HEALTHCARE INDUSTIES

In [ ]:

```
class ClinicAppointment:
    def __init__(self):
        # Dictionary to store appointments: {mobile_number: appointment_details}
        self.appointments = {}
        # List of available slots
        self.time_slots = ["10am", "11am", "12pm", "2pm", "3pm"]
        # Doctors list
        self.doctors = ["Dr. Smith", "Dr. Adams", "Dr. Lee"]
        # Counter to track slot occupancy: {(doctor, slot): count}
        self.slot_occupancy = {}

    def book_appointment(self):
        print("\n--- Book Appointment ---")
        name = input("Enter Patient Name: ")
        age = input("Enter Patient Age: ")
        mobile = input("Enter Mobile Number: ")

        # Check if patient already has an appointment
        if mobile in self.appointments:
            print("Error: This mobile number already has a scheduled appointment
            return

        print(f"Available Doctors: {', '.join(self.doctors)}")
        doctor = input("Enter Preferred Doctor: ")

        if doctor not in self.doctors:
            print("Invalid Doctor name.")
            return

        print(f"Available Time Slots: {', '.join(self.time_slots)}")
        slot = input("Enter Preferred Time Slot: ")

        if slot not in self.time_slots:
            print("Invalid Time Slot.")
            return

        # Check Doctor Availability (Max 3 per slot)
        current_bookings = self.slot_occupancy.get((doctor, slot), 0)
        if current_bookings < 3:
            # Confirm Booking
            self.appointments[mobile] = {
                "name": name,
                "age": age,
                "doctor": doctor,
                "slot": slot
            }
            # Update occupancy
            self.slot_occupancy[(doctor, slot)] = current_bookings + 1
            print(f"\nSuccess! Appointment confirmed for {name} with {doctor} at
        else:
            print(f"\nSorry, {doctor} is fully booked at {slot}. Please choose a

    def view_appointment(self):
        mobile = input("\nEnter Mobile Number to view appointment: ")
        if mobile in self.appointments:
            appt = self.appointments[mobile]
```

```
            print("\n--- Appointment Details ---")
            print(f"Name: {appt['name']}")
            print(f"Doctor: {appt['doctor']}")
            print(f"Time: {appt['slot']}")
        else:
            print("No appointment found for this mobile number.")

    def cancel_appointment(self):
        mobile = input("\nEnter Mobile Number to cancel appointment: ")
        if mobile in self.appointments:
            appt = self.appointments[mobile]
            # Reduce occupancy count
            self.slot_occupancy[(appt['doctor'], appt['slot'])] -= 1
            # Remove appointment
            del self.appointments[mobile]
            print("Appointment cancelled successfully.")
        else:
            print("No appointment found for this mobile number.")

# Simple Menu Loop
def main():
    clinic = ClinicAppointment()
    while True:
        print("\n=== Healthcare Clinic System ===")
        print("1. Book Appointment")
        print("2. View Appointment")
        print("3. Cancel Appointment")
        print("4. Exit")
        choice = input("Select an option: ")

        if choice == '1':
            clinic.book_appointment()
        elif choice == '2':
            clinic.view_appointment()
        elif choice == '3':
            clinic.cancel_appointment()
        elif choice == '4':
            break
        else:
            print("Invalid choice, please try again.")

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

```
=== Healthcare Clinic System ===
1. Book Appointment
2. View Appointment
3. Cancel Appointment
4. Exit
Invalid choice, please try again.

=== Healthcare Clinic System ===
1. Book Appointment
2. View Appointment
3. Cancel Appointment
4. Exit
```

SCHOOL MANAGEMENT SYSTEM

In [ ]:

```python
class SchoolManagement:
    def __init__(self):
        # Dictionary to store student records: {student_id: student_data_dict}
        self.students = {}
        # Counter for automatic unique ID generation
        self.next_id = 1001

    def validate_mobile(self, mobile):
        """Checks if the mobile number is exactly 10 digits."""
        return mobile.isdigit() and len(mobile) == 10

    def new_admission(self):
        print("\n--- New Student Admission ---")
        name = input("Enter Student Name: ")

        try:
            age = int(input("Enter Student Age (5-18): "))
            if not (5 <= age <= 18):
                print("Error: Age must be between 5 and 18.")
                return

            grade = int(input("Enter Class (1-12): "))
            if not (1 <= grade <= 12):
                print("Error: Class must be between 1 and 12.")
                return
        except ValueError:
            print("Error: Age and Class must be numeric.")
            return

        mobile = input("Enter Guardian's Mobile Number (10 digits): ")
        if not self.validate_mobile(mobile):
            print("Error: Mobile number must be exactly 10 digits.")
            return

        # Assign unique ID and store data
        student_id = self.next_id
        self.students[student_id] = {
            "name": name,
            "age": age,
            "class": grade,
            "mobile": mobile
        }

        print(f"\nAdmission Successful! Student ID: {student_id}")
        self.next_id += 1

    def view_student(self):
        try:
            sid = int(input("\nEnter Student ID to search: "))
            if sid in self.students:
                s = self.students[sid]
                print(f"\n--- Student ID: {sid} ---")
                print(f"Name:   {s['name']}")
                print(f"Age:    {s['age']}")
                print(f"Class:  {s['class']}")
                print(f"Mobile: {s['mobile']}")
            else:
                print("Error: Student ID not found.")
        except ValueError:
            print("Error: ID must be numeric.")
```

