**ALL SAINT’S CHURCH SR.SEC. SCHOOL**

**M.I. ROAD , JAIPUR**

**A PROJECT REPORT ON**

**HOSPITAL MANAGEMENT SYSTEM**

**SUBJECT: INFORMATICS PRACTICES (065)**

**Session: 2023-2024**

**SUBMITTED BY SUBMITTED TO-**

**Faizan khan , Mrs.Sharon Hiskiel**

**Ayaan khan , Sufiyan Qureshi**

**CERTIFICATE**

**This is to certify that Faizan , Sufiyan and Ayaan of class XII SCIENCE has sucessfully completed the project on the topic School Management System, In Partial fulfilment of the requirement for the AISSCE Partal Examination of the subject code Informatics Practices(065).**

**The project work reported here is as per the guidelines of CBSE for AISSCE Practical Examination and it s done under the supervision Mrs. Sharon Hiskie, PGT COMPUTER. The project work, carried out by us is not a form of any other project work.**

**Internal Examiner Principal**

**External Examiner School Seal**

**ACKNOWLEDGMENT**

**I would like to express my special thanks to my teacher Mrs. Sharon Hiskiel for mentoring me throughout this project work. I also thank our respected principal Mrs. Shabnam Haque for her motivation and guidance throughout the year.**

**My project is titled as “Hospital Management System” and it has enabled me to do a lot of research and I came to Know about so many new things in software design and development.**

**Also, I would also like to thank my parents who motivated and supported me during my work.**

**Faizan , Sufiyan , Ayaan**

**(XII)**

**INDEX**

* **Python Introduction**
* **MySQL Introduction**
* **Hardware Requirements**
* **Introduction to project****(python)**
* **Database schema Screenshots****(SQL)**
* **User Output**
* **SQL Queries**
* **User Interface Code**
* **Testing**
* **Bibliography**

**PYTHON INTRODUCTION**

**Python is a general purpose, dynamic, high-level, and interpreted programming language. Python is a high level language. It is a free and open source language. It is an interpreted language, as python programs are executed by an interpreter. Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.**

**Pandas is a Python library used for working with data sets.**

**It has functions for analyzing, cleaning, exploring, and manipulating data.**

**The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.**

**Pandas can clean messy data sets, and make them readable and relevant.**

**Relevant data is very important in data science.**

**Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.**

**MySQL INTRODUCTION**

**The “Hospital Management System” created by me is based on PYTHON AND MYSQL.**

**Its an automation of the existing system which enables its user to perform few operations pertaining to management of Hospital as listed below.**

**The Project Enables its user to:**

**1.) Add new Patient, new Doctor and new Fee records.**

**2.) Delete Patient, Doctor and Fee records.**

**3.) Update Patient, Doctor and Fee records.**

**4.) View Patient, Doctor and Fee records from the Database.**

**System Requirements**

**-------------------|Hardware Requirements |--------------**

**Device name DESKTOP-LFQNHSQ**

**Processor Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz**

**Installed RAM 8.00 GB (7.88 GB usable)**

**Device ID A123CFF6-C1D0-41EE-A3A4-21D7FAEBE73B**

**Product ID 00330-50419-68709-AAOEM**

**System type 64-bit operating system, x64-based processor**

**Pen and touch Touch support with 10 touch points**

**INTRODUCTION OF PROJECT**

**PROJECT TITLE-(HOSPITAL MANAGEMENT)**

**DBMS: MySQL**

**Host : localhost**

**User: root Password: root Database: HOSPITAL**

**Table Structure: As per the Screenshot given below:**

**Screenshots OF PROJECT**

**Hosp table has following Schema**

**Staff table has following Schema**

**Fee table has following Schema**

**USER OUTPUT**

**STUDENT MODULE DETAILS:**

**STAFF MODULE DETAILS:**

**FEE MODULE DETAILS:**

**EXIT MODULE DETAILS:**

**EXIT MODULE DETAILS:**

**CHARTS**

**Chart b/w Staff Name & Salary:**

**Chart :Students vs Amount Paid:**

**Chart: Students vs Ages:**

**SQL QUERIES**

**USER OUTPUT(SOURCE CODE)**

**import mysql.connector**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**# Connecting to the MySQL server**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user='root',**

