



# Group 3

Design Report

CS 353 - 02

# NovaCars

## Authors:

Ani Kristo

Burcu Çanakcı

Onatkut Dağtekin

Yasemin Doğancı

## Contents

List of Figures .....	5
1. Revised E/R Model .....	6
2. Relational Schemas, Functional Dependencies and Normalization of Tables .....	7
2.1. User & Customer .....	7
2.1.1. User .....	7
2.1.2. Customer .....	8
2.1.3. Auto .....	9
2.2. Department & Employees .....	10
2.2.1. Department .....	10
2.2.2. Employee .....	11
2.2.3. Clerk .....	12
2.2.4. SalesManager .....	12
2.2.5. Technician .....	13
2.2.6. Manager .....	13
2.3. Suppliers & Spare Parts .....	14
2.3.1. Supplier .....	14
2.3.2. SparePart .....	14
2.3.3. Supply .....	15
2.4. Employee Operations & Transactions .....	16
2.4.1. Operation .....	16
2.4.2. Transaction .....	16
2.4.3. TechnicianOperation .....	17
2.4.4. SparePartOrder .....	17
2.4.5. CustomerOperation .....	18
3. Functional Components .....	20
3.1. Use Cases / Scenarios .....	20
3.1.1. Customer .....	20
3.1.2. Clerk .....	21
3.1.3. Sales Manager .....	22
3.1.4. Technician .....	23
3.1.5. Manager .....	24
3.2. Algorithms .....	26

3.2.1.	Quantity Related Algorithms .....	26
3.2.2.	Customer Related Algorithms:.....	26
3.2.3.	Miscellaneous Algorithms.....	26
3.3.	Data Structures .....	26
4.	UI Design and SQL Statements.....	27
4.1.	Public Website .....	27
4.2.	Sign-in.....	28
4.3.	Create Account.....	29
4.4.	Dashboard: Overview.....	30
4.5.	Manager Dashboard: Department info .....	31
4.6.	Manager Dashboard: Transactions .....	33
4.7.	Customer Profile .....	34
4.8.	Clerk Dashboard: Customer Transactions.....	36
4.9.	Clerk Dashboard: New Customer Transaction .....	38
4.10.	Clerk Dashboard: Supplier Transactions .....	40
4.11.	Clerk Dashboard: Detailed Customer Transaction Information.....	41
4.12.	Clerk Dashboard: Detailed Supplier Transaction Information .....	42
4.13.	Sales Manager Dashboard: Supplier Transactions.....	43
4.14.	Sales Manager Dashboard: New Supplier Transaction .....	45
4.15.	Sales Manager Dashboard: Supplier Information .....	47
5.	Advanced Database Components.....	49
5.1.	Views .....	49
5.1.1	Manager's Service Department View .....	49
5.1.2	Clerk's Transaction View .....	50
5.1.3	Sales Manager's Transaction View.....	51
5.1.4	Customer's Operation View .....	52
5.1.5	Technician's Operation View .....	52
5.2.	Stored procedures .....	52
5.3.	Reports.....	53
5.3.1	Total Money Spent by Customers.....	53
5.3.2	Number of Staff in Each Department .....	53
5.3.3	Dashboard: Overview.....	53
5.3.3	Manager Dashboard: Department Info .....	53

5.3.4 Manager Dashboard: Transactions .....	54
5.3.5 Sales Manager Dashboard .....	55
5.4.5 Clerk Manager Dashboard .....	56
5.4.6 Customer Dashboard .....	57
5.4. Triggers.....	58
5.5. Constraints .....	58
6. Implementation Plan .....	58

## List of Figures

Figure 1 - E/R Diagram for NovaCars system .....	6
Figure 2 - Use Case for Customer.....	20
Figure 3- Use Case for Clerk.....	21
Figure 4- Use Case for Sales Manager.....	22
Figure 5- Use Case for Technician .....	23
Figure 6- Use Case for Manager.....	25
Figure 7 - Website as visible by any unauthenticated user .....	27
Figure 8 - Sign in page .....	28
Figure 9 - Registering for a new customer account .....	29
Figure 10 - Manager Dashboard .....	30
Figure 11 - Department information as displayed to the manager .....	31
Figure 12 - List of transactions done with the operations provided by the department .....	33
Figure 13 - Customer profile information including credentials, owned vehicles and history of transactions.....	34
Figure 14 - Customer Transactions displayed to the clerk.....	36
Figure 15 - New Transaction screen as displayed to a clerk .....	38
Figure 16 - Supplier transactions as displayed to the clerk .....	40
Figure 17 - Modal for displaying detailed information about a customer transaction .....	41
Figure 18 - Detailed information screen for a customer transaction as displayed to a clerk.....	42
Figure 19 - List of supplier transactions as displayed to a sales manager .....	43
Figure 20 - New supplier transaction screen as displayed to a sales manager .....	45
Figure 21 - List of suppliers and spare parts in stock as displayed to a sales manager .....	47

## 1. Revised E/R Model

Upon receiving feedback on the E/R model presented in the Proposal Report, we made the following changes to the diagram. Most importantly, we added Clerk, Sales Manager, Technician and Manager entities for which detailed information can be found in Section 3.1. Note that our system will support two different kinds of Transaction: Customer and Supplier transactions, operated by a Clerk and by a Sales Manager respectively. This is represented with the two WITH relationships in the figure below.

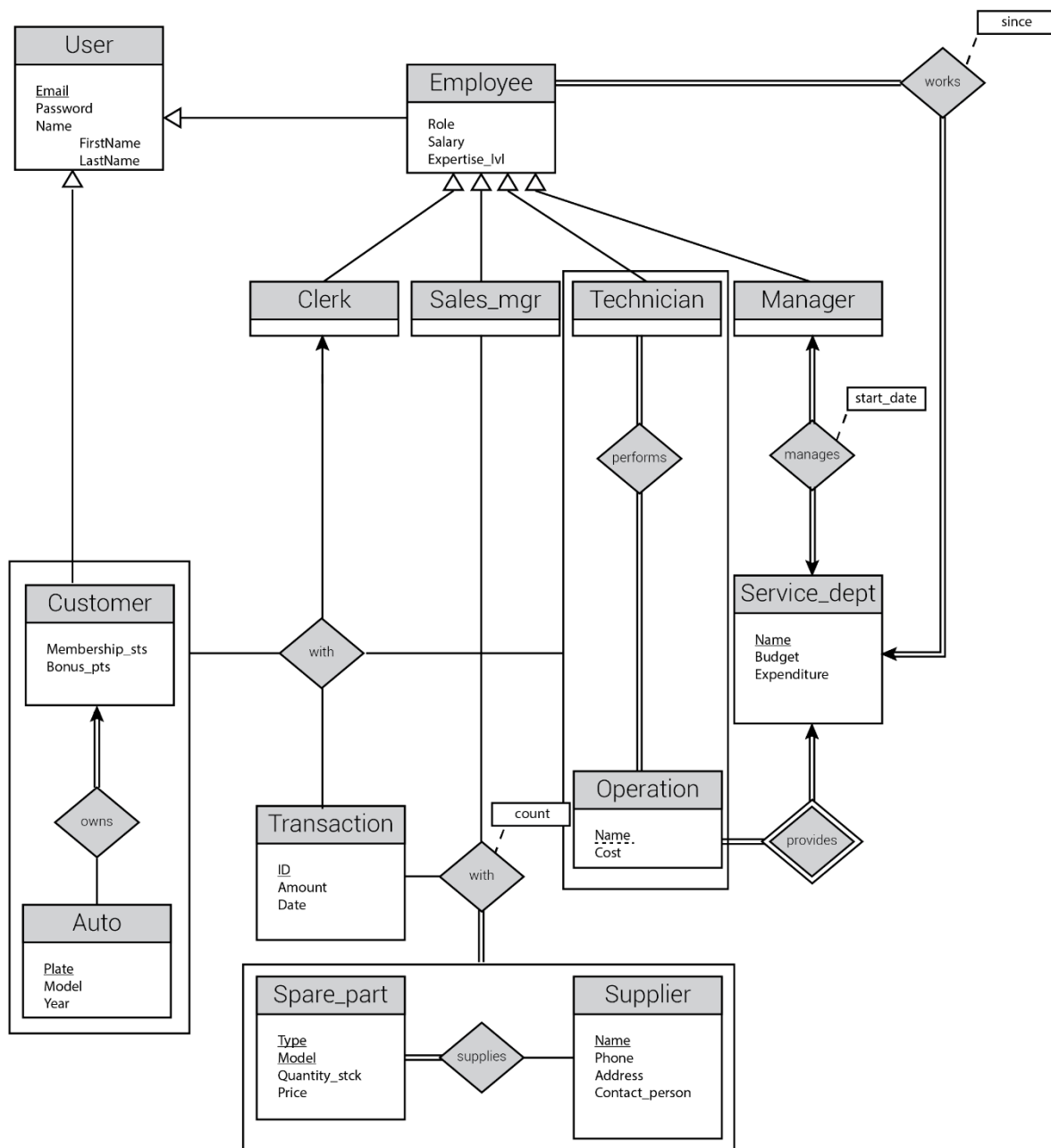


Figure 1 - E/R Diagram for NovaCars system

## 2. Relational Schemas, Functional Dependencies and Normalization of Tables

### 2.1. User & Customer

#### 2.1.1. User

**Relational Model:** User (email, password, first\_name, last\_name)

**Functional Dependencies:** email → password first\_name last\_name

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE User
(email          VARCHAR(50) NOT NULL,
 password      VARCHAR(20) NOT NULL,
 first_name    VARCHAR(20) NOT NULL,
 last_name     VARCHAR(20) NOT NULL,
 PRIMARY KEY (email)
) ENGINE = INNODB;
```

### 2.1.2. Customer

**Relational Model:** Customer (email, membership\_sts, bonus\_pts)

**Functional Dependencies:** email -> membership\_sts bonus\_pts

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

#### **Create relation in SQL:**

```
CREATE TABLE Customer
(email          VARCHAR(50) NOT NULL,
 membership_sts VARCHAR(20) NOT NULL,
 bonus_pts      NUMERIC(5,0),
 PRIMARY KEY (email),
 FOREIGN KEY (email) REFERENCES User(email)
           ON DELETE CASCADE
           ON UPDATE CASCADE
) ENGINE = INNODB;
```



### 2.1.3. Auto

**Relational Model:** Auto (plate, model, year, customer\_email)

**Functional Dependencies:** plate -> model year customer\_email

**Candidate Keys:** {(plate)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Auto
(plate          VARCHAR(10) NOT NULL,
 model          VARCHAR(15) ,
 year           YEAR(4) ,
 customer_email VARCHAR(50) NOT NULL,
 PRIMARY KEY (plate)
 FOREIGN KEY (customer_email) REFERENCES Customer(email)
           ON DELETE CASCADE
           ON UPDATE CASCADE
) ENGINE = INNODB;
```