```
    def update_student(self):
        try:
            sid = int(input("\nEnter Student ID to update: "))
            if sid in self.students:
                print("1. Update Mobile Number")
                print("2. Update Class")
                choice = input("Select option: ")

                if choice == '1':
                    new_mobile = input("Enter new 10-digit mobile: ")
                    if self.validate_mobile(new_mobile):
                        self.students[sid]['mobile'] = new_mobile
                        print("Mobile number updated.")
                    else:
                        print("Invalid mobile format.")
                elif choice == '2':
                    new_class = int(input("Enter new Class (1-12): "))
                    if 1 <= new_class <= 12:
                        self.students[sid]['class'] = new_class
                        print("Class updated.")
                    else:
                        print("Invalid class.")
            else:
                print("Error: Student ID not found.")
        except ValueError:
            print("Error: Invalid input.")

    def remove_student(self):
        try:
            sid = int(input("\nEnter Student ID to remove: "))
            if sid in self.students:
                del self.students[sid]
                print(f"Student record {sid} has been deleted.")
            else:
                print("Error: Student ID not found.")
        except ValueError:
            print("Error: ID must be numeric.")

# Main System Loop
def main():
    school = SchoolManagement()
```

Transport Reservation System (Bus Ticketing)

In [ ]:

```
import random

class BusReservation:
    def __init__(self):
        # Predefined routes with fixed prices
        self.routes = {
            "1": {"path": "Mumbai to Pune", "price": 500},
            "2": {"path": "Delhi to Jaipur", "price": 600},
            "3": {"path": "Bangalore to Chennai", "price": 750},
            "4": {"path": "Hyderabad to Vijayawada", "price": 450}
        }
        # Dictionary to store tickets: {ticket_id: ticket_details}
        self.tickets = {}
        # Track filled seats per route: {route_id: current_seat_count}
```

```
        self.route_occupancy = {route_id: 0 for route_id in self.routes}
        self.max_seats = 40

    def show_routes(self):
        print("\n--- Available Routes ---")
        print(f"{'ID':<5} {'Route':<25} {'Price':<10}")
        for r_id, info in self.routes.items():
            print(f"{r_id:<5} {info['path']:<25} ₹{info['price']:<10}")

    def book_ticket(self):
        self.show_routes()
        route_choice = input("\nEnter Route ID to book: ")

        if route_choice not in self.routes:
            print("Invalid Route ID.")
            return

        # Check seat availability
        if self.route_occupancy[route_choice] >= self.max_seats:
            print("Sorry, this bus is fully booked (40/40 seats filled).")
            return

        print("\n--- Passenger Details ---")
        name = input("Passenger Name: ")
        age = input("Age: ")
        mobile = input("Mobile Number: ")

        # Logic for Seat Allocation and ID Generation
        self.route_occupancy[route_choice] += 1
        seat_no = self.route_occupancy[route_choice]
        ticket_id = f"TIX{random.randint(1000, 9999)}{seat_no}"

        # Store Ticket
        self.tickets[ticket_id] = {
            "name": name,
            "age": age,
            "mobile": mobile,
            "route": self.routes[route_choice]["path"],
            "price": self.routes[route_choice]["price"],
            "seat": seat_no,
            "route_id": route_choice
        }

        print(f"\nBooking Successful! Your Ticket ID is: {ticket_id}")
        print(f"Assigned Seat Number: {seat_no}")

    def view_ticket(self):
        tid = input("\nEnter Ticket ID: ").upper()
        if tid in self.tickets:
            t = self.tickets[tid]
            print("\n--- Ticket Information ---")
            print(f"Ticket ID: {tid}")
            print(f"Passenger: {t['name']} ({t['age']} yrs)")
            print(f"Route:     {t['route']}")
            print(f"Seat No:   {t['seat']}")
            print(f"Amount:    ₹{t['price']}")
        else:
            print("Error: Ticket ID not found.")

    def cancel_ticket(self):
```

```
        tid = input("\nEnter Ticket ID to cancel: ").upper()
        if tid in self.tickets:
            # Get route ID from ticket to free up the seat
            route_id = self.tickets[tid]["route_id"]
            self.route_occupancy[route_id] -= 1

            del self.tickets[tid]
            print(f"Ticket {tid} has been cancelled successfully.")
        else:
            print("Error: Invalid Ticket ID.")

# System Menu
def main():
    system = BusReservation()
    while True:
        print("\n===== Bus Reservation System =====")
        print("1. Show Available Routes")
        print("2. Book Ticket")
        print("3. View Ticket")
        print("4. Cancel Ticket")
        print("5. Exit")

        choice = input("Select an option: ")

        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':
            print("Thank you for using our service!")
            break
        else:
            print("Invalid input, please try again.")

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

In [ ]: