import re

from datetime import datetime

class EventManagement:

    def \_\_init\_\_(self):

        self.clients = {}  # Initialize clients dictionary

        self.staff = {

            "security": [],

            "vendors": [],

            "suppliers": []

        }

        self.admin\_password = "admin"  # Simple password for admin login

    def client\_login(self):

        while True:

            print("\n1. Make a New Booking\n2. View Past Bookings\n3. Delete a Booking\n4. Exit")

            choice = input("Enter option (1/2/3/4): ")

            if choice == '1':

                self.make\_booking()

            elif choice == '2':

                self.view\_past\_bookings()

            elif choice == '3':

                self.delete\_client\_booking()

            elif choice == '4':

                print("Exiting client interface...")

                break

            else:

                print("Invalid option. Please choose 1, 2, 3, or 4.")

    def admin\_login(self):

        password = input("Enter admin password: ")

        if password == self.admin\_password:

            self.admin\_interface()

        else:

            print("Invalid password. Access denied.")

    def admin\_interface(self):

        while True:

            print("\nAdmin Interface")

            print("1. View All Bookings")

            print("2. Manage Venue Security")

            print("3. Manage Staff")

            print("4. Confirm Payments")

            print("5. View All Staff")

            print("6. Cancel Booking")

            print("7. Exit")

            choice = input("Enter option (1/2/3/4/5/6/7): ")

            if choice == '1':

                self.view\_all\_bookings()

            elif choice == '2':

                self.manage\_venue\_security()

            elif choice == '3':

                self.manage\_staff()

            elif choice == '4':

                self.confirm\_payments()

            elif choice == '5':

                self.view\_all\_staff()

            elif choice == '6':

                self.delete\_client\_booking()

            elif choice == '7':

                print("Exiting admin interface...")

                break

            else:

                print("Invalid option. Please choose 1, 2, 3, 4, 5, or 6.")

    def view\_all\_bookings(self):

        print("\nAll Bookings:")

    # Collect all bookings into a single list

        all\_bookings = []

        for cnic, bookings in self.clients.items():

            for booking in bookings:

                all\_bookings.append({

                    "cnic": cnic,

                    "event": booking['event'],

                    "date": booking['date'],

                    "venue": booking['venue'],

                    "total\_cost": booking['total\_cost']

                })

    # Sort bookings by date

        sorted\_bookings = sorted(all\_bookings, key=lambda x: x['date'])

    # Display the sorted bookings

        if sorted\_bookings:

            for booking in sorted\_bookings:

                print(f"Client CNIC: {booking['cnic']}, Event: {booking['event']}, Date: {booking['date'].strftime('%d/%m/%Y')}, Venue: {booking['venue']}, Total Cost: PKR {booking['total\_cost']}")

        else:

            print("No bookings found.")

    def delete\_booking(self):

        cnic = input("Enter the client's CNIC to delete a booking: ")

        if cnic in self.clients:

            print("Client's Bookings:")

            for index, booking in enumerate(self.clients[cnic]):

                print(f"{index + 1}. Event: {booking['event']}, Date: {booking['date'].strftime('%d/%m/%Y')}, Venue: {booking['venue']}, Total Cost: PKR {booking['total\_cost']}")

            booking\_index = int(input("Enter the booking number to delete: ")) - 1

            if 0 <= booking\_index < len(self.clients[cnic]):

                deleted\_booking = self.clients[cnic].pop(booking\_index)

                print(f"Booking for event '{deleted\_booking['event']}' on {deleted\_booking['date'].strftime('%d/%m/%Y')} deleted successfully.")

            else:

                print("Invalid booking number.")

        else:

            print("Client not found.")

    def view\_all\_staff(self):

        print("\nAll Staff:")

        # Collect all staff names along with their roles

        all\_staff = []

        for role, members in self.staff.items():

            for member in members:

                all\_staff.append((member, role))  # Store as a tuple (name, role)

