

**ONLINE CARS SELLING SYSTEM**

**Final Release Document**

– Hanoi, Jul 2022 –

**Table of Contents**

**I. Deliverable Package.………………………………………………………………………… 3**

**II. Installation Guides……………………………………………………………………………. 3**

**III. User Manual…………………………………………………………………………………… 3**

**1. Overview…………………………………………………………………………………. 3**

**2. <<Workflow Name 1>>......................................................................................... 3**

**2. <<Workflow Name 2>>......................................................................................... 3**

**I. OverView**

## **1.** **Description**

The Online Car Selling System is a web-based application that allows users to search and consult information about car models in the market. The system provides a platform for car manufacturers, car dealers and individual car owners to post information about their vehicle models.

The application workflow can be described as follows:

Vehicle Search: Users can search for vehicles based on make, model, engine type, make, and other specifications.

View vehicle details: After finding a suitable vehicle, users can view detailed information about the vehicle, including images, specifications and price.

Contact the seller: If the user is interested in a particular car model, they can contact the seller directly through the system. This can be done via text, email or phone.

Deposit and payment: After deciding to buy a car, users can deposit and pay through the application. The system will securely process the payment and provide confirmation to both the buyer and the seller.

## **2.** **Flow chart**

**II. Deliverable Package**

| **No.** | **File** | **Notes** |
| --- | --- | --- |
| 1 | * fn2.sql | Database, including both table structures and table data |
| 2 | Template2\_SRS\_Group2.docx | Final SRS (Requirement) Document |
| 3 | Template3\_SDS\_Document-Group2.docx | Final SDS (Design) Document |
| 4 | Template5\_Final Release Document- Group 2 | Final status for the application functions, which include:  - Function name, feature, roles, function description (changes compared to the original requirements are highlighted)  - In Charge, Planned iteration & Actual iteration  - Final status & note on the update you made  - The links to the SRS & SDS are tracked |
| 5 | DMS\_Issues-Report.xlsx | Final issues tracking list of the whole project |

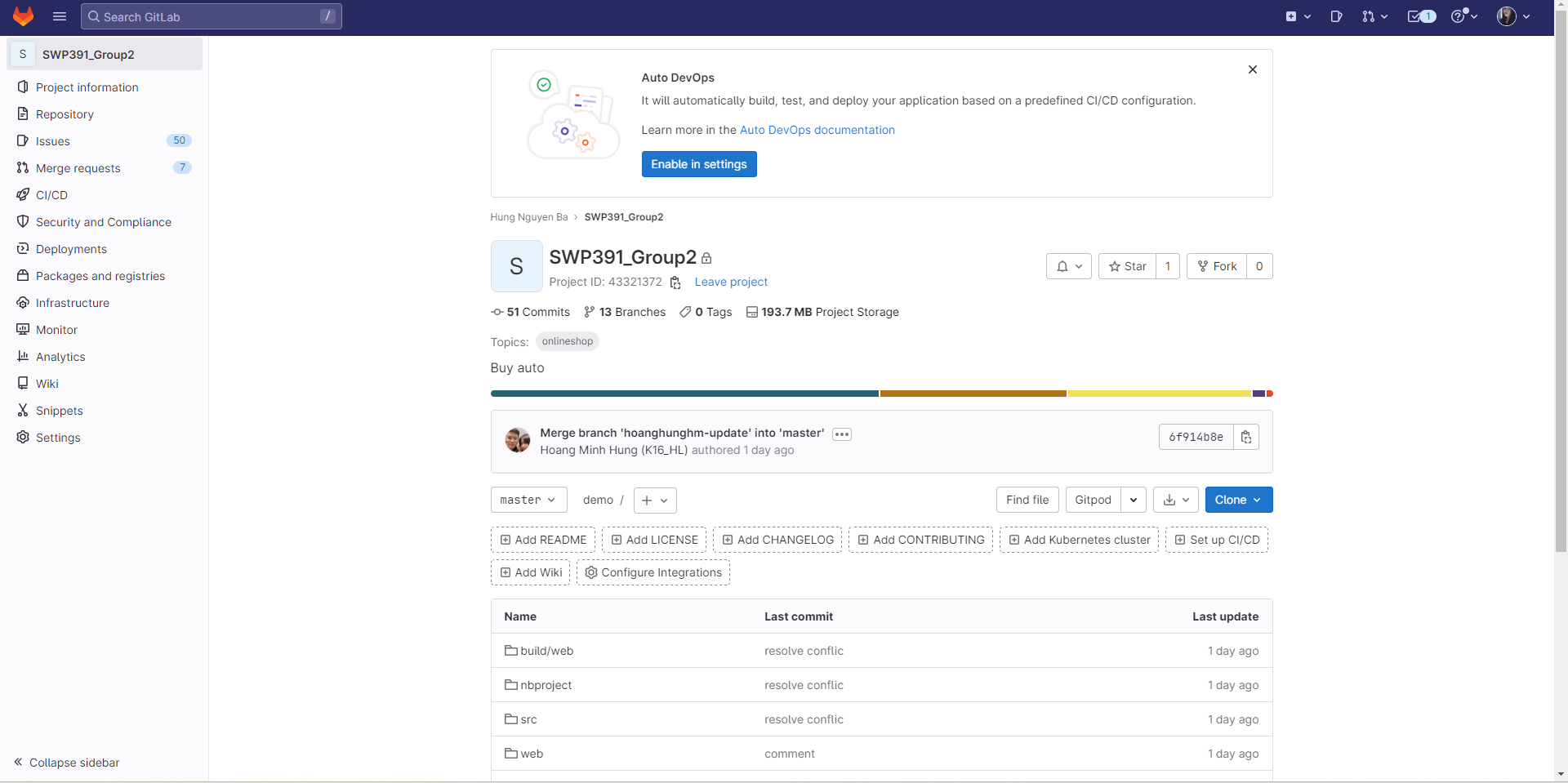
Other related deliverables:

