OLA Cabs (DBMS Project)

By Group 7:

1) Aditya Choudhary [2020489]
2) Abhijeet Singh [2020486]
3) Amit Malik [2020493]
4) Manav Saini [2020518]

Github Repository: https://github.com/abhijeet486/DBMS_project

Brief Overview:

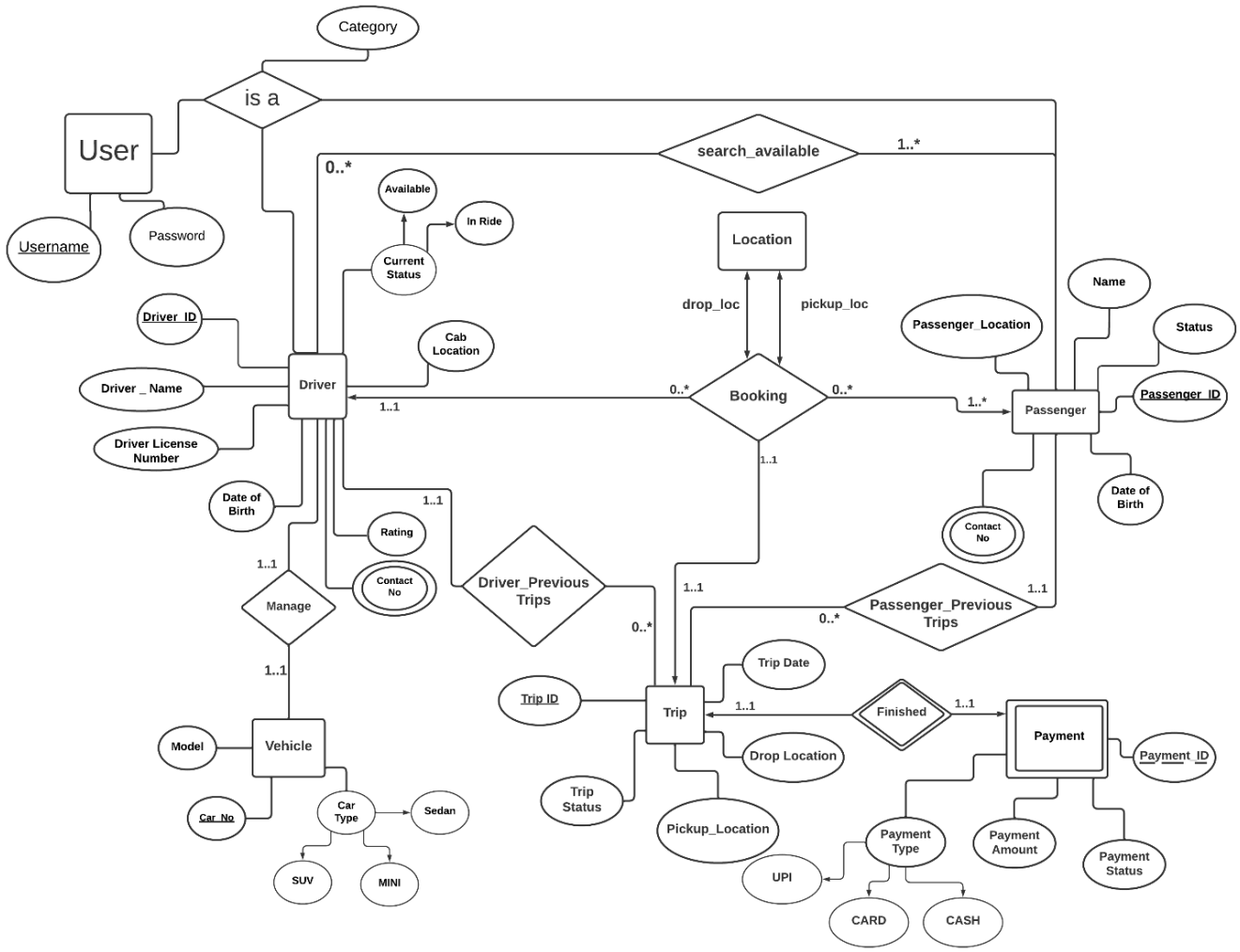
This project provides users with a hassle-free, reliable car taxi service. It uses the concept of DBMS for handling given and generated info.Here easily the customer easily book a taxi for preferred locations at different car type and also give driver to easily interact with customer booking request and do a trip for a living.