## 2.2. Department & Employees

### 2.2.1. Department

**Relational Model:** Department (name, budget, expenditure)

**Functional Dependencies:** name  $\rightarrow$  budget expenditure

**Candidate Keys:** {(name)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Department
(name          VARCHAR(30) NOT NULL,
 budget       NUMERIC(12,2),
 expenditure   NUMERIC(12,2),
 PRIMARY KEY (name)
) ENGINE = INNODB;
```

### 2.2.2. Employee

**Relational Model:** Employee (email, salary, role, expertise\_lvl, dept\_name, since)

**Functional Dependencies:** email -> salary role expertise\_lvl dept\_name since

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

#### Create relation in SQL:

```
CREATE TABLE Employee
(email          VARCHAR(50) NOT NULL,
 salary        NUMERIC(5,2) NOT NULL,
 role          VARCHAR(20),
 expertise_lvl  NUMERIC(2,0),
 dept_name     VARCHAR(30) NOT NULL,
 since         YEAR(4) NOT NULL,
PRIMARY KEY(email)
FOREIGN KEY(dept_name) REFERENCES Department(name)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.2.3. Clerk

**Relational Model:** Clerk (email)

**Functional Dependencies:** No dependencies

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Clerk
(email          VARCHAR(50) NOT NULL,
 PRIMARY KEY(email)
 FOREIGN KEY(email) REFERENCES Employee(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.2.4. SalesManager

**Relational Model:** SalesManager (email)

**Functional Dependencies:** No dependencies

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE SalesManager
(email          VARCHAR(50) NOT NULL,
 PRIMARY KEY(email)
 FOREIGN KEY(email) REFERENCES Employee(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.2.5. Technician

**Relational Model:** Technician (email)

**Functional Dependencies:** No dependencies

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Technician
(email          VARCHAR(50) NOT NULL,
 PRIMARY KEY(email)
 FOREIGN KEY(email) REFERENCES Employee(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.2.6. Manager

**Relational Model:** Manager (email, start\_date)

**Functional Dependencies:** email-> start\_date

**Candidate Keys:** {(email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Manager
(email          VARCHAR(50) NOT NULL,
 start_date     YEAR(4),
 PRIMARY KEY(email)
 FOREIGN KEY(email) REFERENCES Employee(email)
      ON DELETE CASCADE
      ON UPDATE CASCADE
) ENGINE = INNODB;
```

## 2.3. Suppliers & Spare Parts

### 2.3.1. Supplier

**Relational Model:** Supplier (name, phone, address, contact\_name)

**Functional Dependencies:** name → phone address contact\_name

**Candidate Keys:** {(name)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Supplier
(name          VARCHAR(30) NOT NULL,
phone         VARCHAR(12) ,
address       VARCHAR(60) ,
contact_name  VARCHAR(20) NOT NULL,
PRIMARY KEY (name)
) ENGINE = INNODB;
```

### 2.3.2. SparePart

**Relational Model:** SparePart (type, model, stock\_quantity, price)

**Functional Dependencies:** type, model → stock\_quantity, price

**Candidate Keys:** {(type, model)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE SparePart
(type          VARCHAR(15) NOT NULL,
model         VARCHAR(10) NOT NULL,
stock_quantity SMALLINT
price         NUMERIC(10,2) NOT NULL
PRIMARY KEY (type, model)
) ENGINE = INNODB;
```

### 2.3.3. Supply

**Relational Model:** Supply (type, model, supplier\_name)

**Functional Dependencies:** No dependencies

**Candidate Keys:** {(type, model, supplier\_name)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Supply
(type          VARCHAR(15) NOT NULL,
model         VARCHAR(10) NOT NULL,
supplier_name  VARCHAR(30) NOT NULL,
PRIMARY KEY(type, model, supplier_name),
FOREIGN KEY(type, model) REFERENCES SparePart(type, model)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(supplier_name) REFERENCES Supplier(name)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE = INNODB;
```

## 2.4. Employee Operations & Transactions

### 2.4.1. Operation

**Relational Model:** Operation (dept\_name, op\_name, cost)

**Functional Dependencies:** dept\_name, op\_name → cost

**Candidate Keys:** {(name)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Operation
(dept_name VARCHAR(30) NOT NULL,
op_name   VARCHAR(30) NOT NULL,
cost      NUMERIC(8,2),
PRIMARY KEY(op_name, dept_name),
FOREIGN KEY(dept_name) REFERENCES Department(name)
ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.4.2. Transaction

**Relational Model:** Transaction (id, amount, date)

**Functional Dependencies:** id → amount date

**Candidate Keys:** {(id)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE Transaction
(id          MEDIUMINT NOT NULL AUTO_INCREMENT,
amount      NUMERIC(7,2),
date        DATE,
PRIMARY KEY (id)
) ENGINE = INNODB;
```



### 2.4.3. TechnicianOperation

**Relational Model:** TechnicianOperation (dept\_name, op\_name, tech\_email)

**Functional Dependencies:** No dependencies

**Candidate Keys:** {(dept\_name, op\_name, tech\_email)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE TechnicianOperation
(dept_name VARCHAR(30) NOT NULL,
 op_name   VARCHAR(30) NOT NULL,
 tech_email VARCHAR(50) NOT NULL,
 PRIMARY KEY (dept_name, op_name, tech_email),
 FOREIGN KEY(dept_name, op_name) REFERENCES Operation(dept_name,
 op_name)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
 FOREIGN KEY(tech_email) REFERENCES Technician(email)
         ON DELETE CASCADE
         ON UPDATE CASCADE
) ENGINE = INNODB;
```

### 2.4.4. SparePartOrder

**Relational Model:** SparePartOrder (transaction\_id, sales\_mgr\_email, part\_type, part\_model, supplier\_name, count)

**Functional Dependencies:** transaction\_id -> sales\_mgr\_email part\_type part\_model  
supplier\_name count

**Candidate Keys:** {(transaction\_id)}

**Normal Form:** BCNF

**Create relation in SQL:**

```
CREATE TABLE SparePartOrder
```

```

(transaction_id      MEDIUMINT NOT NULL,
 sales_mgr_email    VARCHAR(50) NOT NULL,
 part_type          VARCHAR(15) NOT NULL,
 part_model         VARCHAR(10),
 supplier_name      VARCHAR(30),
 count              SMALLINT NOT NULL,
 PRIMARY KEY(transaction_id),
 FOREIGN KEY(transaction_id) REFERENCES Transaction(id)
     ON DELETE CASCADE
     ON UPDATE CASCADE,
 FOREIGN KEY(sales_mgr_email) REFERENCES SalesManager(email)
     ON DELETE CASCADE
     ON UPDATE CASCADE,
 FOREIGN KEY(supplier_name, part_type, part_model) REFERENCES
 Supply(supplier_name, type, model)
     ON DELETE CASCADE
     ON UPDATE CASCADE
) ENGINE = INNODB;

```

#### 2.4.5. CustomerOperation

**Relational Model:** CustomerOperation (transaction\_id, dept\_name, op\_name, tech\_email, clerk\_email, customer\_email, auto\_plate)

**Functional Dependencies:** transaction\_id → dept\_name op\_name tech\_email clerk\_email  
customer\_email auto\_plate

**Candidate Keys:** {(transaction\_id)}

**Normal Form:** BCNF

**Create relation in SQL:**

```

CREATE TABLE CustomerOperation
(transaction_id MEDIUMINT NOT NULL,
 dept_name     VARCHAR(30) NOT NULL,

```

```

op_name          VARCHAR(30) NOT NULL,
tech_email       VARCHAR(10) NOT NULL,
clerk_email      VARCHAR(50) NOT NULL,
customer_email   VARCHAR(50) NOT NULL,
auto_plate       VARCHAR(10) NOT NULL,
PRIMARY KEY(transaction_id),
FOREIGN KEY (transaction_id) REFERENCES Transaction(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(dept_name, op_name, tech_email) REFERENCES
TechnicianOperation(dept_name, op_name, tech_email)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(clerk_email) REFERENCES Clerk(email)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(auto_plate, customer_email) REFERENCES
Auto(customer_email, plate)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE = INNODB;

```

### 3. Functional Components

In NovaCars database management system, there are 5 types of users. These are: Customers, Clerks, Sales Managers, Technicians and Managers. All users must log into the system before using it or create an account. Depending on the user type, each will have different features.

#### 3.1. Use Cases / Scenarios

##### 3.1.1. Customer

- Customers must log in to the system with e-mails and passwords.
- Customers can access their profiles after logging in which shows the customer information such as owned vehicles, history of operations, their membership types and the amount of points collected.
- Customers can change their passwords.
- Customers can add new vehicles to their accounts.
- Customers can edit a vehicle's information.
- If a customer does not have an account he/she can go to the create account page from the login page and fill out information such as first name, last name, e-mail and password.

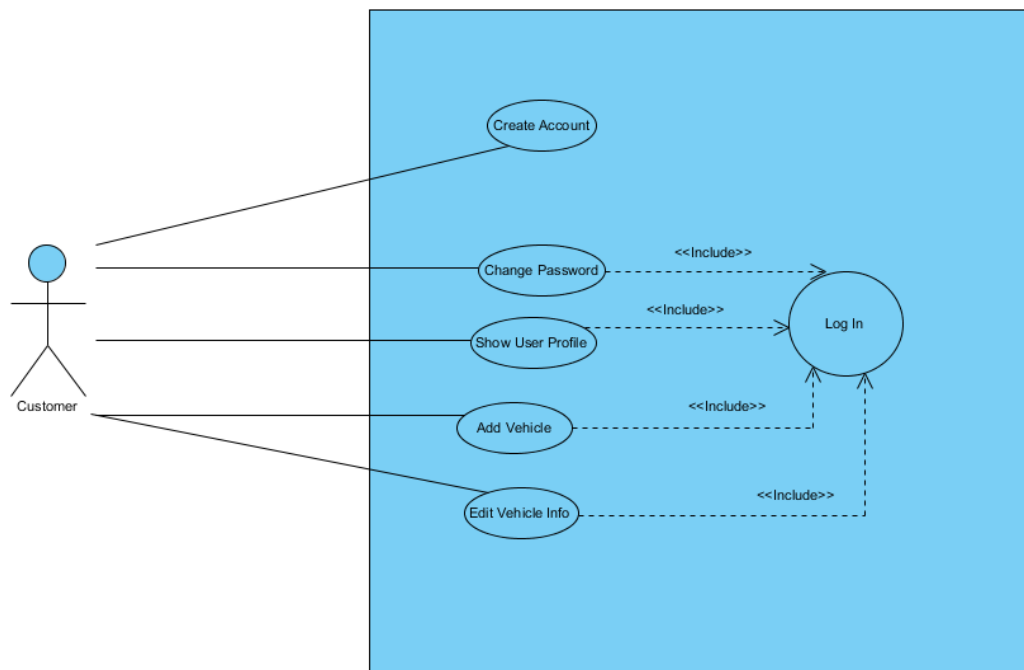


Figure 2 - Use Case for Customer

### 3.1.2. Clerk

- Clerks must log in to the system with e-mails and passwords.
- Clerks can access their profiles after logging in which shows the clerk information such as e-mail, salary and expertise level.
- Clerks can change their passwords.
- Clerks can see their number of transactions and total revenue for a customer from the customer transactions page.
- Clerks can display information transactions with customers which also lists the type of auto, the type of operation, price and the date of transaction.
- Clerks can also see information about the suppliers such as the number of transactions and the total cost and the table for history of supplier's transactions together with spare part type, spare part model, price and date.
- Clerks can add new customer transactions by filling in necessary information such as customer e-mail, auto plate revenue and filer for the transaction.

