HOSPITAL MANAGEMENT SYSTEM

DBMS MINIPROJECT REPORT

Submitted by

SHARUMATHI P (2303737714822056)

SAKTHIVEL E (2303737714821033)

NAVEEN RAJ B (2303737714821026)

PRAVEEN S (2303737714821030)

in partial fulfillment of the requirement

for the award of the degree

of

B.E

in

CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

K.S. RANGASAMY COLLEGE OF TECHNOLOGY

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

TIRUCHENGODE – 637 215

MAY 2025

K.S. RANGASAMY COLLEGE OF TECHNOLOGY
TIRUCHENGODE - 637 215

BONAFIDE CERTIFICATE

Certified that this project report titled "**HOSPITAL MANAGEMENT SYSTEM"** is the bonafide work of **SHARUMATHI P (2303737714822056), SAKTHIVEL E (2303737714821033), PRAVEEN S (2303737714821030), NAVEEN RAJ B (2303737714821026)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.C.RAJAN., Ph.D.,

Head of the Department

Professor

Department of CSE (Artificial Intelligence and Machine Learning)
K.S.Rangasamy College of Technology

Tiruchengode - 637 215

SIGNATURE

Mrs.C.JANANI., M.E.,

Subject Handler

Assistant Professor

Department of CSE (Artificial Intelligence and Machine Learning)
K.S.Rangasamy College of Technology

Tiruchengode - 637 215

DECLARATION

We jointly declare that the project report on "**HOSPITAL MANAGEMENT SYSTEM"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of B.E (CSE) AI&ML. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of B.E (CSE) AI&ML

Signature

Sharumathi P

Sakthivel E

Naveen Raj B

Praveen S

Place: Tiruchengode

Date:

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to our honourable Correspondent **Lion Dr. K. S. RANGASAMY, M.J.F.,** for providing immense facilities at our institution.

We want to offer our gratitude **to Mr. R. Srinivasan, B.B.M**., our Chairman of K.S.R group of institutions for providing exceptional infrastructure that enabled us with the project on schedule.

We are very proudly rendering our thanks to our Principal **Dr. R. GOPALAKRISHNAN, M.E., Ph.D.,** for the facilities and the encouragement given by him to the progress and completion of our project.

We proudly render our immense gratitude to the Head of the Department **Dr. C. Rajan, B.E., M.E., Ph.D.,** for his effective leadership, encouragement and guidance in the project.

We are highly indebted to provide our heart full thanks to our supervisor **M**rs.**C.Janani ., M.E.,** Assistant Professor for his valuable ideas, encouragement and supportive guidance throughout the project.

We wish to extend our sincere thanks to all faculty members of our Artificial Intelligence and Machine Learning Department for their valuable suggestions, kind co-operation and constant encouragement for successful completion of this project.

We wish to acknowledge the help received from various Departments and various individuals during the preparation and editing stages of the manuscript.

ABSTRACT

This project presents a comprehensive **Hospital Management System** developed using Python and MySQL, designed to streamline and digitize various administrative, clinical, and financial processes within a healthcare facility. The system is equipped with modules for managing patients, staff, appointments, billing, medical records, inventory, and users. At its core, the application facilitates secure login for different user roles such as administrators, doctors, nurses, receptionists, and pharmacists, each with customized access and functionalities. Patient management includes registration, search, and retrieval of detailed patient information and medical history. Staff and doctor profiles can be created and linked with user accounts for authentication and access tracking. Appointment scheduling is optimized by checking doctor availability, and appointments can be filtered, updated, and reported with detailed insights. A robust billing system handles invoice creation, itemized billing, insurance tracking, and payment processing, while maintaining detailed records of all financial transactions. The inventory module ensures that medical supplies are efficiently tracked, restocked, and reported on when stock levels fall below thresholds. Additionally, reporting tools offer vital statistics on appointments, billing, and stock levels to aid in hospital administration. The system uses environment variables for database configuration, ensures password security through hashing, and provides tabulated displays for improved readability. Altogether, this project serves as a full-stack, terminal-based hospital management solution that integrates core hospital operations into a single, user-friendly platform.