Project scope:

Finding right type of car by customer over a preferred location and easily do a trip anywhere from one location to other location

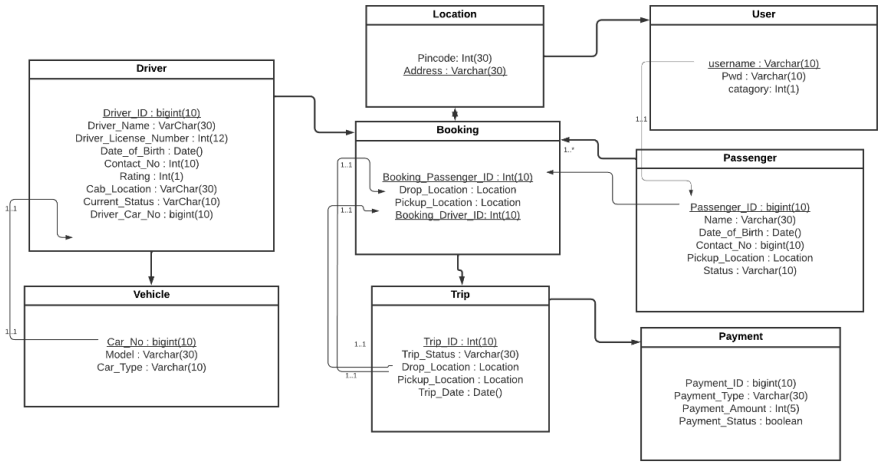
Gives Driver right to earn and opportunity to manage his bookings , trips etc.

Updated ER Diagram:



[Additional PDF of ER Diagram has been attached in the Github Repository]

Updated Relational Schema:



---

create Table Vehicle(
Car_No : Int(12) Primary Key
Model : Varchar(30) NOT NULL
Car_Type : Varchar(10)
check Car_Type in {'Sedan', 'Mini', 'SUV'}
)

create Table Location(
Address : Varchar(30) NOT NULL
Pincode : Int(6) NOT NULL
Primary Key(Address, Pincode)
)

create Table Passenger(
Passenger_ID : Int(10) Primary Key
Name : Varchar(30)
Date_of_Birth : Date()
Contact_No : Int(10)
Pickup_Location : Location NOT NULL
Status : Varchar(10)
)

create Table User(
Username : Varchar(10)
Pwd : Varchar(10)
category : Int(1)
)

create Table Driver(
Driver_ID : Int(10) Primary Key
Driver_Name : VarChar(30)
Driver_License_Number : Int(12) UNIQUE
Date_of_Birth : Date()
Contact_No : Int(10) NOT NULL
Rating : Int(1)
Cab_Location : Location NOT NULL
Current_Status : VarChar(10)
Driver_Car_No : Int(12) Foreign Key References Vehicle(Car_No)
check Current_Status in {'Off Duty','Available','In Ride'}
)

create Table Trip(
Trip_ID : Int(10) Primary Key
Trip_Status : Varchar(30) NOT NULL
Trip_Date_Day : Date() NOT NULL
Trip_Passenger_ID : Int(10) NOT NULL
Trip_Driver_ID : Int(10) NOT NULL
Drop_Location : Location NOT NULL
Pickup_Location : Location NOT NULL
Foreign Key(Trip_Passenger_ID, Trip_Driver_ID, Drop_Location, Pickup_Location)
References Booking(Booking_Passenger_ID, Booking_Driver_ID, Drop_Location, Pickup_Location)
)

create Table Booking(
Booking_Passenger_ID : Int(10) Foreign Key References Passenger(Passenger_ID)
Booking_Driver_ID : Int(10) Foreign Key References Driver(Driver_ID)
Drop_Location : Location NOT NULL
Pickup_Location : Location NOT NULL
Primary Key(Booking_Passenger_ID, Booking_Driver_ID)
check(Drop_Location != Pickup_Location)
)

create Table Payment(
Trip_ID : Int(10) Primary Key references Trip(Trip_ID) ON DELETE CASCADE
Payment_ID : bigint(10) NOT NULL UNIQUE
Payment_Type : Varchar(30)
Payment_Amount : Int(5) NOT NULL
Payment_Status : bool
check Payment_Type in {'CASH', 'CARD', 'UPI', }
)