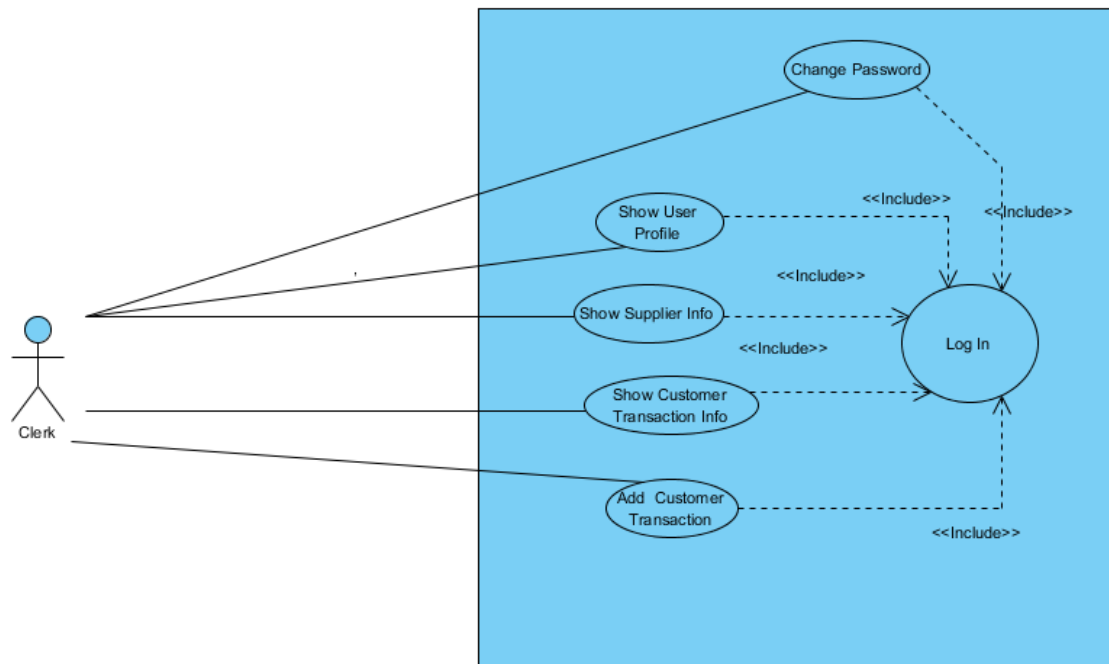


Figure 3- Use Case for Clerk

### 3.1.3. Sales Manager

- Sales managers must log in to the system with e-mails and passwords.
- Sales managers can access their profiles after logging in which shows the sales manager information such as e-mail, salary and expertise level.
- Sales managers can change their passwords.
- Sales managers can see the transactions between the service and the suppliers with information; supplier name, spare part type, spare part model, quantity, price and date of transaction.
- Sales managers can add new supplier transactions between suppliers by entering necessary information such as supplier name, spare part type, spare part model, count and price.
- Sales managers can also show the supplier information such as the name of the supplier name, phone, address together with the parts they supply.

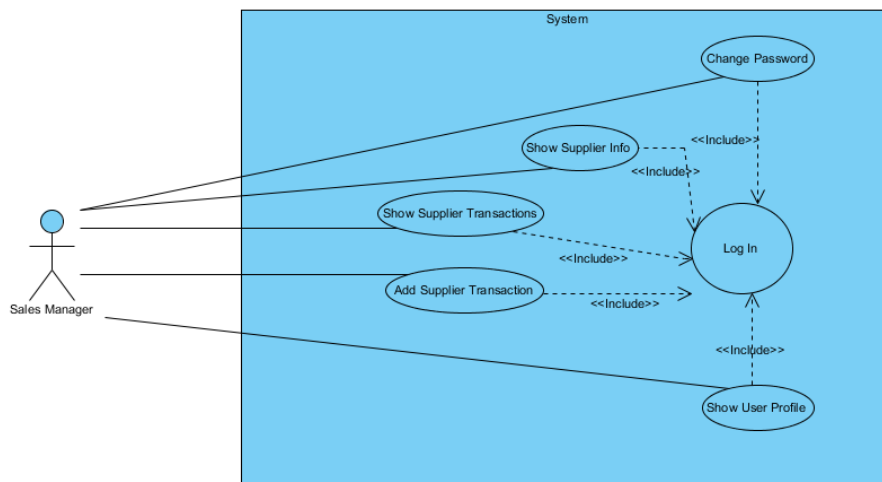


Figure 4- Use Case for Sales Manager

#### 3.1.4. Technician

- Technicians must log in to the system with e-mails and passwords.
- Technicians can access their profiles after logging in which shows the sales manager information such as e-mail, salary and expertise level.
- Technicians can change their passwords.
- Technicians can see the operations they can perform and their history of operations.
- Technicians can add new operations to their histories by filling in required information such as the operation name, the cost of the operation and the auto which the operation was done on.

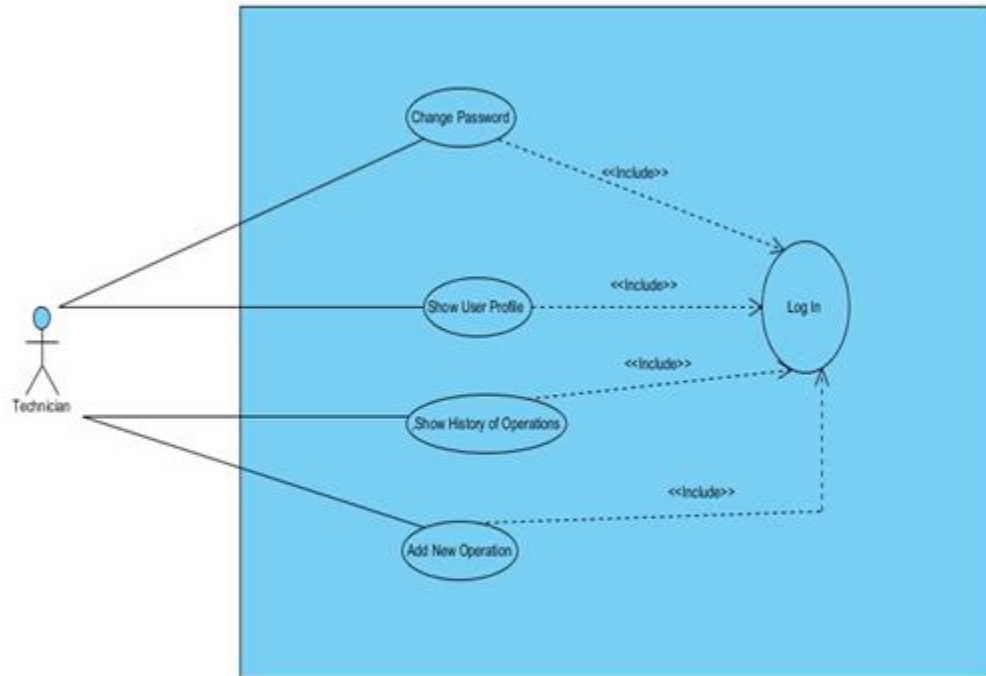


Figure 5- Use Case for Technician

### 3.1.5. Manager

- Managers must log in to the system with e-mails and passwords.
- Managers can change their passwords.
- Managers can access their profiles after logging in which shows the department he/she is managing since which date and personal info such as e-mail, salary and expertise level and their password which can be changed by clicking on change button.
- Managers can see information about their departments and the operations they provide. Information about the departments include allocated budget, total expenditure, total revenue and number of employees.
- Managers can see a list of the employees they are in charge of together with their names, surnames, e-mails, salaries, expertise levels.
- Managers can edit information about their employees.
- Managers can add or remove new employees to and from their departments.
- Managers can show information about customer transactions limited to their department's operations with information such as number of transaction, total revenue and a table for history of customer transactions which includes information about the customer, auto, operation, clerk, price and date.
- Manager can see the transactions of his/her department with information such as customer name, auto type, operation type, clerk name, price and date.



