PM Shri Kendriya Vidyalaya NFC Vigyan Vihar

CLINIC MANAGEMENT SYSTEM

Submitted By:

SHUBH RASTOGI

Class: XII E

Submitted TO:

Pooja Khare

Index

Certificate:

This is to certify that ***SHUBH RASTOGI*** of class ***XII E*** of ***PM SHRI KENDRIYA VIDYALAYA NFC VIGYAN VIHAR*** has completed his project on "**CLINIC MANAGEMENT SYSTEM**" under my supervision. He has shown great interest and sincerity in completing this project

____________________ Computer Science Teacher

____________________ EXTERNAL EXAMINER

Date: _____________

Acknowledgement

I express our immense gratitude to our Computer Science teacher POOJA KHARE for her intellectual vigor and generously given support that has been invaluable in escalating our determination to reach the goal of writing this project successfully.

I can hardly find appropriate words to express our obligations and gratefulness to the principal.

I also feel immense pleasure in recording deep sense of indebtedness, gratitude and sincere thanks to all fellow group mates for their help, company and hard work.

I am especially indebted to our parents for their sincere love, moral support and spontaneous encouragement throughout the entire period of this work.

Thank you!

PROJECT SYNOPSIS:

Introdution :

The **Clinic Management System** is a software solution designed to streamline the operations of medical clinics. It provides an automated way to manage patient records, doctor schedules, and appointment bookings. By replacing manual processes with an efficient, database-driven system, this application reduces errors, saves time, and improves the overall patient experience.

The system is built using **Python** and **MySQL**, combining the ease of scripting with the robustness of relational database management. Its intuitive interface ensures that even users with limited technical expertise can interact with the system effortlessly. The Clinic Management System bridges the gap between technology and healthcare, offering an organized, secure, and user-friendly solution for clinics of any size.

Aim:

The primary aim of the **Clinic Management System** is to enhance the efficiency and reliability of clinic operations by automating routine tasks such as:

- **Managing Patient Records**: Storing and retrieving patient information securely.
- **Scheduling Appointments**: Organizing and tracking appointments to reduce waiting times and avoid conflicts.
- **Doctor Allocation**: Managing doctor schedules and specializations effectively.
- **Data Security and Accuracy**: Ensuring that all data is stored securely and can be accessed with ease when needed.

This project also aims to provide a scalable and adaptable solution that can be expanded to include additional features, such as prescription management, billing, and reporting, as the needs of the clinic grow.

Idea Source:

The idea for the **Clinic Management System** originated from the challenges faced by small and medium-sized medical clinics in managing their day-to-day operations. Some of the pain points that inspired this project include:

1. **Manual Record-Keeping**:
   - Clinics often rely on physical files to store patient and doctor data, which can be prone to loss, mismanagement, or damage.
   - Locating records during emergencies can be time-consuming.
2. **Inefficient Appointment Scheduling**:
   - Without a centralized system, overbooking and appointment conflicts are common.
   - Patients often face long waiting times due to poorly managed schedules.
3. **Error-Prone Processes**:
   - Manual methods for recording patient details, appointment bookings, and doctor schedules can lead to inaccuracies and errors.
4. **Demand for Digital Solutions in Healthcare**:
   - With the increasing adoption of technology in healthcare, clinics require automated systems to stay competitive and improve service quality.

This system addresses these challenges by offering a simple, effective, and scalable solution that automates operations, enhances accuracy, and improves the overall experience for both staff and patients.

Additional Details:

Key Features:

1. **Patient Management**:
   - Add new patient records, including name, age, gender, and phone number.
   - View and search for existing patient details.
2. **Doctor Management**:
   - Store doctor details, including their name, specialization, and contact information.
   - Retrieve and manage doctor schedules.
3. **Appointment Scheduling**:
   - Book appointments by linking patients to available doctors based on their specialization.
   - Assign specific dates and time slots for each appointment.
   - Update the status of appointments (e.g., Scheduled, Completed, or Canceled).
4. **Database Integration**:
   - Data is stored securely in a relational database (MySQL), ensuring durability and easy access.
   - Relationships between tables (e.g., patients and appointments) are maintained for seamless operations.

Future Scope:

1. **Prescription Management**:
   - Enable doctors to record and retrieve patient prescriptions.
   - Automate reminders for follow-ups or prescription renewals.
2. **Billing System**:
   - Generate and track invoices for appointments and other services.
3. **Integration with Online Portals**:
   - Allow patients to book appointments online and view their medical records securely.
4. **Reports and Analytics**:
   - Generate detailed reports for clinic performance, patient visits, and doctor availability.