[Additional PDF of Relational Schema has been attached in the Github Repository]

Views:

1) View booking requests in driver user
2) View customer dashboard (Previous trips summary and current trip status)
3) View driver dashboard (Current trip status)
4) View previous trips in customer user and driver user
5) View manage vehicle page (Driver can switch between different registered vehicles)

Grants:

1) Admin : All privileges in Database as it will handle all operations across tables.

2)Passenger -A) It has select and insert privilege to insert and select data on payment table that he can easily do payment query .
b) It has select privilege (pickuplocation, droplocation) in booking table so that he can view his drop location

c) select privilege to see driver name and his cab location where it is now so that he can easily start his trip .

3) Driver - a) it has privilege to select on payment table where he can see payment received of trip or not by customer and he can see payment status.

b)It has select privilege (pickuplocation, droplocation) in booking table so that he can drop passenger to his drop location

Embedded SQL queries:

1) `Select * from booking where Request_Driver_ID= -1;`
2) `Select count(*) from booking where Request_Passenger_ID={} and Request_Driver_ID<0;`
3) `Select * from booking where Request_Passenger_ID={};`
   `INSERT INTO booking (Drop_Location, Pickup_Location, Request_Passenger_ID, Request_Driver_ID) VALUES ('{}', '{}', '{}', '-1');`

4) Create or Replace view Dashboard as Select Pickup_Location,Drop_Location,Driver_Name as RefName,Contact_number as contactno,Driver_Car_Number from (booking join user on username) join driver where username=Request_Passenger_ID and Request_Driver_ID = Driver_id;
5) Create or Replace view User as Select Driver_id as username,Driver_Name as Name,current_status as status,0 as usertype from Driver where Driver_id="{};
6) Select usertype from User;
7) UPDATE driver SET Driver_Name = '{}' WHERE (Driver_id = '{}');
8) UPDATE driver SET Driver_License_No = '{}' WHERE (Driver_id = '{}');
9) UPDATE driver SET Date_of_Birth = '{}' WHERE (Driver_id = '{}');
10) UPDATE driver SET Contact_number = '{}' WHERE (Driver_id = '{}');
11) Update Passenger SET Name='{}' where Passenger_ID = '{}';
12) Update Passenger SET Date_of_Birth='{}' where Passenger_ID = '{}';
13) Update Passenger SET Contact_Number='{}' where Passenger_ID = '{}';
14) Create or Replace view PreviousTrips as Select * from Trip join Payment on Trip_ID where {}={} and Trip_Id_Pay=Trip_ID;

Indexing:

Entity	**Attribute used for INDEXING**
Booking	Pickup_Location
Booking	Drop_Location
Trip	Trip_Passenger_ID
Trip	Trip_Driver_ID
Passenger	Name
Passenger	Contact_Number
Driver	Driver_Name
Driver	Contact_Number
Booking	Request_Passenger_ID
Booking	Request_Driver_ID
users	Category
Vehicle	Car_Type
Driver	Current_Status

Triggers:

1) If inputted drop location and pickup location by the customer is same then the booking request is not accepted

```
DELIMITER $$
CREATE  TRIGGER location_same
            AFTER INSERT
            ON booking FOR EACH ROW
            BEGIN
              IF (NEW.Pickup_Location = New.Drop_Location)
              THEN
              Delete from booking ;


END IF;
END;
```

