CS 353 Database Systems

Final Report

Car Rental System

Group No 17

Emre Caniklioğlu - 21803577

Celal Berke Can - 21702886

Ege Demirkırkan - 21802482

Berk Saltuk Yılmaz - 21903419

A Brief Description

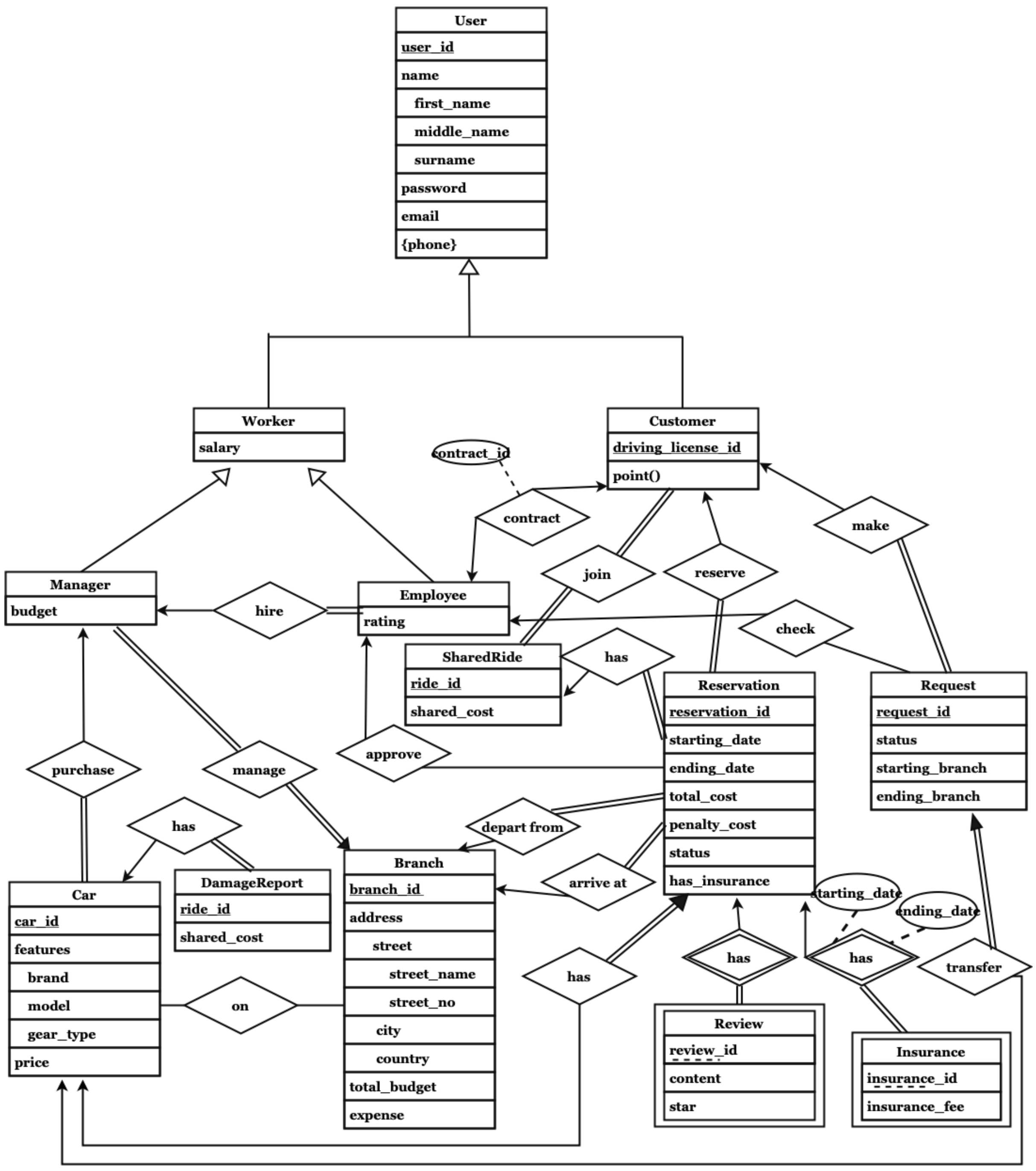
Our project implements a website for a car rental system. This system has a user system which has three different roles: customer, employee and manager. Each role has different sets of pages to interact with.

A customer can login or register to the system. At the home page, the customer can select a starting branch and see the cars at that branch, then can make a reservation or request a car to be transferred to his/her city. After making a reservation, if the reservation is approved by an employee, the customer can return the car to a specific branch. Then this customer pays for this reservation by adding money to her/his account and makes a review for the whole process. These reviews are visible for other customers to have an idea about cars. Also this customer can share rides with other customers.

An employee is basically responsible for handling requests and reservations. When a request or reservation is submitted, the employee from the requested branch approves or declines it. After a customer returns a car, an employee can report whether a car is damaged and charge additional costs.

A manager is responsible for buying new cars, hiring new employees. Each manager manages a branch and has a budget. Also every user can change their password.

Final ER Diagram



Advanced DB Components

Reports:

Report for Customer:

Shows customer’s summary statistics.

```
WITH number_of_reservation_by_customer AS (
   SELECT u.user_id, count(DISTINCT reservation_id) AS total_reservation,
          sum(r.total_cost) AS total_paid
   FROM users AS u
   JOIN reservation r on u.user_id = r.customer_id
   WHERE r.status IN (600)
   GROUP BY u.user_id
)
SELECT u.user_id, u.first_name, nor.total_reservation, nor.total_paid FROM users
AS u
LEFT JOIN number_of_reservation_by_customer AS nor ON nor.user_id = u.user_id;
```

Report for Employee:

Shows employee’s summaries of all tasks completed before.

```
WITH number_of_reservations_validated_by_employee AS (
   SELECT r.validated_by,
       count(DISTINCT r.reservation_id) AS total_reservation_validated
   FROM employee_reservation AS r
   GROUP BY r.validated_by
),
    number_of_request_validated_by_employee AS (
        SELECT req.validated_by,
           count(DISTINCT req.request_id) AS total_request_validated
        FROM employee_request AS req
        GROUP BY req.validated_by
    )
SELECT u.user_id, u.first_name, nor.total_reservation_validated,
noreq.total_request_validated FROM users AS u
RIGHT JOIN number_of_reservations_validated_by_employee AS nor ON
nor.validated_by = u.user_id
RIGHT JOIN number_of_request_validated_by_employee AS noreq ON
noreq.validated_by = u.user_id;
```

Report for Manager:

Shows manager’s summaries of all tasks completed before.

```
WITH manager_amount_of_cars_bought_and_spending AS (
   SELECT
       c.manager_id,
       count(DISTINCT c.car_id) AS amount_of_cars_bought,
       sum(c.price) AS total_spending_for_car
   FROM car AS c
   GROUP BY c.manager_id
),
   amount_of_employees_hired AS (
    SELECT
        emp.manager_id,
        count(DISTINCT emp.user_id) AS total_employee_hired
    FROM employee AS emp
    GROUP BY emp.manager_id
)
SELECT u.user_id, u.first_name, macb.amount_of_cars_bought,
macb.total_spending_for_car, aoeh.total_employee_hired  FROM users as u
RIGHT JOIN manager_amount_of_cars_bought_and_spending AS macb ON macb.manager_id
= u.user_id
RIGHT JOIN amount_of_employees_hired AS aoeh ON aoeh.manager_id = u.user_id;
```

Queries including LIKE and BETWEEN statements:

```
SELECT * from reservation where starting_date between @date1 and @date2
```

This query returns the reservations between two given dates.

```
SELECT * from branch where branch_name LIKE ‘A%’
```

This query returns the branch names starting with a specific letter.

Query Including GROUP BY and HAVING CLAUSE:

```
SELECT user_id, sum(totalCost) from customer natural join reservation group by
user_id having user_id = @user_id
```

This query returns total money spent by a specific customer.

Views:

```
create view reservation_with_car as SELECT * from reservation natural join car
where user_id = @user_id
```

This view is used at my reservations page, it directly joins reservation and car pages. By this, we are able to display a reservation with the information of the car that belongs to the reservation.

Triggers, Constraints, Stored Procedures, Secondary Indices:

We have a trigger that functions when a reservation's status changes. When a reservation’s status is updated to paid status, automatically totalCost is deducted from authorized user’s balance.

We have a constraint to check the role of users added to users table, if role of user to be added is not in the set {‘CUSTOMER’, ‘EMPLOYEE’, ‘MANAGER’}, this user cannot be added to this table.

We have a stored procedure SELECTAllBranches that returns all branches. We use it when we need to list all branches.

In PostgreSQL, every indices besides primary indices are secondary indices. Hence, we did not need to implement such a feature.