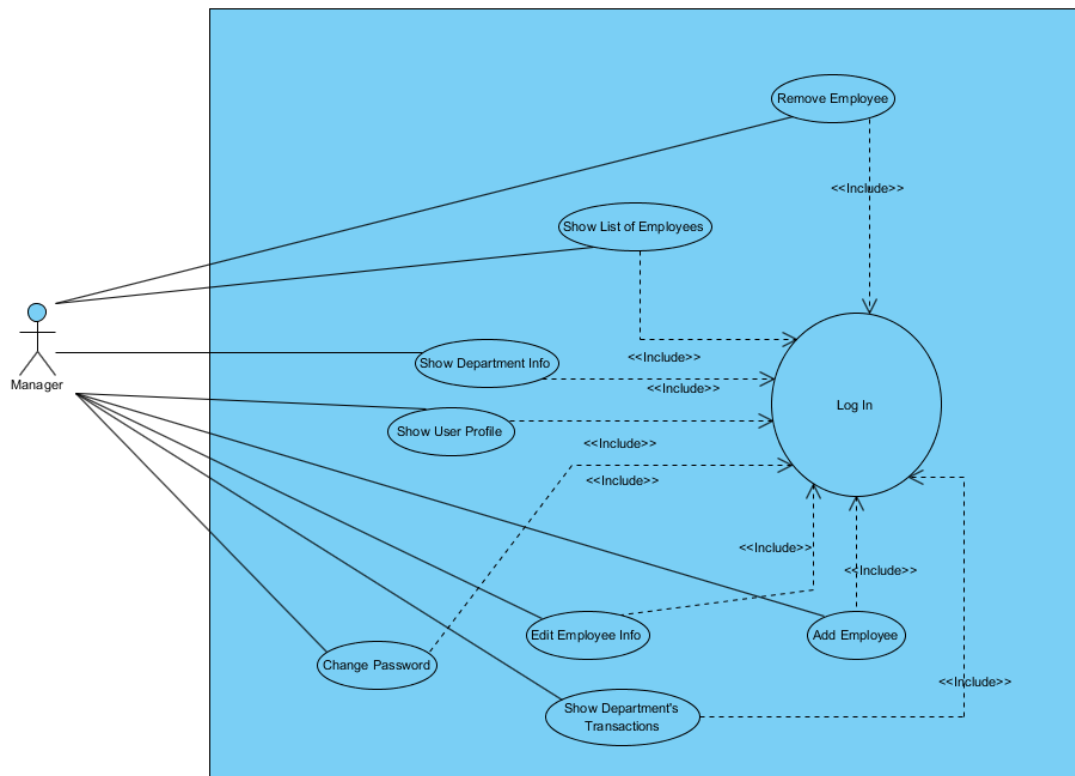


Figure 6- Use Case for Manager

## 3.2. Algorithms

### 3.2.1. Quantity Related Algorithms

In this database spare part have limited quantity. To make transaction doable spare parts must be available. The quantity of spare part is maintained by the quantity\_stck attribute in Spare\_part table.

At creation the value of the stock would be non-negative and it would decrease or increase depending on the type of transaction between the supplier and the sales manager or between customer and clerk. If a transaction is made between the supplier and sales manager, the attribute quantity\_stck in Spare\_part table would increase. If a transaction is made between customer or clerk, the attribute quantity\_stck in Spare\_part table would decrease.

### 3.2.2. Customer Related Algorithms:

The system also tracks customer's information so that customers who use the service more than others would have special benefits. To provide this benefit in the Customer table two attributes are stored: Membership\_sts and Bonus\_pts. When a customer makes a transaction certain percentage of the cost will be converted to bonus points and added to Bonus\_pts attribute.

### 3.2.3. Miscellaneous Algorithms

To remove the logical errors in the system some procedures must be followed. Such as the start date of an employee and start date of a manager. The attribute since in Manager table and the attribute since in the Employee table cannot refer to a future date since an employee must start working on this service center in the past.

## 3.3. Data Structures

For the attributes we will use varchar, integer, date, year data types of MySQL with JDBC.

## 4. UI Design and SQL Statements

### 4.1. Public Website

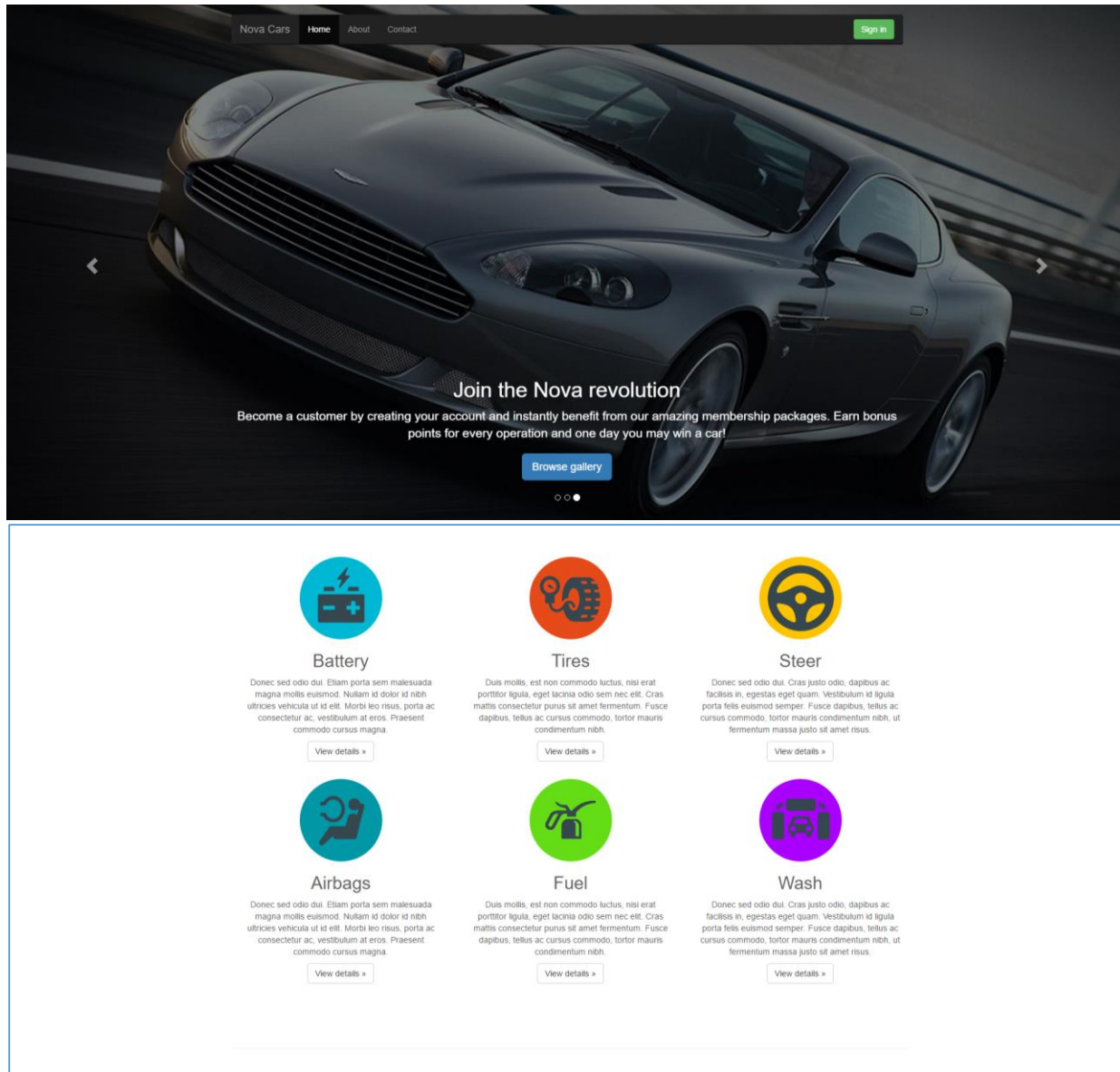


Figure 7 - Website as visible by any unauthenticated user

Figure 7 shows the UI concept which appears to any user without the need for being an employee or customer. It shows the operations we provide and teases the customers to sign up. The content of this page is static.

## 4.2. Sign-in

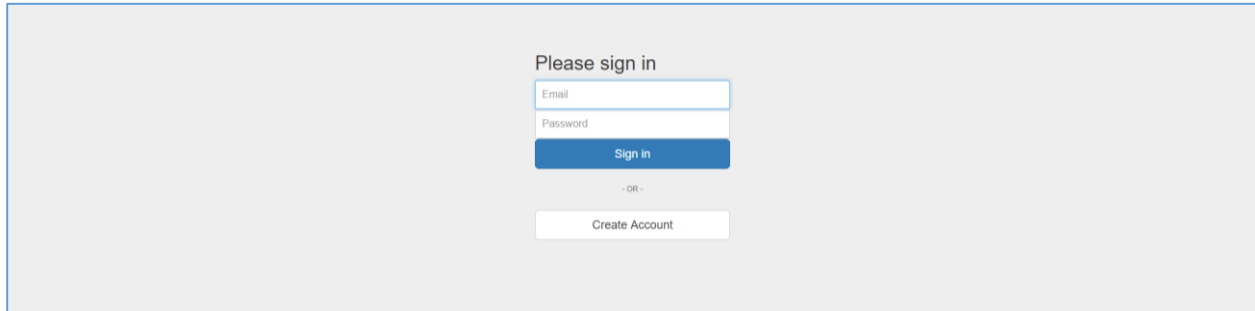
The image shows a sign-in page with a light gray background. At the top, it says "Please sign in". Below this are two input fields: "Email" and "Password". A blue "Sign in" button is positioned below the password field. Underneath the button is a separator that says "- OR -". At the bottom, there is a white "Create Account" button.

Figure 8 - Sign in page

**Process:** Upon clicking the Sign in button as displayed on Figure 78, the user is redirected to this page. The user can be an employee or customer and they sign in in the same place, then the system decides what contents to display. The 'Create Account' button is only meant to be used by future customers, as employees need to be assigned by a manager. The user can sign in by filling the fields E-mail and Password.

**Inputs:** @email, @password

### SQL Statements:

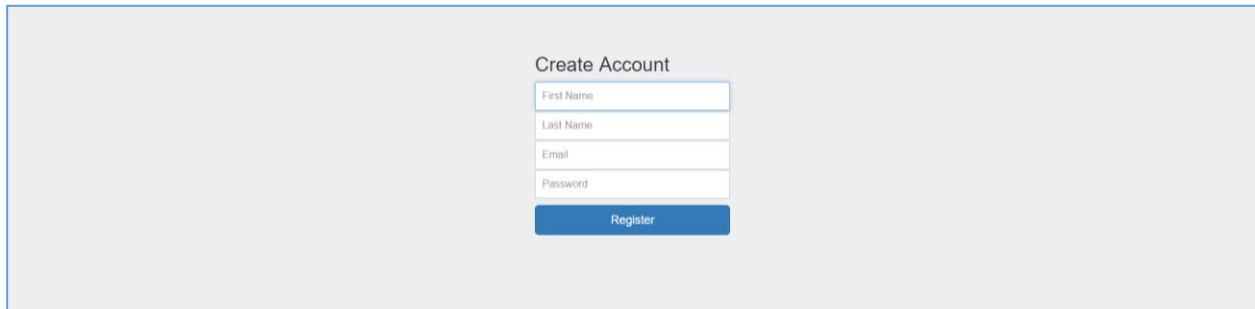
```
SELECT email
FROM User NATURAL JOIN Employee
WHERE email = @email AND password = @password;
```

```
SELECT email
FROM User NATURAL JOIN Customer
WHERE email = @email AND password = @password;
```

**Note:** The system will perform these two queries and decide according to the number of returned rows for the following:

Rows for query 1	Rows for query 2	Decision
0	0	<b>Not a user</b>
0	1	<b>Only a customer</b>
1	0	<b>Only an employee</b>
1	1	<b>An employee and a customer</b>
Otherwise		<b>Error!</b>

### 4.3. Create Account

A screenshot of a web form titled "Create Account". The form is centered on a light gray background. It contains four text input fields stacked vertically, labeled "First Name", "Last Name", "Email", and "Password". Below these fields is a blue button with the text "Register" in white.

