User Guide

Cab-Connect

The project aims to build a database management system for a cab booking system. We will design a front-end application with a robust backend database management system using the various DBMS concepts taught in the course CSE202. The project is aimed to create such a database where a person can easily get a vehicle for its transport without any hassle. Data for all the cabs will be stored, making it easy to assign a cab to the customer.

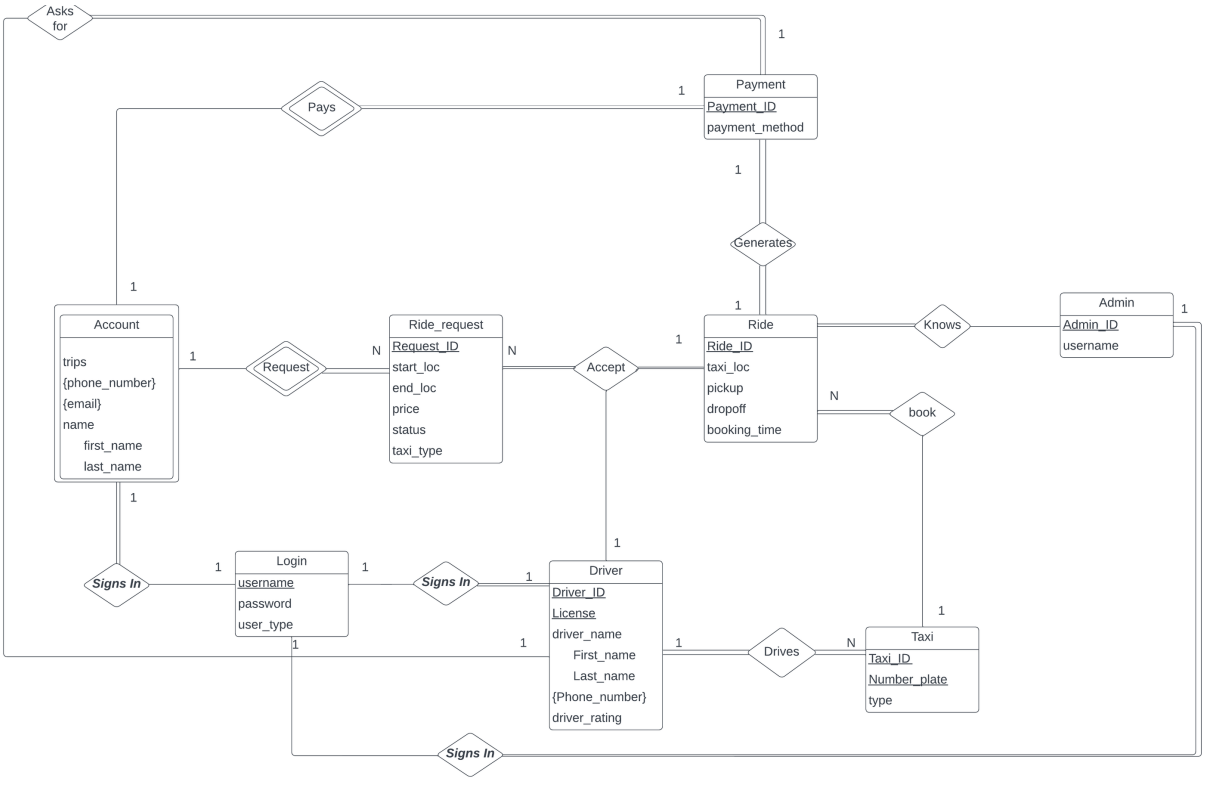
Members:	• **Aditya Bindlish 2021004** • **Aditya Padmagirwar 2020449**
Timeline:	25/01/2023 - 24/04/2023

