

Car Rental System

Mini Project Report

By,
Harikrishnan G (PES1UG21CS219)
Hitesh Singh (PES1UG21CS233)

Table of Contents:

1.	Introduction	Page 2
2	Project Description	Page 2
3	System Features and Function Requirements	Page 3
4	ER Diagram	Page 4
5	Relational Schema	Page 5
6	DDL SQL Commands	Page 6
7	CRUD Operations	Page 9
8	Functionalities	Page 15
9	Procedures and Triggers	Page 21

1. Introduction

A: Need for Car Rental System:

Car Rental services offer cars for people to rent. It is more convenient than buying a car. A car rental company lends vehicles for a price for a stipulated time frame, at a pre-determined rate. Managing the various aspects of the cars, customers and their rentals can become a taxing task and hence a database system would come in handy.

B: Purpose:

The project's goal is to automate car rental and reservation so that customers do not have to waste time calling and waiting for a vehicle. The project aims to develop a system which can save both, time and effort of the user as well as the employees during the procedure of renting a car.

C: Scope:

This project will provide a user friendly yet powerful way of facilitating the car rentals as well as storing data such as past transactions, customer details, employee details and payments.

2. Project Description

A: Overview:

This project aims to develop a web application for car rental management using nextJS.

B: Major Project Functionalities

Once a customer provides their details and creates an account, they can login to the service to view and book cars for rental.

They can choose a car from a branch of their choice and make a booking.

An employee of the branch will be available to contact and brief them about the process and the vehicle.

Payment can be facilitated through the application.

Admins can log into the service to run sales diagnostics across branches.

3. System Features and Function Requirements

SF-1: Login/SignUp:

Customers need to provide their personal details and signUp with the platform to be able to rent cars.

Employees and the Admin can login using their predefined credentials. Functional Requirements include a username and password for each customer and employee.

SF-2: Browse Cars:

Customers can choose which branch they wish to rent from and view a list of cars available at the branch as well as their specifications.

Functional Requirements include displaying of only those cars that are currently available to rent.

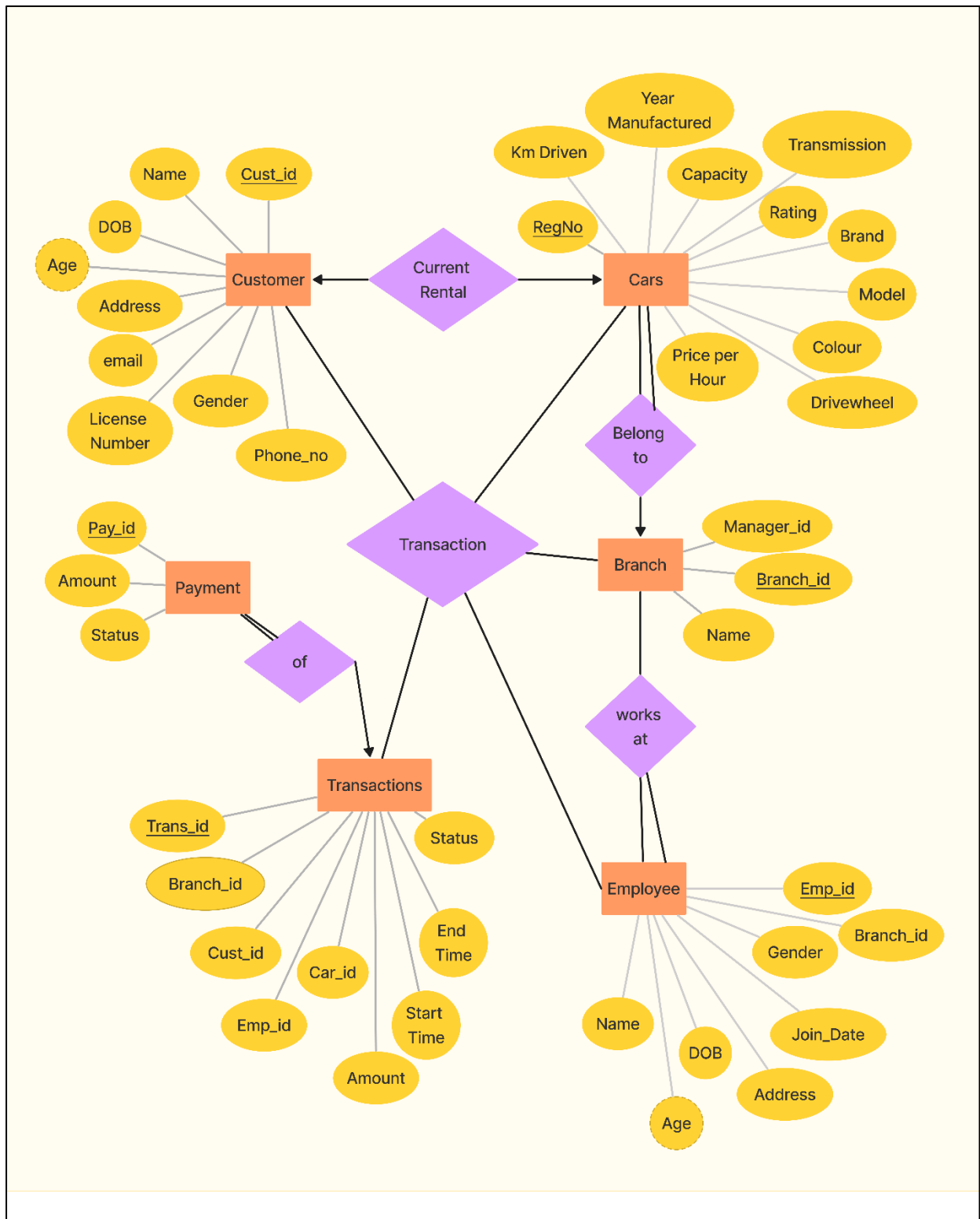
SF-3: Renting a Car:

Customers can make a booking for a car and make a payment for it through the service. Once a booking is confirmed, an employee will be assigned for assistance. Functional Requirements include assigning an employee for the rental and marking the car as reserved in the database so that it does not appear available to other customers.