*Figure 9 - Registering for a new customer account*

**Process:** Upon clicking Create Account button on Figure 89, this screen appears. One can only create a customer account via this page. In order to create an account, the user needs to fill the parts given in the figure: First Name, Last Name, E-mail, and Password.

**Inputs:** @first\_name @last\_name, @email, @password

**SQL Statements:**

```
INSERT INTO User (email, password, first_name, last_name)
VALUES (@email, @password, @first_name, @last_name);
```

```
INSERT INTO Customer (email, membership_sts, bonus_pts)
VALUES (@email, 'Classic', 0);
```

## 4.4. Dashboard: Overview

The screenshot shows a web application interface for 'Nova Cars'. At the top, there is a dark header with the 'Nova Cars' logo on the left and a 'Log out' button on the right. Below the header is a sidebar on the left with a blue 'Overview' tab selected, and other tabs for 'Department Info', 'Transactions', and 'Customer Profile'. The main content area displays the 'Overview' for a manager. It shows the manager's 'Name Surname', their role as 'Manager of Department X', and the start date 'Since YYYY'. Below this, there is an 'Employee profile' section with a light blue header. This section contains four rows of information: 'Email' (name.surname@novacars.com), 'Password' (with a 'Change' button), 'Salary' (XXXX), and 'Expertise Level' (XX).

Figure 10 - Manager Dashboard

**Process:** In the example case for the Overview page, a manager is assumed to log in. Note that the forms displayed in this dashboard is not specific to the manager, any kind of employee will see a similar screen when they login, as the “Overview” tab. As soon as a manager is logged in, the above screen will appear. In this dashboard, the manager will be able to see an overview of their employee profile, information about the department they manage, a list of transaction for that particular department and, if the manager is also a customer, then a customer profile page will also be available. These information screens can be accessed by navigating on the sidebar on the left. In Figure 10 we have displayed the first section (Overview), where some quick information is obtained about the manager. In this section managers are able to see their department, the year which they began working in that service department, their e-mail address, a button for changing their password, a field for their salary, and their expertise level.

**Session Variables:** @email

**Inputs:** @changed\_password

### SQL Statements:

For Changing the Password:

When the manager clicks on the change button near the Password field, an input box will appear to take the input @changed\_password. Near this field, there will be a second input field for confirmation purposes. Once the input is confirmed and the save button is clicked the following query will be executed:

```
UPDATE User SET password = @changed_password WHERE email = @email;
```

## 4.5. Manager Dashboard: Department info

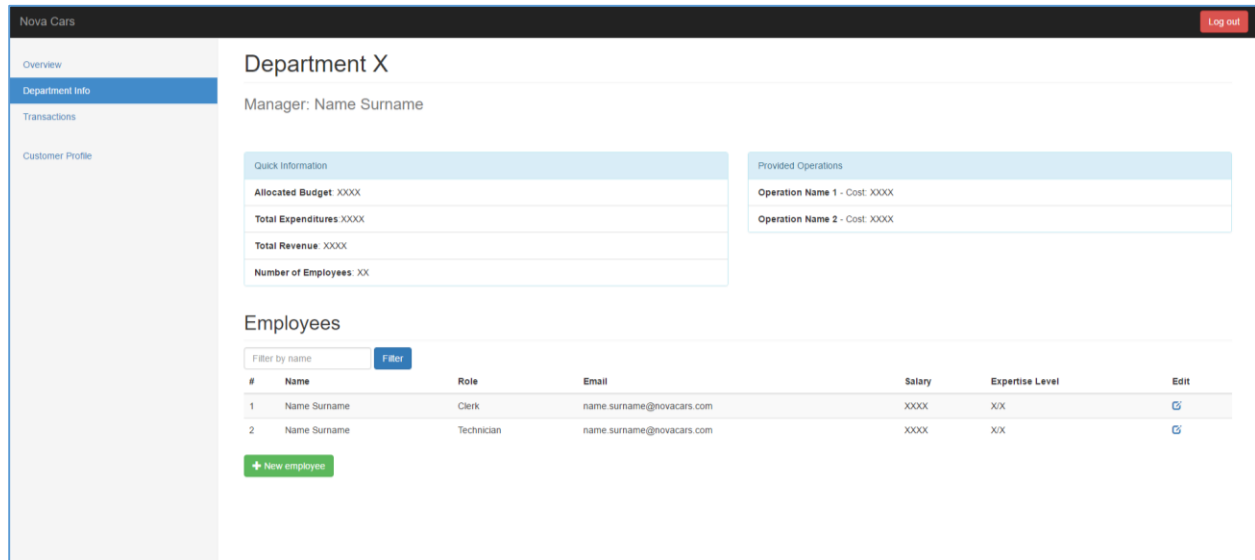


Figure 11 - Department information as displayed to the manager

**Process:** When a manager navigates on the sidebar and chooses the “Department Info” tab, this screen is displayed. In this screen, the department name and the manager of the department can be seen. Quick information about the department - allocated budget, total expenditures, total savings (which is budget - expenditures), and the number of employees working in that department- and the provided operations - with their names and costs- can be seen as a list. Also the manager can use the edit button next to each employee in Employees field and edit the salary, role and expertise level of that employee. The manager can also use a filter in order to find an employee by using their first name or last name with the department name. A manager can add a new employee by clicking to the “New employee” button.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

For future use the variable @mgr\_dept\_name will be assigned to  
`SELECT dept_name FROM Employee WHERE email = @email;`

**For Filtering the Employee Information by Name:**

```
SELECT first_name, last_name, role, salary, expertise_lvl  
FROM User NATURAL JOIN Employee
```

```
WHERE (first_name LIKE @filter OR last_name LIKE @filter) AND  
dept_name = @mgr_dept_name;
```

### **For Editing the Employee Information**

When the edit button is clicked near the row, a pop-up screen will appear for input fields for the salary, role and experience\_lvl fields of the employee. Once changes are made and the save button that will appear is clicked, the following query will execute

(related variable: @emp\_email)

(inputs: @changed\_role, @changed\_salary, @changed\_exp\_lvl)

```
UPDATE Employee
```

```
SET role = @changed_role, salary = @changed_salary, expertise_lvl =  
@changed_exp_lvl WHERE email = @emp_email;
```

Under the editing fields there will be a delete employee button. If this button is clicked the following query will execute.

```
DELETE FROM Employee WHERE email = @emp_email;
```

### **For Adding a New Employee to The Company**

When the new employee button is clicked, a pop-up window with input fields for the following fields for the Employee entity will appear: email, salary, role, expertise\_lvl, since. When the Save button that will appear is clicked the following query will execute.

(inputs = @emp\_email, @salary, @role, @level, @start\_date)

```
INSERT INTO Employee (email, salary, role, expertise_lvl, dept_name,  
since)  
VALUES (@emp_email, @salary, @role, @level, @mgr_dept_name,  
@start_date);
```



## 4.6. Manager Dashboard: Transactions

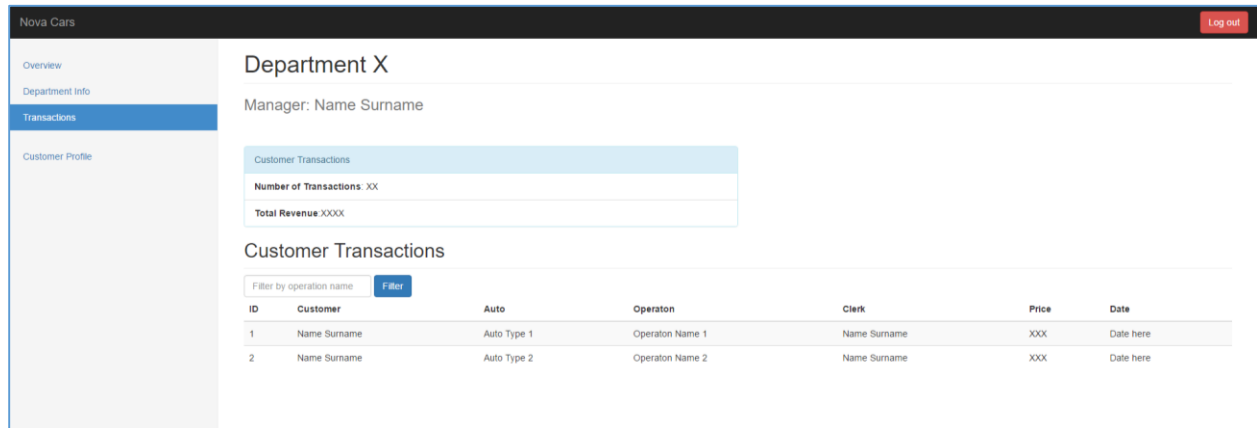


Figure 12 - List of transactions done with the operations provided by the department

**Process:** When a manager navigates on the sidebar and chooses the “Transactions” tab, this screen is displayed. In this screen, the department name and the manager of the department can be seen. Under the “Customer Transactions” field, the number of transactions and the total revenue can be displayed. The transactions can be filtered with a given operation name by the user. Also, it is possible to see the customer names, the type of the auto they have, the list of operations, the name, the price and date of the transaction.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

**For Filtering the Transactions by Operation Name:**

```
SELECT transaction_id, op_name, model, first_name, last_name, amount,
date
FROM (((CustomerOperation JOIN Employee E on E.email = clerk_email)
JOIN Customer C on customer_email = C.email)
JOIN Auto on plate = auto_plate)
JOIN User U on U.email = clerk_email)
JOIN Transaction T on T.id = transaction_id
WHERE op_name = @filter;
```

## 4.7. Customer Profile

The screenshot displays the 'Customer Profile' page in the Nova Cars system. The page is divided into three main sections: Customer profile, Owned vehicles, and History.

**Customer profile:** This section shows the customer's name and surname, their status as an 'Elite+ Customer', and the number of points collected (XXXX). It also displays the email address (name.surname@email.com) and a password field with a 'Change' button.

**Owned vehicles:** This section lists the vehicles owned by the customer. It includes three entries, each with a checkbox, the auto model, and the plate number (YYYY). A green '+ Add' button is located below the list.

