DBMS Project

Taxi Management System

R Srivatsa
PES1UG21CS464

Bhargav Punugupati
PES1UG21CS458

Abstract

The rapid growth of urban populations has intensified the demand for efficient and reliable taxi services, necessitating advanced technological solutions for their management. This paper introduces a robust Database Management System (DBMS) tailored for Taxi Management Systems (TMS) to address the complexities associated with real-time data handling, fleet management, and user interactions.

Our proposed DBMS integrates seamlessly with TMS, offering a scalable and flexible architecture that accommodates the dynamic nature of taxi operations. The system focuses on optimizing key functionalities such as real-time location tracking, driver assignment, fare calculation, and customer interactions. By leveraging relational database principles, the DBMS ensures data integrity, consistency, and security throughout the system.

Key features of the DBMS include:

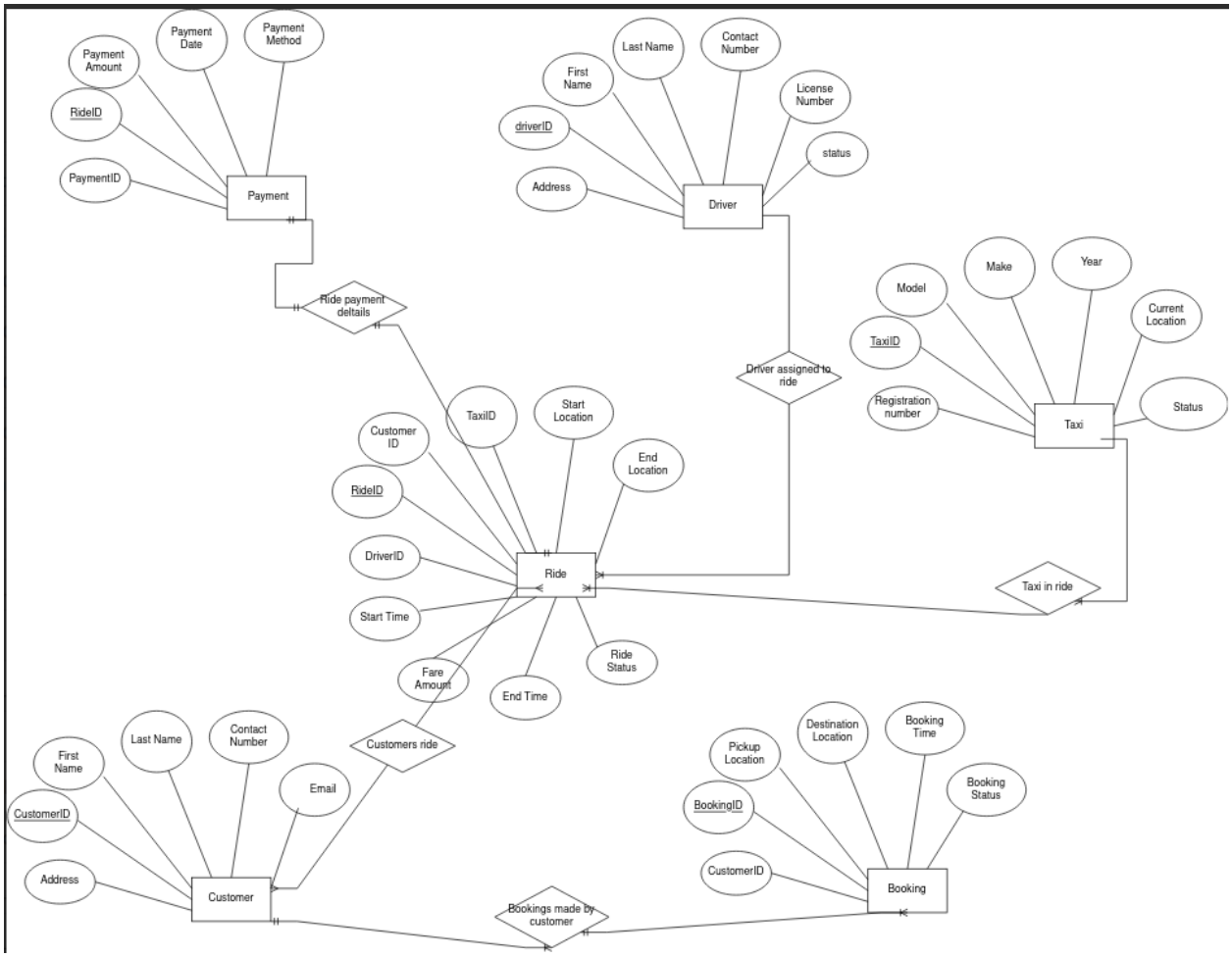
1. Real-time Data Processing: The system employs efficient algorithms for real-time data processing, allowing for instantaneous updates on taxi availability, location, and user requests.