Tabular Representation of data:

Table Name	Field Name	Data Type	Constraints	Description
patients	PATIENT_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each patient.
	NAME	VARCHAR(255)	NOT NULL	Name of the patient.
	AGE	INT	CHECK (AGE > 0)	Age of the patient.
	GENDER	VARCHAR(10)	CHECK (GENDER IN ('Male', 'Female', 'Other'))	Gender of the patient.
	PHONE_NO	VARCHAR(15)	NOT NULL, UNIQUE	Patient's phone number.
doctors	DOCTOR_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each doctor.
	NAME	VARCHAR(255)	NOT NULL	Name of the doctor.
	SPECIALIZATION	VARCHAR(255)	NOT NULL	Doctor's area of expertise (e.g., ENT, Cardiology).
	PHONE_NO	VARCHAR(15)	NOT NULL, UNIQUE	Doctor's contact number.
appointments	APPOINTMENT_ID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each appointment.
	PATIENT_ID	INT	FOREIGN KEY REFERENCES patients(PATIENT_ID)	The patient associated with the appointment.
	DOCTOR_ID	INT	FOREIGN KEY REFERENCES doctors(DOCTOR_ID)	The doctor assigned for the appointment.
	APPOINTMENT_DATE	DATE	NOT NULL	The date of the appointment.
	TIME_SLOT	VARCHAR(10)	NOT NULL	The time slot assigned for the appointment.
	STATUS	VARCHAR(20)	DEFAULT 'Scheduled'	Status of the appointment (Scheduled, Completed, Canceled).

Validation and Add on Features

- ● If a user enters an invalid input, the system will prompt them to reattempt, ensuring a seamless experience. The program is designed to be intuitive and user-centric, making it easy to navigate and interact with. Moreover, the code includes specific features designed to cater to unique requirements, which are clearly detailed and implemented.

SOURCE CODE:

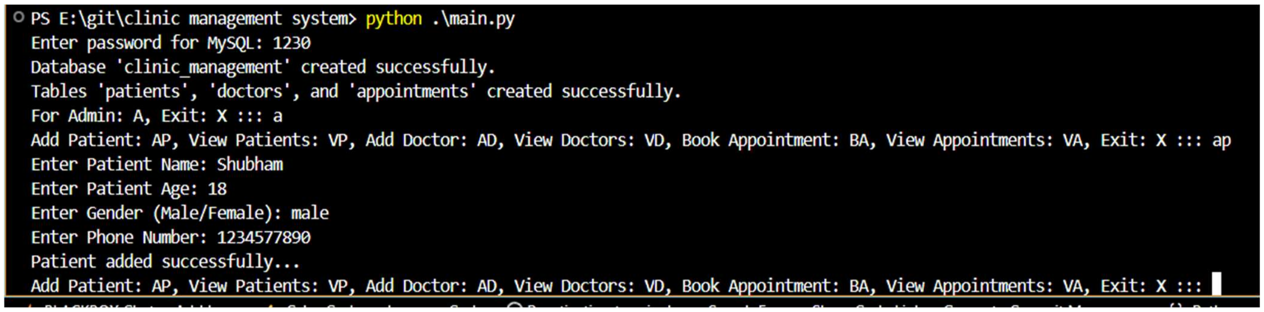
```
import mysql.connector as pymysql

from datetime import datetime

passwrd = None

db = None

C = None

def base_check():

     check = 0

    db = pymysql.connect(host="localhost", user="root", 
password=passwrd)

    cursor = db.cursor()

    cursor.execute('SHOW DATABASES')

    result = cursor.fetchall()

    for r in result:

        for i in r:

            if i == 'clinic_management':
                cursor.execute('USE clinic_management')

                check = 1

    if check != 1:

        create_database()

def table_check():

    db = pymysql.connect(host="localhost", user="root", 
password=passwrd)

    cursor = db.cursor()

    cursor.execute('SHOW DATABASES')

    result = cursor.fetchall()

    for r in result:

        for i in r:

            if i == 'clinic_management':

                cursor.execute('USE clinic_management')

                cursor.execute('SHOW TABLES')

                result = cursor.fetchall()

                if len(result) < 3:
                    create_tables()

                else:

                    print('      Booting systems...')

def create_database():

    try:

        db = pymysql.connect(host="localhost", user="root",
password=passwrd)

        cursor = db.cursor()

        cursor.execute("CREATE DATABASE IF NOT EXISTS
clinic_management")

        db.commit()

        db.close()

        print("Database 'clinic_management' created
successfully.")

    except pymysql.Error as e:

        print(f"Error creating database: {str(e)}")

def create_tables():

    try:
        db = pymysql.connect(host="localhost", user="root",
password=passwrd, database="clinic_management")

        cursor = db.cursor()

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS patients (

                PATIENT_ID INT PRIMARY KEY AUTO_INCREMENT,

                NAME VARCHAR(255),

                AGE INT,

                GENDER VARCHAR(10),

                PHONE_NO VARCHAR(15)

            )

        """)

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS doctors (

                DOCTOR_ID INT PRIMARY KEY AUTO_INCREMENT,

                NAME VARCHAR(255),

                SPECIALIZATION VARCHAR(255),
                PHONE_NO VARCHAR(15)

            )

        """)

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS appointments (

                APPOINTMENT_ID INT PRIMARY KEY AUTO_INCREMENT,

                PATIENT_ID INT,

                DOCTOR_ID INT,

                APPOINTMENT_DATE DATE,

                TIME_SLOT VARCHAR(10),

                STATUS VARCHAR(20) DEFAULT 'Scheduled',

                FOREIGN KEY (PATIENT_ID) REFERENCES
patients(PATIENT_ID),

                FOREIGN KEY (DOCTOR_ID) REFERENCES
doctors(DOCTOR_ID)

            )

        """)

        db.commit()
        db.close()

        print("Tables 'patients', 'doctors', and
'appointments' created successfully.")

    except pymysql.Error as e:

        print(f"Error creating tables: {str(e)}")

def add_patient():

    name = input("Enter Patient Name: ")

    age = int(input("Enter Patient Age: "))

    gender = input("Enter Gender (Male/Female): ")

    phone_no = input("Enter Phone Number: ")

    data = (name, age, gender, phone_no)

    sql = "INSERT INTO patients (NAME, AGE, GENDER, PHONE_NO)
VALUES (%s, %s, %s, %s)"

    try:

        C.execute(sql, data)

        db.commit()

        print('Patient added successfully...')

    except pymysql.Error as e:
        print(f"Error adding patient: {str(e)}")

def view_patients():

    C.execute("SELECT * FROM patients")

    result = C.fetchall()

    for r in result:

        print(r)

def add_doctor():

    name = input("Enter Doctor Name: ")

    specialization = input("Enter Specialization: ")

    phone_no = input("Enter Phone Number: ")

    data = (name, specialization, phone_no)

    sql = "INSERT INTO doctors (NAME, SPECIALIZATION,
PHONE_NO) VALUES (%s, %s, %s)"

    try:

        C.execute(sql, data)

        db.commit()

        print('Doctor added successfully...')
    except pymysql.Error as e:

        print(f"Error adding doctor: {str(e)}")

def view_doctors():

    C.execute("SELECT * FROM doctors")

    result = C.fetchall()

    for r in result:

        print(r)

def book_appointment():

    patient_id = int(input("Enter Patient ID: "))

    doctor_id = int(input("Enter Doctor ID: "))

    appointment_date = input("Enter Appointment Date (YYYY-MM-
DD): ")

    time_slot = input("Enter Time Slot (e.g., 10:00 AM): ")

    data = (patient_id, doctor_id, appointment_date,
time_slot)

    sql = "INSERT INTO appointments (PATIENT_ID, DOCTOR_ID,
APPOINTMENT_DATE, TIME_SLOT) VALUES (%s, %s, %s, %s)"

    try:
        C.execute(sql, data)

        db.commit()

        print('Appointment booked successfully...')

    except pymysql.Error as e:

        print(f"Error booking appointment: {str(e)}")

def view_appointments():

    C.execute("""

        SELECT

            a.APPOINTMENT_ID,

            p.NAME AS PATIENT_NAME,

            d.NAME AS DOCTOR_NAME,

            a.APPOINTMENT_DATE,

            a.TIME_SLOT,

            a.STATUS

        FROM appointments a

        JOIN patients p ON a.PATIENT_ID = p.PATIENT_ID
        JOIN doctors d ON a.DOCTOR_ID = d.DOCTOR_ID

    """)

    result = C.fetchall()

    for r in result:

        print(r)

def main():

    global passwrd

    passwrd = input("Enter password for MySQL: ")

    base_check()

    table_check()

    global db, C

    db = pymysql.connect(host="localhost", user="root",
password=passwrd, database="clinic_management")

    C = db.cursor()

    while True:

        log = input("For Admin: A, Exit: X ::: ")

        if log.upper() == "A":
```

```
        while True:

            menu = input('''Add Patient: AP, View
Patients: VP, Add Doctor: AD, View Doctors: VD, Book
Appointment: BA, View Appointments: VA, Exit: X ::: ''')

            if menu.upper() == 'AP':

                add_patient()

            elif menu.upper() == 'VP':

                view_patients()

            elif menu.upper() == 'AD':

                add_doctor()

            elif menu.upper() == 'VD':

                view_doctors()

            elif menu.upper() == 'BA':

                book_appointment()

            elif menu.upper() == 'VA':

                view_appointments()

            elif menu.upper() == 'X':

                break
                else:
                    print("Wrong Input")
        elif log.upper() == "X":
            print("THANK YOU FOR USING CLINIC MANAGEMENT
SYSTEM")
            break
        else:
            print("Wrong Input")
if __name__ == "__main__":
  main()
```