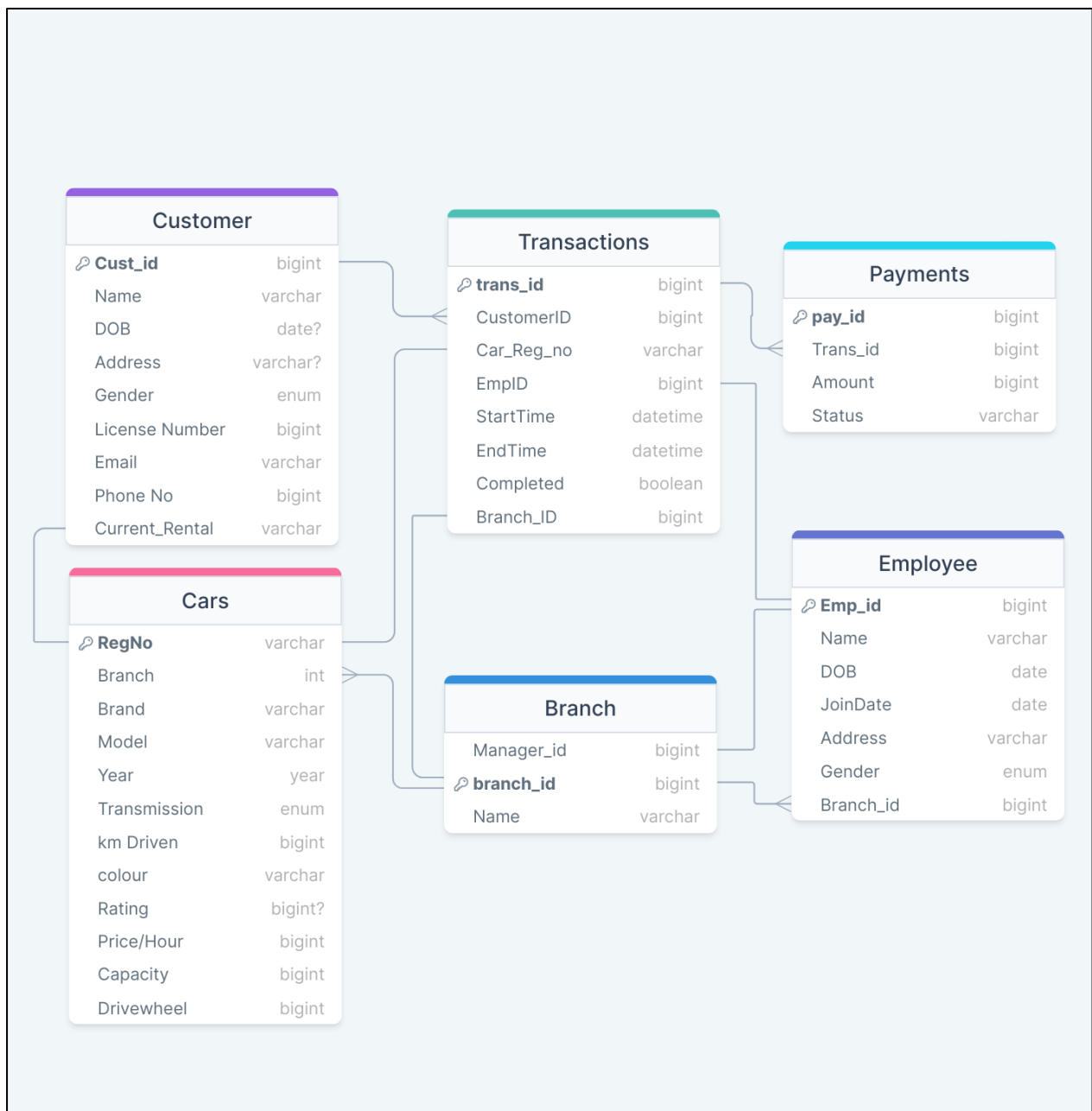
SF-4: Admin Analytics:

Admins can view consolidated information such as demand for a particular brand and branch wise revenue.

4. ER Diagram



5. Relational Shema



6. DDL SQL Commands

The following commands can be run to create the tables and users required for the application.

```
--creating database:

CREATE DATABASE car_rental;
use car_rental;

--Creating tables:

create table customer (
    cust_id bigint PRIMARY KEY auto_increment,
    name varchar(30) not null,
    DOB date not null,
    address varchar(80) not null,
    email varchar(30) not null,
    license_no varchar(20) not null,
    gender enum('Male','Female') not null,
    phone_no bigint not null,
    password varchar(20) not null
);

create table cars(
    reg_no varchar(10) PRIMARY KEY,
    km_driven int,
    year_manf YEAR not null,
    branch_id bigint not null,
    capacity int ,
    transmission enum('Manual','Automatic'),
    rating int ,
    brand varchar(20) not null,
    model varchar(40) not null,
    colour varchar(20),
    drivewheel enum('FWD','RWD','AWD'),
    price_per_hour int not null,
    fuel enum('petrol','diesel','electric','CNG'),
    img varchar(40)
);

create table branch(
    manager_id BIGINT,
    branch_id BIGINT primary key,
    name varchar(40),
    address varchar(80)
```

```

);

create table Employee(
    emp_id bigint PRIMARY KEY auto_increment,
    name varchar(30) not null,
    dob date not null,
    JoinDate date not null,
    email varchar(30) not null,
    address varchar(80),
    Gender enum('Male','Female') not null,
    branch_id bigint,
    password varchar(20) not null
);

create table Transactions(
    trans_id bigint PRIMARY KEY auto_increment,
    cust_id bigint not null,
    car_reg_no varchar(10) not null,
    startTime DateTime ,
    endTime DateTime,
    Status enum('InProgress','Completed','Cancelled','Initiated'),
    amount decimal(10,2),
    branch_id bigint not null
);

create table Payment(
    pay_id bigint PRIMARY KEY,
    trans_id bigint not null,
    amount decimal(10,2),
    status enum('successful','unsuccessful')
);

-- Adding foreign key constraints:

ALTER TABLE cars
ADD FOREIGN KEY (branch_id) REFERENCES branch(branch_id);

ALTER TABLE branch
ADD FOREIGN KEY (manager_id) REFERENCES Employee(emp_id)
on delete set null;

ALTER TABLE Employee
ADD FOREIGN KEY (branch_id) REFERENCES branch(branch_id);

ALTER TABLE Transactions
ADD FOREIGN KEY (cust_id) REFERENCES customer(cust_id);

ALTER TABLE Transactions

```

```
ADD FOREIGN KEY (car_reg_no) REFERENCES cars(reg_no);

ALTER TABLE Transactions
ADD FOREIGN KEY (branch_id) REFERENCES branch(branch_id);

ALTER TABLE Payment
ADD FOREIGN KEY (trans_id) REFERENCES Transactions(trans_id);

--creating roles and thier priviliges:

create role 'Customer';
grant select, update on car_rental.customer to 'Customer';
grant select, update on car_rental.transactions to 'Customer';
grant select on car_rental.cars to 'Customer';

create role 'Employee';
grant select, update on car_rental.employee to 'Employee';
grant select, update, delete on car_rental.transactions to 'Employee';
grant select on car_rental.cars to 'Employee';

create role 'Manager';
grant select, update,delete on car_rental.employee to 'Manager';
grant select, update, delete on car_rental.transactions to 'Manager';
grant select on car_rental.cars to 'Manager';
```


7. CRUD Operations

The following are the queries that are executed through the various Api routes present in the application as well as to initialize the database.

Initialization queries:

```
--Branches:

insert into branch(branch_id, name, address)
values
(1001, 'Cox Town', '#302, 7th Street, Assaye Road, Cox Town, Bangalore'),
(1002, 'Ulsoor', '#411, 9th Street, Cambridge Road, Halasuru, Bangalore'),
(1003, 'Indiranagar', '#233, 2nd Main Road, Indiranagar, Bangalore'),
(1004, 'Majestic', '#12, 1st Main Road, Majestic, Bangalore');

--Cars:

insert into cars(brand,model,drivewheel,fuel,capacity,transmission, reg_no,
price_per_hour, branch_id, km_driven, year_manf, rating, colour, img)
values
("Tata" ,          "Nano Genx" ,          "RWD" , "Petrol" ,4
,"Automatic",    "KA01YH3456", 220, 1001, 25000, '2018',4,
'blue',    'tata_nano'),
("Tata" ,          "Nano Genx" ,          'RWD' , "CNG" , 4
,"Manual",      "KA01HU7543", 200, 1002, 75000, '2014',3,
'blue',    'tata_nano'),
("Datsun" ,        "Redi-Go" ,          "FWD" , "Petrol" ,5
,"Manual",      "KA02GY2345", 200, 1003, 25000, '2018',3,
'red',    'datsun_redigo'),
("Datsun" ,        "Redi-Go" ,          "FWD" , "Petrol" ,5
,"Manual",      "KA02GR7532", 200, 1002, 100000, '2018',3,
'red',    'datsun_redigo'),
("Renault" ,       "Kwid" ,          "FWD" , "Petrol" ,5
,"Manual",      "KA05MX4598", 240, 1003, 25000, '2019',4,
'white', 'renault_kwid'),
("Renault" ,       "Kwid" ,          "FWD" , "Petrol" ,5
,"Manual",      "KA09WQ4918", 240, 1004, 25000, '2018',4,
'white', 'renault_kwid'),
("Maruti Suzuki" , "Eeco" ,          "RWD" , "Petrol" ,5
,'Manual',      "KA21AS8734", 300, 1001, 30000, '2018',2, 'silver',
'ms_ecco'),
("Maruti Suzuki" , "Alto K10" ,          "FWD" , "Petrol" ,5
,"Manual",      "KA06TY7698", 210, 1001, 25000, '2017',4,
'red',    'ms_altok10'),
("Maruti Suzuki" , "Celerio X" ,          "FWD" , "Petrol" ,5
,"Automatic",   "KA08HG3456", 230, 1002, 25000, '2018',4, 'yellow',
'ms_celerio'),
```

```

("Maruti Suzuki" , "Dzire" , "FWD" , "Diesel" ,5
,"Automatic", "KA01UH4523", 260, 1002, 25000, '2018',4,
'white', 'ms_dzire'),
("Maruti Suzuki" , "Dzire" , "FWD" , "Diesel" ,5
,"Manual", "KA01UH4213", 240, 1001, 25000, '2018',4,
'white', 'ms_dzire'),
("Maruti Suzuki" , "Dzire" , "FWD" , "Diesel" ,5
,"Manual", "KA01MG4203", 240, 1003, 40000, '2016',5,
'white', 'ms_dzire'),
("Volkswagen" , "Ameo" , "FWD" , "Diesel" ,5
,"Manual", "KA02NK7867", 260, 1003, 25000, '2018',4, 'yellow',
'vw_ameo'),
("Volkswagen" , "Ameo" , "FWD" , "Diesel" ,5
,"Manual", "KA06YT9067", 260, 1004, 25000, '2016',4, 'yellow',
'vw_ameo'),
("Audi" , "A3" , 'FWD' , "Diesel" ,5
,"Automatic", "KA01HL2002", 420, 1003, 25000, '2018',4,
'black', 'audi_a3'),
("Audi" , "Q3" , "FWD" , "Diesel" ,5
,"Manual", "KA08BH3002", 430, 1003, 25000, '2018',5,
'black', 'audi_q3'),
("Volvo" , "Xc40" , "AWD" , "Petrol" ,5
,"Automatic", "KA23NG4567", 470, 1003, 5000, '2022',5,
'white', 'volvo_xc40'),
("Maruti Suzuki" , "Swift" , "FWD" , "Petrol" ,5
,"Manual", "KA08UY6798", 220, 1004, 25000, '2018',4,
'blue', 'ms_swift')
;

--Customers:

insert into customer(cust_id,name,DOB,license_no,gender,phone_no, email,
address, password)
values
(2002, "Morty", "2002-06-
18","PQ798TY65",'Male',0987654321, 'morty@gmail.com','#4, geneva street,
kormanagala', 'mortypwd'),
(2001, "Rick", "2003-04-15","HY786TY65",'Male',9876543210,
'ricksanchez@gmail.com','#3, polo street, jp nagar', 'rickpwd')
;

--Employees:

insert into Employee(emp_id,name,dob,JoinDate,gender,branch_id, email,
address, password)
values

```

```

(3002, "Angela", "1998-03-017", "2020-03-05", 'Female', 1003,
'angelamartin@gmail.com', '#23 commerical street', 'angelapwd')
(3001, "Kevin", "1995-09-02", "2019-01-15", 'Male', 1003, 'kevin@gmail.com', '#6,
home street, ulsoor', 'keinpwd'),
;

insert into Employee(emp_id,name,dob,JoinDate,gender,branch_id, email,
address, password)
values
(3003, "Jim", "1994-06-09", "2019-01-15", 'Male', 1001, 'jim@gmail.com', '#8,
Coner House', 'jimpwd'),
(3004, "Pam", "1989-05-07", "2020-03-05", 'Female', 1002, 'pam@gmail.com', '#29 VP
street', 'pampwd'),
(3005, "Dwight", "1988-11-17", "2020-03-05", 'Male', 1004, 'dwight@gmail.com',
'#21 Farm Road', 'dwightpwd')
;

```

API queries:

```

JS addCustomer.js X
car_rental > pages > api > db > JS addCustomer.js > handler
25
26 insert into customer(name,DOB,license_no,gender,phone_no, email, address, password)
27 values
28 ("${data.name}", "${data.dob}", "${data.lis_no}", "${data.gender}", "${data.phone_no}", "${data.email}", "${data.address}", '${
29 {data.password}');

```

```

JS addEmployee.js X
car_rental > pages > api > db > JS addEmployee.js > handler > db.query() callback
24
25 insert into Employee(name, email, address,gender,password, JoinDate,branch_id,dob )
26 values
27 ("${req.body.name}", "${req.body.email}", "${req.body.address}", "${req.body.gender}", "${req.body.password}", curdate(), $
28 {req.body.branch_id}, "${req.body.dob}");

```

```

JS authenticate_user.js X
car_rental > pages > api > db > JS authenticate_user.js > handler > db > database
22 select * from customer
23 where (email = '${req.body.email}' or phone_no = '${req.body.phone_no}') and password = '${req.body.password}';
24 ;

```

```

JS bookCar.js X
car_rental > pages > api > db > JS bookCar.js > handler > db.query() callback
20
21 insert into Transactions(cust_id, car_reg_no, branch_id, startTime, endTime,status, amount)
22 values
23 (${req.body.cust_id},
24 "${req.body.car_reg_no}",
25 ${req.body.branch_id},
26 '${req.body.startTime}',
27 '${req.body.endTime}',
28 'InProgress',
29 '${req.body.amount}'
30 )
31 ;

```

```

JS changeManager.js X
car_rental > pages > api > db > JS changeManager.js > handler
24 const queryString =
25 `
26     call changeMan('${req.body.man_id}','${req.body.name}','${req.body.email}','${req.body.address}','${req.body.gender}','${
27     {req.body.password}','${req.body.branch_id}','${req.body.dob}');
`
;

JS checkUser.js X
car_rental > pages > api > db > JS checkUser.js > handler
19 const queryString =
20 `select * from customer
21     where email = '${req.body.email}' or phone_no = '${req.body.phone_no}';
22 `
;

JS custViewInProgressCars.js X
car_rental > pages > api > db > JS custViewInProgressCars.js > handler > db.query() callback
22 `
23     select Cr.brand as brand, Cr.model as model, Cr.reg_no as reg_no, B.name as branch, T.startTime as startTime, T.endTime
24     as endTime, T.amount as amount, T.status as status
25     from transactions T
26     join customer C on T.cust_id = C.cust_id
27     join cars Cr on Cr.reg_no = T.car_reg_no
28     join branch B on B.branch_id = T.branch_id
29     where Status = 'InProgress' and (C.email = "${req.body.email}" or C.phone_no = "${req.body.email}" or C.cust_id LIKE "${
    {req.body.email}");
`
;

JS custViewPastCars.js X
car_rental > pages > api > db > JS custViewPastCars.js > handler
22 `
23     select Cr.brand as brand, Cr.model as model, Cr.reg_no as reg_no, B.name as branch, T.startTime as startTime, T.endTime
24     as endTime, T.amount as amount, T.status as status
25     from transactions T
26     join customer C on T.cust_id = C.cust_id
27     join cars Cr on Cr.reg_no = T.car_reg_no
28     join branch B on B.branch_id = T.branch_id
29     where Status <> 'InProgress' and (C.email = "${req.body.email}" or C.phone_no = "${req.body.email}" or C.cust_id LIKE "${
    {req.body.email}");
`
;

JS deleteEmployee.js X
car_rental > pages > api > db > JS deleteEmployee.js > handler
19 `
20     delete from employee where emp_id = ${req.body.emp_id};
21 `
;

JS empViewInProgressCars.js X
car_rental > pages > api > db > JS empViewInProgressCars.js > handler > db.query() callback
22 `
23     select Cr.brand as brand, Cr.model as model, Cr.reg_no as reg_no, C.name, C.email, C.cust_id, B.name as branch, T.
24     startTime as startTime, T.endTime as endTime, T.amount as amount, T.status as status, T.trans_id as trans_id
25     from transactions T
26     join customer C on T.cust_id = C.cust_id
27     join cars Cr on Cr.reg_no = T.car_reg_no
28     join branch B on B.branch_id = T.branch_id
29     where T.status = 'InProgress' and (
30         T.branch_id = (select branch_id from employee where email = "${req.body.email}")
31         or T.branch_id = (select branch_id from branch where name LIKE "${req.body.branch}")
32     );
`
;