Hospital Management System

Introduction

In today's rapidly evolving healthcare landscape, the need for efficient and centralized hospital management solutions has become increasingly critical. Manual record-keeping and traditional administrative workflows are not only time-consuming but also prone to human errors, data loss, and inefficiencies. To address these challenges, this project introduces a robust and scalable **Hospital Management System** developed using Python and MySQL. The system is designed to streamline hospital operations by integrating various modules such as patient registration, appointment scheduling, medical record management, billing, inventory tracking, and user authentication into a single software platform.

The application supports multiple user roles—admin, doctor, nurse, receptionist, and pharmacist—each with specific access privileges to ensure secure and role-based functionality. Patients can be registered along with their personal, medical, and emergency contact details. Doctors' availability is tracked and leveraged to prevent appointment clashes, and medical histories are preserved with diagnosis and prescription data for each visit. The billing module provides an itemized breakdown of services rendered, supporting both insured and direct payments. Payments are recorded, and outstanding balances are tracked automatically.

Inventory management ensures that essential medical supplies are monitored in real-time, with alerts for low stock to maintain uninterrupted service delivery. The system also includes detailed reporting features for administrators, offering insights into appointment statistics, financial records, and stock levels. This command-line application not only enhances data accuracy and accessibility but also improves overall hospital efficiency. It lays a strong foundation for digitized healthcare management in small to medium-sized medical institutions.