· Tagged source codes: <<https://gitlab.com/hungnbhe160009/demo.git>>

· **III. Installation Guides**

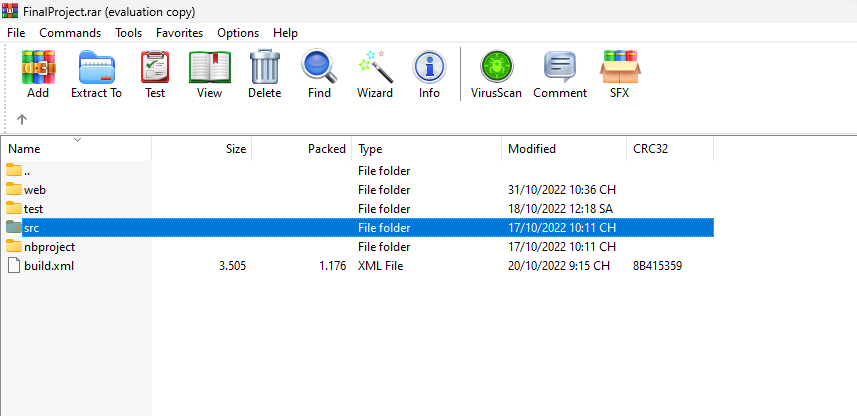
*1.* Step 1: Open gitlab int this link and click download source code of project