OUTPUT:

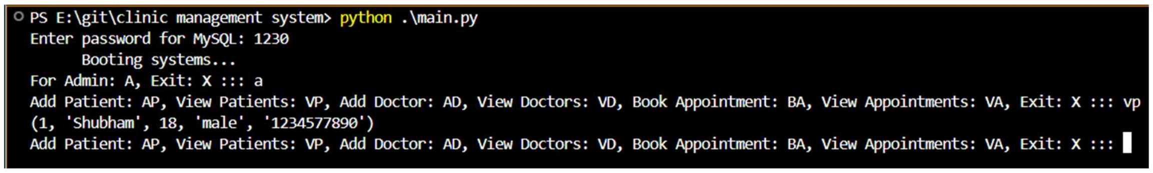
MAIN ADMIN:

➢ADD PATIENT:



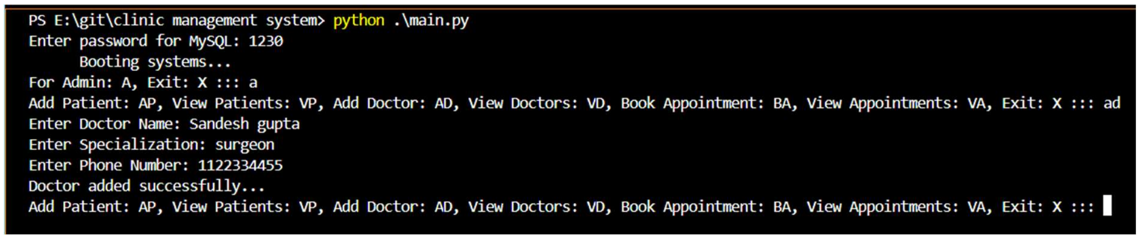
```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
Database 'clinic_management' created successfully.
Tables 'patients', 'doctors', and 'appointments' created successfully.
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: ap
Enter Patient Name: Shubham
Enter Patient Age: 18
Enter Gender (Male/Female): male
Enter Phone Number: 1234577890
Patient added successfully...
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X :::
```

➢VIEW PATIENT:



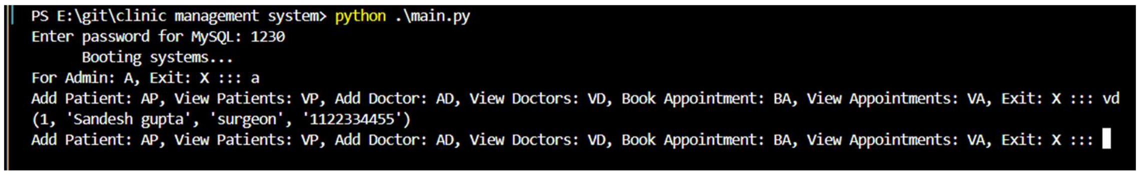
```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: vp
(1, 'Shubham', 18, 'male', '1234577890')
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X :::
```

➢ADD DOCTOR:



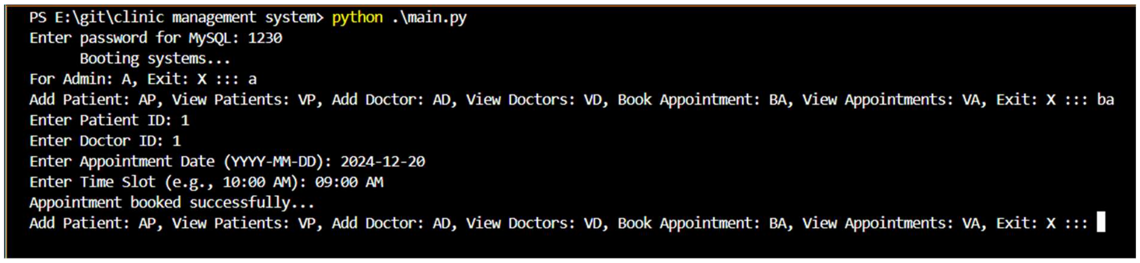
```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: ad
Enter Doctor Name: Sandesh gupta
Enter Specialization: surgeon
Enter Phone Number: 1122334455
Doctor added successfully...
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X :::
```

➢VIEW DOCTOR:



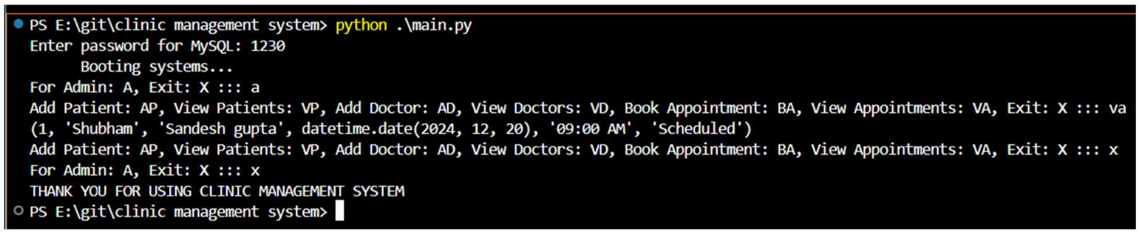
```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: vd
(1, 'Sandesh gupta', 'surgeon', '1122334455')
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X :::
```

➢BOOK APPOINTMENT:



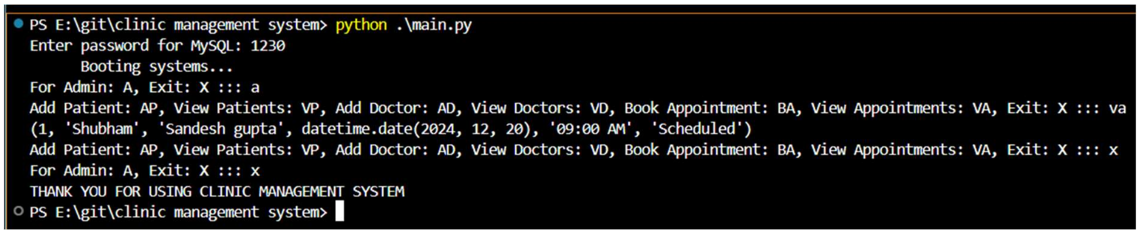
```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: ba
Enter Patient ID: 1
Enter Doctor ID: 1
Enter Appointment Date (YYYY-MM-DD): 2024-12-20
Enter Time Slot (e.g., 10:00 AM): 09:00 AM
Appointment booked successfully...
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X :::
```

➢VIEW APPOINTMENT:



```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: va
(1, 'Shubham', 'Sandesh gupta', datetime.date(2024, 12, 20), '09:00 AM', 'Scheduled')
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: x
For Admin: A, Exit: X ::: x
THANK YOU FOR USING CLINIC MANAGEMENT SYSTEM
PS E:\git\clinic management system>
```

➢EXIT:



```
PS E:\git\clinic management system> python .\main.py
Enter password for MySQL: 1230
      Booting systems...
For Admin: A, Exit: X ::: a
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: va
(1, 'Shubham', 'Sandesh gupta', datetime.date(2024, 12, 20), '09:00 AM', 'Scheduled')
Add Patient: AP, View Patients: VP, Add Doctor: AD, View Doctors: VD, Book Appointment: BA, View Appointments: VA, Exit: X ::: x
For Admin: A, Exit: X ::: x
THANK YOU FOR USING CLINIC MANAGEMENT SYSTEM
PS E:\git\clinic management system>
```

Hardware Requirement
PC/Laptop/MacBook
with Intel
core/i3/i5/i7 or any
equivalent With at
least 2 GB RAM 10
MB free space on
Hard

Disk LCD/LED

Operating System & Compiler
MS Windows/Ubuntu/MacOS

Python IDLE 3.x

OR

colab.research.google.com (gmail account)

and

MySQL 8.x

References

1.Classnotes

2.www.w3schools.com

3.www.geekforgeeks.com

4.Friends