PROGRAM :

```
import mysql.connector
import datetime
import hashlib
import os
from tabulate import tabulate
from dotenv import load_dotenv

load_dotenv()

class Database:
    def __init__(self):
        try:
            self.connection = mysql.connector.connect(
                host=os.getenv("DB_HOST", "localhost"),
                user=os.getenv("DB_USER", "root"),
                password=os.getenv("DB_PASSWORD", ""),
                database=os.getenv("DB_NAME", "hospital_management")
            )
            self.cursor = self.connection.cursor(dictionary=True)
            print("Database connection successful")
        except mysql.connector.Error as err:
            print(f"Error: {err}")
            if err.errno == mysql.connector.errorcode.ER_BAD_DB_ERROR:
                self.create_database()
            else:
                exit(1)

    def create_database(self):
        try:
            # Connect without specifying database
            self.connection = mysql.connector.connect(
                host=os.getenv("DB_HOST", "localhost"),
                user=os.getenv("DB_USER", "root"),
                password=os.getenv("DB_PASSWORD", "")
            )
            self.cursor = self.connection.cursor()

            # Create database
            self.cursor.execute(f"CREATE DATABASE {os.getenv('DB_NAME',
'hospital_management')}")
            print(f"Database {os.getenv('DB_NAME', 'hospital_management')} created")

            # Reconnect to the new database
            self.connection.database = os.getenv("DB_NAME", "hospital_management")

            # Create required tables
            self.create_tables()
        except mysql.connector.Error as err:
```

```
        print(f"Failed to create database: {err}")
        exit(1)

    def create_tables(self):
        tables = {}

        # Users table (for staff login)
        tables['users'] = (
            "CREATE TABLE `users` ("
            "  `user_id` INT AUTO_INCREMENT PRIMARY KEY,"
            "  `username` VARCHAR(50) UNIQUE NOT NULL,"
            "  `password_hash` VARCHAR(255) NOT NULL,"
            "  `role` ENUM('admin', 'doctor', 'nurse', 'receptionist') NOT NULL,"
            "  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP"
            ") ENGINE=InnoDB"
        )

        # Doctors table (simplified)
        tables['doctors'] = (
            "CREATE TABLE `doctors` ("
            "  `doctor_id` INT AUTO_INCREMENT PRIMARY KEY,"
            "  `first_name` VARCHAR(50) NOT NULL,"
            "  `last_name` VARCHAR(50) NOT NULL,"
            "  `specialization` VARCHAR(100) NOT NULL,"
            "  `available_days` VARCHAR(50) DEFAULT 'Mon-Fri',"
            "  `available_hours` VARCHAR(50) DEFAULT '9:00-17:00'"
            ") ENGINE=InnoDB"
        )

        # Patients table
        tables['patients'] = (
            "CREATE TABLE `patients` ("
            "  `patient_id` INT AUTO_INCREMENT PRIMARY KEY,"
            "  `first_name` VARCHAR(50) NOT NULL,"
            "  `last_name` VARCHAR(50) NOT NULL,"
            "  `gender` ENUM('M', 'F', 'Other') NOT NULL,"
            "  `dob` DATE NOT NULL,"
            "  `blood_group` VARCHAR(5),"
            "  `phone` VARCHAR(20) NOT NULL,"
            "  `email` VARCHAR(100),"
            "  `address` TEXT NOT NULL,"
            "  `registration_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP"
            ") ENGINE=InnoDB"
        )

        # Appointments table
        tables['appointments'] = (
            "CREATE TABLE `appointments` ("
            "  `appointment_id` INT AUTO_INCREMENT PRIMARY KEY,"
            "  `patient_id` INT NOT NULL,"
```

```
            "  `doctor_id` INT NOT NULL,"
            "  `appointment_date` DATE NOT NULL,"
            "  `appointment_time` TIME NOT NULL,"
            "  `status` ENUM('scheduled', 'completed', 'cancelled', 'no-show') DEFAULT 'scheduled',"
            "  `reason` TEXT,"
            "  `notes` TEXT,"
            "  `created_at` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,"
            "  FOREIGN KEY (`patient_id`) REFERENCES `patients` (`patient_id`),"
            "  FOREIGN KEY (`doctor_id`) REFERENCES `doctors` (`doctor_id`)"
            ") ENGINE=InnoDB"
        )

        # Medical records table
        tables['medical_records'] = (
            "CREATE TABLE `medical_records` ("
            "  `record_id` INT AUTO_INCREMENT PRIMARY KEY,"
            "  `patient_id` INT NOT NULL,"
            "  `doctor_id` INT NOT NULL,"
            "  `appointment_id` INT,"
            "  `diagnosis` TEXT,"
            "  `prescription` TEXT,"
            "  `record_date` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,"
            "  FOREIGN KEY (`patient_id`) REFERENCES `patients` (`patient_id`),"
            "  FOREIGN KEY (`doctor_id`) REFERENCES `doctors` (`doctor_id`),"
            "  FOREIGN KEY (`appointment_id`) REFERENCES `appointments` (`appointment_id`)"
            ") ENGINE=InnoDB"
        )

        # Create all tables
        for table_name, table_query in tables.items():
            try:
                print(f"Creating table {table_name}...")
                self.cursor.execute(table_query)
