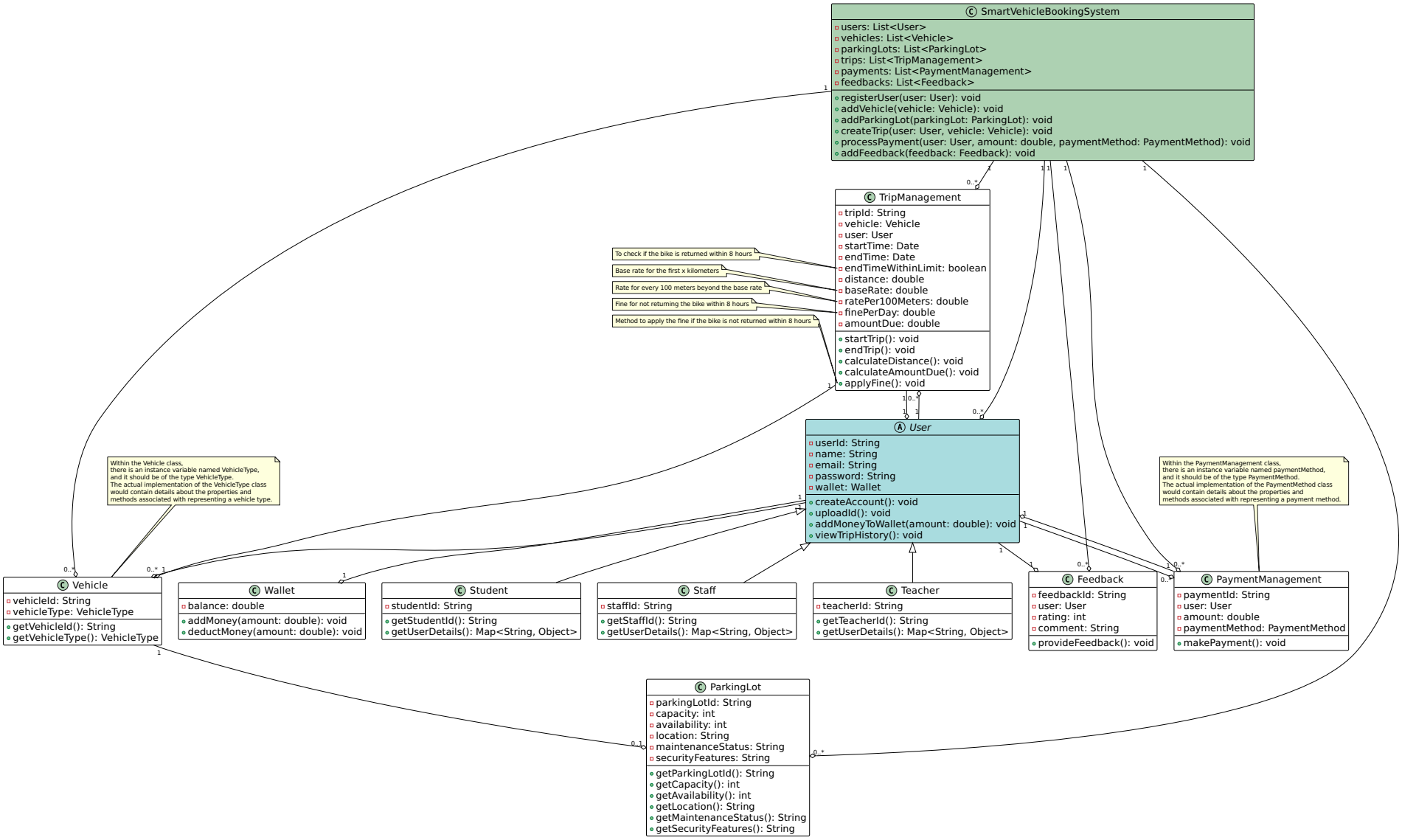
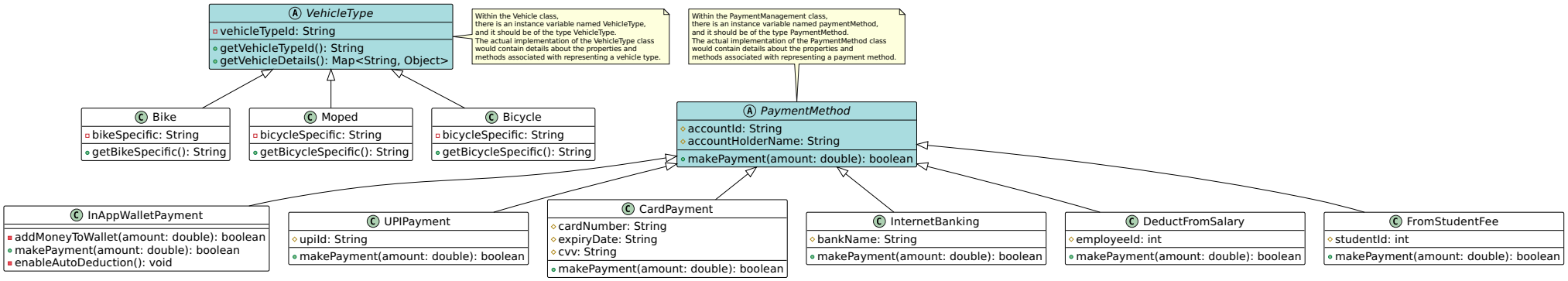
Vilal Ali
Roll Number: 2020900023

