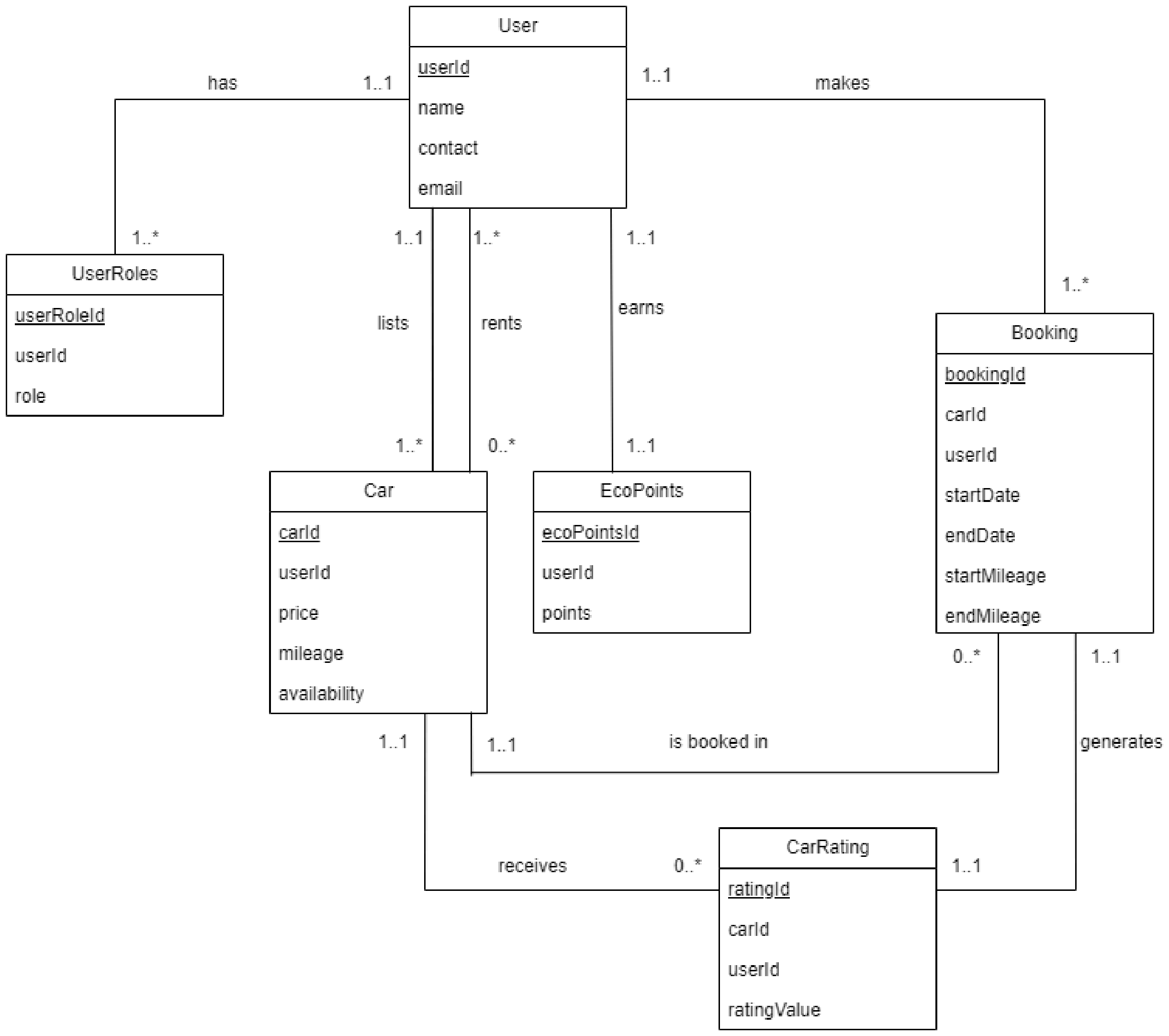
EcoRide: A Car Rental with Environmental Rewards System (Stage2)

1. Introduction

EcoRide is an environmentally conscious car rental platform that aims to reward eco-friendly driving behavior through a unique reward system. This report outlines the UML diagram of the EcoRide database and provides assumptions for each entity, relationship, and the rationale for normalization. We also present the database schema following BCNF/3NF principles, discuss relationship cardinalities, and justify the design choices.