Final List Of Tables

1. Users

Relational Model:

users(user_id, first_name, middle_name, surname, password, email, role)

Primary Key:

Primary Key: user_id

2. Worker

Relational Model:

worker(user_id, salary)
FK: user_id references users(user_id)

Primary Key:

Primary Key: user_id

3. Manager

Relational Model:

manager(user_id, budget, branch_id)
FK: user_id references users(user_id)
FK: manages references branch(branch_id)

Primary Key:
Primary Key: user_id

4. Employee

Relational Model:

employee(user_id, rating, manager_id)

FK: user_id references users(user_id)
FK: manager_id references manager(user_id)

Primary Key:

Primary Key: user_id

5. Customer

Relational Model:

customer(<u>user_id</u>, driving_license_id, point)

Primary Key:

Primary Key: user_id

6. Reservation

Relational Model:

reservation(<u>reservation_id</u>, starting_date, ending_date, total_cost, penalty_cost, status, customer_id, depart_from, arrive_at)

FK: customer_id references customer(user_id)
FK: depart_from references branch(branch_id)
FK: arrive_at references branch(branch_id)

Primary Key:

Primary Key: reservation_id

7. Car

Relational Model:

car(<u>car_id</u>, brand, model, gear_type, price, manager_id,price_per_day)
FK: manager_id references manager(user_id)

Primary Key:
Primary Key: car_id

8. Branch

Relational Model:

branch(branch_id, branch_name, street_name, street_no, city, country, total_budget, expense)

Primary Key:

Primary Key: branch_id

9. Review

Relational Model:

review(review_id,reservation_id, content, star)
FK: reservation_id references reservation(reservation_id)

Primary Key:

Primary Key: {reservation_id, review_id}

10. Request

Relational Model:

request(request_id, start_branch, dest_branch, status)
FK: start_branch references branch(branch_id)
FK: dest_branch references branch(branch_id)

Primary Key:

Primary Key: request_id

11.Insurance

Relational Model:

insurance(reservation_id, insurance_id, starting_date, ending_date, insurance_fee)
FK: reservation_id references reservation(reservation_id)

Primary Key:

Primary Key: {reservation_id, insurance_id}

12. User Phone

Relational Model:

user_phone(user_id, phone)

Primary Key:

Primary Key: {user_id, phone}

13. Damage Report

Relational Model:

damage_report(report_id, car_id, title, content, cost)

Primary Key:

Primary Key: {report_id, car_id}

14. Shared Ride

Relational Model:

shared_ride(ride_id, reservation_id)
FK: reservation_id references Reservation(reservation_id)

Candidate Keys and Primary Key:

Candidate Key: ride_id
Primary Key: ride_id

15. Shared Ride Customer

Relational Model:

customer_shared_ride(ride_id, passenger_id, start_branch)
FK: passenger_id references customer(user_id)
FK: ride_id references shared_ride(ride_id)
FK: start_branch references branch(branch_id)

Primary Key:

Primary Key: {ride_id, passenger_id}

16. Request Car

Relational Model:

request_car(car_id, request_id)
FK: car_id references car(car_id)
FK: request_id references request(request_id)

Primary Key:

Primary Key: {car_id, request_id}

17. Car Reservation

Relational Model:

car_reservation(car_id, reservation_id)
FK: car_id references car(car_id)
FK: reservation_id references reservation(reservation_id)

Primary Key:

Primary Key: {car_id, reservation_id}

18. Car Branch

Relational Model:

car_branch(<u>car_id, branch_id</u>)
FK: car_id references car(car_id)
FK: branch_id references branch(branch_id)

Primary Key:

Primary Key: {car_id, branch_id}

Implementation Details

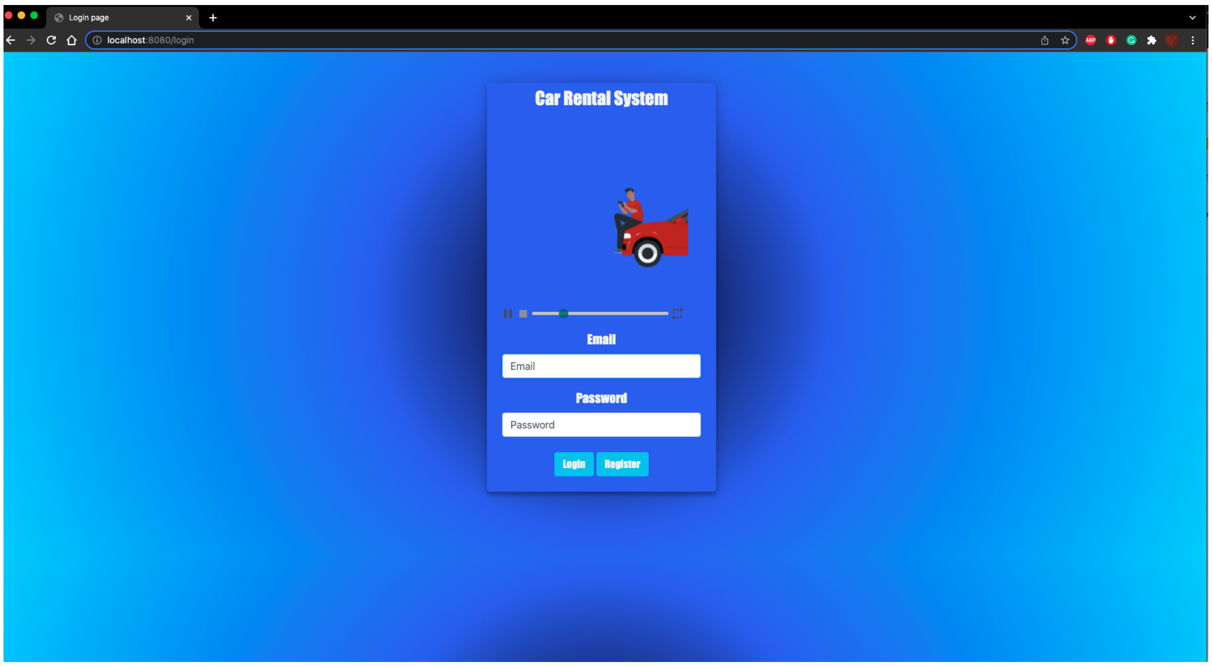
For our web server’s main infrastructure we chose to use Spring Security for the custom URL based authentication, Spring MVC for creating a reliable page model flow and Spring JDBC for executing SQL queries from our server. To solve our HTML templating problem and to dynamically create HTML pages. We chose to use Thymeleaf’s Spring implementation together with the rest of the infrastructure provided by the framework. Due to its simplicity and its powerful component style library we chose to use bootstrap together with some custom CSS classes. For our database we choose PostgreSQL due its high scalability, its reliability and its customizability. For our high level business logic we decided to follow Controler, Service, Repository pattern where each page has its own controller which contains a GET mapping and if necessary a POST mapping. Controllers only handle the incoming request and each controller is connected to a service where the service connects to the required repositories and implements the needed business logic for the given page. One of the first problems we encountered was to figure out how we would design a system where it would be easy for us to share data across pages without using javascript. After some research and some trials we came with some possible solutions to our problem. And decided that we would use Spring’s http session for user related datas together with some important information which need to be stored throughout the session. URL query parameters for forwarding simple information to other pages and hidden input values for populating forms without showing the data to the user. Another problem we encountered was the different access requirements to the pages and its implementation using PostgreSQL and Spring security. Berk Saltuk Yılmaz and Emre Caniklioğlu researched and learned more about the different authority and roles systems used by the others and implemented the security system. Afterwards we divided the main flow of the pages and started to work where Emre Caniklioğlu worked on the branch selection, car selection, make reservation - request, car buy and registration pages. Berk Saltuk Yılmaz worked on the profile details, user’s reservations page, changing user’s password, car return, pay for reservation, and making review + adding balance to user (with Emre Caniklioğlu). Berke Can worked on the employee hire page for managers, our additional feature ride sharing page implementation. Ege Demirkırkan worked on the login page and employee functionalities such as reservation check, request check, damage report check.

