



**Car Rental System Project Design Report**

**CS 353 Database Systems**

**Project Design**

**Car Rental System**

**Group No 17**

**Emre Caniklioğlu - 21803577**

**Celal Berke Can - 21702886**

**Ege Demirkıran - 21802482**

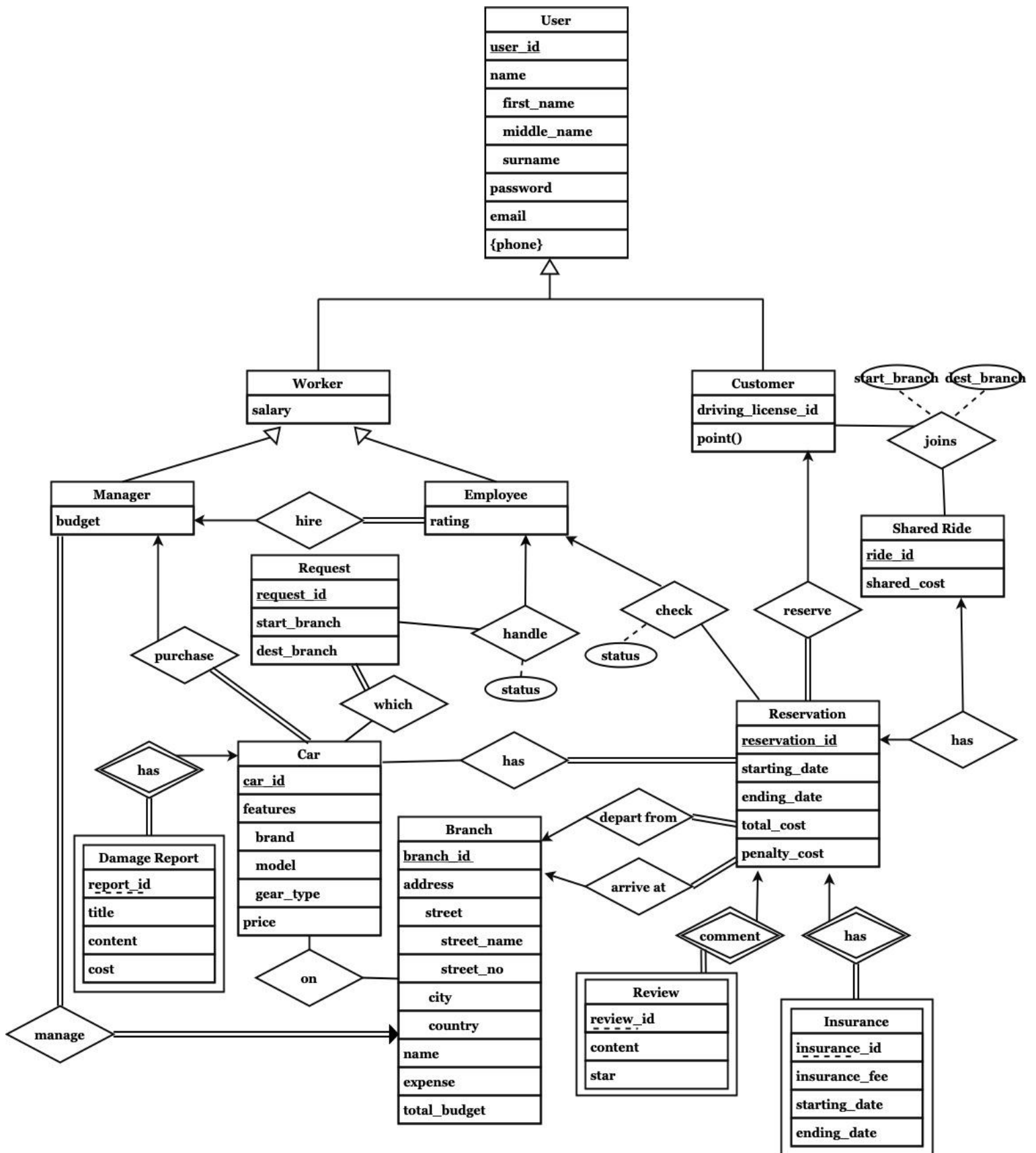
**Berk Saltuk Yılmaz - 21903419**

## **Table of Contents**

<b>1. REVISED ER MODEL</b>	<b>4</b>
<b>2. RELATION SCHEMAS</b>	<b>5</b>
2.1 Users	5
2.2 Worker	6
2.2.1 Manager	7
2.2.2 Employee	8
2.3 Customer	9
2.4 Reservation	10
2.5 Car	12
2.6 Branch	13
2.7 Review	14
2.8 Request	15
2.9 Insurance	16
2.10 User Phone	17
2.11 Damage Report	18
2.12 Shared Ride	19
2.13 Shared Ride Customer	20
2.14 Request Car	21
2.15 Car Reservation	22
2.16 Car Branch	23
<b>3. GUI DESIGN AND SQL QUERIES</b>	<b>24</b>
3.1 Login Page	24
3.2 Worker Login Page	25
3.4 Branch Selection	27
3.5 Car Selection Page	28
3.6 Car Transfer Request	29
3.7 Car Reservation Page	30
3.8 Car Damage Report Page	32
3.9 Return Car Page	33
3.10 Reservation Payment Page	34
3.11 Comment Page	35

3.12 Employee Hire Page	36
3.13 Buy Car Page	37
3.14 Validate Request / Reservation Page	38
3.15 Query Report Page	39
3.16 Ride Sharing Page	40
3.17 Reservation Status Update Page	41
4. IMPLEMENTATION PLAN	42
5. WEBSITE	43

## 1. REVISED ER MODEL



## 2. RELATION SCHEMAS

### 2.1 Users

#### Relational Model:

users(user\_id, first\_name, middle\_name, surname, password, email)

#### Candidate Keys and Primary Key:

Candidate Keys: user\_id, email

Primary Key: user\_id

#### Functional Dependencies:

user\_id -> first\_name, middle\_name, surname, password, email

email -> first\_name, middle\_name, surname, password, user\_id

#### Normal Form:

Boyce-Codd Normal Form (BCNF)

#### SQL Declaration:

```
DROP TABLE IF EXISTS users CASCADE;
CREATE TABLE users(
    user_id SERIAL NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    middle_name VARCHAR(20),
    surname VARCHAR(20) NOT NULL,
    password VARCHAR(40) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE
);
```

## 2.2 Worker

### Relational Model:

Worker(user\_id, salary)

FK: user\_id references User(user\_id)

### Candidate Keys and Primary Key:

Candidate Key: user\_id

Primary Key: user\_id

### Functional Dependencies:

user\_id -> salary

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS worker CASCADE;
CREATE TABLE worker(
  user_id INT PRIMARY KEY,
  salary FLOAT,
  FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

### 2.2.1 Manager

#### Relational Model:

Manager(user\_id, budget, branch\_id)

FK: user\_id references User(user\_id)

FK: manages references Branch(branch\_id)

#### Candidate Keys and Primary Key:

Candidate Keys: user\_id, branch\_id

Primary Key: user\_id

#### Functional Dependencies:

user\_id -> manages, budget

branch\_id -> user\_id, budget

#### Normal Form:

Boyce-Codd Normal Form (BCNF)

#### SQL Declaration:

```
DROP TABLE IF EXISTS manager CASCADE;
CREATE TABLE manager(
  user_id INT PRIMARY KEY,
  budget FLOAT,
  branch_id INT,
  FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE,
  FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE CASCADE
);
```

## 2.2.2 Employee

### Relational Model:

Employee(user\_id, rating, manager\_id)

FK: user\_id references User(user\_id)

FK: manager\_id references Manager(user\_id)