2. Fleet Optimization: Advanced algorithms for dynamic fleet management enable intelligent driver dispatching, considering factors like proximity, traffic conditions, and driver availability to enhance overall service efficiency.

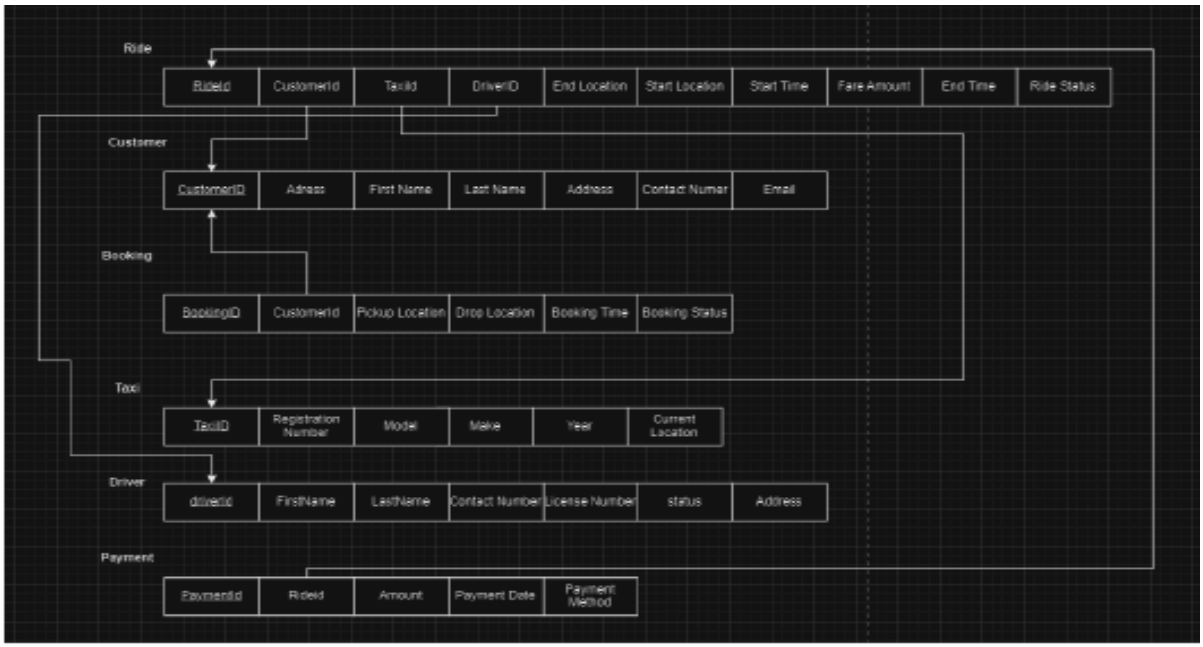
3. User Interaction and Feedback: The DBMS facilitates seamless interactions between users and the TMS through a user-friendly interface, enabling booking, payment, and feedback mechanisms for an enhanced customer experience.

4. Scalability and Performance: The system is designed to handle increasing data volumes as taxi services expand, ensuring optimal performance and responsiveness even under high demand scenarios.

Security and Data Integrity: Robust security measures are implemented to protect sensitive user information and ensure the integrity of the system's database,


Payment Date
Payment Method
Payment Amount
RideID
PaymentID
Payment
Last Name
Contact Number
First Name
License Number
driverID
status
Address
Driver
Ride payment deltails
Driver assigned to ride
Model
Make
Year
Current Location
TaxiID
Registration number
Status
Taxi
Taxi in ride
Customer ID
TaxiID
Start Location
End Location
RideID
DriverID
Start Time
Ride
Fare Amount
End Time
Ride Status
Customers ride
First Name
Last Name
Contact Number
Email
CustomerID
Address
Customer
Pickup Location
Destination Location
Booking Time
Booking Status
BookingID
CustomerID
Booking
Bookings made by customer

