CS353 Project Design Report

CaRent

Group Members

Kaan Ateşel 21703694

Cemal Faruk Güney 21903474

Arda Önal 21903350

Beste Güney 21901631

Instructor: Özgür Ulusoy

TA: Mustafa Can Cavdar

Website: https://kaanatesel.github.io/carent/

Table Of Contents

ER Diagram Revisions

According to the feedback taken by our TA, the ER diagram of our project has been revised.

The below are the points where the revisions are made on the ER diagram:

- Firstly, foreign key representations at the ER entities are deleted and the corresponding relationships are added to the ER diagram instead of foreign keys. These foreign key relations are shown in relational models.
- Total participation required situations are fixed at the ER diagram.
- A new feature is added to the project. At this feature, a new employee type chauffeur was added to the system. This way users will have the ability to rent their cars with or without chauffeurs. In addition to that, when a user chooses to rent a car with a chauffeur, chauffeurs can accept this request or decline. When the chauffeur declines the reservation, the user will get notified and the reservation will be dropped from the system.
- At reservations, an option of insurance is added to the system. Users can choose different types of insurances which have different prices. Insurance price will be added to the reservation cost, if insurance is chosen.
- For login and signup processes in the project, a new entity set called User is created. User is the super entity of all the users of the system(customer, employee) and has a password, email, address and phone number information for registering to the system.

Revised ER Diagram

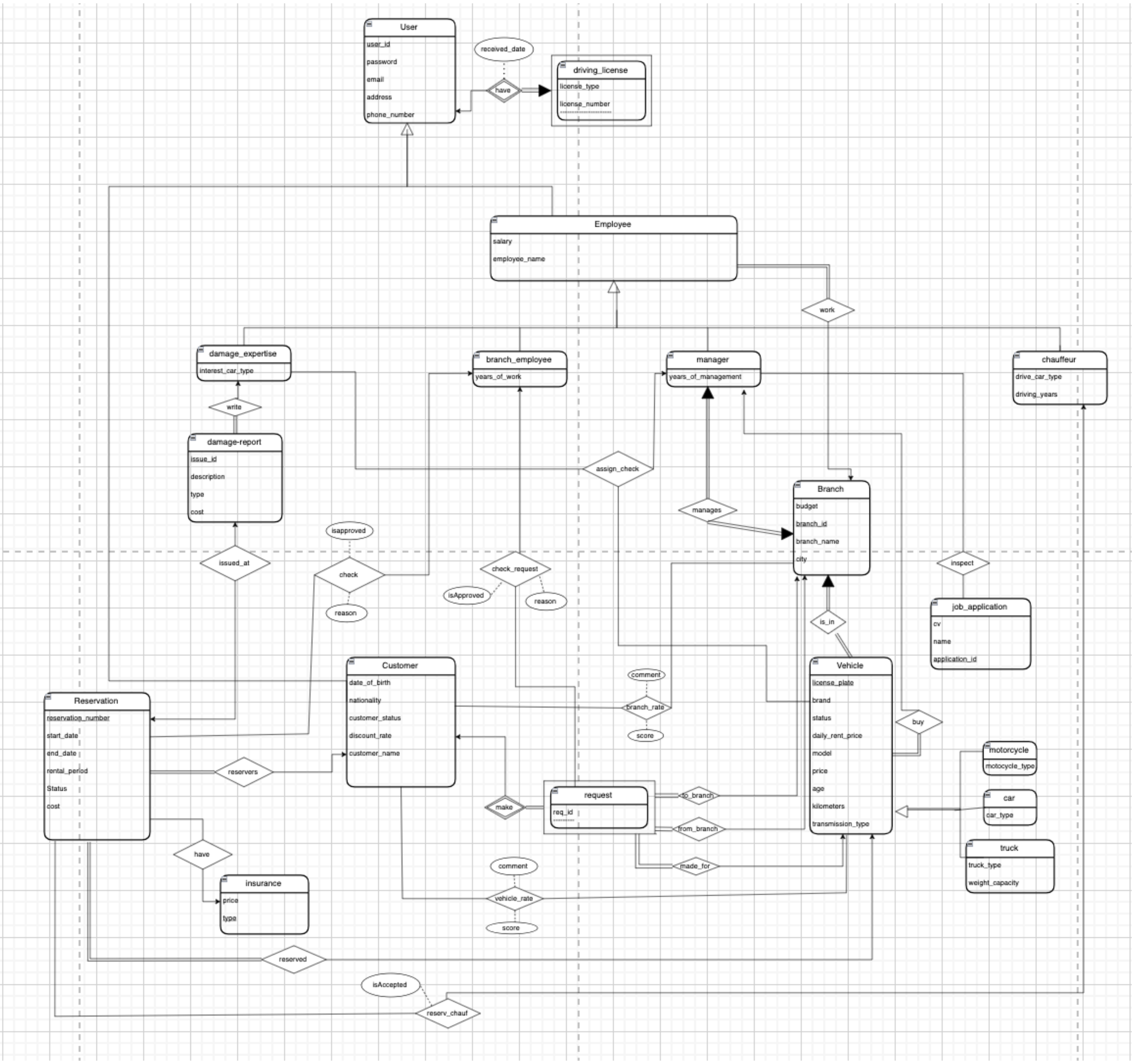


Table Schemas

User

a) Relational Model

User(user_id, password, email, address, phone_number)

b) Functional Dependencies

user_id -> password, email, address, phone_number
email -> user_id, password, address, phone_number
phone_number -> user_id, password, email, address

c) Candidate Keys

{user_id}
{email}
{phone_number}

d) Normal Form

The table is in both BCNF and 3NF normal form because in the functional dependency user_id is a superkey, in the second dependency email is a superkey and in the third dependency phone_number is a superkey.

e) Table Definition

```
create table User(
        user_id  int not null auto_increment,
        password varchar(50) not null,
        email  varchar(50) not null,
        address varchar(50),
        phone_number varchar(15),
        primary key (user_id)
);
```

Driving License

a) Relational Model

driving_license(user_id, license_number, license_type, received_date)

b) Functional Dependencies

user_id, license_number -> license_type, received_date

c) Candidate Keys

{user_id, license_number}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and license_number together with user_id is the superkey of this table.

e) Table Definition

```
create table driving_license(
        user_id int not null auto_increment,
        license_number int,
        license_type char(3),
        received_date date,
        check (license_type in (‘A1’, ‘A2’, ‘A’, ‘M’, ‘B1’, ‘B’, ‘BE’, ‘C1’, ‘C’, ‘CE’)),
        FOREIGN KEY user_id REFERENCES user(user_id),
        PRIMARY KEY (license_number)
);
```