Link: https://gitlab.com/hungnbhe160009/demo.git

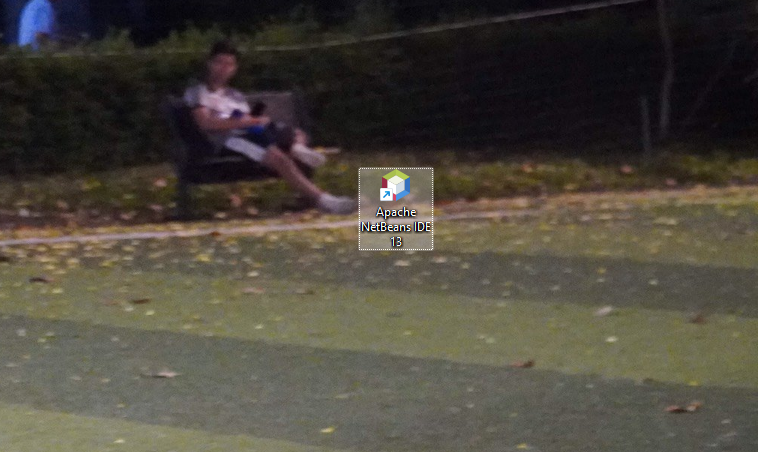


*2.* Step 2: Unzip the file and save in a storage

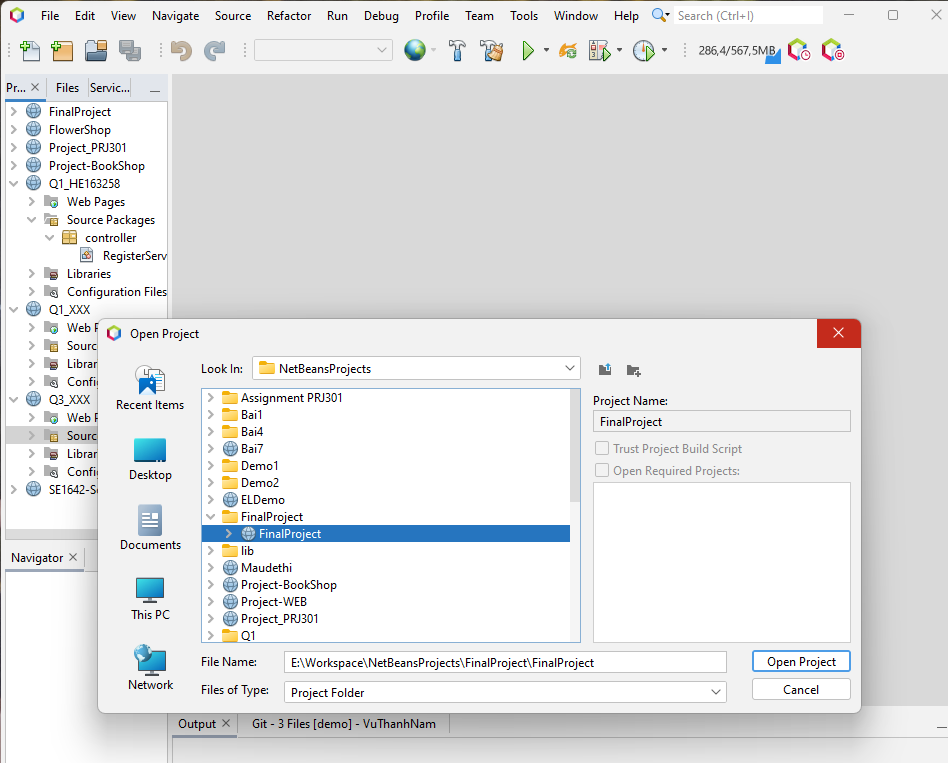
*If you use the RAR you can click in Wizard and choose the storage you want save.*

**

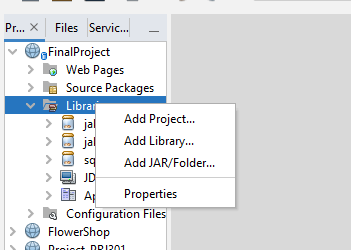
*3.* Step 3: Open Apache Netbeans to open a project