    # Sort the staff names alphabetically

        sorted\_staff = sorted(all\_staff, key=lambda x: x[0])  # Sort by name

    # Display the sorted staff names

        if sorted\_staff:

            print("Sorted Staff List:")

            for name, role in sorted\_staff:

                print(f"{name} (Role: {role.capitalize()})")

        else:

            print("No staff hired.")

    def manage\_venue\_security(self):

        print("\nManage Venue Security:")

        action = input("Do you want to (1) Add Security Staff or (2) Remove Security Staff? (Enter 1 or 2): ")

        if action == '1':

            staff\_name = input("Enter the name of the security staff to add: ")

            self.staff["security"].append(staff\_name)

            print(f"Security staff '{staff\_name}' added successfully.")

        elif action == '2':

            staff\_name = input("Enter the name of the security staff to remove: ")

            if staff\_name in self.staff["security"]:

                self.staff["security"].remove(staff\_name)

                print(f"Security staff '{staff\_name}' removed successfully.")

            else:

                print(f"Security staff '{staff\_name}' not found.")

        else:

            print("Invalid option.")

    def manage\_staff(self):

        print("\nManage Staff:")

        action = input("Do you want to (1) Add Staff, (2) Remove Staff, or (3) View All Staff? (Enter 1, 2, or 3): ")

        if action == '1':

            staff\_type = input("Enter staff type (security/vendors/suppliers): ").lower()

            staff\_name = input("Enter the name of the staff to add: ")

            if staff\_type in self.staff:

                self.staff[staff\_type].append(staff\_name)

                print(f"{staff\_type.capitalize()} '{staff\_name}' added successfully.")

            else:

                print("Invalid staff type.")

        elif action == '2':

            staff\_type = input("Enter staff type (security/vendors/suppliers): ").lower()

            staff\_name = input("Enter the name of the staff to remove: ")

            if staff\_type in self.staff and staff\_name in self.staff[staff\_type]:

                self.staff[staff\_type].remove(staff\_name)

                print(f"{staff\_type.capitalize()} '{staff\_name}' removed successfully.")

            else:

                print(f"{staff\_type.capitalize()} '{staff\_name}' not found.")

        elif action == '3':

         # Collect all staff names

            all\_staff = []

            for role, members in self.staff.items():

                all\_staff.extend(members)  # Add all members to the list

            # Sort the staff names

            sorted\_staff = sorted(all\_staff)

            # Display the sorted staff names

            print("\nSorted List of All Staff:")

            for name in sorted\_staff:

                print(name)

        else:

            print("Invalid option.")

    def confirm\_payments(self):

        print("\nConfirm Payments:")

        cnic = input("Enter the CNIC of the client to confirm payment: ")

        if cnic in self.clients:

            for booking in self.clients[cnic]:

                print(f"Event: {booking['event']}, Date: {booking['date'].strftime('%d/%m/%Y')}, Venue: {booking['venue']}, Total Cost: PKR {booking['total\_cost']}")

            confirm = input("Do you want to confirm payment for all bookings? (yes/no): ").lower()

            if confirm == 'yes':

                print(f"Payments confirmed for client with CNIC: {cnic}.")

            else:

                print("Payment confirmation canceled.")

        else:

            print(f"No bookings found for CNIC {cnic}.")

    def make\_booking(self):

        print("Making a New Booking...")

        cnic = input("Enter your CNIC: ")

        name = input("Enter your name: ")

        contact = input("Enter your Contact Number: ")

        email = input("Enter your Email ID: ")

        if cnic not in self.clients:

            self.clients[cnic] = []

        date = self.get\_valid\_date("Enter date of event (DD/MM/YYYY): ")

        event\_name = input("Choose the type of event \n-----------------------\nCorporate Event \nWedding Reception \nAnniversary Party \nGraduation Party \nAward Ceremony \n-----------------------\nChoose the type of Event: ")