Smart Vehicle Booking System - Class Diagram

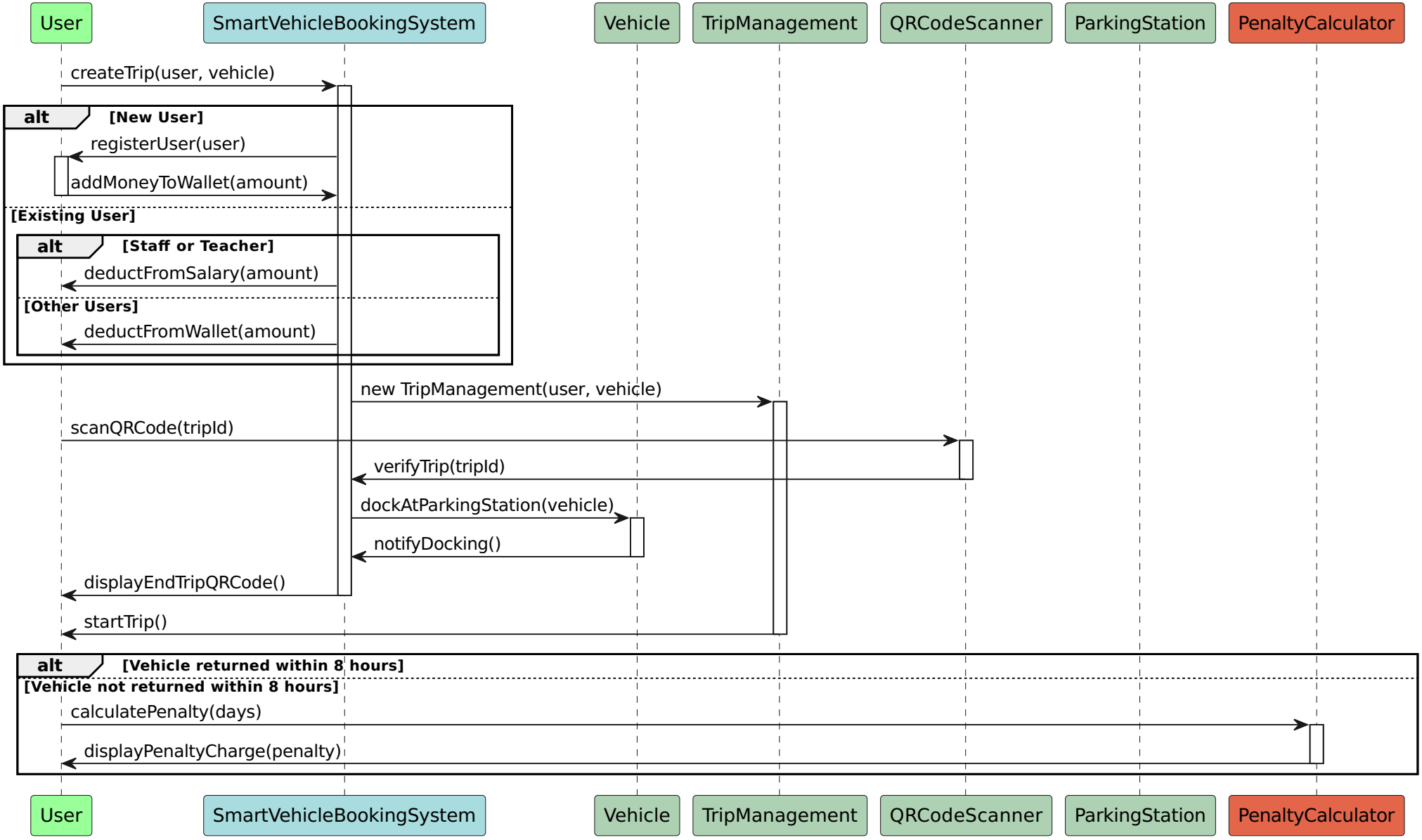


Designed By 2020900023



Designed By 2020900023

Sequence Diagram for Trip Process



Designed By 2020900023

Name: Vilal Ali

Roll Number: 2020900023

Course Name: SE (C6-401)

Smart Vehicle Booking System - Class Diagram Report

Introduction: The Smart Vehicle Booking System is designed to manage the booking and utilization of vehicles within an IIIT-H. The system facilitates user registration, vehicle addition, parking lot management, trip creation, payment processing, and feedback collection.

Functionality, Relationships and Descriptions of Each Class:

1. **User Class:**
    - **Attributes:** userId, name, email, password, wallet.
    - **Methods:** createAccount(), uploadId(), addMoneyToWallet(amount).
    - **Functionality:** Represents a generic user with attributes such as userId, name, email, password, and a linked wallet for financial transactions. Provides methods for account creation, ID upload, wallet management, and viewing trip history. ViewTripHistory()
    - **Assumption:** The system assumes that there is a secure authentication mechanism in place for user registration and login, which is not explicitly represented in the class diagram.
    - **Relationships:**
        - Associated with Wallet, TripManagement, PaymentManagement, and Vehicle.
        - Users interact with the system by creating trips, making payments, and managing their wallets.
2. **Student, Staff, Teacher (Inherited from User):**
    - **Attributes:** studentId, staffId, teacherId
    - **Methods:** getStudentId(), getStaffId(), getTeacherId(), getUserDetails()
    - **Functionality:** Differentiates users based on their roles (Student, Staff, Teacher). Each inherits the common attributes and methods from the User class and includes role-specific attributes and methods.

- **Assumption:** The getUserDetails() method in the Student, Staff, and Teacher classes assumes that it provides relevant details about the specific user role. The specific details returned by this method are not detailed in the class diagram.
- **Relationships:** Inheritance relationship with the User class.