**password='root'**

**)**

**print("-" \* 165)**

**print(" " \* 68 + "Welcome to Hospital Management System")**

**print("-" \* 165)**

**# Creating a database**

**def create\_database():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')**

**cursor.execute('USE Hosp')**

**# Creating Patients table**

**def create\_patients\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')**

**# Creating Doctors table**

**def create\_doctors\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')**

**# Creating Appointments table**

**def create\_appointments\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO\_INCREMENT PRIMARY KEY, patient\_id INT, doctor\_id INT, date DATE, time VARCHAR(255))')**

**# Define the function to add a new patient**

**def add\_patient():**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"**

**val = (name, age, gender, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view patient details**

**def view\_patients():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Patients")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='g':# Adding line chart**

**plt.figure(figsize=(8, 6))**

**plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')**

**plt.xlabel('Patient ID')**

**plt.ylabel('Age')**

**plt.title('Age Distribution of Patients')**

**plt.show()**

**# Define the function to update patient details**

**def update\_patient():**

**patient\_id = int(input("Enter patient ID to update: "))**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"**

**val = (name, age, gender, contact, patient\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete patient details**

**def delete\_patient():**

**patient\_id = int(input("Enter patient ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Patients WHERE Id = %s"**

**val = (patient\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new doctor**

**def add\_doctor():**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"**

**val = (name, specialization, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view doctor details**

**def view\_doctors():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Doctors")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**import mysql.connector**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**# Connecting to the MySQL server**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user='root',**

**password='root'**

**)**

**print("-" \* 165)**

**print(" " \* 68 + "Welcome to Hospital Management System")**

**print("-" \* 165)**

**# Creating a database**

**def create\_database():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')**

**cursor.execute('USE Hosp')**

**# Creating Patients table**

**def create\_patients\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')**

**# Creating Doctors table**

**def create\_doctors\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')**

**# Creating Appointments table**

**def create\_appointments\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO\_INCREMENT PRIMARY KEY, patient\_id INT, doctor\_id INT, date DATE, time VARCHAR(255))')**

**# Define the function to add a new patient**

**def add\_patient():**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"**

**val = (name, age, gender, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view patient details**

**def view\_patients():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Patients")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='g':# Adding line chart**

**plt.figure(figsize=(8, 6))**

**plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')**

**plt.xlabel('Patient ID')**

**plt.ylabel('Age')**

**plt.title('Age Distribution of Patients')**

**plt.show()**

**# Define the function to update patient details**

**def update\_patient():**

**patient\_id = int(input("Enter patient ID to update: "))**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"**

**val = (name, age, gender, contact, patient\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete patient details**

**def delete\_patient():**

**patient\_id = int(input("Enter patient ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Patients WHERE Id = %s"**

**val = (patient\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new doctor**

**def add\_doctor():**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"**

**val = (name, specialization, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view doctor details**

**def view\_doctors():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Doctors")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**df = pd.DataFrame({'name': lst1, 'specialization': lst2, 'contact': lst3})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting data for the pie chart**

**specializations = [row[2] for row in result]**

**unique\_specializations = list(set(specializations))**

**specialization\_counts = [specializations.count(spec) for spec in unique\_specializations]**

**# Plotting the pie chart**

**plt.figure(figsize=(8, 8))**

**plt.pie(specialization\_counts, labels=unique\_specializations, autopct='%1.1f%%', startangle=140)**

**plt.title('Specialization Distribution of Doctors')**

**plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.**

**plt.show()**

**# Define the function to update doctor details**

**def update\_doctor():**

**doctor\_id = int(input("Enter doctor ID to update: "))**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"**

**val = (name, specialization, contact, doctor\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete doctor details**

**def delete\_doctor():**

**doctor\_id = int(input("Enter doctor ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Doctors WHERE Id = %s"**

**val = (doctor\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new appointment**

**def add\_appointment():**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Appointments (patient\_id, doctor\_id, date, time) VALUES (%s, %s, %s, %s)"**