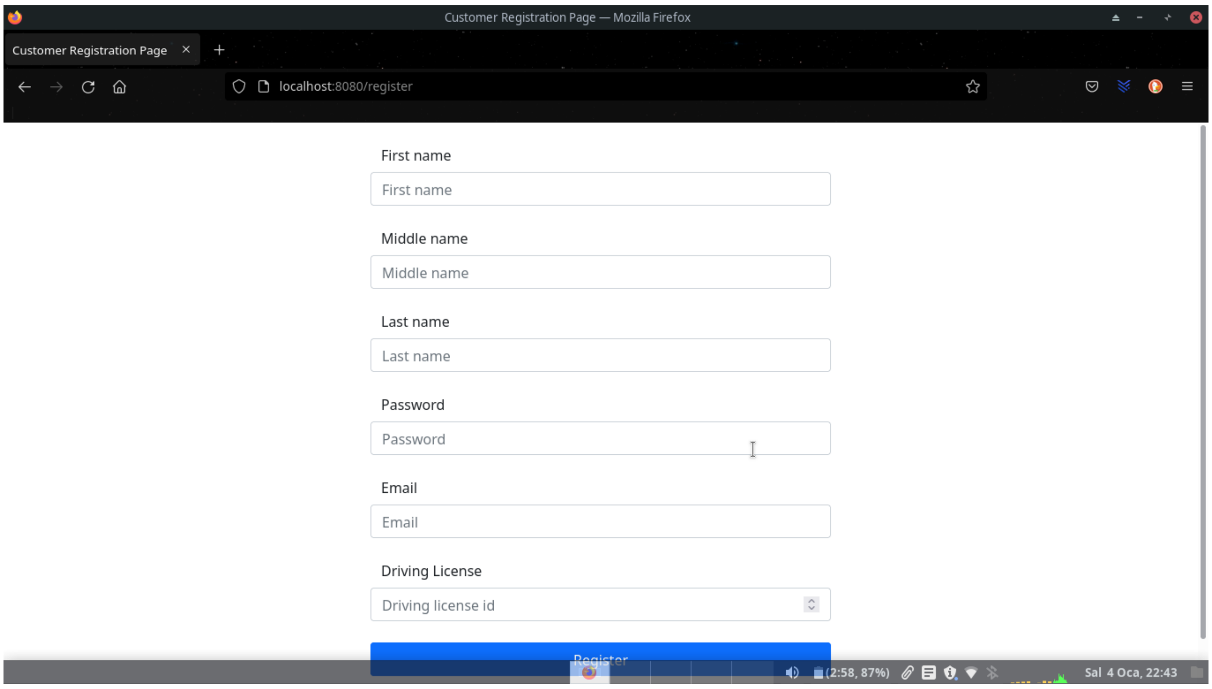
Advanced Database Features Used

User Manual

Login & Register Page

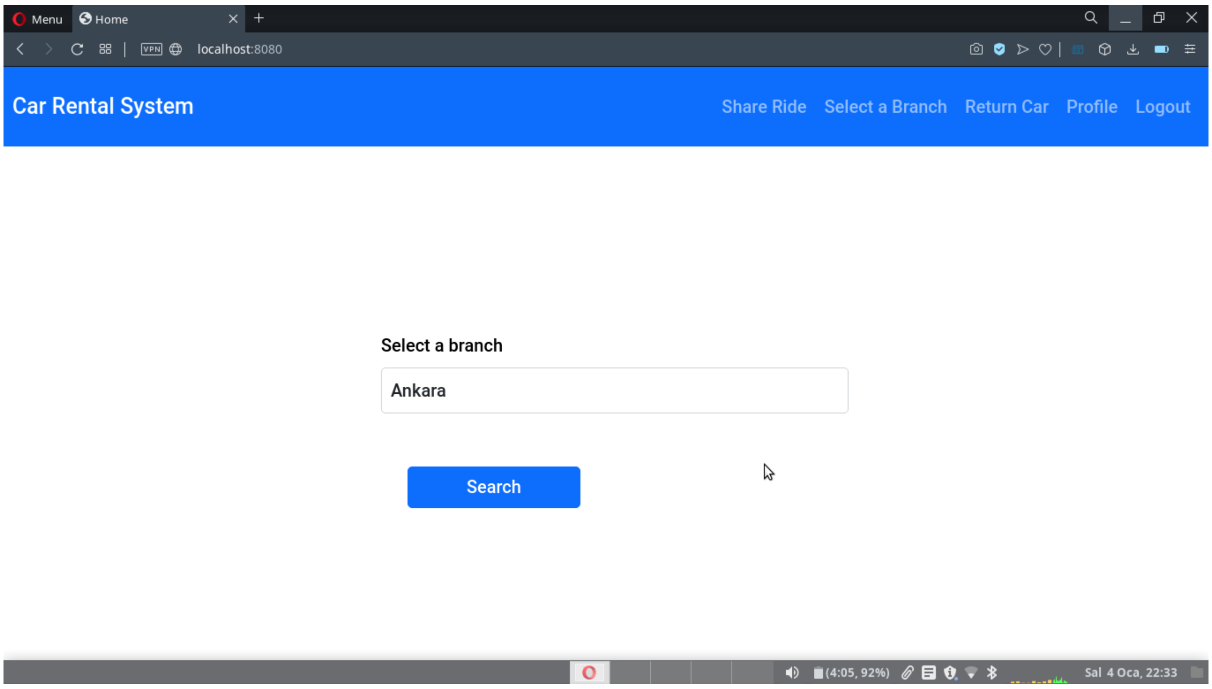
Users can login with their email and password, or can register by filling a registration form.





Select Branch Page

This page is the entry point to our web based application. It gives an option input for selecting which branch the customer wants to reserve a car from.

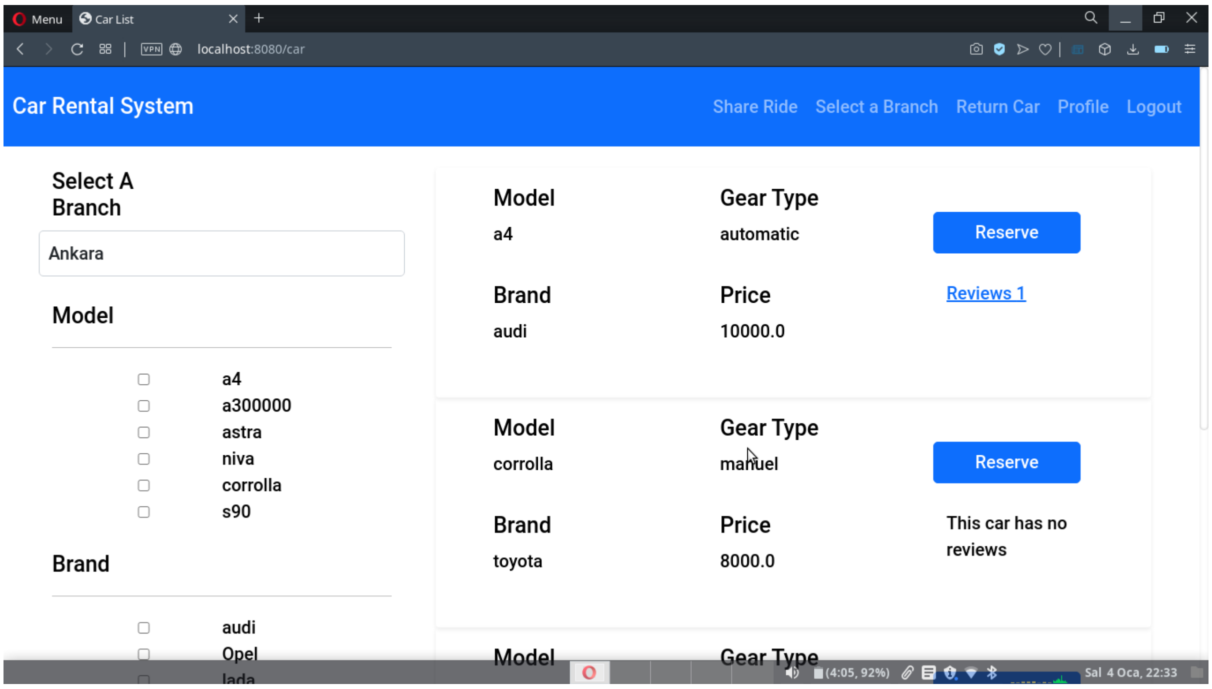


Select a branch

Ankara

Car List Page

After selecting a branch in the select branch page customer is forwarded to this page for browsing through the cars listed at the right handside of the page. On the left hand side of the page there is a form filtering the car and it can be submitted using the search button at the end of the form. From top to button forms logic is as follows; branch option enables customers to change the previously selected branch and search for other cars in different branches. model, brand and gear type check boxes are connected to each other using logical and operator and as individual filters they are inclusive meaning if a customer chooses 'audi' and 'toyota' together and no other filter options from model, brand or gear type is chosen than the car list on the right will be filtered with the constraint that car's model is either 'audi' or 'toyota'. But if there is another option selected from another checkbox group, for example 'audi' and 'toyota' is selected from the models. And 'manual', 'automatic' are selected from the gear types. List on the right side will be filtered with the constraint that the car's model is either ('audi' or 'toyota') and its gear type is ('manual' or 'automatic'). If there are no cars in the selected branch which satisfy the required filter options. Then it will search all of the cars in all of the branches and display them with the option of requesting to the specified branch. If there are cars which satisfy the filtering options it will display them with the reservation option rather than the request option. And under each car if it has a review there is a link for navigating to a page where reviews are displayed.


Car Rental System

Select A Branch

Ankara

Model

- a4
- a300000
- astra
- niva
- corrolla
- s90

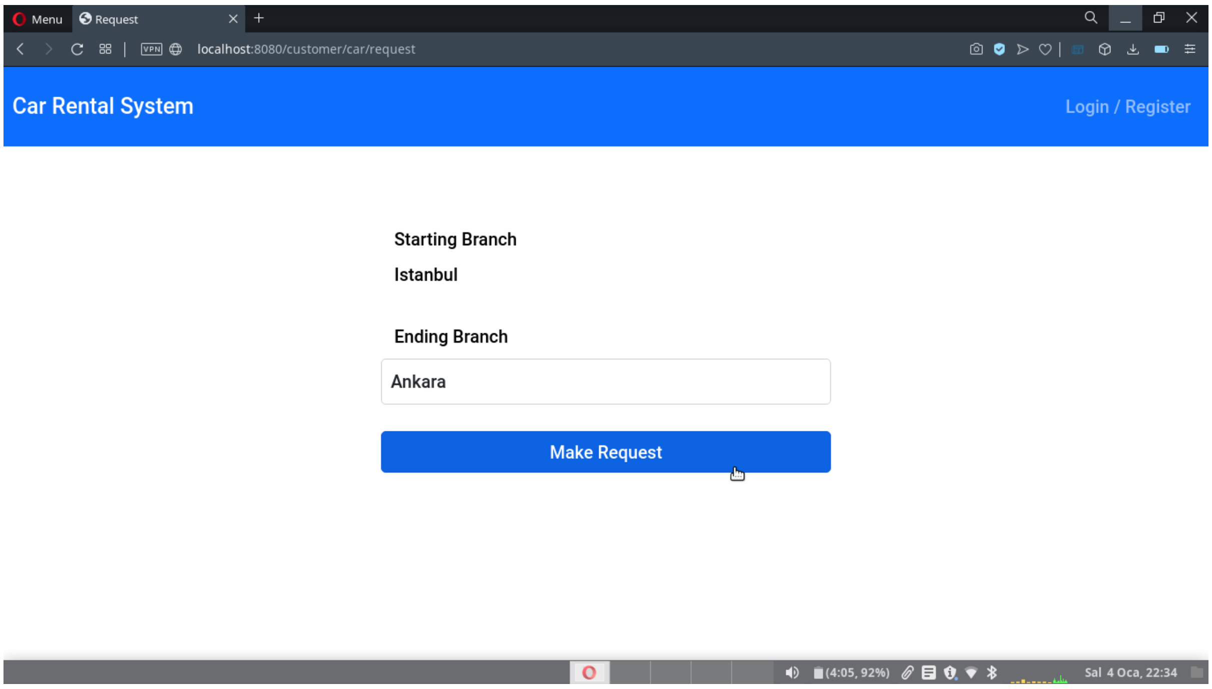
Brand

- audi
- Opel
- lada

Model	Gear Type	Reserve
a4	automatic	
Brand	Price	Reviews 1
audi	10000.0	
Model	Gear Type	Reserve
corrolla	manuel	
Brand	Price	This car has no reviews
toyota	8000.0	

Request Page

This page allows users to specify a branch and request the car from its current branch to the specified branch.