Employee

a) Relational Model

Employee(user_id, salary, employee_name, branch_id)

b) Functional Dependencies

user_id -> salary, employee_name, branch_id

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table employee(
        user_id int not null auto_increment,
        salary numeric(8,2),
        employee_name varchar(20),
        branch_id int not null,
        FOREIGN KEY branch_id REFERENCES branch(branch_id),
        FOREIGN KEY user_id REFERENCES user(user_id),
        PRIMARY KEY (user_id)
);
```

Customer

a) Relational Model

Customer(user_id, date_of_birth, nationality, customer_status, customer_name)

b) Functional Dependencies

user_id -> date_of_birth, nationality, customer_status, discount_rate, customer_name

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table customer(
        user_id int not null auto_increment,
        date_of_birth date,
        nationality varchar(20),
        customer_status varchar(10),
        customer_name varchar(20),
        check (customer_status in (‘Gold’, ‘Silver’, ‘Premium’, ‘Normal’)),
        FOREIGN KEY user_id REFERENCES user(user_id),
        FOREIGN KEY customer_status REFERENCES
customer_discount(customer_status),
        PRIMARY KEY (user_id)
);
```

Customer Discount

a) Relational Model

customer_discount(<u>customer_status</u>, discount_rate)

b) Functional Dependencies

customer_status -> discount_rate

c) Candidate Keys

{customer_status}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and customer_status is the superkey of this table.

e) Table Definition

```
create table customer_discount(
        customer_status varchar(10),
        discount_rate int,
        check (discount_rate in (10, 20, 30)),
        check (customer_status in (‘Gold’, ‘Silver’, ‘Premium’)),
        primary key(customer_status)
);
```

Branch

a) Relational Model

Branch(branch_id, budget, branch_name, city, manager_id)

b) Functional Dependencies

branch_id -> budget, branch_name, city, manager_id

c) Candidate Keys

{branch_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and branch_id is the superkey of this table.

e) Table Definition

```
create table branch(
        branch_id int not null auto_increment,
        budget int,
        manager_id int,
        branch_name varchar(20),
        FOREIGN KEY manager_id REFERENCES manager(user_id),
        PRIMARY KEY (branch_id)
);
```

Manager

a) Relational Model

Manager(user_id, years_of_management)

b) Functional Dependencies

user_id -> years_of_management

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table manager(
        user_id int not null auto_increment,
        years_of_management int,
        FOREIGN KEY user_id REFERENCES Employee(user_id),
        PRIMARY KEY (user_id)
);
```

Vehicle

a) Relational Model

Vehicle(license_plate, status, daily_rent_price, model, price, age, kilometers, transmission_type, buying_manager_id, branch_id)

b) Functional Dependencies

license_plate -> status, daily_rent_price, model, price, age, kilometers, transmission_type, buying_manager_id, branch_id

c) Candidate Keys

{license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and license_plate is the superkey of this table.

e) Table Definition

```
create table vehicle(
        license_plate varchar(8) not null,
        status varchar(20),
        daily_rent_price float,
        model varchar(15),
        price int,
        age int,
        kilometers int,
        transmission_type varchar(10),
        buying_manager_id int not null,
        branch_id int,
        check (status in (‘on_rent’, ‘available’, ‘on_transfer’, ‘unavailable’, ‘reserved’),
        check (transmission_type in (‘Automatic’, ‘Manual’),
        PRIMARY KEY (license_plate),
        FOREIGN KEY buying_manager_id REFERENCES Manager(user_id),
        FOREIGN KEY branch_id REFERENCES branch(branch_id),
        FOREİGN KEY model REFERENCES model_brand(model)
);
```

Model Brand

a) Relational Model

model_brand(model, brand)

b) Functional Dependencies

model -> brand

c) Candidate Keys

{model}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and model is the superkey of this table.

e) Table Definition

```
create table model_brand(
        model model varchar(15),
        brand varchar(20),
        primary key(model)
);
```

Car

a) Relational Model

Car(license_plate, car_type)

b) Functional Dependencies

license_plate -> car_type

c) Candidate Keys

{license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and license_plate is the superkey of this table.

e) Table Definition

```
create table car(
        license_plate varchar(8) not null,
        car_type char(1),
        check (car_type in (‘A’, ‘B’, ‘C’, ‘D’)),
        FOREIGN KEY license_plate REFERENCES vehicle(license_plate),
        PRIMARY KEY (license_plate) );
```

Truck

a) Relational Model

Truck(license_plate, truck_type, weight_capacity)

b) Functional Dependencies

license_plate -> truck_type, weight_capacity

c) Candidate Keys

{license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and license_plate is the superkey of this table.

e) Table Definition

```
create table truck(
        license_plate varchar(8) not null,
        truck_type char(1),
        weight_capacity int,
        check (truck_type in (‘A’, ‘B’, ‘C’, ‘D’)),
        FOREIGN KEY license_plate REFERENCES vehicle(license_plate),
        PRIMARY KEY (license_plate)
);
```

Motorcycle

a) Relational Model

Motorcycle(license_plate, motorcycle_type)

b) Functional Dependencies

license_plate -> motorcycle_type

c) Candidate Keys

{license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and license_plate is the superkey of this table.

e) Table Definition

```
create table motorcycle(
        license_plate varchar(8) not null,
        motorcycle_type char(1),
        check (motorcycle_type in (‘A’, ‘B’, ‘C’, ‘D’)),
        FOREIGN KEY license_plate REFERENCES vehicle(license_plate),
        PRIMARY KEY (license_plate)
);
```

Job Application

a) Relational Model

Job_application(application_id, cv, name)

b) Functional Dependencies

application_id -> cv, name

c) Candidate Keys

{application_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and application_id is the superkey of this table.

e) Table Definition

```
create table job_application(
        application_id int not null auto_increment,
        cv blob,
        name varchar(20),
        primary key (application_id)
);
```

Inspect

a) Relational Model

inspect(<u>manager_id</u>, <u>application_id</u>)

b) Functional Dependencies

Only trivial functional dependencies.

c) Candidate Keys

{application_id, manager_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only trivial functional dependencies.

e) Table Definition

```
create table inspect(
        application_id int not null,
        manager_id int,
        FOREIGN KEY application_id REFERENCES job_application(application_id),
        FOREIGN KEY manager_id REFERENCES manager(user_id),
        PRIMARY KEY (manager_id, application_id)
);
```

Branch Employee

a) Relational Model

branch_employee(user_id, years_of_work)

b) Functional Dependencies

user_id -> years_of_work

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table branch_employee(
        user_id int not null,
        years_of_work int,
        FOREIGN KEY user_id REFERENCES Employee(user_id),
        PRIMARY KEY (user_id)
);
```

Damage Expertise

a) Relational Model

damage_expertise(user_id, interest_car_type)

b) Functional Dependencies

user_id -> interest_car_type

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table damage_expertise(
        user_id int not null,
        interest_car_type varchar(20),
        FOREIGN KEY user_id REFERENCES Employee(user_id),
        PRIMARY KEY (user_id)
);
```

Damage Report

a) Relational Model

damage_report(issue_id, description, type, cost, author_expertise_id, issued_reservation)

b) Functional Dependencies

issue_id-> description, type, cost, author_expertise_id, issued_reservation

c) Candidate Keys

{issue_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and issue_id is the superkey of this table.

e) Table Definition