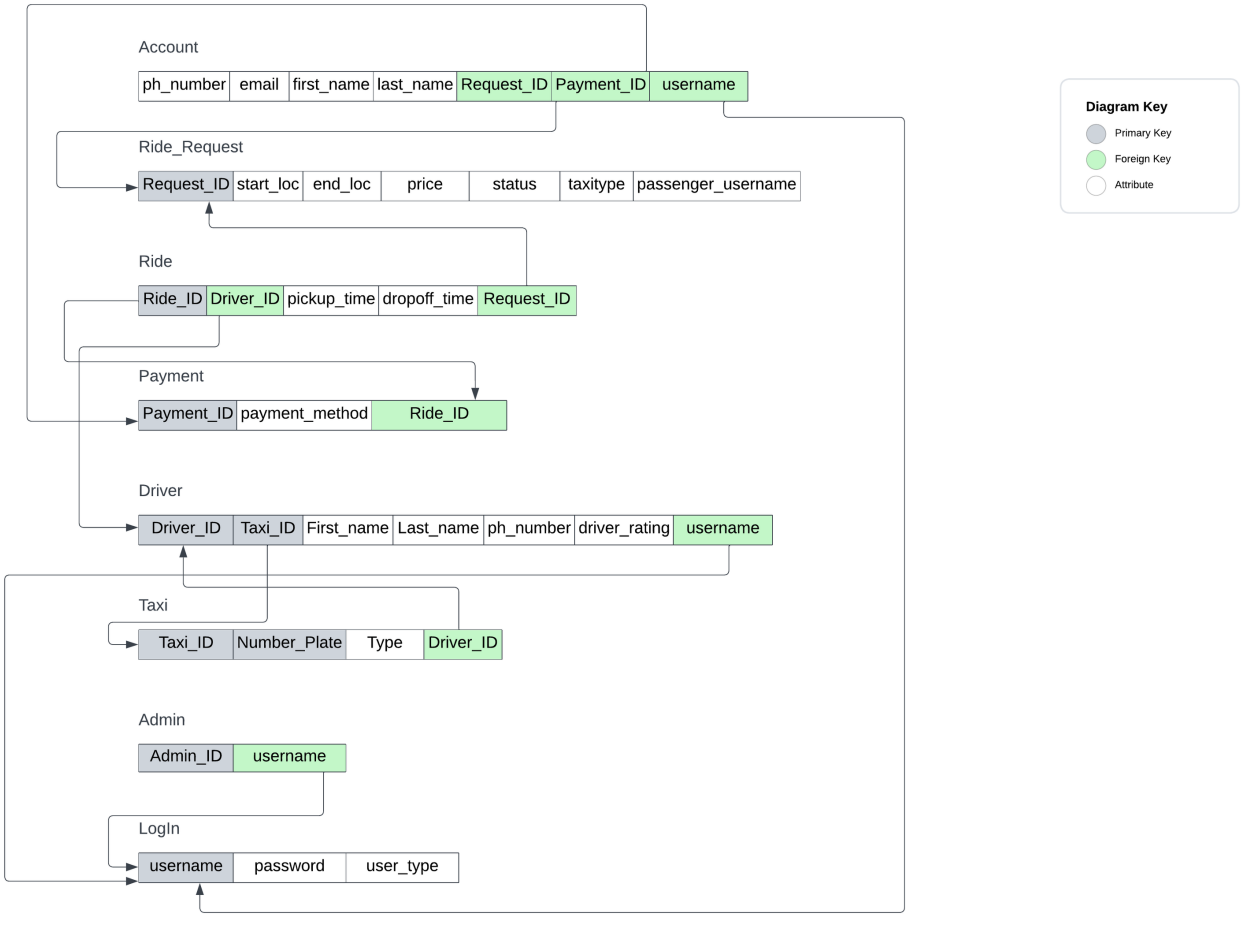
Project Details

Background

Cab-Connect is an online taxi booking service. The user can hire a cab from any location with ease. It tries to tackle the problem of finding an auto or a car to get to meetings. Waiting for auto-rickshaws in the heat. The customer can reserve a car according to their needs through this app from wherever, and the cab will arrive without delay.

