ASSESMENT(PYTHON CORE)

Healthcare Industry

Design a Python class ClinicAppointment that manages patient appointments in a clinic. The system should have the following features:

➔ Book Appointment:

- Prompt for patient name, age, mobile number, and preferred doctor.
- Show time slots (10am, 11am, 12pm, 2pm, 3pm).
- Check slot availability and confirm booking.

➔ View/Cancel Appointment:

- Allow patient to view or cancel their appointment using mobile number.

➔ Doctor Availability:

- Maintain a maximum of 3 appointments per time slot per doctor.

➔ Data Persistence:

- Store appointments in memory only (no files/dbs required).

```
class ClinicAppointment:
    def __init__(self):
        # doctor_appointments will store data in this format:
        # { doctor_name: { "10am": [patients], "11am": [...], ... } }
        self.doctor_appointments = {}
        self.time_slots = ["10am", "11am", "12pm", "2pm", "3pm"]
        self.max_per_slot = 3  # Maximum 3 appointments per slot per doctor

    def book_appointment(self):
        name = input("Enter patient name: ")
        age = input("Enter patient age: ")
        mobile = input("Enter mobile number: ")
        doctor = input("Enter preferred doctor name: ")

        # Initialize doctor if not already added
        if doctor not in self.doctor_appointments:
            self.doctor_appointments[doctor] = {slot: [] for slot in self.time_slots}

        print("\nAvailable time slots:")
        for slot in self.time_slots:
```

```
            available = self.max_per_slot -
len(self.doctor_appointments[doctor][slot])
            print(f"{slot} - {available} slot(s) left")

        slot = input("\nChoose your preferred time slot: ")

        # Check if slot exists and available
        if slot not in self.time_slots:
            print("Invalid time slot selected!")
            return

        if len(self.doctor_appointments[doctor][slot]) >=
self.max_per_slot:
            print("Sorry, this slot is full. Please choose another
time.")
            return

        # Add appointment
        appointment = {
            "name": name,
            "age": age,
            "mobile": mobile,
            "doctor": doctor,
```

```
            "slot": slot
        }

self.doctor_appointments[doctor][slot].append(appointment)
        print(f"\n Appointment booked successfully for {name}
at {slot} with Dr. {doctor}")


    def view_appointment(self):
        mobile = input("Enter your mobile number to view
appointment: ")


        found = False
        for doctor, slots in self.doctor_appointments.items():
            for slot, patients in slots.items():
                for p in patients:
                    if p["mobile"] == mobile:
                        print(f"\n Appointment Details:")
                        print(f"Name: {p['name']}")
                        print(f"Age: {p['age']}")
                        print(f"Doctor: {p['doctor']}")
                        print(f"Time Slot: {p['slot']}")
                        found = True
        if not found:
```

```
            print(" No appointment found for this mobile number.")

    def cancel_appointment(self):
        mobile = input("Enter your mobile number to cancel appointment: ")

        for doctor, slots in self.doctor_appointments.items():
            for slot, patients in slots.items():
                for p in patients:
                    if p["mobile"] == mobile:
                        slots[slot].remove(p)
                        print(f"\n Appointment for {p['name']} canceled successfully.")
                        return
        print(" No appointment found to cancel.")

    def show_menu(self):
        while True:
            print("\n--- Clinic Appointment System ---")
            print("1. Book Appointment")
            print("2. View Appointment")
            print("3. Cancel Appointment")
```

```
            print("4. Exit")

            choice = input("Enter your choice: ")

            if choice == '1':
                self.book_appointment()
            elif choice == '2':
                self.view_appointment()
            elif choice == '3':
                self.cancel_appointment()
            elif choice == '4':
                print(" Thank you! Exiting system.")
                break
            else:
                print("Invalid choice. Please try again.")
# Run the system
if __name__ == "__main__":
    clinic = ClinicAppointment()
    clinic.show_menu()
```

School Management System

Design a Python class SchoolManagement that helps manage student admissions and records. The system should support:

➔ New Admission:

- Collect student name, age, class (1–12), and guardian's mobile number.
- Assign a unique student ID automatically.
- Validate age: must be between 5 and 18.
- Validate mobile number: must be 10 digits.

➔ View Student Details:

- Allow lookup using student ID.

➔ Update Student Info:

- Update mobile number or class.

➔ Remove Student Record:

- Remove a student using their student ID.