```
create table job_application(
        issue_id int not null,
        description text,
        type varchar(20),
        cost float,
        author_expertise_id int not null,
        issued_reservation int not null,
        FOREIGN KEY author_expertise_id REFERENCES damage_expertise(user_id),
        FOREIGN KEY issued_reservation REFERENCES reservation(reservation_id),
        PRIMARY KEY (issue_id)
);
```

Reservation

a) Relational Model

Reservation(reservation_number, start_date, end_date, rental_period, status, cost, reserver, checked_by, isApproved, reason, insurance_type, license_plate, reserved_chauf_id, isChaufAccepted)

b) Functional Dependencies

reservation_number-> start_date, end_date, rental_period, status, cost, reserver, checked_by, isApproved, reason, insurance_type, license_plate, reserved_chauf_id, isChaufAccepted

c) Candidate Keys

{reservation_number}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and reservation_number is the superkey of this table.

e) Table Definition

```
create table reservation(
        reservation_number int not null auto_increment,
        start_date date,
        end_date date,
        rental_period AS (DATEDIFF (dd, [start_date], [end_date]])),
        status varchar(10),
        cost float,
        reserver int not null,
        checked_by int not null,
        isApproved varchar(5),
        reason text,
        insurance_type varchar(10),
        license_plate varchar(8),
        reserved_chauf_id int,
        isChaufAccepted varchar(5),
        check (isApproved  in (‘true’, ‘false’),
        check (isChaufAccepted in (‘true’, ‘false’),
        check (status in (‘on_rent’, ‘accepted’, ‘not_accepted’, ‘canceled’, ‘paid’, not_paid)),
        FOREIGN KEY reserver REFERENCES Customer(user_id),
        FOREIGN KEY checked_by REFERENCES branch_employee(user_id),
        FOREIGN KEY license_plate REFERENCES vehicle(license_plate),
        FOREIGN KEY reserved_chauf_id REFERENCES chauffeur(user_id),
        FOREIGN KEY insurance_type REFERENCES insurance(insurance_type),
        PRIMARY KEY (reservation_number));
```

Insurance

a) Relational Model

insurance(insurance_type, insurance_price)

b) Functional Dependencies

insurance_type -> insurance_price

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and insurance_type is the superkey of this table.

e) Table Definition

```
create table insurance(
        insurance_price float,
        insurance_type varchar(10),
        primary key(insurance_type),
);
```

Chauffeur

a) Relational Model

chauffeur(user_id, drive_car_type, driving_years)

b) Functional Dependencies

user_id -> drive_car_type, driving_years

c) Candidate Keys

{user_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and user_id is the superkey of this table.

e) Table Definition

```
create table chauffeur(
        user_id int not null,
        drive_car_type varchar(20),
        driving_years int,
        FOREIGN KEY user_id REFERENCES Employee(user_id),
        PRIMARY KEY (user_id) );
```

Request

a) Relational Model

Request(req_id, made_by_customer, from_branch, to_branch, requested_vehicle, checked_by_employee, isApproved, reason)

b) Functional Dependencies

req_id-> made_by_customer, from_branch, to_branch, requested_vehicle, checked_by_employee, isApproved, reason

c) Candidate Keys

{req_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and req_id is the superkey of this table.

e) Table Definition

```
create table request(
        req_id int not null,
        made_by_customer int not null,
        from_branch int not null,
        to_branch int not null,
        requested_vehicle varchar(8) not null,
        checked_by_employee int not null,
        isApproved varchar,
        reason text,
        FOREIGN KEY made_by_customer REFERENCES Customer(user_id),
        FOREIGN KEY from_branch REFERENCES Branch(branch_id),
        FOREIGN KEY to_branch REFERENCES Branch(branch_id),
        FOREIGN KEY requested_vehicle REFERENCES Vehicle(license_plate),
        FOREIGN KEY checked_by_employee REFERENCES
        branch_employee(user_id),
        PRIMARY KEY (req_id)
);
```

Vehicle Rate

a) Relational Model

Vehicle_Rate(customer_id, license_plate, comment, score)

b) Functional Dependencies

customer_id, license_plate -> comment, score

c) Candidate Keys

{customer_id, license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and customer_id and license_plate couple is the superkey of this table.

e) Table Definition

```
create table vehicle_rate(
        customer_id int not null,
        license_plate varchar(8) not null,
        comment text,
        score int,
        check (score in (1, 2, 3, 4, 5),
        FOREIGN KEY customer_id REFERENCES Customer(user_id),
        FOREIGN KEY license_plate REFERENCES Vehicle(license_plate),
        PRIMARY KEY (customer_id, license_plate)
);
```

Branch Rate

a) Relational Model

Branch_Rate(customer_id, branch_id, comment, score)

b) Functional Dependencies

customer_id, branch_id-> comment, score

c) Candidate Keys

{customer_id, branch_id}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only one non-trivial functional dependency and customer_id and branch_id couple is the superkey of this table.

e) Table Definition