ER Diagram and Relational Schema


Asks for
1
Payment
Payment_ID
payment_method
Pays
1
1
Generates
1
1
Account
trips
{phone_number}
{email}
name
first_name
last_name
1
Request
N
Ride_request
Request_ID
start_loc
end_loc
price
status
taxi_type
N
Accept
1
Ride
Ride_ID
taxi_loc
pickup
dropoff
booking_time
Knows
Admin
Admin_ID
username
1
N
book
1
1
Signs In
1
Login
username
password
user_type
1
Signs In
1
Driver
Driver_ID
License
driver_name
First_name
Last_name
{Phone_number}
driver_rating
1
1
1
Drives
N
Taxi
Taxi_ID
Number_plate
type
1
1
Signs In


Account
ph_number
email
first_name
last_name
Request_ID
Payment_ID
username
Ride_Request
Request_ID
start_loc
end_loc
price
status
taxitype
passenger_username
Ride
Ride_ID
Driver_ID
pickup_time
dropoff_time
Request_ID
Payment
Payment_ID
payment_method
Ride_ID
Driver
Driver_ID
Taxi_ID
First_name
Last_name
ph_number
driver_rating
username
Taxi
Taxi_ID
Number_Plate
Type
Driver_ID
Admin
Admin_ID
username
LogIn
username
password
user_type
Diagram Key
Primary Key
Foreign Key
Attribute

All SQL Tables

```
CREATE TABLE `admin` (
  `admin_ID` varchar(10) NOT NULL,
  `username` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`admin_ID`),
  KEY `username` (`username`),
  CONSTRAINT `admin_ibfk_1` FOREIGN KEY (`username`) REFERENCES `login` (`username`)
);

CREATE TABLE `driver` (
  `driver_ID` varchar(10) NOT NULL,
  `username` varchar(10) DEFAULT NULL,
  `f_name` varchar(20) NOT NULL,
  `l_name` varchar(20) NOT NULL,
  `ph_num` varchar(10) NOT NULL,
  `rating` int DEFAULT NULL,
  `taxi_ID` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`driver_ID`),
  UNIQUE KEY `driver_ID` (`driver_ID`),
  KEY `username` (`username`),
  KEY `driver_f_name` (`f_name`),
  CONSTRAINT `driver_ibfk_1` FOREIGN KEY (`username`) REFERENCES `login` (`username`)
);

CREATE TABLE `login` (
  `username` varchar(10) NOT NULL,
  `passwor` varchar(10) DEFAULT NULL,
  `user_type` varchar(10) DEFAULT NULL,
  PRIMARY KEY (`username`),
```

```
  UNIQUE KEY `username` (`username`),
  KEY `user_type` (`user_type`)
) ;

CREATE TABLE `passenger` (
  `username` varchar(10) DEFAULT NULL,
  `f_name` varchar(20) NOT NULL,
  `l_name` varchar(20) NOT NULL,
  `ph_num` varchar(10) NOT NULL,
  `email` varchar(80) DEFAULT NULL,
  `rating` int DEFAULT NULL,
  `request_ID` int DEFAULT NULL,
  KEY `username` (`username`),
  KEY `request_ID` (`request_ID`),
  KEY `passenger_f_name` (`f_name`),
  CONSTRAINT `passenger_ibfk_1` FOREIGN KEY (`username`) REFERENCES `login` (`username`),
  CONSTRAINT `passenger_ibfk_2` FOREIGN KEY (`request_ID`) REFERENCES `ride_request` (`request_ID`)
) ;

CREATE TABLE `payment` (
  `payment_ID` varchar(10) NOT NULL,
  `ride_ID` varchar(10) DEFAULT NULL,
  `payment_method` varchar(10) NOT NULL,
  PRIMARY KEY (`payment_ID`),
  UNIQUE KEY `payment_ID` (`payment_ID`),
  KEY `ride_ID` (`ride_ID`),
  KEY `payment_method` (`payment_method`),
  CONSTRAINT `payment_ibfk_1` FOREIGN KEY (`ride_ID`) REFERENCES `ride` (`ride_ID`)
) ;
```

```
CREATE TABLE `ride` (
  `ride_ID` varchar(10) NOT NULL,
  `request_ID` int DEFAULT NULL,
  `driver_ID` varchar(10) DEFAULT NULL,
  `pickup_time` varchar(10) NOT NULL,
  `drop_time` varchar(10) NOT NULL,
  PRIMARY KEY (`ride_ID`),
  UNIQUE KEY `ride_ID` (`ride_ID`),
  KEY `driver_ID` (`driver_ID`),
  KEY `request_ID` (`request_ID`),
  KEY `pickup_time` (`pickup_time`),
  CONSTRAINT `ride_ibfk_1` FOREIGN KEY (`driver_ID`) REFERENCES `driver` (`driver_ID`),
  CONSTRAINT `ride_ibfk_2` FOREIGN KEY (`request_ID`) REFERENCES `ride_request` (`request_ID`),
  CONSTRAINT `ride_ibfk_3` FOREIGN KEY (`driver_ID`) REFERENCES `driver` (`driver_ID`),
  CONSTRAINT `ride_ibfk_4` FOREIGN KEY (`request_ID`) REFERENCES `ride_request` (`request_ID`)
) ;

CREATE TABLE `ride_request` (
  `request_ID` int NOT NULL,
  `passenger_username` varchar(10) DEFAULT NULL,
  `pickup_loc` varchar(10) NOT NULL,
  `drop_loc` varchar(10) NOT NULL,
  `fare` int NOT NULL DEFAULT '50',
  `taxi_type` varchar(10) DEFAULT NULL,
  `status` varchar(10) NOT NULL,
  PRIMARY KEY (`request_ID`),
  UNIQUE KEY `request_ID` (`request_ID`),
  KEY `taxi_type` (`taxi_type`),
  CONSTRAINT `ride_request_chk_1` CHECK ((`fare` >= 50))
) ;
```

```
CREATE TABLE `taxi` (
 `taxi_ID` varchar(10) NOT NULL,
 `number_plate` varchar(10) NOT NULL,
 `taxi_type` varchar(10) DEFAULT NULL,
 `driver_ID` varchar(10) DEFAULT NULL,
 PRIMARY KEY (`taxi_ID`),
 UNIQUE KEY `taxi_ID` (`taxi_ID`),
 UNIQUE KEY `number_plate` (`number_plate`),
 KEY `driver_ID` (`driver_ID`),
 KEY `taxi_type` (`taxi_type`),
 CONSTRAINT `taxi_ibfk_1` FOREIGN KEY (`driver_ID`) REFERENCES `driver` (`driver_ID`)
) ;
```

