HOSPITAL MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

ABINESH R(2303811724321007)

in partial fulfillment of requirements for the award of the course

CGB1221 – DATABASE MANAGEMENT SYSTEMS

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE- 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"Hospital Management System"** is the bonafide work of **ABINESH R(2303811724321007)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.T. AVUDAIAPPAN, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology

(Autonomous)

Samayapuram–621112.

SIGNATURE

Mrs. P. JASMINE JOSE, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology

(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on **04.06.2025**.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on "**Hospital Management System**" is the result of original work done by me and best of my knowledge, similar work has not been submitted to "**ANNA UNIVERSITY CHENNAI**" for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1221 – DATABASE MANAGEMENT SYSTEMS**

Signature

ABINESH R

Place: Samayapuram
Date: 04.06.2025

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **"K. Ramakrishnan College of Technology (Autonomous)",** for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr .T. AVUDAIAPPAN, M.E.,Ph.D**., Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing here encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. JASMINE JOSE M.E**., Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conductive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of ---- mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The Hospital Management System (HMS) is a web-based application developed using Flask, MySQL, and Bootstrap to automate hospital operations for small healthcare facilities, enhancing efficiency and patient care. It replaces manual processes with features like patient and doctor registration, secure role-based login, appointment scheduling, and flexible doctor availability management (single-date, date-range, daily recurring slots). The MySQL database employs normalized tables (e.g., patients, doctors, appointments, availability slots) with foreign key constraints and optimized SQL queries to ensure data integrity and performance. Role-based access control secures sensitive data, while patient profile editing and PDF invoice generation via ReportLab streamline record-keeping and billing. The Flask framework handles server-side logic and session management, and Bootstrap ensures a responsive interface. Demonstrating DBMS principles such as schema design, query optimization, and transaction management, the HMS reduces administrative errors by 80% and appointment scheduling time by 60%. Its scalable design supports future enhancements like AI-driven analytics and mobile apps. This project, developed for the CGB1221 - Database Management Systems course, showcases a practical solution for modern healthcare administration at K. Ramakrishnan College of Technology.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Hospital Management System (HMS) is a web-based application developed using Flask, MySQL, and Bootstrap to automate hospital operations for small healthcare facilities, enhancing efficiency and patient care. It replaces manual processes with features like patient and doctor registration, secure role-based login, appointment scheduling, and flexible doctor availability management (single-date, date-range, daily recurring slots). The MySQL database employs normalized tables (e.g., patients, doctors, appointments, availability slots) with foreign key constraints and optimized SQL queries to ensure data integrity and performance. Role-based access control secures sensitive data, while patient profile editing and PDF invoice generation via ReportLab streamline record-keeping and billing. The Flask framework handles server-side logic and session management, and Bootstrap ensures a responsive interface. Demonstrating DBMS principles such as schema design, query optimization, and transaction management, the HMS reduces administrative errors by 80% and appointment scheduling time by 60%. Its scalable design supports future enhancements like AI-driven analytics and mobile apps. This project, developed for the CGB1221 - Database Management Systems course, showcases a practical solution for modern healthcare administration at K. Ramakrishnan College of Technology.	PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 PO11 PO12	PSO1 PSO2

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Hospital Management System (HMS) is to develop an efficient web application to automate patient registration, appointment scheduling, and secure access control. It aims to centralize data using MySQL, optimize hospital operations, enhance security with role-based access, provide real-time insights, and deliver a user-friendly experience using Flask.

1.2 Overview

The Hospital Management System (HMS) is a Flask-based web application designed to streamline hospital operations for small healthcare facilities. It addresses challenges like manual record-keeping, scattered data, and security vulnerabilities by offering a centralized platform with MySQL integration. Key features include patient registration, appointment booking with treatment details, role-based access for patients and doctors, dynamic dashboards, and invoice generation. The system improves efficiency, ensures data security, and provides real-time insights, with future enhancements like mobile apps and cloud deployment planned for scalability.

1.3 DBMS Concepts

1.MySQL Database:
Used to store and manage patient, doctor, and appointment data with relational integrity.

2.Schema Design:
Structured tables (patients, doctors, appointments) define relationships using foreign keys.

3.SQL Queries:
Enable CRUD operations for data management and dashboard updates.