2. UML Diagram



3. Entities

1. **User:**
   This entity represents each account registered in the system. This entity includes general user details that apply to both buyers and sellers, like name and contact information.

   Assumptions:

   - **Unique Identification:** Each user is uniquely identified by a userId.

   - **Roles Management:** Users can have one or multiple roles (e.g., buyer, seller), managed through the UserRoles table.

   - **Contact Information Validity:** Users must provide valid and up-to-date contact information for communication.

2. **UserRoles :**
   This entity separates roles from the User entity, allowing each user to have one or multiple roles (buyer, seller)

   Assumptions:

   - **Role Flexibility:** Each user can have one or multiple roles (e.g., buyer, seller) to support varied functionalities within the platform.

   - **Role Management:** New roles can be added in the future without modifying the existing UserRoles structure, promoting scalability.

3. <u>**Car :**</u>
   This entity represents each car listed on the platform. The userId serves as a foreign key that indicates the car's owner or renter. When a car is booked, the userId can also reference the buyer in the Booking entity.

   Assumptions:

   - Each car is owned by one seller, indicated by UserID when the car is listed.

   - Each car can have multiple bookings, where UserID will refer to the buyer in those records.

   - The availability attribute indicates whether a car can be rented.

   - Each car has a unique identifier (CarID).

4. <u>**Booking :**</u>
   This entity represents each rental transaction or reservation of a car by a user. It tracks details specific to the rental, such as which car was rented, the involved user and additional information necessary for eco-point calculation and historical record-keeping.

   Assumptions:

   - **Single User per Booking:** Each booking is associated with one buyer (user), meaning that joint bookings or shared rentals are not supported.

   - **Single Car per Booking:** Each booking is associated with one specific car, so multiple cars cannot be booked under a single transaction.

- **Availability Check:** The system ensures that a car is available for the specified dates before a booking can be confirmed, preventing scheduling conflicts.

- **Booking Cancellation:** Cancellations are allowed, but rules and fees may apply based on the booking timeline.

5. **<u>EcoPoints :</u>**
   The EcoPoints entity is designed to track the eco-friendly rewards that users accumulate based on their rental behavior and car ratings. This system encourages environmentally conscious actions by offering points based on specific criteria related to car usage.

   Assumptions:

   - **Unique Record per User:** Each user has a unique entry in the EcoPoints table, meaning that a user's eco-points will be accumulated in a single record.

   - **Dynamic Calculation:** The Points attribute will be adjusted based on metrics collected from related entities, such as fuel efficiency data from Booking and ratings from CarRating.

   - **Threshold for Rewards:** Users can use accumulated points once they reach a predefined threshold. Buyers can redeem points for free rentals, while sellers can use points for benefits like free car servicing.

   - **Non-Monetary Value:** EcoPoints are a reward metric rather than a currency, meaning they cannot be redeemed for cash but only for specific platform benefits.

6. **CarRating :**

   This entity stores information about the ratings given to cars by users (buyers) after each completed rental. This entity helps track the quality and reliability of each car, directly affecting the eco-points for sellers and indirectly influencing the choices of future renters.

   Assumptions:

   - **One Rating per Booking:** Each booking can have only one associated car rating, reflecting the buyer's feedback for that rental period.

   - **EcoPoint Influence:** The CarRating score impacts the eco-points accumulated by sellers. Higher ratings contribute positively to eco-points, incentivizing sellers to maintain high standards for their vehicles.

   - **Historical Tracking:** Each rating is linked to a specific booking and car, allowing the system to analyze the car's historical performance and identify consistently well-rated vehicles.