3. **Wallet Class:**
    - **Functionality:** Manages the user's financial transactions with attributes like balance. Provides methods for adding and deducting money.
    - **Attributes:** balance
    - **Methods:** addMoney(amount), deductMoney(amount)
    - **Assumption:** The Wallet class assumes the presence of external systems or services for secure financial transactions. The specifics of wallet management, such as integration with banking systems, are not detailed in the class diagram.
    - **Relationships:** Associated with User for managing financial transactions.
4. **Vehicle Class:**
    - **Attributes:** vehicleId, vehicleType
    - **Methods:** getVehicleId(), getVehicleType()
    - **Functionality:** Represents a vehicle with attributes like vehicleId and vehicleType. Provides methods to retrieve vehicle information.
    - **Assumption:** The VehicleType class assumes that there are predefined types such as Bike, Moped, and Bicycle. Details about the specific attributes and characteristics of each vehicle type are not explicitly defined in the class diagram.
    - **Relationships:** Associated with ParkingLot (0..1) to represent the parking status of the vehicle.
5. **ParkingLot Class:**
    - **Attributes:** parkingLotId, capacity, availability, location, maintenanceStatus, securityFeatures.
    - **Methods:** getParkingLotId(), getCapacity(), getAvailability(), getLocation(), getMaintenanceStatus(), getSecurityFeatures()
    - **Functionality:** Represents a parking lot with attributes like parkingLotId, capacity, availability, location, maintenanceStatus, and securityFeatures. Provides methods to retrieve parking lot details.
    - **Assumption-1:** The maintenanceStatus attribute in the ParkingLot class assumes predefined statuses, such as "Under Maintenance" or "Operational." The specific statuses and the maintenance process are not detailed in the class diagram.

- **Assumption-2:** The securityFeatures attribute in the ParkingLot class assumes predefined features for security, such as surveillance cameras or access control systems. The specific security features and their implementation are not explicitly defined in the class diagram.

- **Relationships:**

    - Aggregated by SmartVehicleBookingSystem.

    - Associated with Vehicle (0..1) to represent the parking status of a vehicle.

6. **TripManagement Class:**

    - **Attributes:** tripId, vehicle, user, startTime, endTime, endTimeWithinLimit, distance, baseRate, ratePer100Meters, finePerDay, amountDue

    - **Methods:** startTrip(), endTrip(), calculateDistance(), calculateAmountDue(), applyFine()

    - **Functionality:**

        - Represents a class for managing and tracking information related to a trip, including the trip ID, vehicle, user, start time, end time, distance, base rate, rate per 100 meters, fine per day, and amount due.

        - The startTrip method initiates a new trip, recording the start time.

        - The endTrip method concludes the trip, recording the end time and calculating the distance traveled, the amount due, and applying any fines if necessary.

        - The calculateDistance method calculates the distance traveled during the trip based on the vehicle's movements.

        - The calculateAmountDue method calculates the total amount due for the trip, including the base rate, distance-based charges, and any fines.

        - The applyFine method checks if the trip duration exceeds the allowed limit (8 hours) and applies a fine if it does.

    - **Assumption:** The system assumes the availability of a reliable mechanism for calculating trip distance. The method calculateDistance() is left abstract in the class diagram to allow for different implementations.

    - **Relationships:**

        - Associated with User (1) and Vehicle (1) for trip creation.

        - Aggregated by SmartVehicleBookingSystem.

7. **PaymentManagement Class:**

    - **Attributes:** paymentId, user, amount, paymentMethod

    - **Methods:** makePayment()

- **Functionality:** Handles payment-related information with attributes like paymentId, user, amount, and paymentMethod. Provides a method to make payments.
- **Assumption:** The PaymentMethod class assumes secure handling of sensitive payment information. It is expected that encryption and secure protocols are implemented to protect user financial data.
- **Relationships:**
    - Associated with User (1) for payment processing.
    - Aggregated by SmartVehicleBookingSystem.

