import json

import random

import string

class User:

    def \_\_init\_\_(self, user\_id, username, password):

        self.\_user\_id = user\_id

        self.username = username

        self.password = password

class Passenger(User):

    def \_\_init\_\_(self, user\_id, username, password, payment\_method):

        super().\_\_init\_\_(user\_id, username, password)

        self.payment\_method = payment\_method

class Driver(User):

    def \_\_init\_\_(self, user\_id, username, password, license\_number=None, vehicle=None):

        super().\_\_init\_\_(user\_id, username, password)

        self.license\_number = license\_number or self.\_generate\_license()

        self.vehicle = vehicle or self.\_generate\_vehicle()

        self.earnings = {}

    def \_generate\_license(self):

        return ''.join(random.choices(string.ascii\_uppercase + string.digits, k=8))

    def \_generate\_vehicle(self):

        return Vehicle(''.join(random.choices(string.ascii\_uppercase + string.digits, k=6)), 'Bus', 20)

    def view\_earnings(self):

        if not self.earnings:

            print("\nNo trips or earnings available yet.")

            return

        print("\nEarnings Account of the Driver:")

        total\_earnings = sum(self.earnings.values())

        print(f"Total Earnings: ${total\_earnings:.2f}")

        while True:

            trip\_id\_input = input("Enter Trip ID to view earnings (or type 'back' to return to menu): ").strip()

            if trip\_id\_input.lower() == 'back':

                break

            try:

                trip\_id = int(trip\_id\_input)

                trip = next((t for t in Trip.trips if t.\_trip\_id == trip\_id and t.driver == self), None)

                if trip:

                    print(f"\nTrip ID: {trip.\_trip\_id}")

                    print(f"Route: {trip.origin} to {trip.destination}")

                    earnings = self.earnings.get(trip\_id, 0)

                    print(f"Earnings from this trip: ${earnings:.2f}")

                    if trip.cancellation\_history:

                        print("\nCancellation History:")

                        for cancellation in trip.cancellation\_history:

                            print(f"  - {cancellation['passenger'].username} canceled {cancellation['quantity']} tickets. "

                                  f"Deduction: ${cancellation['deduction']:.2f}")

                    else:

                        print("No cancellations for this trip.")

                    if trip.passenger\_bookings:

                        print("\nCurrent Passenger Bookings:")

                        for passenger\_id, quantity in trip.passenger\_bookings.items():

                            passenger = next((p for p in LoginSystem.users if p.\_user\_id == passenger\_id), None)

                            if passenger:

                                print(f"  - {passenger.username}: {quantity} ticket(s)")

                    else:

                        print("No passenger bookings for this trip.")

                else:

                    print("Trip not found or not assigned to you.")

            except ValueError:

                print("Invalid Trip ID. Please enter a valid number or 'back'.")

class Vehicle:

    def \_\_init\_\_(self, vehicle\_number, vehicle\_type, vehicle\_capacity):

        self.vehicle\_number = vehicle\_number

        self.vehicle\_type = vehicle\_type

        self.vehicle\_capacity = vehicle\_capacity

class Trip:

    trips = []

    def \_\_init\_\_(self, trip\_id, origin, destination, driver, price\_per\_ticket, available\_tickets):

        self.\_trip\_id = trip\_id

        self.origin = origin

        self.destination = destination

        self.driver = driver

        self.price\_per\_ticket = price\_per\_ticket

        self.available\_tickets = available\_tickets

        self.passenger\_bookings = {}

        self.cancellation\_history = []

        Trip.trips.append(self)

    def trip\_summary(self):

        return f"Trip ID {self.\_trip\_id}: {self.origin} to {self.destination}, " \

               f"Driver: {self.driver.username}, Vehicle: {self.driver.vehicle.vehicle\_number}, Available Tickets: {self.available\_tickets}, " \

               f"Price: ${self.price\_per\_ticket:.2f}"

    def book\_ticket(self, passenger, quantity):

        if quantity <= self.available\_tickets:

            self.available\_tickets -= quantity

            total\_cost = quantity \* self.price\_per\_ticket

            self.passenger\_bookings[passenger.\_user\_id] = self.passenger\_bookings.get(passenger.\_user\_id, 0) + quantity

            driver\_share = total\_cost \* 0.2

            self.driver.earnings[self.\_trip\_id] = self.driver.earnings.get(self.\_trip\_id, 0) + driver\_share

            print(f"Passenger {passenger.username} booked {quantity} ticket(s) on Trip ID {self.\_trip\_id}.")

            print(f"Driver {self.driver.username}'s earnings updated: ${driver\_share:.2f} added.")

            return total\_cost

        else:

            print("Not enough tickets available.")

            return 0

    def cancel\_ticket(self, passenger, quantity):

        if passenger.\_user\_id in self.passenger\_bookings and self.passenger\_bookings[passenger.\_user\_id] >= quantity:

            # Update available tickets

            self.available\_tickets += quantity

            # Calculate refund and deduction amounts

            refund\_amount = quantity \* self.price\_per\_ticket

            driver\_share = refund\_amount \* 0.2  # Deduction from driver's earnings

            self.driver.earnings[self.\_trip\_id] -= driver\_share

            # Update passenger bookings

            self.passenger\_bookings[passenger.\_user\_id] -= quantity

            # Remove passenger from bookings if no tickets left

            if self.passenger\_bookings[passenger.\_user\_id] == 0:

                del self.passenger\_bookings[passenger.\_user\_id]

            # Log the cancellation in the history

            self.cancellation\_history.append({

                'passenger': passenger,

                'quantity': quantity,

                'deduction': driver\_share

            })