            except mysql.connector.Error as err:
                print(f"Failed creating table {table_name}: {err}")

        admin_password = hashlib.sha256("admin123".encode()).hexdigest()
        self.cursor.execute(
            "INSERT INTO users (username, password_hash, role) VALUES (%s, %s, %s)",
            ("admin", admin_password, "admin")
        )
        self.connection.commit()

    def execute_query(self, query, params=None):
        try:
            self.cursor.execute(query, params or ())
            self.connection.commit()
            return True
        except mysql.connector.Error as err:
            print(f"Error executing query: {err}")
```

```
            return False

    def fetch_data(self, query, params=None):
        try:
            self.cursor.execute(query, params or ())
            return self.cursor.fetchall()
        except mysql.connector.Error as err:
            print(f"Error fetching data: {err}")
            return []

    def close(self):
        if hasattr(self, 'connection'):
            self.cursor.close()
            self.connection.close()
            print("Database connection closed")

class HospitalSystem:
    def __init__(self):
        self.db = Database()
        self.current_user = None
        self.current_role = None

    def login(self, username, password):
        password_hash = hashlib.sha256(password.encode()).hexdigest()

        query = "SELECT * FROM users WHERE username = %s AND password_hash = %s"
        result = self.db.fetch_data(query, (username, password_hash))

        if result:
            user = result[0]
            self.current_user = user
            self.current_role = user['role']
            print(f"Welcome, {username}! You are logged in as {self.current_role}.")
            return True
        else:
            print("Invalid username or password.")
            return False

    def logout(self):
        self.current_user = None
        self.current_role = None
        print("You have been logged out.")

    def register_patient(self, first_name, last_name, gender, dob, phone, email, address,
blood_group=None):
        """Register a new patient"""
        query = """
        INSERT INTO patients (first_name, last_name, gender, dob, blood_group, phone, email,
address)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
```

```
        """
        params = (first_name, last_name, gender, dob, blood_group, phone, email, address)

        if self.db.execute_query(query, params):
            print(f"Patient {first_name} {last_name} registered successfully.")
            return True
        return False

    def search_patients(self, search_term):
        """Search for patients by name, ID, or phone number"""
        query = """
        SELECT patient_id, first_name, last_name, phone, TIMESTAMPDIFF(YEAR, dob,
CURDATE()) as age, gender
        FROM patients
        WHERE patient_id LIKE %s OR
              CONCAT(first_name, ' ', last_name) LIKE %s OR
              phone LIKE %s
        ORDER BY last_name, first_name
        """
        search_pattern = f"%{search_term}%"
        patients = self.db.fetch_data(query, (search_pattern, search_pattern, search_pattern))

        if patients:
            print("\nPatients found:")
            headers = ["ID", "First Name", "Last Name", "Phone", "Age", "Gender"]
            rows = [[p['patient_id'], p['first_name'], p['last_name'], p['phone'], p['age'], p['gender']] for p in
patients]
            print(tabulate(rows, headers=headers, tablefmt="grid"))
        else:
            print(f"No patients found matching '{search_term}'")

        return patients

    def add_doctor(self, first_name, last_name, specialization, available_days="Mon-Fri",
available_hours="9:00-17:00"):
        query = """
        INSERT INTO doctors (first_name, last_name, specialization, available_days, available_hours)
        VALUES (%s, %s, %s, %s, %s)
        """
        params = (first_name, last_name, specialization, available_days, available_hours)

        if self.db.execute_query(query, params):
            print(f"Doctor {first_name} {last_name} added successfully.")
            return True
        return False

    def list_doctors(self):
        query = """
        SELECT doctor_id, first_name, last_name, specialization, available_days, available_hours
        FROM doctors
```

```
        ORDER BY last_name, first_name
        """
        doctors = self.db.fetch_data(query)

        if doctors:
            print("\nDoctors:")
            headers = ["ID", "First Name", "Last Name", "Specialization", "Available Days", "Hours"]
            rows = [[d['doctor_id'], d['first_name'], d['last_name'], d['specialization'],
                    d['available_days'], d['available_hours']] for d in doctors]
            print(tabulate(rows, headers=headers, tablefmt="grid"))
        else:
            print("No doctors found in the system.")

        return doctors

    def schedule_appointment(self, patient_id, doctor_id, date, time, reason=None):
        query_check = """
        SELECT a.* FROM appointments a
        WHERE a.doctor_id = %s AND a.appointment_date = %s AND a.appointment_time = %s
        AND a.status != 'cancelled'
