-Platform:Operates on various platforms, including Windows, Unix, and Linux.

2.2.3. HTML (HyperText Markup Language)

HTML is the standard markup language for creating web pages. It =structures web content and is the foundational element of any web application.

Key Features:

- Elements and Tags:Basic building blocks including headings, paragraphs, links, and images.
- Attributes:Provide additional information about elements, such as class and id.
- Forms: Used for collecting user input.
- Multimedia Support: Integrates images, audio, and video.

2.2.4 CSS (Cascading Style Sheets)

CSS is a style sheet language used to describe the presentation of a document written in HTML or XML. It controls the layout and appearance of web pages.

Key Features:

- Selectors and Properties: Define styles for HTML elements.
- Box Model:Fundamental layout principle including margins, borders, padding, and content.
- Flexbox and Grid: Modern layout systems for creating responsive designs.
- Animations and Transitions: Add interactivity and visual effects.

2.2.5.JavaScript

JavaScript is a high-level, versatile programming language that enables interactive web pages. It is an essential part of web applications, allowing client-side script to interact with users and create dynamic content.

Key Features:

- Event Handling:Responds to user actions like clicks and form submissions.
- DOM Manipulation:Interacts directly with HTML and CSS.
- Asynchronous Programming:Manages asynchronous operations using callbacks, promises, and async/await.
- Frameworks and Libraries:Rich ecosystem including React, Angular, Vue.js, and jQuery.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 REQUIREMENTS SPECIFICATION

Functional Requirements:

3.1.1 User Management

- **Patient Registration:** Allow patients to create and manage their accounts, providing necessary details such as name, contact information, medical history, and insurance details.
- **Healthcare Provider Registration**: Enable doctors and healthcare staff to register and manage their profiles, including specializations, available time slots, and contact details.

3.1.2 Appointment Scheduling

- **Slot Availability:** Display real-time availability of appointment slots for various healthcare providers.
- **Booking Appointments:** Allow patients to book, reschedule, or cancel appointments.

3.1.3 Treatment and Medical Records

Medical History Management: Enable patients and doctors to view and update medical histories and treatment records securely.

3.1.4 Billing and Payments

- **Billing Information:** Generate and display bills for treatments and appointments.
- **Online Payment Integration:** Allow patients to pay bills online via secure payment gateways.

3.1.5 Reporting and Analytics

- **Appointment Reports**: Generate reports on scheduled, completed, and missed appointments.
- **Patient Reports**: Provide insights into patient demographics, visit frequencies, and common treatments.
- **Financial Reports:** Track revenue from appointments and treatments.

2. Non-Functional Requirements

3.2.1 Usability

- **User Interface:** Design an intuitive and easy-to-navigate interface for both patients and healthcare providers.
- **Accessibility:** Ensure the system is accessible to users with disabilities, adhering to relevant accessibility standards.

3.2.2 Performance

- **Scalability:** Ensure the system can handle a growing number of users and increased data load without performance degradation.
- **Response Time:** Optimize the system to provide fast response times for user actions.

3.2.3 Security

- **Data Encryption:** Encrypt sensitive patient and financial data both in transit and at rest.
- **Authentication and Authorization:** Implement robust authentication mechanisms and role-based access control to ensure only authorized users can access specific functionalities.

HARDWARE REQUIREMENTS

- Processor: Intel i7
- RAM:16 GB
- Storage:512 GB SSD
- Network:Stable internet connection

SOFTWARE REQUIREMENTS

Server Software:

- Operating System: Windows Server
- Web Server:Apache HTTP Server 2.4
- Database Server:MySQL 5.7
- Server-Side Scripting:PYTHON 3.12.7

Development Tools:

- XAMPP:Includes Apache, MySQL, PHP, and PHPMyAdmin
- Version Control:Git

Client Software:

- Operating System:Windows 11
- Web Browser:Latest version of Chrome, Firefox, Safari, or Edge

Development Tools:

- Code Editor: Visual Studio Code, Sublime Text, or Atom
- Browser Developer Tools:For debugging
- Version Control: Git