### Candidate Keys and Primary Key:

Candidate Keys: user\_id

Primary Key: user\_id

### Functional Dependencies:

user\_id -> rating, manager\_id

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS employee CASCADE;
CREATE TABLE employee(
  user_id INT PRIMARY KEY,
  rating FLOAT,
  manager_id INT,
  FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE,
  FOREIGN KEY(manager_id) REFERENCES manager(user_id) ON DELETE CASCADE
);
```



## 2.3 Customer

### Relational Model:

Customer(user\_id, driving\_license\_id, point)

### Candidate Keys and Primary Key:

Candidate Keys: user\_id, driving\_license\_id

Primary Key: user\_id

### Functional Dependencies:

user\_id -> driving\_license\_id, point

driving\_license\_id -> user\_id, point

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS customer CASCADE;
CREATE TABLE customer(
  user_id INT PRIMARY KEY,
  driving_license_id INT NOT NULL UNIQUE,
  point FLOAT,
  FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

## 2.4 Reservation

### Relational Model:

Reservation(reservation\_id, starting\_date, ending\_date, total\_cost, penalty\_cost, status, customer\_id, employee\_id, depart\_from, arrive\_at)

FK: customer\_id references Customer(user\_id)

FK: employee\_id references Employee(user\_id)

FK: depart\_from references Branch(branch\_id)

FK: arrive\_at references Branch(branch\_id)

### Candidate Keys and Primary Key:

Candidate Key: reservation\_id, employee\_id, customer\_id

Primary Key: reservation\_id

### Functional Dependencies:

reservation\_id -> starting\_date, ending\_date, total\_cost, penalty\_cost, status, customer\_id, employee\_id, depart\_from, arrive\_at

reserved\_by -> starting\_date, ending\_date, total\_cost, penalty\_cost, status, reservation\_id, employee\_id, depart\_from, arrive\_at

checked\_by -> starting\_date, ending\_date, total\_cost, penalty\_cost, status, customer\_id, reservation\_id, depart\_from, arrive\_at

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
CREATE TABLE reservation(  
  reservation_id SERIAL NOT NULL PRIMARY KEY,  
  starting_date DATE,  
  ending_date DATE,  
  total_cost FLOAT,  
  penalty_cost FLOAT DEFAULT 0,  
  status INT,  
  customer_id INT NOT NULL,
```

```
employee_id INT NOT NULL,  
depart_from INT,  
arrive_at INT,  
FOREIGN KEY (customer_id) REFERENCES customer(user_id),  
FOREIGN KEY (employee_id) REFERENCES employee(user_id),  
FOREIGN KEY (depart_from) REFERENCES branch(branch_id),  
FOREIGN KEY (arrive_at) REFERENCES branch(branch_id)  
);
```

## 2.5 Car

### Relational Model:

Car(car\_id, brand, model, gear\_type, price, manager\_id)

FK: manager\_id references Manager(user\_id)

### Candidate Keys and Primary Key:

Candidate Keys: car\_id, manager\_id

Primary Key: car\_id

### Functional Dependencies:

car\_id -> brand, model, gear\_type, price, manager\_id

manager\_id -> car\_id, brand, model, gear\_type, price,

on\_branch -> manager\_id, car\_id, brand, model, gear\_type, price

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS car CASCADE;
CREATE TABLE car(
  car_id SERIAL NOT NULL PRIMARY KEY ,
  brand VARCHAR(30),
  model VARCHAR(30),
  gear_type VARCHAR(10),
  price FLOAT,
  manager_id INT NOT NULL,
  FOREIGN KEY (manager_id) REFERENCES manager(user_id) ON DELETE CASCADE
);
```

## 2.6 Branch

### Relational Model:

Branch(branch\_id, branch\_name, street\_name, street\_no, city, country, total\_budget, expense)

### Candidate Keys and Primary Key:

Candidate Key: branch\_id

Primary Key: branch\_id

### Functional Dependencies:

branch\_id -> branch\_name, street\_name, street\_no, city, country, total\_budget, expense

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS branch CASCADE;
CREATE TABLE branch(
  branch_id SERIAL NOT NULL PRIMARY KEY,
  branch_name VARCHAR(30),
  street_name VARCHAR(30),
  street_no VARCHAR(10),
  city VARCHAR(30),
  country VARCHAR(30) DEFAULT 'Turkey',
  total_budget INT,
  expense INT
);
```

## 2.7 Review

### Relational Model:

Review(reservation\_id, review\_id, content, star)

FK: reservation\_id references Reservation(reservation\_id)

### Candidate Keys and Primary Key:

Candidate Key: {reservation\_id, review\_id}

Primary Key: {reservation\_id, review\_id}

### Functional Dependencies:

reservation\_id, review\_id -> content, star

### Normal Form:

Third Normal Form (3NF)

### SQL Declaration:

```
DROP TABLE IF EXISTS review CASCADE;
CREATE TABLE review(
  review_id SERIAL NOT NULL ,
  reservation_id INT NOT NULL ,
  content VARCHAR(255),
  star INT,
  PRIMARY KEY (review_id, reservation_id),
  FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id) ON DELETE
  CASCADE
);
```

## 2.8 Request

### Relational Model:

Request(request\_id, start\_branch, dest\_branch, status, employee\_id)

FK: start\_branch references Branch(branch\_id)

FK: dest\_branch references Branch(branch\_id)

FK: employee\_id references Employee(employee\_id))

### Candidate Keys and Primary Key:

Candidate Key: request\_id, employee\_id

Primary Key: request\_id

### Functional Dependencies:

request\_id -> start\_branch, dest\_branch, status, employee\_id

employee\_id -> request\_id, start\_branch, dest\_branch, status

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS request CASCADE;
CREATE TABLE request(
    request_id SERIAL NOT NULL PRIMARY KEY,
    start_branch INT,
    dest_branch INT,
    employee_id INT,
    status INT ,
    FOREIGN KEY (start_branch) REFERENCES branch(branch_id) ON DELETE CASCADE,
    FOREIGN KEY (dest_branch) REFERENCES branch(branch_id) ON DELETE CASCADE,
    FOREIGN KEY (employee_id) REFERENCES employee(user_id) ON DELETE CASCADE
);
```

## 2.9 Insurance

### Relational Model:

insurance(reservation\_id, insurance\_id, starting\_date, ending\_date, insurance\_fee)

FK: reservation\_id references Reservation(reservation\_id)

### Candidate Keys and Primary Key:

Candidate Key: {reservation\_id, insurance\_id}

Primary Key: {reservation\_id, insurance\_id}

### Functional Dependencies:

reservation\_id, insurance\_id -> starting\_date, ending\_date, insurance\_fee

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS insurance CASCADE;
CREATE TABLE insurance(
  insurance_id SERIAL NOT NULL,
  reservation_id INT NOT NULL,
  starting_date DATE,
  ending_date DATE,
  insurance_fee INT,
  PRIMARY KEY (insurance_id, reservation_id),
  FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id) ON DELETE
  CASCADE
);
```



## 2.10 User Phone

### Relational Model:

user\_phone(user\_id, phone)

### Candidate Keys and Primary Key:

Candidate Key: {user\_id, phone}

Primary Key: {user\_id, phone}

### Functional Dependencies:

No functional dependencies

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS user_phone CASCADE;  
CREATE TABLE user_phone(  
  user_id INT NOT NULL PRIMARY KEY,  
  phone INT NOT NULL UNIQUE,  
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE  
);
```

## 2.11 Damage Report

### Relational Model:

damage\_report(report\_id, car\_id, title, content, cost)

### Candidate Keys and Primary Key:

Candidate Key: {report\_id, car\_id}

Primary Key: {report\_id, car\_id}

### Functional Dependencies:

report\_id, car\_id -> content, cost

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS damage_report CASCADE;
CREATE TABLE damage_report(
    report_id SERIAL NOT NULL,
    car_id INT NOT NULL,
    title VARCHAR(50),
    content VARCHAR(200),
    cost INT,
    PRIMARY KEY (report_id, car_id),
    FOREIGN KEY (car_id) REFERENCES car(car_id) ON DELETE CASCADE
);
```

## 2.12 Shared Ride

### Relational Model:

shared\_ride(ride\_id, shared\_cost, reservation\_id)

FK: reservation\_id references Reservation(reservation\_id)

### Candidate Keys and Primary Key:

Candidate Key: ride\_id, reservation\_id

Primary Key: ride\_id

### Functional Dependencies:

ride\_id -> shared\_cost, reservation\_id

reservation\_id -> ride\_id, shared\_cost, reservation\_id

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS shared_ride CASCADE;
CREATE TABLE shared_ride(
    ride_id SERIAL NOT NULL PRIMARY KEY,
    reservation_id INT NOT NULL,
    shared_cost INT,
    FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id) ON
DELETE CASCADE
);
```

## 2.13 Shared Ride Customer

### Relational Model:

customer\_shared\_ride(ride\_id, passenger\_id, start\_branch, dest\_branch)

FK: passenger\_id references customer(user\_id)

FK: ride\_id references shared\_ride(ride\_id)

FK: start\_branch references Branch(branch\_id)

FK: dest\_branch references Branch(branch\_id)

### Candidate Keys and Primary Key:

Candidate Key: {ride\_id, passenger\_id}

Primary Key: {ride\_id, passenger\_id}

### Functional Dependencies:

ride\_id, passenger\_id -> start\_branch, dest\_branch

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS customer_shared_ride CASCADE;
CREATE TABLE customer_shared_ride(
  ride_id INT NOT NULL,
  passenger_id INT NOT NULL,
  start_branch INT,
  dest_branch INT,
  PRIMARY KEY (ride_id, passenger_id),
  FOREIGN KEY(ride_id) REFERENCES shared_ride(ride_id) ON DELETE CASCADE,
  FOREIGN KEY(passenger_id) REFERENCES customer(user_id) ON DELETE CASCADE,
  FOREIGN KEY(start_branch) REFERENCES branch(branch_id) ON DELETE CASCADE,
  FOREIGN KEY(dest_branch) REFERENCES branch(branch_id) ON DELETE CASCADE
);
```

