NexusRide

Features and Database Schema Design

1 Introduction

This document presents the feature set and a probable relational database schema for a **University Transport Management System** designed for staff members, drivers, and transport administrators. The system supports both subscription-based and token-based transportation, real-time seat availability, and administrative control through a Transport Officer (TO).

2 User Types and Features

2.1 Staff Members

All staff members can sign up and log in to the system. Based on their status, they are categorized as **Non-Subscribers** or **Subscribers**.

2.1.1 Non-Subscriber Dashboard

- View real-time seat availability for multiple vehicles and routes
- Purchase travel tokens
- View travel and payment history
- Cancel tokens with real-time seat availability updates

2.1.2 Subscriber Dashboard

- Apply for transport subscription (subject to approval)
- Take leave for one or multiple days, releasing reserved seats
- Change route for the current day
- Change pickup location for the current day
- Request guest pickup services for seminars or official events

2.2 Driver

Drivers log in to a dedicated dashboard.

- View assigned vehicle and route
- View passenger list (name, mobile number, pickup location)
- Start trip and trigger notifications to passengers

2.3 Transport Officer (TO)

The Transport Officer manages system operations and oversight.

- Approve or reject staff subscriptions
- Assign vehicles to routes
- Assign drivers to vehicles
- Add new vehicles
- Mark vehicles as under repair or unavailable
- Handle sudden vehicle unavailability
- Allocate vehicles for guest pickups
- View analytical reports on payments and trips

3 Database Schema Design

3.1 Users

```
users (
    user_id PK,
    name,
    email UNIQUE,
    password_hash,
    mobile_number,
    role ENUM('TO','DRIVER','STAFF'),
    is_active BOOLEAN,
    created_at
)
```

3.2 Staff Profiles

```
staff_profiles (
    staff_id PK,
    user_id FK -> users.user_id,
    staff_code UNIQUE,
    department,
    is_subscriber BOOLEAN,
    default_route_id FK -> routes.route_id,
    default_pickup_stop_id FK -> route_stops.stop_id
)
```

3.3 Driver Profiles

```
driver_profiles (
    driver_id PK,
    user_id FK -> users.user_id,
    license_number,
    assigned_vehicle_id FK -> vehicles.vehicle_id
)
```

3.4 Vehicles

```
vehicles (
    vehicle_id PK,
    vehicle_number UNIQUE,
    capacity,
    status ENUM(’AVAILABLE’,’IN_SERVICE’,’UNDER_REPAIR’),
    created_at
)
```

3.5 Routes and Stops

```
routes (
    route_id PK,
    route_name,
    start_point,
    end_point,
    is_active BOOLEAN
)
```

```
route_stops (
    stop_id PK,
    route_id FK -> routes.route_id,
    stop_name,
    sequence_number
)
```

3.6 Vehicle–Route Assignment

```
vehicle_routes (
    vehicle_route_id PK,
    vehicle_id FK -> vehicles.vehicle_id,
    route_id FK -> routes.route_id,
    assigned_date
)
```

3.7 Subscriptions and Leaves

```
subscriptions (
    subscription_id PK,
    staff_id FK -> staff_profiles.staff_id,
    route_id FK -> routes.route_id,
    start_date,
    end_date,
    approved_by FK -> users.user_id,
    status ENUM(’PENDING’,’APPROVED’,’REJECTED’)
)
```

```
subscription_leaves (
    leave_id PK,
    subscription_id FK -> subscriptions.subscription_id,
    from_date,
    to_date,
    reason
)
```

3.8 Tokens

```
tokens (
    token_id PK,
    staff_id FK -> staff_profiles.staff_id,
    route_id FK -> routes.route_id,
    travel_date,
    pickup_stop_id FK -> route_stops.stop_id,
    status ENUM('ACTIVE','CANCELLED','USED'),
    created_at
)
```

3.9 Trips and Seat Allocation

```
trips (
    trip_id PK,
    vehicle_id FK -> vehicles.vehicle_id,
    driver_id FK -> driver_profiles.driver_id,
    route_id FK -> routes.route_id,
    trip_date,
    start_time,
    status ENUM('SCHEDULED','STARTED','COMPLETED')
)
```

```
seat_allocations (
    allocation_id PK,
    trip_id FK -> trips.trip_id,
    staff_id FK -> staff_profiles.staff_id,
    seat_type ENUM('SUBSCRIPTION','TOKEN','GUEST'),
    pickup_stop_id FK -> route_stops.stop_id
)
```

3.10 Payments

```
payments (
    payment_id PK,
    user_id FK -> users.user_id,
    amount,
    payment_type ENUM('SUBSCRIPTION','TOKEN'),
    status ENUM('SUCCESS','FAILED'),
    transaction_time
)
```

3.11 Vehicle Maintenance

```
vehicle_maintenance (
    maintenance_id PK,
    vehicle_id FK -> vehicles.vehicle_id,
    reported_by FK -> users.user_id,
    reason,
    start_date,
    end_date
)
```

3.12 Guest Requests

```
guest_requests (
    guest_request_id PK,
    staff_id FK -> staff_profiles.staff_id,
    event_name,
    route_id FK -> routes.route_id,
    pickup_stop_id FK -> route_stops.stop_id,
    event_date,
    status ENUM(’PENDING’,’APPROVED’,’REJECTED’)
)
```

3.13 Notifications

```
notifications (
    notification_id PK,
    user_id FK -> users.user_id,
    message,
    is_read BOOLEAN,
    created_at
)
```

4 Entity Relationships Summary

- One **user** can be a staff member or a driver (one-to-one relationship).
- One **route** has multiple pickup stops (one-to-many).
- Vehicles and routes have a many-to-many relationship over time.

- A staff member can have multiple tokens and subscriptions.
- Each trip contains multiple seat allocations.
- Transport Officer actions are recorded through approval and maintenance records.

5 Conclusion

The proposed database schema efficiently supports real-time seat management, subscription and token-based transport, driver operations, and administrative control. The design is normalized, scalable, and suitable for academic and real-world implementation.