**val = (patient\_id, doctor\_id, date, time)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view appointment details**

**def view\_appointments():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Appointments")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")#patient\_id, doctor\_id, date, time**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'patient\_id': lst1, 'doctor\_id': lst2, 'date': lst3, 'time': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting dates from the 'date' column**

**dates = [row[3] for row in result]**

**# Counting the occurrences of each date**

**date\_counts = {}**

**for date in dates:**

**date\_counts[date] = date\_counts.get(date, 0) + 1**

**# Sorting the dates for better visualization**

**sorted\_dates = sorted(date\_counts.keys())**

**# Creating a bar chart**

**plt.figure(figsize=(10, 6))**

**plt.bar(sorted\_dates, [date\_counts[date] for date in sorted\_dates])**

**plt.xlabel('Date')**

**plt.ylabel('Number of Appointments')**

**plt.title('Number of Appointments on Each Date')**

**plt.xticks(rotation=45)**

**plt.show()**

**# Define the function to update appointment details**

**def update\_appointment():**

**appointment\_id = int(input("Enter appointment ID to update: "))**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Appointments SET patient\_id = %s, doctor\_id = %s, date = %s, time = %s WHERE Id = %s"**

**val = (patient\_id, doctor\_id, date, time, appointment\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete appointment details**

**def delete\_appointment():**

**appointment\_id = int(input("Enter appointment ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Appointments WHERE Id = %s"**

**val = (appointment\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Get the user's choice**

**def getchoice():**

**while True:**

**create\_database()**

**create\_patients\_table()**

**create\_doctors\_table()**

**create\_appointments\_table()**

**print("1. Add Patient 2. View Patients 3. Update Patient 4. Delete Patient")**

**print("5. Add Doctor 6. View Doctors 7. Update Doctor 8. Delete Doctor")**

**print("9. Add Appointment 10. View Appointments 11. Update Appointment 12. Delete Appointment")**

**print("13. Exit")**

**opp = input("Enter your choice: ")**

**if opp == '1':**

**add\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '2':**

**view\_patients()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '3':**

**update\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '4':**

**delete\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '5':**

**add\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '6':**

**view\_doctors()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '7':**

**update\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '8':**

**delete\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '9':**

**add\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '10':**

**view\_appointments()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '11':**

**update\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '12':**

**delete\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '13':**

**print('Exited !')**

**break**

**# Recall Choice function**

**getchoice()**

**# Disconnecting from the server**

**mydb.close()import mysql.connector**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**# Connecting to the MySQL server**

**mydb = mysql.connector.connect(**

**host="localhost",**

**user='root',**

**password='root'**

**)**

**print("-" \* 165)**

**print(" " \* 68 + "Welcome to Hospital Management System")**

**print("-" \* 165)**

**# Creating a database**

**def create\_database():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE DATABASE IF NOT EXISTS Hosp')**

**cursor.execute('USE Hosp')**

**# Creating Patients table**

**def create\_patients\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Patients (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), age INT, gender VARCHAR(10), contact VARCHAR(15))')**

**# Creating Doctors table**

**def create\_doctors\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Doctors (Id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), specialization VARCHAR(255), contact VARCHAR(15))')**

**# Creating Appointments table**

**def create\_appointments\_table():**

**cursor = mydb.cursor()**

**cursor.execute('CREATE TABLE IF NOT EXISTS Appointments (Id INT AUTO\_INCREMENT PRIMARY KEY, patient\_id INT, doctor\_id INT, date DATE, time VARCHAR(255))')**

**# Define the function to add a new patient**

**def add\_patient():**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Patients (name, age, gender, contact) VALUES (%s, %s, %s, %s)"**