```
create table branch_rate(
        customer_id int not null,
        branch_id int not null,
        comment text,
        score int,
        check (score in (1, 2, 3, 4, 5),
        FOREIGN KEY customer_id REFERENCES Customer(user_id),
        FOREIGN KEY branch_id REFERENCES Branch(branch_id),
        PRIMARY KEY (customer_id, branch_id)
);
```

Assign Check

a) Relational Model

assign_check(assigned_expertise_id, assigning_manager_id, assigned_vehicle_license_plate)

b) Functional Dependencies

Only trivial functional dependencies.

c) Candidate Keys

{assigned_expertise_id, assigning_manager_id, assigned_vehicle_license_plate}

d) Normal Form

The table is in both BCNF and 3NF normal form because of the fact that this table has only trivial functional dependencies.

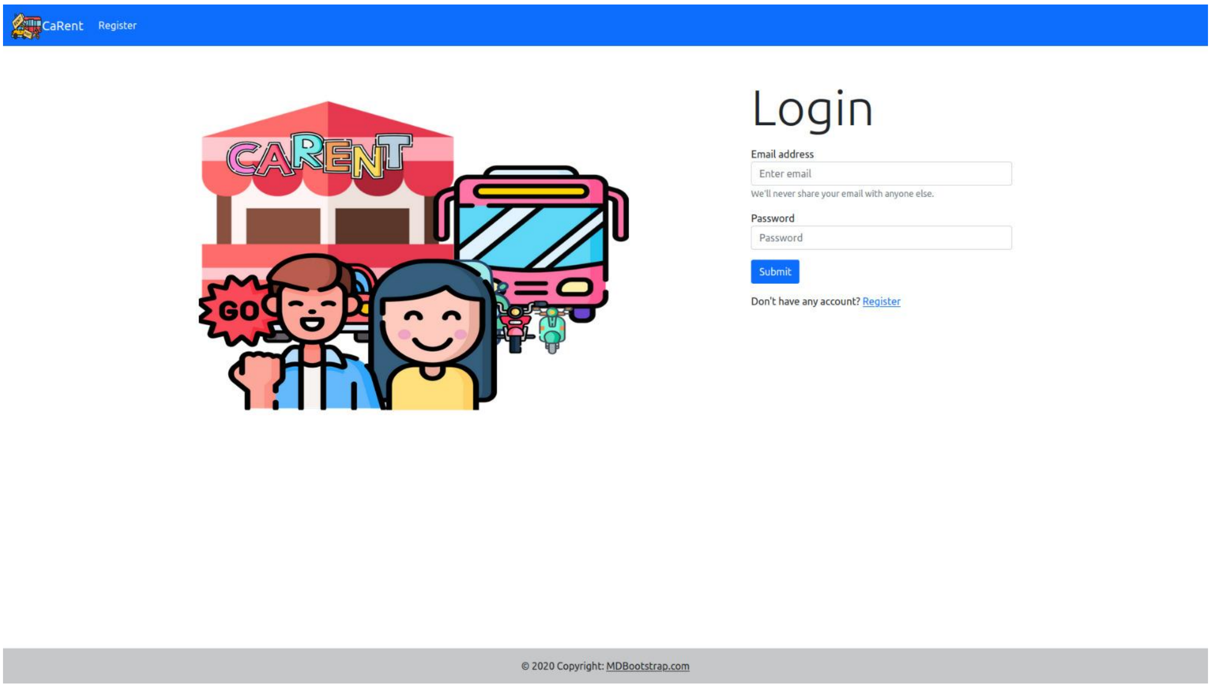
e) Table Definition

```
create table assign_check(
        assigned_expertise_id int not null,
        assigning_manager_id int not null,
        assigned_vehicle_license_plate varchar(8) not null,
        FOREIGN KEY assigned_expertise_id REFERENCES
        damage_expertise(user_id),
        FOREIGN KEY assigning_manager_id REFERENCES Manager(user_id),
        FOREIGN KEY assigned_vehicle_license_plate REFERENCES
        Vehicle(license_plate),
        PRIMARY KEY (assigned_expertise_id, assigning_manager_id,
        assigned_vehicle_license_plate)
);
```

User Interface Design and Corresponding SQL Statements