```

```

JS empViewPastCars.js X
car_rental > pages > api > db > JS empViewPastCars.js > handler > queryString
22
23     select Cr.brand as brand, Cr.model as model, Cr.reg_no as reg_no, C.name, C.email, C.cust_id, B.name as branch, T.
24         startTime as startTime, T.endTime as endTime, T.amount as amount, T.status as status , T.trans_id as trans_id
25     from transactions T
26     join customer C on T.cust_id = C.cust_id
27     join cars Cr on Cr.reg_no = T.car_reg_no
28     join branch B on B.branch_id = T.branch_id
29     where T.status <> 'InProgress' and (
30         T.branch_id = (select branch_id from employee where email = "${req.body.email}")
31         or T.branch_id = (select branch_id from branch where name LIKE "${req.body.branch}")
32     );

```

```

JS getAllBranches.js X
car_rental > pages > api > db > JS getAllBranches.js > handler > db.query() callback
13
14     select branch_id, name from branch;
15

```

```

JS getAnalytics.js X
car_rental > pages > api > db > JS getAnalytics.js > handler
19
20     select B.branch_id, B.name as branch_name, E.emp_id as manager_id, E.name as manager_name, sum(T.amount) as revenue,
21         count(*) as sales
22     from transactions T
23     join cars C on C.reg_no = T.car_reg_no
24     join branch B on T.branch_id = B.branch_id
25     join employee E on E.emp_id = B.manager_id
26     GROUP BY B.branch_id, B.name , E.emp_id, E.name;
27

```

```

JS getBranchIDs.js X
car_rental > pages > api > db > JS getBranchIDs.js > handler
19
20     select branch_id from employee where email = "${req.body.email}";
21

```

```

JS getEmployees.js X
car_rental > pages > api > db > JS getEmployees.js > handler
20
21     select * from employee E where E.branch_id = (
22         select branch_id from employee M
23         where M.email = "${req.body.email}" or M.emp_id LIKE '${req.body.emp_id}'
24     )
25     and E.email <> "${req.body.email}" and E.emp_id NOT LIKE '${req.body.emp_id}' ;
26

```

```

JS getManagers.js X
car_rental > pages > api > db > JS getManagers.js > handler > queryString
19
20     select B.branch_id, E.name as name, B.name as branch_name, E.emp_id, E.gender, E.JoinDate from branch B
21     join employee E on b.manager_id = E.emp_id;
22

```

```

JS markAsCompleted.js X
car_rental > pages > api > db > JS markAsCompleted.js > handler > queryString
13
14     update transactions set status = 'Completed' where trans_id = ${req.body.trans_id};
15
16

```

JS searchCars.js X

car_rental > pages > api > db > JS searchCars.js > handler > db.query() callback

```
21 const querystring =
22 `select * from cars C
23   where C.branch_id = (
24     select B.branch_id from branch B where B.name = '${req.body.branch}'
25   )
26   and not exists(
27     select * from transactions T
28     where T.car_reg_no = C.reg_no and(
29       startTime BETWEEN '${req.body.startTime}' AND '${req.body.endTime}'
30       or endTime BETWEEN '${req.body.startTime}' AND '${req.body.endTime}'
31     )
32   )`;
```

JS searchFilteredCars.js X

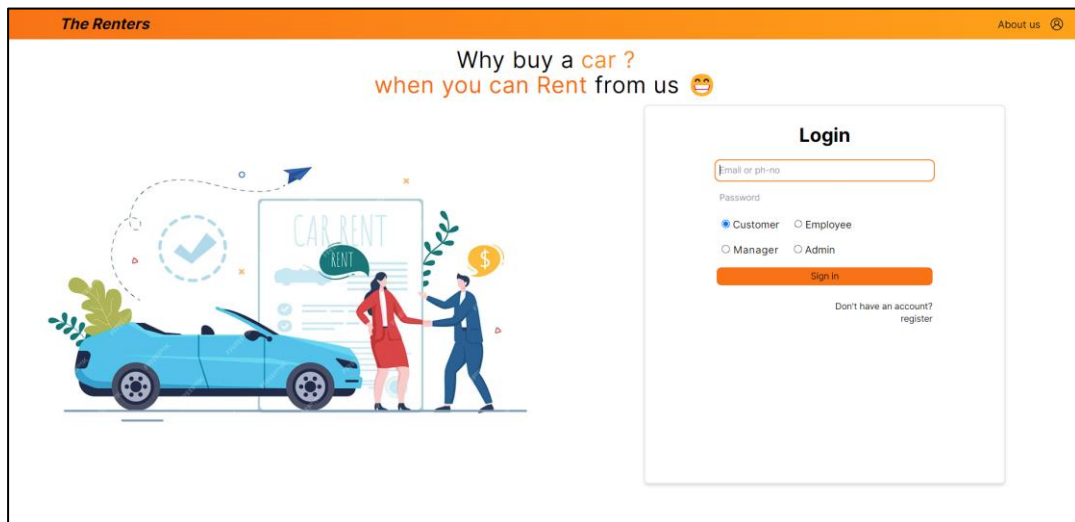
car_rental > pages > api > db > JS searchFilteredCars.js > handler > db.query() callback

```
15 `select * from cars C
16   where C.branch_id = (
17     select B.branch_id from branch B where B.name = '${req.body.branch}'
18   )
19   and not exists(
20     select * from transactions T
21     where T.car_reg_no = C.reg_no and(
22       startTime BETWEEN '${req.body.startTime}' AND '${req.body.endTime}'
23       or endTime BETWEEN '${req.body.startTime}' AND '${req.body.endTime}'
24     )
25   )
26   and C.fuel LIKE '${req.body.fuel}'
27   and C.transmission LIKE '${req.body.transmission}'
28   and C.rating LIKE '${req.body.ratings}'
29   and C.capacity LIKE '${req.body.capacity}'
30   ORDER BY C.price_per_hour ${req.body.sort}
31   ;
32 `;
```

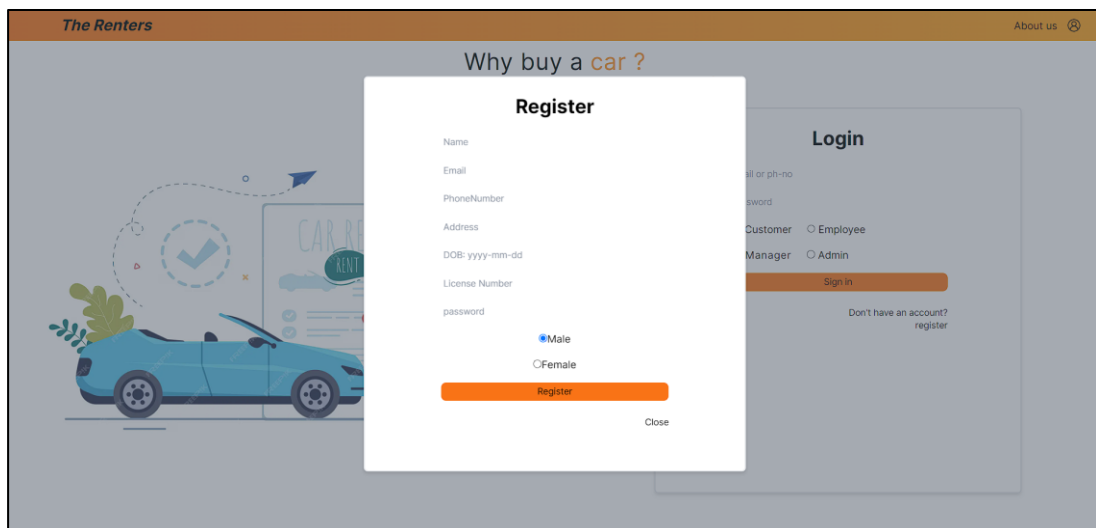
8. Functionalities

Login/Sign-up:

Customers, employees, managers and the admin can all login from the same page. New customers can create a new account for themselves.



The screenshot shows the 'The Renters' website interface. The header is orange with the text 'The Renters' on the left and 'About us' with a user icon on the right. The main content area has a light blue background with the text 'Why buy a car ? when you can Rent from us 😊'. Below this text is an illustration of a blue car, a woman in a red dress, and a man in a blue suit. To the right of the illustration is a white 'Login' form. The form contains a text input for 'Email or ph-no', a text input for 'Password', and four radio buttons for user roles: 'Customer' (selected), 'Employee', 'Manager', and 'Admin'. Below the radio buttons is an orange 'Sign in' button. At the bottom of the form, there is a link that says 'Don't have an account? register'.



The screenshot shows the 'The Renters' website interface with the 'Register' form open. The header is orange with the text 'The Renters' on the left and 'About us' with a user icon on the right. The main content area has a light blue background with the text 'Why buy a car ?'. Below this text is an illustration of a blue car, a woman in a red dress, and a man in a blue suit. The 'Register' form is a white modal box with the following fields: 'Name', 'Email', 'PhoneNumber', 'Address', 'DOB: yyyy-mm-dd', 'License Number', and 'password'. Below the 'password' field are two radio buttons for gender: 'Male' (selected) and 'Female'. At the bottom of the form is an orange 'Register' button and a 'Close' link. In the background, the 'Login' form is visible but dimmed.

Search Cars:

Customers can search cars by selecting a branch and a date-time range.

The Renters

About us

Rent a Car

Book Your best drive now

Select Branch

Start dd-mm-yyyy --:-- -- End dd-mm-yyyy --:-- --

Let's select the car

View Current Bookings

View Past Bookings

View Bookings:

Customers can view their pasts and ongoing bookings.