Relational Schema



Crud Operations

```
CREATE TABLE `cabs` (
  `vehicle_number` int NOT NULL,
  `registration_number` varchar(255) NOT NULL,
  `seating_capacity` int NOT NULL,
  `model` varchar(255) NOT NULL,
  `type` varchar(255) NOT NULL,
  `status` varchar(255) NOT NULL,
  `driverID` int NOT NULL,
  PRIMARY KEY (`vehicle_number`),
  UNIQUE KEY `vehicle_number` (`vehicle_number`),
  UNIQUE KEY `registration_number` (`registration_number`),
  UNIQUE KEY `vehicle_number_2` (`vehicle_number`),
  UNIQUE KEY `vehicle_number_3` (`vehicle_number`),
  UNIQUE KEY `vehicle_number_4` (`vehicle_number`),
  KEY `fk_cabs_drivers` (`driverID`),
  CONSTRAINT `fk_cabs_drivers` FOREIGN KEY (`driverID`) REFERENCES `drivers` (`driver_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
DROP TABLE IF EXISTS `customer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `customer` (
  `userID` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `address` varchar(255) NOT NULL,
  `phone` int DEFAULT NULL,
  `DOB` date NOT NULL,
  `username` varchar(255) NOT NULL,
  `password_hash` varchar(255) NOT NULL,
  PRIMARY KEY (`userID`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `name` (`name`,`phone`)
) ENGINE=InnoDB AUTO_INCREMENT=971524507 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```sql
DROP TABLE IF EXISTS `drivers`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `drivers` (
  `driver_id` int NOT NULL AUTO_INCREMENT,
  `phone` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `address` varchar(255) NOT NULL,
  `dob` date NOT NULL,
  `license_number` varchar(255) NOT NULL,
  `rating_score` int NOT NULL,
  PRIMARY KEY (`driver_id`),
  UNIQUE KEY `license_number` (`license_number`),
  CONSTRAINT `check_phone_length` CHECK ((length(`phone`) <= 10)),
  CONSTRAINT `drivers_chk_1` CHECK (((`rating_score` >= 0) and (`rating_score` <= 5)))
) ENGINE=InnoDB AUTO_INCREMENT=999317053 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

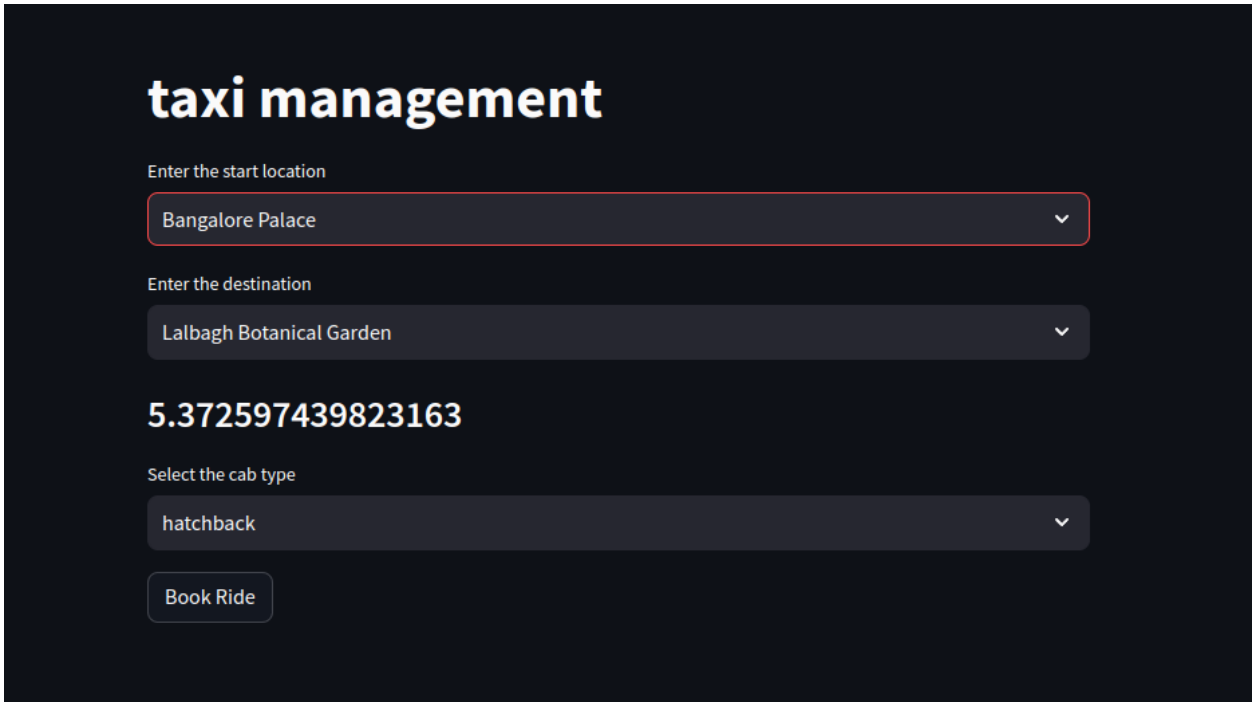
```sql
DROP TABLE IF EXISTS `locations`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `locations` (
  `location_id` int NOT NULL AUTO_INCREMENT,
  `location_name` varchar(255) NOT NULL,
  `latitude` double NOT NULL,
  `longitude` double NOT NULL,
  PRIMARY KEY (`location_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```sql
CREATE TABLE `payments` (
  `payment_id` int NOT NULL AUTO_INCREMENT,
  `payment_type` varchar(255) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `customer_id` int NOT NULL,
  `driver_id` int NOT NULL,
  `booking_id` int DEFAULT NULL,
  PRIMARY KEY (`payment_id`),
  KEY `customer_id` (`customer_id`),
  KEY `driver_id` (`driver_id`),
  KEY `booking_id` (`booking_id`),
  CONSTRAINT `fk_payments_rides` FOREIGN KEY (`payment_id`) REFERENCES `rides` (`booking_id`),
  CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`userID`),
  CONSTRAINT `payments_ibfk_2` FOREIGN KEY (`driver_id`) REFERENCES `drivers` (`driver_id`),
  CONSTRAINT `payments_ibfk_3` FOREIGN KEY (`booking_id`) REFERENCES `rides` (`booking_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