3.3 Architecture Diagram

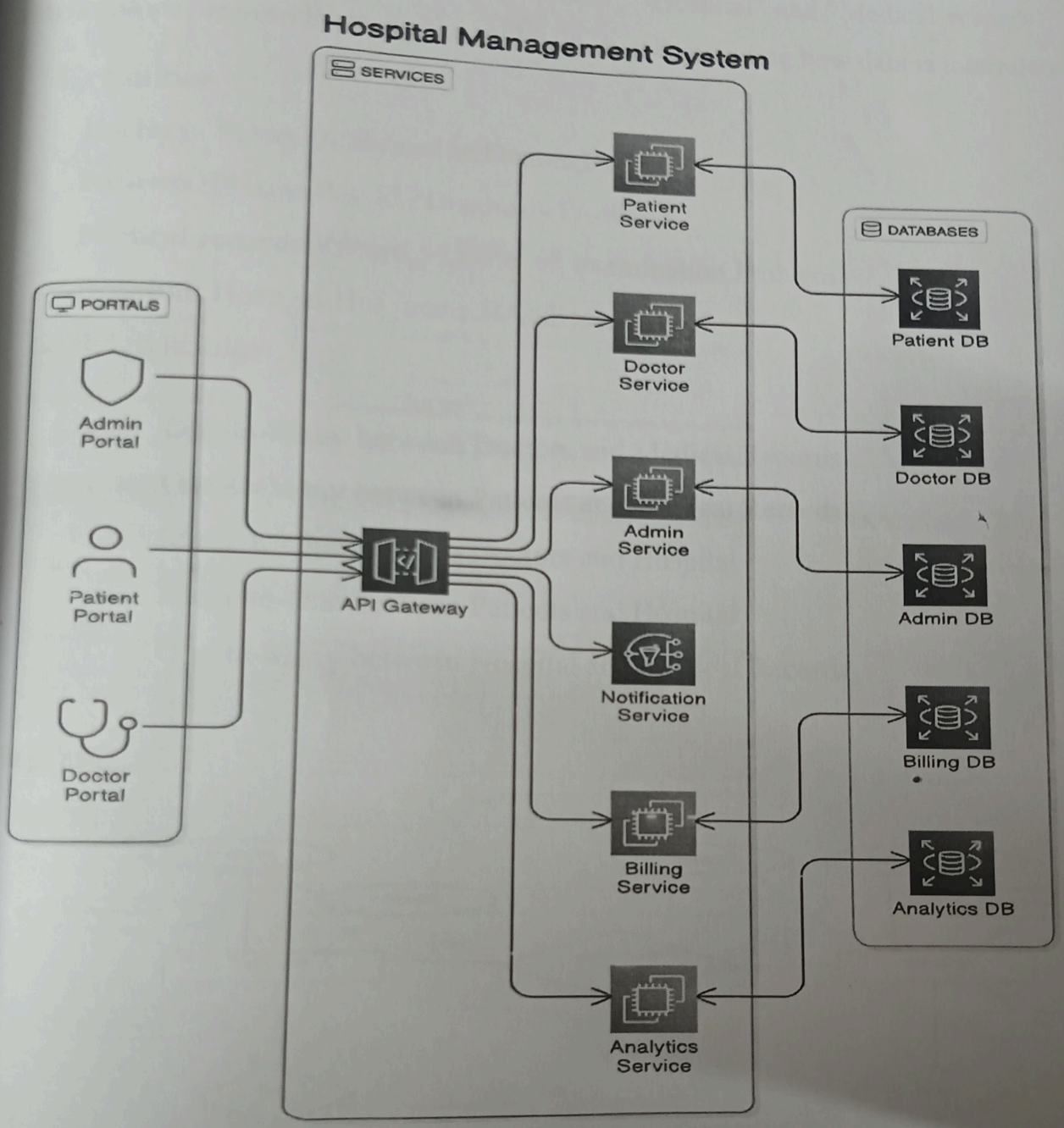


FIGURE 3.1: SYSTEM ARCHITECTURE

The architecture of the Hospital Management System (HMS) is designed to ensure seamless interaction between patients, healthcare providers, and administrative staff through a robust and scalable framework.

3.4 ER Diagram

An Entity-Relationship (ER) diagram for the Hospital Management System typically includes entities such as `Patients`, `Doctors`, 'Hospital' and `Medical records`. The relationships between these entities will help in understanding how data is interrelated.

1. **Entities:**

   Doctors: Salary,Qualification,Dname,Doc_id

   Patients :PName,Pat_id,PDiagnosis,PAddress

   Medical records: Record_id,Date_of_examination,Problem

   Hospital: Hosp_id,Hos_name,HAddress,Hcity

2. **Relationships:**

   - One-to-Many between Doctors and Medical Records
   - One-to-Many between Patients and Medical Records
   - Many-to-One between Doctors and Hospital
   - Many-to-One between Patients and Hospital
   - One-to-Many between Hospital and Medical Records

Diagram:

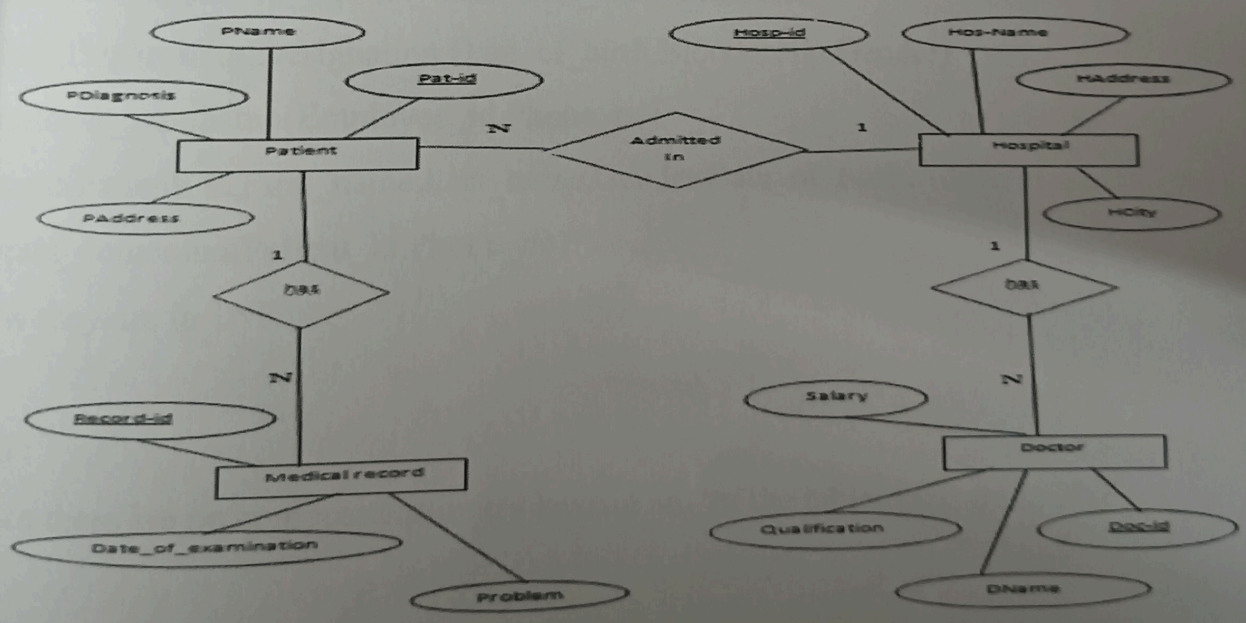


FIGURE 3.2: ER DIAGRAM

3.5 Normalization

i. Normalized Table Set:

Doctor(Doctor_id,Speciality,Experience,Consulting_time,Username,Password)
Medical_records(Patient_id,M_id,Treatment_plan,Joined _date,Diagnosis)
Appointment (Appointment_id, Doctor_id, Patient_id, Date, Status,Consultation_time)
Bill(Bill_id,Patient_id,Treatment_id,Room_id,Bill_amt,Bill_date)
Room(Room_id,Patient_id,Nurse_id,Room_type,Date_admitted,Datedischarged,RPrice)

ii. Unnormalized Table Set:

Employee(Employee_id,First_name,Last_name,Address,
Salary,Department,Designation,Date_of_birth,Bloodgroup,Gender,Phone_no)
Patient(Patient_id,First_name,Last_name,Gender,Date_o f_birth,Address,Phone_no)

1NF:

The attributes Phone_no may have multiple values for a single ID.
Employee(Employee_id,First_name,Last_name,Address,
Salary,Department,Designation,Date_of_birth,Bloodgroup,Gender)
Employee_contactno(Employee_id,Phone_no)
Patient(Patient_id,First_name,Last_name,Gender,Date_of_birth,Address)
Patient_contactno(Patient_id,Phone_no)
Now they are in 1NF

2NF:

Since there are no composite primary keys in any of the table, they are in 2NF.

3NF:

Since there is no transitivity among any attributes, the tables are in 3NF

Final normalized tables:

Employee(Employee_id,First_name,Last_name,Address,Salary,Department,Designation, Date_of_birth,Bloodgroup,Gender)

Employee_contactno(Employee_id,Phone_no)

Doctor(Doctor_id,Speciality,Experience,Consulting_time,Username,Password)

Patient(Patient_id,First_name,Last_name,Gender,Date_of_birth,Address)

Patient_contactno(Patient_id,Phone_no)

Medical_records(Patient_id,M_id,Treatment_plan,Joined_dat e,Diagnosis)

Appointment (Appointment_id, Doctor_id, Patient_id, Date, Status,Consultation_time)

Bill(Bill_id,Patient_id,Treatment_id,Room_id,Bill_amt,Bill_d ate)