2) Driver's age should be 18 or above

```
DELIMITER $$
CREATE  TRIGGER check_date_of_birth
            AFTER INSERT
            ON driver FOR EACH ROW
            BEGIN
              IF ( '2002-01-01'< NEW.Date_of_birth <'2022-12-31')
              THEN
              Delete from driver ;

END IF;
END;
```

3) Payment amount cannot be negative

```
DELIMITER $$
CREATE  TRIGGER pay_out
            AFTER INSERT
            ON payment FOR EACH ROW
            BEGIN
              IF (New.Payment_Amount <0)
              THEN
              Delete from payment ;

END IF;
END;
```

4) Two registered cars cannot have same car number

```
DELIMITER $$
CREATE  TRIGGER SAME_CAR_NO
            AFTER UPDATE
            ON vehicle FOR EACH ROW
            BEGIN
              IF (New.Car_no = Old.Car_no)
              THEN
              Delete from vehicle ;

END IF;
END;
```

SQL Queries:

Query 1: View all Driver name and driver license number of driver having an suv rating between 1 to 10

Sol:

```
Select DISTINCT  Driver_Name, Driver_License_NO,Car_Type
From Driver,Vehicle
Where Car_Type = 'SUV'
```

Query 2: Find TRIP ID AND TRIP STATUS between 2021-04-28 to 2021-06-28

Sol:

```
Select distinct Trip_ID , Trip_Status, Trip_Date_Day
from  trip
where       2021-04-28 <Trip_Date_Day < 2021-06-28;
```

Query 3: Find contact number of driver whose rating is above 7 and current status of driver is not in ride

Sol:

```
Select Distinct Contact_number,Rating,Current_status
from driver
where rating >7 and Current_status ='FALSE';
```

Query 4: Find all drop location of car type sedan where it drop passenger.

Sol:

```
select distinct Drop_Location , Car_Type
From trip, Vehicle
Where Car_Type ='SEDAN';
```

Query 5: Find previous trips of passenger done with the driver currently in ride if any.

Sol:

```
with tp as
(Select * from trip
group by Trip_Passenger_ID
having Trip_Passenger_ID = 0)
```

```
Select tp1.* from tp tp1, tp tp2 where tp2.Trip_Status='FALSE' and
tp1.Trip_Status='TRUE' and tp1.Trip_Driver_ID=tp2.Trip_Driver_ID;
```

Query 6: Find passenger_id for whole passenger who paid over amount 3k using payment type “CASH”

Sol:

```
select Trip_Passenger_ID,sum(Payment_Amount) as Paid
from trip,payment
where Trip_ID=Trip_Id_Pay and Payment_Type='CASH'
group by Trip_Passenger_ID
having paid>3000;
```

Query 7: Find driver who has more than 3 ride who have payment amount >=1000 today

Sol:

```
select Trip_ID,count(*) as Noofrides
from trip,payment
where Trip_Id_Pay=Trip_ID and Payment_Amount>1000
group by Trip_Date_Day
having Trip_Date_Day='2022-04-28' and Noofrides>3;
```

Query 8:Find trip with maximum payment made on particular day .

Sol:

```
select Trip_Date_Day,max(Payment_Amount) as Max_Transaction
from trip,payment
where Trip_Id_Pay=Trip_ID
group by Trip_Date_Day;
```

Query 9: Create a view Dashboard where select drop location and pick up location where as name as reference , contact number and join on username and where request passenger id is equal to request passenger id .

Sol:

```
Create or Replace view Dashboard as
Select b.Pickup_Location,b.Drop_Location,p.Name as RefName,p.Contact_Number as
contactno
From ( Select * from booking join user on username
Where username=Request_Driver_ID) as b join passenger p where
b.Request_Passenger_ID = p.Passenger_ID ;
```

Query 10: Ceate a view manage vehicles where it select driver id as username and and driver name and where driver car number is equal to car no.

Sol:

```
Create view ManageVehicles as Select d.Driver_id
as username,d.Driver_Name as Name,v.car_no,v.car_type,v.car_model
from Driver d,Vehicle v
where d.Driver_id="{}" and d.Driver_Car_Number=v.car_no;
```