ECIS 637T – Database/Big Data Mgmt

Mobility Project

Submitted by:
Karuna Upadhyaya

Fall Semester, 2023
Christian Brothers University

Application Design Requirements

1. User Registration – User can create an account using the registration form. It asks First Name, Last Name, Phone Number, Email, Password, Address, City, State and Zip to create an account.

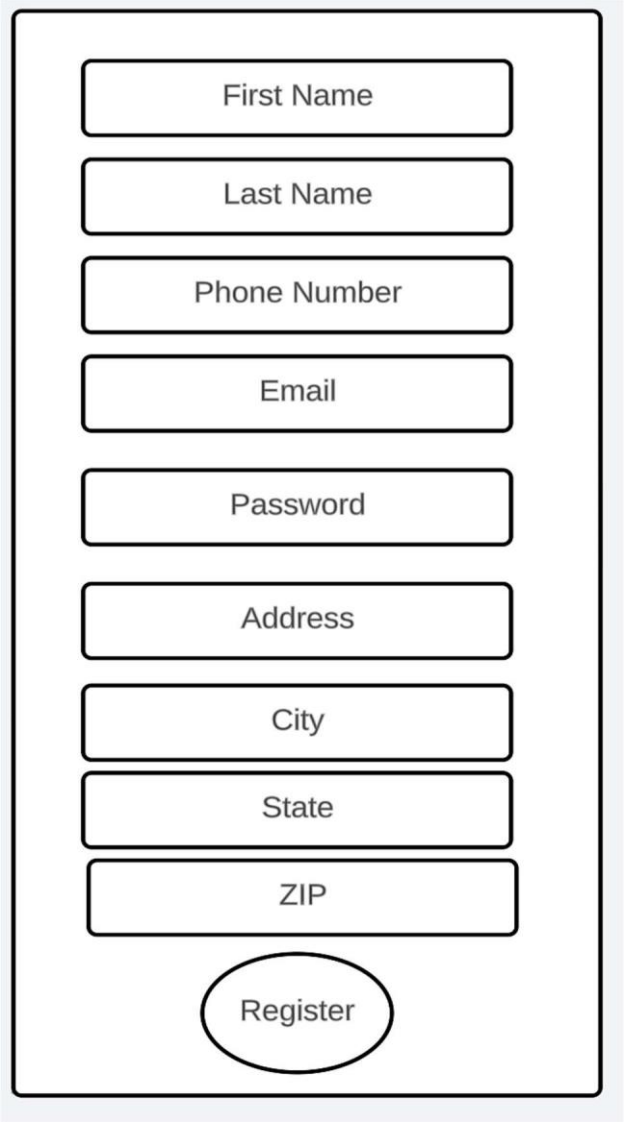

Fig 1 User Registration

2. Driver Registration – New driver can create an account using the registration page. It takes First Name, Last Name, Phone Number, Email, Password, Address, City, State, Zip, Driver License Number, Car Number to create an account.

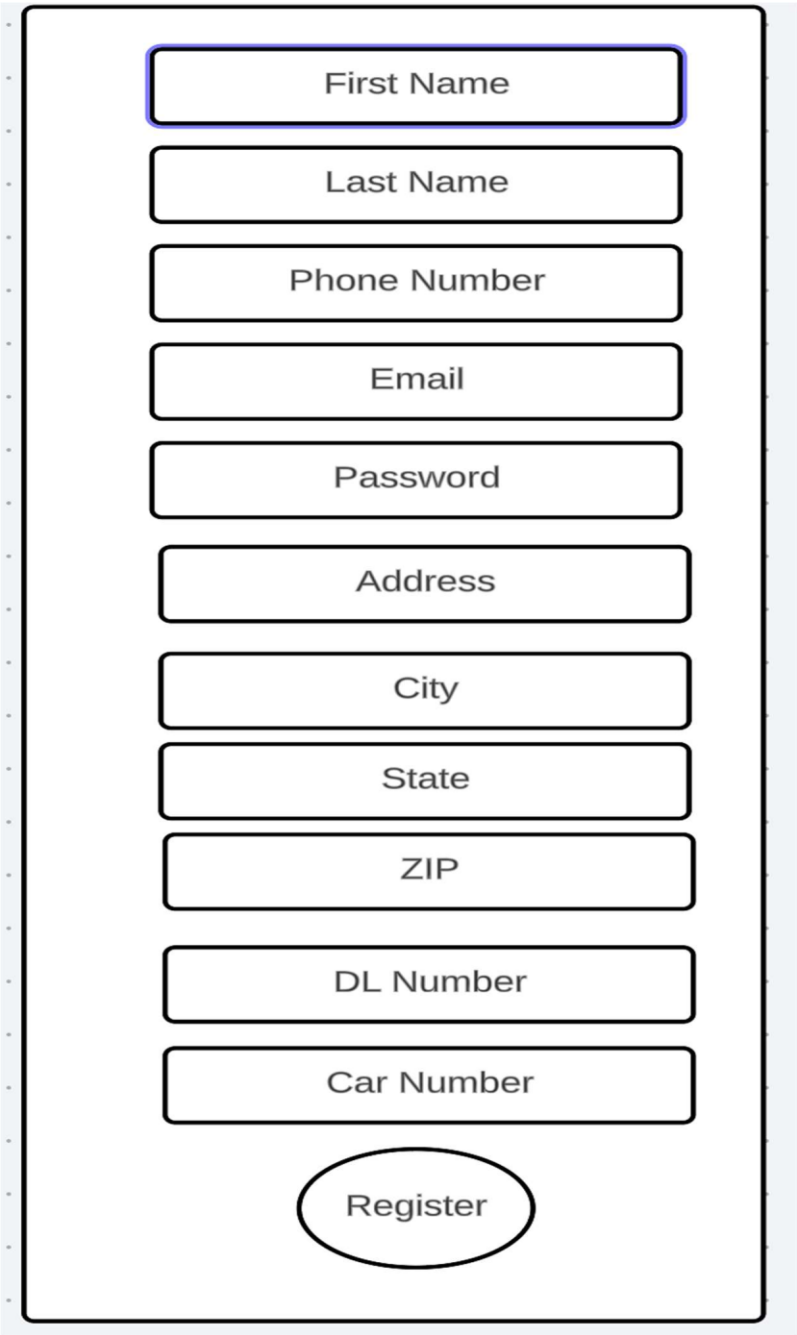

Fig 2 Driver Registration

3. Login Page – User or Driver can with account using the phone number or Gmail account.

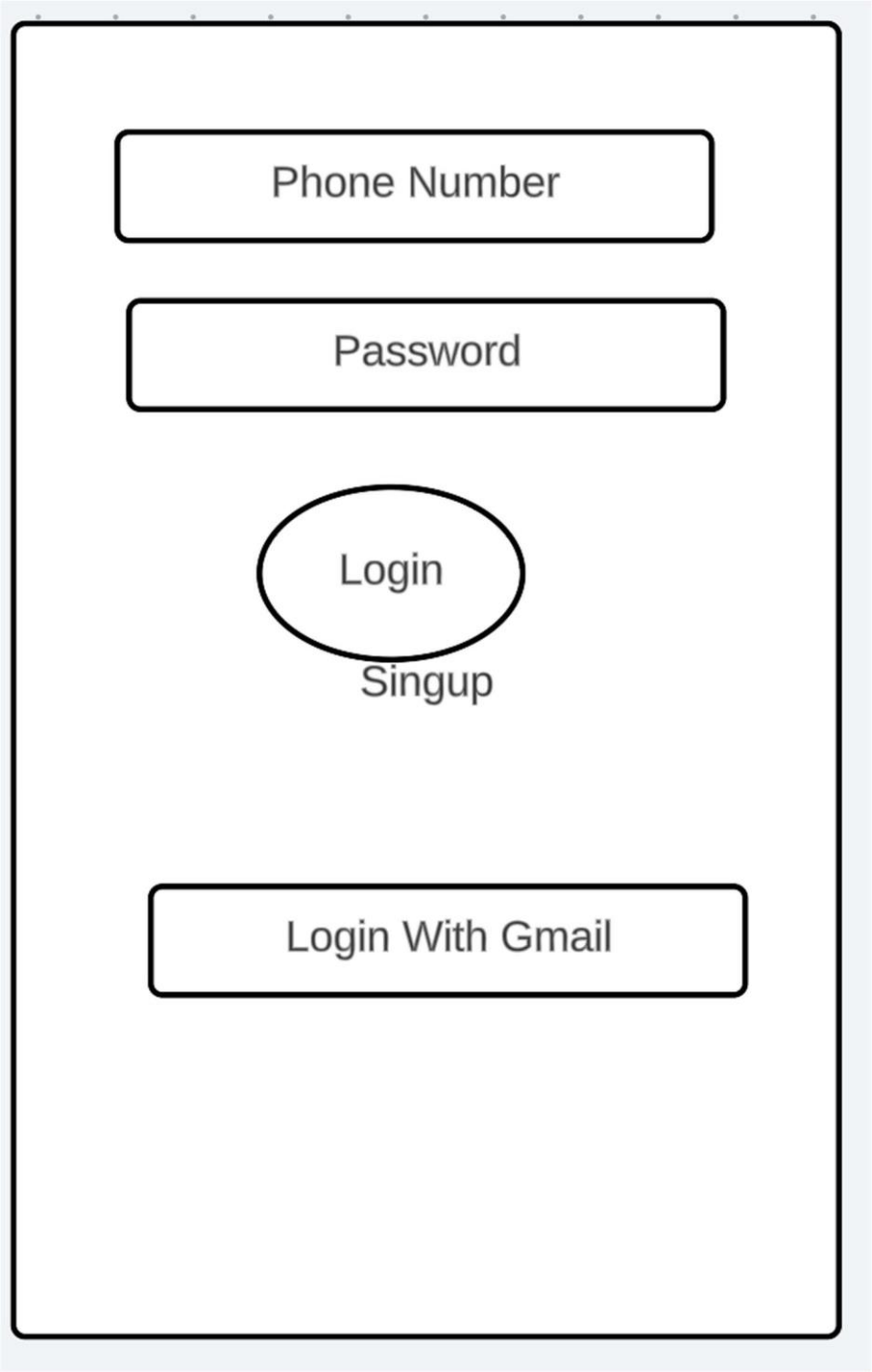

Fig 3 Login Page

4. User Profile Change Page – User can change his details using the profile change page. He can change his profile picture, address, city, zip, phone number, email, and password.