10 SQL Queries

Q Find the price of rides with their taxi types

```
select ride.ride_ID, ride.request_ID, ride_request.request_ID, ride_request.fare, ride_request.taxitype
from ride
join ride_request
on ride.request_ID=ride_request.request_ID;
```

Q Find rides with less fare

```
select ride_ID, driver_ID, ride_request.fare
from ride
join ride_request
on ride.request_ID=ride_request.request_ID
order by fare desc;
```

Q Find the username of everyone

```
select username
from passenger
union
select username
from driver
union
select username
from admin
order by username;
```

Q Find passengers with the same rating as drivers

```
select passenger.f_name, passenger.request_ID, driver.driver_id, driver.f_name
from passenger
join driver
on passenger.rating= driver.rating;
```

Q Find passenger and their requests

```
select passenger.f_name, passenger.request_ID, ride_request.taxitype, ride_request.pickup_loc,
ride_request.drop_loc, ride_request.fare
from passenger
join ride_request
on passenger.request_ID=ride_request.request_ID
order by fare desc;
```

Q Find whether the ride is accepted or rejected

```
select passenger.f_name, passenger.request_ID, ride_request.status
from passenger
join ride_request
on passenger.request_ID=ride_request.request_ID
order by fare desc;
```

Q Find passengers with the same pickup and drop location

```
select passenger.f_name, passenger.request_ID, ride_request.pickup_loc, ride_request.drop_loc
from passenger
join ride_request
on passenger.request_ID=ride_request.request_ID
where pickup_loc='A' and drop_loc='D';
```

Q Find all taxis and their ride requests

```
select taxi.taxi_type, taxi.taxi_ID, ride_request.request_ID
from taxi
join ride_request
on taxi_type=taxitype;
```

Q Find the status of all taxis

```
select taxi.taxi_type, taxi.taxi_ID, request_ID, ride_request.status
from taxi
join ride_request
on taxi_type=taxitype;
```

Q Find drivers and their taxis

```
select taxi.taxi_ID, taxi.taxi_type, driver.driver_ID
from taxi
join driver
on taxi.taxi_ID= driver.taxi_ID;
```

OLAP Queries

Q1 Find the total number of rides for each driver and taxi type and the payment method

```
SELECT
   Driver.driver_id,
   Driver.f_name,
   Driver.l_name,
   Ride_request.taxitype,
   Payment.payment_method,
   COUNT(*) AS num_rides
FROM Driver
   JOIN Taxi ON Driver.driver_id = Taxi.driver_id
   JOIN Ride ON Driver.driver_id = Ride.driver_id
   JOIN Ride_request ON Ride.request_id = Ride_request.request_id
   join payment on ride.ride_ID= payment.ride_ID
GROUP BY Driver.driver_id, Driver.f_name, Driver.l_name, Ride_request.taxitype, Payment.payment_method
WITH ROLLUP;
```

Q2 Find the total number of completed rides and the total amount earned by each driver

```
SELECT driver.f_name, driver.l_name, COUNT(*) as num_rides, SUM(fare) as total_earnings
FROM ride JOIN driver ON ride.driver_id = driver.driver_id
JOIN ride_request ON ride.request_id = ride_request.request_id
GROUP BY driver.f_name, driver.l_name with rollup
UNION
SELECT 'Total', '', COUNT(*) as num_rides, SUM(fare) as total_earnings
FROM ride_request
WHERE status = 'Accepted';
```

Q3 Find the number of rides for each taxi type and driver rating, including subtotals for each taxi type and a grand total

```
SELECT ride_request.taxitype,  Driver.rating,  COUNT(*) AS num_rides
FROM Driver
   JOIN Taxi ON Driver.driver_id = Taxi.driver_id
```

```
   JOIN ride ON ride.driver_ID = Taxi.driver_ID
   JOIN Ride_request ON Ride_request.request_id = Ride.request_id
GROUP BY  Ride_request.taxitype, driver.rating with rollup;
```

Q4 Find the number of passengers by their requested taxi type and whether the request is accepted or rejected

```
SELECT user_type, taxitype, status, COUNT(*) AS count
FROM login
   JOIN Ride_request ON login.username = Ride_request.passenger_username
GROUP BY  user_type, taxitype, status with rollup
UNION
SELECT user_type, taxitype, status, COUNT(*) AS count
FROM login
   JOIN Ride_request ON login.username = Ride_request.passenger_username
GROUP BY user_type, taxitype, status
ORDER BY user_type, taxitype, status;
```

Triggers

```
-- 1 Whenever a new passenger is signed up, a message is shown 'added new passenger'
DELIMITER $$
create trigger trigger_18 before insert
on login
for each row begin
insert into trigger_test values('added new passenger');
end $$; DELIMITER ;
insert into login values('p51', 123, 'passenger');
select*from trigger_test;

-- 2 Whenever there's an update on the ride request table, the trigger checks whether the far is above 220. If not above 220, then the status is set to 'Rejected'
DELIMITER $$
```

```
CREATE TRIGGER ride_request_farecheck

AFTER UPDATE ON Ride_request FOR EACH ROW

BEGIN

 IF NEW.fare < 220 THEN

   UPDATE Ride_request

   SET status = 'Rejected'

   WHERE request_id = NEW.request_id;

 END IF;

END$$

DELIMITER ;
```

Transactions

Non-conflicting Transactions:

```
--1

START TRANSACTION;

INSERT INTO Ride_request (request_id, passenger_username, pickup_loc, drop_loc, fare, taxitype, status)

VALUES (51, 'p51', 'A', 'C', 400, 'Sedan', 'Pending');

UPDATE Ride_request SET status = 'Rejected' WHERE request_id = 51;

COMMIT;


-- 2

START TRANSACTION;

UPDATE Driver SET f_name = 'Johnny', l_name = 'Doe' WHERE driver_id ='D12';

UPDATE Driver SET rating =rating + 1 WHERE driver_id = 'D12';

COMMIT;


-- 3
```

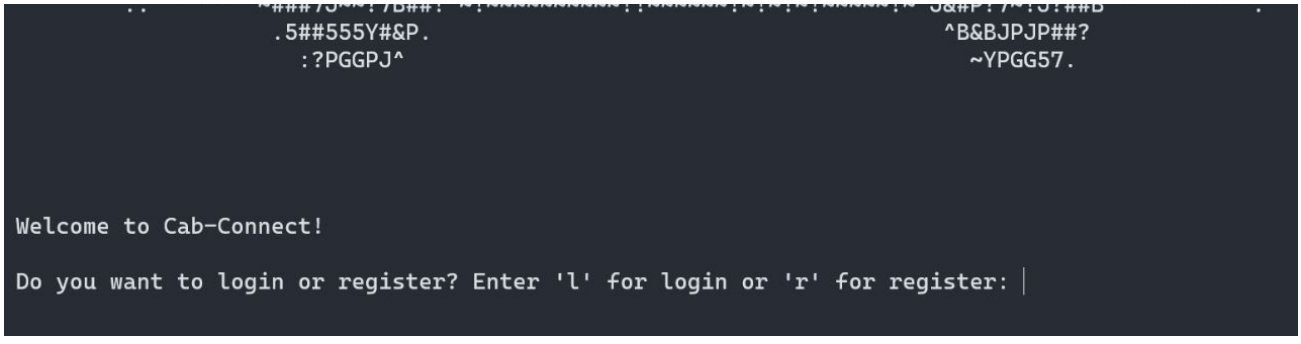
```
start transaction;

-- Delete a ride

delete from payment where ride_id='R40';

DELETE FROM Ride WHERE ride_id = 'R40';

-- Update the ride request status

UPDATE Ride_request SET status = 'Rejected' WHERE request_id = 40;

COMMIT;


-- 4

-- new driver is logged in

insert into login(username, passwor, user_type)

values('d41', 123, 'driver');

-- new taxi is assigned to the driver

insert into taxi (taxi_id, number_plate, taxi_type)

values ('T41', 'HR20', 'XL');

-- driver gives information

insert into driver (driver_id, username, f_name, l_name, ph_num, rating, taxi_id)

values ('D41', 'd41', 'Hira', 'Lal', 2397496326, 4, 'T41');

-- the new driver is given a ride

INSERT INTO Ride (ride_id, request_ID, driver_ID, pickup_time, drop_time)

VALUES ('R41', 41, 'D41', '7mins', '22mins');

-- status of the ride is changed as the ride is given to a new driver

UPDATE Ride_request SET status = 'Accepted' WHERE request_id = 51;

commit;
```

Conflicting Transactions

```
-- 1

start transaction;
```

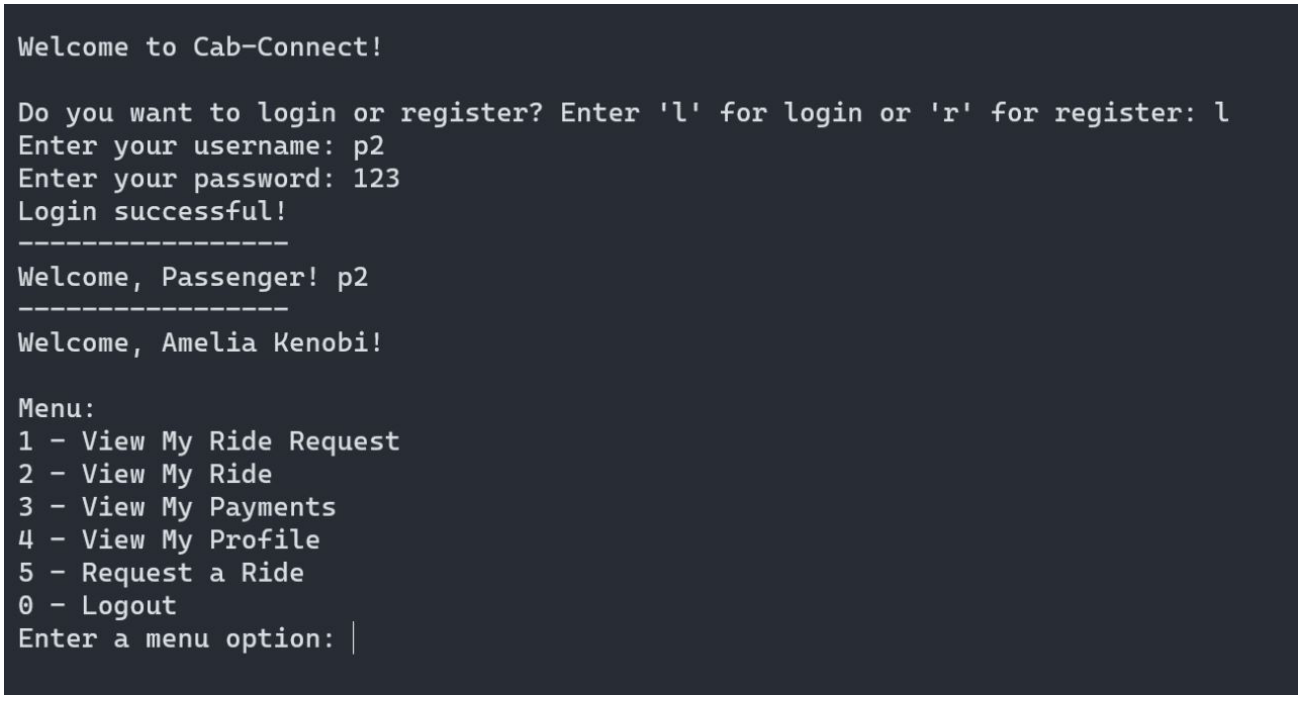
```
UPDATE ride_request SET status = 'Rejected' WHERE request_id= 32;

UPDATE driver SET rating = rating - 1 WHERE driver_id = 'D32';

COMMIT;

start transaction;

UPDATE driver SET rating = rating + 1 WHERE driver_id = 'D32';

UPDATE ride_request SET status = 'Accepted' WHERE request_id = 32 ;

COMMIT;


-- 2

start transaction;

UPDATE passenger SET f_name = 'Alice' WHERE username = 'p11';

UPDATE ride_request SET drop_loc = 'C' WHERE request_id = 11;

COMMIT;

start transaction;

UPDATE ride_request SET drop_loc = 'B' WHERE request_id = 11;

UPDATE passenger SET f_name = 'Bob' WHERE username = 'p11';

COMMIT;
```