CHAPTER 4

PROGRAM CODE

4.1 Program code

```
from flask import Flask,render_template,request,session,redirect,url_for,flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import login_user,logout_user,login_manager,LoginManager
from flask_login import login_required,current_user
# from flask_mail import Mail
import json


# MY db connection
local_server= True
app = Flask(_name_)
app.secret_key='hmsprojects'


# this is for getting unique user access
login_manager=LoginManager(app)
login_manager.login_view='login'

# SMTP MAIL SERVER SETTINGS

# app.config.update(
#    MAIL_SERVER='smtp.gmail.com',
#    MAIL_PORT='465',
#    MAIL_USE_SSL=True,
#    MAIL_USERNAME="add your gmail-id",
#    MAIL_PASSWORD="add your gmail-password"
# )
# mail = Mail(app)


@login_manager.user_loader
def load_user(user_id):
```

```
    return User.query.get(int(user_id))



#app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/databas_table_name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/hmdbms'
db=SQLAlchemy(app)



# here we will create db models that is tables
class Test(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(100))
    email=db.Column(db.String(100))

class User(UserMixin,db.Model):
    id=db.Column(db.Integer,primary_key=True)
    username=db.Column(db.String(50))
    usertype=db.Column(db.String(50))
    email=db.Column(db.String(50),unique=True)
    password=db.Column(db.String(1000))

class Patients(db.Model):
    pid=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
    gender=db.Column(db.String(50))
    slot=db.Column(db.String(50))
    disease=db.Column(db.String(50))
    time=db.Column(db.String(50),nullable=False)
    date=db.Column(db.String(50),nullable=False)
    dept=db.Column(db.String(50))
    number=db.Column(db.String(50))

class Doctors(db.Model):
    did=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
```

```
    doctorname=db.Column(db.String(50))
    dept=db.Column(db.String(50))

class Trigr(db.Model):
    tid=db.Column(db.Integer,primary_key=True)
    pid=db.Column(db.Integer)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
    action=db.Column(db.String(50))
    timestamp=db.Column(db.String(50))


# here we will pass endpoints and run the fuction
@app.route('/')
def index():
    return render_template('index.html')


@app.route('/doctors',methods=['POST','GET'])
def doctors():

    if request.method=="POST":

        email=request.form.get('email')
        doctorname=request.form.get('doctorname')
        dept=request.form.get('dept')

        query=Doctors(email=email,doctorname=doctorname,dept=dept)
        db.session.add(query)
        db.session.commit()
        flash("Information is Stored","primary")

    return render_template('doctor.html')
```

```
@app.route('/patients',methods=['POST','GET'])
@login_required
def patient():
 # doct=db.engine.execute("SELECT * FROM doctors")
 doct=Doctors.query.all()

 if request.method=="POST":
  email=request.form.get('email')
  name=request.form.get('name')
  gender=request.form.get('gender')
  slot=request.form.get('slot')
  disease=request.form.get('disease')
  time=request.form.get('time')
  date=request.form.get('date')
  dept=request.form.get('dept')
  number=request.form.get('number')
  # subject="HOSPITAL MANAGEMENT SYSTEM"
  if len(number)<10 or len(number)>10:
   flash("Please give 10 digit number")
   return render_template('patient.html',doct=doc
    db.session.add(query)
  db.session.commit()
  flash("Booking Confirmed","info")


 return render_template('patient.html',doct=doct)


@app.route('/bookings')
@login_required
def bookings():
 em=current_user.email
 if current_user.usertype=="Doctor":
  # query=db.engine.execute(f"SELECT * FROM patients")
  query=Patients.query.all()
  return render_template('booking.html',query=query)
 else:
  # query=db.engine.execute(f"SELECT * FROM patients WHERE email='{em}'")
```

```
    query=Patients.query.filter_by(email=em)
    print(query)
    return render_template('booking.html',query=query)



@app.route("/edit/<string:pid>",methods=['POST','GET'])
@login_required
def edit(pid):
  if request.method=="POST":
    email=request.form.get('email')
    name=request.form.get('name')
    gender=request.form.get('gender')
    slot=request.form.get('slot')
    disease=request.form.get('disease')
    time=request.form.get('time')
    date=request.form.get('date')
    dept=request.form.get('dept')
    number=request.form.get('number')
    post=Patients.query.filter_by(pid=pid).first()
    post.email=email
    post.name=name
    post.gender=gender
    post.slot=slot
    post.disease=disease
    post.time=time
    post.date=date
    post.dept=dept
    post.number=number
    db.session.commit()

    flash("Slot is Updates","success")
    return redirect('/bookings')

  posts=Patients.query.filter_by(pid=pid).first()
  return render_template('edit.html',posts=posts)



@app.route("/delete/<string:pid>",methods=['POST','GET'])
```

```
@login_required
def delete(pid):
 # db.engine.execute(f"DELETE FROM patients WHERE patients.pid={pid}")
 query=Patients.query.filter_by(pid=pid).first()
 db.session.delete(query)
 db.session.commit()
 flash("Slot Deleted Successful","danger")
 return redirect('/bookings')
```

```
@app.route('/signup',methods=['POST','GET'])
def signup():
 if request.method == "POST":
  username=request.form.get('username')
  usertype=request.form.get('usertype')
  email=request.form.get('email')
  password=request.form.get('password')
  user=User.query.filter_by(email=email).first()
  # encpassword=generate_password_hash(password)
  if user:
   flash("Email Already Exist","warning")
   return render_template('/signup.html')

   myquery=User(username=username,usertype=usertype,email=email,password=password)
  db.session.add(myquery)
  db.session.commit()
  flash("Signup Succes Please Login","success")
  return render_template('login.html')

 return render_template('signup.html')

@app.route('/login',methods=['POST','GET'])
def login():
```

```
    if request.method == "POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()

        if user and user.password == password:
            login_user(user)
            flash("Login Success","primary")
            return redirect(url_for('index'))
        else:
            flash("invalid credentials","danger")
            return render_template('login.html')





    return render_template('login.html')

@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash("Logout SuccessFul","warning")
    return redirect(url_for('login'))



@app.route('/test')
def test():
    try:
        Test.query.all()
        return 'My database is Connected'
    except:
        return 'My db is not Connected'


@app.route('/details')
@login_required
```

```
def details():
    posts=Trigr.query.all()
    # posts=db.engine.execute("SELECT * FROM trigr")
    return render_template('trigers.html',posts=posts)

@app.route('/search',methods=['POST','GET'])
@login_required
def search():
    if request.method=="POST":
        query=request.form.get('search')
        dept=Doctors.query.filter_by(dept=query).first()
        name=Doctors.query.filter_by(doctorname=query).first()
        if name:

            flash("Doctor is Available","info")
        else:

            flash("Doctor is Not Available","danger")
    return render_template('index.html')


app.run(debug=True)
```

CHAPTER 5
RESULTS AND DISCUSSION

5.1 RESULTS

The successful implementation of the Hospital Management System (HMS) resulted in a significant enhancement in both patient satisfaction and operational efficiency within the healthcare facility. Patients now benefit from an intuitive, user-friendly platform that simplifies appointment scheduling, providing real-time availability, instant booking confirmations, and automated reminders. The streamlined process has reduced wait times and improved patient access to healthcare services. Healthcare providers have experienced a reduction in administrative burdens, allowing them to focus more on patient care. The secure management of patient data and medical records has ensured confidentiality and compliance with healthcare regulations. Additionally, the system's comprehensive reporting and analytics capabilities have provided valuable insights into patient demographics, treatment patterns, and financial performance, aiding in informed decision-making and resource allocation. Overall, the HMS has proven to be a robust solution that effectively addresses the needs of modern healthcare management.