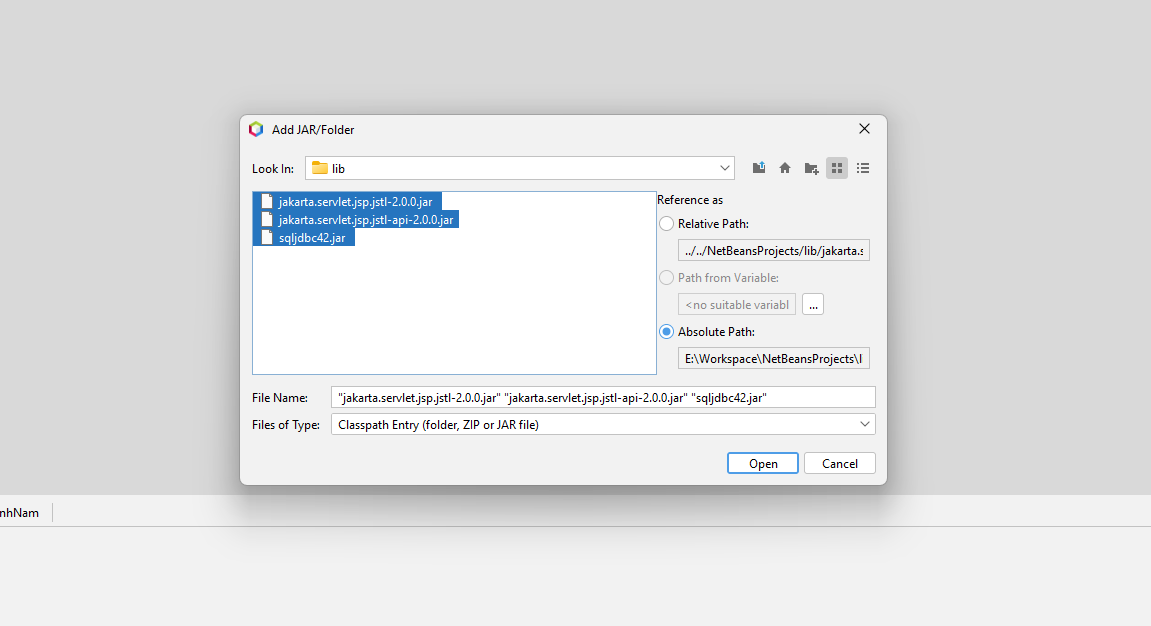


*4.* Step 4: Select “file” at the top left of the screen. Then choose “Open project” then select the project you saved in the folder before. And click Open Project.

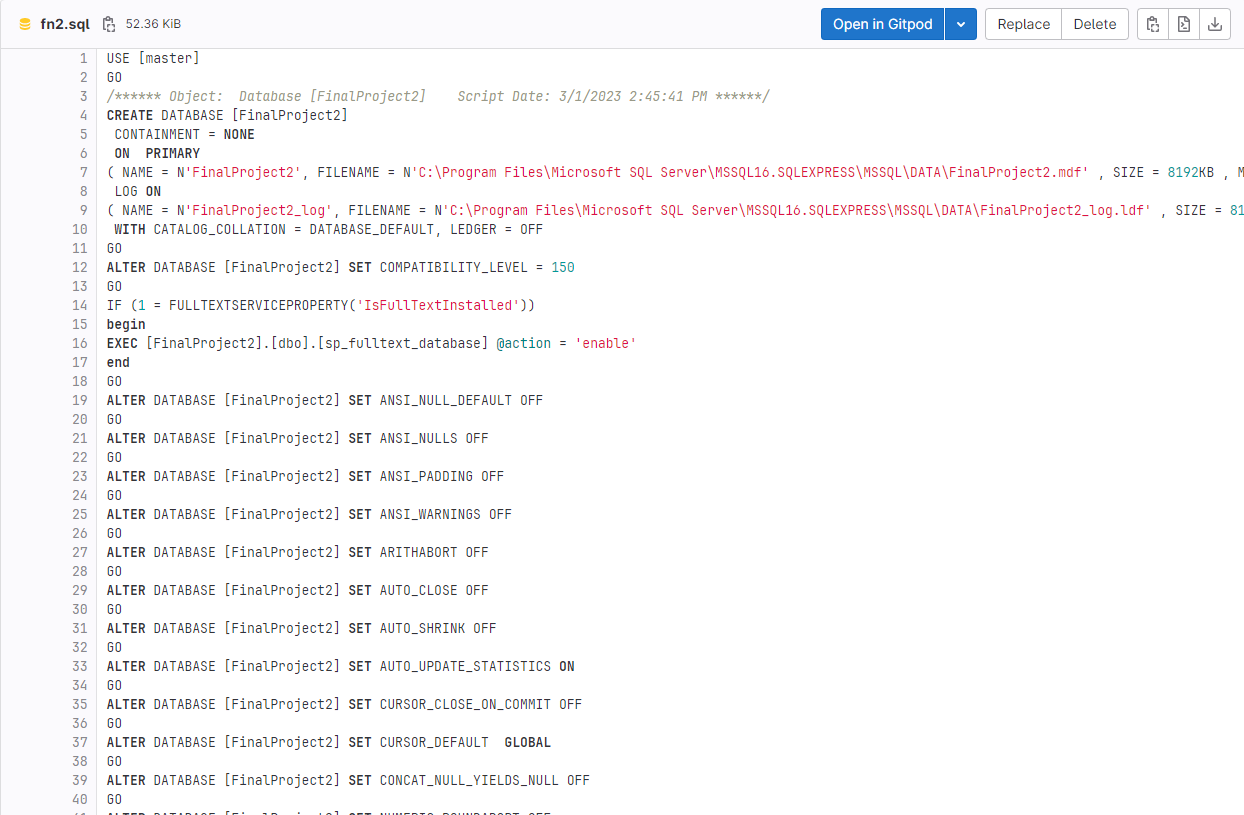


*5.* Step 5: Right click on the library then select Add Jar/Folder. Then Open lib which have a list library and framework for the main project.