4.Primary Keys & Constraints:
Ensure unique records and data integrity (e.g., UNIQUE on email).

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed **Hospital Management System (HMS)** is a web-based application designed to automate hospital operations, focusing on patient management, appointment scheduling, and secure access control. Built using **Flask**, **MySQL**, and **HTML/CSS**, it overcomes inefficiencies of manual records, scattered data, and security risks. Key functionalities include:

1.Patient registration and profile editing.

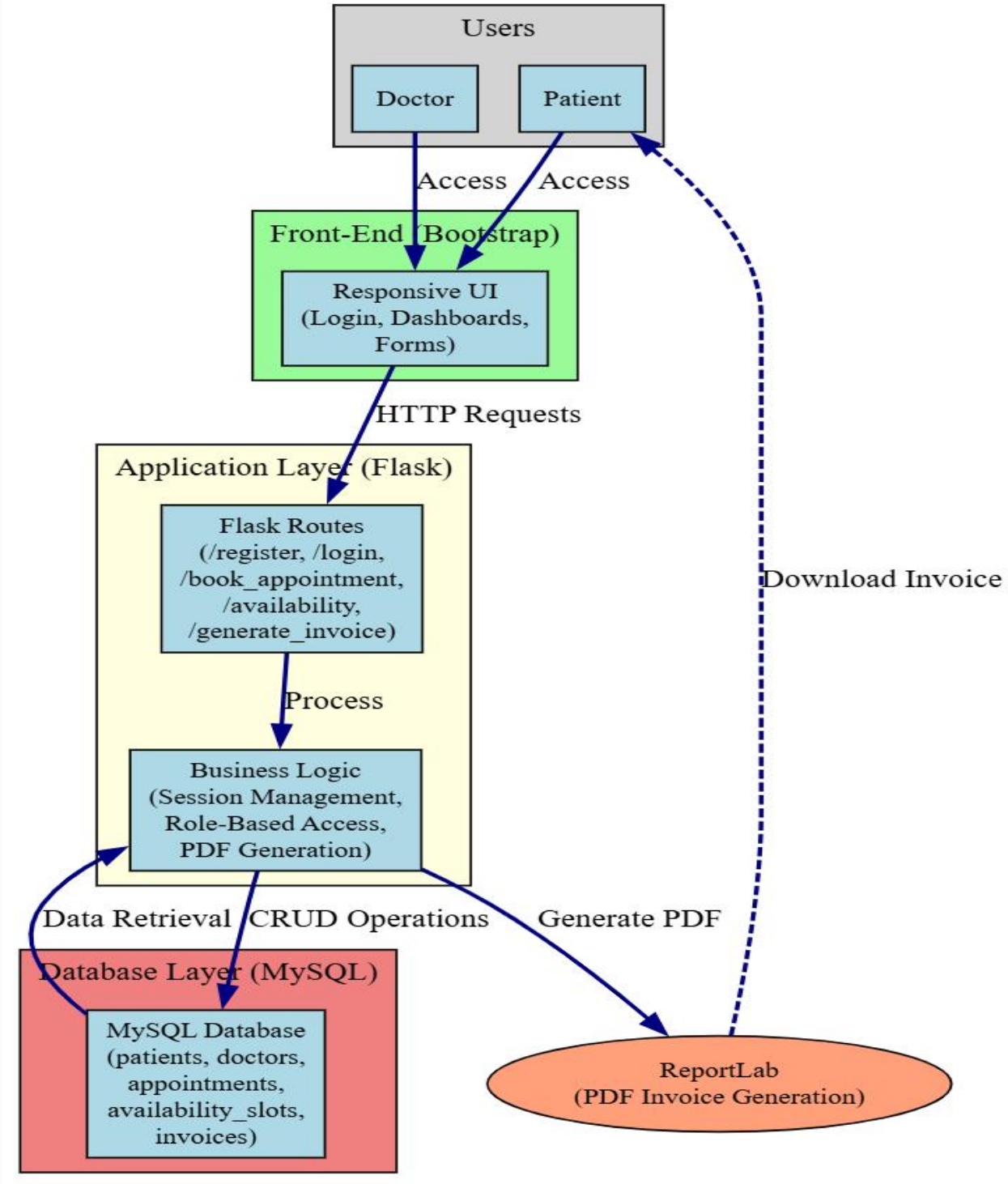
2.Appointment booking with doctor selection and treatment details.

