**SWE Vehicle Management System**

**Team members:**

Askaraliyev Nurzhigit - Design of website and application. Report and User Manual.

Chokushev Nursultan - Application Development (main page, tasks page, sandwich menu, driver, staff and automatic login)

Durmagambetova Aniyar - Backend for website and application

Issatay Ayan - Application Development (Fueling Person and Reporting)

Jussupov Nizami - Frontend for website and backend integration

Dinmukhamed Tolegenov - Application Development (Maintenance Person Interface and debugging)

**Preparation Stage**

**Stages:**

1. Use case
2. Class
3. Activity
4. ERD
5. SQL Script
6. Testing Database
7. Figma Design

**Web Interface**

<https://vms-sigma.vercel.app/auth/login>

**Pages:**

1. Login Page
2. New Users Registration Page
3. Vehicle Registration Page
4. Task Assignment Page

**Driver Application**

**Pages:**

1. Login Page
2. Tasks Page
3. Driver History Page
4. Driver Information Page
5. Main Page with Driver Location

**Preparation Stage**

Every Project Milestone completed in this course is included in this project report as they all contributed to the creation of the project. First, we did Use Case, Class, and Activity Diagrams to see the processes, classes, and their interactions schematically.

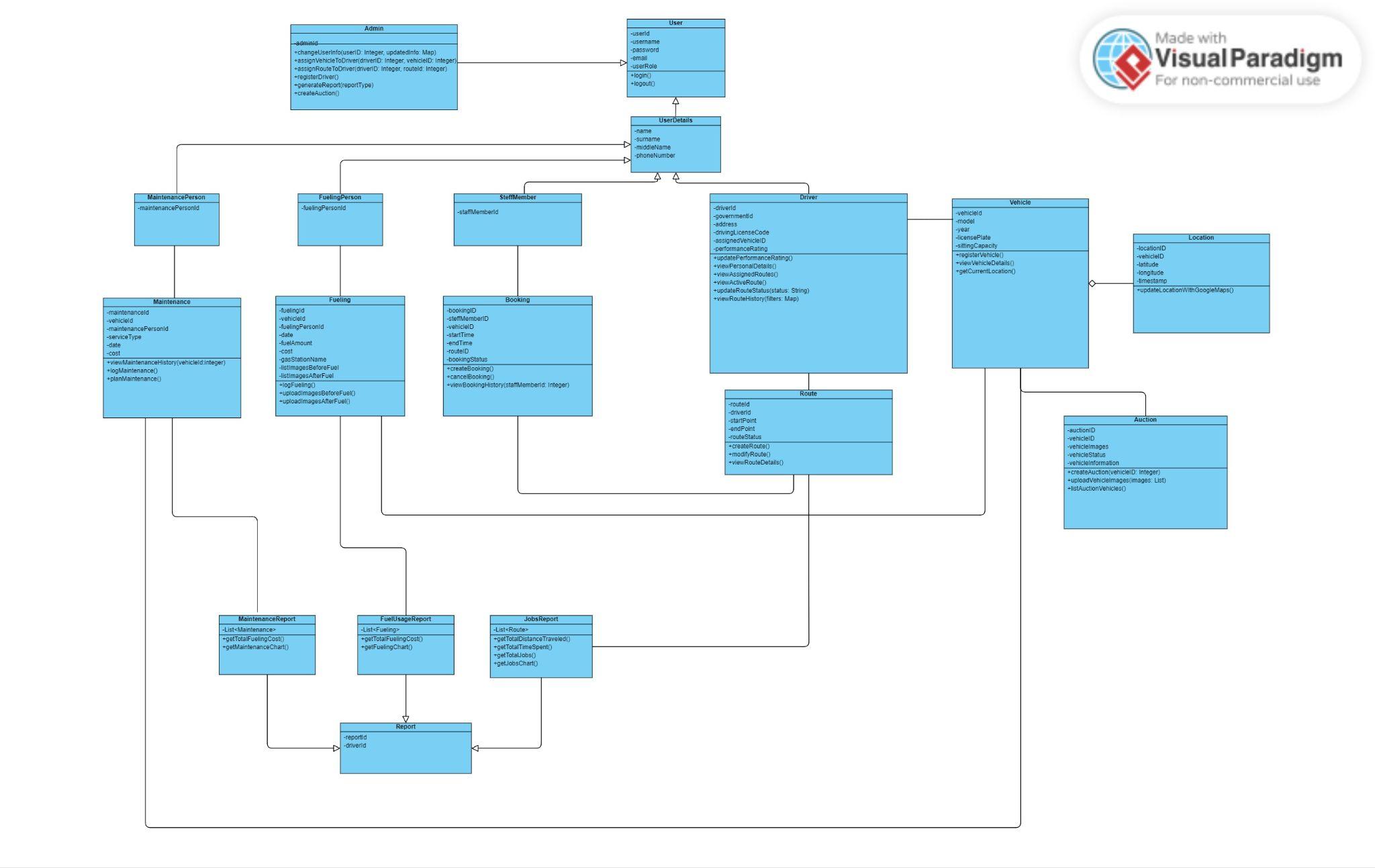


Figure 1. Class Diagram

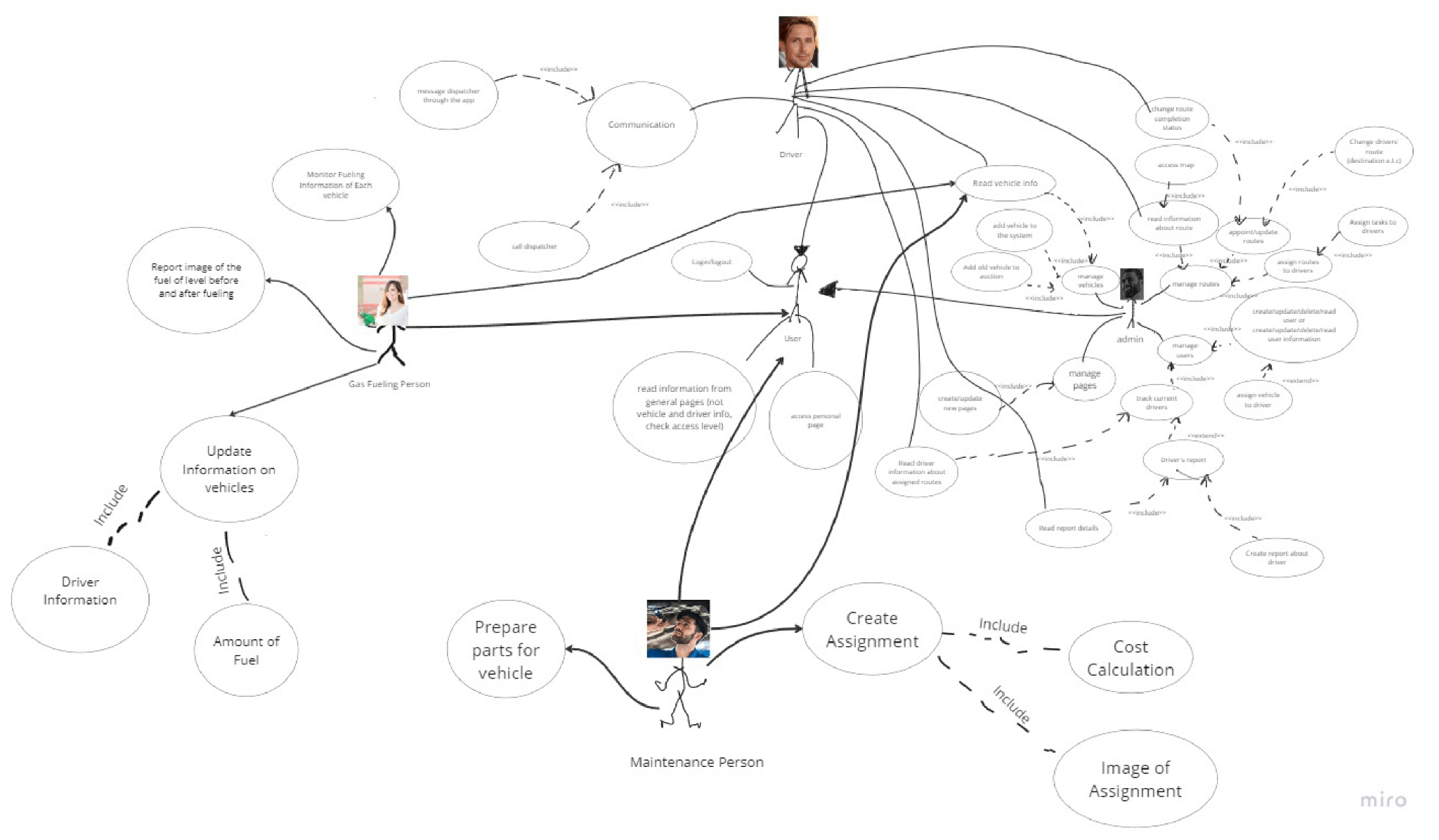


Figure 2. Use Case

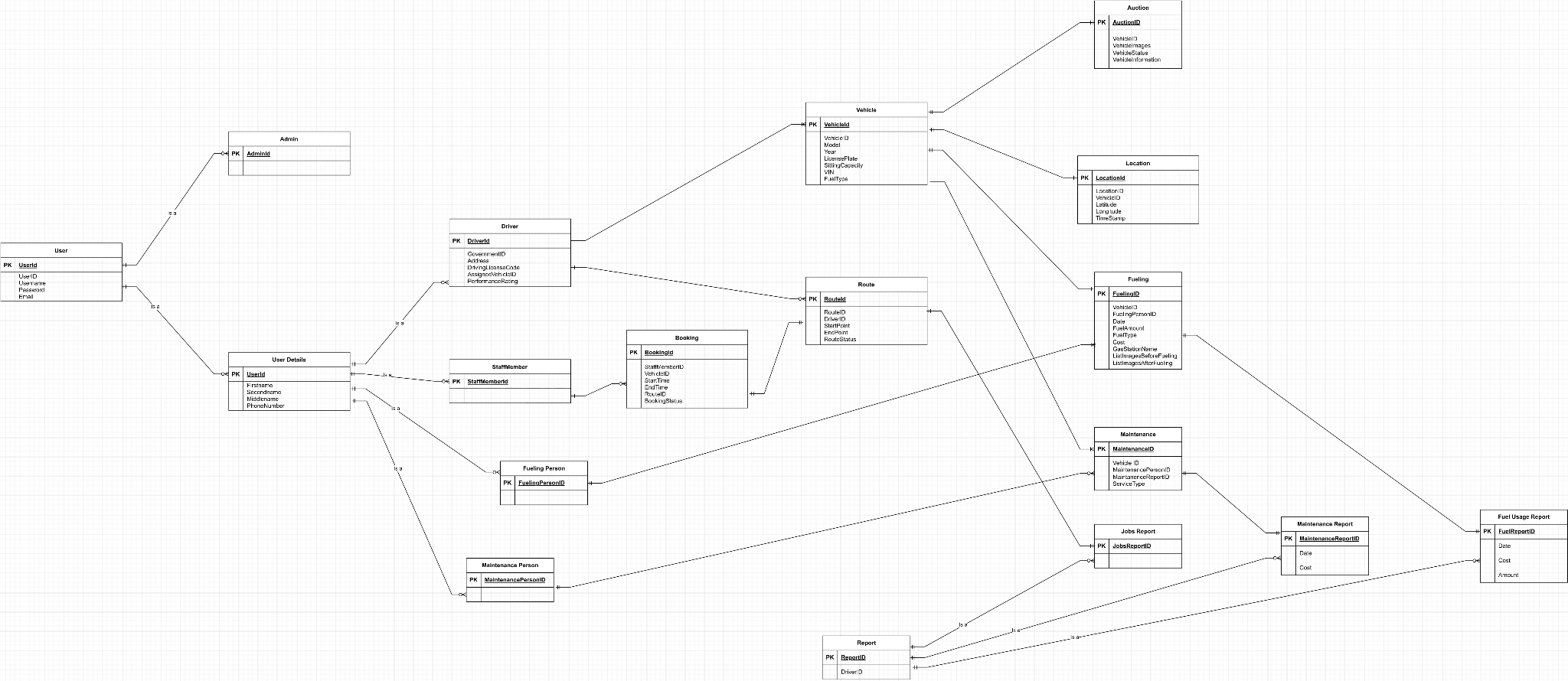


Figure 3. Entity-Relationship Diagram

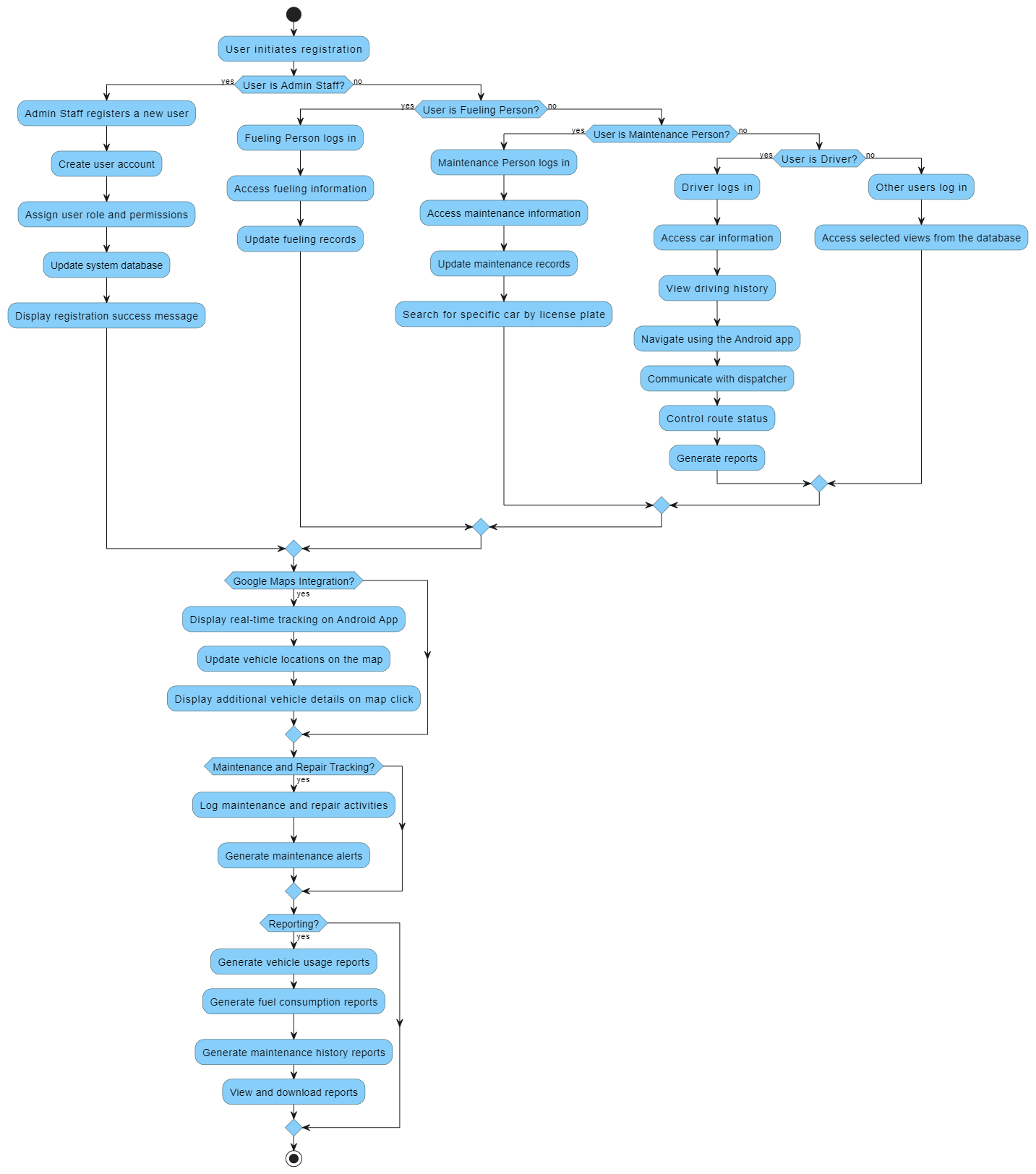


Figure 4. Activity Diagram

SQL Script

CREATE TABLE IF NOT EXISTS "User" (

UserId SERIAL PRIMARY KEY,

Username VARCHAR(50) NOT NULL,

Password VARCHAR(50) NOT NULL,

Email VARCHAR(100),

UserRole VARCHAR(20) NOT NULL

);

CREATE TABLE IF NOT EXISTS UserDetails (

UserDetailsId SERIAL PRIMARY KEY,

UserId INT NOT NULL,

Name VARCHAR(50),

Surname VARCHAR(50),

PhoneNumber VARCHAR(15),

FOREIGN KEY (UserId) REFERENCES "User"(UserId)

);

CREATE TABLE IF NOT EXISTS Admin (

AdminId SERIAL PRIMARY KEY,

UserId INT NOT NULL,

FOREIGN KEY (UserId) REFERENCES "User"(UserId)

);

CREATE TABLE IF NOT EXISTS MaintenancePerson (

MaintenancePersonId SERIAL PRIMARY KEY,

UserId INT NOT NULL,

FOREIGN KEY (UserId) REFERENCES "User"(UserId)

);

CREATE TABLE IF NOT EXISTS FuelingPerson (

FuelingPersonId SERIAL PRIMARY KEY,

UserId INT NOT NULL,

FOREIGN KEY (UserId) REFERENCES "User"(UserId)

);

CREATE TABLE IF NOT EXISTS StaffMember (

StaffMemberId SERIAL PRIMARY KEY,

UserId INT NOT NULL,

FOREIGN KEY (UserId) REFERENCES "User"(UserId)

);

CREATE TABLE IF NOT EXISTS Vehicle (

VehicleId SERIAL PRIMARY KEY,

Model VARCHAR(50) NOT NULL,

Year INT,

LicensePlate VARCHAR(15),

SittingCapacity INT,

VIN VARCHAR(17) NOT NULL,

FuelType VARCHAR(20)

);

CREATE TABLE IF NOT EXISTS Driver (

DriverId SERIAL PRIMARY KEY,

Address VARCHAR(255),

PhoneNumber VARCHAR(15),

DriversLicenseNumber VARCHAR(50),

PerformanceRating VARCHAR(50),

AssignedVehicleId INT,

FOREIGN KEY (AssignedVehicleId) REFERENCES Vehicle(VehicleId)

);

CREATE TABLE IF NOT EXISTS Route (

RouteId SERIAL PRIMARY KEY,

DriverId INT NOT NULL,

StartPoint VARCHAR(100),

EndPoint VARCHAR(100),

RouteStatus VARCHAR(20),

FOREIGN KEY (DriverId) REFERENCES Driver(DriverId)

);

CREATE TABLE IF NOT EXISTS Booking (

BookingId SERIAL PRIMARY KEY,

RouteId INT NOT NULL,

FOREIGN KEY (RouteId) REFERENCES Route(RouteId)

);

CREATE TABLE IF NOT EXISTS Report (

ReportId SERIAL PRIMARY KEY,

DriverId INT NOT NULL,

FOREIGN KEY (DriverId) REFERENCES Driver(DriverId)

);

CREATE TABLE IF NOT EXISTS Auction (

AuctionId SERIAL PRIMARY KEY,

VehicleId INT NOT NULL,

StartDate DATE NOT NULL,

EndDate DATE NOT NULL,