Car Rental System

Starting Branch

Istanbul

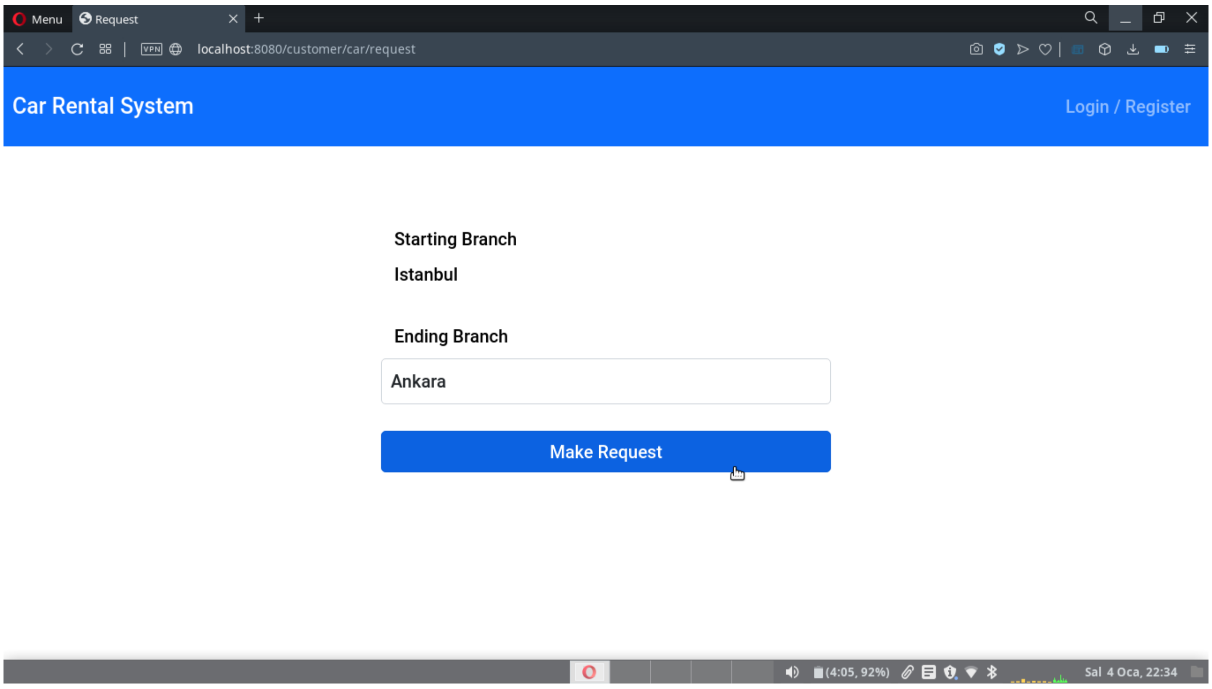
Ending Branch

Ankara

Make Request

Reservation Page

This page allows users to enter the required information and allows them to create a new record of reservations.


Request

localhost:8080/customer/car/request

Car Rental System

Login / Register

Starting Branch

Istanbul

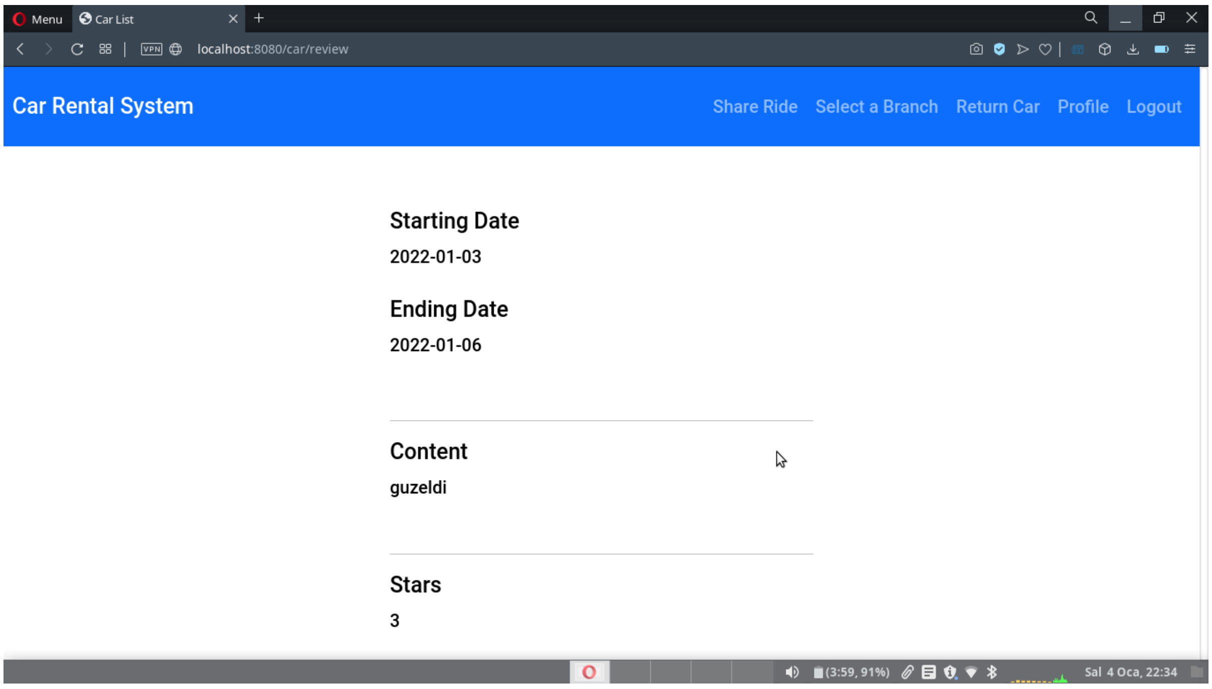
Ending Branch

Ankara

Make Request

Review Page

This page displays all of the previous reviews for the selected car.


Car List

localhost:8080/car/review

Car Rental System

Share Ride Select a Branch Return Car Profile Logout

Starting Date

2022-01-03

Ending Date

2022-01-06

Content

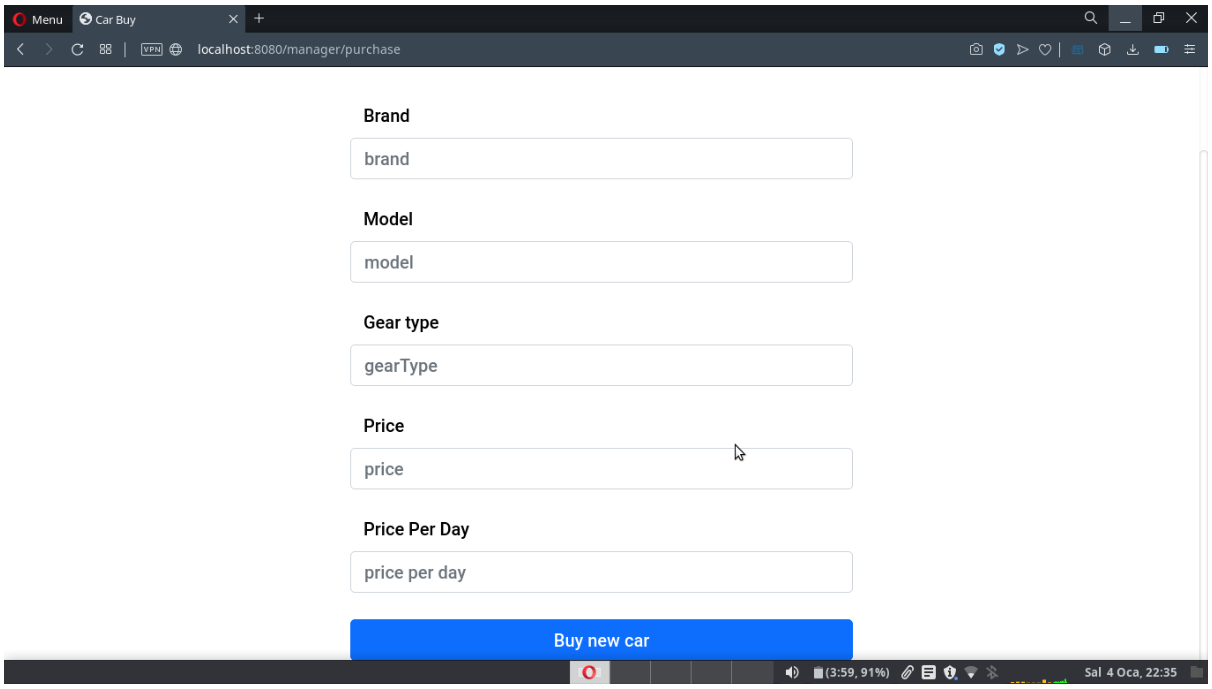
guzeldi

Stars

3

Car Buy Page

Using this page a user who is logged in as a manager can fill the form and buy a new car to his or her branch.