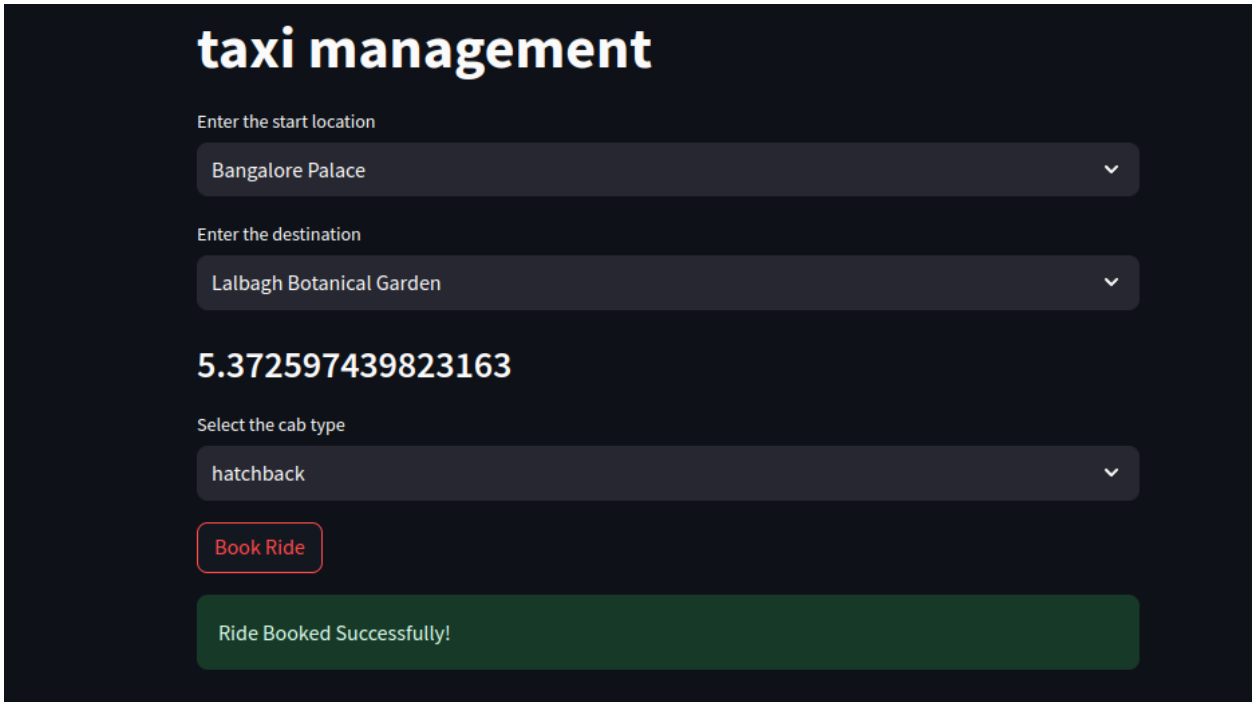
```sql
DROP TABLE IF EXISTS `rides`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `rides` (
  `booking_id` int NOT NULL AUTO_INCREMENT,
  `pickup_location` varchar(255) NOT NULL,
  `drop_location` varchar(255) NOT NULL,
  `start_time` datetime NOT NULL,
  `end_time` datetime NOT NULL,
  `rating` int NOT NULL,
  `distance` int NOT NULL,
  `customer_Id` int NOT NULL,
  `driver_ID` int NOT NULL,
  `status` enum('in transit','not in transit') NOT NULL DEFAULT 'not in transit',
  PRIMARY KEY (`booking_id`),
  KEY `fk_cabs_customers` (`customer_Id`),
  KEY `fk_cabs_customers1` (`driver_ID`),
  CONSTRAINT `fk_cabs_customers` FOREIGN KEY (`customer_Id`) REFERENCES `customer` (`userID`),
  CONSTRAINT `fk_cabs_customers1` FOREIGN KEY (`driver_ID`) REFERENCES `drivers` (`driver_id`),
  CONSTRAINT `drop_location` CHECK (((`drop_location` in (_utf8mb4'Lalbagh Botanical Garden',_utf8mb4'Cubbon Park',_utf8mb4'Bangalore Palace',_utf8mb4'Vidhana Soudha
  CONSTRAINT `pickup_location` CHECK ((`pickup_location` in (_utf8mb4'Lalbagh Botanical Garden',_utf8mb4'Cubbon Park',_utf8mb4'Bangalore Palace',_utf8mb4'Vidhana Sou
  CONSTRAINT `valid_drop_location` CHECK ((`drop_location` in (_utf8mb4'Lalbagh Botanical Garden',_utf8mb4'Cubbon Park',_utf8mb4'Bangalore Palace',_utf8mb4'Vidhana S
  CONSTRAINT `valid_pickup_location` CHECK ((`pickup_location` in (_utf8mb4'Lalbagh Botanical Garden',_utf8mb4'Cubbon Park',_utf8mb4'Bangalore Palace',_utf8mb4'Vidha
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Function