CurrentBid DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (VehicleId) REFERENCES Vehicle(VehicleId)

);

CREATE TABLE IF NOT EXISTS Location (

LocationId SERIAL PRIMARY KEY,

VehicleId INT NOT NULL,

Coordinates VARCHAR(255) NOT NULL,

FOREIGN KEY (VehicleId) REFERENCES Vehicle(VehicleId)

);

CREATE TABLE IF NOT EXISTS Fueling (

FuelId SERIAL PRIMARY KEY,

FuelType VARCHAR(255) NOT NULL,

Price DECIMAL(10, 2) NOT NULL

);

CREATE TABLE IF NOT EXISTS Maintenance (

MaintenanceId SERIAL PRIMARY KEY,

VehicleId INT NOT NULL,

MaintenancePersonId INT NOT NULL,

ServiceType VARCHAR(255) NOT NULL,

FOREIGN KEY (VehicleId) REFERENCES Vehicle(VehicleId),

FOREIGN KEY (MaintenancePersonId) REFERENCES

MaintenancePerson(MaintenancePersonId)

);

CREATE TABLE IF NOT EXISTS JobsReport (

ReportId SERIAL PRIMARY KEY,

DriverId INT NOT NULL,

FOREIGN KEY (DriverId) REFERENCES Driver(DriverId)

);

CREATE TABLE IF NOT EXISTS FuelUsageReport (

ReportId SERIAL PRIMARY KEY,

Date DATE NOT NULL,

Cost DECIMAL(10, 2) NOT NULL,

Amount INT NOT NULL,

FOREIGN KEY (ReportId) REFERENCES Report(ReportId)

);

Testing Database

-- Inserting data into User table

INSERT INTO "User"(UserId, Username, Password, Email, UserRole)

VALUES (1, 'jdoe', 'password123', 'jdoe@example.com', 'Driver');

INSERT INTO "User"(UserId, Username, Password, Email, UserRole)

VALUES (2, 'jdoeAdmin', 'password123', 'jdoeAdmin@example.com',

'Admin');

INSERT INTO "User"(UserId, Username, Password, Email, UserRole)

VALUES (3, 'jdoeMaintenance', 'password123',

'jdoeMaintenance@example.com', 'Maintenance');

INSERT INTO "User"(UserId, Username, Password, Email, UserRole)

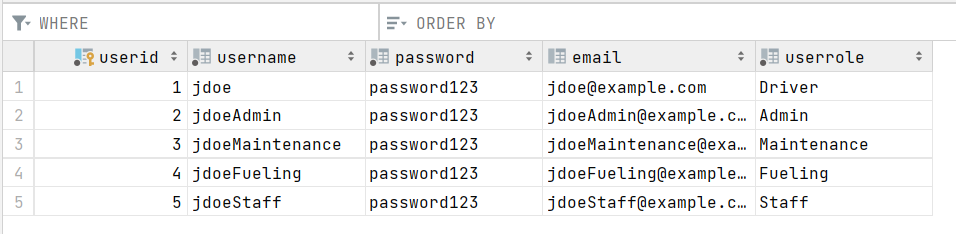
VALUES (4, 'jdoeFueling', 'password123', 'jdoeFueling@example.com',

'Fueling');

INSERT INTO "User"(UserId, Username, Password, Email, UserRole)

VALUES (5, 'jdoeStaff', 'password123', 'jdoeStaff@example.com',

'Staff');

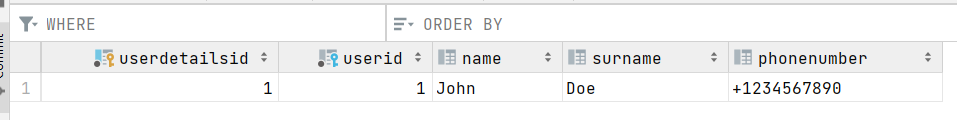


-- Inserting data into UserDetails table

INSERT INTO UserDetails(UserDetailsId, UserId, Name, Surname,

PhoneNumber)

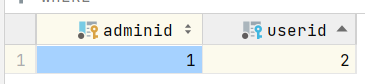
VALUES (1, 1, 'John', 'Doe', '+1234567890');



-- Inserting data into Admin table

INSERT INTO Admin(AdminId, UserId)

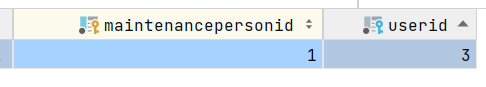
VALUES (1, 2);



-- Inserting data into MaintenancePerson table

INSERT INTO MaintenancePerson(MaintenancePersonId, UserId)

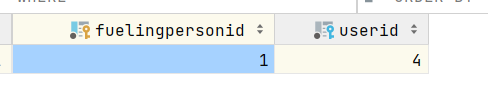
VALUES (1, 3);



-- Inserting data into FuelingPerson table

INSERT INTO FuelingPerson(FuelingPersonId, UserId)

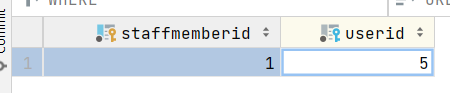
VALUES (1, 4);



-- Inserting data into StaffMember table

INSERT INTO StaffMember(StaffMemberId, UserId)

VALUES (1, 5);

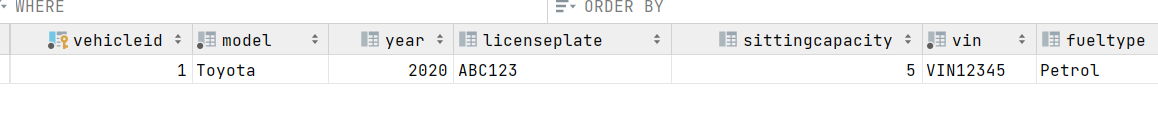


-- Inserting data into Vehicle table

INSERT INTO Vehicle(VehicleId, Model, Year, LicensePlate,

SittingCapacity, VIN, FuelType)

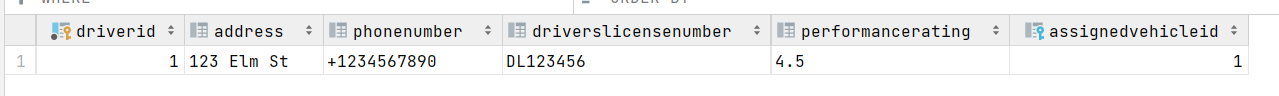
VALUES (1, 'Toyota', 2020, 'ABC123', 5, 'VIN12345', 'Petrol');

-- Inserting data into Driver table

INSERT INTO Driver(DriverId, Address, PhoneNumber,

DriversLicenseNumber, PerformanceRating, AssignedVehicleId)

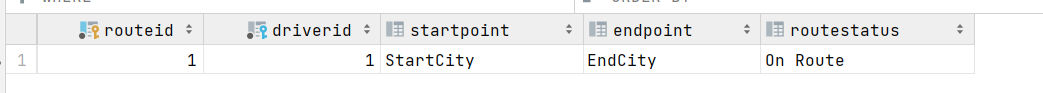
VALUES (1, '123 Elm St', '+1234567890', 'DL123456', 4.5, 1);

-- Inserting data into Route table

INSERT INTO Route(RouteId, DriverId, StartPoint, EndPoint,

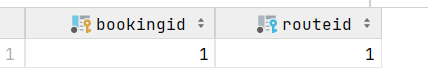
RouteStatus)

VALUES (1, 1, 'StartCity', 'EndCity', 'On Route');

-- Inserting data into Booking table

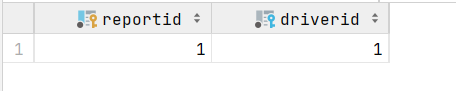
INSERT INTO Booking(BookingId, RouteId)

VALUES (1, 1);



-- Inserting data into Report table

INSERT INTO Report(ReportId, DriverId)

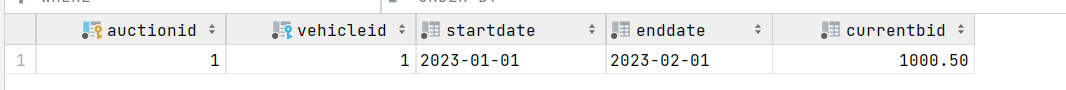
VALUES (1, 1);