Book Your best drive now

Select Branch

Start dd-mm-yyyy --:-- -- End dd-mm-yyyy --:-- --

Let's select the car

Hide Current Bookings

Car	Reg. No.	Branch	Start Time	End Time	Amount	Status
Maruti Suzuki Swift	KA08UY6798	Majestic	2023-11-25 15:58	2023-11-27 15:58	10560.00	InProgress
Maruti Suzuki Dzire	KA01UH4523	Uisoor	2023-11-24 16:17	2023-11-28 16:17	24960.00	InProgress
Maruti Suzuki Dzire	KA01UH4213	Cox Town	2023-11-25 16:54	2023-11-29 16:54	23040.00	InProgress

Hide Past Bookings

Car	Reg. No.	Branch	Start Time	End Time	Amount	Status
Datsun Redi-Go	KA02GY2345	Indiranagar	2023-11-22 15:57	2023-11-24 15:57	9600.00	Completed

Filter Cars:

Customers can filter the results of available cars.

The Renters

About us

Filters

Indiranagar

→

Sort by:

Low-High

High-Low

Reset

Fuel type:

Petrol

Diesel

EV

CNG

Capacity:

Four

Five

Seven

Transmission:

Manual


Automatic

Ratings:


3

4


5




Volvo Xc40
Automatic petrol 5 Seats
470rs per Hour




Audi Q3
Manual diesel 5 Seats
430rs per Hour




Audi A3
Automatic diesel 5 Seats
420rs per Hour



Volkswagen Ameo
Manual diesel 5 Seats
260rs per Hour




Maruti Suzuki Dzire
Manual diesel 5 Seats
240rs per Hour



Renault Kwid
Manual petrol 5 Seats
240rs per Hour

Pre-booking Summary:

Customers can see all details of their booking before proceeding to pay.



Amount:

₹22560

Proceed to Pay

Volvo

Xc40

Reg Number: KA23NG4567

Colour: white

Fuel: petrol

Drivewheel: AWD

Transmission: Automatic

Year Manufactured: 2022

Volvo

Xc40

Reg Number: KA23NG4567

Colour: white

Fuel: petrol

Drivewheel: AWD

Transmission: Automatic

Year Manufactured: 2022

Price per Hour: 470

Rating: 5

Seating Capacity: 5

Trip Duration:

2023-11-29 13:29:00 → 2023-12-01 13:29:00

Proceed to Pay

Employee Booking Management:

Employees can manage bookings of their branch. They can mark any ongoing booking as completed.

The Renters								About us
Welcome Emp !								
Let's look at the Rent Information								
								Completed Ongoing
Completed:								
Car	Reg. No.	Customer Id	Customer Name	Start Time	End Time	Amount	Status	
Datsun Redi-Go	KA02GY2345	2002	Morty	2023-11-22 15:57	2023-11-24 15:57	9600.00	Completed	
Audi A3	KA01HL2002	2004	Peter Parker	2023-11-24 04:03	2023-11-28 04:03	40320.00	Completed	
Audi Q3	KA08BH3002	2003	Mary Jane	2023-11-24 06:04	2023-11-26 06:04	20640.00	Completed	

The Renters								About us
Welcome Emp !								
Let's look at the Rent Information								
								Completed Ongoing
Ongoing:								
Car	Reg. No.	Customer ID	Customer Name	Start Time	End Time	Amount	Status	
Volvo Xc40	KA23NG4567	2002	Morty	2023-11-29 07:59	2023-12-01 07:59	22560.00		Mark As Completed

View/Add Employees:

The manager of a branch can view all employees of their branch as well as add a new employee.

The Renters					About us
Employee details of your branch					Switch to Rent info
ID	Name	Gender	Join Date	Action	
3014	kelly	Female	2023-11-22T18:30:00.000Z	Remove	
3022	packer	Male	2023-11-22T18:30:00.000Z	Remove	
3023	karen	Female	2023-11-22T18:30:00.000Z	Remove	
3024	phylis	Female	2023-11-22T18:30:00.000Z	Remove	
					Add new employee

The Renters					About us
Employee details of your branch					Switch to Rent info
ID	Name	Gender	Join Date	Action	
3014	kelly	Female		Remove	
3022	packer	Male		Remove	
3023	karen	Female		Remove	
3024	phylis	Female		Remove	
					Add new employee

Register

Name

Email

Address

DOB: yyyy-mm-dd

password

☒ Male

☐ Female

Confirm

Close

View Managers:

The admin can view all branch managers and even change the manager of a branch.

