PM Shri Kendriya Vidyalaya NFC Vigyan Vihar

**CLINIC MANAGEMENT SYSTEM**



Submitted By: Submitted TO:

SHUBH RASTOGI Pooja Khare

Class: XII E

Index

|  |  |  |
| --- | --- | --- |
| Sno | Description | Pageno |
| 1 | Certificate | 3 |
| 2 | Acknowledgement & References | 4 |
| 3 | Project Synopsis | 5 |
| 4 | Source Code | 12 |
| 5 | Output Screen | 17 |
| 6 | Hardware & Software requirement | 24 |
| 7 | Biblography | 27 |

# **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:

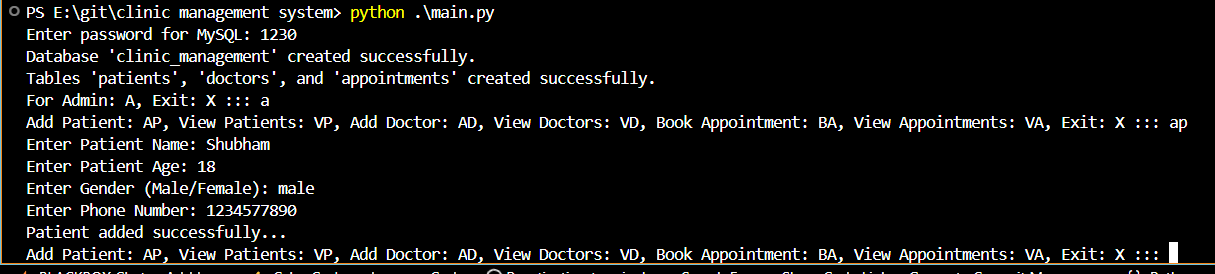
print("Wrong Input")

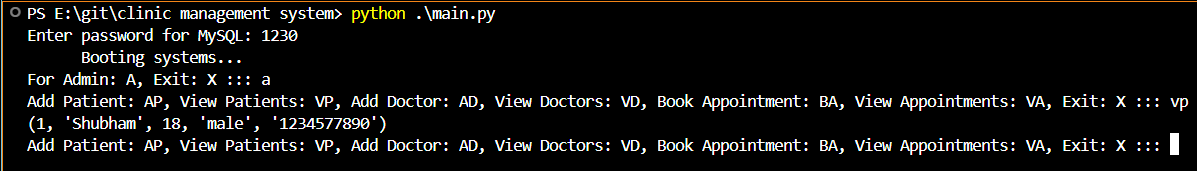
if \_\_name\_\_ == "\_\_main\_\_":

main()