-- Inserting data into Auction table

INSERT INTO Auction(AuctionId, VehicleId, StartDate, EndDate,

CurrentBid)

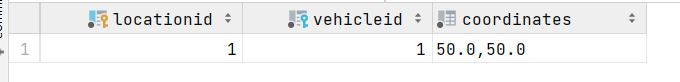
VALUES (1, 1, '2023-01-01', '2023-02-01', 1000.50);



-- Inserting data into Location table

INSERT INTO Location(LocationId, VehicleId, Coordinates)

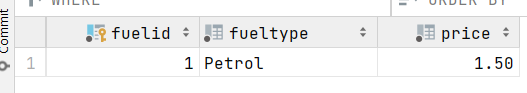
VALUES (1, 1, '50.0,50.0');



-- Inserting data into Fueling table

INSERT INTO Fueling(FuelId, FuelType, Price)

VALUES (1, 'Petrol', 1.50);

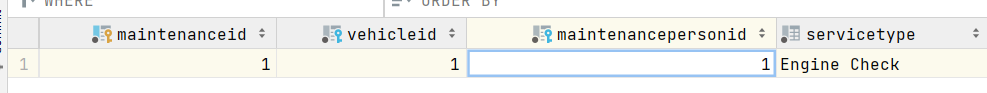


-- Inserting data into Maintenance table

INSERT INTO Maintenance(MaintenanceId, VehicleId, MaintenancePersonId,

ServiceType)

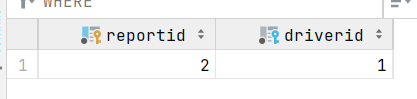
VALUES (1, 1, 1, 'Engine Check');



-- Inserting data into JobsReport table

INSERT INTO JobsReport(ReportId, DriverId)

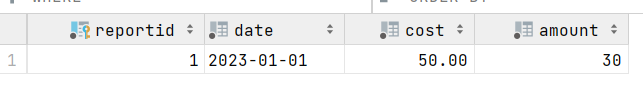
VALUES (2, 1);



-- Inserting data into FuelUsageReport table

INSERT INTO FuelUsageReport(ReportId, Date, Cost, Amount)

VALUES (1, '2023-01-01', 50.00, 30);

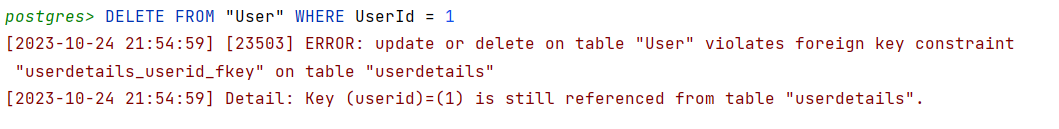


-- Check that user details can't exist without a user

DELETE FROM "User" WHERE UserId = 1;

-- This should raise a foreign key constraint error because of

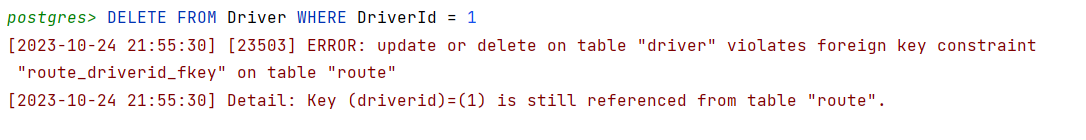
UserDetails



-- Check that a route can't exist without a driver

DELETE FROM Driver WHERE DriverId = 1;

-- This should raise a foreign key constraint error because of Route



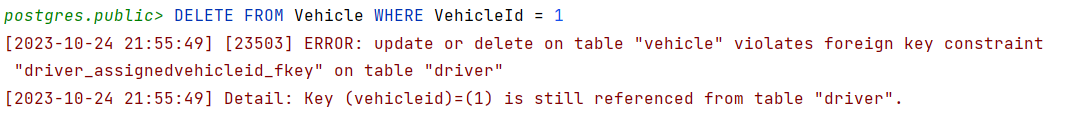
-- Check that a vehicle can't be deleted if it's referenced in an

auction

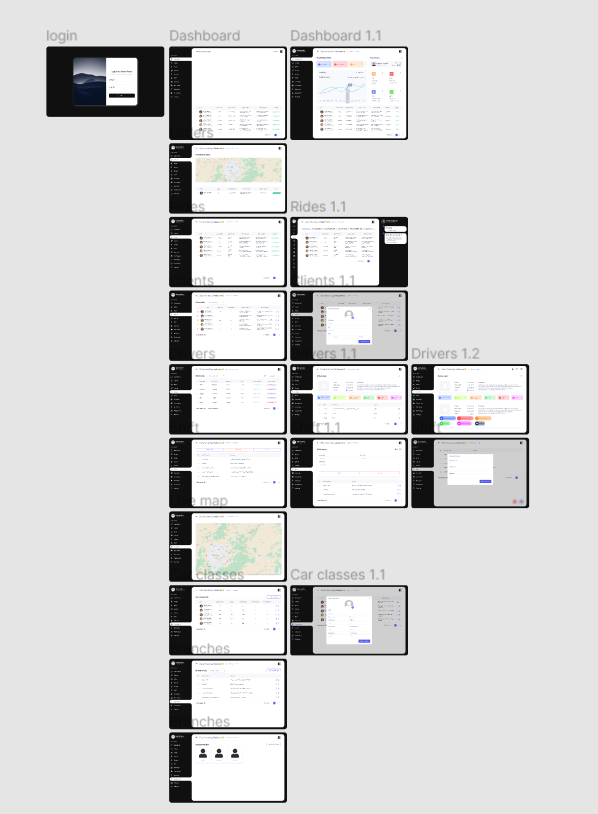
DELETE FROM Vehicle WHERE VehicleId = 1;