**val = (name, age, gender, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view patient details**

**def view\_patients():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Patients")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'Name': lst1, 'Age': lst2, 'Gender': lst3, 'contact': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch=='g':# Adding line chart**

**plt.figure(figsize=(8, 6))**

**plt.plot([row[0] for row in result], [row[2] for row in result], marker='o', linestyle='-', color='b')**

**plt.xlabel('Patient ID')**

**plt.ylabel('Age')**

**plt.title('Age Distribution of Patients')**

**plt.show()**

**# Define the function to update patient details**

**def update\_patient():**

**patient\_id = int(input("Enter patient ID to update: "))**

**name = input("Enter patient name: ")**

**age = int(input("Enter patient age: "))**

**gender = input("Enter patient gender: ")**

**contact = input("Enter patient contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Patients SET name = %s, age = %s, gender = %s, contact = %s WHERE Id = %s"**

**val = (name, age, gender, contact, patient\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete patient details**

**def delete\_patient():**

**patient\_id = int(input("Enter patient ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Patients WHERE Id = %s"**

**val = (patient\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new doctor**

**def add\_doctor():**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Doctors (name, specialization, contact) VALUES (%s, %s, %s)"**

**val = (name, specialization, contact)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view doctor details**

**def view\_doctors():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Doctors")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**df = pd.DataFrame({'name': lst1, 'specialization': lst2, 'contact': lst3})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting data for the pie chart**

**specializations = [row[2] for row in result]**

**unique\_specializations = list(set(specializations))**

**specialization\_counts = [specializations.count(spec) for spec in unique\_specializations]**

**# Plotting the pie chart**

**plt.figure(figsize=(8, 8))**

**plt.pie(specialization\_counts, labels=unique\_specializations, autopct='%1.1f%%', startangle=140)**

**plt.title('Specialization Distribution of Doctors')**

**plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.**

**plt.show()**

**# Define the function to update doctor details**

**def update\_doctor():**

**doctor\_id = int(input("Enter doctor ID to update: "))**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"**

**val = (name, specialization, contact, doctor\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete doctor details**

**def delete\_doctor():**

**doctor\_id = int(input("Enter doctor ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Doctors WHERE Id = %s"**

**val = (doctor\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new appointment**

**def add\_appointment():**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Appointments (patient\_id, doctor\_id, date, time) VALUES (%s, %s, %s, %s)"**

**val = (patient\_id, doctor\_id, date, time)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view appointment details**

**def view\_appointments():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Appointments")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")#patient\_id, doctor\_id, date, time**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'patient\_id': lst1, 'doctor\_id': lst2, 'date': lst3, 'time': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting dates from the 'date' column**

**dates = [row[3] for row in result]**

**# Counting the occurrences of each date**

**date\_counts = {}**

**for date in dates:**

**date\_counts[date] = date\_counts.get(date, 0) + 1**

**# Sorting the dates for better visualization**

**sorted\_dates = sorted(date\_counts.keys())**

**# Creating a bar chart**

**plt.figure(figsize=(10, 6))**

**plt.bar(sorted\_dates, [date\_counts[date] for date in sorted\_dates])**

**plt.xlabel('Date')**

**plt.ylabel('Number of Appointments')**

**plt.title('Number of Appointments on Each Date')**

**plt.xticks(rotation=45)**

**plt.show()**

**# Define the function to update appointment details**

**def update\_appointment():**

**appointment\_id = int(input("Enter appointment ID to update: "))**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Appointments SET patient\_id = %s, doctor\_id = %s, date = %s, time = %s WHERE Id = %s"**

**val = (patient\_id, doctor\_id, date, time, appointment\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete appointment details**

**def delete\_appointment():**

**appointment\_id = int(input("Enter appointment ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Appointments WHERE Id = %s"**