Brand

brand

Model

model

Gear type

gearType

Price

price

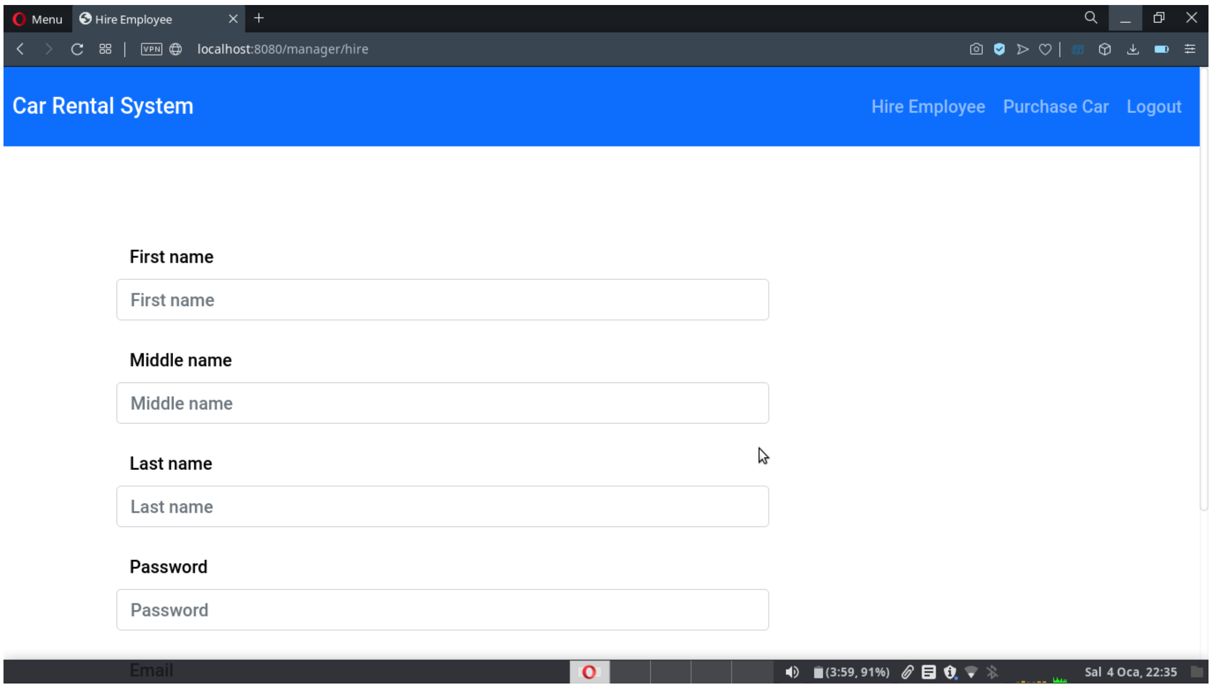
Price Per Day

price per day

Buy new car

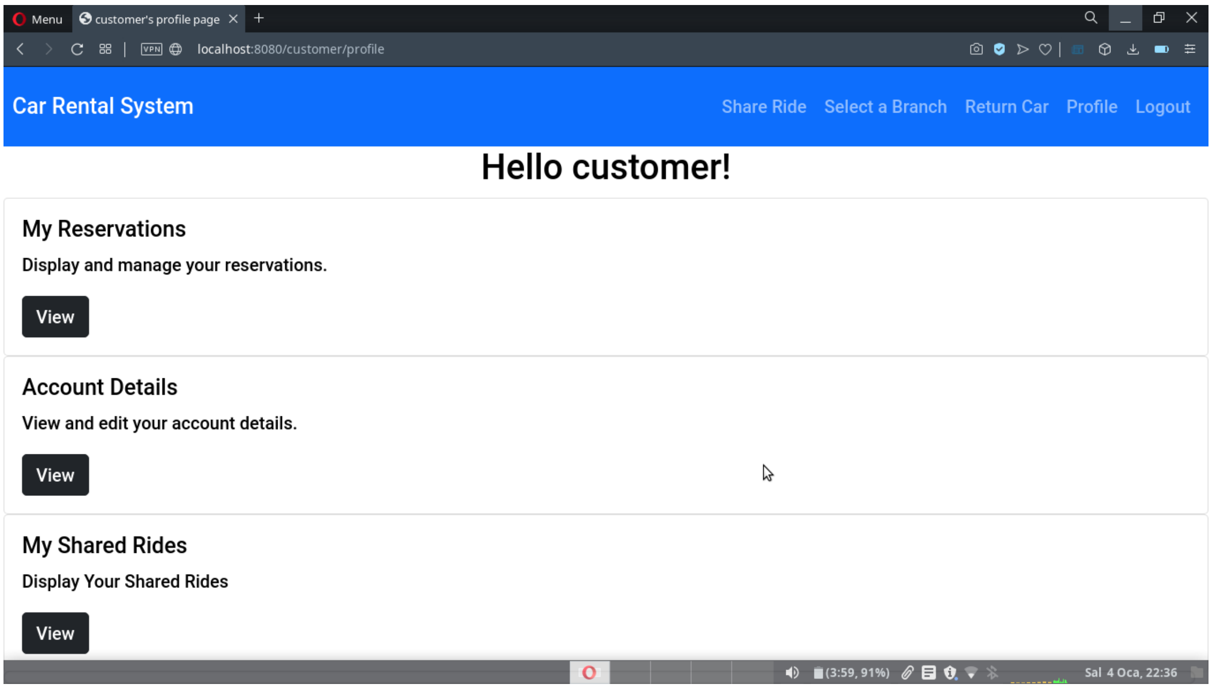
Employee Hire Page

Using this page a manager can fill the form and hire a new employee to the branch that he or she works in.



Profile Page

From this page customers can access their reservations, their shared rides and their user information.



Hello customer!

My Reservations

Display and manage your reservations.

View

Account Details

View and edit your account details.

View

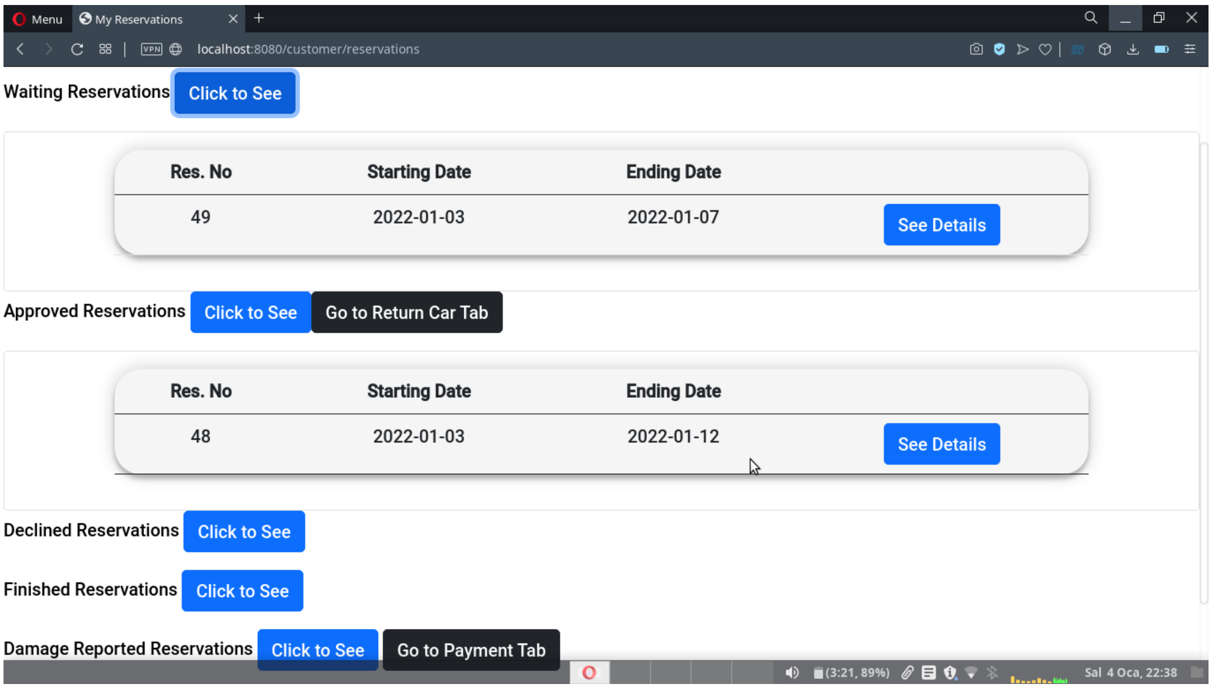
My Shared Rides

Display Your Shared Rides

View

My Reservations Tab

At this page customers can see the details of all their reservations grouped according to reservation status. From this page customers can access the reservation payment page for their damage-reported and nondamage reported reservations, they also can access the return car page for their approved reservations. Finally, customers can make a review for their paid reservations from this page.