        # Venue selection

        venues = {

            "The Pearl Palace": 500000,

            "The Clifton Marquee": 650000,

            "Courtyard Venues": 450000,

            "The Garden Oasis": 550000,

            "The Skyline Banquet Hall": 700000,

            "The Skyloft Banquet": 650000,

            "The Mansion Marquee": 750000

        }

        print("\nAvailable Venues:")

        for idx, (venue, price) in enumerate(venues.items(), 1):

            print(f"{idx}. {venue} - PKR {price}")

        selected\_venue\_idx = int(input("Select a venue by number: ")) - 1

        selected\_venue = list(venues.keys())[selected\_venue\_idx]

        venue\_price = venues[selected\_venue]

        # Time of day selection

        time\_of\_day = input("Is the event during the day or night? (day/night): ").lower().strip()

        if time\_of\_day == "night":

            venue\_price \*= 1.2  # Increase price for night events

        print(f"Selected venue: {selected\_venue} with cost PKR {venue\_price}")

        seats = self.get\_valid\_seats()

        catering\_choice = input("Choose catering option (internal/external): ").lower().strip()

        total\_food\_cost = 0

        theme\_price = 0

        total\_technical\_cost = 0

        ticket\_quantity = 0

        ticket\_price = 100

        agency\_price = 0

        if catering\_choice == "internal":

            print("Catering: Internal selected.")

            food\_options = [

                ("Pasta", 1000), ("Salad", 500), ("Soup", 700), ("Rice", 400), ("Cake", 800),

                ("Grilled Chicken", 1200), ("Vegetable Stir-fry", 1000), ("Beef Wellington", 2000),

                ("Mushroom Risotto", 1500), ("Lasagna", 1300), ("Roast Beef", 1800), ("Prawns", 2200),

                ("Chicken Wings", 900), ("Vegetable Platter", 600), ("Fried Rice", 600), ("Mashed Potatoes", 500),

                ("Fruit Salad", 700), ("Cheese Platter", 1500), ("Spring Rolls", 800), ("Tacos", 600),

                ("Sandwiches", 500), ("Pizza", 1000), ("Burgers", 700), ("Fries", 400), ("Moussaka", 1400),

                ("Paella", 1800), ("Quiche", 900), ("Chili", 1100), ("BBQ Ribs", 2000), ("Curry Chicken", 1200),

                ("Dumplings", 800), ("Grilled Fish", 1800 ), ("Mac and Cheese", 900), ("Steak", 2500),

                ("Fried Chicken", 1200), ("Steamed Vegetables", 500), ("Vegetarian Tacos", 700),

                ("Crispy Chicken Tenders", 900), ("Vegetable Samosas", 600), ("Stuffed Mushrooms", 1000),

                ("Pork Ribs", 2000), ("Pasta Salad", 800), ("Chicken Caesar Salad", 1000), ("Garlic Bread", 400),

                ("Brownie", 600), ("Cupcakes", 500), ("Fruit Tart", 800), ("Chocolate Mousse", 700),

                ("Ice Cream", 400), ("Fruit Punch", 600), ("Lemonade", 400)

            ]

            print("\nChoose your food options:")

            for idx, (food, price) in enumerate(food\_options, 1):

                print(f"{idx}. {food} - PKR {price}")

            selected\_foods = input("Enter the numbers of the foods you want, separated by commas: ").split(',')

            food\_choices = []

            for idx in selected\_foods:

                try:

                    food\_idx = int(idx.strip()) - 1

                    food, price = food\_options[food\_idx]

                    food\_choices.append(food)

                    total\_food\_cost += price

                except (ValueError, IndexError):

                    print(f"Invalid choice: {idx.strip()}. Please enter valid numbers.")

            print(f"\nSelected foods: {', '.join(food\_choices)}")

            print(f"Total food cost: PKR {total\_food\_cost}")

            decoration\_themes = [

                ("Blue and Silver Theme", 10000), ("Gold and White Theme", 15000), ("Red and Black Theme", 12000),

                ("Black and Gold Theme", 18000), ("White and Green Theme", 13000), ("Pink and White Theme", 11000),

                ("Golden and Red Theme", 14000), ("Elegant Black Theme", 16000), ("Bohemian Style", 17000),