## 2.14 Request Car

### Relational Model:

request\_car(car\_id, request\_id)

FK: car\_id references Car(car\_id)

FK: request\_id references Request(request\_id)

### Candidate Keys and Primary Key:

Candidate Key: {car\_id, request\_id}

Primary Key: {car\_id, request\_id}

### Functional Dependencies:

No functional dependencies

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS request_car CASCADE;
CREATE TABLE request_car(
  car_id INT NOT NULL,
  request_id INT NOT NULL,
  PRIMARY KEY (car_id, request_id),
  FOREIGN KEY(car_id) REFERENCES car(car_id) ON DELETE CASCADE,
  FOREIGN KEY(request_id) REFERENCES request(request_id) ON DELETE CASCADE
);
```

## 2.15 Car Reservation

### Relational Model:

customer\_shared\_ride(car\_id, reservation\_id)

FK: car\_id references Car(car\_id)

FK: reservation\_id references reservation(reservation\_id)

### Candidate Keys and Primary Key:

Candidate Key: {car\_id, reservation\_id}

Primary Key: {car\_id, reservation\_id}

### Functional Dependencies:

No functional dependencies

### Normal Form:

Boyce-Codd Normal Form (BCNF)

### SQL Declaration:

```
DROP TABLE IF EXISTS car_reservation CASCADE;
CREATE TABLE car_reservation(
  car_id INT,
  reservation_id INT,
  PRIMARY KEY (car_id, reservation_id),
  FOREIGN KEY(car_id) REFERENCES car(car_id) ON DELETE CASCADE,
  FOREIGN KEY(reservation_id) REFERENCES reservation(reservation_id) ON DELETE
  CASCADE
);
```

## 2.16 Car Branch

### Relational Model:

car\_branch(car\_id, branch\_id)

FK: car\_id references Car(car\_id)

FK: branch\_id references branch(branch\_id)

### Candidate Keys and Primary Key:

Candidate Key: {car\_id, branch\_id}

Primary Key: {car\_id, branch\_id}

### Functional Dependencies:

No functional dependencies

### Normal Form:

Boyce-Codd Normal Form (BCNF)

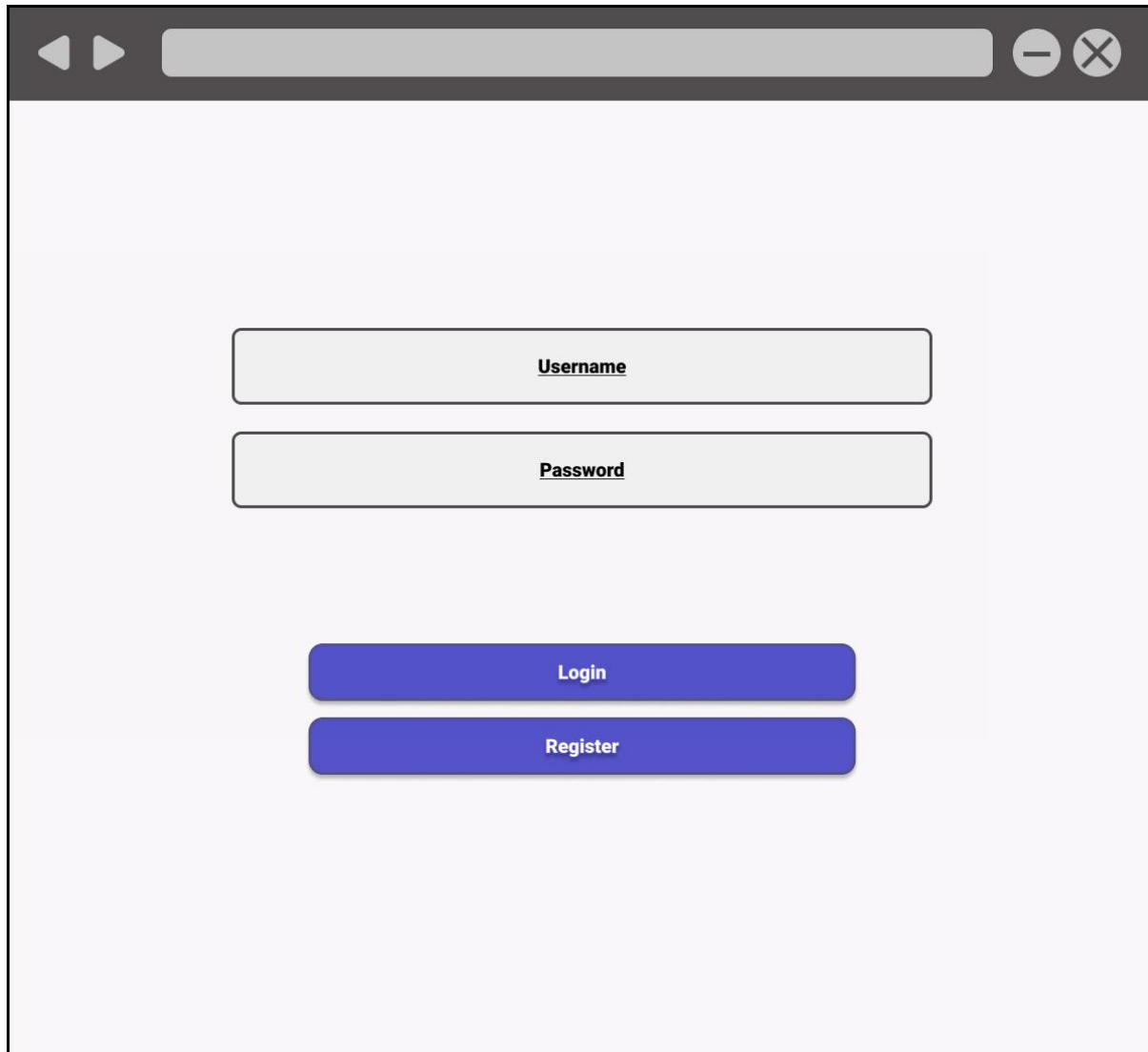
### SQL Declaration:

```
DROP TABLE IF EXISTS car_branch CASCADE;
CREATE TABLE car_branch (
    car_id INT NOT NULL,
    branch_id INT NOT NULL,
    PRIMARY KEY (car_id, branch_id),
    FOREIGN KEY(car_id) REFERENCES car(car_id) ON DELETE CASCADE,
    FOREIGN KEY(branch_id) REFERENCES branch(branch_id) ON DELETE CASCADE
);
```

### 3. GUI DESIGN AND SQL QUERIES

#### 3.1 Login Page

This page will authenticate customers and grant them access to authenticated services such as car reservation or request.



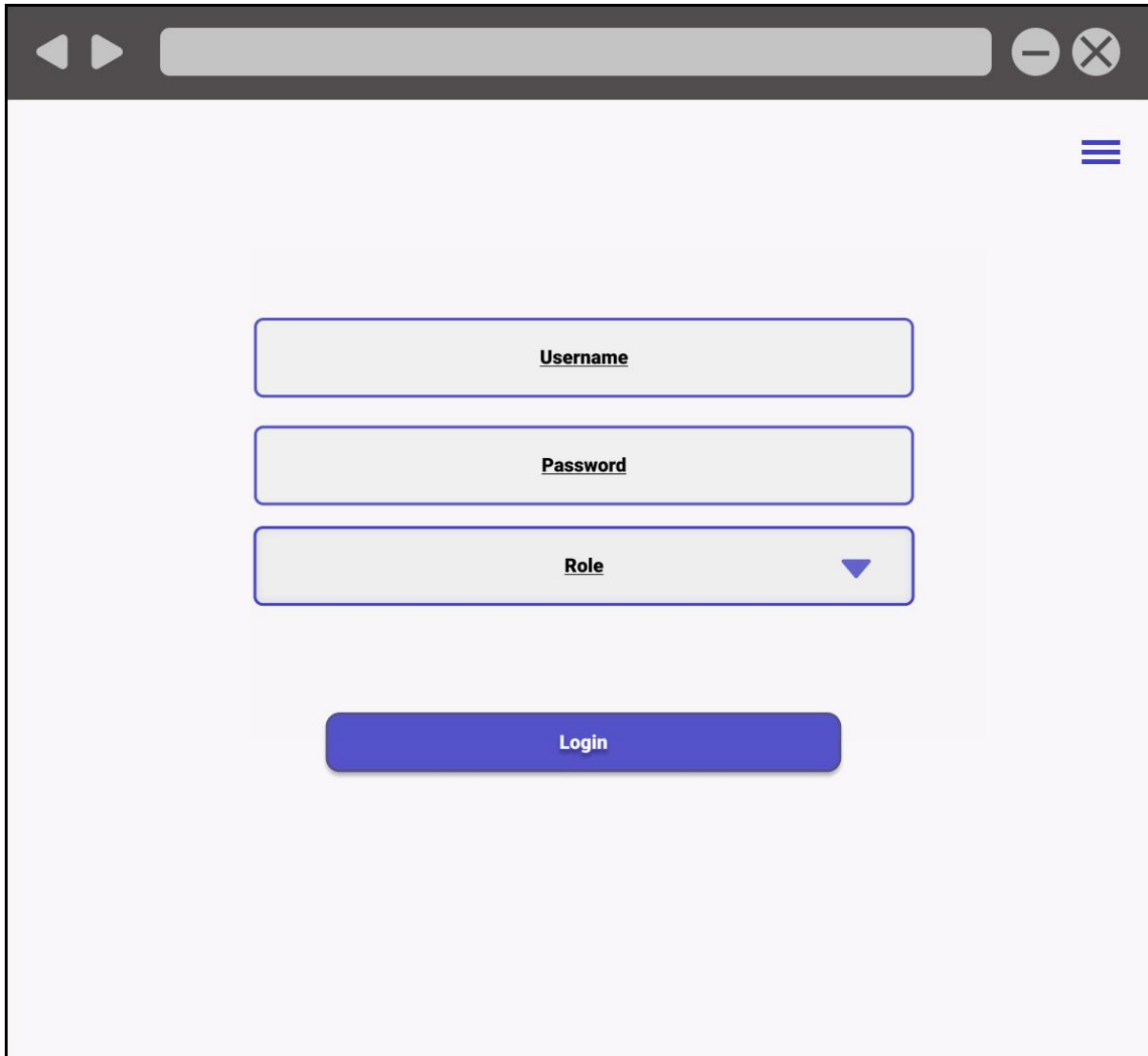
The image shows a web browser window with a dark gray header containing navigation icons (back, forward, and address bar). The main content area is light gray and contains a login form. The form consists of two light gray input fields with rounded corners. The first field is labeled "Username" and the second is labeled "Password". Below these fields are two blue buttons with rounded corners. The first button is labeled "Login" and the second is labeled "Register".

```
SELECT user_id FROM users WHERE email = @email AND password = @password;
```



### 3.2 Worker Login Page

This page will authenticate employees and managers and it will grant them access to the system. Role value will be later used for determining whether a manager or an employee is logging in.

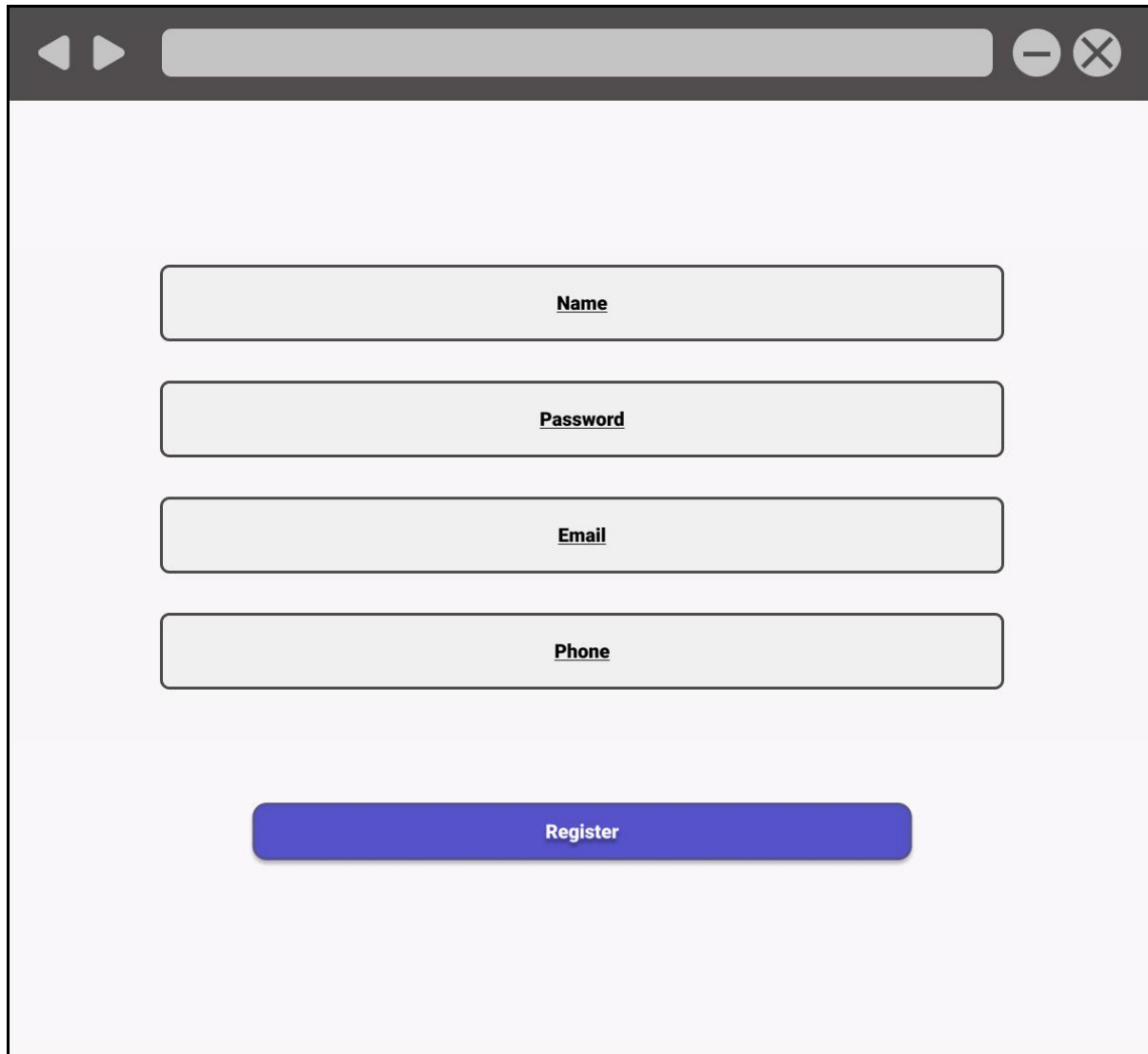


```
SELECT user_id, password FROM users
WHERE email = @email
AND password = @password;
```

```
SELECT * FROM employee WHERE user_id = @user_id AND password = @password;;
SELECT * FROM manager WHERE user_id = @user_id AND password = @password;;
```

### 3.3 Register Page

This page will register a customer to the records.



The image shows a web browser window with a registration form. The browser's address bar is empty. The form consists of four text input fields stacked vertically, each with a placeholder label: 'Name', 'Password', 'Email', and 'Phone'. Below these fields is a single blue button with the text 'Register'.

```
INSERT INTO users (first_name, middle_name, surname, password, email) VALUES  
(@first_name, @middle_name, @middle_name, @password, @email);
```

```
SELECT user_id FROM users WHERE email = @email;
```

```
INSERT INTO customer (user_id, driving_license_id, point) VALUES (@user_id,  
@license_id, @points);
```

### 3.4 Branch Selection

The image shows a web application interface. At the top, there is a dark header bar containing navigation arrows, a search bar, and window controls (minimize, maximize, close). Below the header, the main content area is light gray. In the top right corner of the main area, there is a hamburger menu icon. Centered in the main area is a white rectangular box with a thin blue border. Inside this box, there is a dropdown menu labeled 'Starting Branch' with a blue downward arrow on the right. Below the dropdown menu is a solid blue button with the text 'List Cars' in white.