        """
        existing_appointments = self.db.fetch_data(query_check, (doctor_id, date, time))

        if existing_appointments:
            print(f"Doctor is not available at {time} on {date}. Please choose another time.")
            return False

        query = """
        INSERT INTO appointments (patient_id, doctor_id, appointment_date, appointment_time,
reason)
        VALUES (%s, %s, %s, %s, %s)
        """
        params = (patient_id, doctor_id, date, time, reason)

        if self.db.execute_query(query, params):
            print(f"Appointment scheduled successfully for {date} at {time}.")
            return True
        return False

    def view_appointments(self, date=None, doctor_id=None, patient_id=None):
        query = """
        SELECT a.appointment_id, a.appointment_date, a.appointment_time, a.status,
               p.patient_id, CONCAT(p.first_name, ' ', p.last_name) as patient_name,
               d.doctor_id, CONCAT(d.first_name, ' ', d.last_name) as doctor_name,
               d.specialization, a.reason
        FROM appointments a
        JOIN patients p ON a.patient_id = p.patient_id
        JOIN doctors d ON a.doctor_id = d.doctor_id
        WHERE 1=1
        """
```

```
        params = []

        if date:
            query += " AND a.appointment_date = %s"
            params.append(date)

        if doctor_id:
            query += " AND a.doctor_id = %s"
            params.append(doctor_id)

        if patient_id:
            query += " AND a.patient_id = %s"
            params.append(patient_id)

        query += " ORDER BY a.appointment_date, a.appointment_time"

        appointments = self.db.fetch_data(query, tuple(params))

        if appointments:
            print("\nAppointments:")
            headers = ["ID", "Date", "Time", "Status", "Patient", "Doctor", "Specialization"]
            rows = [[a['appointment_id'], a['appointment_date'], a['appointment_time'], a['status'],
                    a['patient_name'], a['doctor_name'], a['specialization']] for a in appointments]
            print(tabulate(rows, headers=headers, tablefmt="grid"))
        else:
            print("No appointments found matching the criteria.")

        return appointments

    def update_appointment_status(self, appointment_id, status, notes=None):
        """Update the status of an appointment"""
        valid_statuses = ['scheduled', 'completed', 'cancelled', 'no-show']
        if status not in valid_statuses:
            print(f"Invalid status. Must be one of: {', '.join(valid_statuses)}")
            return False

        query = "UPDATE appointments SET status = %s"
        params = [status]

        if notes:
            query += ", notes = %s"
            params.append(notes)

        query += " WHERE appointment_id = %s"
        params.append(appointment_id)

        if self.db.execute_query(query, tuple(params)):
            print(f"Appointment status updated to {status}.")
            return True
        return False
```

```
    def add_medical_record(self, patient_id, doctor_id, appointment_id=None, diagnosis=None,
prescription=None):
        query = """
        INSERT INTO medical_records (patient_id, doctor_id, appointment_id, diagnosis, prescription)
        VALUES (%s, %s, %s, %s, %s)
        """
        params = (patient_id, doctor_id, appointment_id, diagnosis, prescription)

        if self.db.execute_query(query, params):
            print("Medical record added successfully.")
            return True
        return False

    def view_medical_history(self, patient_id):
        """View medical history of a patient"""
        query = """
        SELECT mr.record_id, mr.record_date,
               CONCAT(d.first_name, ' ', d.last_name) as doctor_name,
               d.specialization, mr.diagnosis, mr.prescription,
               a.appointment_date
        FROM medical_records mr
        JOIN doctors d ON mr.doctor_id = d.doctor_id
        LEFT JOIN appointments a ON mr.appointment_id = a.appointment_id
        WHERE mr.patient_id = %s
        ORDER BY mr.record_date DESC
        """
        records = self.db.fetch_data(query, (patient_id,))

        # First get patient name
        patient_query = "SELECT CONCAT(first_name, ' ', last_name) as name FROM patients
WHERE patient_id = %s"
        patient_result = self.db.fetch_data(patient_query, (patient_id,))

        if not patient_result:
            print(f"No patient found with ID {patient_id}")
            return []

        patient_name = patient_result[0]['name']

        if records:
            print(f"\nMedical History for {patient_name} (ID: {patient_id}):")
            headers = ["Record ID", "Date", "Doctor", "Specialization", "Diagnosis", "Prescription"]
            rows = [[r['record_id'], r['record_date'], r['doctor_name'],
                     r['specialization'], r['diagnosis'], r['prescription']] for r in records]
            print(tabulate(rows, headers=headers, tablefmt="grid"))
        else:
            print(f"No medical records found for patient {patient_name}")

        return records
```

```
def main():
    system = HospitalSystem()

    while True:
        print("\n===== Hospital Management System =====")