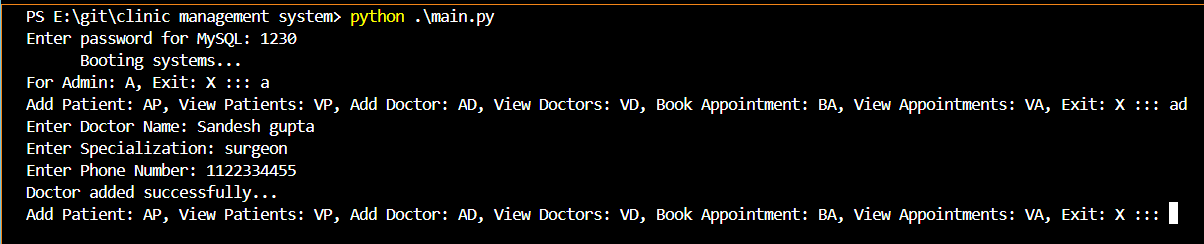
# **OUTPUT:**

**MAIN ADMIN:**

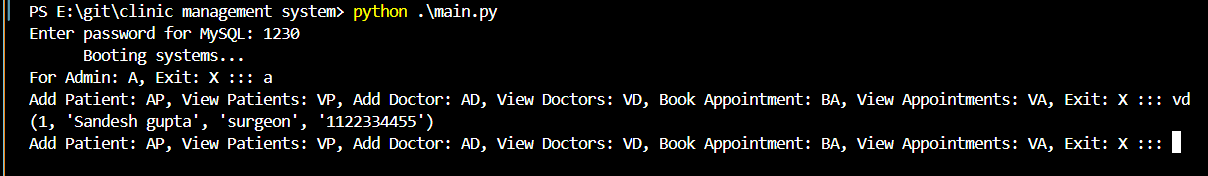
* ADD PATIENT:
* VIEW PATIENT:



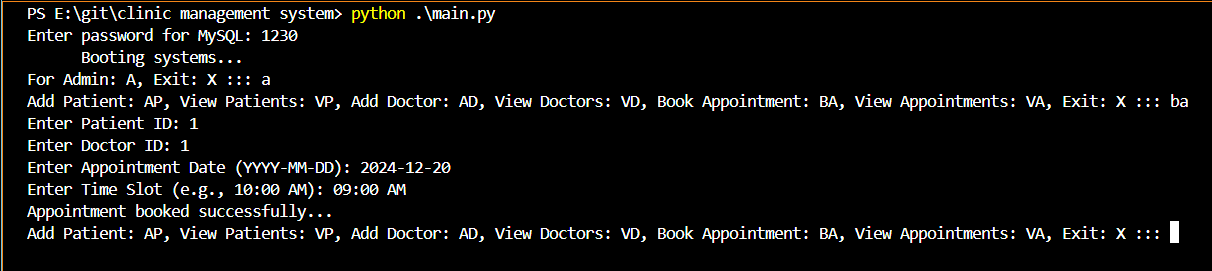
* ADD DOCTOR:



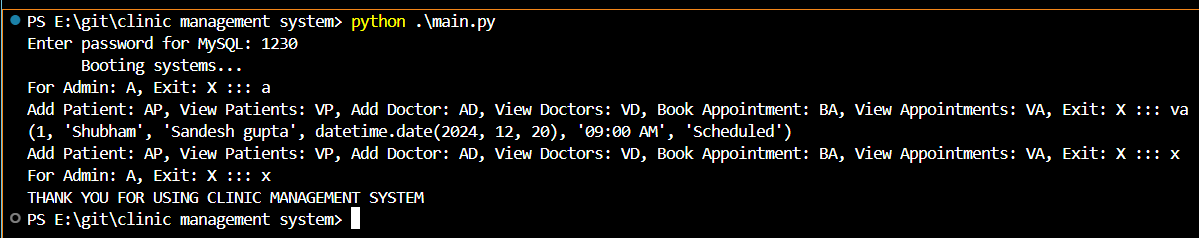
* VIEW DOCTOR:



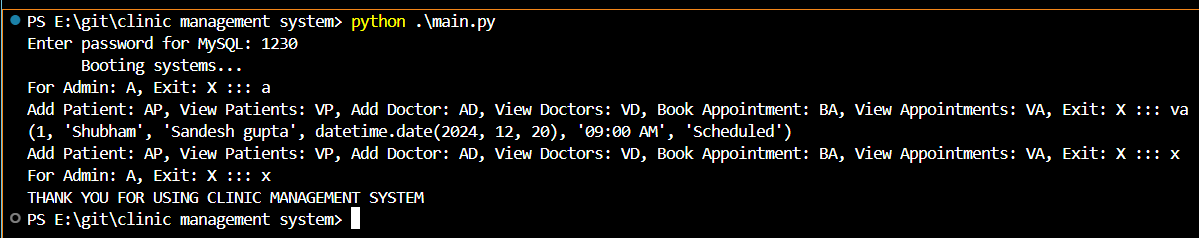
* BOOK APPOINTMENT:



* VIEW APPOINTMENT:



* EXIT:



|  |
| --- |
| 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