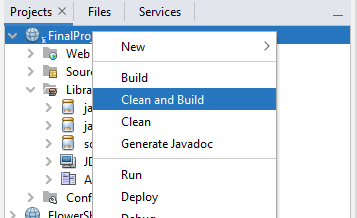


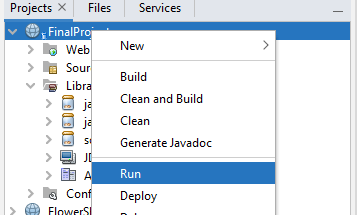


*6.* Step 6: Open The script fn2.sql And run it on Microsoft SQL server (you must install Microsoft SQL Server to run)



*7.* Step 7: After add library and data for the project Right click on project choose Clean and Build. Then Click run project or you can click on green rectangle on top.





*8.* Step 8: After running you the website will display on local web. Now you can use the local web and test all function of website.



# **IV. User Manual**

## **1. Homepage**

## 

**a, Screen**



**b, Description**

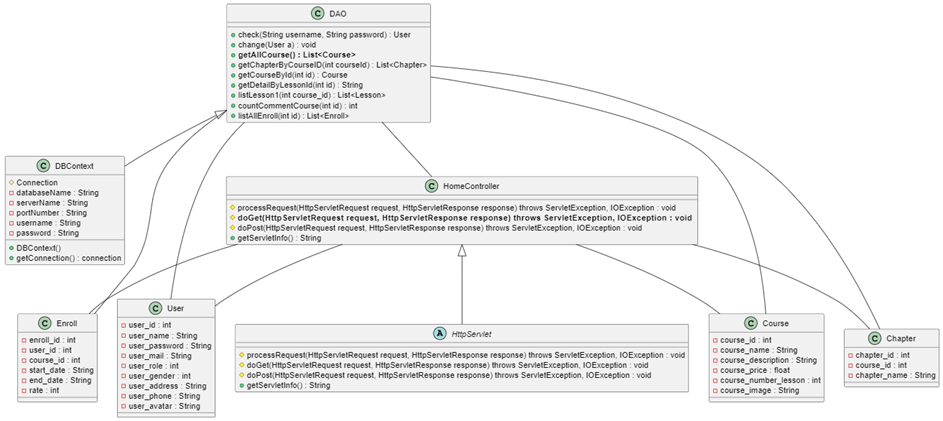
- This will be the main screen for user when entering the website. The HomeController will be responsible for display all items that home page contains.

- Functional Requirements:

· Entering website will show logos, Car, News , Installment, Promotion .

· Cars must be displayed by using DAO class, connect to database and then display on website.

**c, Class diagram**



**d, Class Specifications**

***HomeController***

| No | Method | Description |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | List all courser from databse into homepage |

***DBContext***

| No | Method | Description |
| --- | --- | --- |
| 01 | getConnection() | To connect with database |
| 02 | closeResultSet(ResultSet resultSet) | To close result set |
| 03 | closePreparedStatement(PreparedStatement preparedStatement) | To close prepared statement |
| 04 | closeConnection(Connection connection) | To close connection |

***User***

| No | Method | Description |
| --- | --- | --- |
| 01 | Constructors | Entity initialization |
| 02 | Getter | Get value of each attributes |
| 03 | Setter | Set value of each attributes |