8. **Feedback Class:**
    - **Attributes:** feedbackId, user, rating, comment
    - **Methods:** provideFeedback()
    - **Functionality:** Captures user feedback with attributes like feedbackId, user, rating, and comment. Provides a method to submit feedback.
    - **Assumption:** The Feedback class assumes that the rating attribute is an integer within a predefined range (e.g., 1 to 5). The specific range and the interpretation of ratings are not explicitly defined in the class diagram.
    - **Relationships:**
        - Associated with User (1).
        - Aggregated by SmartVehicleBookingSystem.
9. **SmartVehicleBookingSystem Class:**
    - **Attributes:** users, vehicles, parkingLots, trips, payments, feedbacks
    - **Methods:** registerUser(user), addVehicle(vehicle), addParkingLot(parkingLot), createTrip(user, vehicle), processPayment(user, amount, paymentMethod), addFeedback(feedback)
      Functionality: Represents the overall system with lists of users, vehicles, parking lots, trips, payments, and feedbacks. Provides methods for user registration, adding vehicles and parking lots, creating trips, processing payments, and collecting feedback.
    - **Relationships:**
        - Aggregates User, Vehicle, ParkingLot, TripManagement, PaymentManagement, and Feedback.
        - Manages the overall flow and coordination of the system.
10. **PaymentMethod Class and Subclasses:**
    - **Functionality:** Represents different payment methods (InAppWallet, UPI, Card, etc.) with attributes like accountId, accountHolderName, expiryDate, etc. Provides a method to make payments.

- **Assumption:** The makePayment() method in the PaymentManagement class assumes successful payment processing. The actual implementation details, such as integration with payment gateways, are not specified in the class diagram.
- **Relationships:**
    - Used by PaymentManagement for handling various payment methods.

11. VehicleType Class and Subclasses:

- **Functionality:** Represents different types of vehicles (Bike, Moped, Bicycle) with attributes specific to each type. Provides methods to retrieve vehicle details.
- **Relationships:** Used by Vehicle for categorizing and retrieving vehicle information.

In summary, the classes in the Smart Vehicle Booking System collaborate to handle user interactions, manage financial transactions, facilitate trip bookings, and collect user feedback. The relationships and aggregations ensure a cohesive and organized system that caters to the various aspects of a vehicle booking platform.

PTO

Sequence Diagram for Trip Process – Sequence Diagram Report

1. **User Creation and Registration:**
    - **Descriptions:**
        - If the user is a new user, they need to register by providing necessary information.
        - Existing users may include staff or teachers who have a different payment deduction mechanism.
    - **Assumptions:**
        - The registration process includes providing essential user details.
        - Existing users may have different payment methods or sources.
2. **Adding Money to Wallet:**
    - **Descriptions:**
        - New users can add money to their wallet during the registration process.
        - Existing users, other than staff or teachers, can add money to their wallet.
    - **Assumptions:**
        - Users have a wallet associated with their account for trip payments.
3. **Deducting from Salary (Staff or Teacher):**
    - **Descriptions:**
        - This functionality is specific to staff or teacher users.
        - It provides an alternative payment method for certain user roles.
    - **Assumptions:**
        - Staff and teacher users have a salary associated with their account.
4. **Trip Management Initialization:**
    - **Descriptions:**
        - TripManagement is responsible for managing the lifecycle of a trip.
        - The trip is initialized with user and vehicle information.
    - **Assumptions:**
        - TripManagement handles the overall trip-related processes.
5. **QR Code Scanning and Verification:**
    - **Descriptions:**
        - Users use their smartphones or devices to scan the QR code generated for the trip.
        - QRCodeScanner communicates with the Smart Vehicle Booking System to verify the trip.
    - **Assumptions:**
        - QR codes are generated for each trip to facilitate user verification.

6. **Vehicle Docking and Notification:**
    - **Descriptions:**
        - Vehicle sends a notification to the system when it docks at a parking station.
    - **Assumptions:**
        - The system relies on notifications from the vehicle to track its status.
7. **Display End Trip QR Code:**
    - **Descriptions:**
        - Users receive a QR code on their devices when the trip is completed successfully.
    - **Assumptions:**
        - The QR code serves as a confirmation of the trip completion.
8. **Trip Start and End:**
    - **Descriptions:**
        - Users start the trip after successful verification, and it concludes when the vehicle is returned.
    - **Assumptions:**
        - The TripManagement system oversees the entire trip process.
9. **Penalty Calculation and Display:**
    - **Descriptions:**
        - If the vehicle is not returned within the specified time, a penalty is calculated and communicated to the user.
    - **Assumptions:**
        - Penalties are calculated based on the number of days the vehicle is overdue.