5.2 DISCUSSION

The development and deployment of the Hospital Management System (HMS) underscored the transformative impact of digital solutions in healthcare management. Throughout the project, several key challenges were addressed, including the integration of real-time scheduling, the secure handling of sensitive patient data, and ensuring system accessibility across various devices and platforms. The project highlighted the importance of user-centered design, as patient and healthcare provider feedback was crucial in refining the system's functionality and interface. Moreover, the adherence to regulatory standards, such as HIPAA, ensured that the system not only improved operational efficiency but also maintained the highest levels of data security and patient privacy. The positive outcomes of the HMS project, such as increased patient satisfaction and reduced administrative workload, demonstrate the potential for digital systems to enhance healthcare delivery.

pid	email	name	gender	slot	disease	time	date	dept
2	anees1@gmail.com	anees1	Male1	evening1	cold1	21:20:00	2020-02-02	ortho11predic
5	patient@gmail.com	rehman khan	Male	morning	fevr	18:06:00	2020-11-18	Cardiologists
7	patient@gmail.com	patien	Male	evening	cold	22:18:00	2020-11-05	Dermatologist
8	patient@gmail.com	anees	Male	evening	cold	22:18:00	2020-11-05	Dermatologist
9	aneesurrehman423@gmail.com	anees	Male	morning	cold	17:27:00	2020-11-26	Anesthesiolog
10	anees@gmail.com	anees	Male	evening	fever	16:25:00	2020-12-09	Cardiologists
15	khushi@gmail.com	anees	Male	morning	corona	20:42:00	2021-01-23	Anesthesiolog
16	khushi@gmail.com	khushi	Female	evening	fever	15:46:00	2021-01-31	Endocrinologi
17	aneeqah@gmail.com	khushi	Female	evening	fever	15:48:00	2021-01-23	Endocrinologi
		aneeqah	Female					