***DAO***

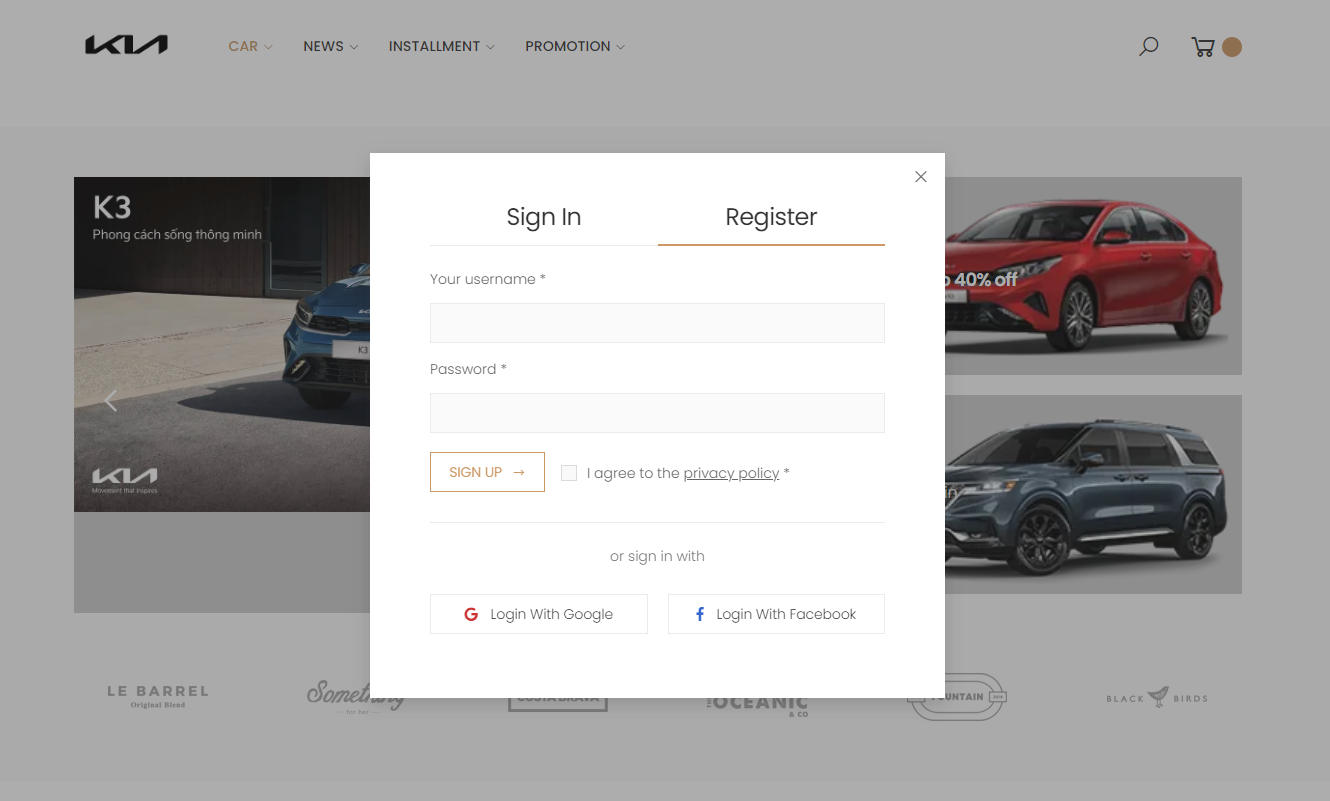
| No | Method | Description |
| --- | --- | --- |
| 01 | GetAllCourse(List<Course>) | Get all Cousre from database |

**e, Sequence Diagram(s)**

******

## **2. Register**

**a, Screen**

**

**b, Description**

- This will be what screen should be when user register a new account. The SignUpServlet class will responsible for the function.

- Functional Requirements:

· Check if username is existed or not. If it was existed, display an error that user should choose another username to register.

· Username and password must be validated.

· When user type in password and confirm password, that two passwords must be exactly the same. Otherwise, system should display an error show that passwords don’t match each other.

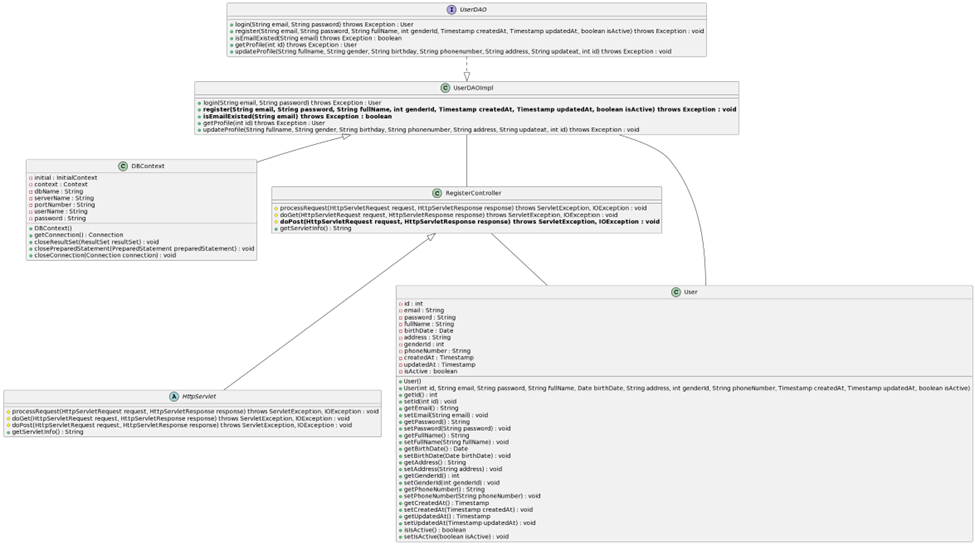
· Register successfully will save username and password into database.