**History:** This section shows a table of transactions. The table has columns for ID, Auto, Operation, Price, Date, and Points Earned.

ID	Auto	Operation	Price	Date	Points Earned
1	Auto Type 1	Operation Name 1	XXX	Date here	XX
2	Auto Type 2	Operation Name 2	XXX	Date here	XX

Figure 13 - Customer profile information including credentials, owned vehicles and history of transactions

**Process:** With the “Customer Profile” tab on the sidebar, it is possible to display a customer’s profile. In this screen the name and surname of the customer is shown at the top. Below that the type and bonus points of the customer can be shown. The e-mail address can be displayed and with the button under this field, the customer is able to change his/her password. The vehicles owned by the customer can also be seen as a list and it is possible to add a new vehicle to that list, as well as editing the automobile information with the edit button next to each vehicle. The history of transactions is also shown on this screen. This field includes: Auto type, Operations, Price, Date and the points earned from each transaction.

**Session Variables:** @email

**Inputs:** @changed\_password

**SQL Statements:**

**For Changing the Password:**

When the customer clicks on the change button near the Password field, an input box will appear to take the input @changed\_password. Near this field, there will be a second input field for confirmation purposes. Once the input is confirmed and the save button is clicked the following query will be executed:

```
UPDATE User SET password = @changed_password WHERE email = @email;
```

### **For Editing the Automobile Information**

When the edit button is clicked near the automobile information, a pop-up screen will appear for input fields for the year and model fields of the automobile (in case information was misentered before). Once changes are made and the save button that will appear is clicked, the following query will execute

(related variable: @auto\_plate)

(inputs: @changed\_year, @changed\_model)

```
UPDATE Auto
```

```
SET year = @changed_year, model = @changed_model
```

```
WHERE plate = @auto_plate;
```

Under the editing fields there will be a delete auto button. If this button is clicked the following query will execute.

```
DELETE FROM Auto WHERE plate = @auto_plate;
```

### **For Adding a New Automobile**

When the add button is clicked below the autos list, a pop-up window with input fields for the following fields for the Auto entity will appear: plate, model, year. When the Save button that will appear is clicked the following query will execute.

(inputs = @plate, @model, @year)

```
INSERT INTO Auto (plate, model, year, customer_email)
```

```
VALUES (@plate, @model, @year, @email);
```

## 4.8. Clerk Dashboard: Customer Transactions

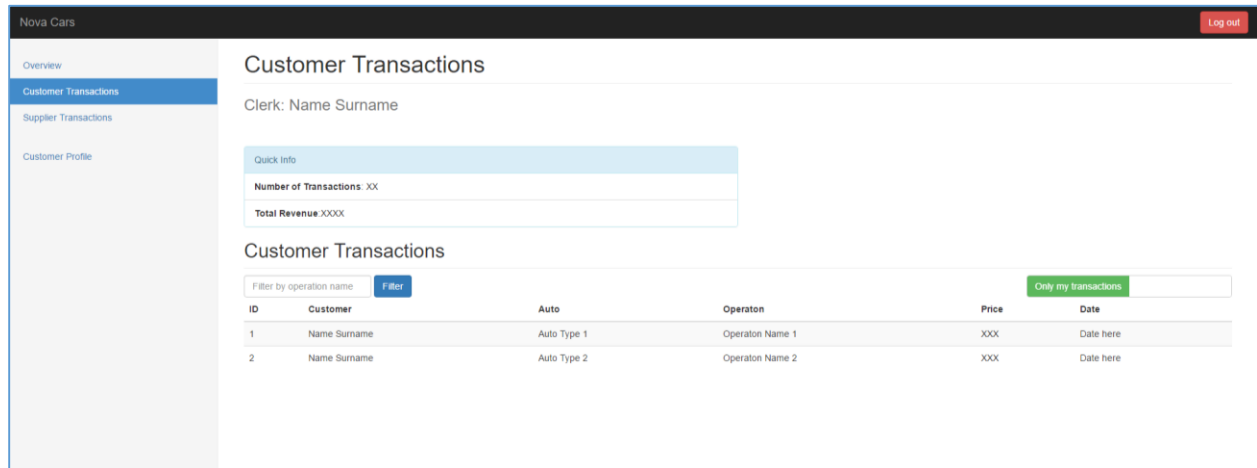


Figure 14 - Customer Transactions displayed to the clerk

**Process:** When the clerk opens the “Customer Transactions” screen the name and surname of that clerk is displayed, as well as the “Quick info” field which has the number of transactions and the total revenue fields. A clerk is capable of filter the customer transactions by a given operation name in the text box, and click the filter button. A customer transactions list is provided at the bottom. The clerk can also choose to display “Only my transactions” or “All transactions” from the button provided on the right. The clerk can add new transaction to any customer.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

**For Filtering the Transactions by Operation Name:**

```
SELECT transaction_id, op_name, model, first_name, last_name, amount,
date
FROM (((CustomerOperation join Customer C on customer_email = C.email)
JOIN User U on U.email = customer_email)
JOIN Auto on on_plate = auto_plate)
JOIN Transaction T on T.id = transaction_id
WHERE op_name = @filter;
```

**For Filtering the Transactions by Operation Name and Only Clerk's Transactions Shown:**

```
SELECT transaction_id, op_name, model, first_name, last_name, amount,  
date
```

```
FROM (((CustomerOperation join Customer C on customer_email = C.  
email)
```

```
JOIN User U on U.email = customer_email)
```

```
JOIN Auto on on_plate = auto_plate)
```

```
JOIN Transaction T on T.id = transaction_id
```

```
WHERE op_name = @filter AND clerk_email = @email;
```

**For Adding a New Customer Transaction**

When the add new transaction button is clicked by the clerk, the clerk will be redirected to the New Customer Transaction Screen.

## 4.9. Clerk Dashboard: New Customer Transaction

Operation Name	Providing Department	Cost	Technician
Operation name here	Department name here	XXXX	selectedemail@novacorp.com
Operation name here	Department name here	XXXX	selectedemail@novacorp.com

Figure 15 - New Transaction screen as displayed to a clerk

**Process:** After hitting the “add new transaction” button the clerk is redirected to this page. For adding a new transaction, the clerk has to fill the empty fields: customer e-mail, vehicle plate number and revenue. The operations can be chosen from the list below with the toggles, which shows different operations with their department and cost. A technician can be chosen from the list provided by clicking the arrow button provided. After all the selections are made, the chosen operations are listed above the “Complete Transaction” button, and the clerk can click on this button to complete the transaction.

**Session Variables:** @email

**Inputs:** @filter, @customer\_email, @auto\_plate, @revenue

**Related Variables:**

@transaction\_id := id that is autogenerated incrementally

@date := date of the current day

@total\_cost := sum of all operation costs

**SQL Statements:**

**To display the Technician List When ScrollForm is Clicked:**

```
SELECT email FROM Technician
```

**For Inserting Transaction Entries to the CustomerOperation Table:**

(related variable: @curr\_oper := the name of a toggled operation)

(input for each operation: @operator\_email)

For each operation selected get the cost of the operation via

```
SELECT cost FROM operation WHERE op_name = @curr_oper;
```

Then sum these up to calculate @total\_cost

First the transaction entry will be made:

```
INSERT INTO Transaction (id, amount, date)
VALUES (@transaction_id, @total_cost + @revenue, @date);
```

Then for each toggled operation, the following query will be executed.

```
INSERT INTO CustomerOperation (transaction_id, op_name, clerk_email,
operator_email, customer_email, auto_plate)
VALUES (@transaction_id, @curr_oper, @email, @operator_email,
@customer_email, @auto_plate);
```

Finally, 1% of the @total\_amount will be added to the customer's bonus points'.

```
UPDATE Customer SET bonus_pts = bonus_pts + @total_amount/100 WHERE
email = @customer_email;
```

## 4.10. Clerk Dashboard: Supplier Transactions

ID	Supplier name	Spare part type	Spare part model	Spare part count	Price	Date
1	Name Surname	Spare Part Type 1	Spare Part Model 1	XX	XXX	Date here
2	Name Surname	Spare Part Type 2	Spare Part Model 2	XX	XXX	Date here

Figure 16 - Supplier transactions as displayed to the clerk

**Process:** The clerk is able to see the “Supplier Transactions” page and have some “Quick info” about these transactions such as the number of transactions and the total cost. Supplier transactions are listed with the details: supplier name, spare part type, model and count, and the price and date of the transaction. It is possible to filter the transactions by supplier name or the type of the spare part. The clerk can also choose to display “Only my transactions” or “All transactions” from the button provided on the right.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

**For Filtering the Transactions by Supplier or Spare Part Type:**

```
SELECT transaction_id, supplier_name, part_type, part_model, count,  
amount, date
```

```
FROM SparePartOrder JOIN Transaction T on T.id = transaction_id
```

```
WHERE part_type = @filter OR supplier_name = @filter;
```



#### 4.11. Clerk Dashboard: Detailed Customer Transaction Information

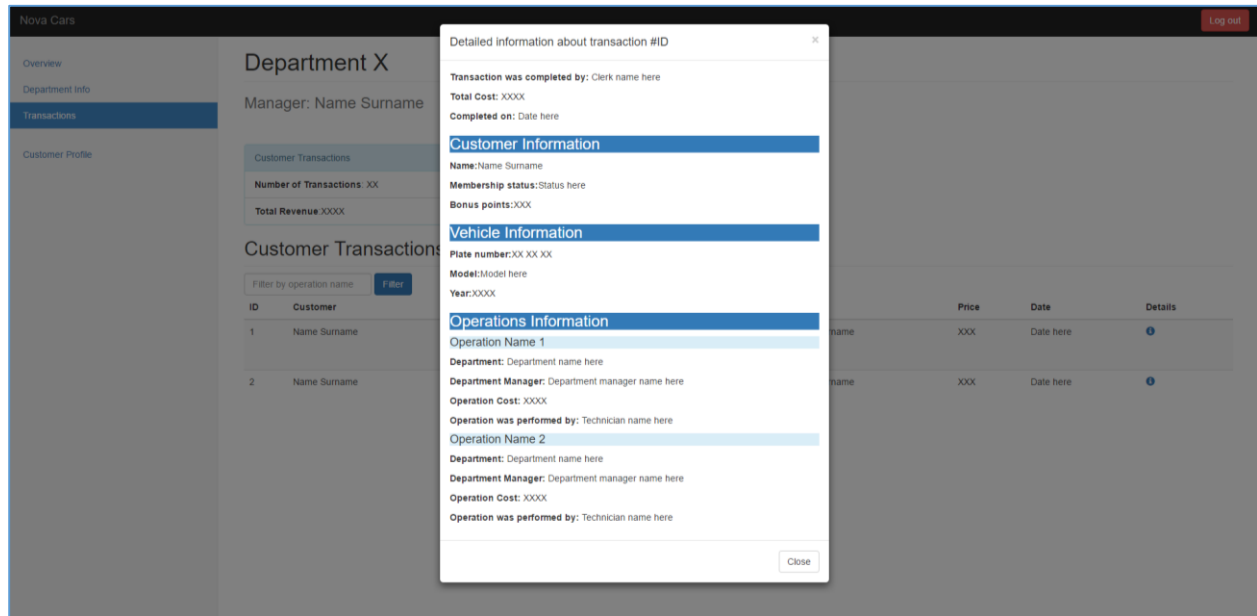


Figure 17 - Modal for displaying detailed information about a customer transaction

**Process:** The detailed information about the customer transaction includes fields: by which technician the transaction was completed, the total cost, and the date the transaction was completed. This page also displays the customer information, the vehicle information and the operations information in that transaction. The screen can be closed via the “Close” button at the bottom.