Waiting Reservations Click to See

Res. No	Starting Date	Ending Date	
49	2022-01-03	2022-01-07	See Details

Approved Reservations Click to See Go to Return Car Tab

Res. No	Starting Date	Ending Date	
48	2022-01-03	2022-01-12	See Details

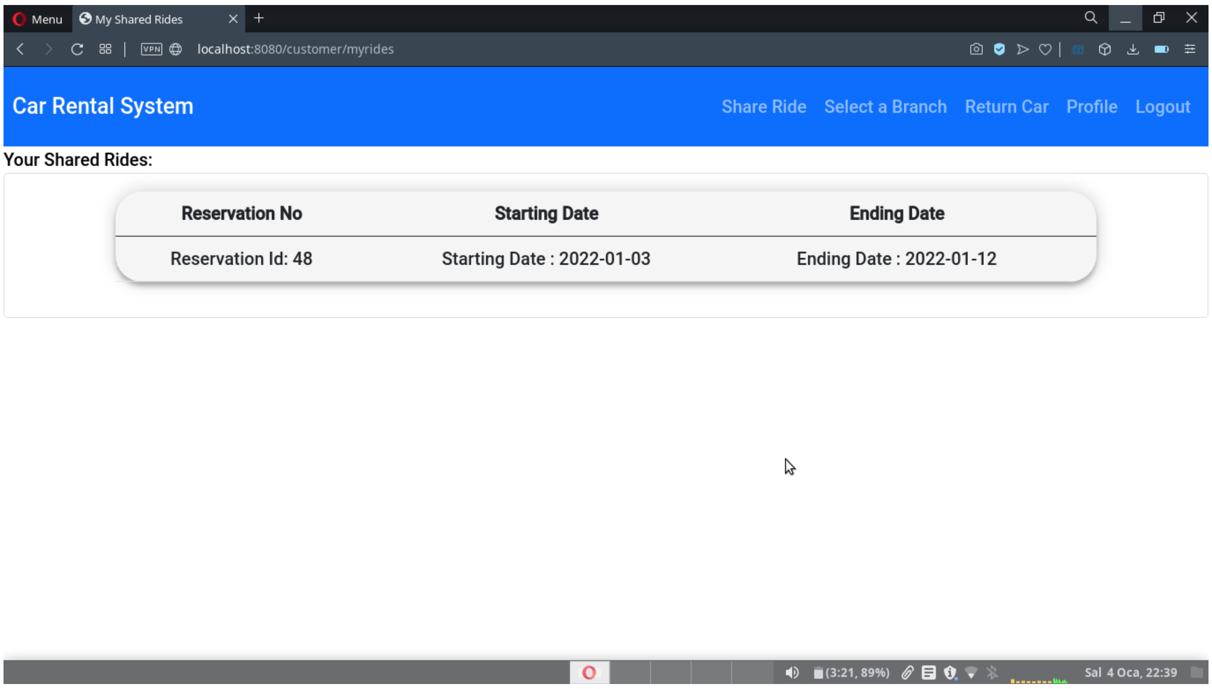
Declined Reservations Click to See

Finished Reservations Click to See

Damage Reported Reservations Click to See Go to Payment Tab

My Shared Rides Tab

At this page customer can see the share ride information

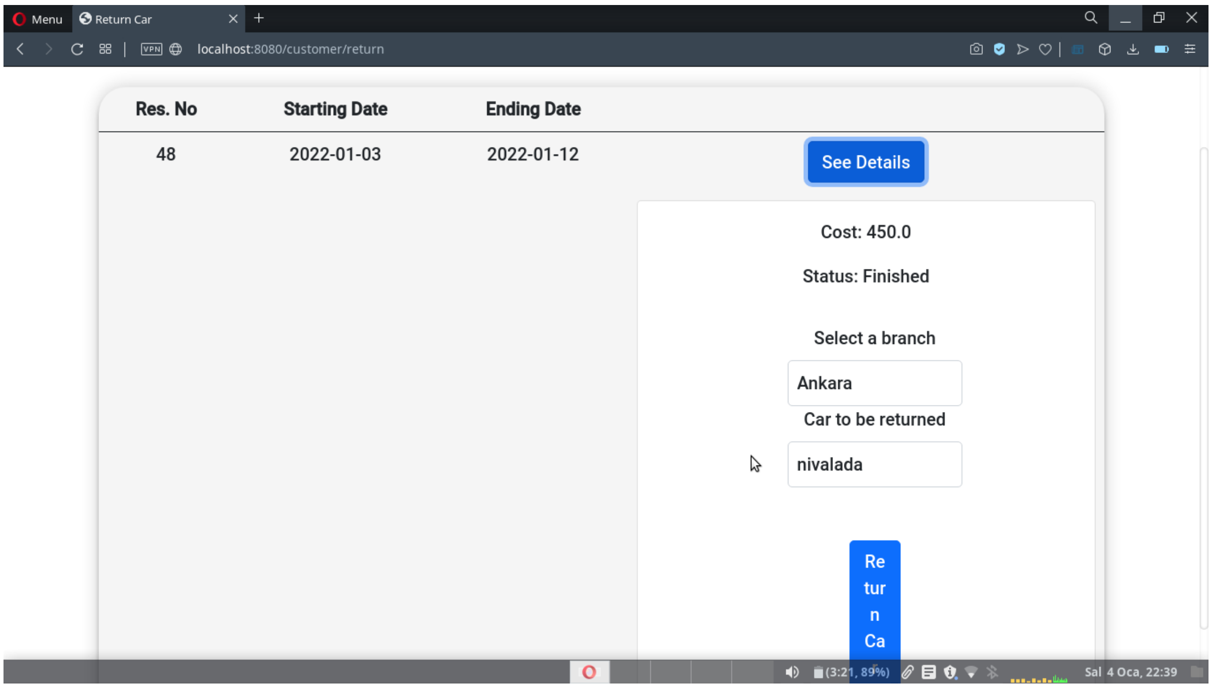


Your Shared Rides:

Reservation No	Starting Date	Ending Date
Reservation Id: 48	Starting Date : 2022-01-03	Ending Date : 2022-01-12

Return Car

At this page customers can see their approved reservations, display their details and return the car of these approved reservations to the branch that they chose.



Res. No	Starting Date	Ending Date
48	2022-01-03	2022-01-12

Cost: 450.0

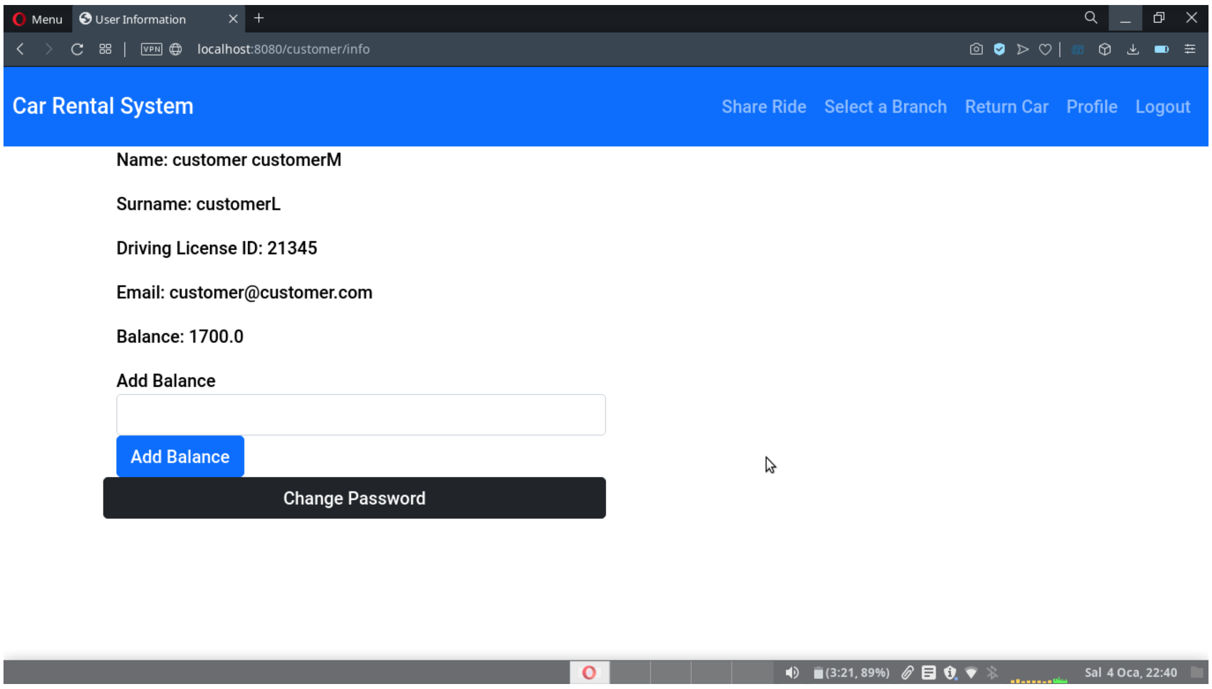
Status: Finished

Select a branch: Ankara

Car to be returned: nivalada

User Info Tab

At this page, customers can display their user information together with their customer information including driving license ID and balance. They also can access the change password tab from this page and also by specifying an amount they can put up some money to their accounts.