```
SELECT * FROM branch WHERE branch_id = @branch_id;
```

### 3.5 Car Selection Page


Starting Date ▼

Ending Date ▼

Model ▼

Km


Search



Available Until	15:11:2021
Model	Audi
Gear-Type	Automatic
Km	50.000

Make Reservation

Make Request



Available Until	15:11:2021
Model	Audi
Gear-Type	Automatic
Km	50.000

Make Reservation

Make Request


```
SELECT * FROM car
branch_id = @branch_id AND
WHERE model = @model AND km <= @km;
```

### 3.6 Car Transfer Request

Starting Date ▼


Ending Date ▼

Model ▼



Make Reservation

Make Request



Make Reservation

Make Request

Delivery Branch ▼

Make Request

```
SELECT branch_id FROM branch WHERE branch_id = @branch_id;  
INSERT INTO request (start_branch, dest_branch, status) VALUES  
(@starting_branch_id, @ending_branch_id, 0);
```

```
INSERT INTO request_car (car_id, request_id) VALUES (@car_id, @request_id);
```

### 3.7 Car Reservation Page

Using this page customers will be able to reserve cars. If the insurance button is checked then it will create a record of insurance and if the share button is checked it will put this reservation to the ride sharing board. Where other customers can join in on the reservation.

The screenshot shows a web application for car reservations. On the left, there are two car cards, each with a blue Audi sedan image and 'Make Reservation' and 'Make Request' buttons. Above these cards are three dropdown menus for 'Starting Date', 'Ending Date', and 'Model'. On the right, there is a form with five input fields: 'Select Starting Date', 'Select Ending Date', 'Delivery Branch', 'Insurance' (with an unchecked checkbox), and 'Share' (with an unchecked checkbox). A 'Make Reservation' button is at the bottom of the right section. The interface is presented in a window-like format with navigation arrows and a close button.

```
SELECT branch_id FROM branch WHERE branch_name = @branch_name;

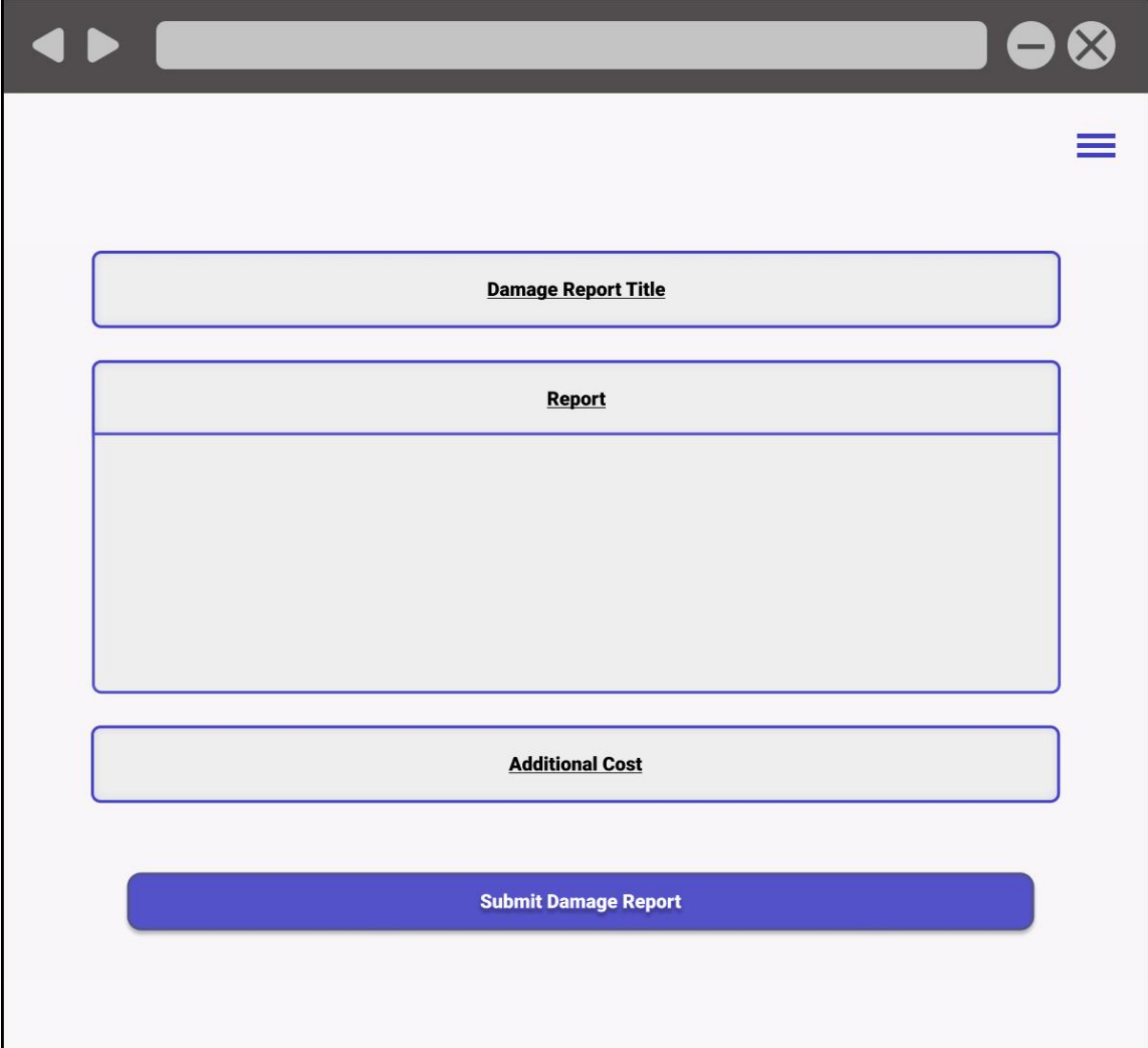
INSERT INTO reservation (starting_date, ending_date, total_cost,
penalty_cost, status, has_insurance, customer_id, depart_from, arrive_at)
VALUES (@starting_date, @ending_date, @total_cost , 0, 0, @has_insurance,
@user_id, @starting_branch, @ending_branch);
```

```
INSERT INTO car_reservation (car_id, reservation_id) VALUES (@car_id,  
@reservation_id);
```

```
INSERT INTO shared_ride (reservation_id, shared_cost) VALUES  
(@reservation_id, @cost);
```

### 3.8 Car Damage Report Page

Employee will enter his report about the car's condition using the following page. Using this report additional costs will be calculated.

A screenshot of a web application window titled "Car Damage Report Page". The window has a dark header bar with navigation arrows, a search bar, and window control buttons (minimize, maximize, close). The main content area is light gray and contains a form with four input fields: "Damage Report Title", "Report", "Additional Cost", and a "Submit Damage Report" button. The "Report" field is a large text area. A hamburger menu icon is visible in the top right corner of the page content.

Damage Report Title

Report

Additional Cost

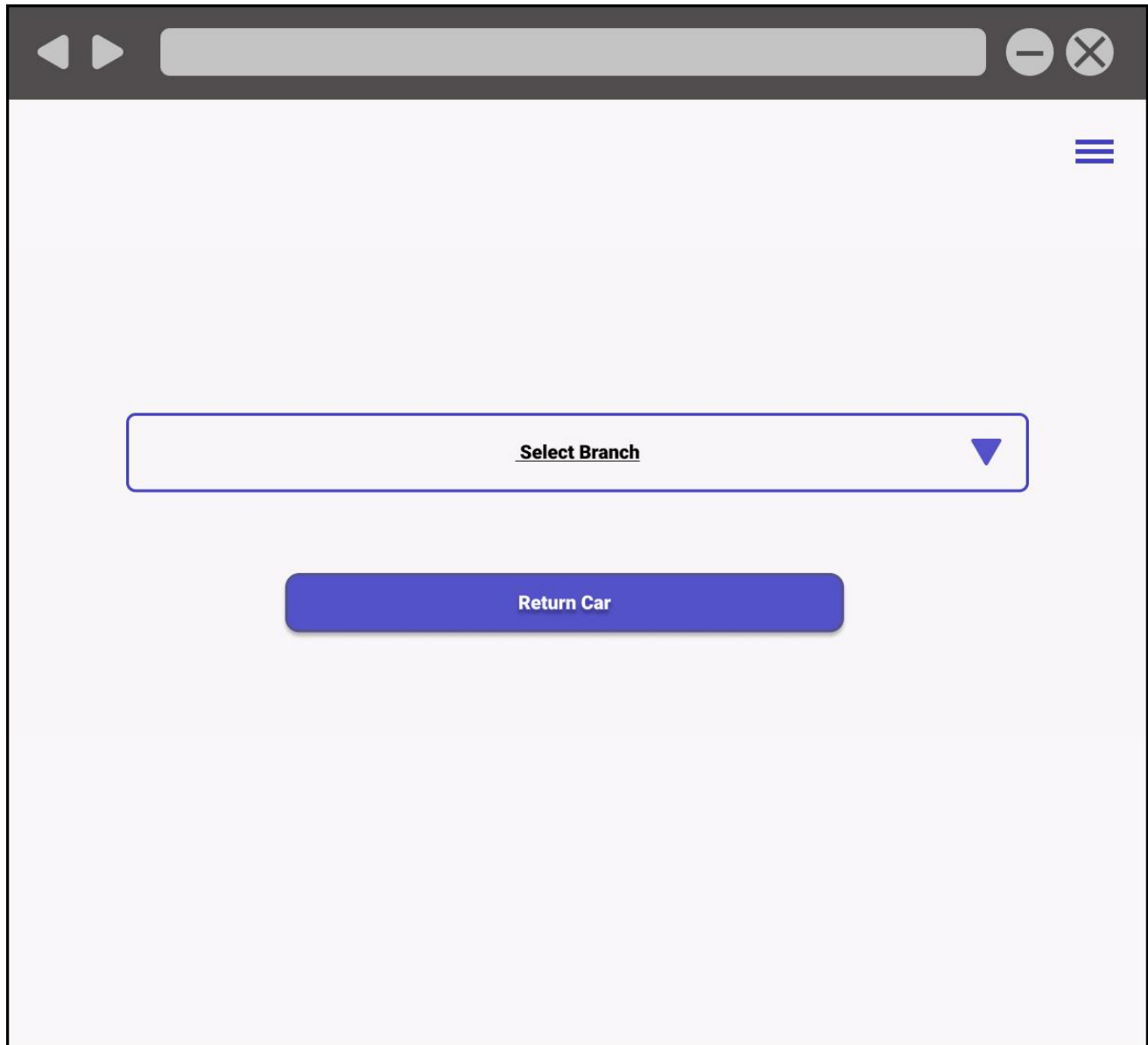
Submit Damage Report

```
INSERT INTO damage_report (car_id, title, content, cost) VALUES (@car_id, @title, @content, @cost);
```



### 3.9 Return Car Page

This page will be accessible by employees and managers and it will enable them to return cars into the selected branch.



The screenshot shows a web browser window with a dark header bar containing navigation arrows, a search bar, and window control buttons. The main content area has a light gray background. In the top right corner, there is a hamburger menu icon. Centered on the page is a white rectangular input field with a blue border, containing the text "Select Branch" and a blue downward-pointing triangle on the right. Below this field is a solid blue button with the text "Return Car" in white.

```
INSERT INTO car_branch (car_id, branch_id) VALUES (@car_id, @branch_id);
```

### 3.10 Reservation Payment Page

This page will allow customers to pay for the service they received after they deliver the vehicle to the delivery branch.

The screenshot shows a web browser window with a reservation payment page. The page has a light purple background and a dark grey header bar with navigation icons (back, forward, search, and close). A hamburger menu icon is in the top right corner. The main content area contains a reservation summary box and a cost breakdown section.

**Reservation**

From Istanbul A	From 15-12-2021
To Ankara A	To 15-12-2021

**Cost** 500tl   **Late Delivey Cost** 0TL   **Damage Cost** 100 TL   **Pay**

```
INSERT INTO damage_report (car_id, title, content, cost) VALUES (@car_id, @title, @content, @cost);
```

```
UPDATE reservation SET penalty_cost = @late_delivery_cost, total_cost = @late_delivery_cost + @damage_cost WHERE reservation_id = @reservation_id;
```

### 3.11 Comment Page

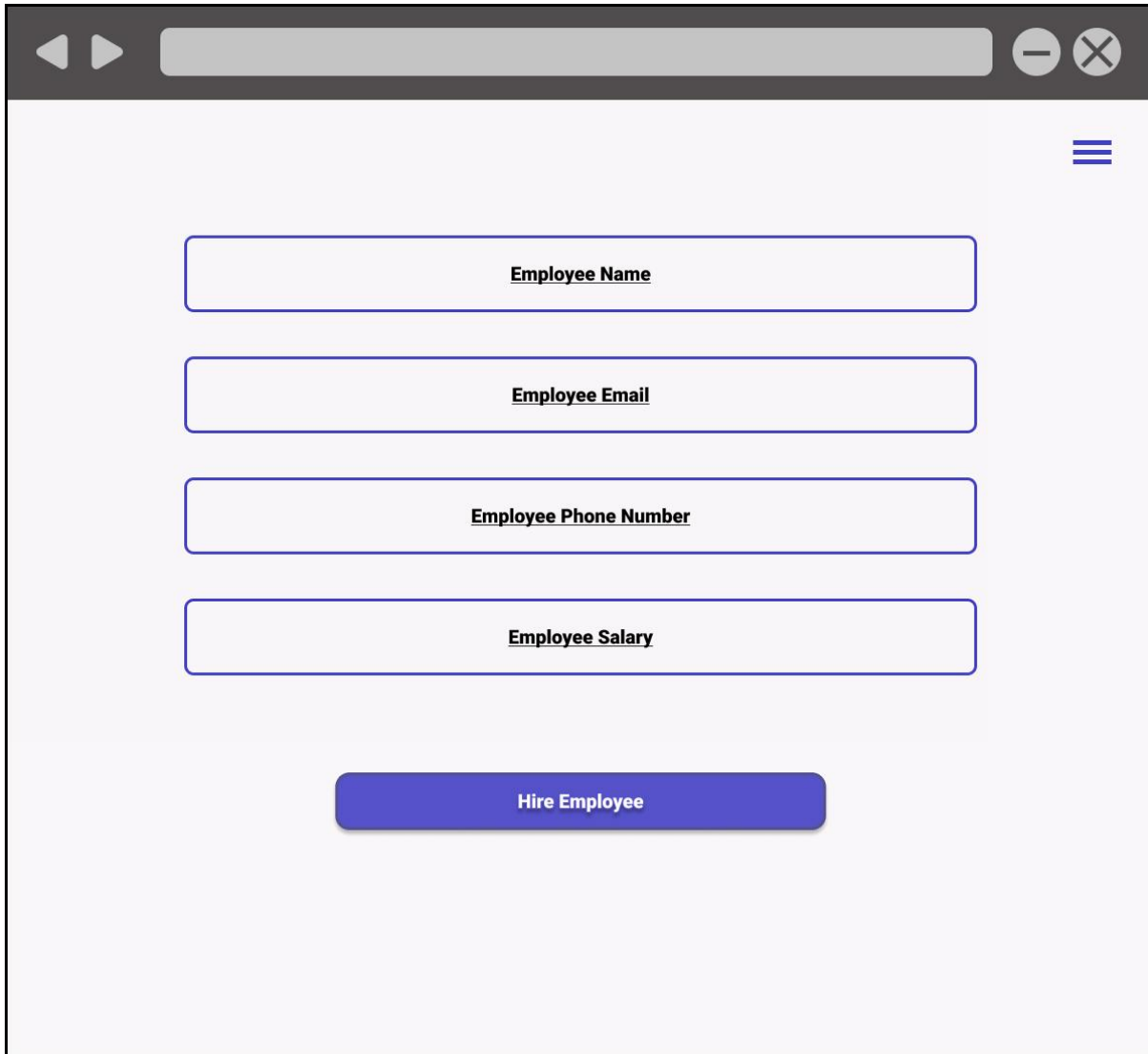
This page will be available to the customers after their payment is processed. They will be able to post reviews about their experiences.

The image shows a web browser window with a dark header bar containing navigation arrows and window control buttons (minimize, maximize, close). The main content area is light gray and contains a review form. The form consists of a 'Title' input field, a 'Content' text area, a five-star rating system, and a 'Post Review' button. A close button (X) is located in the top right corner of the form area.

```
INSERT INTO review (reservation_id, content, star) VALUES (@reservation_id,
@content, @star);
```

### 3.12 Employee Hire Page

This page will be available to the managers. It will provide managers with a form for hiring new employees to their branches.