## 4.12. Clerk Dashboard: Detailed Supplier Transaction Information

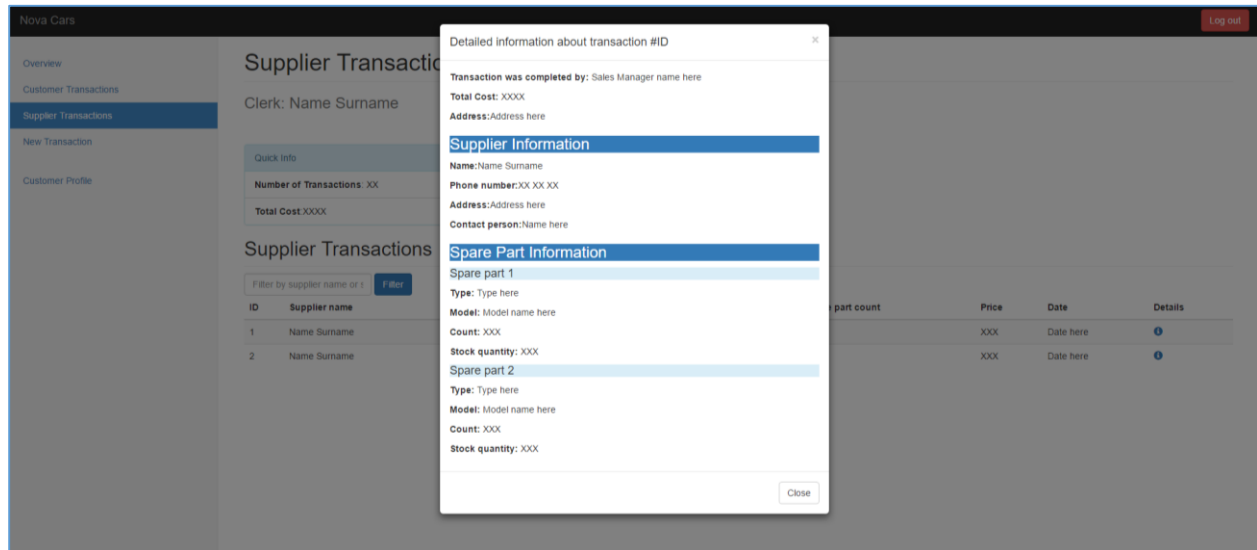


Figure 18 - Detailed information screen for a customer transaction as displayed to a clerk

**Process:** After clicking on the details button on Supplier Transactions, the detailed information page is displayed. The detailed information about the supplier transaction includes fields: by which technician the transaction was completed, the total cost, the address of the supplier and the sales manager responsible for the transaction.

This page also displays the supplier information and spare part information as a list. The screen can be closed via the “Close” button at the bottom.

#### 4.13. Sales Manager Dashboard: Supplier Transactions

The screenshot shows a web application interface for 'Nova Cars'. On the left is a sidebar with navigation links: Overview, Suppliers Info, Supplier Transactions (highlighted), New Transaction, and Customer Profile. The main content area is titled 'Supplier Transactions' and includes a 'Clerk: Name Surname' label. Below this is a 'Quick Info' box with 'Number of Transactions: XX' and 'Total Cost: XXXX'. The main section is a table of 'Supplier Transactions' with columns: ID, Supplier name, Spare part type, Spare part model, Spare part count, Price, Date, and Details. The table contains two rows of data. Above the table is a filter input and a 'Filter' button. To the right of the table are buttons for 'All transactions' and 'Details'. At the bottom left of the table area is a '+ New Transaction' button.

ID	Supplier name	Spare part type	Spare part model	Spare part count	Price	Date	Details
1	Name Surname	Spare Part Type 1	Spare Part Model 1	XX	XXX	Date here	<a href="#">Details</a>
2	Name Surname	Spare Part Type 2	Spare Part Model 2	XX	XXX	Date here	<a href="#">Details</a>

Figure 19 - List of supplier transactions as displayed to a sales manager

**Process:** After navigating through the sidebar and choosing the “Supplier Transactions” tab, the sales manager is directed to this screen. This screen has a “Quick info” about these transactions such as the number of transactions and the total cost. Supplier transactions are listed with the details: supplier name, spare part type, model and count, and the price and date of the transaction. It is possible to filter the transactions by supplier name or the type of the spare part. The sales manager can also choose to display “Only my transactions” or “All transactions” from the button provided on the right. It is also possible to see the details with the button next to each supplier transaction. It is possible to add a new transaction with the “New Transaction” button.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

**For Filtering the Transactions by Supplier or Spare Part Type:**

```
SELECT transaction_id, supplier_name, part_type, part_model, count,  
amount, date
```

```
FROM SparePartOrder JOIN Transaction T on T.id = transaction_id
```

```
WHERE type = @filter OR supplier_name = @filter;
```

**For Filtering the Transactions by Operation Name and Only Clerk's Transactions Shown:**

```
SELECT transaction_id, supplier_name, part_type, part_model, count,  
amount, date  
  
FROM SparePartOrder JOIN Transaction T on T.id = transaction_id  
  
WHERE (type = @filter OR supplier_name = @filter) AND sales_mgr_email  
= @email;
```

**For Adding a New Spare Part**

When the add new spare part button is clicked, a pop-up window with input fields for the following fields for the SparePart entity will appear: type, model, price. When the Save button that will appear is clicked the following query will execute.

(inputs = @type, @model, @price)

```
INSERT INTO Auto (type, model, stock_quantity, price)  
VALUES (@type, @model, 0, @price);
```

**For Adding a New Supplier Transaction**

When the add new transaction button is clicked by the clerk, the clerk will be redirected to the New Supplier Transaction Screen.

#### 4.14. Sales Manager Dashboard: New Supplier Transaction

Figure 20 - New supplier transaction screen as displayed to a sales manager

**Process:** After clicking on the “New Transaction” on Supplier Transactions page, or navigating on the sidebar, New Transaction page is displayed. In this page, the sales manager chooses a supplier name from a list by clicking the arrow button. Then, he fills in the required parts: Spare part type, model and count. Then he completes this new transaction via clicking the “Complete Transaction” button.

**Session Variables:** @email

**Inputs:** @part\_type, @part\_model, @supplier\_name, @count

**Related Variables:**

@transaction\_id := id that is autogenerated incrementally

@date := date of the current day

@total\_cost := sum of all cost of all spare parts

**SQL Statements:**

**To Display Supplier List When ScrollForm Is Clicked:**

```
SELECT name FROM Supplier
```

**For Inserting Transaction Entries to the SparePartOrder Table:**

@total\_cost is calculated by multiplying

```
SELECT price FROM SparePart WHERE type = @part_type AND model =  
@part_model;
```

First the transaction entry will be made:

```
INSERT INTO Transaction (id, amount, date)  
VALUES (@transaction_id, @total_cost, @date);
```

Then:

```
INSERT INTO SparePartOrder (transaction_id, sales_mgr_email,  
part_type, part_model, supplier_name, count)
```

```
VALUES (@transaction_id, @email, @part_type, @part_model,  
@supplier_name, @count);
```

## 4.15. Sales Manager Dashboard: Supplier Information

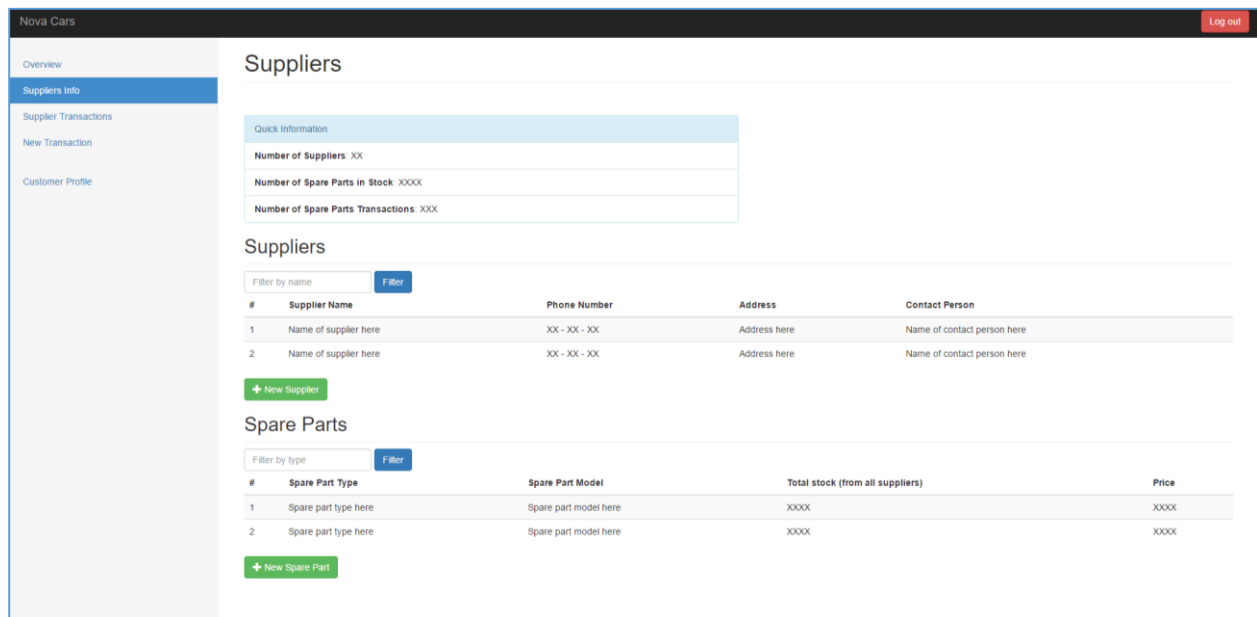


Figure 21 - List of suppliers and spare parts in stock as displayed to a sales manager

**Process:** After clicking on “Suppliers Info” tab on the sidebar, Suppliers screen is displayed. In this screen, “Quick info” field keeps the number of suppliers, number of spare parts in stock and the number of spare parts transactions. There is a list for suppliers and another one for spare parts. It is possible to filter the suppliers list by name and the spare parts by type. Also the sales manager can add new suppliers and new spare parts by clicking the corresponding buttons. Suppliers list has: Suppliers name, phone number, address and contact person. In Spare Parts list there are: spare part type, model and total stock from all suppliers as well as the price of that part.