FIGURE 5.4 PATIENTS TABLE

id	name	email
1	ANEES	ARK@GMAIL.COM
2	test	test@gmail.com

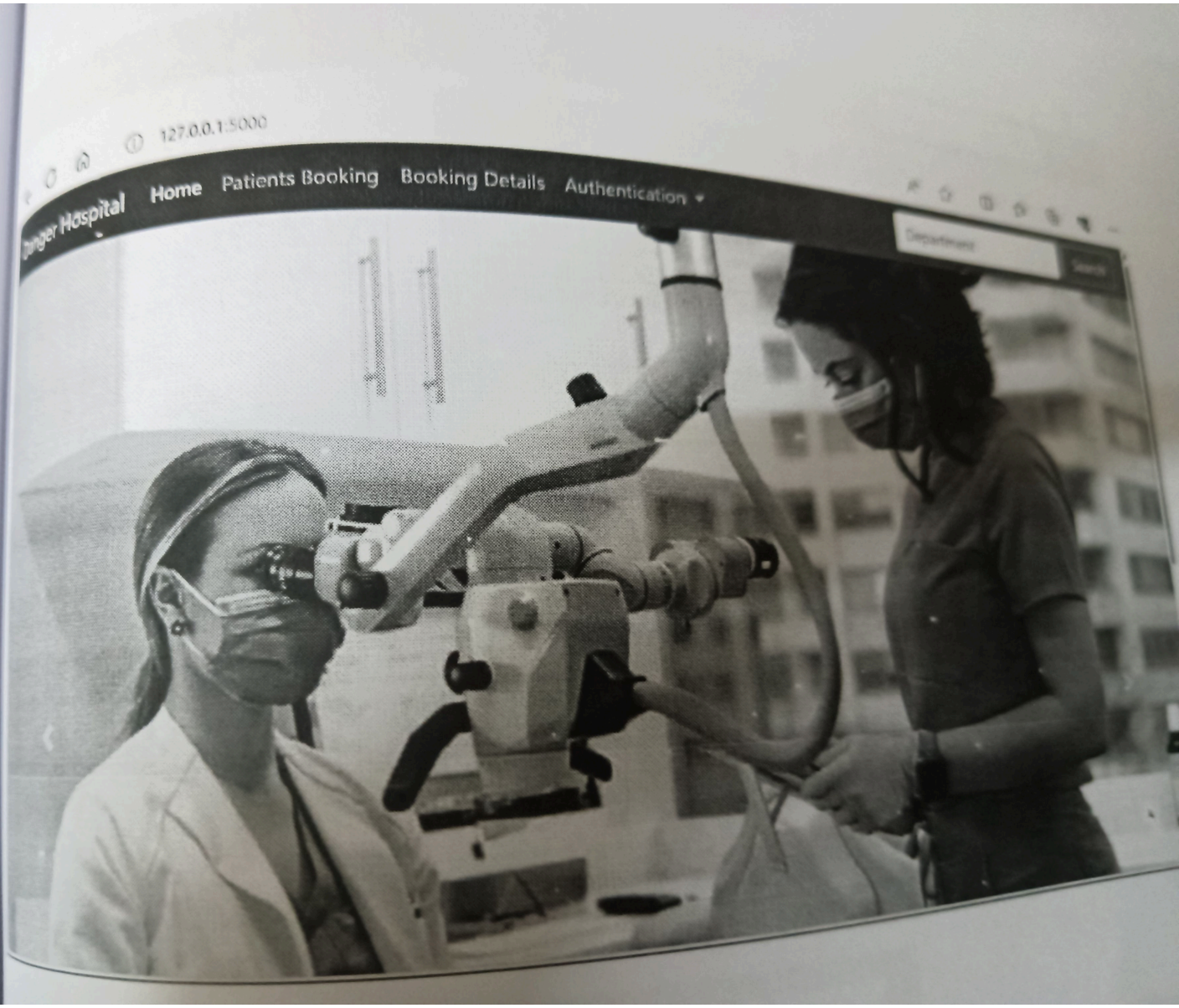
FIGURE 5.5 TEST TABLE

tid	pid	email	name	action	timestamp
1	12	anees@gmail.com	ANEES	PATIENT INSERTED	2020-12-02 16:35:10
2	11	anees@gmail.com	anees	PATIENT INSERTED	2020-12-02 16:37:34
3	10	anees@gmail.com	anees	PATIENT UPDATED	2020-12-02 16:38:27
4	11	anees@gmail.com	anees	PATIENT UPDATED	2020-12-02 16:38:33
5	12	anees@gmail.com	ANEES	Patient Deleted	2020-12-02 16:40:40
6	11	anees@gmail.com	anees	PATIENT DELETED	2020-12-02 16:41:10
7	13	testing@gmail.com	testing	PATIENT INSERTED	2020-12-02 16:50:21
8	13	testing@gmail.com	testing	PATIENT UPDATED	2020-12-02 16:50:32
9	13	testing@gmail.com	testing	PATIENT DELETED	2020-12-02 16:50:57
10	14	aneeqah@gmail.com	aneeqah	PATIENT INSERTED	2021-01-22 15:18:09
11	14	aneeqah@gmail.com	aneeqah	PATIENT UPDATED	2021-01-22 15:18:29
12	14	aneeqah@gmail.com	aneeqah	PATIENT DELETED	2021-01-22 15:41:48
13	15	khushi@gmail.com	khushi	PATIENT INSERTED	2021-01-22 15:43:02
14	15	khushi@gmail.com	khushi	PATIENT UPDATED	2021-01-22 15:43:11
15	16	khushi@gmail.com	khushi	PATIENT INSERTED	2021-01-22 15:43:37
16	16	khushi@gmail.com	khushi	PATIENT UPDATED	2021-01-22 15:43:49
17	17	aneeqah@gmail.com	aneeqah	PATIENT INSERTED	2021-01-22 15:44:41
18	17	aneeqah@gmail.com	aneeqah	PATIENT UPDATED	2021-01-22 15:44:52

FIGURE 5.6 TRIGR TABLE

id	username	usertype	email	password
13	anees	Doctor	anees@gmail.com	pbkdf2:sha256:150000$xAKZCiJG$4c7a7e704708f86659
14	aneeqah	Patient	aneeqah@gmail.com	pbkdf2:sha256:150000$Yf51IIDC$028cff81a536ed9d4771
15	khushi	Patient	khushi@gmail.com	pbkdf2:sha256:150000$BeSHeRKV$a8b27379ce9b2499d