                ("Modern Chic Theme", 19000), ("Rustic Charm", 15000), ("Vintage Romance", 16000),

                ("Floral Elegance", 18000), ("Bright Pastels", 12000), ("Neon Lights Theme", 20000)

            ]

            print("\nChoose decoration theme:")

            for idx, (theme, price) in enumerate(decoration\_themes, 1):

                print(f"{idx}. {theme} - PKR {price}")

            selected\_theme\_idx = input("Enter the number of the decoration theme you want: ")

            try:

                theme\_choice = decoration\_themes[int(selected\_theme\_idx) - 1]

                theme\_name, theme\_price = theme\_choice

                print(f"Selected decoration theme: {theme\_name}")

                print(f"Decoration cost: PKR {theme\_price}")

            except (ValueError, IndexError):

                print(f"Invalid choice: {selected\_theme\_idx}. Please select a valid number.")

        elif catering\_choice == "external":

            print("Catering: External selected.")

            # External catering logic can be added if needed.

        else:

            print("Invalid catering choice. Please select either 'internal' or 'external'.")

        technical\_equipment = input("Do you need technical equipment? (yes/no): ").lower().strip()

        if technical\_equipment == "yes":

            equipment\_options = [

                ("Microphone", 1000), ("Speaker", 1500), ("Projector", 2500), ("Lighting", 2000)

            ]

            print("\nAvailable Technical Equipment:")

            for idx, (equipment, price) in enumerate(equipment\_options, 1):

                print(f"{idx}. {equipment} - PKR {price}")

            selected\_equipment = input("Enter the numbers of the equipment you need, separated by commas: ").split(',')

            for idx in selected\_equipment:

                try:

                    equipment\_idx = int(idx.strip()) - 1

                    equipment, price = equipment\_options[equipment\_idx]

                    total\_technical\_cost += price

                except (ValueError, IndexError):

                    print(f"Invalid choice: {idx.strip()}. Please enter valid numbers.")

        parking\_tickets = input("Do you need parking tickets? (yes/no): ").lower().strip()

        if parking\_tickets == "yes":

            ticket\_quantity = int(input(f"How many tickets do you need (Price per ticket: PKR {ticket\_price})? "))

            print(f"Total parking ticket cost: PKR {ticket\_quantity \* ticket\_price}")

        photography = input("Do you want photography services? (yes/no): ").lower().strip()

        if photography == "yes":

            photography\_agencies = [

                ("Blue Light Studios", 40000), ("The Canva Studio", 50000), ("Jimmy's Studio", 60000), ("Star Studio", 75000)

            ]

            print("\nPhotography Agencies and Prices:")

            for idx, (agency, price) in enumerate(photography\_agencies, 1):

                print(f"{idx}. {agency} - PKR {price}")

            selected\_photography = input("Enter the number of the agency you want: ")

            try:

                agency\_choice = photography\_agencies[int(selected\_photography) - 1]

                agency\_name, agency\_price = agency\_choice

                print(f"Selected photography agency: {agency\_name}")

                print(f"Photography cost: PKR {agency\_price}")

            except (ValueError, IndexError):

                print("Invalid choice. Please select a valid number.")

        total\_cost = total\_food\_cost + theme\_price + total\_technical\_cost + venue\_price + (ticket\_quantity \* ticket\_price) + agency\_price

        print(f"Total cost of your booking: PKR {total\_cost}")

        booking = {

            "name": name,

            "cnic": cnic,

            "date": datetime.strptime(date, "%d/%m/%Y"),  # Convert date to datetime object

            "event": event\_name,

            "seats": seats,

            "venue": selected\_venue,

            "food": food\_choices,

            "decoration": theme\_name if 'theme\_name' in locals() else None,

            "total\_cost": total\_cost

        }

        self.clients[cnic].append(booking)

        print(f"Booking confirmed for {name} on {date} at {selected\_venue}.")