**Session Variables:** @email

**Inputs:** @filter

**SQL Statements:**

**For Filtering the Suppliers by name:**

```
SELECT * FROM Supplier where name = @filter;
```

**For Adding a New Supplier**

When the add new supplier button is clicked, a pop-up window with input fields for the following fields for the Supplier entity will appear: name, phone, address, contact\_name. When the Save button that will appear is clicked the following query will execute.

(inputs = @name, @phone, @address, @contact\_person)

```
INSERT INTO Supplier (name, phone, address, contact_person)
VALUES (@name, @phone, @address, @contact_person);
```

**For Filtering the Spare Parts by type:**

```
SELECT * FROM SparePart where type = @filter;
```

**For Adding a New Spare Part**

When the add new spare part button is clicked, a pop-up window with input fields for the following fields for the SparePart entity will appear: type, model, price. When the Save button that will appear is clicked the following query will execute.

(inputs = @type, @model, @price)

```
INSERT INTO SparePart (type, model, stock_quantity, price)
VALUES (@type, @model, 0, @price);
```



## 5. Advanced Database Components

### 5.1. Views

#### 5.1.1 Manager's Service Department View

Manager can only see employees under his/her department. When a manager logs in, he should be able to see employees under his control. This view can be used while listing staff belonging to a certain manager.

```
CREATE VIEW manager_employees AS

SELECT m.email, first_name, last_name, role, email, salary,
expertise_lvl

FROM (User u NATURAL JOIN Employee e), Manager m

WHERE e.dept_name = (

    SELECT dept_name

    FROM Employee e2

    WHERE e2.email = m.email

);
```

It is supposed that the manager has logged into the system using his/her own email.

### 5.1.2 Clerk's Transaction View

Clerk should be able to see the transaction he/she has done.

```
CREATE VIEW clerk_transaction AS
SELECT cust.Name, co.op_name,t.price,t.Date
FROM transaction t
NATURAL JOIN CustomerOperation co
NATURAL JOIN Customer cust, Clerk c
WHERE co.auto_plate = SOME (
    SELECT cust2.auto
    FROM Customer cust2
)
AND
c.email = (
    SELECT c2.email
    FROM clerk c2
    WHERE c2.email = c.email
);
```

It is supposed that the clerk has logged into the system using his/her own email.

### 5.1.3 Sales Manager's Transaction View

Sales managers must be able to only see the transactions he has done in the system.

```
CREATE VIEW salesmgr_transaction AS

SELECT sm.email, s.supplier, s.type, s.model, spo.count,
t.price,t.date

FROM ((
    Transaction t
    JOIN SparePartOrder spo
    ON t.id = spo.transaction_id
)
JOIN Supply s
ON s.model = spo.part_model
AND s.type = spo.part_type
), SalesManager sm,
WHERE sm.email = spo.sales_mgr_email;
```

It is supposed that the sales manager has logged into the system using his/her own email.

#### 5.1.4 Customer's Operation View

Customers should be able to see the transactions they have done

```
CREATE VIEW customer_operation AS

SELECT c.email, co.auto_plate, a.type, t.price, t.Date,
(t.price/100) AS bonus_pts

FROM ((
    Transaction t
    JOIN CustomerOperation co
    ON t.id = co.transaction_id
)
JOIN Auto a
ON co.auto_plate = a.plate
),Customer c
WHERE c.email = co.customer_email;
```

It is supposed that the customer has logged into the system using his/her own email.

#### 5.1.5 Technician's Operation View

Technicians should be able to see the operation they have done.

```
CREATE VIEW technician_operation AS

SELECT t.email, to.op_name, to.price

FROM TechnicianOperation to

NATURAL JOIN Technician t;
```

It is supposed that the technician has logged into the system using his/her own email.

### 5.2. Stored procedures

Stored procedures will be used during the creation of a transaction and operation. For each transaction and operation, a new tuple for that corresponding table will be created. For transaction it will have the corresponding attributes; id, amount, date. When a transaction is done quantity of spare parts will be updated. For operations it will department name, operation name and technician name. The operations cost will be added to the department's expenditures. So using procedures for these processes will be useful.

If a transaction gets cancelled the spare parts will be updated if needed, with removing the desired row from the table. A procedure will be used to reverse the effect the operation has done by deleting the specific row.

### 5.3. Reports

#### 5.3.1 Total Money Spent by Customers

```
SELECT sum(t.Amount)

FROM Transaction t ,CustomerOperation co

WHERE t.id = co.transaction_id;

GROUP BY co.customer_email;
```

#### 5.3.2 Number of Staff in Each Department

```
SELECT count(*)

FROM Department d, Employee e

WHERE d.Name = e.dept_name

GROUP BY d.Name;
```

#### 5.3.3 Dashboard: Overview

```
SELECT User.first_name, User.last_name, Manager.start_date,
Manager.email, User.password, Employee.salary,
Employee.expertise_lvl, Employee.dept_name

FROM Manager

NATURAL JOIN Employee

NATURAL JOIN User
```

#### 5.3.3 Manager Dashboard: Department Info

##### 5.3.3.2 Operations list

```
SELECT op_name, cost

FROM Operation

WHERE dept_name = (

    SELECT dept_name

FROM Employee

WHERE email = @manager_email)
```

#### 5.3.3.3 Employee list

```
SELECT first_name, last_name, role, salary, expertise_lvl
FROM User natural join Employee
WHERE dept_name = (
    SELECT dept_name
FROM Employee
WHERE email = @manager_email
)
```

#### 5.3.4 Manager Dashboard: Transactions

##### 5.3.4.1 Quick Information: Total Revenue

```
SELECT SUM(cost)
FROM CustomerOperation
WHERE dept_name = (
    SELECT dept_name
FROM Employee
WHERE email = @manager_email
)
```

#### 5.3.4.1 Customer transactions

```
SELECT CustomerOperation.transaction_id,  
CustomerOperation.op_name, Auto.model, U.first_name, U.last_name,  
T.amount, T.date  
  
FROM (((  
  
        CustomerOperation  
  
        JOIN Employee E  
  
        ON E.email = clerk_email  
  
    )  
  
    JOIN Customer C  
  
    ON customer_email = C.email  
  
    )  
  
    JOIN Auto  
  
    ON plate = auto_plate  
  
    )  
  
    JOIN User U  
  
    ON U.email = clerk_email  
  
    )  
  
JOIN Transaction T  
  
ON T.transaction_id = CustomerOperation.transaction_id
```

### 5.3.5 Sales Manager Dashboard

#### 5.3.5.1 Supplier Transactions

```
SELECT spo.supplier_name, spo.part_model, spo.part_type,  
spo.count, t.price, t.date  
  
FROM SparePartOrder spo, SalesManager sm, Transaction t  
  
WHERE spo.sales_mgr_email = @salesmgr_email  
  
AND spo.transaction_id = t.id;
```

## 5.4.5 Clerk Manager Dashboard

### 5.4.5.1 Customer Transactions

```
SELECT u.first_name,u.last_name, a.model,t.amount,t.date
FROM User u, Auto a, Transaction t, Clerk c, CustomerOperation co
WHERE t.id = co.transaction_id
AND u.email = co.customer_email
AND auto_plate = (
    SELECT c2.auto
    FROM Customer c2
    WHERE c2.email = u.email
);
```

### 5.4.5.2 New Customer Transaction

```
SELECT op.op_name,op.dept_name,op.amount, to.email
FROM Operation op, TechnicianOperation to
WHERE op.dept_name = to.dept_name
AND op.op_name = to.op_name;
```

### 5.4.5.4 Supplier Transaction

```
SELECT Spa.name, spa.part_type, spa.part_model, spa.count,
t.price, t.date
FROM SparePartOrder spa, Transaction t
WHERE spa.transaction_id = t.id;
```



#### 5.4.6 Customer Dashboard

```
SELECT a.model, op.co_name, co.price, t.date, (op.price/100)
FROM (
    Transaction t
    JOIN CustomerOperation co
    ON t.id = co.transaction_id
), Auto a
WHERE a.customer_email = @customer_email
AND a.plate = co.auto_plate
```

##### 5.4.6.1 Customer Dashboard

```
SELECT a.model, co.op_name, co.price,t.date, co.price/100
FROM ((
    Auto a
    JOIN Customer c
    ON a.email = c.email
)
JOIN CustomerOperation co
ON co.customer_email = c.email
)
JOIN transaction t
ON t.id = co.transaction_id
WHERE c.email = @customer_email;
```

#### 5.4. Triggers

- When a transaction between a supplier and sales manager occurs quantity of that certain spare part will be increased.
- When a transaction between a customer and clerk occurs quantity of that certain spare part might be decreased depending on operation.
- When a transaction is done, a certain percentage of the cost will be added to that customer's bonus point.
- When a transaction is done, the customer may choose to decrease its' cost by spending his/her bonus points.
- If a transaction is cancelled between a supplier and a sales manager quantity of that certain spare part will be decreased.
- If a transaction is cancelled between a clerk and a customer, quantity of that certain spare part might be increased depending on operation and bonus points will be removed from that customer's account.

#### 5.5. Constraints

- System cannot be used without logging in.
- The usernames of the staff and the customers are their e-mails.
- Upon deletion of a user, user's information from other tables will be deleted as well.
- Managers are only responsible for their own staff.
- For a spare part quantity must be positive for transaction to happen.
- A Manager's start\_date and employee since date cannot be a future date.
- Customers bonus\_pts attribute cannot be non-negative.
- Only clerks can create transactions with customers and sales managers with suppliers.
- The transaction between a customer and clerk cannot have a positive number.
- The transaction between a sales manager and supplier cannot have a negative number.

### 6. Implementation Plan

For this database management system, for data layer, we will use MySQL. For application logic and UI will use HTML, JavaScript and PHP.