```
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` FUNCTION `calculate_distance`(start_location_name VARCHAR(255), end_location_name VARCHAR(255)) RETURNS double
    DETERMINISTIC
BEGIN
  DECLARE start_lat DOUBLE;
  DECLARE start_lng DOUBLE;
  DECLARE end_lat DOUBLE;
  DECLARE end_lng DOUBLE;

  DECLARE earth_radius DOUBLE;
  DECLARE delta_lat DOUBLE;
  DECLARE delta_lng DOUBLE;
  DECLARE a DOUBLE;
  DECLARE c DOUBLE;
  DECLARE distance DOUBLE;

  SET earth_radius = 6371.0710;

  # Extract latitude and longitude values from start_location
  SELECT latitude, longitude
  INTO start_lat, start_lng
  FROM locations
  WHERE location_name = start_location_name;


  # Extract latitude and longitude values from end_location
  SELECT latitude, longitude
  INTO end_lat, end_lng
  FROM locations
  WHERE location_name = end_location_name;

  SET delta_lat = RADIANS(end_lat) - RADIANS(start_lat);
  SET delta_lng = RADIANS(end_lng) - RADIANS(start_lng);

  SET a = SIN(delta_lat / 2) * SIN(delta_lat / 2) +
        COS(RADIANS(start_lat)) * COS(RADIANS(end_lat)) *
        SIN(delta_lng / 2) * SIN(delta_lng / 2);
  SET c = 2 * ATAN2(SQRT(a), SQRT(1 - a));
  SET distance = earth_radius * c;

