**Online Taxi System Database Project**

# **Project Overview:**

The Online Taxi System Database is designed to manage taxi bookings seamlessly by connecting users with drivers, simplifying the booking process, and efficiently tracking rides, payments, and feedback. The system ensures a smooth experience by organizing driver, vehicle, and ride data while enhancing user satisfaction. This database project aims to provide a solid backend structure for modern urban transportation needs, incorporating scalable and efficient design principles.

# **Step 1: Database Planning**

Database planning involved understanding the core functionality of an online taxi system. This step included identifying the essential data elements and their relationships to ensure a well-structured and efficient design.

## **Key Areas of Focus**

1.**Drivers**: Driver details such as ID, name, license, vehicle, and availability.

2. **Customers**: Customer information like ID, contact details, ride history, and payment preferences.

3. **Rides**: Ride specifics including route, fare, and associated driver and customer.

4. **Payments**: Payment methods, amounts, and transaction histories.

5. **Feedback**: Ratings and reviews from customers for drivers and rides.

## **Planning Outcome**

This phase laid the groundwork for a relational database, ensuring the inclusion of primary and foreign keys to define relationships. Mapping these data flows helped envision a system that supports:

- Efficient driver management.

- Smooth ride tracking.

- Reliable payment processing.

- Feedback collection to improve service.

# **Step 2: System Definition**

## **Mission Statement**

“**To create a reliable and user-friendly online taxi system that enhances urban transportation by efficiently connecting passengers with drivers while ensuring safety and convenience**.”

## **Objectives**

- Manage driver profiles, vehicle details, and availability status.

- Track customer profiles, ride history, and payment preferences.

- Monitor ride requests and driver assignments in real time.

- Maintain accurate records of payments and transaction history.

- Enable feedback collection for continuous service improvement.

- Integrate features like ride reminders and notifications (future enhancements).

## **Scope**

This system caters to taxi service operators, drivers, and customers, offering comprehensive management of:

- Driver registrations and vehicle assignments.

- Customer ride bookings and history.

- Payment processing and feedback tracking.

Future updates may include real-time ride tracking and AI-driven route optimization.

# **Step 3: Requirements Collection and Analysis**

## **Requirement Gathering**

### **Techniques Used**

1. **Interviews:**

- Drivers shared their need for pre-ride details and post-ride summaries, such as fare breakdowns and ratings.

- Administrators emphasized the importance of real-time ride tracking and system analytics.

2. **Surveys**:

- Customers expressed preferences for viewing driver details, estimated arrival times, and secure payment methods.

## **Analysis**

### **Key Findings**

- **Drivers:** Simplified access to ride details boosts efficiency.

- **Customers**: Secure payments and tracking features increase satisfaction.

- **Administrators:** Comprehensive reports support better decision-making.

# **Step 4: Database Design**

## **Entities and Attributes**

### **Entities and Their Attributes**

1.**User:**

- User\_ID (PK), Name, Phone\_Number, Email, Registration\_Date.

2. **Driver:**

- Driver\_ID (PK), Name, License\_Number, Phone\_Number, Vehicle\_ID (FK).

3. **Vehicle:**

- Vehicle\_ID (PK), License\_Plate, Model, Capacity, Vehicle\_Type.

4. **Ride:**

- Ride\_ID (PK), Start\_Location, End\_Location, Ride\_Date, Fare.

5. **Booking:**

- Booking\_ID (PK), User\_ID (FK), Ride\_ID (FK), Booking\_Date, Status.

6. **Payment:**

- Payment\_ID (PK), Ride\_ID (FK), Payment\_Method, Amount, Payment\_Date.

7**. Feedback**:

- Feedback\_ID (PK), User\_ID (FK), Driver\_ID (FK), Rating, Comment.

8. **Location:**

- Location\_ID (PK), Address, City, State, Zip\_Code.