FIGURE 5.7 USER TABLE


127.0.0.1:5000
Home
Patients Booking
Booking Details
Authentication

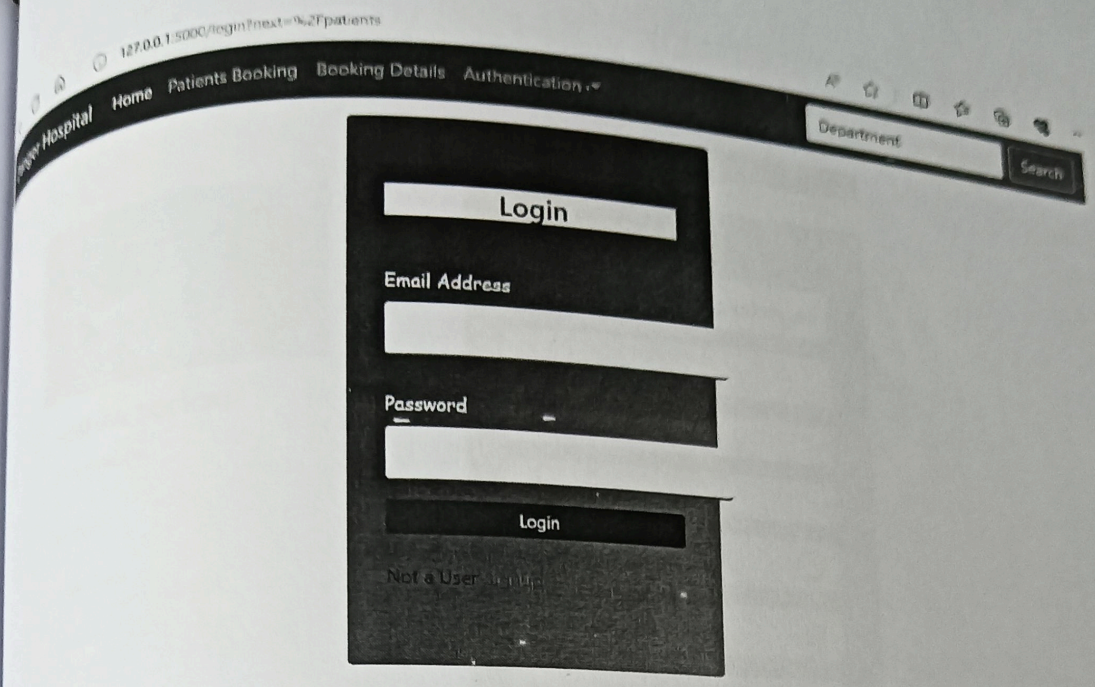


FIGURE 5.9 LOGIN PAGE

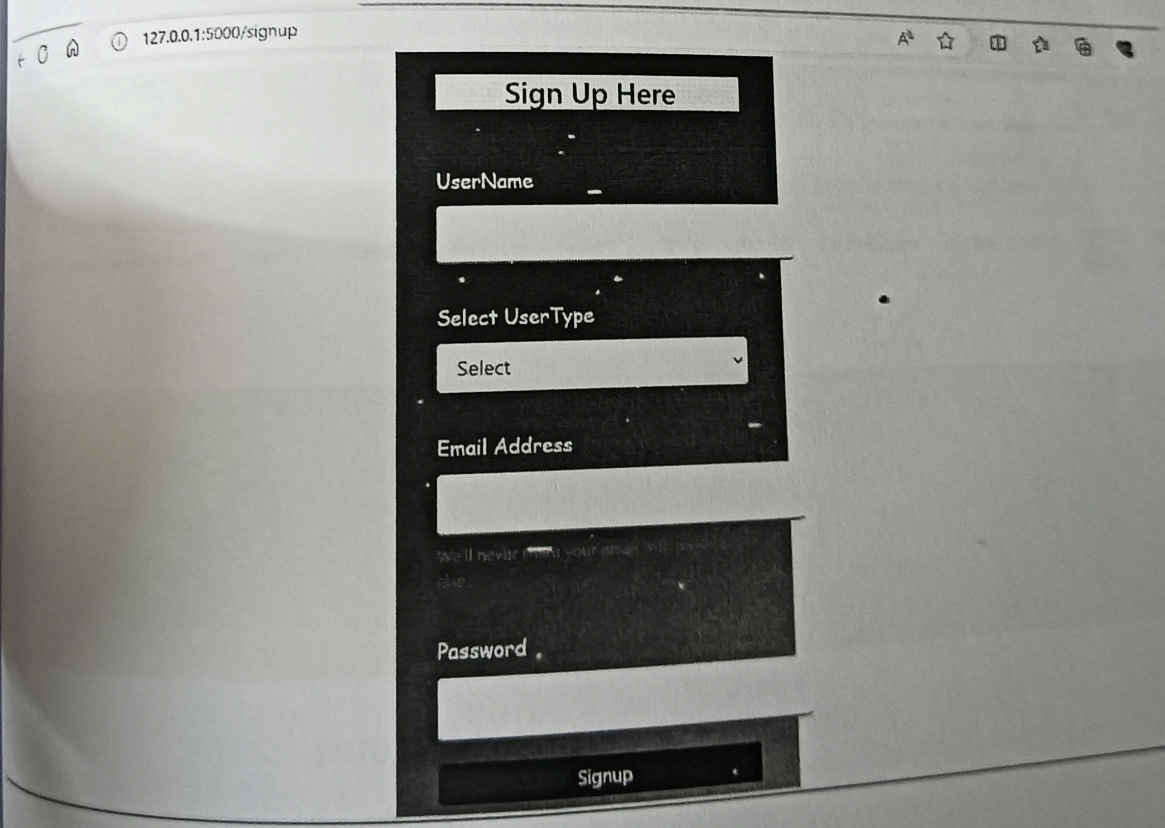


FIGURE 5.10 SIGN UP PAGE

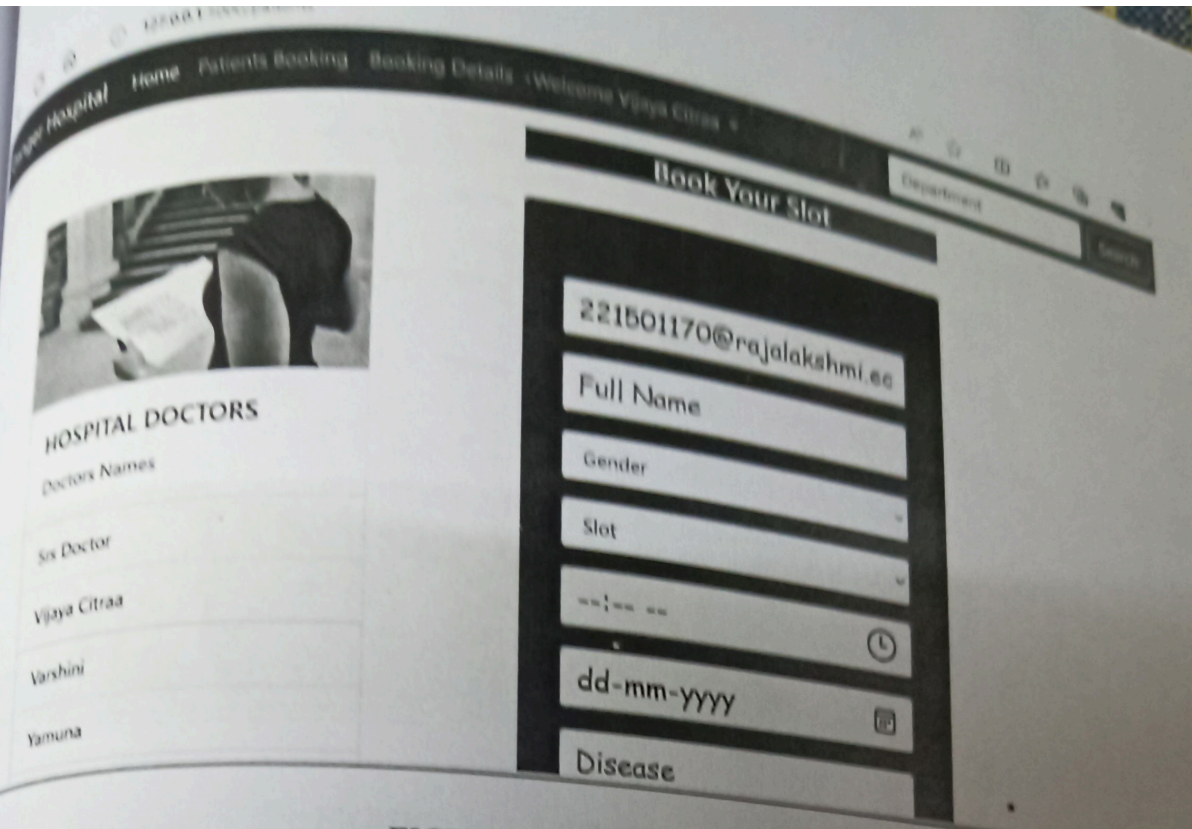


FIGURE 5.11 BOOKING PAGE

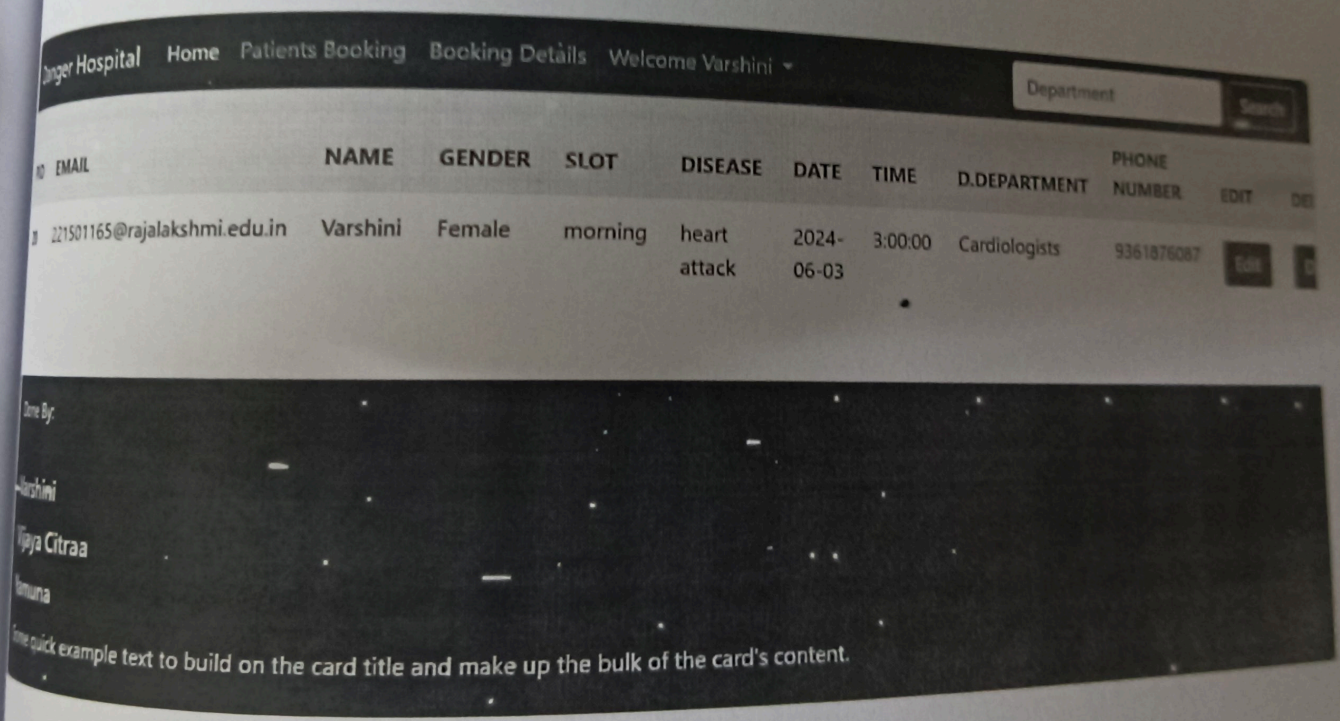


FIGURE 5.12 BOOKING CONFIRMED

127.0.0.1:5000/bookings

Danger Hospital Home Patients Booking Booking Details Welcome Varshini

Department

PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER
20	221501165@rajalakshmi.edu.in	Varshini	Female	morning	heart attack	2024-06-03	3:00:00	Cardiologists	936187608
22	221501165@rajalakshmi.edu.in	Nidarshana	Female	morning	corona symptoms	2024-06-04	2:00:00	corona	74182196
23	221501165@rajalakshmi.edu.in	sowmiya	Female	night	cancer	2024-06-05	23:59:00	Anesthesiologists	7418219

Done By:

REFERENCES

[1] Phil Hanna. (2003). JSP 2.0: The complete reference.Tata McGraw Hill Edition.

[2] J. Clerk Maxwell. (1892). A treatise on electricity and magnetism. (3rd ed.). Oxford: Clarendon, pp.68–73.

[3] Ali Bahrami. (1988). Object-oriented system development. (3rd ed.). Tata McGraw Hill Edition.

[4] Ivan Bayross. (2009). SQL, PL/SQL programming language of Oracle. (2nd ed.). BPB Publication.

[5] Tarhan, A., Turetken, O., & van den Biggelaar, F. J. (2015). Assessing healthcare process maturity: challenges of using a business process maturity model.

[6] Areda, C.A., Galato, D. & Federal, D. (2015). Mapping of processes in a hospital pharmacy: tool for quality management and improvement, Brazilian Journal of Hospital Pharmacy and Health Services, 6(3)

[7] Schriek, M., Türetken, O. & Kaymak, U. (2016). A maturity model for care pathways. Twenty-Fourth European Conference on Information Systems, Research Paper 127 (PDF)

[8] David Lake, Rodolfo Milito, Monique Morrow & Rajesh Vargheese. (2014). Internet of things: Architectural framework for ehealth security. Journal of ICT, River Publications, pp. 101-328.