  RETURN distance;
END ::
```

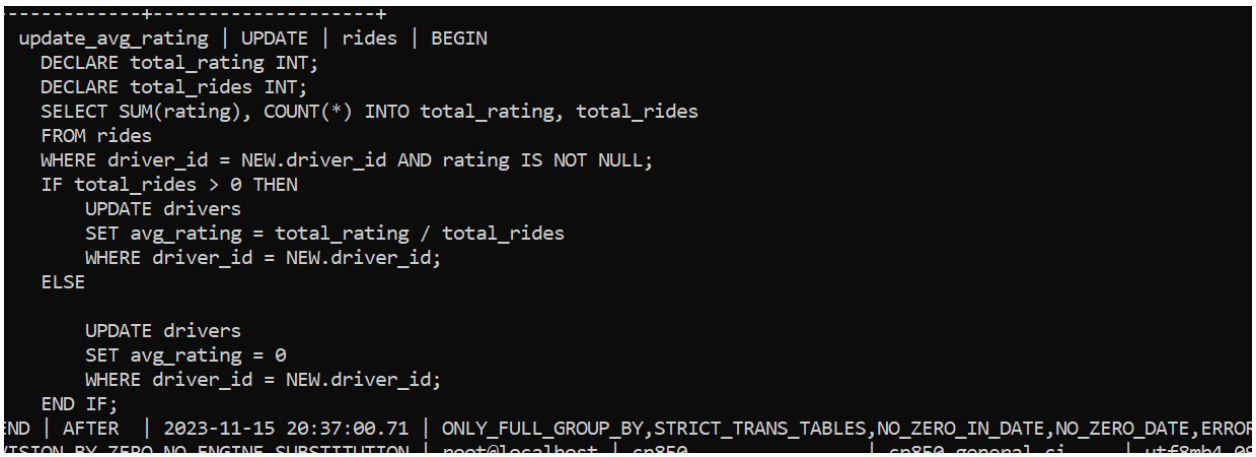



```
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` FUNCTION `getRandomRegistrationNumber`(cabType VARCHAR(255)) RETURNS varchar(255) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
    DECLARE random_reg_number VARCHAR(255);

    SELECT registration_number
    INTO random_reg_number
    FROM cabs
    WHERE type = cabType AND status = 'Available'
    ORDER BY RAND()
    LIMIT 1;

    RETURN random_reg_number;
END ;;
DELIMITER ;
```



this successfully allocates a cab of type ‘hatchback’ which is available ie not in another ride

Trigger



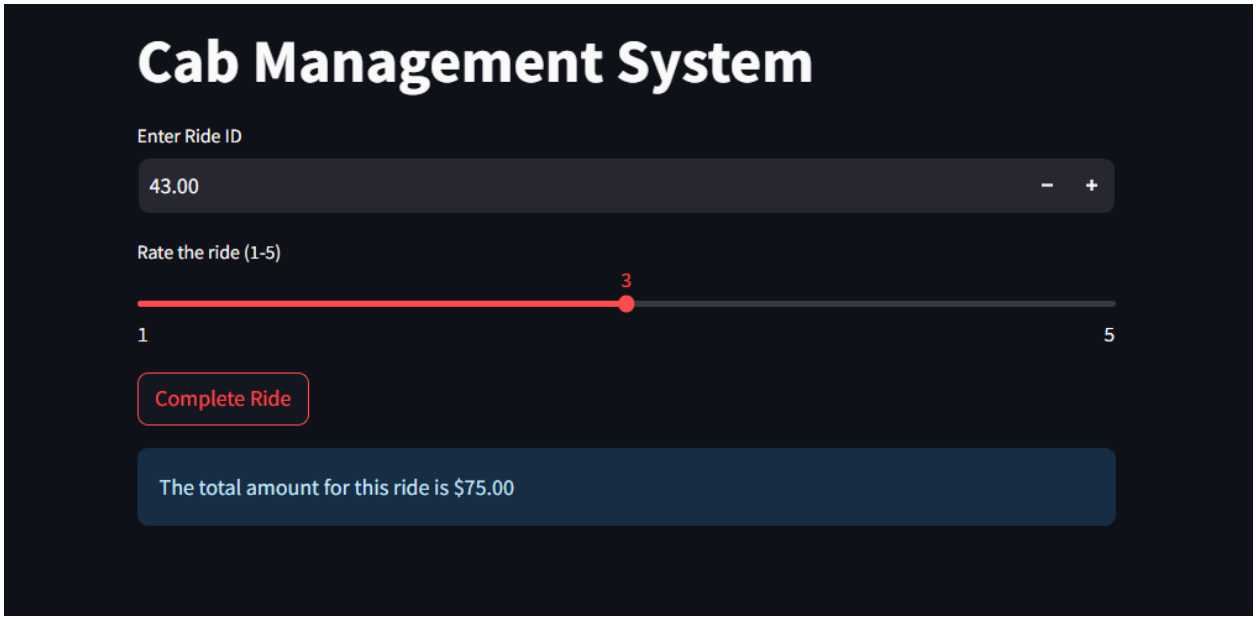
```
------------+--------------------+
 update_avg_rating | UPDATE | rides | BEGIN
   DECLARE total_rating INT;
   DECLARE total_rides INT;
   SELECT SUM(rating), COUNT(*) INTO total_rating, total_rides
   FROM rides
   WHERE driver_id = NEW.driver_id AND rating IS NOT NULL;
   IF total_rides > 0 THEN
       UPDATE drivers
       SET avg_rating = total_rating / total_rides
       WHERE driver_id = NEW.driver_id;
   ELSE