## **Relationships**

## **Relationships with degree and cardinality between entities:**

## 

**User Makes Booking**

**Degree**: 2

**Cardinality**: One-to-Many (One user can make multiple bookings, but each booking is linked to only one user)

**Booking Confirms Ride**

**Degree**: 2

**Cardinality:** One-to-One (Each booking corresponds to a unique ride)

**Driver Operates Vehicle**

**Degree**: 2

**Cardinality:** One-to-One (Each driver is assigned one vehicle, and each vehicle has one designated driver) **Driver Completes Ride**

**Degree**: 2

**Cardinality**: One-to-Many (One driver can complete multiple rides, but each ride is associated with a single driver)

**User Provides Feedback**

**Degree**: 2

**Cardinality**: One-to-Many (One user can provide feedback for multiple rides, but each feedback entry is tied to one user)

**Driver Receives Feedback**

**Degree**: 2

**Cardinality**: One-to-Many (A driver can receive multiple feedback entries, each linked to a different ride or user)

**Ride Generates Payment**

**Degree:** 2

**Cardinality**: One-to-One (Each ride has one associated payment record, and each payment is for one ride)

**Ride Occurs\_at Location**

**Degree**: 2

**Cardinality**: Many-to-One (Multiple rides can start or end at the same location, but each ride has a specific start and end location)

# **Step 5: DBMS Selection**

**Chosen DBMS**: MySQL

### **Reasons for Selection**

1. **Performance:** MySQL is optimized for high-speed transactions, ideal for real-time ride bookings.

2. **Scalability**: Easily supports growing data needs as the taxi system expands.

3. **Compatibility**: Works seamlessly with web and mobile applications.

4. **Community Support**: Extensive resources and community forums for troubleshooting.

# **Step 6: Application Design**

## **System Modules and User Interaction**

The application design focuses on creating a user-friendly interface and backend structure to manage the taxi booking process. Key modules include:

1**. User Management:**

**- Features:**

- User registration and login.

- Profile management (update name, email, or phone number).

- View ride history.

- **Data Interaction**:

- Fetches and updates user details from the database.

2. **Driver Management**:

- **Features:**

- Register and assign drivers to vehicles.

- Update availability status.

- View ride history and feedback.

- **Data Interaction**:

- Links drivers to vehicles and manages their schedules.

**3. Ride Management:**

**- Features**:

- Real-time ride booking.

- Automatic driver assignment based on location.

- Status updates (e.g., "Ongoing," "Completed").

- **Data Interaction:**

- Tracks ride details and stores them in the Ride entity.

4. **Payment Processing:**

**- Features:**

- Multiple payment methods (credit card, mobile wallet, cash).

- Generate receipts.

- View transaction history.

- **Data Interaction:**

- Links rides to payments for accurate fare tracking.

5. **Feedback System**:

- **Features:**

- Customers can rate rides and provide comments.

- Drivers can view feedback to improve services.

- **Data Interaction**:

- Stores feedback linked to both users and drivers.

6**. Admin Dashboard:**

- **Features:**

- Overview of system performance.

- Analytics on rides, payments, and feedback.

- Manage user and driver accounts.

- **Data Interaction:**

- Fetches data across entities to generate reports.

## **User Views and Data Cross-Reference**

|  |  |
| --- | --- |
| **USER VIEW** | **DATA USED** |
| User Management | User profiles, Account types |
| Driver Management | Driver profiles, Vehicle details |
| Ride Management | Ride schedules, Ride history |
| Payment Processing | Payment records, Transaction types |
| Feedback System | Customer feedback, Ratings |
| Admin Dashboard | System analytics, User statistics |

# **ER Diagram**

## **Overview**

The ER diagram captures the relationships and attributes of the primary entities:

- **One-to-One**: Driver to Vehicle, Booking to Ride, Ride to Payment.

- **One-to-Many**: User to Booking, Driver to Feedback, Driver to Rides.

- **Many-to-One**: Multiple rides linked to common locations.