The Renters

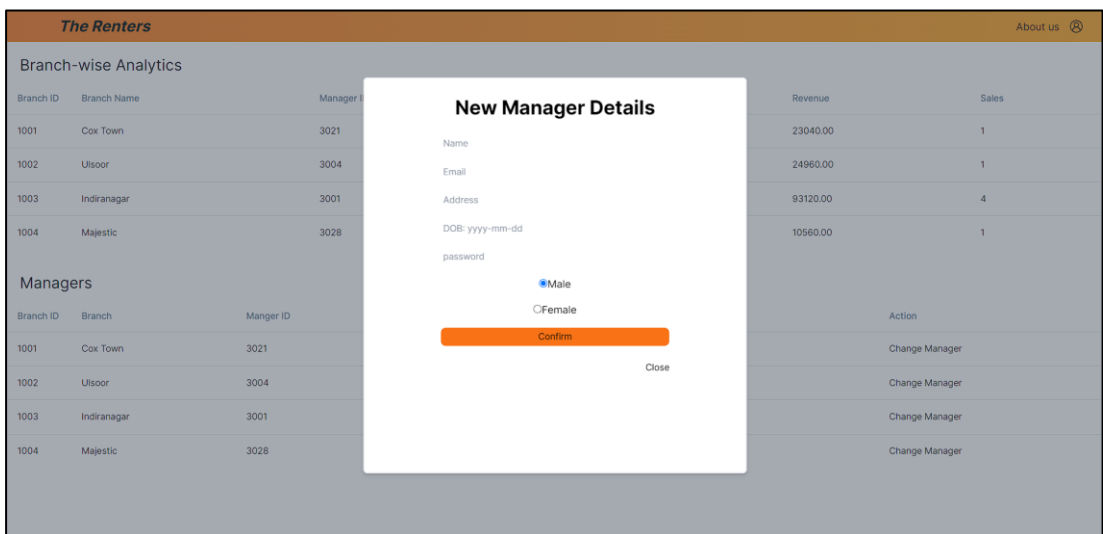
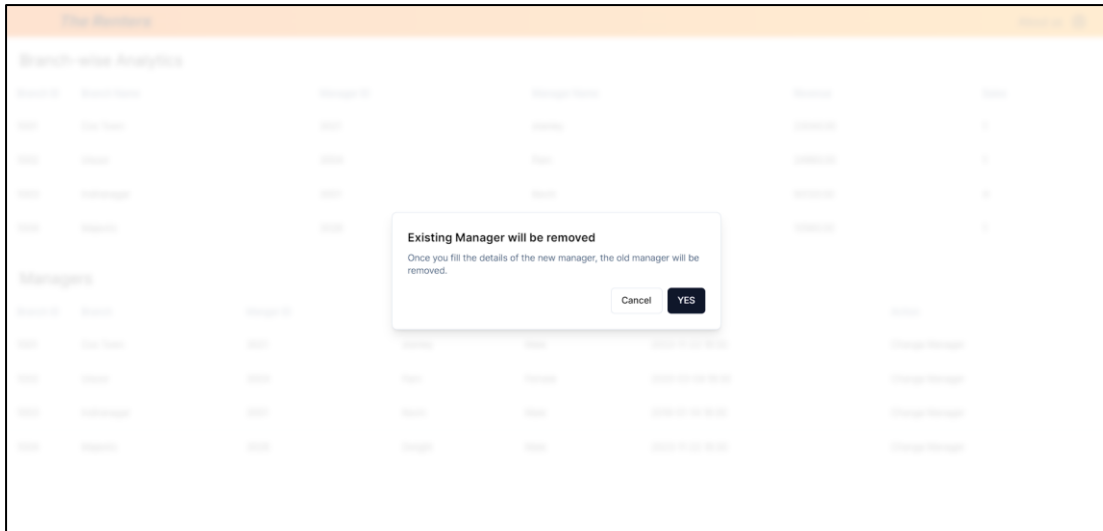
About us

Branch-wise Analytics

Branch ID	Branch Name	Manager ID	Manager Name	Revenue	Sales
1001	Cox Town	3021	stanley	23040.00	1
1002	Ulsoor	3004	Pam	24960.00	1
1003	Indiranagar	3001	Kevin	93120.00	4
1004	Majestic	3028	Dwight	10560.00	1

Managers

Branch ID	Branch	Manger ID	Name	Gender	Join Date	Action
1001	Cox Town	3021	stanley	Male	2023-11-22 18:30	Change Manager
1002	Ulsoor	3004	Pam	Female	2020-03-04 18:30	Change Manager
1003	Indiranagar	3001	Kevin	Male	2019-01-14 18:30	Change Manager
1004	Majestic	3028	Dwight	Male	2023-11-22 18:30	Change Manager



Branch-wise Analytics:

Admin can also see the revenue and sales numbers of each branch.

The Renters							About us
Branch-wise Analytics							
Branch ID	Branch Name	Manager ID	Manager Name	Revenue	Sales		
1001	Cox Town	3021	stanley	23040.00	1		
1002	Ulsoor	3004	Pam	24960.00	1		
1003	Indiranagar	3001	Kevin	93120.00	4		
1004	Majestic	3028	Dwight	10560.00	1		
Managers							
Branch ID	Branch	Manger ID	Name	Gender	Join Date	Action	
1001	Cox Town	3021	stanley	Male	2023-11-22 18:30	Change Manager	
1002	Ulsoor	3004	Pam	Female	2020-03-04 18:30	Change Manager	
1003	Indiranagar	3001	Kevin	Male	2019-01-14 18:30	Change Manager	
1004	Majestic	3028	Dwight	Male	2023-11-22 18:30	Change Manager	

9. Procedures and Triggers

Procedures:

```
-- changing manager:

drop procedure if exists changeMan;
Delimiter $$
CREATE PROCEDURE changeMan (
    IN old_man_id bigint,
    IN gname VARCHAR(30),
    IN gemail VARCHAR(30),
    IN gaddress VARCHAR(80),
    IN gggender VARCHAR(10),
    IN gpassword VARCHAR(20),
    IN gbranch_id bigint,
    IN gdob date
)
BEGIN

    insert into Employee(name, email, address,gender,password,
JoinDate,branch_id,dob)
    values
    (gname,gemail,gaddress,gggender,gpassword,curdate(), gbranch_id,gdob);

    delete from employee where emp_id = old_man_id;

    update branch set manager_id = (select emp_id from employee where email =
gemail)
    where branch_id = gbranch_id;

end $$
Delimiter ;

-----
```

Triggers:

```
--- checking for duplicate employee:

Delimiter &&
CREATE TRIGGER employee_insert_trigger
    BEFORE INSERT
    ON employee
    FOR EACH ROW
    BEGIN
    IF EXISTS (SELECT 1
```

```

        FROM employee
        WHERE email = NEW.email)
    THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User Already Exists ';
    END IF;
END&&
Delimiter ;

--- checking for duplicate customer:

Delimiter &&
CREATE TRIGGER customer_insert_trigger
    BEFORE INSERT
    ON customer
    FOR EACH ROW
    BEGIN
        IF EXISTS (SELECT 1
            FROM customer
            WHERE email = NEW.email)
        THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User Already Exists ';
        END IF;
    END&&
Delimiter ;

```