       UPDATE drivers
       SET avg_rating = 0
       WHERE driver_id = NEW.driver_id;
   END IF;
END | AFTER  | 2023-11-15 20:37:00.71 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR
```

Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `ProcessRidePayment`(
    IN p_booking_id INT
)
BEGIN
    DECLARE p_distance FLOAT;
    DECLARE p_base_fare FLOAT DEFAULT 20;
    DECLARE p_rate_per_km FLOAT DEFAULT 5;
    DECLARE p_additional_charges FLOAT DEFAULT 5;
    DECLARE p_payment_type VARCHAR(255) DEFAULT 'UPI';
    DECLARE p_customer_id INT;
    DECLARE p_driver_id INT;
    DECLARE p_total_bill FLOAT;


    SELECT
        distance,
        customer_id,
        driver_id
```

```
    SELECT
        distance,
        customer_id,
        driver_id
    INTO
        p_distance,
        p_customer_id,
        p_driver_id
    FROM rides
    WHERE booking_id = p_booking_id;


    SET p_total_bill = p_base_fare + (p_distance * p_rate_per_km) + p_additional_charges;

    INSERT INTO payments (booking_id, payment_type, amount, customer_id, driver_id)
    VALUES (p_booking_id, p_payment_type, p_total_bill, p_customer_id, p_driver_id);
END
```