Flow OF The CLI



This is the beginning page. The person can either log in or register as a new user.

LOGIN AS A PASSENGER:



- View My Ride Request: The passengers can view the requests he/she has made.
- View My Ride: The passenger can view the rides he/ she has requested for.
- View My Payments: The passenger can view the payments done for the rides.
- View My Profile: The user can view and update his/her data in the profile.
- Request A Ride: The passenger can request a ride for him/herself.
- Logout: The user can log out from the app.

LOGIN AS A DRIVER:

- View My Ride: The driver can view the rides he/she has.
- View My Taxi: The driver can view the taxi details he/she owns.
- Find A Ride: The driver can look for ride requests according to the type of vehicle he owns.

```
2 - View My Taxi
3 - View My Profile
4 - Find a Ride
0 - Logout
Enter a menu option: 4
There are no pending ride requests.
```

LOGIN AS AN ADMIN:

```
Welcome to Cab-Connect!

Do you want to login or register? Enter 'l' for login or 'r' for register: l
Enter your username: a1
Enter your password: 123
Login successful!
-----------------
Welcome, Admin! a1

Menu:
1 - View Drivers
2 - View Admins
3 - View Passengers
4 - View Rides
5 - View Ride Requests
6 - View Payments
7 - Custom Query
8 - Ten Queries
9 - Run Transactions
0 - Logout
Enter a menu option:
```

- View Drivers: The admin can access all the present drivers.
- View Admins: The admin can see all the other available admins.
- View Passengers: Admin can view all the passengers.
- View Rides: The admin can view all the present rides.
- View Ride Requests: The admin can see all the ride requests.
- View Payments: The admin can see all the payments done by the passengers.
- Custom Query: This can show any other data the admin wants to access.

- Ten Queries: These are the top ten queries for the admins.
- Run Transactions: The admins can run transactions.

REGISTERING AS A NEW USER:

```
Do you want to login or register? Enter 'l' for login or 'r' for register: r
Are you a driver, passenger or admin? Enter 'd' or 'D' for driver, 'p' or 'P' for passenger, and 'a' or 'A' for admin: d
Enter your first name: Henry
Enter your last name: John
Enter your phone number: 4568356343
Enter your number plate: JK67
Enter your taxi type: Go
Enter your password: 123
Your Username is the Driver_ID:  D64
Ok, you are a driver.
Registration successful!
Please log in to your new account.
Enter your username:
```