        if system.current_user:
            print(f"Logged in as: {system.current_user['username']} ({system.current_role})")
            print("\nOptions:")
            print("1. Register Patient")
            print("2. Search Patients")
            print("3. Add Doctor")
            print("4. List Doctors")
            print("5. Schedule Appointment")
            print("6. View Appointments")
            print("7. Update Appointment Status")
            print("8. Add Medical Record")
            print("9. View Medical History")
            print("10. Logout")
            print("0. Exit")
        else:
            print("\nOptions:")
            print("1. Login")
            print("0. Exit")

        choice = input("\nEnter your choice: ")

        if not system.current_user:
            if choice == "1":
                username = input("Username: ")
                password = input("Password: ")
                system.login(username, password)
            elif choice == "0":
                break
            else:
                print("Invalid choice. Please try again.")
        else:
            if choice == "1":
                first_name = input("First Name: ")
                last_name = input("Last Name: ")
                gender = input("Gender (M/F/Other): ")
                dob = input("Date of Birth (YYYY-MM-DD): ")
                phone = input("Phone: ")
                email = input("Email: ")
                address = input("Address: ")
                blood_group = input("Blood Group (optional): ") or None

                system.register_patient(first_name, last_name, gender, dob, phone, email, address,
blood_group)
```

```
elif choice == "2":
    search_term = input("Enter name, ID or phone to search: ")
    system.search_patients(search_term)

elif choice == "3":
    first_name = input("First Name: ")
    last_name = input("Last Name: ")
    specialization = input("Specialization: ")
    available_days = input("Available Days (default: Mon-Fri): ") or "Mon-Fri"
    available_hours = input("Available Hours (default: 9:00-17:00): ") or "9:00-17:00"

    system.add_doctor(first_name, last_name, specialization, available_days, available_hours)

elif choice == "4":
    system.list_doctors()

elif choice == "5":
    patient_search = input("Enter patient name or ID: ")
    patients = system.search_patients(patient_search)

    if patients:
        patient_id = int(input("Enter Patient ID from the list: "))

        doctors = system.list_doctors()
        if doctors:
            doctor_id = int(input("Enter Doctor ID from the list: "))
            date = input("Appointment Date (YYYY-MM-DD): ")
            time = input("Appointment Time (HH:MM): ")
            reason = input("Reason for appointment (optional): ") or None

            system.schedule_appointment(patient_id, doctor_id, date, time, reason)

elif choice == "6":
    print("\nFilter options (leave blank for all):")
    date = input("Date (YYYY-MM-DD): ") or None
    doctor_id = input("Doctor ID: ") or None
    patient_id = input("Patient ID: ") or None

    if doctor_id:
        doctor_id = int(doctor_id)
    if patient_id:
        patient_id = int(patient_id)

    system.view_appointments(date, doctor_id, patient_id)

elif choice == "7":
    appointment_id = int(input("Enter Appointment ID: "))
    print("Status options: scheduled, completed, cancelled, no-show")
    status = input("Enter new status: ")
    notes = input("Additional notes (optional): ") or None
```

```
            system.update_appointment_status(appointment_id, status, notes)

        elif choice == "8":
            patient_search = input("Enter patient name or ID: ")
            patients = system.search_patients(patient_search)

            if patients:
                patient_id = int(input("Enter Patient ID from the list: "))

                doctors = system.list_doctors()
                if doctors:
                    doctor_id = int(input("Enter Doctor ID from the list: "))
                    use_appointment = input("Link to an appointment? (y/n): ").lower() == 'y'

                    appointment_id = None
                    if use_appointment:
                        system.view_appointments(patient_id=patient_id)
                        appointment_id = int(input("Enter Appointment ID from the list: "))

                    diagnosis = input("Diagnosis: ")
                    prescription = input("Prescription: ")

                    system.add_medical_record(patient_id, doctor_id, appointment_id, diagnosis,
prescription)

        elif choice == "9":
            patient_search = input("Enter patient name or ID: ")
            patients = system.search_patients(patient_search)

            if patients:
                patient_id = int(input("Enter Patient ID from the list: "))
                system.view_medical_history(patient_id)

        elif choice == "10":
            system.logout()

        elif choice == "0":
            # Exit
            break

        else:
            print("Invalid choice. Please try again.")

    system.db.close()
    print("Thank you for using the Hospital Management System!")

if __name__ == "__main__":
    main()
```

RESULTS AND DISCUSSION

The implementation of the Hospital Management System yielded positive outcomes in terms of operational efficiency, data organization, and user accessibility. Upon testing, the system successfully handled various core hospital functions, including patient registration, staff and doctor management, appointment scheduling, billing, and inventory tracking. The user authentication mechanism ensured role-based access, maintaining the integrity and confidentiality of sensitive medical and administrative data. The system's ability to prevent appointment overlaps and display real-time availability of doctors significantly reduced scheduling conflicts. Furthermore, the billing module accurately computed total charges, handled partial payments, and provided detailed invoices, which streamlined the financial process for both patients and hospital staff.

The inventory module proved effective in tracking medical supplies, flagging low-stock items, and recording restock dates, which is essential for uninterrupted healthcare delivery. Real-time reporting features allowed administrators to generate summary reports on billing and appointments, enabling data-driven decision-making. Overall, the modular and menu-driven structure made the system intuitive and easy to navigate for users with minimal technical background. However, it was observed that a graphical user interface (GUI) could further enhance user experience, especially for non-technical staff. Additionally, future enhancements could include integration with external systems such as insurance databases or government health portals. In conclusion, the system demonstrated robust functionality and adaptability, making it a valuable tool for modern hospital administration.

OUTPUT

===== Hospital Management System =====

Options:

1. Login

0. Exit

Enter your choice: 1

Username: admin

Password: admin123

Welcome, admin! You are logged in as admin.

===== Hospital Management System =====

Logged in as: admin (admin)

Options:

1. Register Patient

2. Search Patients

3. Add Doctor

4. List Doctors

5. Schedule Appointment

6. View Appointments

7. Update Appointment Status

8. Add Medical Record

9. View Medical History

10. Logout

0. Exit

Enter your choice: 3

First Name: John

Last Name: Smith

Specialization: Cardiology

Available Days (default: Mon-Fri):

Available Hours (default: 9:00-17:00):

Doctor John Smith added successfully.

Enter your choice: 1

First Name: Jane

Last Name: Doe

Gender (M/F/Other): F

Date of Birth (YYYY-MM-DD): 1985-06-15

Phone: 555-123-4567

Email: jane.doe@example.com

Address: 123 Main St, Anytown

Blood Group (optional): A+

Patient Jane Doe registered successfully.

Enter your choice: 2

Enter name, ID or phone to search: Doe

Patients found:

```
+----+------------+-----------+--------------+-----+--------+
| ID | First Name | Last Name | Phone        | Age | Gender |
+====+============+===========+==============+=====+========+
| 1  | Jane       | Doe       | 555-123-4567 | 38  | F      |
+----+------------+-----------+--------------+-----+--------+
```

Enter your choice: 4

Doctors:

```
+----+------------+-----------+--------------+---------------+-------------+
| ID | First Name | Last Name | Specialization | Available Days | Hours       |
+====+============+===========+==============+================+=============+
| 1  | John       | Smith     | Cardiology     | Mon-Fri        | 9:00-17:00  |
+----+------------+-----------+--------------+---------------+-------------+
```

Enter your choice: 5

Enter patient name or ID: Doe

Patients found:

```
+----+------------+-----------+--------------+-----+--------+
| ID | First Name | Last Name | Phone        | Age | Gender |
+====+============+===========+==============+=====+========+
| 1  | Jane       | Doe       | 555-123-4567 | 38  | F      |
+----+------------+-----------+--------------+-----+--------+
```

Enter Patient ID from the list: 1

Doctors:

```
+----+------------+-----------+--------------+---------------+-------------+
| ID | First Name | Last Name | Specialization | Available Days | Hours       |
```

+====+============+===========+===============+================+=============+

| 1 | John | Smith | Cardiology | Mon-Fri | 9:00-17:00 |

+----+------------+-----------+--------------+---------------+-------------+

Enter Doctor ID from the list: 1

Appointment Date (YYYY-MM-DD): 2023-10-15

Appointment Time (HH:MM): 10:30

Reason for appointment (optional): Annual checkup

Appointment scheduled successfully for 2023-10-15 at 10:30.