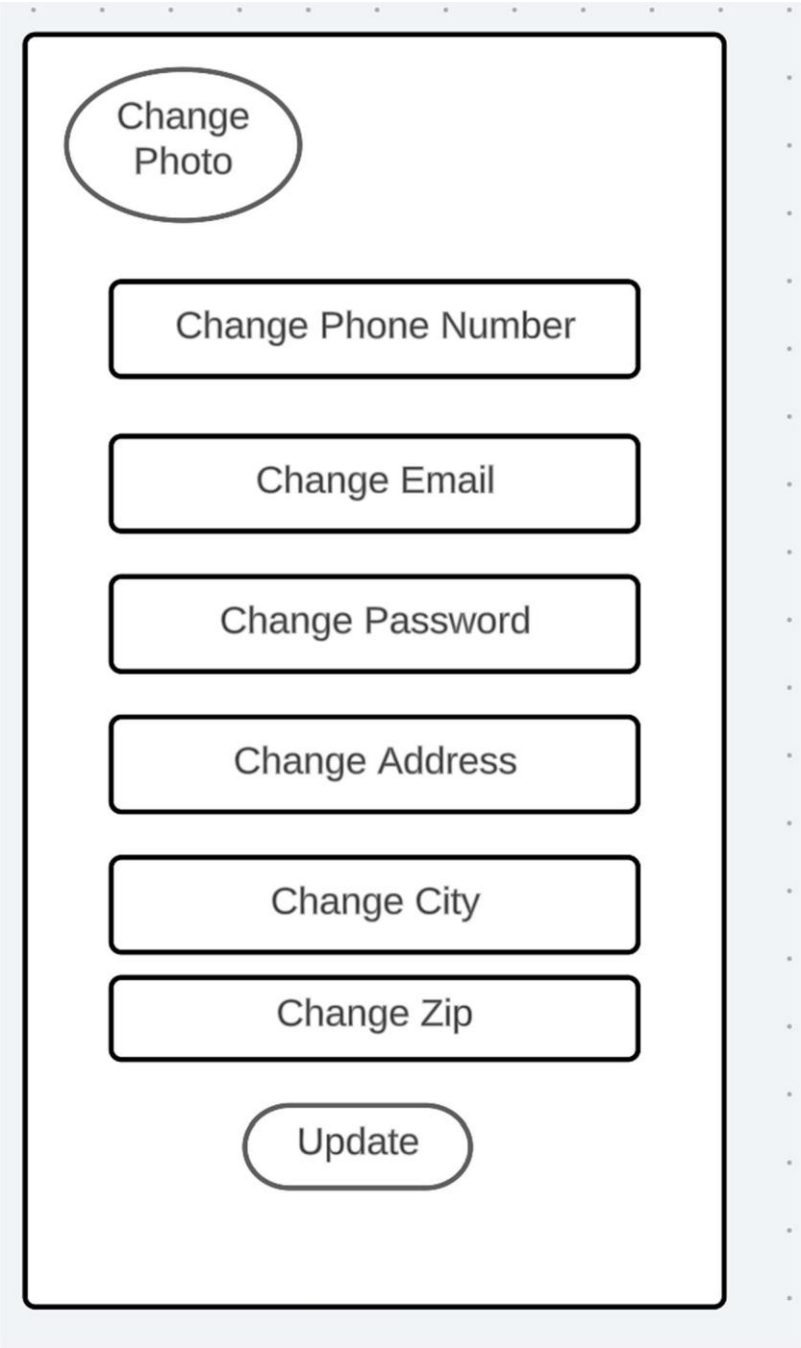

Fig 4 User Profile Change Page

5. Driver Profile Change Page – Driver can change his details using the profile change page.

He can change his profile picture, address, city, zip, phone number, email, and password.

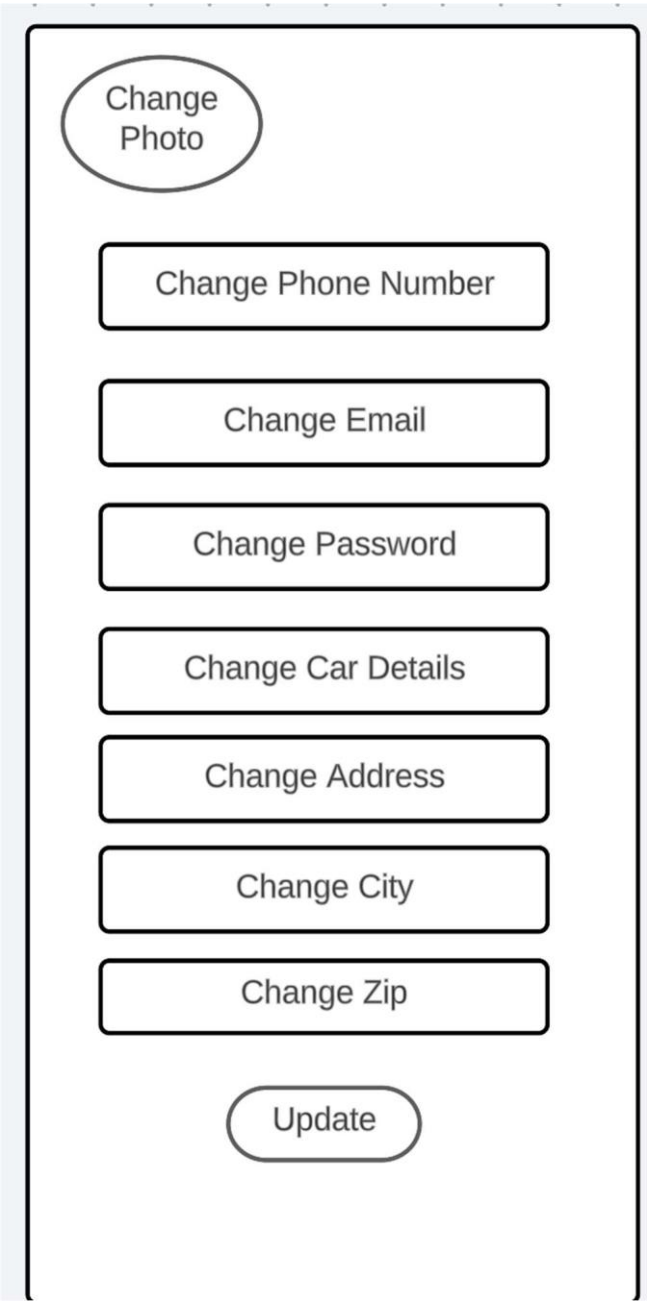

Fig 5 Drive Profile Change Page

6. History Page – User and Driver can see previous rides on the history page. Here they can see price, duration, and location details of the ride.

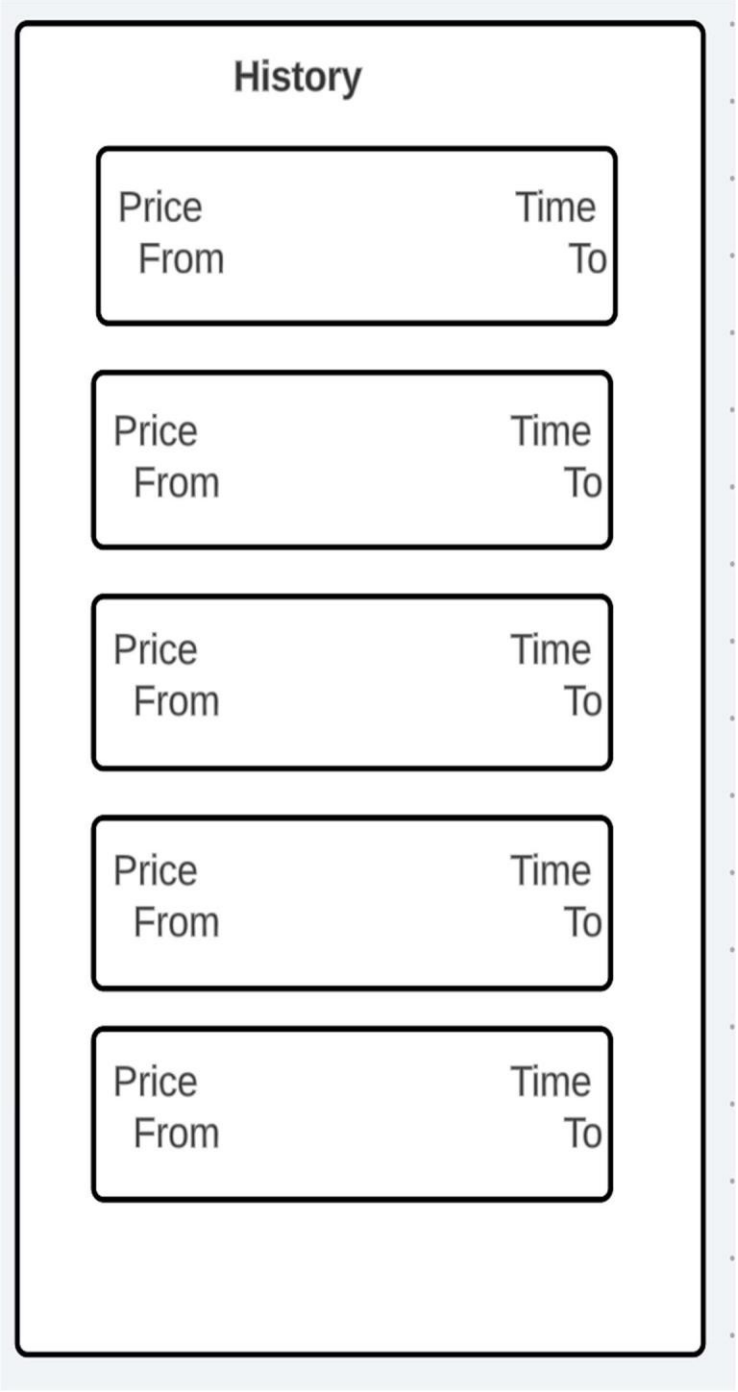

Fig 6 History Page

7. Drivers Payment Page – Drive can see his earnings and he can see option to withdraw that amount.

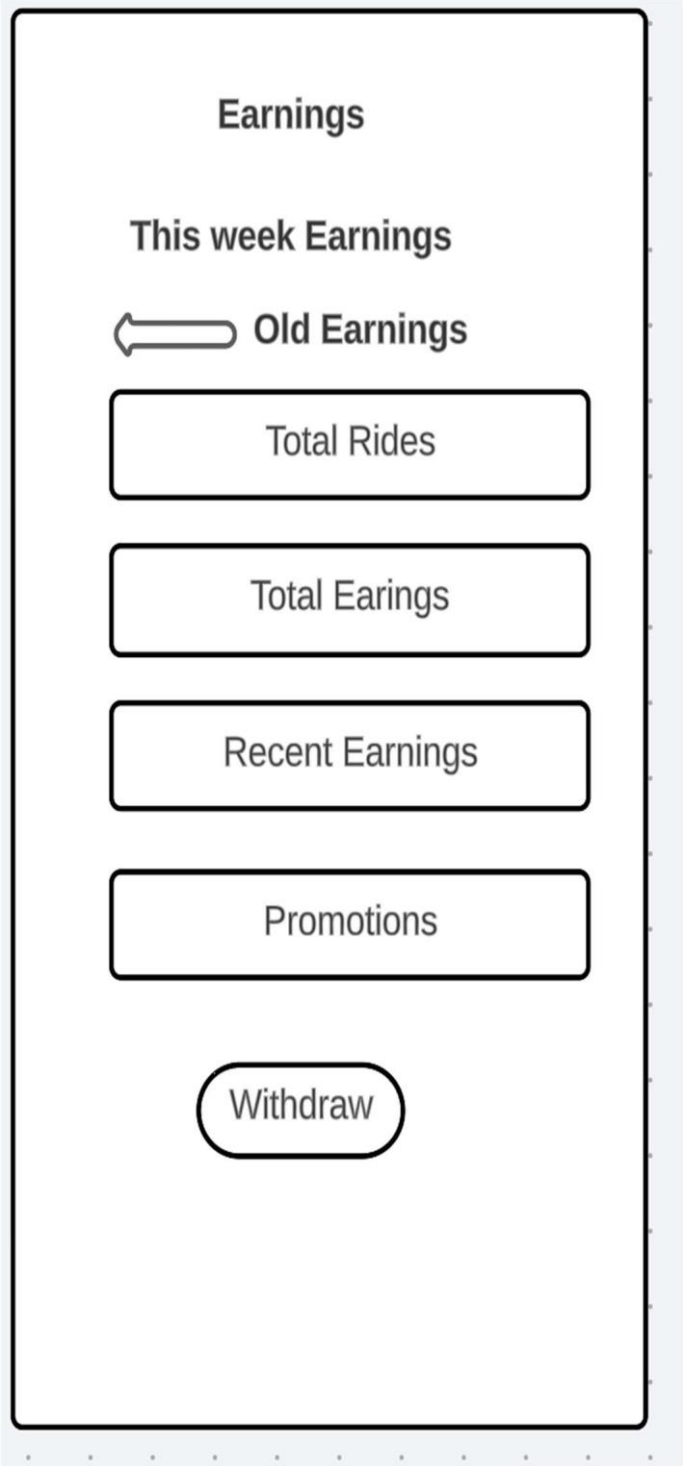

Fig 7 Drivers Payment Page

8. Customers Payment Page – Customers can see already added payment methods on payments page. He can also add new payment methods or promotional codes on this page.

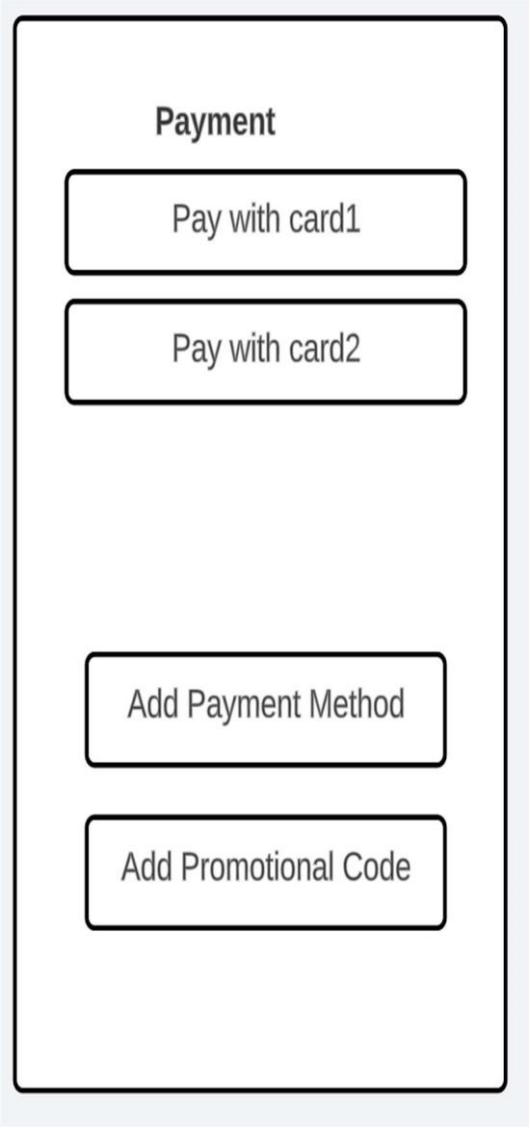

Fig 8 Customers Payment Page

Design Requirements

1. ER Diagram:

Entity Relationship (ER) diagram[1] shows the relationship between tables in a database. It helps us to understand how many tables in database, how many columns in each table, primary keys and foreign keys also.

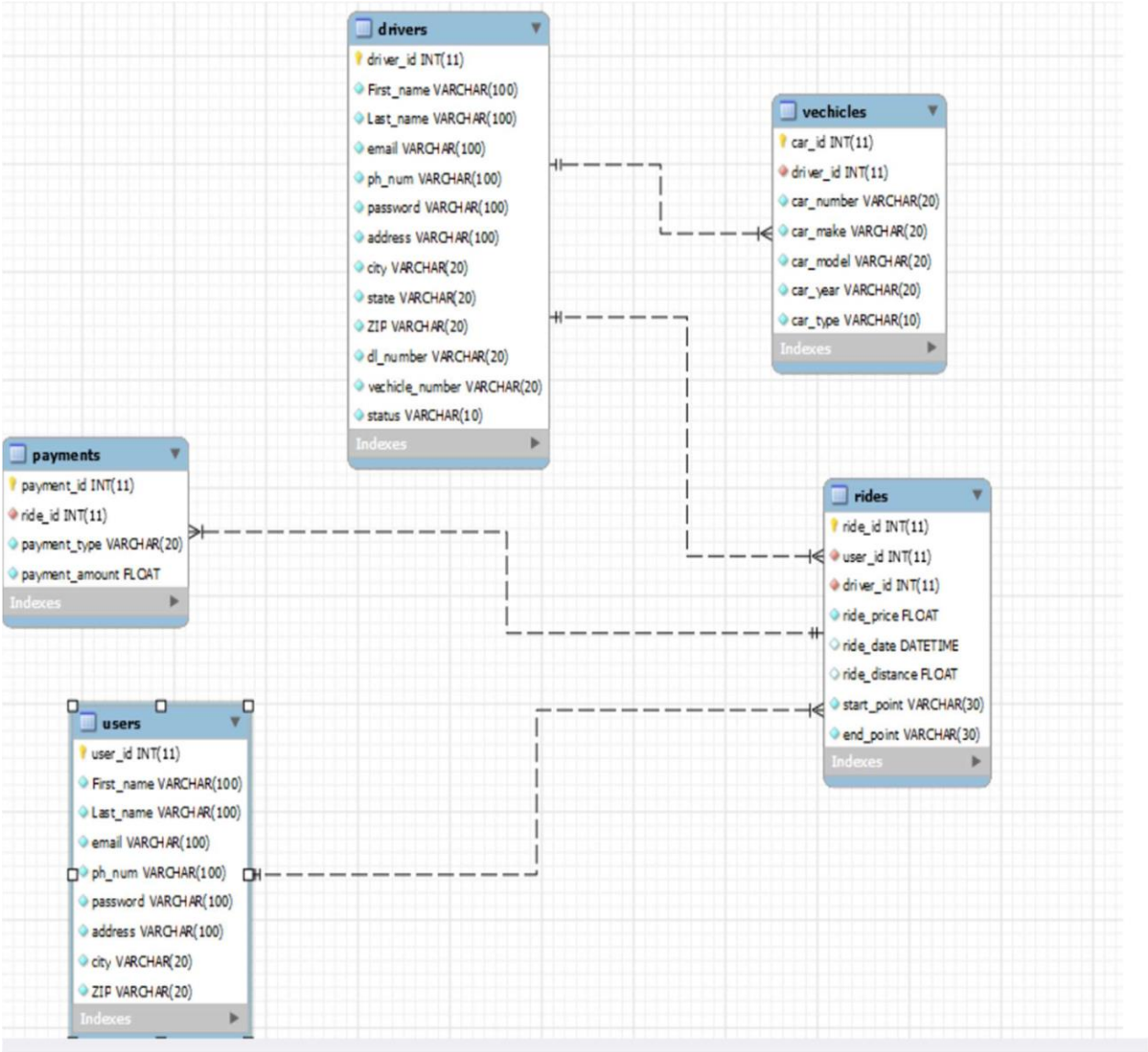

Fig 9 ER Diagram

2. Database & Tables:

We created new database with name 'Mobility_Project' using the following query.

```
Create database Mobility_Project;
```

And we used this database for our project. And created different tables like Users, Drivers, Vehicles, Rides and Payments.

```
use Mobility_Project;
```

1 Users Table – Users table is used to store user information. It has First Name, Last Name, Phone Number, Email, Password, Address, City, State and Zip columns.

```
CREATE TABLE `users` (
  `user_id` int NOT NULL PRIMARY KEY ,
  `First_name` varchar(100) NOT NULL,
  `Last_name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `ph_num` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `address` varchar(100) NOT NULL,
  `city` varchar(20) NOT NULL,
  `state` varchar(20) NOT NULL,
  `ZIP` varchar(20) NOT NULL
);
```

2 Drivers Table – Drives table is used to store the driver's information. It has First Name, Last Name, Email, Phone Number, Password, Address, City, State, Zip, DL Number and Car Number columns.

```sql
CREATE TABLE `drivers` (
  `driver_id` int PRIMARY KEY NOT NULL,
  `First_name` varchar(100) NOT NULL,
  `Last_name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `ph_num` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `address` varchar(100) NOT NULL,
  `city` varchar(20) NOT NULL,
  `state` varchar(20) NOT NULL,
  `ZIP` varchar(20) NOT NULL,
  `dl_number` varchar(20) NOT NULL,
  `vechicle_number` varchar(20) NOT NULL,
  `status` varchar(10) NOT NULL
);
```

3 Payments Table – Payments table is used to store the payment information. It has columns to store Payment ID, Ride ID, Payment Type and Payment amount information.

```sql
CREATE TABLE `payments` (
  `payment_id` int NOT NULL PRIMARY KEY,
  `ride_id` int NOT NULL,
  `payment_type` varchar(20) NOT NULL,
  `payment_amount` float NOT NULL
);
```

4 Rides Table – Rides table is used to store the rides information. It has Ride ID, User ID, Driver ID, Ride Price, Ride Date, Ride Distance, Start Point and End Point columns.

```sql
CREATE TABLE `rides` (
  `ride_id` int NOT NULL PRIMARY KEY,
  `user_id` int NOT NULL,
  `driver_id` int NOT NULL,
  `ride_price` float NOT NULL,
  `ride_date` datetime DEFAULT NULL,
  `ride_distance` float DEFAULT NULL,
  `start_point` VARCHAR(30) NOT NULL,
  `end_point` VARCHAR(30) NOT NULL
);
```

5 Vehicles Table – Vehicles table is used to store Car ID, Driver ID, Car Number, Car Make, Car Model, Car Year and Car Type information.

```
CREATE TABLE `vechicles` (
    `car_id` int NOT NULL PRIMARY KEY,
    `driver_id` int NOT NULL,
    `car_number` varchar(20) NOT NULL,
    `car_make` varchar(20) NOT NULL,
    `car_model` varchar(20) NOT NULL,
    `car_year` varchar(20) NOT NULL,
    `car_type` varchar(10) NOT NULL
) ;
```

Normalization:

Normalization in SQL reduces data redundancy. Normalization techniques are used to divide the complex table into smaller tables. All our tables are already normalized. [2]

3. Primary Keys & Foreign Keys:

Primary key[3] is a column used in a table which is used to identity the data of a database. Foreign key is a column which references a column of another table. We created primary keys for all tables using PRIMARY KEY when we created the tables. The following are primary keys:

Table Name	Primary Key Column Name
Users	user_id
Drivers	driver_id
Vechicles	car_id
Payments	payment_id
Rides	ride_id

And we created the following Foreign Keys:

```
ALTER TABLE rides
ADD FOREIGN KEY (driver_id) REFERENCES drivers(driver_id);

ALTER TABLE rides
ADD FOREIGN KEY (user_id) REFERENCES users(user_id);

ALTER TABLE payments
ADD FOREIGN KEY(ride_id) REFERENCES rides(ride_id);

ALTER TABLE vechicles
ADD FOREIGN KEY(driver_id) REFERENCES drivers(driver_id);
```

4. Insert Test Data:

Inserted the test data to ‘users’ table using the following query.

```
INSERT into users values(
1, 'brad', 'pit', 'bradp@gmail.com', '9999101222', 'abc123', '142 Winchister Rd', 'Memphis',  'TN','38111'),
(2, 'Alex', 'Law', 'alex123@gmail.com', '99991113333', 'abcd1234', '300 Summer Ave', 'Memphis',  'TN','38111'),
(3, 'Josh','Lee', 'jlee@gmail.com','9898123143', 'abc123de', '400 N Germnanton Pakrway', 'Memphis', 'TN', '38016'),
(4, 'Jacob', 'Cruine', 'jcruine@gmail.com', '9999332222', 'jcruine23', '194Winchister Rd', 'Memphis',  'TN', '38111'),
(5, 'Krishna', 'Jampana', 'kjampana@gmail.com', '9129457811', 'kjampanna34', '310 Summer Ave', 'Memphis',   'TN','38111'),
(6, 'Mohammed','N', 'nmohammed@gmail.com','9898143124', 'nmohammed123de', '454 N Germnanton Pakrway', 'Memphis',  'TN','38016'),
(7, 'Sam', 'abc', 'sama@gmail.com', '9992211233', 'samabc', '14 Park Avenue', 'Memphis',  'TN','38017'),
(8, 'Vasthav', 'd', 'dvasthav@gmail.com', '98822113333', 'vasthav234', '320 Walnut Grove Rd', 'Memphis',  'TN','38111'),
(9, 'Kein','L', 'lkevin@gmail.com','91238143124', 'lkevin123de', '445 Summer Avenue', 'Memphis',  'TN','38016'),
(10, 'David','Jhonson', 'djhonson@gmail.com','912643124', 'david123de', '423 N Germnanton Pakrway', 'Memphis',  'TN','38016'),
(11, 'Tevin','Shaw', 'tshaw@gmail.com','91238143124', 'tshaw12', '423 Dexter Lake', 'Memphis', 'TN', '38016');
```

Inserted the test data to ‘drivers’ table using the following query.

```
INSERT into drivers values(
1, 'Murthy', 'Jampana', 'mjampana@gmail.com', '9123045122', 'mj123$', '123 Popular Rd', 'Memphis', 'TN', '38111', 'L3123591', 'TN123A', 'YES'
(2, 'Marcus', 'Lee', 'marcuslee@gmail.com', '9123457122', 'abc123$', '45 Popular Rd', 'Memphis', 'TN', '38111', 'L3123591', 'TN123A',  'NO'),
(3, 'Ron', 'Smith', 'ron123@gmail.com', '91781457122', 'xyz123$', '1234 Summer Ave', 'Memphis', 'TN', '38111', 'M3123591', 'TN789',  'YES'),
(4, 'Chirs', 'Barrow', 'chris122@gmail.com', '5412290123', 'abcd123$', '595 Addison St', 'Memphis', 'TN', '38107', 'L561891', 'TN567', 'YES')
(5, 'James', 'K', 'kjames@gmail.com', '9323457122', 'xyz456$', '598 Popular Rd', 'Memphis', 'TN', '38111', 'L3123571', 'TN423A0',  'Yes'),
(6, 'Oliver', 'B', 'boliver@gmail.com', '91781457132', '123$', '14 Summer Ave', 'Memphis', 'TN', '38111', 'L3233591', 'TN146711', 'NO'),
(7, 'Kim', 'Marcus', 'kmarcus122@gmail.com', '5415280153', 'abcd123$', '199 Addison St', 'Memphis', 'TN', '38107', 'J561891', 'TN56778', 'YES
(8, 'Henry', 'Jackson', 'henryjackson@gmail.com', '91375567122', 'henry123$', '15 Popular Rd', 'Memphis', 'TN', '38111', 'U313591', 'TN123A',
(9, 'Williams', 'Comax', 'williams123@gmail.com', '91481457122', 'will123$', '1431 Summer Ave', 'Memphis', 'TN', '38111', 'T315491', 'TN8919'
(10, 'Praveen', 'K', 'kpraveen123@gmail.com', '9132490123', 'praveen123$', '23rd Addison St', 'Memphis', 'TN', '38107', 'L581791', 'TN56812',
(11, 'Alexandar', 'P', 'palex@gmail.com', '9126757122', 'abc123$', '459 Popular Rd', 'Memphis', 'TN', '38111', 'L3223591', 'TN123A45', 'NO'),
(12, 'Derek', 'Sam', 'sderek@gmail.com', '9178647122', 'xyz123$', '34 Summer Ave', 'Memphis', 'TN', '38111', 'M3143591', 'TN78909', 'NO'),
(13, 'Kumar', 'A', 'kumara22@gmail.com', '6452901123', 'akumar123$', '55 Highland St', 'Memphis', 'TN', '38107', 'L591891', 'TN15677', 'YES')
(14, 'Karuna', 'U', 'karunau12@gmail.com', '5140211123', 'karuna11$', '123 Park Avenue', 'Memphis', 'TN', '38107', 'L612891', 'TN17677', 'YES
```

Inserted the test data to vechicles using the following query.

```
INSERT into vechicles values(
1, 1,'TN123A', 'Toyota', 'Camry', '2022', 'Sedan'),
(2, 2,'TN132A', 'Honda', 'CRV', '2020', 'SUV'),
(3, 3, 'TN7891', 'Nissan', 'Roughe', '2021', 'SUV'),
(4, 4, 'TN5677', 'Honda', 'Accord', '2022', 'Sedan'),
(5, 5, 'TN423A0', 'Nissan', 'Roughe', '2021', 'SUV'),
(6, 6, 'TN16711', 'Nissan', 'Roughe', '2020', 'SUV'),
(7, 7, 'TN56778', 'Honda', 'Odessy', '2022', 'SUV'),
(8, 8, 'TN123A', 'Nissan', 'Maxima', '2021', 'Sedan'),
(9, 9, 'TN8919', 'Honda','Civic', '2020', 'Sedan'),
(10, 10, 'TN56812', 'Honda', 'Pilot', '2021', 'SUV'),
(11, 11, 'TN1245E', 'Hyundai', 'Sonata', '2021', 'Sedan'),
(12, 12, 'TN78909', 'Nissan', 'Altima', '2022', 'Sedan'),
(13, 13, 'TN15677', 'Honda0', 'Accoard', '2022', 'Sedan'),
(14, 14, 'TN17677', 'Toyota', 'Fortuner', '2023', 'SUV');
```

Inserted the test data into rides using the following query.

```
INSERT into rides values(
1, 2, 3, '14.97', '2022-01-23 9:30:00', '8', '1498 N Germnanton Parkway', '650 E Pakrway'),
(2, 1, 2, '23.31', '2022-1-24 11:15:00', '15', '262 Danny Thomas Pl', '6498 Summer Ave'),
(3, 2, 1, '14.23', '2022-01-24 12:09:00', '12', '2300 Summer Ave', '380 Market Blvd'),
(4, 7, 4, '14.97', '2022-02-25 9:30:00', '18', '1418 N Germnanton Parkway', '670 E Pakrway'),
(5, 5, 3, '18.31', '2022-02-25 11:15:00', '19', '262 Summer Avenue', '6498 Summer Ave'),
(6, 2, 4, '12.23', '2022-02-25 12:09:00', '14', '1700 Park Ave', '380 Highland St'),
(7, 1, 9, '17.31', '2022-02-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(8, 5, 3, '17.23', '2022-02-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(9, 3, 9, '17.31', '2022-02-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(10, 5, 3, '17.23', '2022-03-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(11, 11, 3, '18.31', '2022-03-25 11:15:00', '19', '262 Danny Thomas Pl', '6498 Summer Ave'),
(12, 8, 4, '12.23', '2023-03-25 12:09:00', '14', '192 Summer Ave', '318 Highland St'),
(13, 5, 9, '14.31', '2023-03-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(14, 4, 3, '18.23', '2023-03-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(15, 7, 9, '21.31', '2023-03-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(16, 8, 3, '25.23', '2023-01-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(17, 5, 9, '19.31', '2023-02-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(18, 2, 3, '17.23', '2023-03-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(19, 2, 3, '18.31', '2023-02-25 11:15:00', '19', '262 Danny Thomas Pl', '6498 Summer Ave'),
(20, 7, 4, '12.23', '2023-02-25 12:09:00', '14', '1900 Summer Ave', '380 Highland St'),
(21, 7, 9, '17.31', '2023-02-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
(22, 6, 3, '18.23', '2023-01-26 12:09:00', '22', '130 Summer Ave', '785 Popular Ae'),
(23, 5, 9, '18.31', '2023-01-26 11:15:00', '28', '482 Danny Thomas Pl', '6498 Summer Ave'),
```

Inserted test data to payments using the following query.

```
INSERT into payments values(
1, 1, 'Credit Card', '14.97'),
(2, 2, 'Debit Card', '23.31'),
(3, 3, 'Gift Card', '14.23'),
(4, 4, 'Credit Card', '24.97'),
(5, 5, 'Debit Card', '23.31'),
(6, 6, 'Gift Card', '14.23');
```

Analytics Requirement

1. How many drivers do you have registered?

```
select count(*) as num_of_drivers from drivers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_of_drivers
14

2. Number of active vs inactive in this month?

   At this moment, 9 are active and 5 are inactive.

```
select status, count(*) as active_drivers from drivers
group by status;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

status	active_drivers
NO	5
YES	9

3. How many customers registered?

```
select count(*) as num_of_customers from users;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_of_customers
11

11 customers registered.

4. Who is the driver with most number of rides?

```
select First_Name, Last_Name, count(r.ride_id) as number_of_rides
from drivers d JOIN rides r on d.driver_id=r.driver_id
group by r.driver_id order by number_of_rides desc limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

First_Name	Last_Name	number_of_rides
Ron	Smith	15

Ron Smith has 15 rides.

5. Who is the passenger with most number of rides?

```
select First_Name, Last_Name, count(r.ride_id) as number_of_rides
from users u JOIN rides r on u.user_id=r.user_id
group by r.user_id order by number_of_rides desc limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

First_Name	Last_Name	number_of_rides
Krishna	Jampana	7

Krishn has 7 rides.

6. Top 5 drivers by revenue by month- January, February, March?

January -

```
select First_Name, Last_Name, sum(ride_price) as total_revenue from rides
JOIN drivers on rides.driver_id = drivers.driver_id
where month(ride_date)=1
group by rides.driver_id
order by total_revenue desc limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

First_Name	Last_Name	total_revenue
Ron	Smith	58.429999351501465
Williams	Comax	35.619998931884766
Marcus	Lee	23.309999465942383
Karuna	U	17.229999542236328
Murthy	Jampana	14.229999542236328

February:

```
select First_Name, Last_Name, sum(ride_price) as total_revenue from rides
JOIN drivers on rides.driver_id = drivers.driver_id
where month(ride_date)=2
group by rides.driver_id
order by total_revenue desc limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

First_Name	Last_Name	total_revenue
Ron	Smith	105.53999710083008
Williams	Comax	79.5399980545044
Chirs	Barrow	39.429999351501465
Karuna	U	13.229999542236328

March:

```
select First_Name, Last_Name, sum(ride_price) as total_revenue from rides
JOIN drivers on rides.driver_id = drivers.driver_id
where month(ride_date)=3
group by rides.driver_id
order by total_revenue desc limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

First_Name	Last_Name	total_revenue
Ron	Smith	70.99999809265137
Williams	Comax	35.61999988555908
Karuna	U	13.229999542236328
Chirs	Barrow	12.229999542236328
Kim	Marcus	8.300000190734863

7. Top 5 customers by revenue this year?

```
select First_Name, Last_Name, sum(ride_price) as total_revenue from rides
JOIN users on rides.user_id = users.user_id
where year(ride_date)=2023
group by rides.user_id
order by total_revenue desc limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

First_Name	Last_Name	total_revenue
Alex	Law	92.45999717712402
Vasthav	d	72.91999816894531
Sam	abc	63.07999801635742
brad	pit	60.519999504089355
Jacob	Cruine	52.76999855041504

8. Has our revenue increased last month compared to last year same month?

```
select year(ride_date) as ride_year, month(ride_date) as ride_month,
sum(ride_price) as total_revenue from rides
where month(ride_date) = 8
group by ride_year, ride_month;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ride_year	ride_month	total_revenue
2022	8	87.30999755859375
2023	8	61.76999855041504

Revenue has been increased.

Customer Questions

Here we are taking customer as ‘Sam’. So, we are executing queries against him. His ID number is 7.

1. What is the most expensive ride?

```
select u.First_Name, u.Last_Name, max(r.ride_price) as expensive_ride
from users u join rides r on u.user_id=r.user_id where u.user_id=7;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

First_Name	Last_Name	expensive_ride
Sam	abc	21.31

2. How many rides did I taken?

```
select u.First_Name, u.Last_Name, count(*) as num_of_rides
from users u join rides r on u.user_id=r.user_id where u.user_id=7;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

First_Name	Last_Name	num_of_rides
Sam	abc	6

3. How much money did I spend with your business year todate?

```
select u.First_Name, u.Last_Name, sum(r.ride_price) as expensive_ride
from users u join rides r on u.user_id=r.user_id where u.user_id=7 and year(ride_date)=2023;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

First_Name	Last_Name	expensive_ride
Sam	abc	63.07999801635742

Operations Questions

1. What is the size of each database?

   Database size[4] is – 144KB

```
select table_schema as 'DB Name', round(sum(data_length+index_length)/1024,1) "DB Size in KB"
FROM information_schema.tables where table_schema="mobility_project";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DB Name	DB Size in KB
mobility_project	144.0

2. Create index on a table and measure the impact on a data retrieval with and without index.

   Initially we ran the following query without index. It took 0.016 sec.

```
select * from rides
where driver_id = 10;
```

   Later we added index [5] on driver_id using the following query.

```
CREATE INDEX idx_ride
ON rides (driver_id);
```

   Again, we ran the query again. Now its showing 0.00sec. i.e data retrieval takes less time with index

129	09:57:31	select * from rides where driver_id = 10 LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
130	09:59:11	CREATE INDEX idx_ride ON rides (driver_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
131	09:59:18	select * from rides where driver_id = 10 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

References:

1. Create ER Diagram of a Database in MySQL Workbench | by Tushar Soam | Medium
2. https://www.guru99.com/database-normalization.html
3. https://www.geeksforgeeks.org/difference-between-primary-key-and-foreign-key/
4. How to check MySQL database and table sizes (a2hosting.com)
5. https://dev.mysql.com/doc/refman/8.0/en/create-index.html