Name: customer customerM

Surname: customerL

Driving License ID: 21345

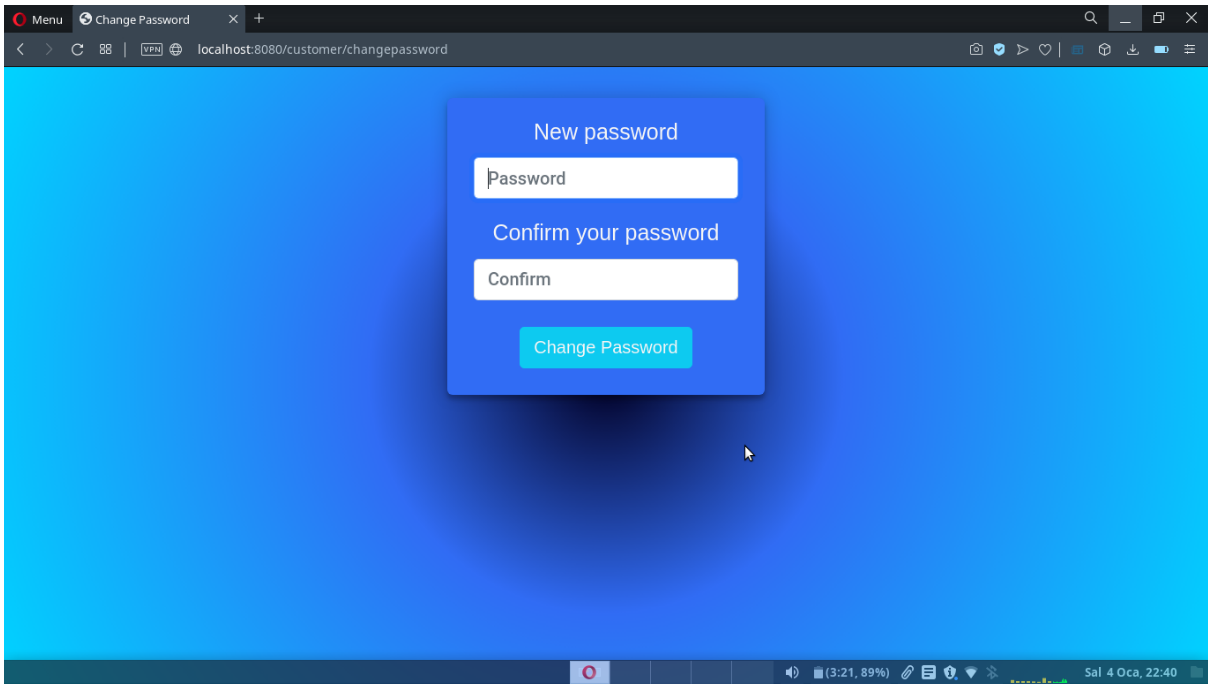
Email: customer@customer.com

Balance: 1700.0

Add Balance

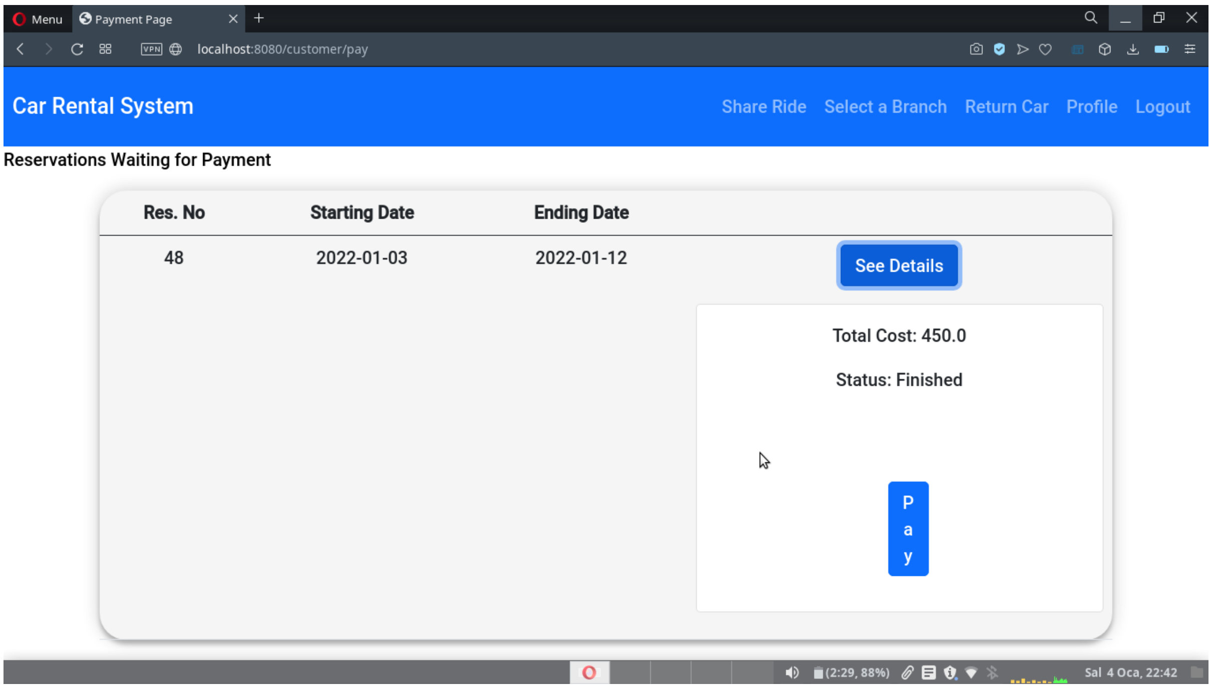
Change Password Tab

At this page users can change their passwords. If the two passwords entered in both password fields match, the password change operation is successful.



Payment Page

At this page, damage reported and nondamage reported reservations are listed and the customer can pay for one of these reservations if he/she has enough balance. Note that we specify total cost as # of days * daily price of cars + damage costs if there are any.



Reservations Waiting for Payment

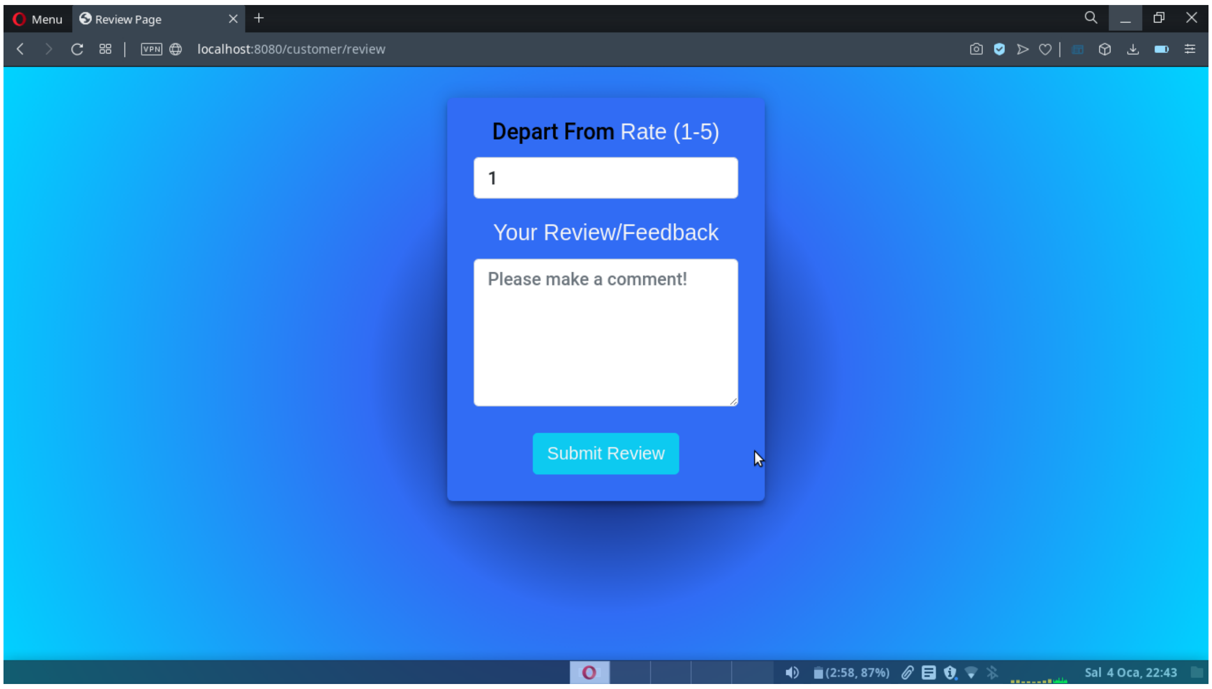
Res. No	Starting Date	Ending Date	
48	2022-01-03	2022-01-12	See Details

Total Cost: 450.0

Status: Finished

Make Review Page

After selecting a paid reservation to review at the “My reservations” tab, customers can rate the car from 1 to 5 and make a comment about the car and the reservation experience, then of course submit this form by clicking the “Review” tab.



Depart From Rate (1-5)

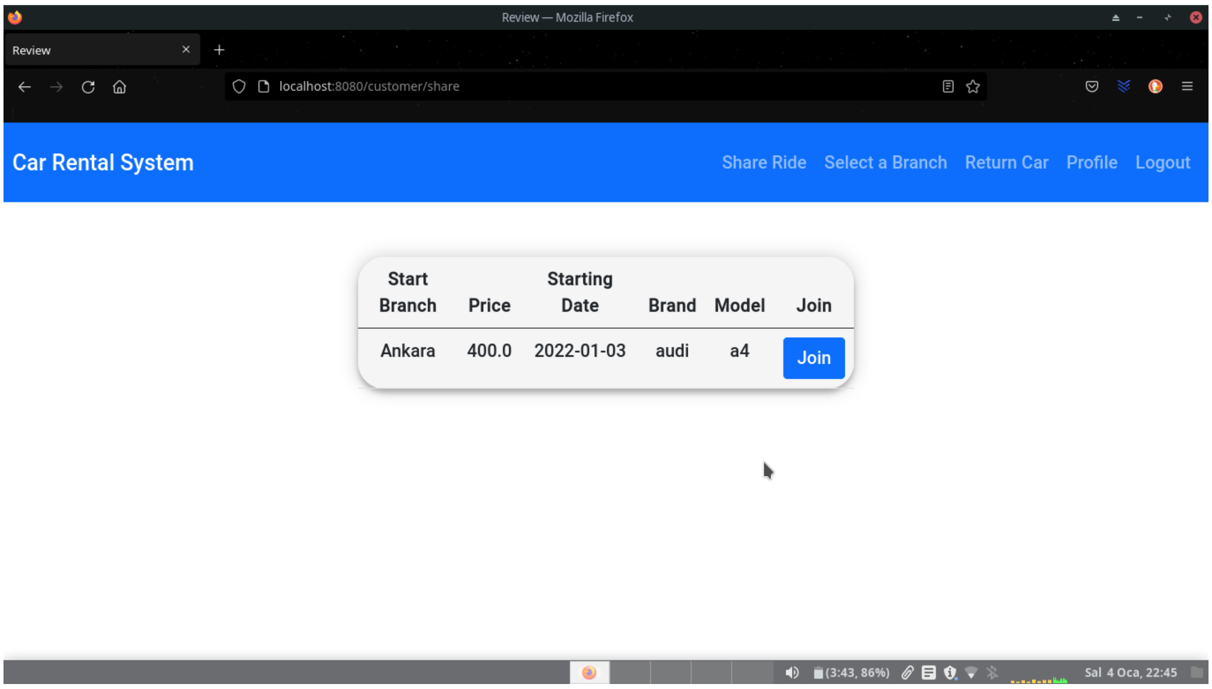
1

Your Review/Feedback

Please make a comment!