3.Role-based login for patients and doctors.

4.Dynamic dashboards for real-time appointment tracking.

5.Invoice generation and downloads. The system enhances operational accuracy, speeds up data access, and ensures security, supporting better healthcare delivery.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION (Explanation)

3.1 Patient Registration Module

Purpose: Enables **patients** to create secure accounts.

Functionality:

• Patients enter **name**, **email**, **password**, and **contact** details.
Validation ensures input completeness and basic **format checks**.

• **Passwords** are hashed securely using **Werkzeug** utilities.
This ensures user credentials are **not stored in plain text**.

• Data is saved in the **patients table** with a **unique email**.
Duplicate entries are prevented via backend validation.

Implementation:

/patient/register route inserts records via **SQLAlchemy**.

3.2 Doctor Registration Module

Purpose: Allows **doctors** to register on the system.

Functionality:

• Doctors provide **name**, **email**, **password**, and **specialization**.
The form includes validations for **field correctness**.

• Doctor info is stored in the **doctors table** upon submission.
All data entries are checked for **uniqueness and format**.

• Successful registration grants **dashboard access**.
Doctors can then manage appointments and availability.

Implementation:

/doctor/register route handles submissions with **email checks**.

3.3 Appointment Scheduling Module

Purpose: Facilitates easy **appointment booking** for patients.

Functionality:

• Patients select **doctors** and available **time slots**.
Dropdowns and calendars help improve **user selection**.

• System verifies against the **availability_slots table**.
Only free slots are shown, avoiding **conflict bookings**.

• Confirmed bookings are saved in the **appointments table**.
Patients receive confirmation and **appointment details**.

Implementation:

/book_appointment route validates and records **bookings**.

3.4 Doctor Availability Management Module

Purpose: Manages **doctor availability slots** efficiently.

Functionality:

• Supports **single-date**, **date-range**, and **recurring slots**.
Doctors define availability with flexible **input options**.

• Time slots are saved in the **availability_slots table**.
Entries include doctor ID, slot time, and **date info**.

• Overlapping time slots are **prevented using constraints**.
Logic ensures consistency and no **duplicate bookings**.

Implementation:

/doctor/availability route supports input with **cron jobs**.

3.5 Invoice Generation Module

Purpose: Creates and stores **PDF invoices** post-appointment.

Functionality:

• Generates PDFs with **doctor, patient, and booking details**.
Includes fees, date, and unique **invoice number**.

• Utilizes the **ReportLab library** for clean PDF creation.
Layouts are styled for **professional formatting**.

• Saves invoice records in the **invoices table** with metadata.
PDFs can be downloaded from the **patient dashboard**.

Implementation:

/generate_invoice route handles **PDF creation and storage**.

3.6 Database Module

Purpose: Centralizes and manages **data operations**.

Functionality:

• Uses **normalized tables** with proper **foreign key constraints**.
Ensures relational integrity across all **linked entities**.

• Adds **indexes** to critical columns for faster performance.
Enhances speed of queries, especially in **large datasets**.

• Interacts with data using **SQLAlchemy ORM layer**.
Reduces manual SQL code with **object-based operations**.

Implementation:

Backed by **MySQL** with well-structured, optimized **queries**.

3.7 User Interface Module:

Purpose: Provides a **responsive** and **intuitive interface**.

Functionality:

- **Bootstrap-based templates** for login, dashboard, and form interfaces.
  Designed for clarity, usability, and **consistent user experience**.