➔ Exit System

```
class SchoolManagement:
    def __init__(self):
        # Dictionary to store all student records
        # Format: {student_id: {"name": ..., "age": ..., "class": ..., "mobile": ...}}
        self.students = {}
        self.next_id = 1  # Auto-increment student ID

    def new_admission(self):
        print("\n--- New Student Admission ---")
        name = input("Enter student name: ")
        age = int(input("Enter student age: "))
        student_class = int(input("Enter student class (1–12): "))
        mobile = input("Enter guardian's mobile number: ")

        #  Validations
        if age < 5 or age > 18:
            print(" Invalid age! Must be between 5 and 18.")
            return
        if not (mobile.isdigit() and len(mobile) == 10):
            print(" Invalid mobile number! Must be exactly 10 digits.")
```

```
            return
        if student_class < 1 or student_class > 12:
            print(" Invalid class! Must be between 1 and 12.")
            return

        #  Assign unique ID
        student_id = self.next_id
        self.next_id += 1

        #  Store details
        self.students[student_id] = {
            "name": name,
            "age": age,
            "class": student_class,
            "mobile": mobile
        }

        print(f"\n Admission successful! Student ID is
{student_id}")

    def view_student(self):
        print("\n--- View Student Details ---")
```

```
    student_id = int(input("Enter student ID: "))

    if student_id in self.students:
        s = self.students[student_id]
        print(f"\n📋 Student Details (ID: {student_id})")
        print(f"Name: {s['name']}")
        print(f"Age: {s['age']}")
        print(f"Class: {s['class']}")
        print(f"Guardian's Mobile: {s['mobile']}")
    else:
        print(" Student ID not found!")

def update_student(self):
    print("\n--- Update Student Information ---")
    student_id = int(input("Enter student ID: "))

    if student_id not in self.students:
        print(" Student not found!")
        return

    print("1. Update Mobile Number")
    print("2. Update Class")
```

```
    choice = input("Enter choice: ")

    if choice == '1':
        new_mobile = input("Enter new mobile number: ")
        if new_mobile.isdigit() and len(new_mobile) == 10:
            self.students[student_id]['mobile'] = new_mobile
            print(" Mobile number updated successfully!")
        else:
            print(" Invalid mobile number!")
    elif choice == '2':
        new_class = int(input("Enter new class (1–12): "))
        if 1 <= new_class <= 12:
            self.students[student_id]['class'] = new_class
            print(" Class updated successfully!")
        else:
            print(" Invalid class!")
    else:
        print(" Invalid choice!")

def remove_student(self):
    print("\n--- Remove Student Record ---")
    student_id = int(input("Enter student ID to remove: "))
```

```
        if student_id in self.students:
            del self.students[student_id]
            print(" Student record deleted successfully.")
        else:
            print(" Student ID not found!")

    def show_menu(self):
        while True:
            print("\n===== SCHOOL MANAGEMENT SYSTEM =====")
            print("1. New Admission")
            print("2. View Student Details")
            print("3. Update Student Info")
            print("4. Remove Student Record")
            print("5. Exit")
            choice = input("Enter your choice: ")

            if choice == '1':
                self.new_admission()
            elif choice == '2':
                self.view_student()
```

```
            elif choice == '3':
                self.update_student()
            elif choice == '4':
                self.remove_student()
            elif choice == '5':
                print(" Exiting system. Goodbye!")
                break
            else:
                print(" Invalid choice. Try again.")


# Run the system
if __name__ == "__main__":
    sm = SchoolManagement()
    sm.show_menu()
```

Transport Reservation System (Bus Ticketing)

Design a Python class BusReservation that simulates a basic bus ticket booking system. Features should include:

➔ Show Available Routes:

- Predefined city routes with fixed prices.
- Example: "Mumbai to Pune - ₹500", "Delhi to Jaipur - ₹600", etc.

➔ Book Ticket:

- Enter passenger name, age, mobile, and route.
- Assign seat number (max 40 per bus per route).
- Generate a unique ticket ID.

➔ View Ticket:

- Lookup using ticket ID.

➔ Cancel Ticket:

- Cancel the ticket if it exists.

➔ Exit

```
class BusReservation:

    def __init__(self):
        self.routes = {
            "Mumbai to Pune": 500,
            "Delhi to Jaipur": 600,
            "Bangalore to Chennai": 550,
            "Hyderabad to Goa": 700
        }

        self.bookings = {}
        self.seats = {route: [] for route in self.routes}
        self.ticket_counter = 1

    def show_routes(self):
        print("\nAvailable Routes:")
        for route, price in self.routes.items():
            print(route, "- ₹", price)

    def book_ticket(self):
        name = input("Enter passenger name: ")
        age = input("Enter age: ")
```

```
mobile = input("Enter mobile number: ")

self.show_routes()
route = input("Enter route: ")

if route not in self.routes:
    print("Invalid route selected!")
    return

if len(self.seats[route]) >= 40:
    print("Sorry! Bus is fully booked for this route.")
    return

seat_no = len(self.seats[route]) + 1
self.seats[route].append(seat_no)

ticket_id = "T" + str(self.ticket_counter)
self.ticket_counter += 1

self.bookings[ticket_id] = {
    "name": name,
    "age": age,
```

```
            "mobile": mobile,
            "route": route,
            "seat": seat_no
        }

        print("\nTicket Booked Successfully!")
        print("Your Ticket ID:", ticket_id)
        print("Seat Number:", seat_no)

    def view_ticket(self):
        ticket_id = input("Enter Ticket ID: ")

        if ticket_id in self.bookings:
            ticket = self.bookings[ticket_id]
            print("\nTicket Details:")
            print("Name:", ticket["name"])
            print("Route:", ticket["route"])
            print("Seat No:", ticket["seat"])
        else:
            print("Ticket not found!")

    def cancel_ticket(self):
```

```
        ticket_id = input("Enter Ticket ID to cancel: ")

        if ticket_id in self.bookings:
            route = self.bookings[ticket_id]["route"]
            seat = self.bookings[ticket_id]["seat"]

            self.seats[route].remove(seat)
            del self.bookings[ticket_id]

            print("Ticket cancelled successfully!")
        else:
            print("Invalid Ticket ID!")
system = BusReservation()

while True:
    print("\n--- Bus Reservation System ---")
    print("1. Show Routes")
    print("2. Book Ticket")
    print("3. View Ticket")
    print("4. Cancel Ticket")
    print("5. Exit")
```

```
choice = input("Enter choice: ")


if choice == "1":
    system.show_routes()
elif choice == "2":
    system.book_ticket()
elif choice == "3":
    system.view_ticket()
elif choice == "4":
    system.cancel_ticket()
elif choice == "5":
    break
else:
    print("Invalid choice!")
```