Share Ride Page

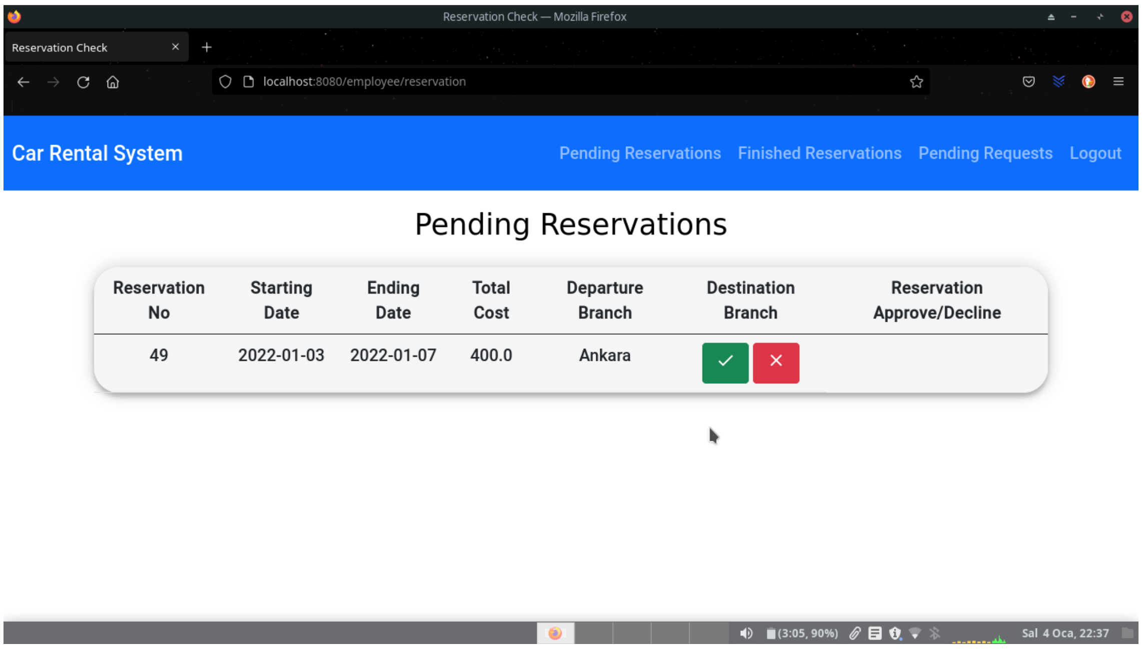
If a customer chooses the option “Share” while making a reservation, this reservation will be displayed at the Share Ride tab for other customers to join. Customers can choose the ride suited for them and click the “Join” button then they can see it in their profile page.



Start Branch	Price	Starting Date	Brand	Model	Join
Ankara	400.0	2022-01-03	audi	a4	Join

Employee Reservation Check Page

Using this page, employees can approve or reject waiting customer reservation requests. If an employee approves the reservation, he/she can return his/her car using the return car page any time, otherwise the customer's reservation is rejected and a new reservation has to be made.

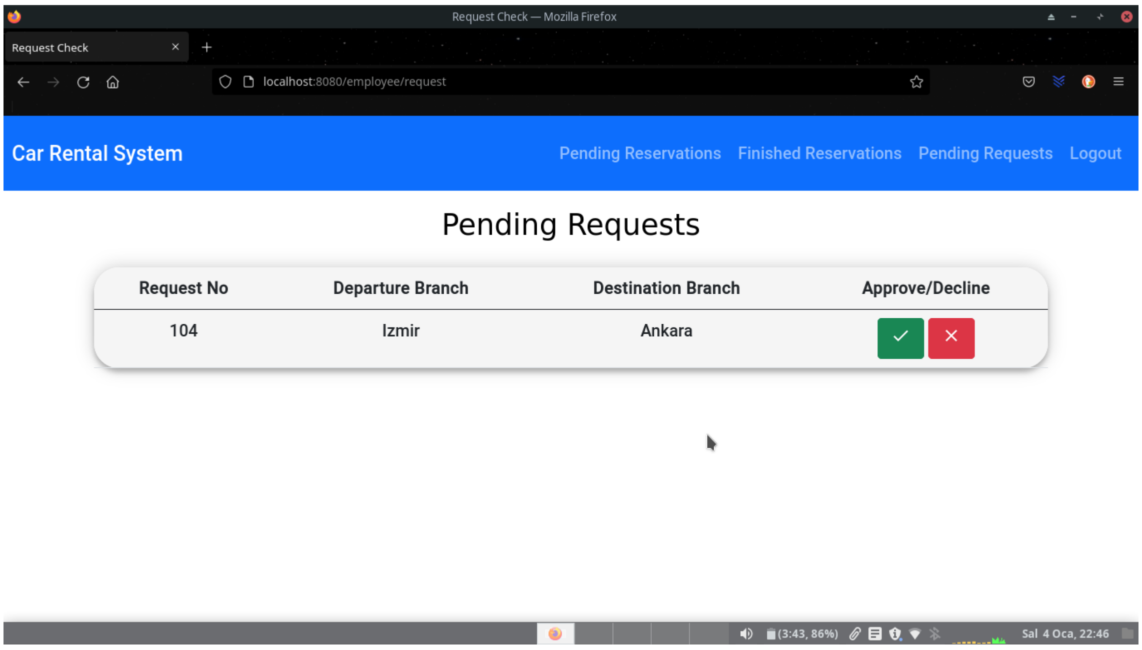


Pending Reservations

Reservation No	Starting Date	Ending Date	Total Cost	Departure Branch	Destination Branch	Reservation Approve/Decline
49	2022-01-03	2022-01-07	400.0	Ankara	✓ ✕	

Employee Request Check Page

Using this page, employees can approve or reject waiting customer car transfer requests. If an employee approves the request, the car will be removed from the branch and added to the customer profile, otherwise the car will stay on the branch until another request regarding that car will be approved.

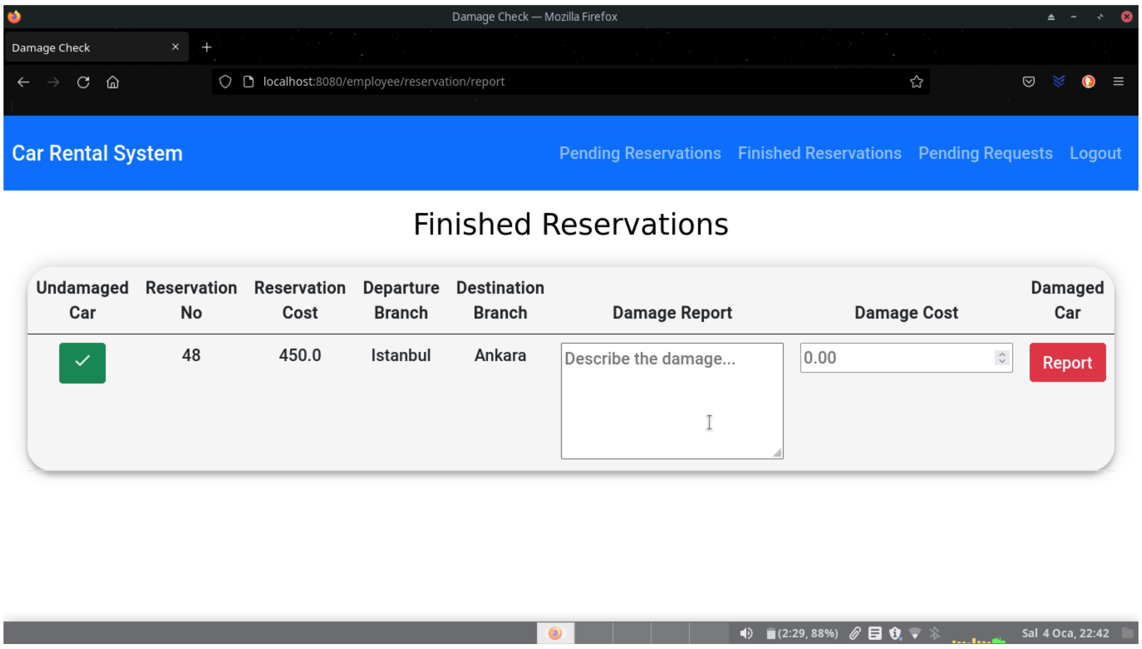


Pending Requests

Request No	Departure Branch	Destination Branch	Approve/Decline
104	Izmir	Ankara	✓ ✕

Employe Damage Check Page

Using this page, after the reservation status will become finished, that is, the customer returns his/her car via the return car page, and the employee checks for the damage on the car. If any damage is found, a damage report will be submitted together with the damage cost, otherwise an undamaged car button will be clicked and reservations status will become undamaged. From now on, the user can pay the expense via the return car page, he must pay the additional cost due to damage to the car, if any.



Finished Reservations

Undamaged Car	Reservation No	Reservation Cost	Departure Branch	Destination Branch	Damage Report	Damage Cost	Damaged Car
✓	48	450.0	Istanbul	Ankara	Describe the damage...	0.00	Report

WEBSITE

Website: https://edemirkirkan.github.io/Car-Rental-System/
Repository:https://github.com/coconatree/car_renatl_project