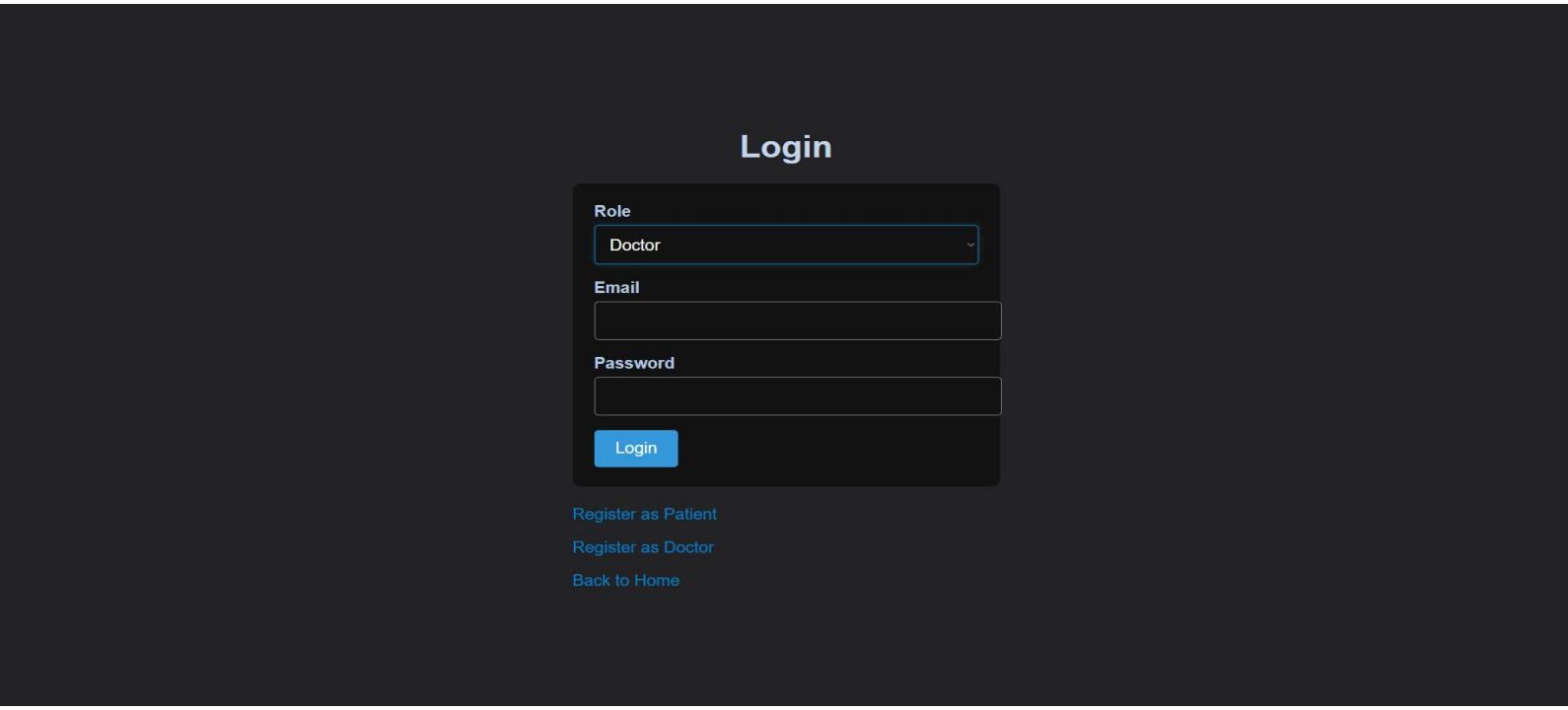
- **Role-based dashboards** customized for **patients** and **doctors**.
  Each role sees relevant data with a personalized **navigation flow**.
- **Responsive design** works seamlessly on **mobile** and **desktop** devices.
  Ensures **accessibility** and ease of use across various screen sizes.