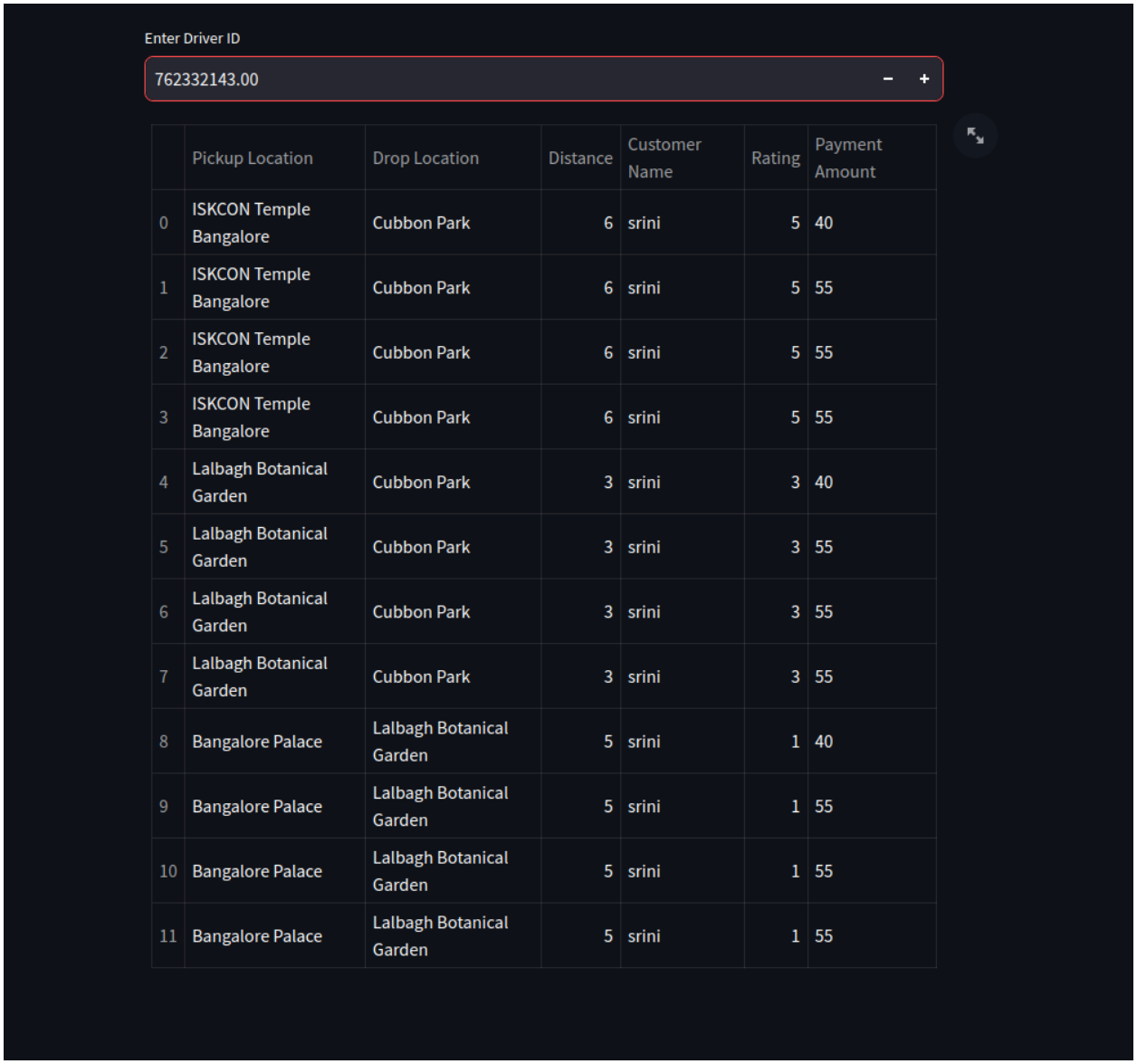
Output



Join Operation

```
query = """
    SELECT
        rides.pickup_location,
        rides.drop_location,
        rides.distance,
        customer.name as customer_name,
        rides.rating,
        payments.amount as payment_amount
    FROM
        rides
        LEFT JOIN drivers ON rides.driver_ID = drivers.driver_id
        LEFT JOIN customer ON rides.customer_Id = customer.userID
        LEFT JOIN payments ON rides.driver_ID = payments.driver_id
    WHERE
        rides.driver_id = %s
"""
```

Output



	Pickup Location	Drop Location	Distance	Customer Name	Rating	Payment Amount
0	ISKCON Temple Bangalore	Cubbon Park	6	srini	5	40
1	ISKCON Temple Bangalore	Cubbon Park	6	srini	5	55
2	ISKCON Temple Bangalore	Cubbon Park	6	srini	5	55
3	ISKCON Temple Bangalore	Cubbon Park	6	srini	5	55
4	Lalbagh Botanical Garden	Cubbon Park	3	srini	3	40
5	Lalbagh Botanical Garden	Cubbon Park	3	srini	3	55
6	Lalbagh Botanical Garden	Cubbon Park	3	srini	3	55
7	Lalbagh Botanical Garden	Cubbon Park	3	srini	3	55
8	Bangalore Palace	Lalbagh Botanical Garden	5	srini	1	40
9	Bangalore Palace	Lalbagh Botanical Garden	5	srini	1	55
10	Bangalore Palace	Lalbagh Botanical Garden	5	srini	1	55
11	Bangalore Palace	Lalbagh Botanical Garden	5	srini	1	55

TRIGGER

calculate_avg_rating | INSERT | rides | BEGIN

```
    DECLARE total_rating INT;
    DECLARE total_rides INT;
```

```
SELECT SUM(rating), COUNT(*) INTO total_rating, total_rides
FROM rides
WHERE driver_id = NEW.driver_id AND rating IS NOT NULL;

IF total_rides > 0 THEN
   UPDATE drivers
   SET avg_rating = total_rating / total_rides
   WHERE driver_id = NEW.driver_id;
ELSE
   UPDATE drivers
    SET avg_rating = 0
   WHERE driver_id = NEW.driver_id;
END IF;
```