# *ONLINE TAXI SYSTEM DATABASE*


**DATABASE COURSE**

**ASSIGNMENT NO 3:**

**SUBMITTED TO MAM HINA RASHID**

**TEAM MEMBERS**:

ALIZA MARYAM-BSE233020

MAIRA TARIQ-BSE233037

TALHA USMAN-BSE233017

***Link e din URL*** : https://www.linkedin.com/in/aliza-maryam-939397340

https://www.linkedin.com/in/talha-usman-269a81340/

https://www.linkedin.com/in/maira-tariq-8228b9276

***Git hub URL:*** https://github.com/alizamaryam39

https://github.com/talha12-code/DATABASE.git

https://github.com/maira12-code/database.git

# Contents

# Entities and Attributes in Your Online Taxi System Database

## Entities:

**1. User**

- **Attributes**:
  - user_id: Unique identifier for the user.
  - name: Name of the user.
  - email: Contact email, unique for every user.
  - phone: User's phone number, also unique.
  - password: Encrypted user password.
  - created_at: Timestamp for when the user signed up.

**2. Driver**

- **Attributes**:
  - driver_id: Unique identifier for each driver.
  - name: Name of the driver.
  - license_number: Unique license number for the driver.
  - phone: Contact number, unique for every driver.
  - vehicle_id: Reference to the assigned vehicle.
  - status: Driver's availability (e.g., 'Available', 'Busy').

**3. Vehicle**

- **Attributes**:
  - vehicle_id: Unique identifier for each vehicle.
  - vehicle_type: Type of vehicle (e.g., car, van, bike).
  - model: Vehicle model name.
  - plate_number: Unique vehicle registration number.
  - driver_id: Reference to the assigned driver.

**4. Ride**

- **Attributes**:
  - ride_id: Unique identifier for each ride.
  - driver_id: Reference to the driver.
  - user_id: Reference to the user who booked the ride.
  - pickup_location and drop_location: Ride locations.
  - fare: Cost of the ride.
  - ride_date: Timestamp of when the ride occurred.

**5. Booking**

- **Attributes**:
  - booking_id: Unique identifier for each booking.
  - user_id: Reference to the user who made the booking.
  - ride_id: Reference to the ride.
  - booking_date: Timestamp of the booking.
  - status: Booking status (e.g., 'Pending', 'Completed', 'Cancelled').

**6. Payment**

- **Attributes**:
  - payment_id: Unique identifier for each payment.
  - booking_id: Reference to the booking.
  - amount: Payment amount.
  - payment_method: Mode of payment (e.g., Cash, Card, Online).
  - payment_date: Timestamp for when the payment was made.

**7. Feedback**

- **Attributes**:
  - feedback_id: Unique identifier for each feedback entry.
  - user_id: Reference to the user providing feedback.
  - ride_id: Reference to the ride.
  - rating: Numerical rating (1-5).
  - comments: Textual feedback.
  - feedback_date: Timestamp of the feedback.

**8. Location**

- **Attributes**:
  - location_id: Unique identifier for each location.
  - city_name: Name of the city.
  - latitude and longitude: Geographical coordinates.

---

## Entity Relationships

### 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)

---

# 2. Challenges and Considerations During Design

**Handling Many-to-Many Relationships**

- **Challenge**:
  In the Online Taxi System, many-to-many relationships may arise, such as a Driver being able to operate multiple Rides, and a User being linked to multiple Bookings.
- **Solution**:
  We created join tables like Booking to handle such relationships effectively. The Booking table serves as a bridge between the User and the Ride, linking them with foreign keys and maintaining the relationship structure.

**Data Integrity**

- **Challenge**:
  Maintaining data integrity is critical to ensuring consistency in the database. For instance, if a User or

Driver is deleted, all associated records (such as Bookings or Rides) should be handled properly to avoid orphaned data.

- **Solution**:
  - We used **foreign key constraints** to ensure proper relationships between tables.
  - Implemented **onDelete: CASCADE** on relevant tables such as Ride, Booking, and Payment to automatically remove related entries when a parent record is deleted.

**Scalability and Performance**

- **Challenge**:
  As the Online Taxi System grows, the database will handle an increasing number of Users, Drivers, Vehicles, and Rides. Efficient querying of frequently accessed data, such as active drivers or ride history, becomes essential.
- **Solution**:
  - Added **indexes** on frequently queried fields such as email in the User table, status in the Booking table, and license_number in the Driver table.
  - Optimized database design by normalizing tables to reduce redundancy and improve query performance.