Note: In the UI's customer and chauffeur id's refer to the user_id attribute, just to show who is logged in, it is written that way.

Login



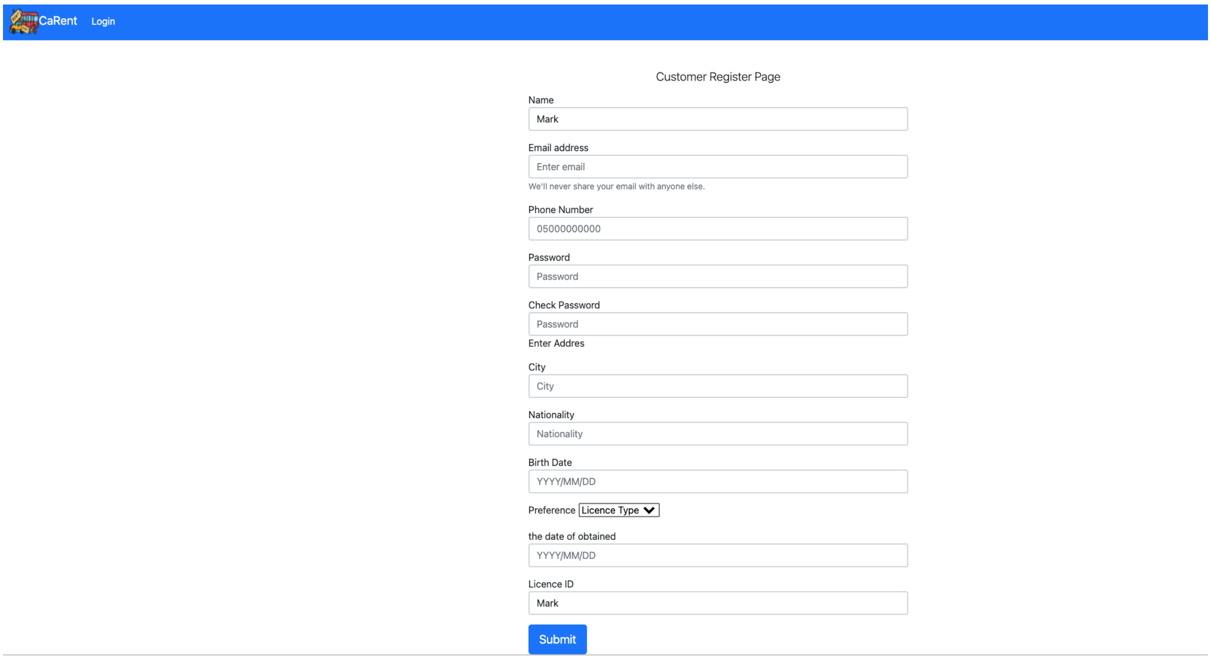
We'll never share your email with anyone else.

Don't have any account? Register

© 2020 Copyright: MDBootstrap.com

SQL Query:
select * from user where email = @email_address and password = @password

SignUp Customer



Customer Register Page

Name: Mark

Email address: Enter email

We'll never share your email with anyone else.

Phone Number: 05000000000

Password: Password

Check Password: Password

Enter Addres

City: City

Nationality: Nationality

Birth Date: YYYY/MM/DD

Preference: Licence Type

the date of obtained: YYYY/MM/DD

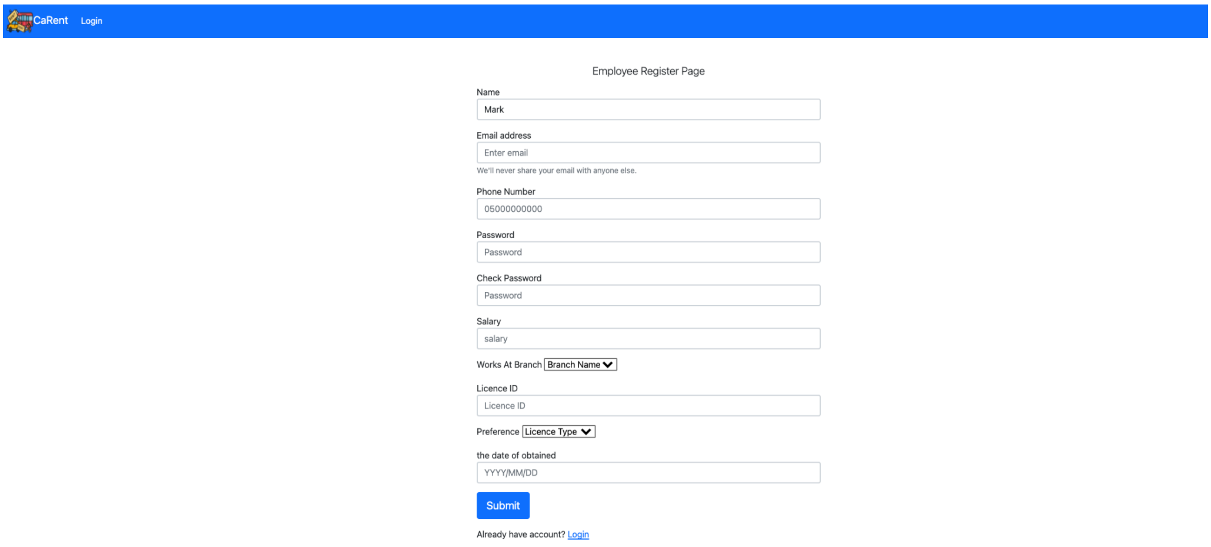
Licence ID: Mark

SQL Queries:
insert into user values (0, @password, @email_address, @phone_number)

insert into driving_license values (0, @license_id, @license_type, @the_date_of_obtained)

insert into customer values (0, @birth_date, @nationality, ‘Normal’, @name)

SignUp Employee



Employee Register Page

Name
Mark

Email address
Enter email
We'll never share your email with anyone else.

Phone Number
05000000000

Password
Password

Check Password
Password

Salary
salary

Works At Branch Branch Name

Licence ID
Licence ID

Preference Licence Type

the date of obtained
YYYY/MM/DD

Already have account? Login

SQL Queries:

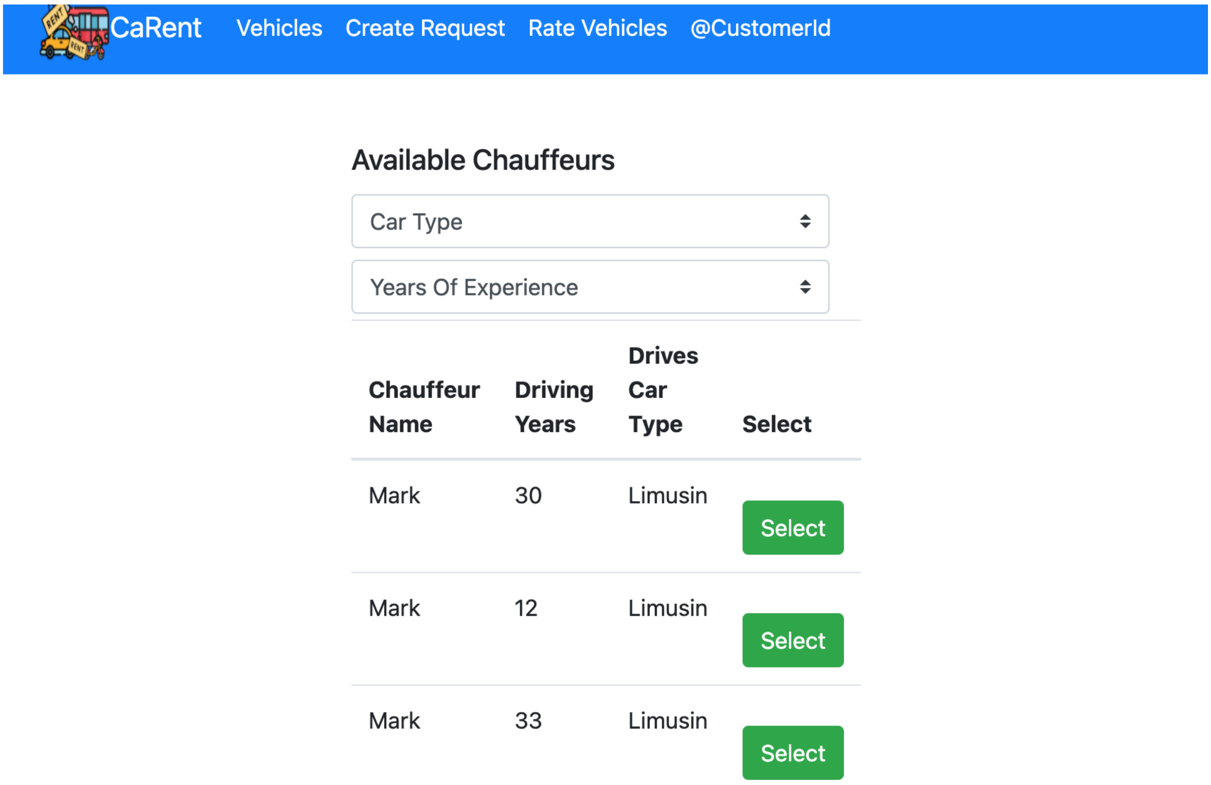
```
insert into user values (0, @password, @email_address, @phone_number)