-- This should raise a foreign key constraint error because of

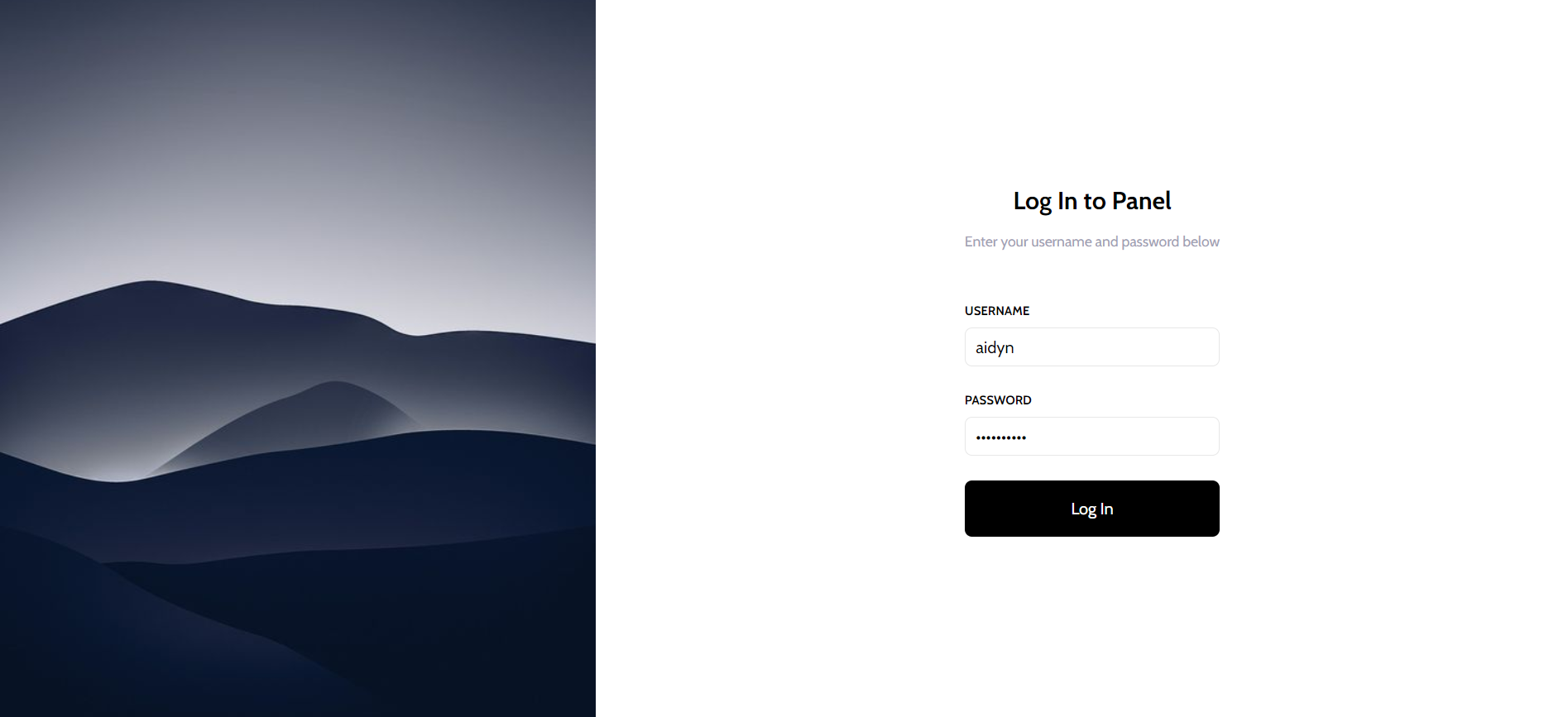
either Auction or Driver



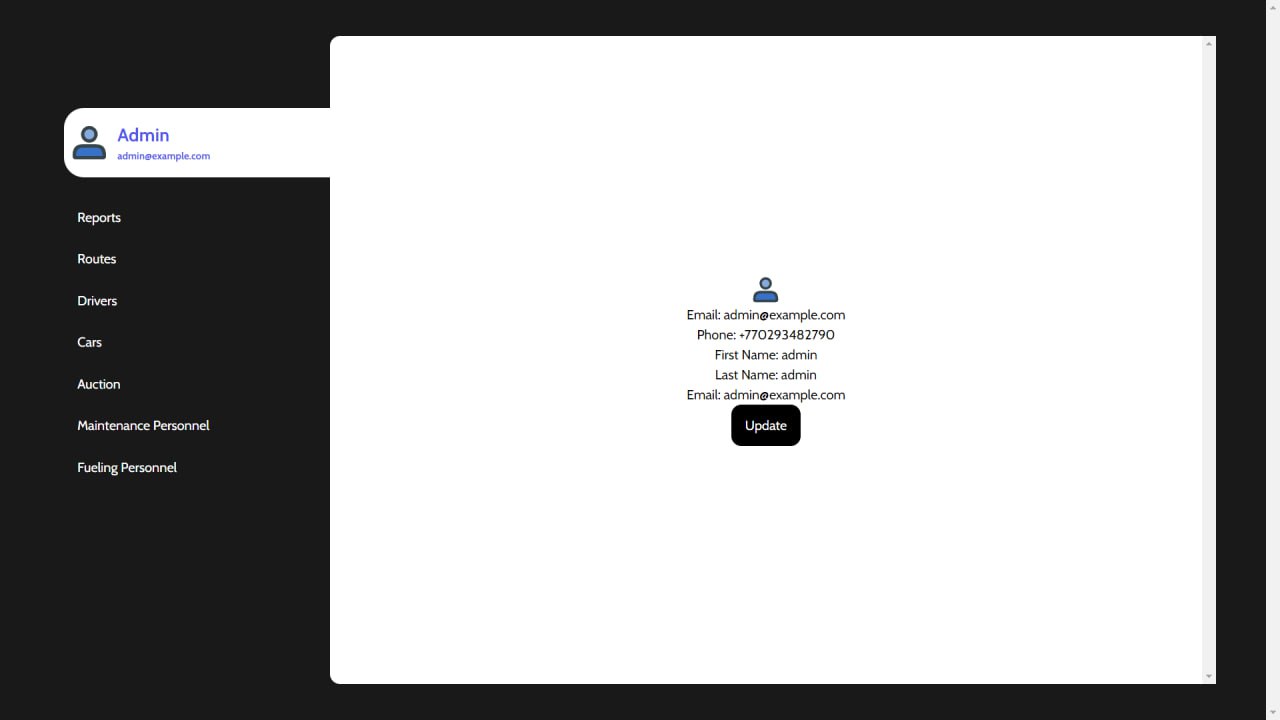
Figma Design



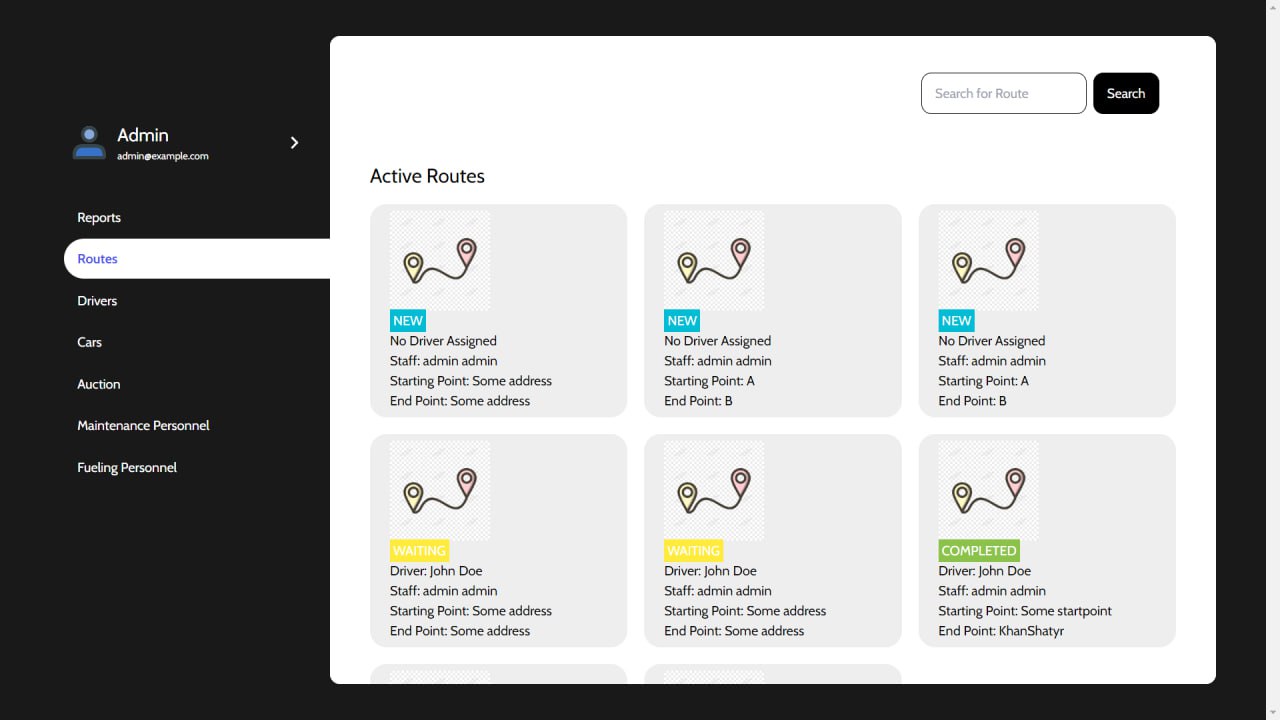
Web



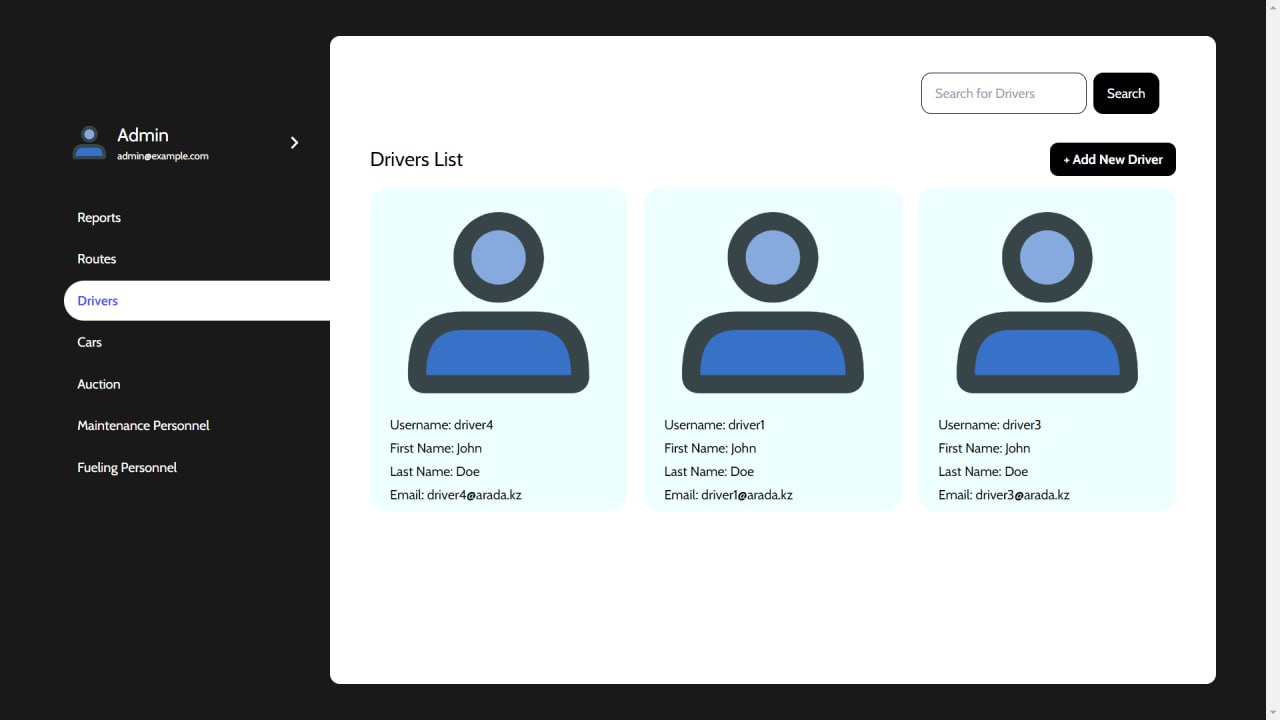
Login page looks like this and allows the administrator to enter the system using his credentials. If successful, he is sent to the main dashboard.



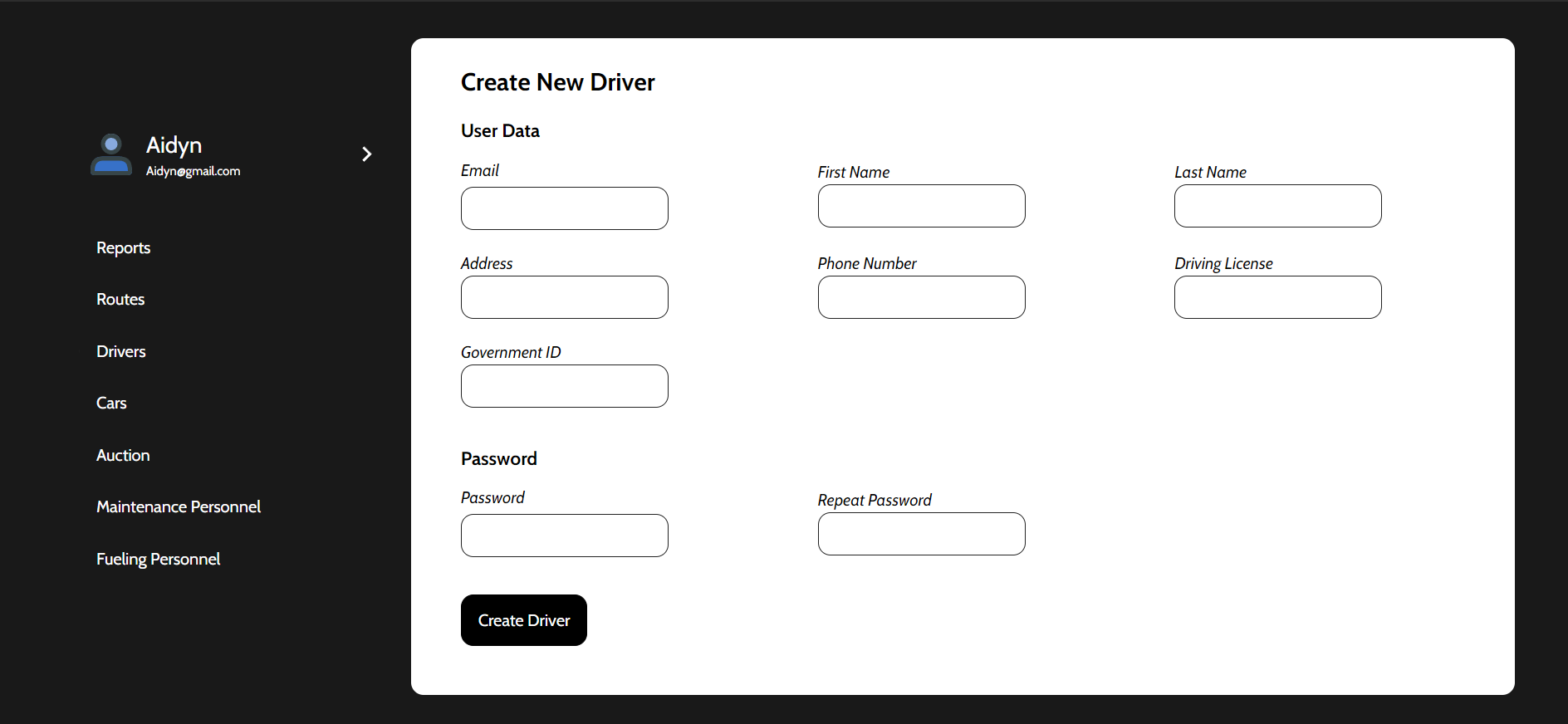
On the first page administrator is redirected to the main page with his information credentials and a dashboard navigation menu on the left.



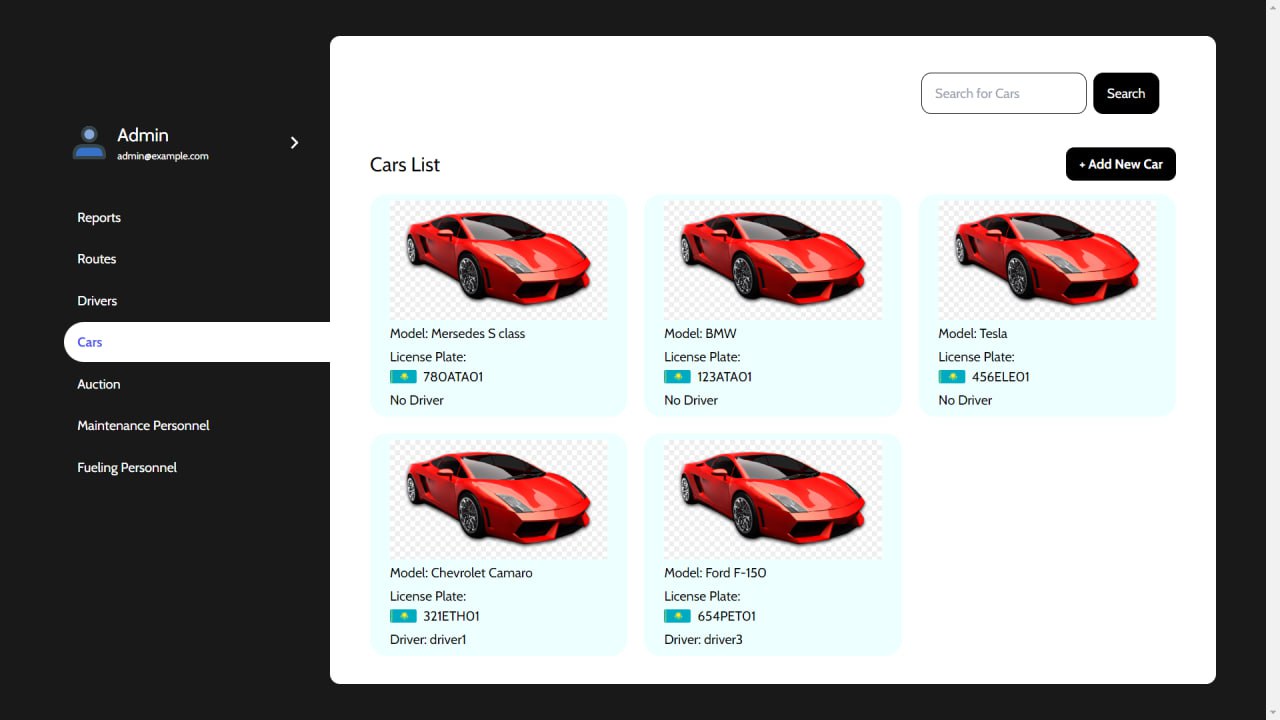
In one of the pages of the admin dashboard he can see all of the active and inactive appointments or tasks that get assigned to each driver individually by the administrator. They provide information on the starting and end destinations. Administrators can add new appointments and save them to the database.



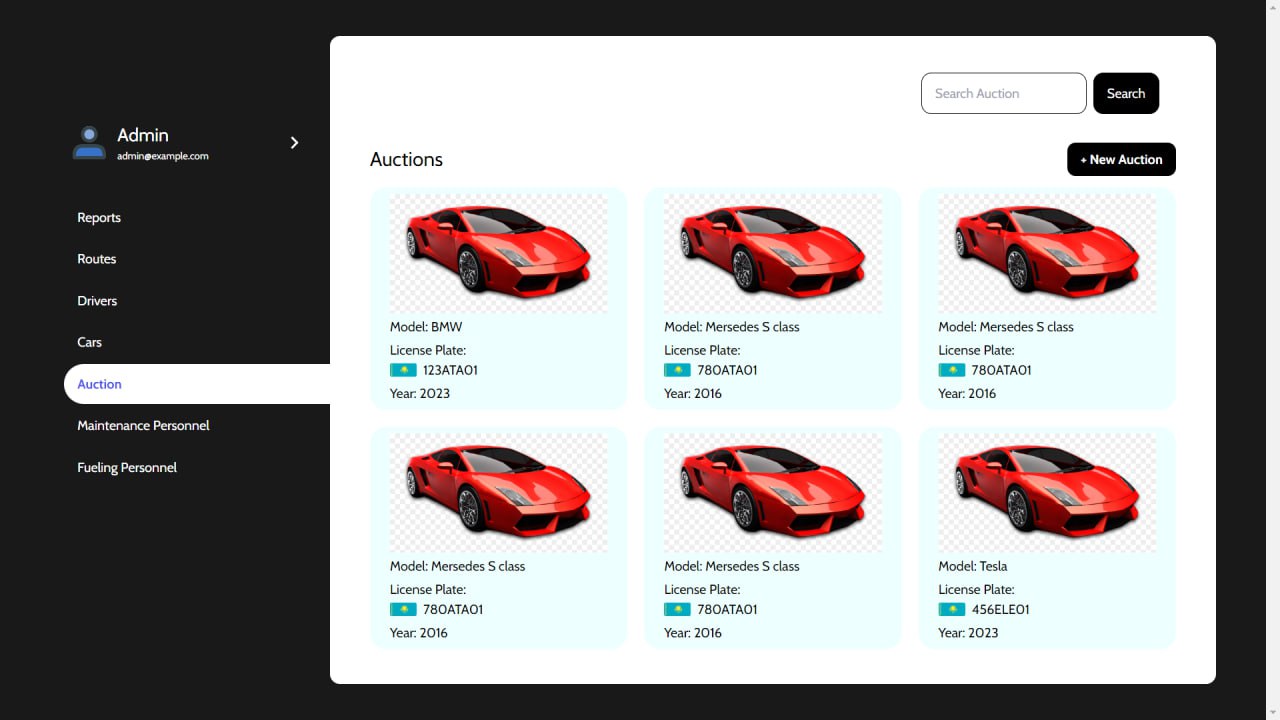
Here is the drivers list that shows all the relevant information about the drivers, whom the administrator can contact either via email or phone. If the drivers list becomes too big in the future, there is a search bar to help administrators find needed drivers faster. Above the table, there is a button, which redirects to the next page.



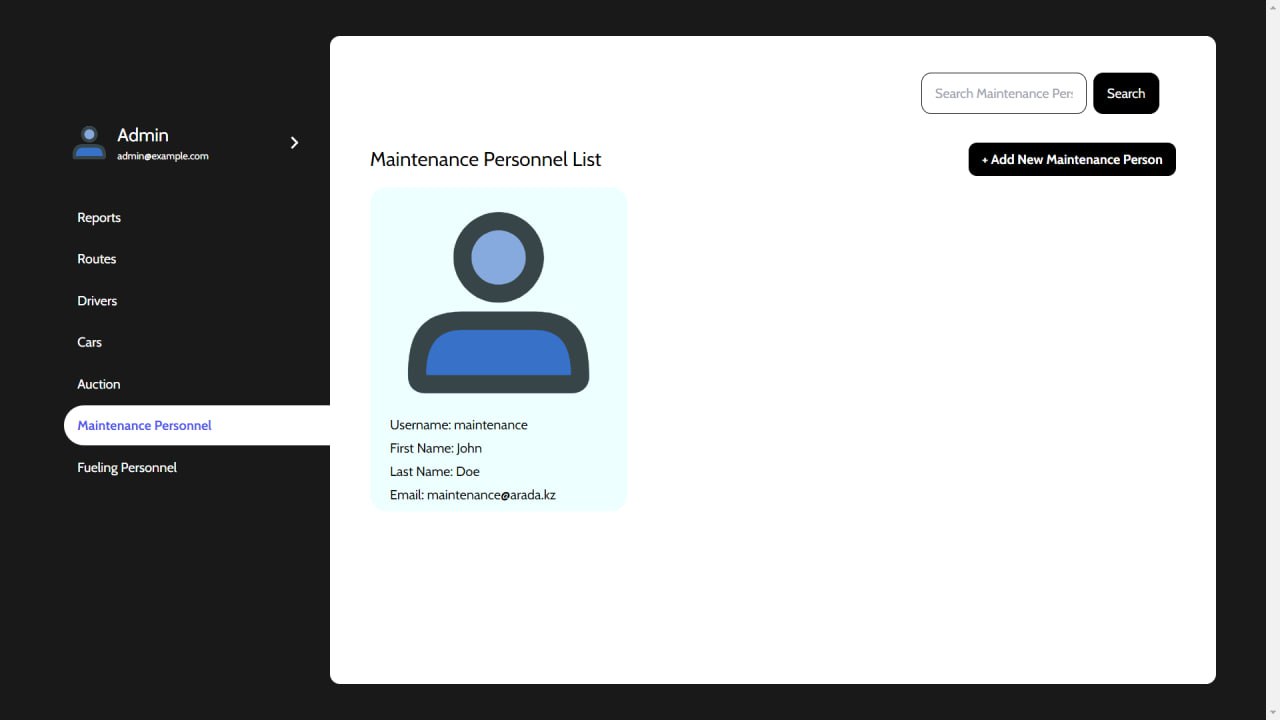
Here the administrator can fill in all information about the new driver and send it to the database. Pretty straightforward new user registration process.

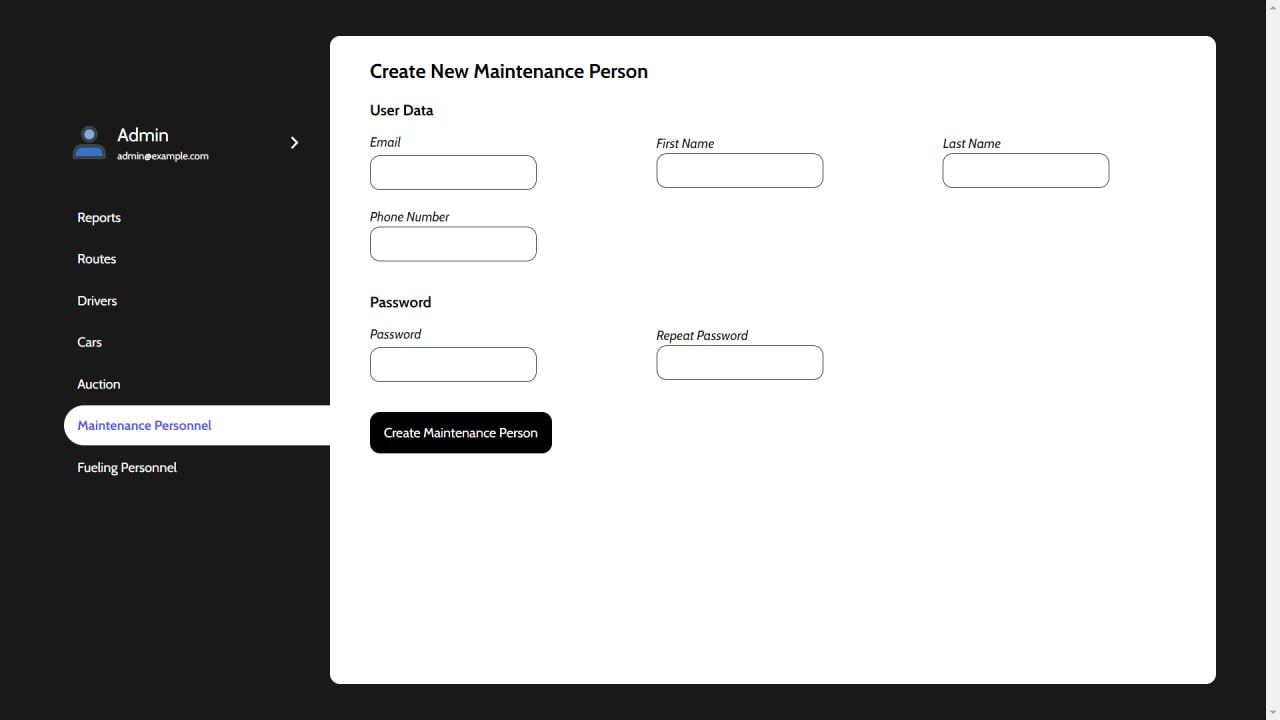


Regarding the cars, we have similar functionality as with the drivers list.

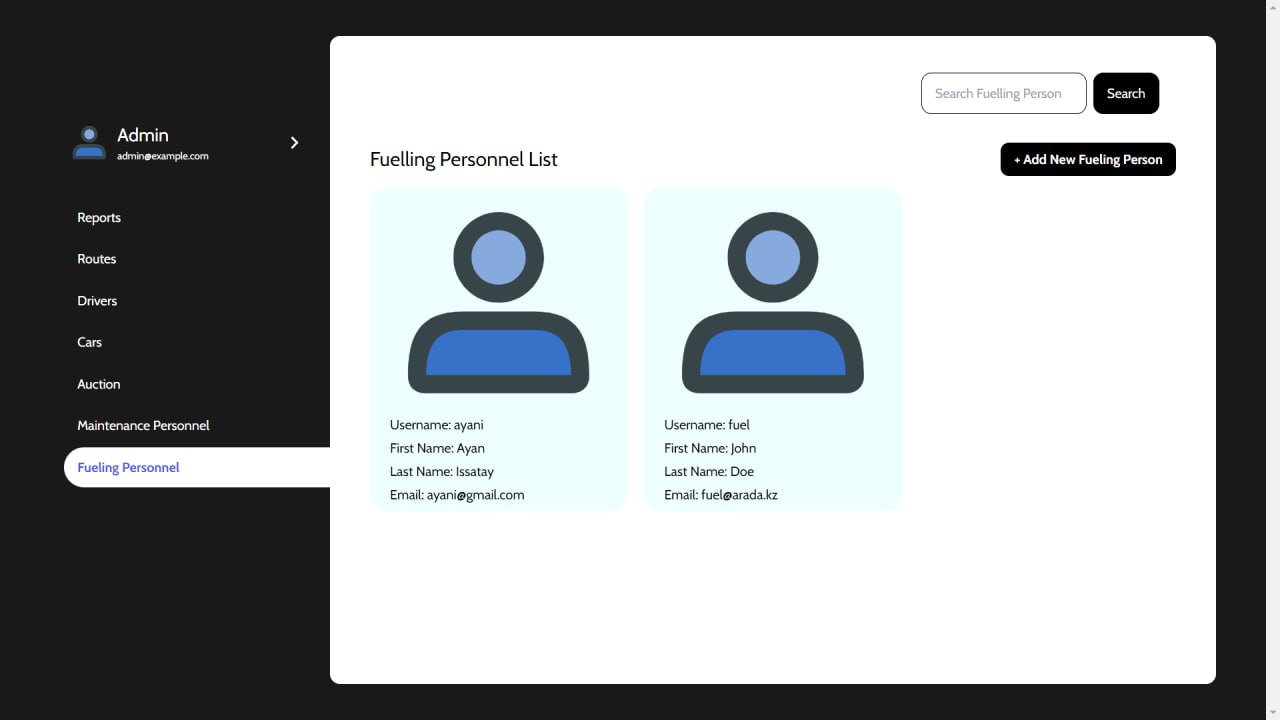


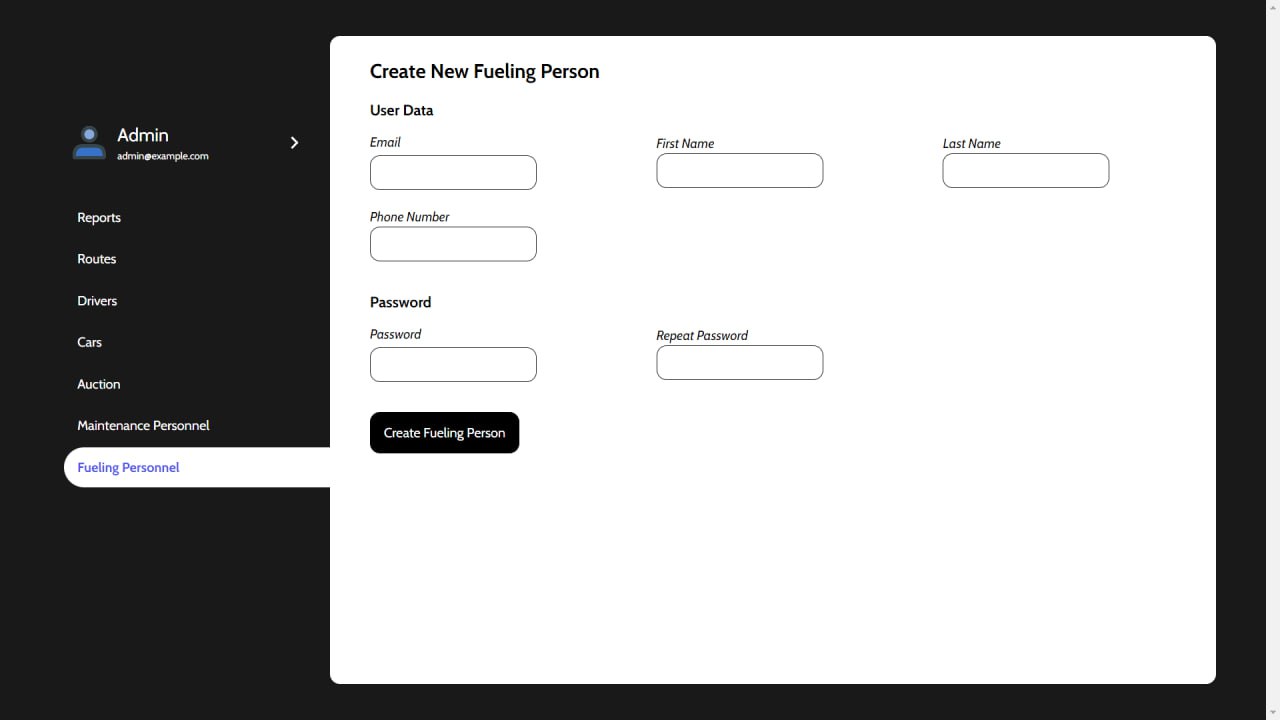
If the new car is added to the system, it is mandatory to fill in the information about the model, year, fuel amount, license plate, and sitting capacity.



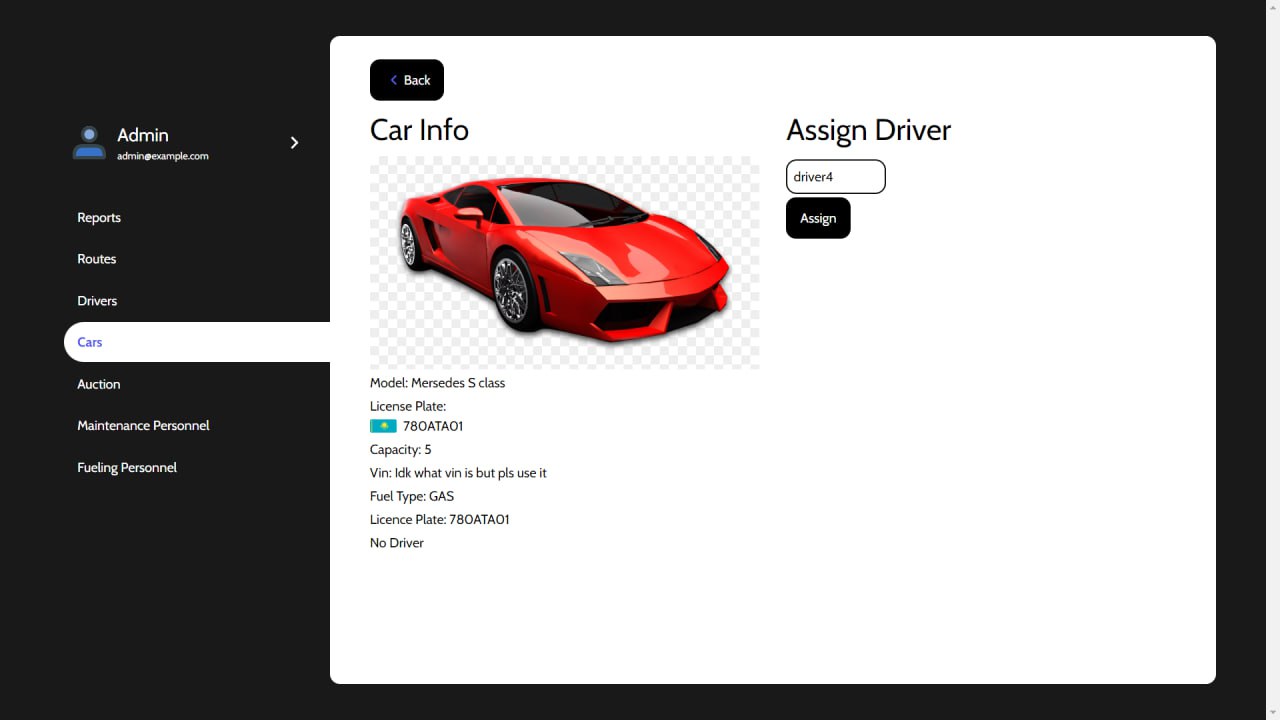


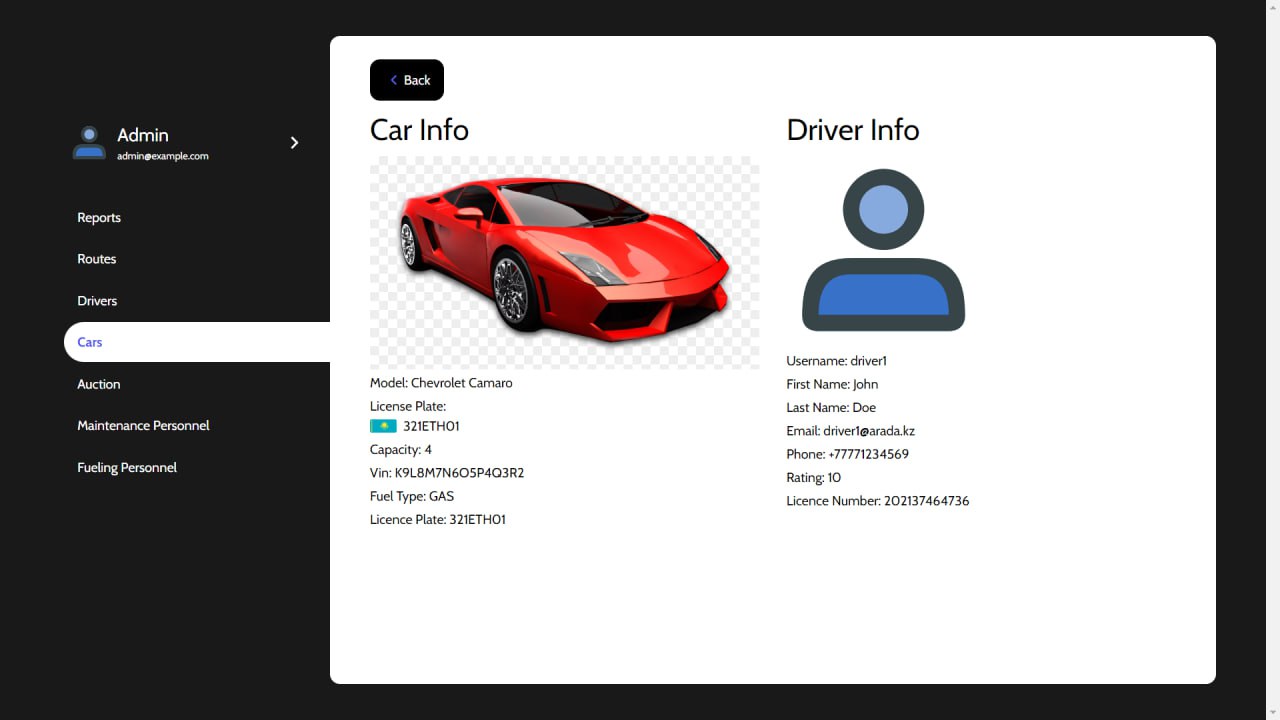
Maintenance people are also a part of the system, so we made their list and registration similar to those of the drivers.

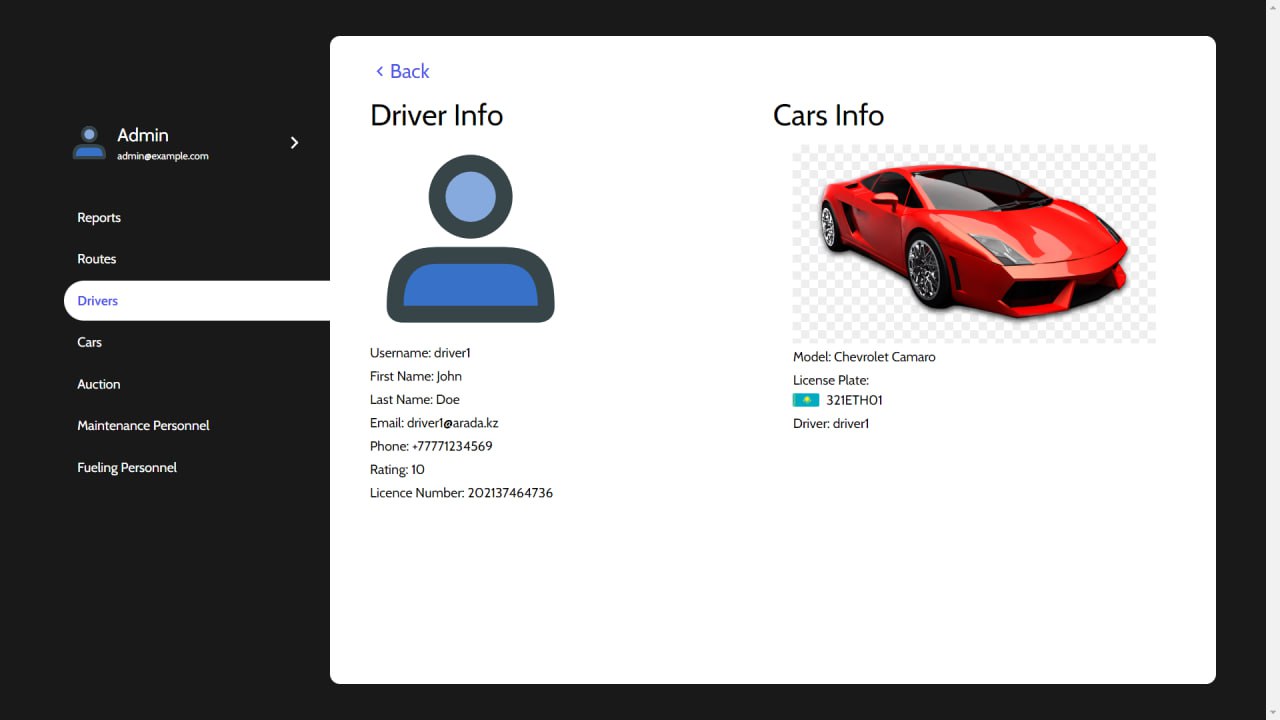


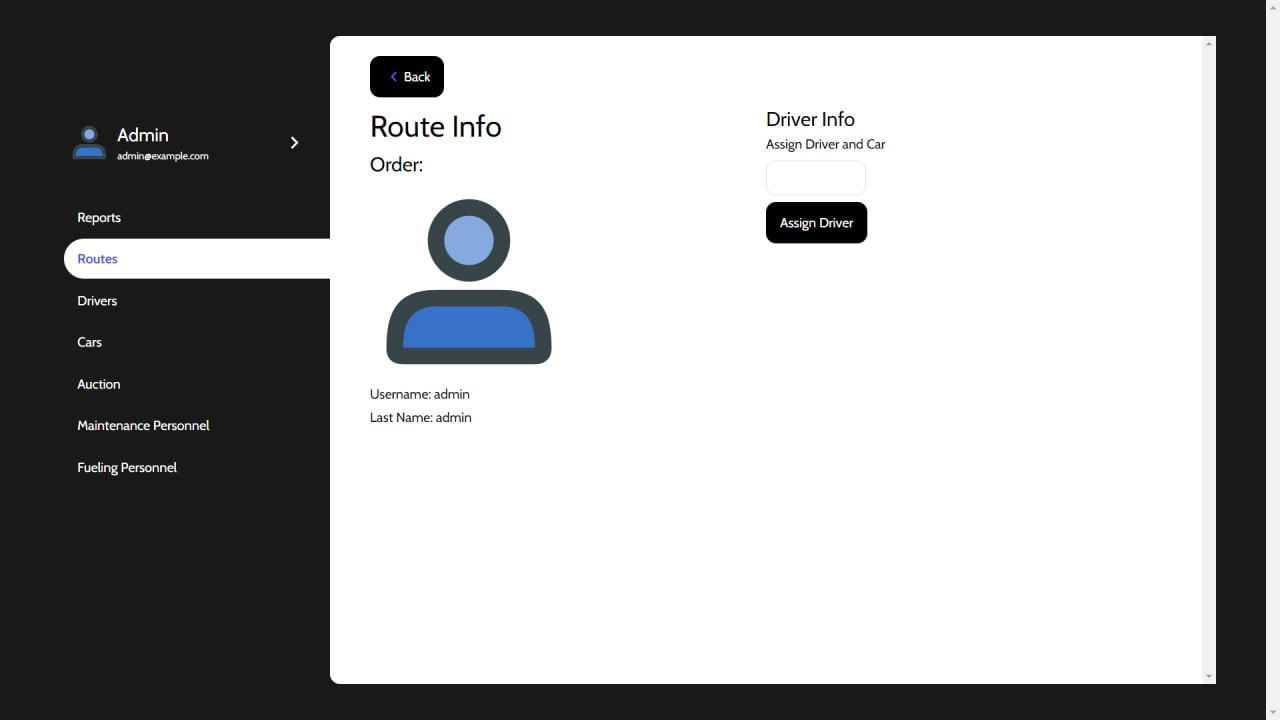


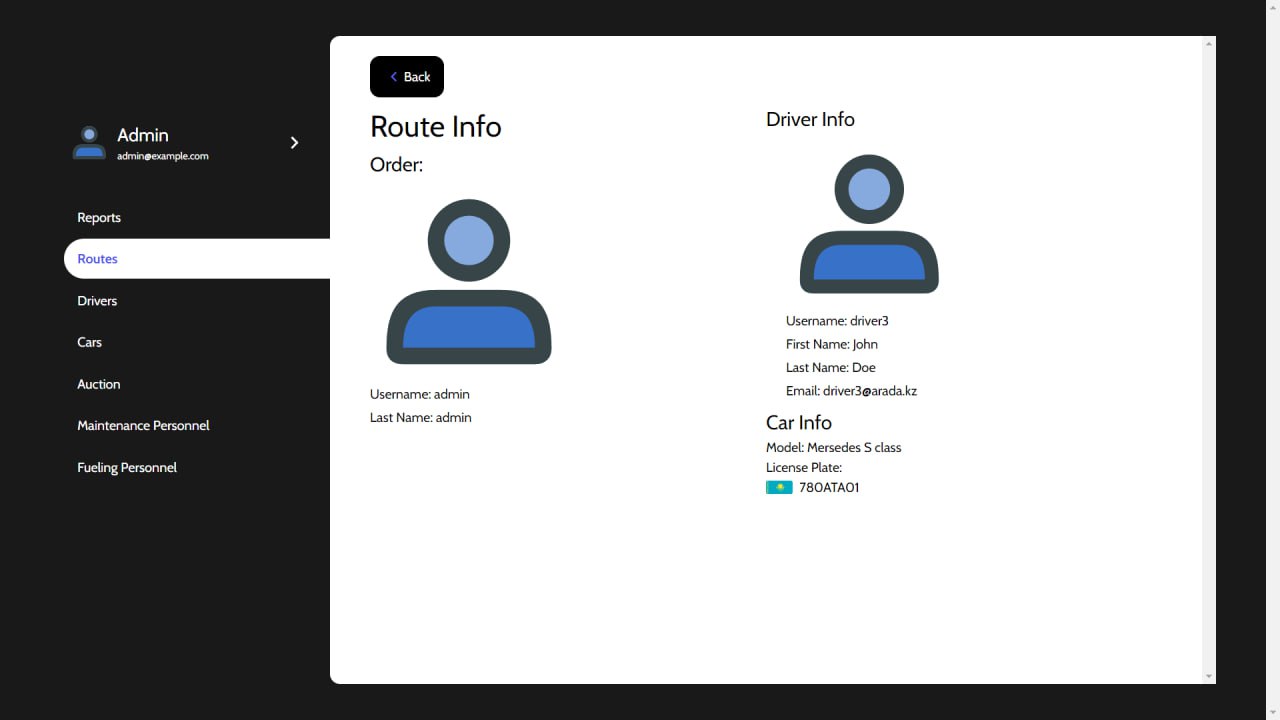
Fueling Personnel list and registration is similar to the Drivers’ and Maintenance’s.





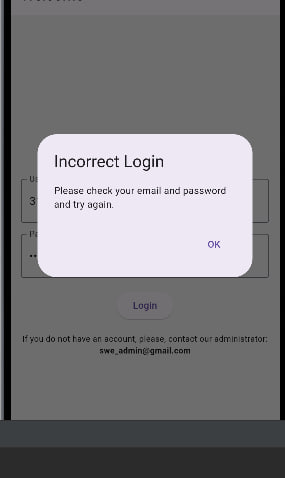




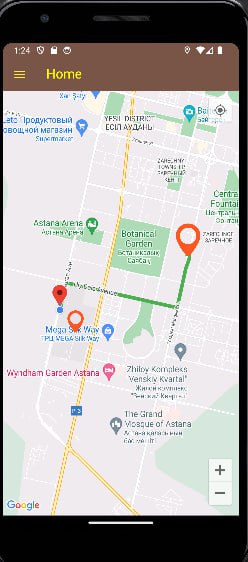
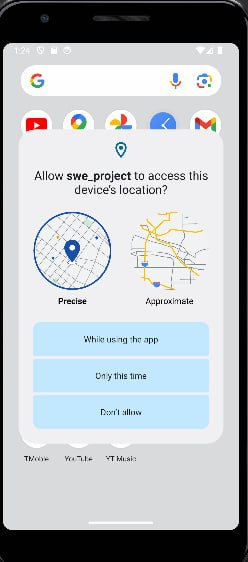


Application

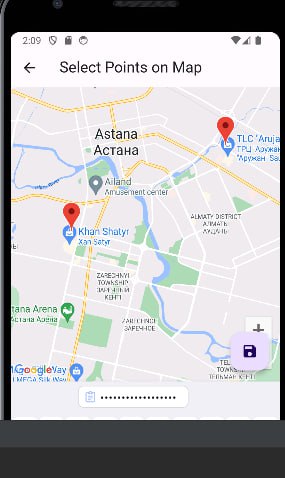
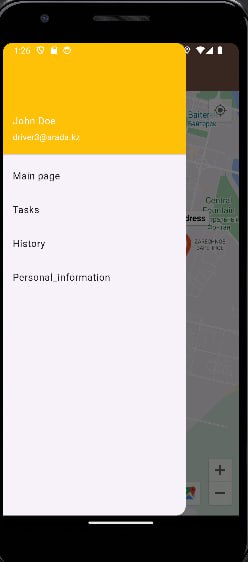
When welcomed in the application, the user is shown a login page, where he or she needs to enter correct username and password, or if not registered in the system, contact the administrator and provide details further explained by the administrator.



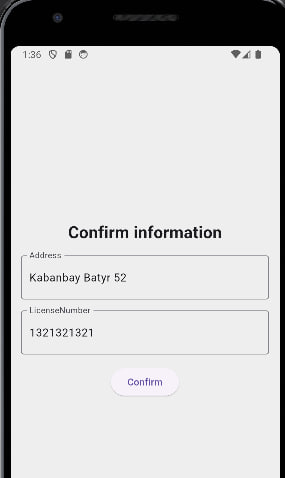
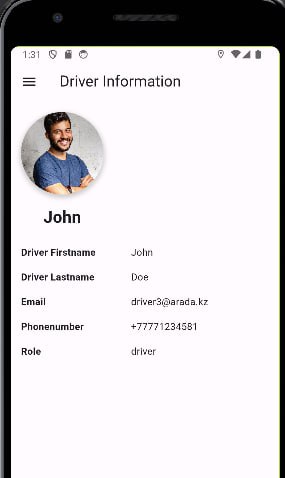
After successful login, the user is redirected to the Main Page with the map that shows the user's current location on the map. It also shows the destination and the path to it, making it easier for drivers to navigate.



Here, there is a sandwich menu for the smooth navigation of the user between tabs. There is a map with the user's location and if he is assigned any tasks, then he gets them shown on the map as well.



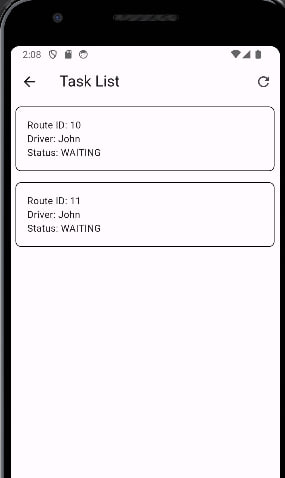
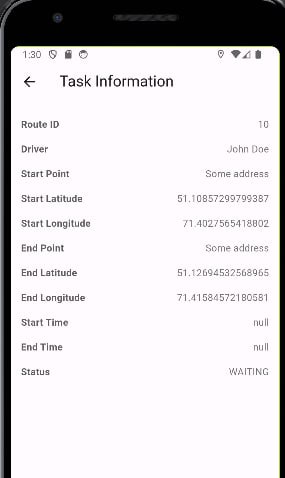
Users can see information about himself and overall statistics. If the provided information is incorrect he can contact the administrator to correct it.



For this user, currently there are no tasks or driving history; however, we will correct it in the future versions of this application.

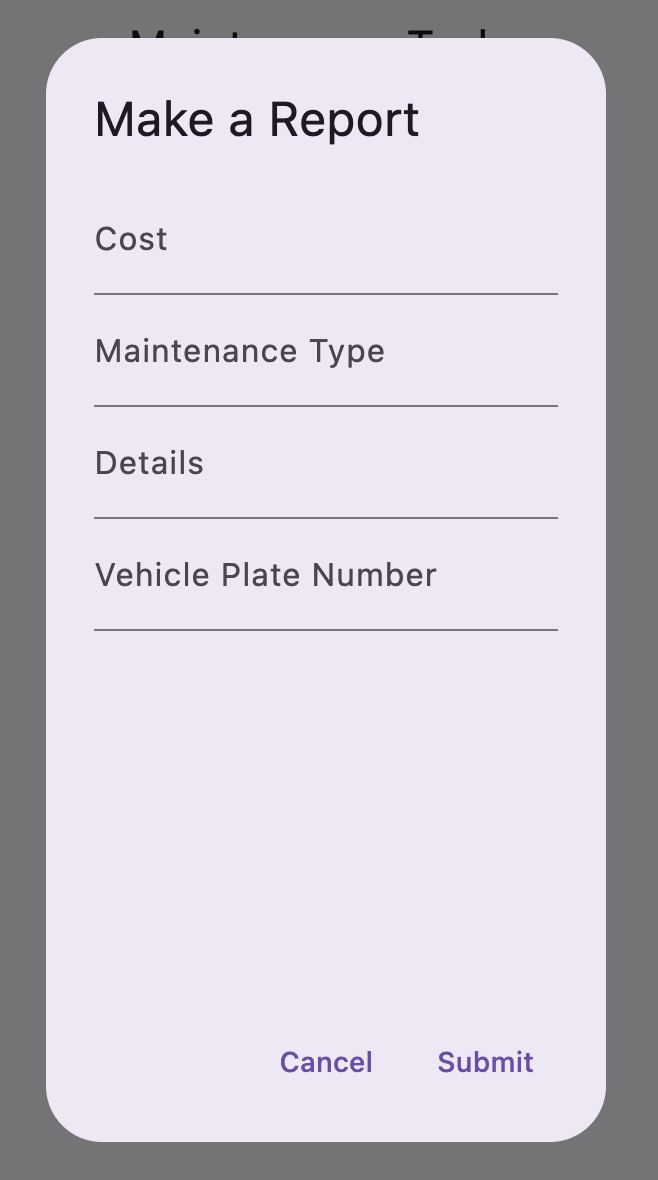
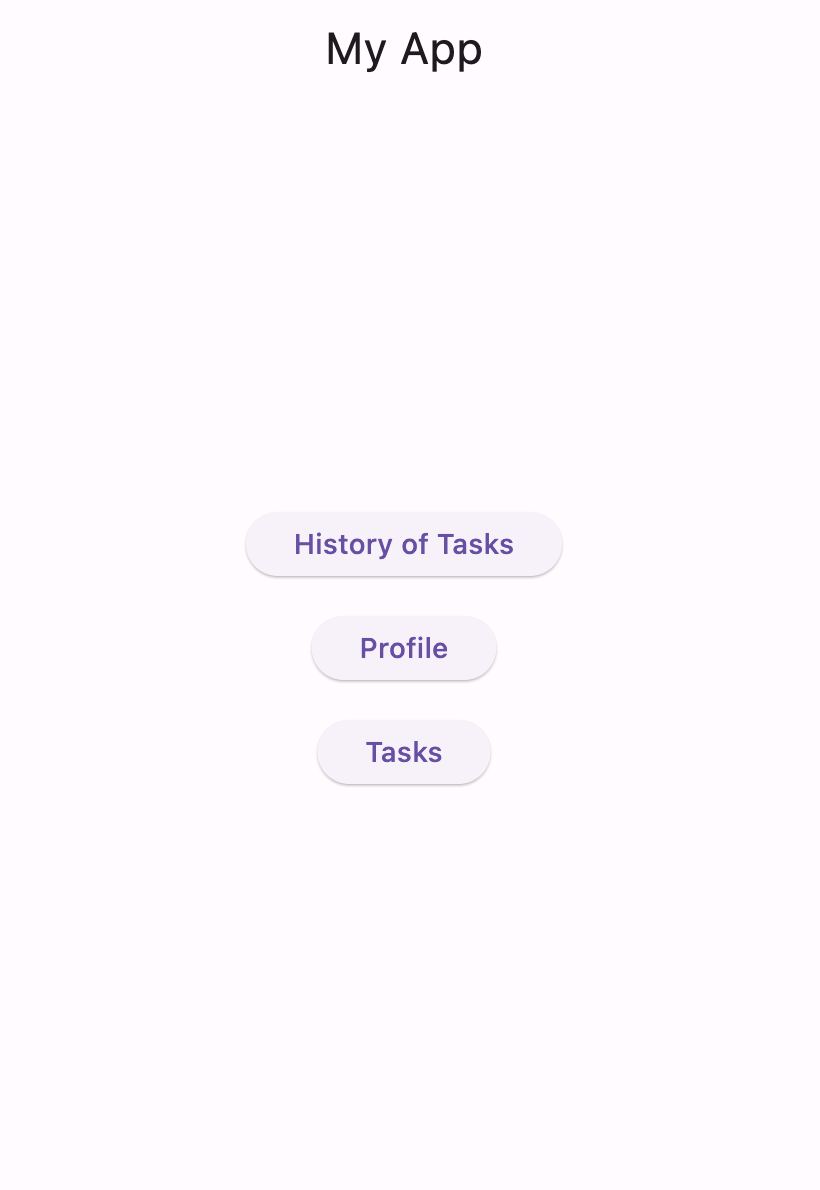


Task information is provided above in the little boxes where you see the Driver, RouteID, and the status of the Task. There are two different views for the Driver and Maintenance Person.