4. Relations

- **User and UserRoles :** A User has 1 or more UserRoles. A UserRole is associated with exactly 1 User (One-to-Many).

- **User and Car :** A User(seller) can list 1 or more Cars. A Car is listed by exactly 1 User(seller) (One-to-Many).
  A User(buyer) can rent zero or more Car whereas a Car can be A car can be rented multiple times by different users (Many-to-Many Relationship).

- **User and Booking :** A User can make 1 or more Bookings. A Booking is associated with exactly 1 User (One-to-Many Relationship).

- **Car and Booking :** A Car can be booked 0 or more times. A Booking involves exactly 1 Car (One-to-Many Relationship).

- **User and EcoPoints :** A User earns exactly 1 EcoPoints record. An EcoPoints record belongs to exactly 1 User (One-to-One Relationship).

- **Car and CarRating :** A Car can receive 0 or more CarRatings. A CarRating is for exactly 1 Car (One-to-Many Relationship).

- **Booking and CarRating :** A Booking can generate exactly 1 CarRating. A CarRating is generated by exactly 1 Booking. (One-to-One Relationship).

5. Normalization

The functional dependencies in our database schema are as follows:

- userId → name, contact, email
- carId → userId, price, mileage, availability
- ecoPointsId → userId, points, rewardStatus
- bookingId → carId, userId, startDate, endDate, startMileage, endMileage
- ratingId → carId, userId, ratingValue
- userRoleId → userId, role

A relation is in **Third Normal Form (3NF)** if at least one of the following conditions holds for every non-trivial functional dependency $X \to Y$:

- X is a superkey.
- Y is a prime attribute (i.e., each element of Y is part of some candidate key).

In our schema, all relations satisfy these conditions, ensuring that each table is in 3NF. For instance:

- In the Car table, the primary key carId determines all other attributes such as userId, price, and mileage, with no transitive dependencies.
- In the Booking table, bookingId determines attributes like carId, userId, startDate, endDate, startMileage, endMileage. There are no dependencies among the non-key attributes, so this table also conforms to 3NF.

A relation is in **Boyce-Codd Normal Form (BCNF)** if:

- The relation is already in 3NF.
- For every functional dependency $X \rightarrow Y$ in the relation, X should be a superkey.

In our schema:

- User: userId → name, contact, email. Here, userId is the superkey.
- Car: carId → userId, price, mileage, availability. The carId is the superkey for all functional dependencies.
- EcoPoints: ecoPointsId → userId, points. The ecoPointsId is the superkey, ensuring BCNF compliance.
- Booking: bookingId → carId, userId, startDate, endDate, startMileage, endMileage. The bookingId is the superkey.
- CarRating: ratingId → carId, userId, ratingValue. The ratingId is the superkey.
- UserRoles: userRoleId → userId, role. The userRoleId is the superkey.

Thus, every functional dependency has a superkey on the left-hand side, meaning all the relations in our schema are in BCNF. This ensures that our database is free from update anomalies and is fully optimized for both efficiency and integrity.

6. Relational Schema

```
User(
        userId: INT [PK],
        name: VARCHAR(50),
        contact: VARCHAR(15),
        email: VARCHAR(50)
)

UserRoles(
        userRoleId: INT [PK],
        userId: INT [FK to User.userId],
         role: ENUM("buyer", "seller")
)

Car(
        carId: INT [PK],
        userId: INT [FK to User.userId],
        price: DECIMAL,
        mileage: INT,
        availability: BOOLEAN
)

Booking(
        bookingId: INT [PK],
        carId: INT [FK to Car.carId],
        userId: INT [FK to User.userId],
        startDate: DATE,
        endDate: DATE,
        startMileage: INT,
         endMileage: INT
)


EcoPoints(
    ecoPointsId: INT [PK],
    userId: INT [FK to User.userId],
    points: INT
)

CarRating(
    ratingID: INT [PK],
    carId: INT [FK to Car.carId],
    userId: INT [FK to User.userId],
    ratingValue: INT
)
```