```
/* Creates a new user record */
```

```
INSERT INTO users (first_name, middle_name, surname, password, email) VALUES  
(@first_name, @middle_name, @surname, @password, @email);
```

```
/* Retrieves the new records id */
```

```
SELECT users.user_id FROM users WHERE users.email = @email;
```

```
/* Creates a new worker record */
```

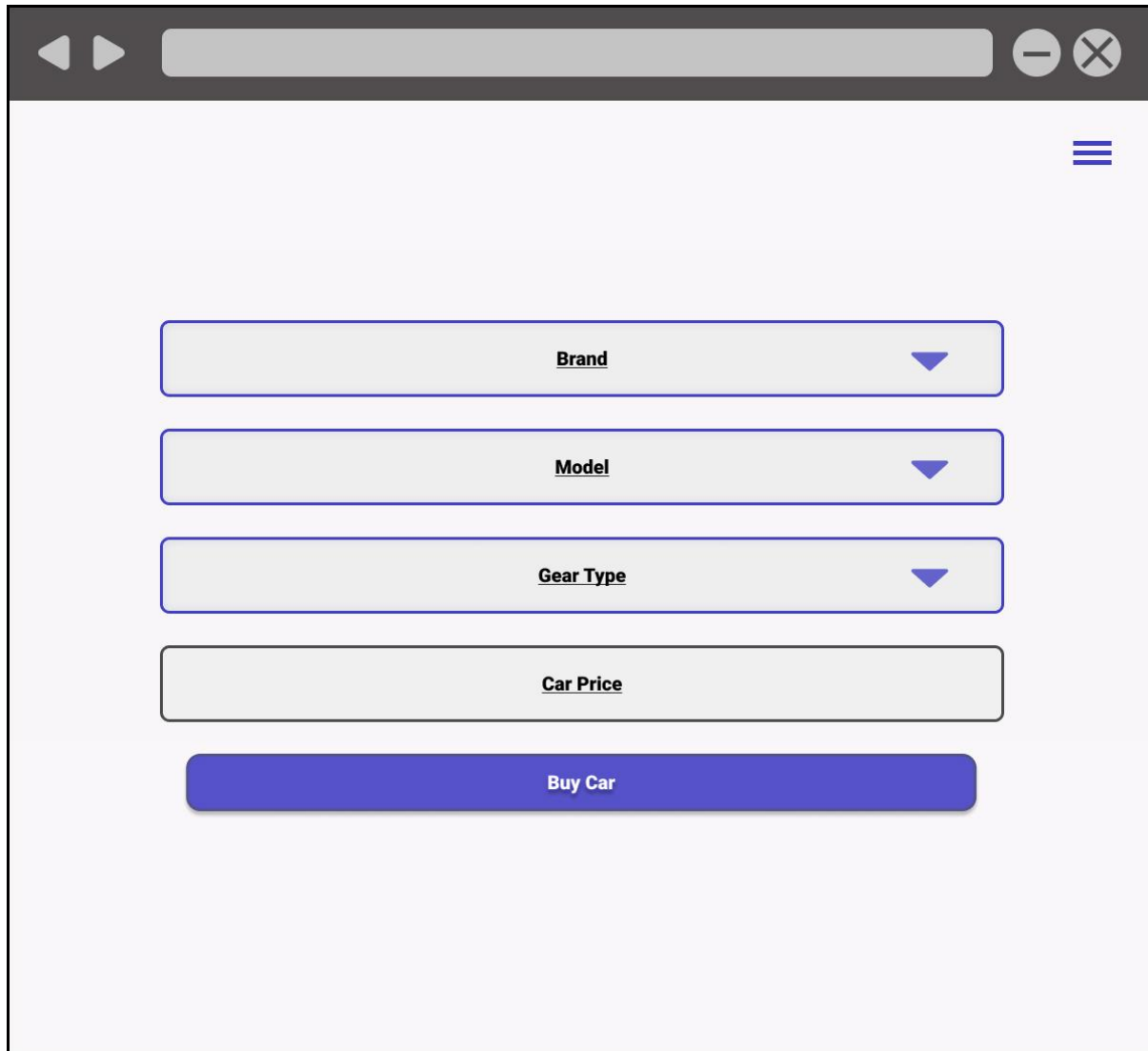
```
INSERT INTO worker (@user_id, @salary) VALUES (@user_id, @salary);
```

```
/* Creates a new employee record */
```

```
INSERT INTO employee (user_id, rating, hired_by) VALUES (@user_id, 0,  
@manager_id);
```

### 3.13 Buy Car Page

This page will allow managers to buy new cars for their branches.



The screenshot shows a web application interface for buying cars. It features a browser window with a dark header bar containing navigation arrows, a search bar, and window control buttons. The main content area has a light purple background and a hamburger menu icon in the top right. The form consists of five stacked input fields: three dropdown menus labeled 'Brand', 'Model', and 'Gear Type', followed by a text input field labeled 'Car Price', and a blue 'Buy Car' button at the bottom.

```
INSERT INTO car (brand, model, gear_type, price, manager_id)
VALUES (@brand, @model, @gear_type, @price, @manager_id);
INSERT INTO car_branch (@car_id, @branch_id);
```

### 3.14 Validate Request / Reservation Page

This page will give employees and managers an interface for approving reservations and requests made by the customers.

The screenshot shows a web application interface for managing reservations and requests. At the top, there is a navigation bar with a hamburger menu icon on the right. Below the navigation bar, there is a table with the following columns: **From**, **To**, **Starting Date**, **Ending Date**, and **Request / Resevation**. The table contains five rows of data. The first row is highlighted and expanded to show details for a reservation from Ankara A to Izmir B, starting on 20:11:2021 and ending on 25:11:2021. The details are organized into two sections: **Customer** and **Car**. The Customer section shows Name: Emre and Experience: 3 years. The Car section shows Brand: Audi, Model: A4, and Gear Type: Manuel. Each row in the table has an **Approve** button and a dropdown arrow.

From	To	Starting Date	Ending Date	Request / Resevation	Approve
Ankara A	Izmir B	20:11:2021	25:11:2021	Reservation	Approve ▼
Istanbul E	Izmir B	16:11:2021	27:11:2021	Reservation	Approve ▼
Konya A	Fethiye C	16:11:2021	27:11:2021	Request	Approve ▼
Ankara D	Edirne A	12:11:2021	26:11:2021	Reservation	Approve ▼
Istanbul E	Izmir B	19:11:2021	26:11:2021	Request	Approve ▼

**Customer**  
Name : Emre  
Experience : 3 years

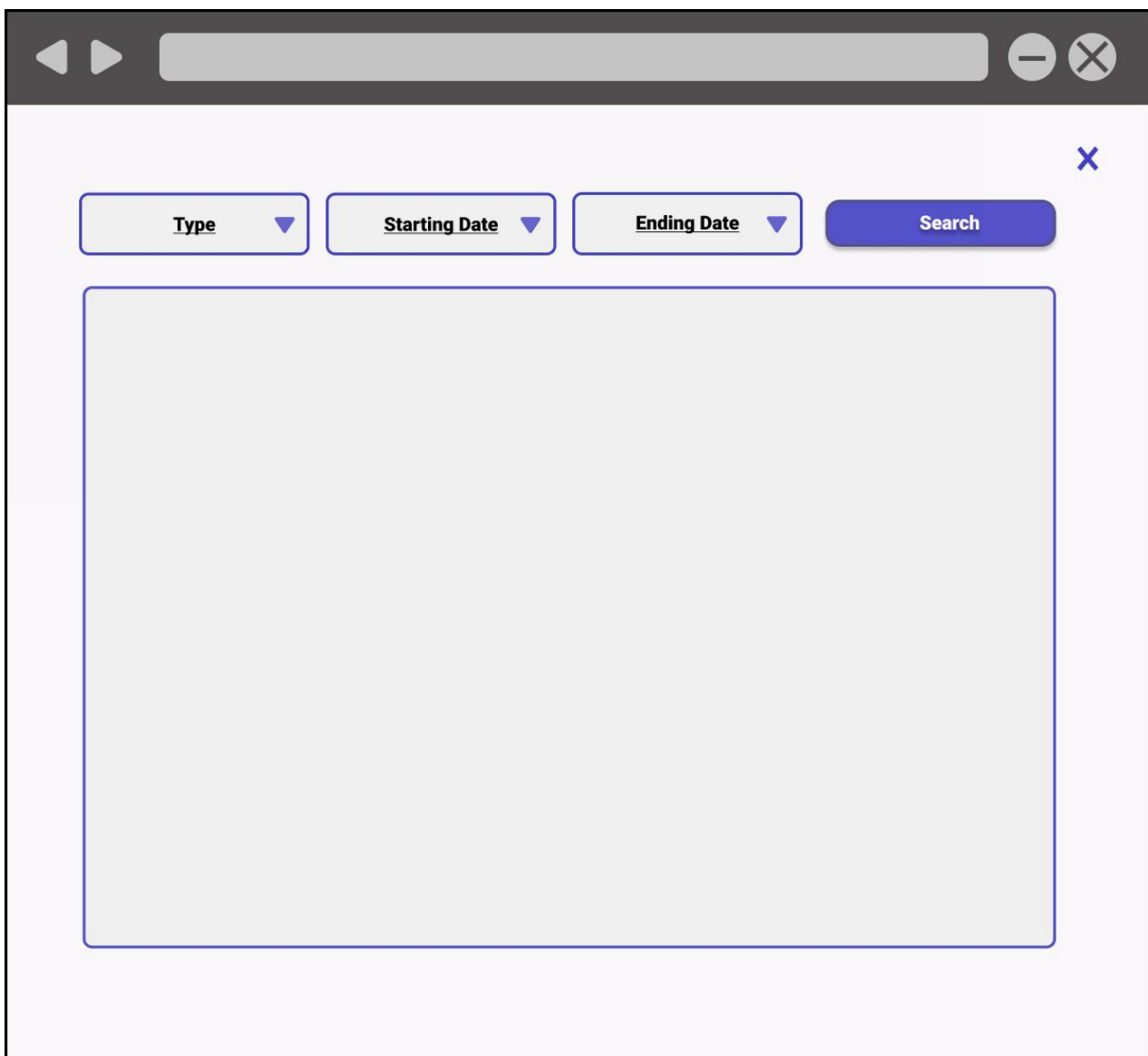
**Car**  
Brand : Audi  
Model : A4  
Gear Type : Manuel