---

# 3.Define Tables, Fields, and Data Types

CREATE DATABASE ONLINE_TAXI_SYSTEM:

```
MariaDB [(none)]> CREATE DATABASE ONLINE_TAXI_SYSTEM;
Query OK, 1 row affected (0.003 sec)
```

```
MariaDB [(none)]> USE ONLINE_TAXI_SYSTEM;
Database changed
```

USER SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE USER;
+------------+--------------+------+-----+---------------------+----------------+
| Field      | Type         | Null | Key | Default             | Extra          |
+------------+--------------+------+-----+---------------------+----------------+
| user_id    | int(11)      | NO   | PRI | NULL                | auto_increment |
| name       | varchar(100) | NO   |     | NULL                |                |
| email      | varchar(100) | NO   | UNI | NULL                |                |
| phone      | varchar(15)  | NO   | UNI | NULL                |                |
| password   | varchar(100) | NO   |     | NULL                |                |
| created_at | timestamp    | NO   |     | current_timestamp() |                |
+------------+--------------+------+-----+---------------------+----------------+
6 rows in set (0.017 sec)
```

DRIVER SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE DRIVER;
+----------------+-------------------------------+------+-----+-----------+----------------+
| Field          | Type                          | Null | Key | Default   | Extra          |
+----------------+-------------------------------+------+-----+-----------+----------------+
| driver_id      | int(11)                       | NO   | PRI | NULL      | auto_increment |
| name           | varchar(100)                  | NO   |     | NULL      |                |
| license_number | varchar(50)                   | NO   | UNI | NULL      |                |
| phone          | varchar(15)                   | NO   | UNI | NULL      |                |
| status         | enum('Available','Busy')      | YES  |     | Available |                |
| vehicle_id     | int(11)                       | YES  | MUL | NULL      |                |
+----------------+-------------------------------+------+-----+-----------+----------------+
6 rows in set (0.074 sec)
```

VEHICLE SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE VEHICLE;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| vehicle_id   | int(11)      | NO   | PRI | NULL    | auto_increment |
| vehicle_type | varchar(50)  | NO   |     | NULL    |                |
| model        | varchar(100) | NO   |     | NULL    |                |
| plate_number | varchar(20)  | NO   | UNI | NULL    |                |
| driver_id    | int(11)      | YES  | MUL | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
5 rows in set (0.072 sec)
```

RIDE SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE RIDE;
+-----------------+---------------+------+-----+---------------------+----------------+
| Field           | Type          | Null | Key | Default             | Extra          |
+-----------------+---------------+------+-----+---------------------+----------------+
| ride_id         | int(11)       | NO   | PRI | NULL                | auto_increment |
| driver_id       | int(11)       | NO   | MUL | NULL                |                |
| user_id         | int(11)       | NO   | MUL | NULL                |                |
| pickup_location | varchar(255)  | NO   |     | NULL                |                |
| drop_location   | varchar(255)  | NO   |     | NULL                |                |
| fare            | decimal(10,2) | NO   |     | NULL                |                |
| ride_date       | timestamp     | NO   |     | current_timestamp() |                |
+-----------------+---------------+------+-----+---------------------+----------------+
7 rows in set (0.074 sec)
```

BOOKING SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE BOOKING;
+--------------+-----------------------------------------+------+-----+---------------------+----------------+
| Field        | Type                                    | Null | Key | Default             | Extra          |
+--------------+-----------------------------------------+------+-----+---------------------+----------------+
| booking_id   | int(11)                                 | NO   | PRI | NULL                | auto_increment |
| user_id      | int(11)                                 | NO   | MUL | NULL                |                |
| ride_id      | int(11)                                 | NO   | MUL | NULL                |                |
| booking_date | timestamp                               | NO   |     | current_timestamp() |                |
| status       | enum('Pending','Completed','Cancelled') | YES  |     | Pending             |                |
+--------------+-----------------------------------------+------+-----+---------------------+----------------+
5 rows in set (0.073 sec)
```

PAYMENT SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE PAYMENT;
+----------------+-------------------------------+------+-----+---------------------+----------------+
| Field          | Type                          | Null | Key | Default             | Extra          |
+----------------+-------------------------------+------+-----+---------------------+----------------+
| payment_id     | int(11)                       | NO   | PRI | NULL                | auto_increment |
| booking_id     | int(11)                       | NO   | MUL | NULL                |                |
| amount         | decimal(10,2)                 | NO   |     | NULL                |                |
| payment_method | enum('Cash','Card','Online')  | NO   |     | NULL                |                |
| payment_date   | timestamp                     | NO   |     | current_timestamp() |                |
+----------------+-------------------------------+------+-----+---------------------+----------------+
5 rows in set (0.020 sec)
```