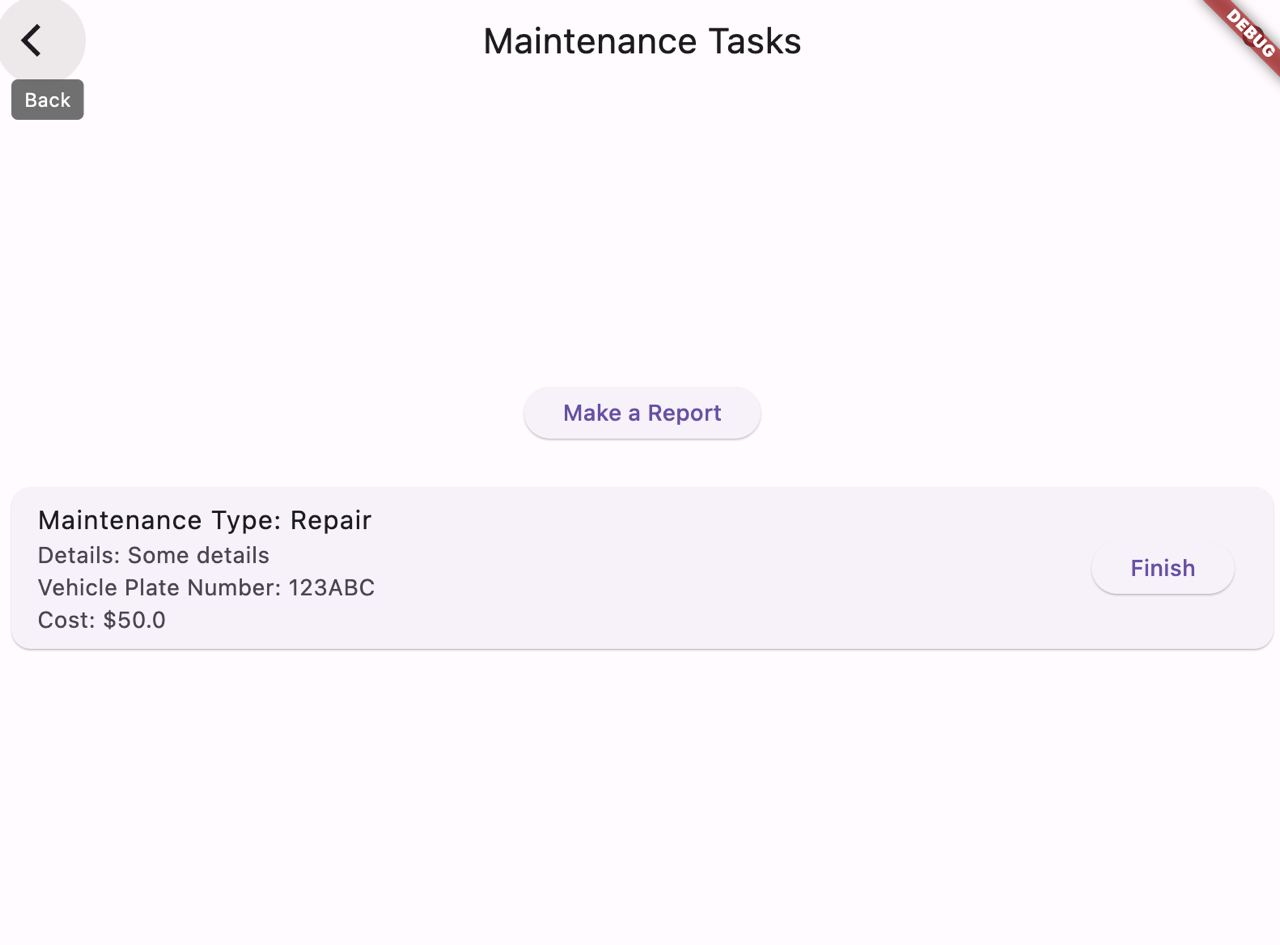
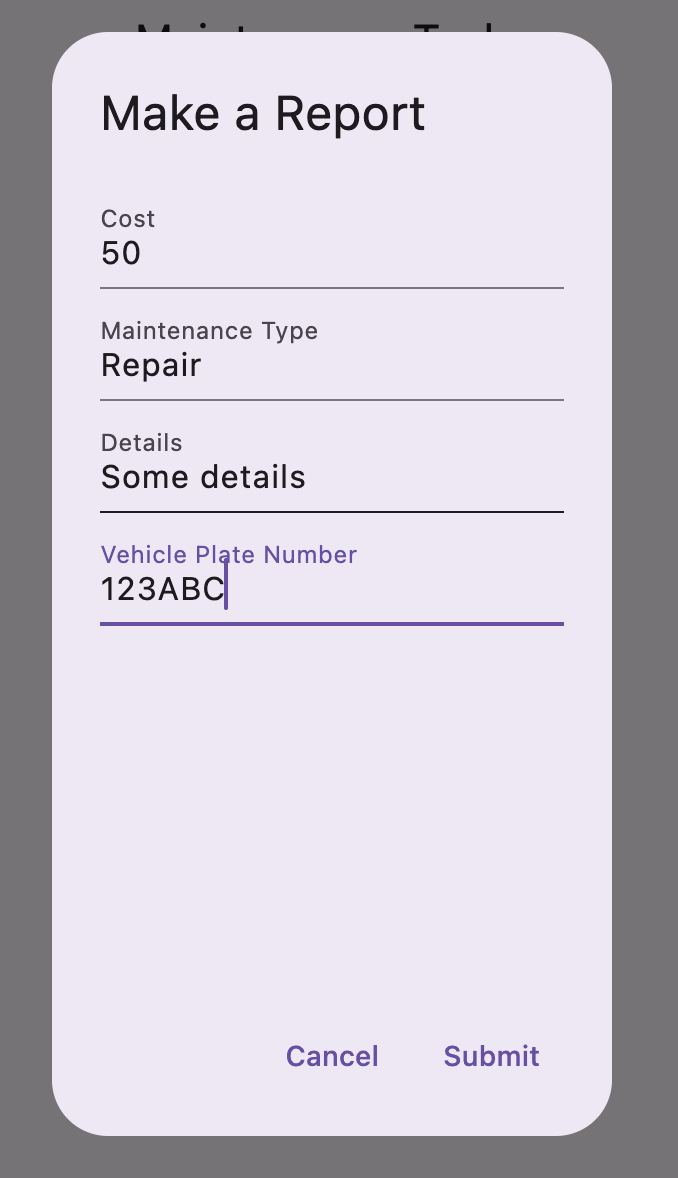


Above is the more detailed information about the tasks. It is used to track the start and end locations of the route that will be used later to project onto map and show for the drivers.

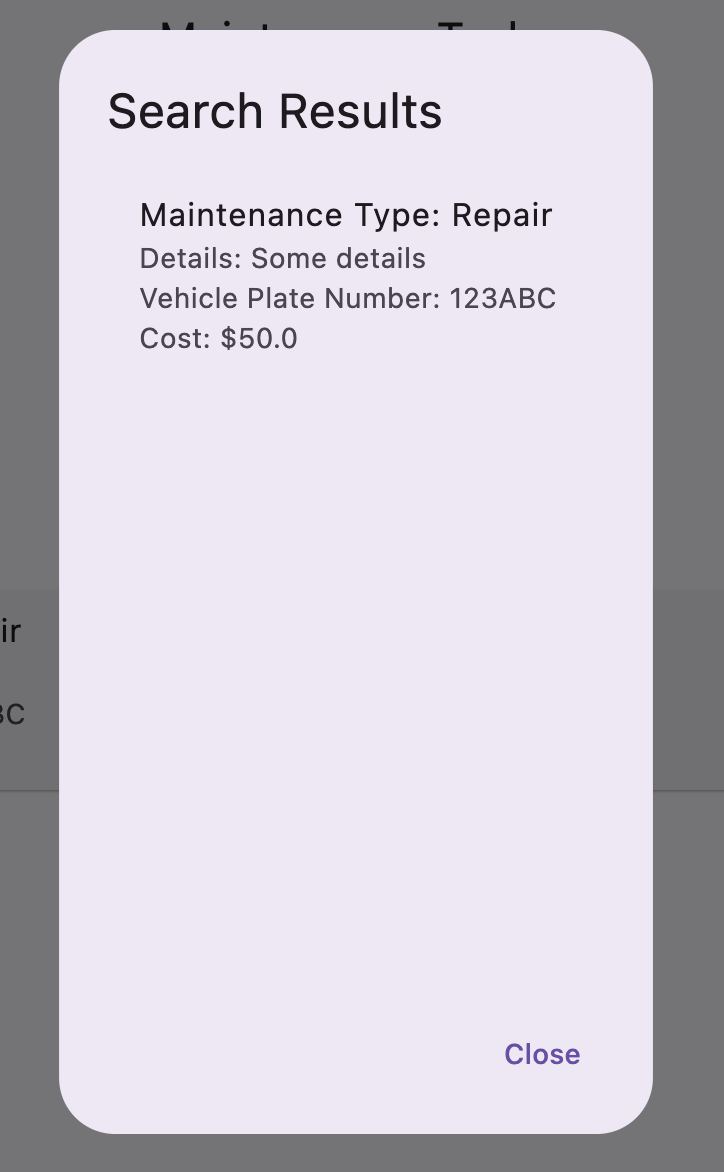
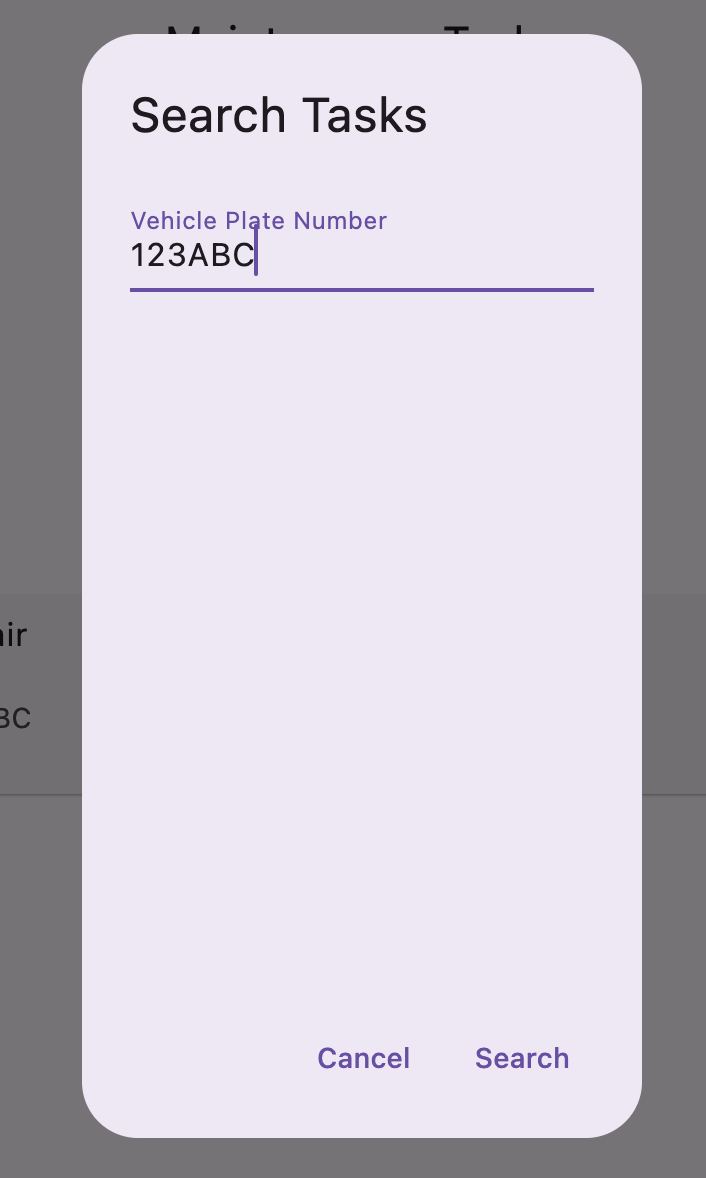
Maintenance User Reporting



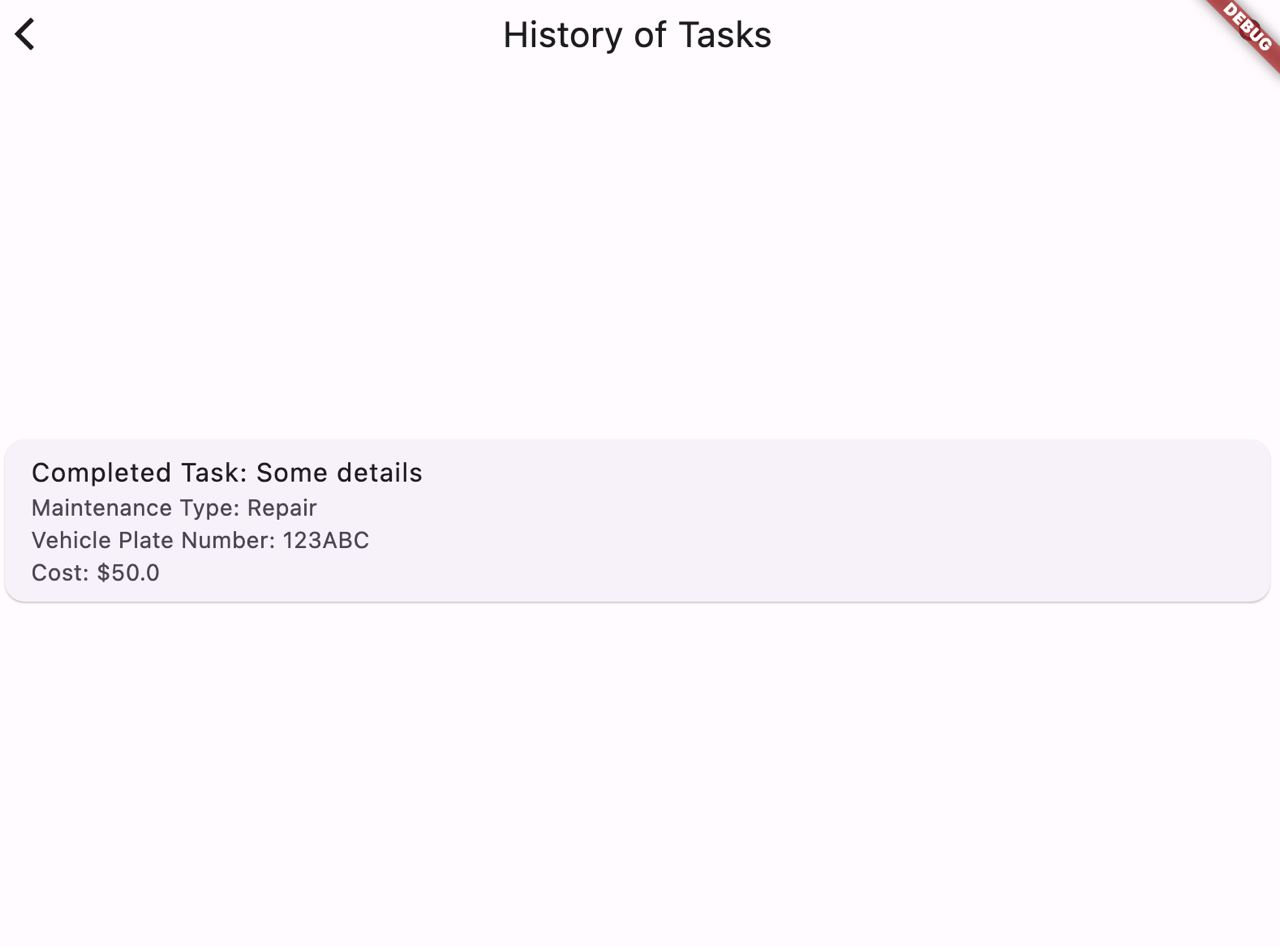
After the login you are redirected to the Maintenance Person interface. Where you can see frther instructions for the Maintenance People to do works on cars.



They have functionality to view Maintenance Tasks and Create a Report.



Additionally, you can search for the tasks if there are a lot of tasks to choose from.



Overall history of tasks is stored in the database and displayed to the User if he needs it.