insert into driving_license values (0, @license_id,@license_type, @the_date_of_obtained)

insert into employee(user_id, salary, employee_name, branch_id)
        select 0, @salary, @name, branch_id
        from branch
        where branch.branch_name = @branch_name
```

Chaeffur System

Listing All Available Chauffeurs



Available Chauffeurs

Chauffeur Name	Driving Years	Drives Car Type	Select
Mark	30	Limusin	Select
Mark	12	Limusin	Select
Mark	33	Limusin	Select

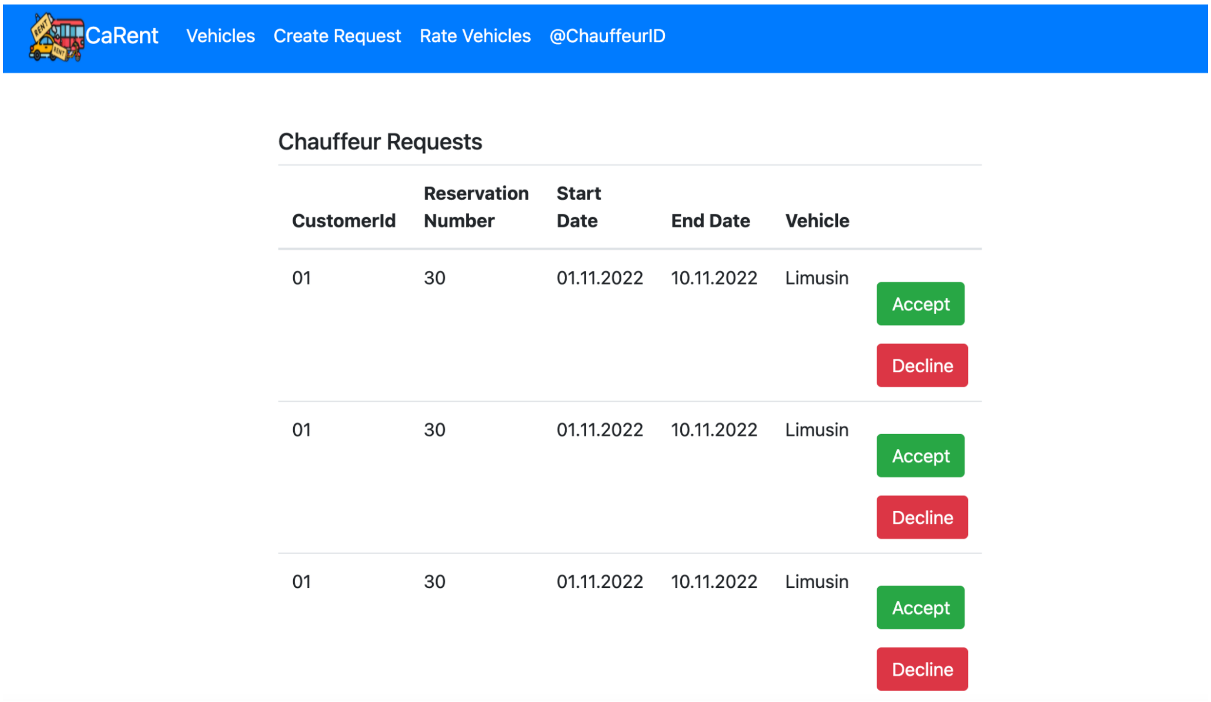
SQL Queries:

-Listing All Chauffeurs
select employee_name, drive_car_type, driving_years from employee natural join chauffeur where chauffeur.user_id not in (select reserved_chauf_id from reservation where reservation.status = ‘on_rent’)

-Applying Filters to Chauffeurs
select employee_name, drive_car_type, driving_years from employee natural join chauffeur where chauffeur.driving_years > @years_of_experience and chauffeur.drive_car_type = @car_type and chauffeur.user_id not in (select reserved_chauf_id from reservation where reservation.status = ‘on_rent’)

When the user selects a chauffeur, it will be sent to the reservation page with the selected chauffeur field and the user interface and query for this page is given at reservation queries.

Chauffeurs Accept or Decline Reservation Requests



Chauffeur Requests

CustomerId	Reservation Number	Start Date	End Date	Vehicle	
01	30	01.11.2022	10.11.2022	Limusin	Accept Decline
01	30	01.11.2022	10.11.2022	Limusin	Accept Decline
01	30	01.11.2022	10.11.2022	Limusin	Accept Decline

SQL Queries:

-Chauffeurs list all requests:

```
select reservation_number, start_date, end_date, model, reserver from reservation natural join vehicle where isChaufAccepted is null and reserved_chauf_id = @chaufferID
```

-If chauffeur accepts the request

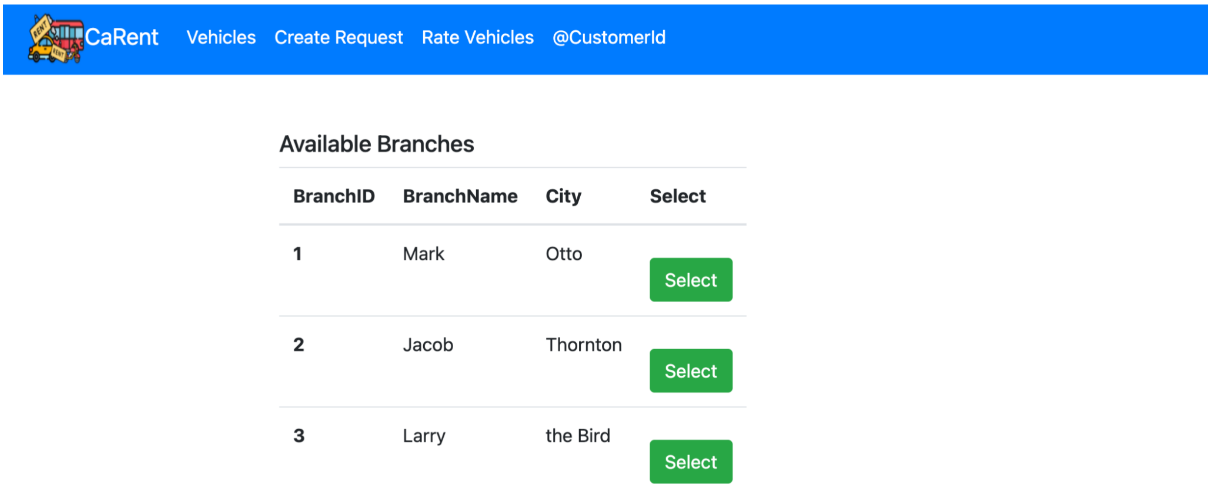
```
update reservation set reservation.isChaufAccepted = ‘true’ where reservation_number = @reservation_number
```

-If chauffeur declines the request