        # Payment method

        payment\_method = input("Choose payment method (cash/bank): ").lower().strip()

        if payment\_method == "cash":

            print("You have chosen to pay by cash.")

        elif payment\_method == "bank":

            bank\_name = input("Enter your bank name: ")

            account\_number = input("Enter your account number: ")

            print(f"You have chosen to pay by bank transfer to {bank\_name}, Account Number: {account\_number}.")

        else:

            print("Invalid payment method selected. Please choose either 'cash' or 'bank'.")

        # Print receipt

        print("\n--- Receipt ---")

        print(f"Name: {name}")

        print(f"CNIC: {cnic}")

        print(f"Contact: {contact}")

        print(f"Email: {email}")

        print(f"Event: {event\_name}")

        print(f"Date: {date}")

        print(f"Venue: {selected\_venue}")

        print(f"Seats: {seats}")

        print(f"Food Choices: {', '.join(food\_choices)}")

        print(f"Decoration Theme: {theme\_name if 'theme\_name' in locals() else 'None'}")

        print(f"Total Cost: PKR {total\_cost}")

        print(f"Payment Method: {'Cash' if payment\_method == 'cash' else 'Bank Transfer'}")

        if payment\_method == "bank":

            print(f"Bank Name: {bank\_name}")

            print(f"Account Number: {account\_number}")

        print("-------------------")

    def get\_valid\_date(self, prompt):

        while True:

            date = input(prompt)

            if self.validate\_date(date):

                return date

            else:

                print("Invalid date format. Please use DD/MM/YYYY.")

    def validate\_date(self, date\_str):

        date\_regex = r"^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/\d{4}$"

        return bool(re.match(date\_regex, date\_str))

    def get\_valid\_seats(self):

        while True:

            try:

                seats = int(input("Enter number of seats: "))

                if 0 < seats <= 1000:

                    return seats

                else:

                    print("Seats not available. Please enter a valid number (max 1000).")

            except ValueError:

                print("Please enter a valid number.")

    def view\_past\_bookings(self):

        cnic = input("Enter your CNIC to view past bookings: ")

        if cnic in self.clients and self.clients[cnic]:

            print(f"Your Past Bookings for CNIC {cnic}:")

        # Sort bookings by venue name and then by date

            sorted\_bookings = sorted(self.clients[cnic], key=lambda x: (x['venue'], x['date']))

            for booking in sorted\_bookings:

                print(f"Event: {booking['event']} on {booking['date'].strftime('%d/%m/%Y')} at {booking['venue']} with {booking['seats']} seats, Total Cost: PKR {booking['total\_cost']}.")

        else:

            print(f"No bookings found for CNIC {cnic}.")

    def delete\_client\_booking (self):

        cnic = input("Enter your CNIC to delete a booking: ")

        if cnic in self.clients and self.clients[cnic]:

            print("Your past bookings:")

            for idx, booking in enumerate(self.clients[cnic]):

                print(f"{idx + 1}. Event: {booking['event']} on {booking['date'].strftime('%d/%m/%Y')} at {booking['venue']} with {booking['seats']} seats, Total Cost: PKR {booking['total\_cost']}.")

            try:

                choice = int(input("Enter the number of the booking you want to delete: ")) - 1

                if 0 <= choice < len(self.clients[cnic]):

                    del self.clients[cnic][choice]

                    print("Booking deleted successfully.")

                else:

                    print("Invalid selection.")

            except ValueError:

                print("Invalid input. Please enter a valid number.")

        else:

            print(f"No bookings found for CNIC {cnic}.")

    def main(self):

        while True:

            print("\nWelcome to Event Management System")

            print("1. Client Login")

            print("2. Admin Login")

            print("3. Exit")

            choice = input("Enter choice (1/2/3): ")

            if choice == '1':

                self.client\_login()

            elif choice == '2':

                self.admin\_login()

            elif choice == '3':

                print("Exiting system...")

                break

            else:

                print("Invalid option. Please choose 1, 2, or 3.")

# To run the program

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

    event\_management\_system = EventManagement()

    event\_management\_system.main()