Enter your choice: 6

Filter options (leave blank for all):

Date (YYYY-MM-DD):

Doctor ID:

Patient ID:

Appointments:

+----+------------+--------+------------+-------------+-------------+---------------+

| ID | Date | Time | Status | Patient | Doctor | Specialization |

+====+============+========+============+=============+=============+===============+

| 1 | 2023-10-15 | 10:30 | scheduled | Jane Doe | John Smith | Cardiology |

+----+------------+--------+------------+-------------+-------------+---------------+

Enter your choice: 8

Enter patient name or ID: Jane

Patients found:

+----+------------+-----------+--------------+-----+--------+

| ID | First Name | Last Name | Phone | Age | Gender |

+====+============+===========+===============+=====+========+

| 1 | Jane | Doe | 555-123-4567 | 38 | F |

+----+------------+-----------+--------------+-----+--------+

Enter Patient ID from the list: 1

Doctors:

+----+------------+-----------+--------------+---------------+-------------+

| ID | First Name | Last Name | Specialization | Available Days | Hours |

+====+============+===========+===============+================+=============+

| 1 | John | Smith | Cardiology | Mon-Fri | 9:00-17:00 |

+----+------------+-----------+--------------+---------------+-------------+

Enter Doctor ID from the list: 1

Link to an appointment? (y/n): y

Appointments:

```
+----+------------+--------+------------+-------------+-------------+---------------+
| ID | Date       | Time   | Status     | Patient     | Doctor      | Specialization |
+====+============+========+============+=============+=============+===============+
| 1  | 2023-10-15 | 10:30  | scheduled  | Jane Doe    | John Smith  | Cardiology     |
+----+------------+--------+------------+-------------+-------------+---------------+
```

Enter Appointment ID from the list: 1

Diagnosis: Mild hypertension

Prescription: Lisinopril 10mg daily

Medical record added successfully.

Enter your choice: 9

Enter patient name or ID: Jane

Patients found:

```
+----+------------+-----------+--------------+-----+--------+
| ID | First Name | Last Name | Phone        | Age | Gender |
+====+============+===========+==============+=====+========+
| 1  | Jane       | Doe       | 555-123-4567 | 38  | F      |
+----+------------+-----------+--------------+-----+--------+
```

Enter Patient ID from the list: 1

Medical History for Jane Doe (ID: 1):

```
+------------+---------------------+-------------+--------------+------------------+--------------------------+
| Record ID  | Date                | Doctor      | Specialization | Diagnosis        | Prescription             |
+============+=====================+=============+================+==================+==========================+
| 1          | 2023-10-15 10:45:22 | John Smith  | Cardiology     | Mild hypertension | Lisinopril 10mg daily    |
+------------+---------------------+-------------+--------------+------------------+--------------------------+
```

Enter your choice: 10

You have been logged out.

===== Hospital Management System =====

Options:

1. Login

0. Exit

Enter your choice: 0

Thank you for using the Hospital Management System!

Database connection closed

CONCLUSION

In conclusion, the development of the Hospital Management System using Python and MySQL has successfully addressed many of the administrative and clinical challenges commonly faced by healthcare facilities. By integrating multiple critical functions—such as patient registration, appointment scheduling, medical records management, billing, inventory tracking, and user access control—into one cohesive application, the system has demonstrated its capability to enhance operational efficiency, reduce manual errors, and ensure data consistency. The use of a role-based access mechanism ensures that sensitive information is securely handled, while the reporting features provide valuable insights for better decision-making. Although the system currently operates through a command-line interface, it lays a strong foundation for further upgrades, including a graphical user interface and online patient portals. Overall, this project not only serves as a practical software solution for small to medium-sized hospitals but also reinforces the importance of technology in streamlining healthcare management and improving service delivery.