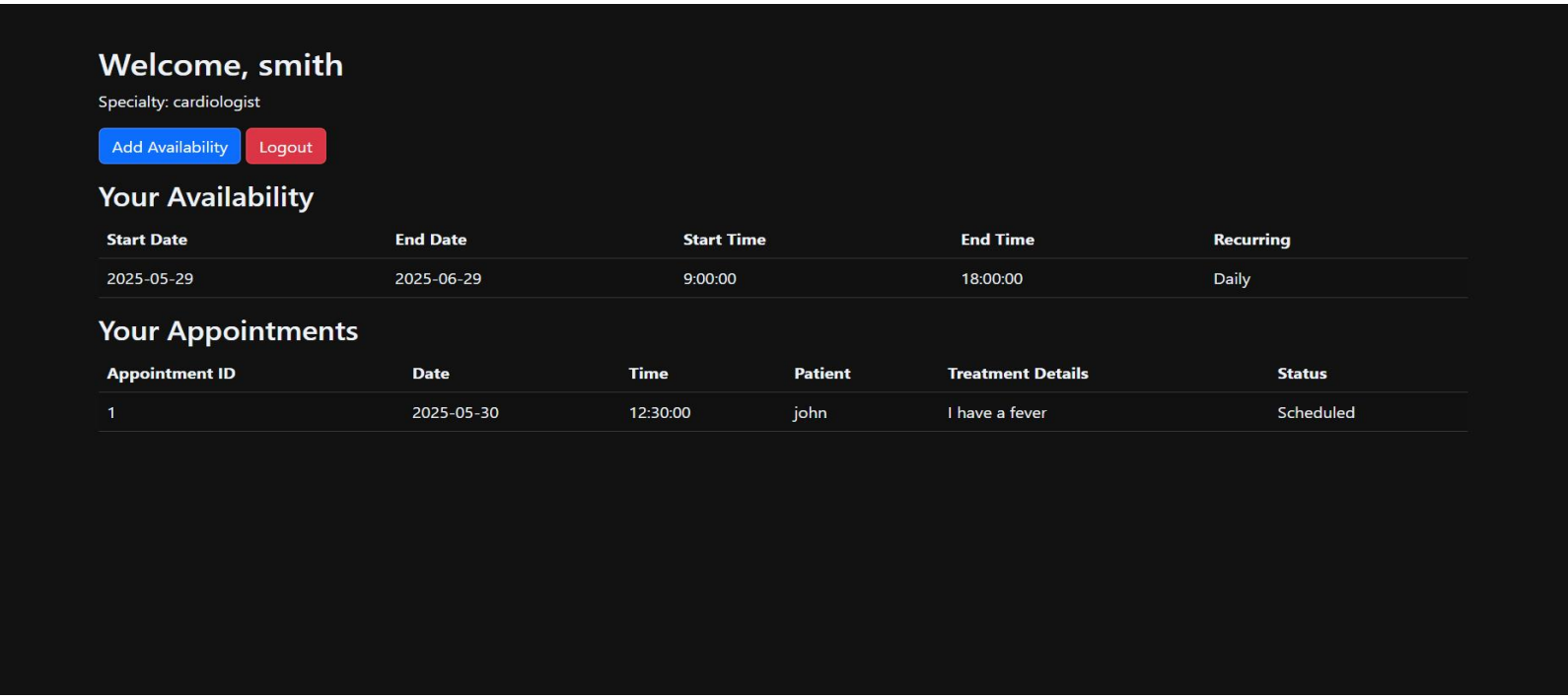
Implementation:

HTML templates (e.g., **patient_dashboard.html**) styled using **Bootstrap CSS**.

CHAPTER 4

RESULTS AND DISCUSSION





CHAPTER 5
CONCLUSION

The HMS demonstrates the effective application of database management principles in a real-world healthcare system. By automating key processes, it enhances efficiency, security, and user satisfaction. The project meets the objectives of CGB1221, showcasing skills in Flask, MySQL, and Bootstrap. Future work could expand the system with features like patient medical records and real-time analytics.

REFERENCES:

• Flask Documentation, https://flask.palletsprojects.com/, 2024.

` • MySQL Reference Manual, https://dev.mysql.com/doc/, 2024.

• Bootstrap 5 Documentation, https://getbootstrap.com/docs/5.0/, 2024.

• Elmasri, R., Navathe, S., ”Fundamentals of Database Systems,” 7th Ed., Pearson, 2016.