```
update reservation set reservation.isChaufAccepted = ‘false’ where reservation_number = @reservation_number
```

Car Rental

Listing all available Branches



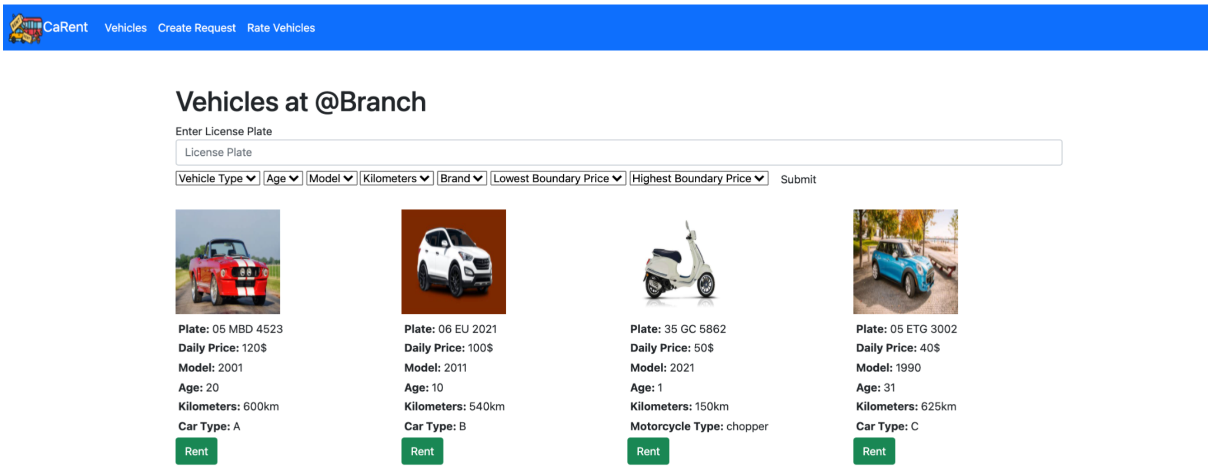
Available Branches

BranchID	BranchName	City	Select
1	Mark	Otto	Select
2	Jacob	Thornton	Select
3	Larry	the Bird	Select

SQL Queries:
-Listing All Branches
select branch_name, city, branch_id from branch

-Selecting One Branch
select branch_name, city, branch_id from branch where branch_id = @branchID

List all available Cars and Apply Filters



Vehicles at @Branch

Enter License Plate

License Plate

Plate: 05 MBD 4523
Daily Price: 120$
Model: 2001
Age: 20
Kilometers: 600km
Car Type: A

Plate: 06 EU 2021
Daily Price: 100$
Model: 2011
Age: 10
Kilometers: 540km
Car Type: B

Plate: 35 GC 5862
Daily Price: 50$
Model: 2021
Age: 1
Kilometers: 150km
Motorcycle Type: chopper

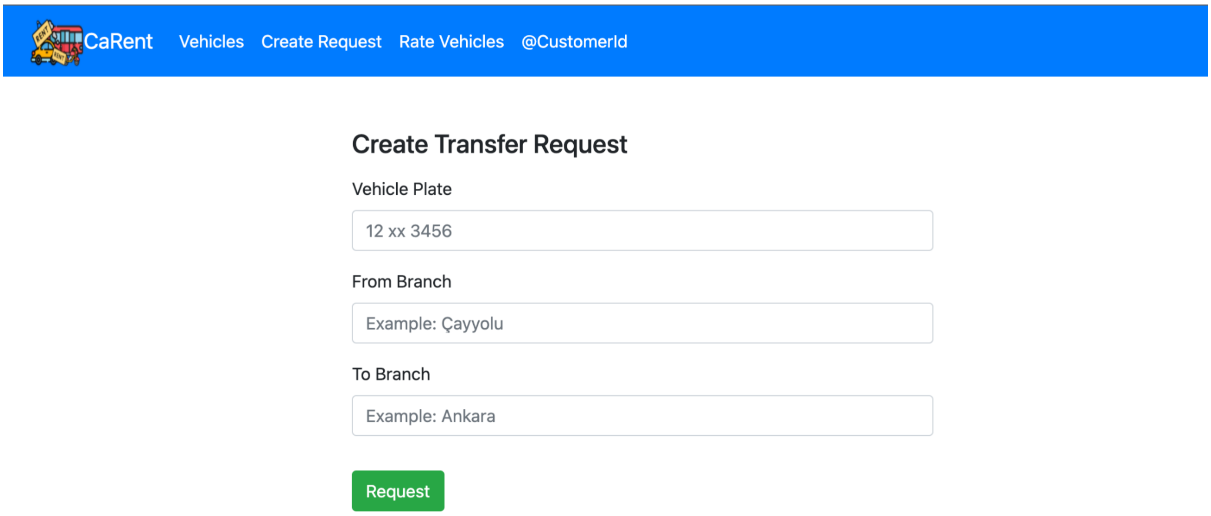
Plate: 05 ETG 3002
Daily Price: 40$
Model: 1990
Age: 31
Kilometers: 625km
Car Type: C

SQL Queries:

-Listing all vehicles at the branch:
select license_plate, model, brand, age, kilometers, daily_rent_price from vehicle natural join model_brand where vehicle.status = ‘available’ and vehicle.branch_id = (select branch_id from branch where branch_name = @branch)

-Filtering Features:
select license_plate, model, brand, age, kilometers, daily_rent_price from vehicle natural join model_brand where vehicle.status = ‘available’ and vehicle.branch_id = (select branch_id from branch where branch_name = @branch) and brand = @brand and age = @age and kilometers = @kilometers and daily_rent_price between @lowest_boundary_price and @highest_boundary_price

-If car doesn't exist in that branch:



Create Transfer Request

Vehicle Plate

12 xx 3456

From Branch

Example: Çayyolu

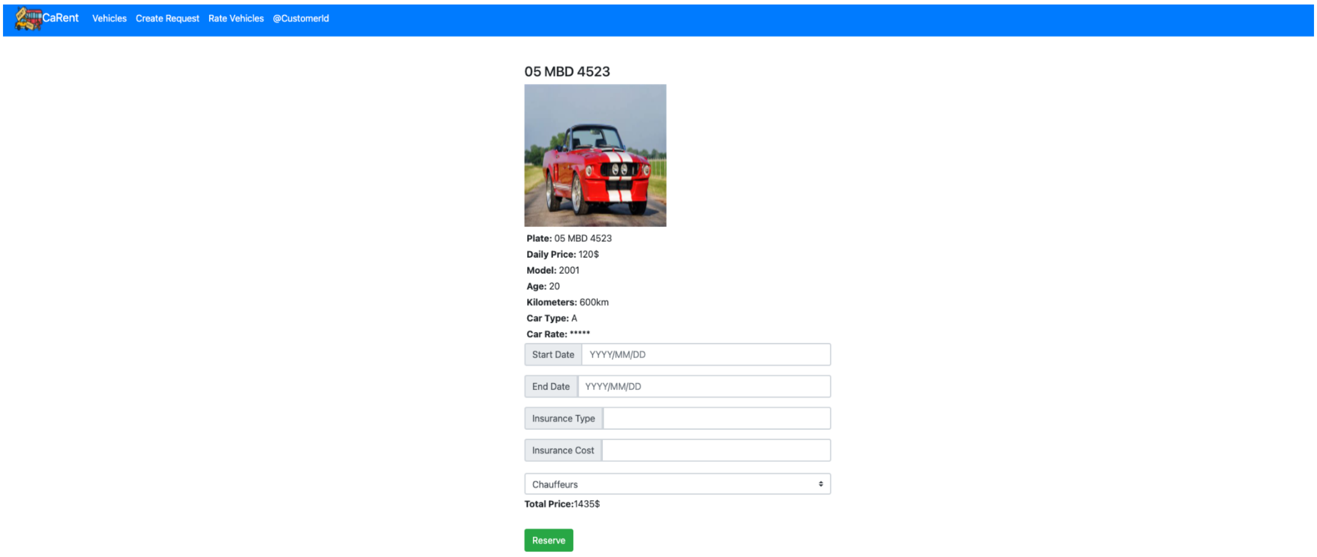
To Branch

Example: Ankara

-Filtering branches where car exists
select branch_id from vehicle natural join model_brand where vehicle.status = 'available' and brand = @brand and age = @age and kilometers = @kilometers and daily_rent_price between @lowest_boundary_price and @highest_boundary_price

-Creating a request
insert into request values(0, @customer_id, @from_branch, @to_branch, @license_plate, null, null, null)

Selecting a Car and Making Reservations



05 MBD 4523

Plate: 05 MBD 4523
Daily Price: 120$
Model: 2001
Age: 20
Kilometers: 600km
Car Type: A
Car Rate: *****

Start Date YYYY/MM/DD

End Date YYYY/MM/DD

Insurance Type

Insurance Cost

Chauffeurs

Total Price:1435$

SQL Queries:

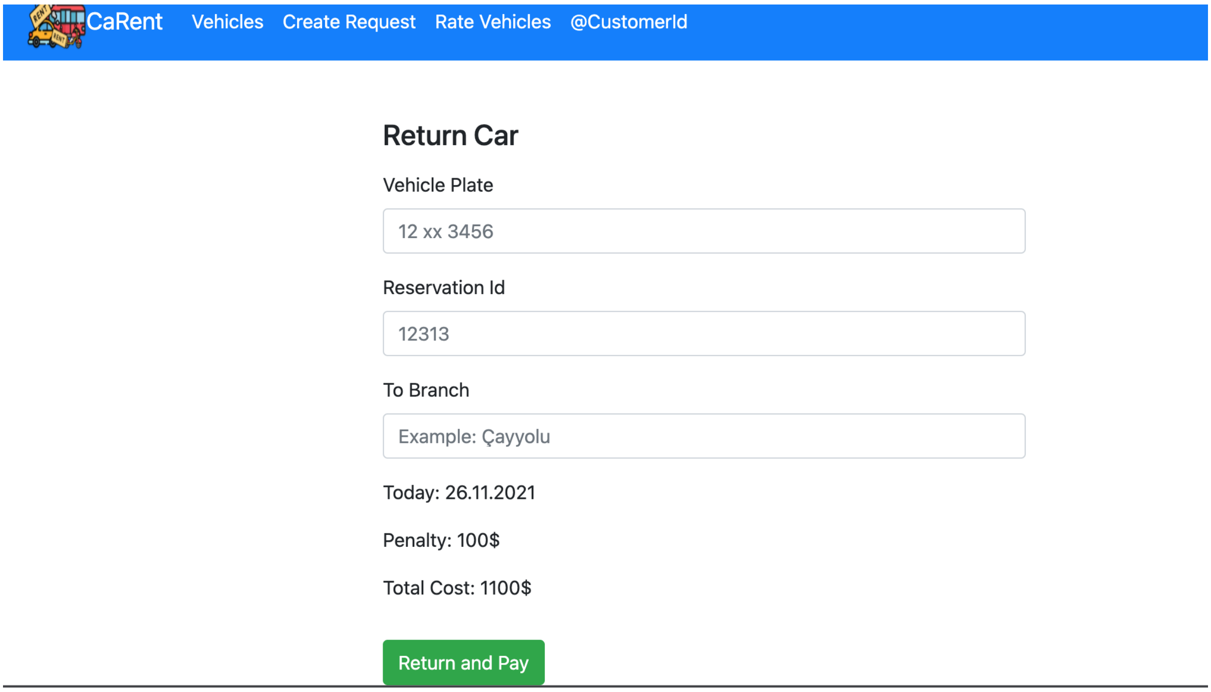
-Selecting:

```
select * from vehicle where license_plate = @plate
```

- Making Reservation:

```
insert into reservation values (0, @start_date, @end_date, @end_date-@start_date,
‘not_accepted’, @total_price, @customer_id, null,null,null,@insurance_type, @plate,
@chaeffurs, null)
```

Return The Car and Pay



Return Car

Vehicle Plate

12 xx 3456

Reservation Id

12313

To Branch

Example: Çayyolu

Today: 26.11.2021

Penalty: 100$

Total Cost: 1100$

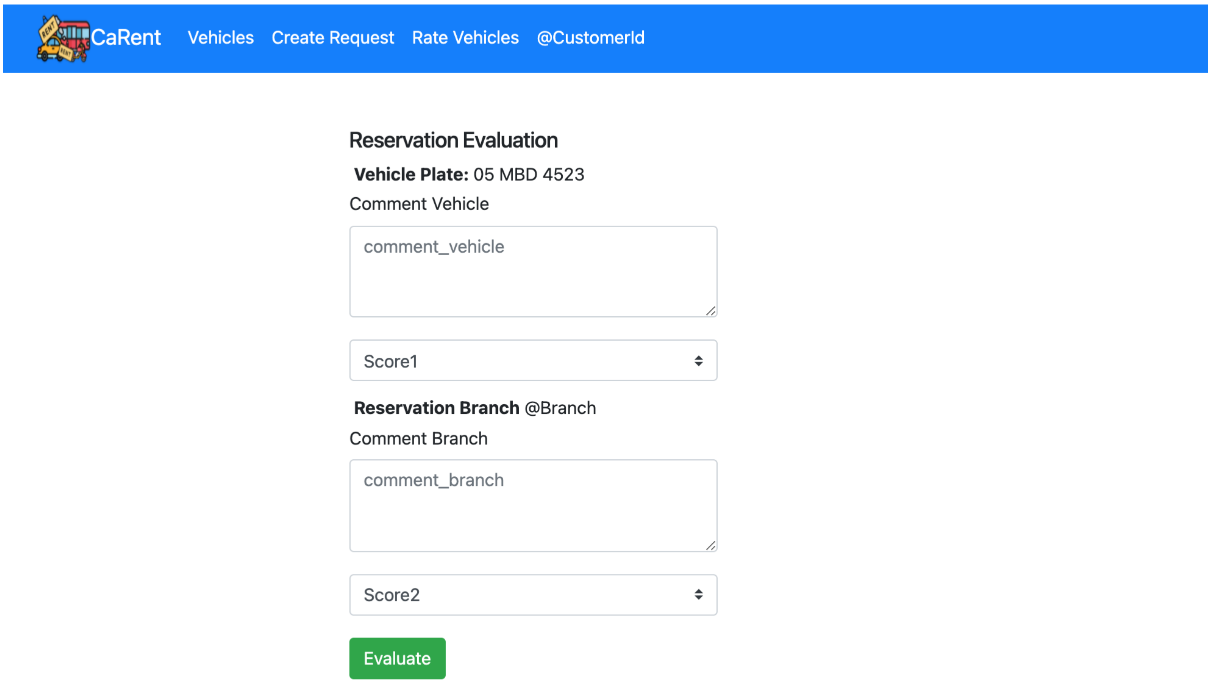
SQL Queries:

-First updating the branch information of the vehicle:
update vehicle set vehicle.branch_id = (select branch_id from branch where branch_name = @to_branch) where vehicle.license_plate = @plate

update reservation set cost = @penalty + cost where reservation_number = @reservation_id

-If it is paid:
update reservation set status = ‘paid’ where reservation_number = @reservation_id

Giving Feedback



Reservation Evaluation

Vehicle Plate: 05 MBD 4523

Comment Vehicle

comment_vehicle

Score1

Reservation Branch @Branch

Comment Branch

comment_branch

Score2

SQL Queries:

insert into vehicle_rate(@customer_id, @vehicle_plate, @comment_vehicle, @score1)

insert into branch_rate(@customer_id, @branch, @comment_branch, @score2)