FEEDBACK SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE FEEDBACK;
+---------------+-----------+------+-----+---------------------+----------------+
| Field         | Type      | Null | Key | Default             | Extra          |
+---------------+-----------+------+-----+---------------------+----------------+
| feedback_id   | int(11)   | NO   | PRI | NULL                | auto_increment |
| user_id       | int(11)   | NO   | MUL | NULL                |                |
| ride_id       | int(11)   | NO   | MUL | NULL                |                |
| rating        | int(11)   | YES  |     | NULL                |                |
| comments      | text      | YES  |     | NULL                |                |
| feedback_date | timestamp | NO   |     | current_timestamp() |                |
+---------------+-----------+------+-----+---------------------+----------------+
6 rows in set (0.074 sec)
```

LOCATION SCHEMA:

```
MariaDB [ONLINE_TAXI_SYSTEM]> DESCRIBE LOCATION;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| location_id | int(11)      | NO   | PRI | NULL    | auto_increment |
| city_name   | varchar(100) | NO   |     | NULL    |                |
| latitude    | decimal(9,6) | NO   |     | NULL    |                |
| longitude   | decimal(9,6) | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.010 sec)
```

# TABLE CREATION CODE:

USER TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE USER(
    -> user_id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(100) NOT NULL,
    -> email VARCHAR(100) UNIQUE NOT NULL,
    -> phone VARCHAR(15) UNIQUE NOT NULL,
    -> password VARCHAR(100) NOT NULL,
    -> created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
Query OK, 0 rows affected (0.097 sec)
```

DRIVER TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE DRIVER(
    -> driver_id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(100) NOT NULL,
    ->  license_number VARCHAR(50) UNIQUE NOT NULL,
    -> phone VARCHAR(15) UNIQUE NOT NULL,
    -> status ENUM('Available','Busy')DEFAULT 'Available');
Query OK, 0 rows affected (0.102 sec)
```

```
MariaDB [ONLINE_TAXI_SYSTEM]> ALTER TABLE DRIVER ADD COLUMN vehicle_id INT,
    -> ADD FOREIGN KEY(vehicle_id) REFERENCES VEHICLE(vehicle_id);
Query OK, 0 rows affected (0.173 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

VEHICLE TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE VEHICLE(
    -> vehicle_id INT AUTO_INCREMENT PRIMARY KEY,
    -> vehicle_type VARCHAR(50) NOT NULL,
    -> model VARCHAR(100) NOT NULL,
    -> plate_number VARCHAR(20) UNIQUE NOT NULL,
    -> driver_id INT,
    -> FOREIGN KEY (driver_id) REFERENCES DRIVER(driver_id));
Query OK, 0 rows affected (0.098 sec)
```

RIDE TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE RIDE(
    ->  ride_id INT AUTO_INCREMENT PRIMARY KEY,
    -> driver_id INT NOT NULL,
    -> user_id INT NOT NULL,
    -> pickup_location VARCHAR(255) NOT NULL,
    -> drop_location VARCHAR(255) NOT NULL,
    ->  fare DECIMAL(10,2) NOT NULL,
    -> ride_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->  FOREIGN KEY (driver_id) REFERENCES Driver(driver_id),
    ->  FOREIGN KEY (user_id) REFERENCES USER(user_id));
Query OK, 0 rows affected (0.058 sec)
```

BOOKING TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE BOOKING(
    ->  booking_id INT AUTO_INCREMENT PRIMARY KEY,
    -> user_id INT NOT NULL,
    ->   ride_id INT NOT NULL,
    -> booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    -> status ENUM('Pending', 'Completed', 'Cancelled') DEFAULT 'Pending',
    ->  FOREIGN KEY (user_id) REFERENCES USER(user_id),
    -> FOREIGN KEY (ride_id) REFERENCES RIDE(ride_id));
Query OK, 0 rows affected (0.101 sec)
```

PAYMENT TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE PAYMENT(
    ->  payment_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  booking_id INT NOT NULL,
    ->   amount DECIMAL(10,2) NOT NULL,
    -> payment_method ENUM('Cash', 'Card', 'Online') NOT NULL,
    ->   payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ->  FOREIGN KEY (booking_id) REFERENCES BOOKING(booking_id));
Query OK, 0 rows affected (0.094 sec)
```

FEEDBACK TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE FEEDBACK(
    ->  feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    -> user_id INT NOT NULL,
    ->   ride_id INT NOT NULL,
    ->  rating INT CHECK (rating BETWEEN 1 AND 5),
    ->  comments TEXT,
    -> feedback_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    -> FOREIGN KEY (user_id) REFERENCES USER(user_id),
    ->  FOREIGN KEY (ride_id) REFERENCES RIDE(ride_id));
Query OK, 0 rows affected (0.100 sec)
```

LOCATION TABLE:

```
MariaDB [ONLINE_TAXI_SYSTEM]> CREATE TABLE LOCATION(
    ->  location_id INT AUTO_INCREMENT PRIMARY KEY,
    -> city_name VARCHAR(100) NOT NULL,
    ->  latitude DECIMAL(9,6) NOT NULL,
    -> longitude DECIMAL(9,6) NOT NULL);
Query OK, 0 rows affected (0.092 sec)
```

# 4. Establish Relationships Between Tables

In our schema, we have set up **foreign key relationships** to ensure proper connections between related tables. These relationships maintain data consistency and integrity across the database.

**Examples**:

- In the **Ride** table, **user_id** references the **User** table, and **driver_id** references the Driver table.
- The **Booking** table links to the **Ride** table via **ride_id** and to the User table via **user_id.**
- The **Vehicle** table connects to the **Driver** table through **driver_id.**
- The **Payment** table references the **Booking** table using **booking_id.**

These relationships ensure that the data flows logically and maintains its integrity within the system.

# 5. Transform ERD into Relational Data Model (RDM)

After analyzing the **Entity-Relationship Diagram (ERD)**, we transformed it into a **Relational Data Model (RDM)** by creating relational tables and defining their relationships. Each entity in the ERD corresponds to a table, and relationships are implemented using foreign keys.

# 6.Database Design Decisions Report for Online Taxi System

**Introduction**

In this project, we designed a relational database for an **Online Taxi System** to manage users, drivers, vehicles, bookings, rides, payments, and other essential features. The design followed the principles of the **Database Design Life Cycle (DDLC)** to ensure an efficient, scalable, and reliable data model. This report outlines the major decisions made during the database design process, covering entities, attributes, relationships, and considerations for scalability and data integrity.

## 1. Entity and Relationship Design

**Entities and Attributes**

- **User**: Represents the customers using the platform. Attributes include user_id, name, email, phone, and address.
- **Driver**: Represents drivers registered on the platform. Attributes include driver_id, name, license_number, and availability_status.
- **Vehicle**: Represents the vehicles operated by drivers. Attributes include vehicle_id, make, model, year, and driver_id (linking to the Driver entity).
- **Booking**: Captures booking details made by users. Attributes include booking_id, user_id, ride_id, status, and created_at.
- **Ride**: Represents the rides booked and completed. Attributes include ride_id, driver_id, pickup_location, dropoff_location, start_time, end_time, and fare_amount.

- **Location**: Stores data about pickup and dropoff points. Attributes include location_id, latitude, longitude, and address.
- **Feedback**: Stores feedback provided by users for drivers. Attributes include feedback_id, user_id, ride_id, rating, and comments.
- **Payment**: Tracks payments for completed rides. Attributes include payment_id, ride_id, amount, payment_status, and transaction_date.

### Relationships

- **User → Booking**: One user can make multiple bookings, but each booking is linked to only one user (one-to-many).
- **Driver → Vehicle**: Each driver operates one vehicle, and each vehicle is assigned to one driver (one-to-one).
- **Booking → Ride**: Each booking corresponds to one ride, creating a one-to-one relationship.
- **Ride → Payment**: Each ride has a corresponding payment, creating another one-to-one relationship.
- **Ride → Feedback**: One ride can generate one feedback entry per user, making it a one-to-many relationship.
- **Ride → Location**: Multiple rides can start or end at the same location, making this a many-to-one relationship.

## 2. Design Considerations

**Normalization and Redundancy Avoidance**

To minimize data redundancy, we normalized the database up to the **Third Normal Form (3NF)**:

- Separated the **Vehicle** and **Driver** entities to avoid storing duplicate driver or vehicle details.
- The **Location** entity was introduced to manage frequently used pickup and dropoff locations, ensuring consistency.

**Scalability and Performance**

- **Indexes** were added to frequently queried fields such as email in the **User** table and status in the **Booking** table to improve query performance.
- **ON DELETE CASCADE** rules were applied to maintain referential integrity. For example, deleting a user automatically removes their bookings and feedback records.

**Data Integrity**

- **Foreign Key Constraints**: These ensure that relationships between tables remain consistent.
- **Cascade Updates**: Any changes to a referenced key propagate across related tables to avoid orphaned records.

## 3. Future Expansion Considerations

The database design is flexible and ready for future growth:

- **User Reviews**: A Review table can be added to allow users to review rides or drivers.
- **Dynamic Pricing**: The **Ride** table can be extended with fields like demand_factor for surge pricing.
- **Driver Ratings**: Integrating cumulative driver ratings based on user feedback.
- **Multi-Language Support**: Extend the **Location** table with fields for translated address details.