```
-- take the necessary ones..
SELECT * FROM reservation R, car_reservation CR
  JOIN insurance I  ON R.reservation_id = I.reservation_id
  JOIN customer  C  ON C.user_id = R.customer_id
  JOIN branch    B1 ON b1.branch_id = R.depart_from
  JOIN branch    B2 ON b2.branch_id = R.arrive_at
  JOIN car        ON car.car_id = CR.car_id
WHERE CR.reservation_id = R.reservation_id;
```

```
-- take the necessary ones.  
SELECT * FROM request req  
JOIN branch B1      on b1.branch_id = req.dest_branch  
JOIN branch B2      on b2.branch_id = req.start_branch  
JOIN request_car rcar on rcar.request_id = req.request_id  
JOIN car            on car.car_id = rcar.car_id
```

### 3.15 Query Report Page

This page will give workers an interface for retrieving information about past reservations and requests.



The screenshot shows a web browser window with a dark header bar containing navigation arrows and a search bar. Below the header, the page has a light blue background. At the top right, there is a close button (X). Below this, there are three dropdown menus labeled "Type", "Starting Date", and "Ending Date", each with a downward arrow. To the right of these is a blue "Search" button. Below the search filters is a large, empty rectangular box with a blue border, intended for displaying the query results.

### 3.16 Ride Sharing Page

This page allows customers to join other customers' reservations. By clicking the share button.



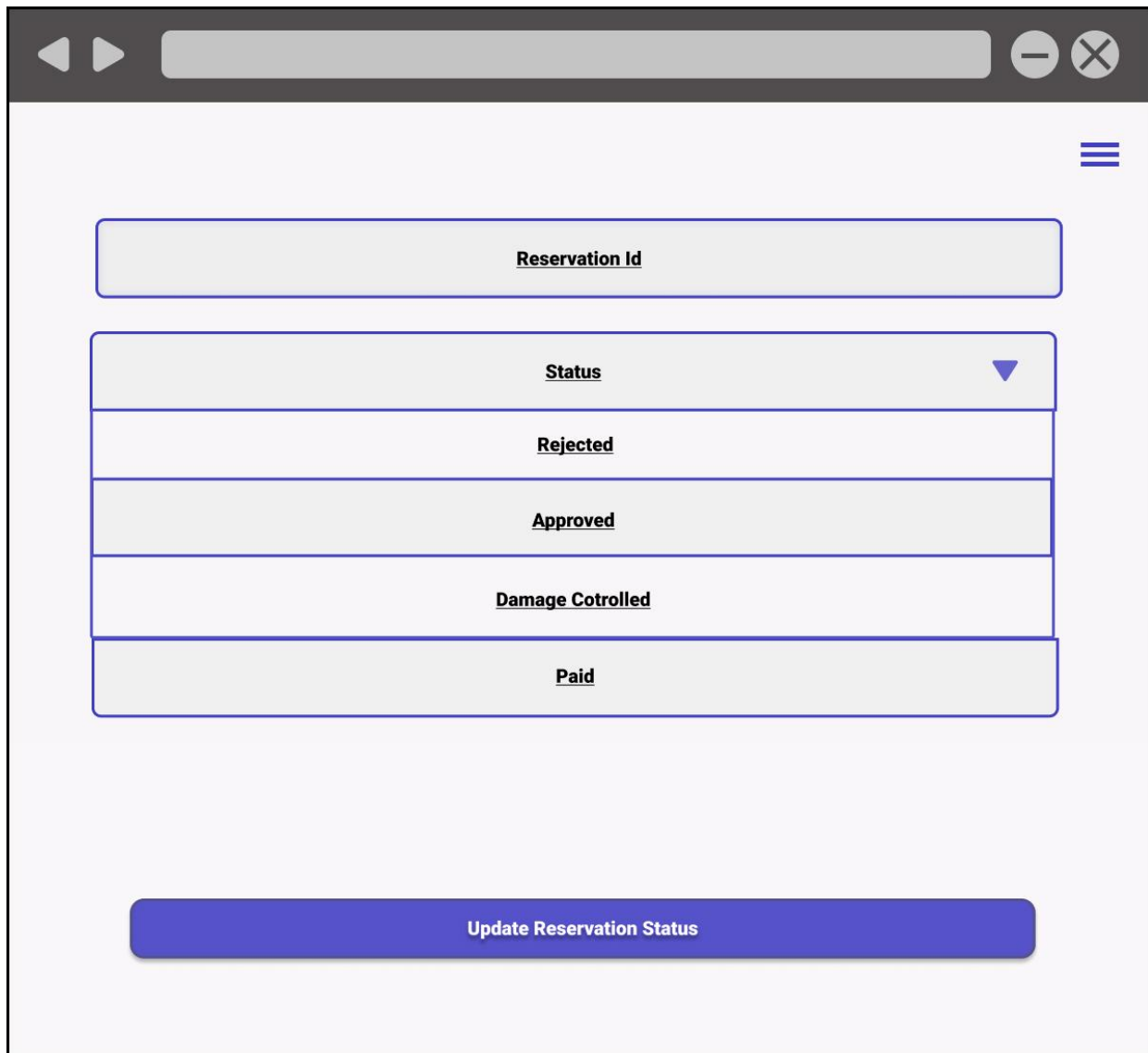
```
SELECT * FROM shared_ride SR
JOIN reservation R ON R.reservation_id = SR.reservation_id;

INSERT INTO customer_shared_ride (ride_id, passenger_id) VALUES (@ride_id,
@customer_id);
```



### 3.17 Reservation Status Update Page

This page lets employees and managers update the status of a given reservation.



The screenshot shows a web application interface for updating a reservation status. It features a browser window with a dark header bar containing navigation arrows, a search bar, and window control buttons. The main content area has a light purple background and a hamburger menu icon in the top right. The form consists of a text input for 'Reservation Id', a dropdown menu for 'Status' with options 'Rejected', 'Approved', 'Damage Cotrolled', and 'Paid', and a large blue button labeled 'Update Reservation Status' at the bottom.

```
UPDATE reservation SET status = @status WHERE reservation_id =  
@reservation_id;
```

## 4. IMPLEMENTATION PLAN

In the implementation, we are going to start by implementing our database using PostgreSQL.

We are going to use Java's Spring Framework for our web server. Spring JDBC will be used to query the database. Spring Security will be used for authenticating URLs, and Thymleaf will be used as the HTML templating tool. We are going to use a Controller Service Repository design where each controller is responsible for a designated page, and each repository is responsible for its respective tables. Services will be the components where we will encapsulate our business logic and error handling. For serving HTML documents, we will use Spring MVC's model view controller. For live data updates, we will use javascript together with the rest of the controllers in the server. For form validations, we will use javascript together with Thymeleaf. For improving the security and reliability of the system, we will create built-in assertions to our database. For styling, we will use the following tools: CSS, SASS, Bootstrap.

We can say that we are going to lay our project's foundations on PostgreSQL Database and Java Spring Framework.

## 5. WEBSITE

Website: <https://edemirkirkan.github.io/Car-Rental-System/>

Repository: <https://github.com/edemirkirkan/Car-Rental-System>