            # Display cancellation information

            print(f"Passenger {passenger.username} canceled {quantity} ticket(s) on Trip ID {self.\_trip\_id}.")

            print(f"Refund Amount: ${refund\_amount:.2f}")

            print(f"Driver {self.driver.username}'s earnings updated: ${driver\_share:.2f} deducted.")

            print(f"Remaining Earnings for Trip {self.\_trip\_id}: ${self.driver.earnings[self.\_trip\_id]:.2f}")

            return f"Refunded amount: ${refund\_amount:.2f}"

        else:

            return "Passenger is not booked on this trip or insufficient tickets to cancel."

class LoginSystem:

    def \_\_init\_\_(self):

        self.users = []

        self.last\_registered\_user = None

        self.\_load\_earnings()

    def add\_user(self, user):

        self.users.append(user)

    def login(self, username, password):

        for user in self.users:

            if user.username == username and user.password == password:

                return user

        return None

    def register(self):

        print("\nRegistration Process:")

        user\_type = input("Enter user type (driver/passenger): ").strip().lower()

        username = input("Create a username: ")

        password = input("Create a password: ")

        if user\_type == "driver":

            driver = Driver(len(self.users) + 1, username, password)

            self.add\_user(driver)

            self.last\_registered\_user = driver

            print(f"Driver {username} registered successfully.")

            print(f"Your License Number: {driver.license\_number}")

            print(f"Your Vehicle Number: {driver.vehicle.vehicle\_number}")

            return driver

        elif user\_type == "passenger":

            payment\_method = input("Enter your payment method (e.g., Credit Card, PayPal): ")

            passenger = Passenger(len(self.users) + 1, username, password, payment\_method)

            self.add\_user(passenger)

            self.last\_registered\_user = passenger

            print(f"Passenger {username} registered successfully.")

            return passenger

        else:

            print("Invalid user type. Please enter 'driver' or 'passenger'.")

            return None

    def \_save\_earnings(self):

        earnings\_data = {}

        for user in self.users:

            if isinstance(user, Driver):

                earnings\_data[user.username] = user.earnings

        with open('driver\_earnings.json', 'w') as f:

            json.dump(earnings\_data, f, indent=4)

    def \_load\_earnings(self):

        try:

            with open('driver\_earnings.json', 'r') as f:

                earnings\_data = json.load(f)

                for username, earnings in earnings\_data.items():

                    driver = next((u for u in self.users if isinstance(u, Driver) and u.username == username), None)

                    if driver:

                        driver.earnings = earnings

        except FileNotFoundError:

            pass

# Initialize LoginSystem

login\_system = LoginSystem()

# Main Program Loop

user = None

while True:

    try:

        if user:

            print(f"\nYou are logged in as {user.\_\_class\_\_.\_\_name\_\_} ({user.username})")

            if isinstance(user, Passenger):

                while True:

                    print("\nChoose an option:")

                    print("1. View available trips")

                    print("2. Book a ticket")

                    print("3. Cancel trip")

                    print("4. Logout")

                    choice = input("Enter your choice: ").strip()

                    if choice == "1":

                        if Trip.trips:

                            for trip in Trip.trips:

                                print(trip.trip\_summary())

                        else:

                            print("No trips available yet.")

                    elif choice == "2":

                        if Trip.trips:

                            trip\_id = int(input("Enter Trip ID: "))

                            quantity = int(input("Enter tickets: "))

                            trip = next((t for t in Trip.trips if t.\_trip\_id == trip\_id), None)

                            if trip:

                                trip.book\_ticket(user, quantity)

                        else:

                            print("No trips available yet.")

                    elif choice == "3":

                        if Trip.trips:

                            trip\_id = int(input("Enter Trip ID to cancel: "))

                            trip = next((t for t in Trip.trips if t.\_trip\_id == trip\_id), None)

                            if trip:

                                quantity = int(input("How many tickets to cancel? "))

                                print(trip.cancel\_ticket(user, quantity))

                        else:

                            print("No trips available yet.")

                    elif choice == "4":

                        user = None

                        break

            elif isinstance(user, Driver):

                while True:

                    print("\nChoose an option:")

                    print("1. Create a trip")

                    print("2. View all trips")

                    print("3. View Earnings Account of the Driver")

                    print("4. Logout")

                    choice = input("Enter your choice: ").strip()

                    if choice == "1":

                        origin = input("Enter trip origin: ")

                        destination = input("Enter trip destination: ")

                        price\_per\_ticket = float(input("Enter price per ticket: "))

                        available\_tickets = int(input("Enter number of tickets available: "))

                        trip = Trip(len(Trip.trips) + 1, origin, destination, user, price\_per\_ticket, available\_tickets)

                        print(f"Trip created successfully: {trip.trip\_summary()}")

                    elif choice == "2":

                        if Trip.trips:

                            for trip in Trip.trips:

                                print(trip.trip\_summary())

                        else:

                            print("No trips available yet.")

                    elif choice == "3":

                        user.view\_earnings()

                    elif choice == "4":

                        user = None

                        break

        else:

            print("\nWelcome to the Booking System")

            print("1. Login")

            print("2. Register")

            print("3. Exit")

            choice = input("Enter your choice: ").strip()

            if choice == "1":

                username = input("Enter your username: ")

                password = input("Enter your password: ")

                user = login\_system.login(username, password)

                if not user:

                    print("Invalid login credentials.")

            elif choice == "2":

                login\_system.register()

            elif choice == "3":

                login\_system.\_save\_earnings()

                print("Goodbye!")

                break

    except Exception as e:

        print(f"Error: {e}")