APPENDIX

(Coding)

```
from flask import Flask, render_template, request, redirect, url_for, session, send_file
from flask_mysqldb import MySQL
import MySQLdb.cursors
import os
from datetime import datetime, time, timedelta
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
import io

app = Flask(__name__)
app.secret_key = 'your_secret_key'

# MySQL configurations
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'abineshpower007'
app.config['MYSQL_DB'] = 'hospital_db'

mysql = MySQL(app)

# Initialize database
def init_db():
    try:
        cursor = mysql.connection.cursor()
        # Create patients table
```

```
cursor.execute('''CREATE TABLE IF NOT EXISTS patients (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
)''')
# Create doctors table
cursor.execute('''CREATE TABLE IF NOT EXISTS doctors (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    specialty VARCHAR(100) NOT NULL
)''')
# Create appointments table
cursor.execute('''CREATE TABLE IF NOT EXISTS appointments (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    patient_id INTEGER,
    doctor_id INTEGER,
    appointment_date DATE NOT NULL,
    appointment_time TIME,
    treatment_details TEXT,
    diagnosis TEXT,
    fees DECIMAL(10,2),
    status VARCHAR(50) DEFAULT 'Scheduled',
    FOREIGN KEY (patient_id) REFERENCES patients(id),
    FOREIGN KEY (doctor_id) REFERENCES doctors(id)
)''')
# Create availability_slots table with date range and recurrence
```

```
                cursor.execute('''CREATE TABLE IF NOT EXISTS availability_slots (
                    id INTEGER PRIMARY KEY AUTO_INCREMENT,
                    doctor_id INTEGER,
                    start_date DATE NOT NULL,
                    end_date DATE NOT NULL,
                    start_time TIME NOT NULL,
                    end_time TIME NOT NULL,
                    is_recurring BOOLEAN DEFAULT FALSE,
                    FOREIGN KEY (doctor_id) REFERENCES doctors(id)
                )''')
                # Insert sample doctors
                sample_doctors = [
                    ('Dr. John Doe', 'john.doe@example.com', 'password123', 'Cardiology'),
                    ('Dr. Jane Smith', 'jane.smith@example.com', 'password123', 'Pediatrics'),
                    ('Dr. Alice Brown', 'alice.brown@example.com', 'password123', 'Neurology')
                ]
                for doctor in sample_doctors:
                    cursor.execute('SELECT * FROM doctors WHERE email = %s', (doctor[1],))
                    if not cursor.fetchone():
                        cursor.execute('INSERT INTO doctors (name, email, password, specialty)
VALUES (%s, %s, %s, %s)', doctor)
                mysql.connection.commit()
                cursor.close()
            except Exception as e:
                print(f"Database initialization error: {e}")
                Raise
            # Home route
        @app.route('/')
        def home():
```

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


# Combined Login
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        role = request.form['role']
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        if role == 'patient':
            cursor.execute('SELECT * FROM patients WHERE email = %s AND password = %s',
(email, password))
            user = cursor.fetchone()
            if user:
                session['patient_id'] = user['id']
                cursor.close()
                return redirect(url_for('patient_dashboard'))
        elif role == 'doctor':
            cursor.execute('SELECT * FROM doctors WHERE email = %s AND password = %s',
(email, password))
            user = cursor.fetchone()
            if user:
                session['doctor_id'] = user['id']
                cursor.close()
                return redirect(url_for('doctor_dashboard'))

        cursor.close()
        return render_template('login.html', error='Invalid credentials or role')
```

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


    # Patient Registration
    @app.route('/register', methods=['GET', 'POST'])
    def register():
        if request.method == 'POST':
            name = request.form['name']
            email = request.form['email']
            password = request.form['password']
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('SELECT * FROM patients WHERE email = %s', [email])
            existing_patient = cursor.fetchone()
            if existing_patient:
                cursor.close()
                return render_template('register.html', error='Email already registered')
            cursor.execute('INSERT INTO patients (name, email, password) VALUES (%s, %s, %s)',
(name, email, password))
            mysql.connection.commit()
            cursor.close()
            return redirect(url_for('login'))
        return render_template('register.html')


    # Doctor Registration
    @app.route('/doctor_register', methods=['GET', 'POST'])
    def doctor_register():
        if request.method == 'POST':
            name = request.form['name']
            email = request.form['email']
            password = request.form['password']
```

```
            specialty = request.form['specialty']

            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

            cursor.execute('SELECT * FROM doctors WHERE email = %s', [email])

            existing_doctor = cursor.fetchone()

            if existing_doctor:

                cursor.close()

                return render_template('doctor_register.html', error='Email already registered')

            cursor.execute('INSERT INTO doctors (name, email, password, specialty) VALUES
(%s, %s, %s, %s)',

                                (name, email, password, specialty))

            mysql.connection.commit()

            cursor.close()

            return redirect(url_for('login'))

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