**val = (appointment\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Get the user's choice**

**def getchoice():**

**while True:**

**create\_database()**

**create\_patients\_table()**

**create\_doctors\_table()**

**create\_appointments\_table()**

**print("1. Add Patient 2. View Patients 3. Update Patient 4. Delete Patient")**

**print("5. Add Doctor 6. View Doctors 7. Update Doctor 8. Delete Doctor")**

**print("9. Add Appointment 10. View Appointments 11. Update Appointment 12. Delete Appointment")**

**print("13. Exit")**

**opp = input("Enter your choice: ")**

**if opp == '1':**

**add\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '2':**

**view\_patients()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '3':**

**update\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '4':**

**delete\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '5':**

**add\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '6':**

**view\_doctors()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '7':**

**update\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '8':**

**delete\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '9':**

**add\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '10':**

**view\_appointments()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '11':**

**update\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '12':**

**delete\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '13':**

**print('Exited !')**

**break**

**# Recall Choice function**

**getchoice()**

**# Disconnecting from the server**

**mydb.close()'specialization': lst2, 'contact': lst3})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting data for the pie chart**

**specializations = [row[2] for row in result]**

**unique\_specializations = list(set(specializations))**

**specialization\_counts = [specializations.count(spec) for spec in unique\_specializations]**

**# Plotting the pie chart**

**plt.figure(figsize=(8, 8))**

**plt.pie(specialization\_counts, labels=unique\_specializations, autopct='%1.1f%%', startangle=140)**

**plt.title('Specialization Distribution of Doctors')**

**plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.**

**plt.show()**

**# Define the function to update doctor details**

**def update\_doctor():**

**doctor\_id = int(input("Enter doctor ID to update: "))**

**name = input("Enter doctor name: ")**

**specialization = input("Enter doctor specialization: ")**

**contact = input("Enter doctor contact number: ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Doctors SET name = %s, specialization = %s, contact = %s WHERE Id = %s"**

**val = (name, specialization, contact, doctor\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete doctor details**

**def delete\_doctor():**

**doctor\_id = int(input("Enter doctor ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Doctors WHERE Id = %s"**

**val = (doctor\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Define the function to add a new appointment**

**def add\_appointment():**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "INSERT INTO Appointments (patient\_id, doctor\_id, date, time) VALUES (%s, %s, %s, %s)"**

**val = (patient\_id, doctor\_id, date, time)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) inserted.")**

**# Define the function to view appointment details**

**def view\_appointments():**

**cursor = mydb.cursor()**

**cursor.execute("SELECT \* FROM Appointments")**

**result = cursor.fetchall()**

**print("Press (l) to see in the form of list")**

**print("Press (f) to see in the form of DataFrame")**

**print("Press (g) to see in the form of graph")**

**ch = input("Enter your choice: ")#patient\_id, doctor\_id, date, time**

**if ch=='f':**

**result\_list = [list(row) for row in result]**

**lst1 = [row[0] for row in result\_list]**

**lst2 = [row[1] for row in result\_list]**

**lst3 = [row[2] for row in result\_list]**

**lst4 = [row[3] for row in result\_list]**

**df = pd.DataFrame({'patient\_id': lst1, 'doctor\_id': lst2, 'date': lst3, 'time': lst4})**

**print(df.to\_markdown())**

**elif ch=='l':**

**result\_list = [list(row) for row in result]**

**print(result\_list)**

**elif ch == 'g':**

**# Extracting dates from the 'date' column**

**dates = [row[3] for row in result]**

**# Counting the occurrences of each date**

**date\_counts = {}**

**for date in dates:**

**date\_counts[date] = date\_counts.get(date, 0) + 1**

**# Sorting the dates for better visualization**

**sorted\_dates = sorted(date\_counts.keys())**

**# Creating a bar chart**

**plt.figure(figsize=(10, 6))**

**plt.bar(sorted\_dates, [date\_counts[date] for date in sorted\_dates])**

**plt.xlabel('Date')**

**plt.ylabel('Number of Appointments')**

**plt.title('Number of Appointments on Each Date')**

**plt.xticks(rotation=45)**

**plt.show()**

**# Define the function to update appointment details**

**def update\_appointment():**

**appointment\_id = int(input("Enter appointment ID to update: "))**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_id = int(input("Enter doctor ID: "))**

**date = input("Enter appointment date (YYYY-MM-DD): ")**

**time = input("Enter appointment time (HH:MM AM/PM): ")**

**cursor = mydb.cursor()**

**sql = "UPDATE Appointments SET patient\_id = %s, doctor\_id = %s, date = %s, time = %s WHERE Id = %s"**

**val = (patient\_id, doctor\_id, date, time, appointment\_id)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) updated.")**

**# Define the function to delete appointment details**

**def delete\_appointment():**

**appointment\_id = int(input("Enter appointment ID to delete: "))**

**cursor = mydb.cursor()**

**sql = "DELETE FROM Appointments WHERE Id = %s"**

**val = (appointment\_id,)**

**cursor.execute(sql, val)**

**mydb.commit()**

**print(cursor.rowcount, "record(s) deleted.")**

**# Get the user's choice**

**def getchoice():**

**while True:**

**create\_database()**

**create\_patients\_table()**

**create\_doctors\_table()**

**create\_appointments\_table()**

**print("1. Add Patient 2. View Patients 3. Update Patient 4. Delete Patient")**

**print("5. Add Doctor 6. View Doctors 7. Update Doctor 8. Delete Doctor")**

**print("9. Add Appointment 10. View Appointments 11. Update Appointment 12. Delete Appointment")**

**print("13. Exit")**

**opp = input("Enter your choice: ")**

**if opp == '1':**

**add\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '2':**

**view\_patients()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '3':**

**update\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '4':**

**delete\_patient()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '5':**

**add\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '6':**

**view\_doctors()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '7':**

**update\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '8':**

**delete\_doctor()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '9':**

**add\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '10':**

**view\_appointments()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '11':**

**update\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '12':**

**delete\_appointment()**

**input("Press ENTER KEY to continue.....")**

**print()**

**elif opp == '13':**

**print('Exited !')**

**break**

**# Recall Choice function**

**getchoice()**

**# Disconnecting from the server**

**mydb.close()**

**CONCLUSION**

**Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. Software testing also provides an, independent view of the software to allow the business to appreciate and understand the risk at implementation of the software.**

**Test techniques include, but are not limited to the process of executing a program or an application with the intent of finding software bugs.**

**It can also be stated as the process of validating and verifying that**

**a software program/Application / product meets the business**

**and technical requirements that guided the its design and**

**development, so that it works as expected and can be**

**implemented with the same characteristics. Software**

**testing, depending on the testing method employed, can**

**be implemented at any time in the development process however**

**the most test effort is employed after the requirements have**

**been defined and coding process has been completed.**

**Future Scope of Project**

**Some suggestions for the “Future Scope of Project”:**

**1.Integration with Online Platforms:Explore the possibility of integration the "Hospital Management" wth online platforms for a more seamless experience.This could include student and parent portals, online fee payment gateways and communication platforms.**

**2.** **Enhanced Security Measures:** **Implement advanced security features to protect sensitive student and staff information. This may include two-factor authentication, encrypted databases, and regular security audits.**

**3.** **Parent-Teacher Communication Platform:** **Develop a dedicated platform for effective communication between parents and teachers.**

**This could include features such as messaging, progress reports and scheduling parent-teacher meetings.**

**4. Customization and Scalability:** **Ensure that the system is easily customizable to meet the unique needs of different schools. Make it scalable so that it can accommodate the growth of data and users**

**Over time.**

**5. User Training and Support:** **Provide comprehensive user training modules and support systems for administrators,** **teachers, and parents to ensure smooth adoption and effective use of the( Hospital Management System).**

**BIBLOGRAPHY**

* **GOOGLE**
* [**www.wikipedia.com**](http://www.wikipedia.com/)
* [**www.geeksforgeeks.org**](http://www.geeksforgeeks.org/)
* **NCERT**
* **KIPS**
* **SUMITA ARORA**
* **PREETI ARORA**