· Allow system to remember user, so that user don’t have to login again.

- Security:

· All passwords must be encrypted and stored securely.

**c, Class diagram**



**d, Class Specifications**

#### RegisterController

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | doPost(HttpServletRequest request, HttpServletResponse response) | Get email, password and fullname from register form, check it and add to the database |

#### DBContext

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | getConnection() | To connect with database |
| 02 | closeResultSet(ResultSet resultSet) | To close result set |
| 03 | closePreparedStatement(PreparedStatement preparedStatement) | To close prepared statement |
| 04 | closeConnection(Connection connection) | To close connection |

#### UserDAO

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | register(String email, String password, String fullName, int genderId, Timestamp createdAt, Timestamp updatedAt, boolean isActive) | Interface abstract method register with email, password, and fullName |
| 02 | isEmailExisted(String email) | Interface abstract method check email is existed in the system or not |

#### UserDAOImpl

Class Methods

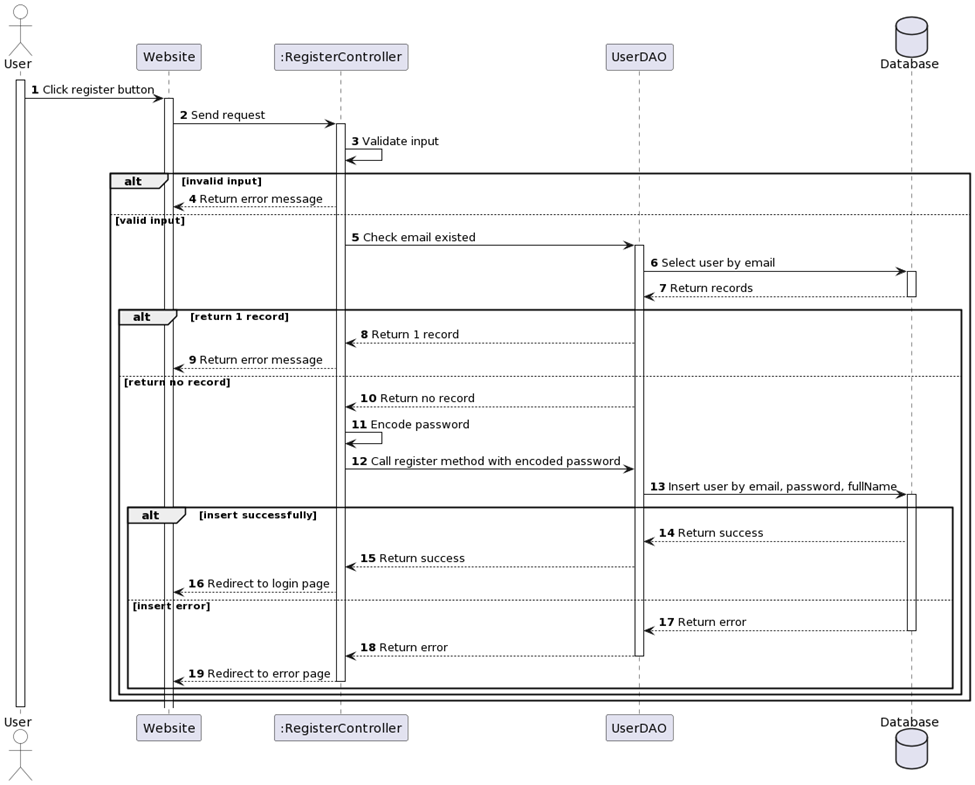
| No | Method | Description |
| --- | --- | --- |
| 01 | register(String email, String password, String fullName, int genderId, Timestamp createdAt, Timestamp updatedAt, boolean isActive) | Register account of the system with email, password and fullName |
| 02 | isEmailExisted(String email) | Check email is existed in the system or not |

#### User

Class Methods

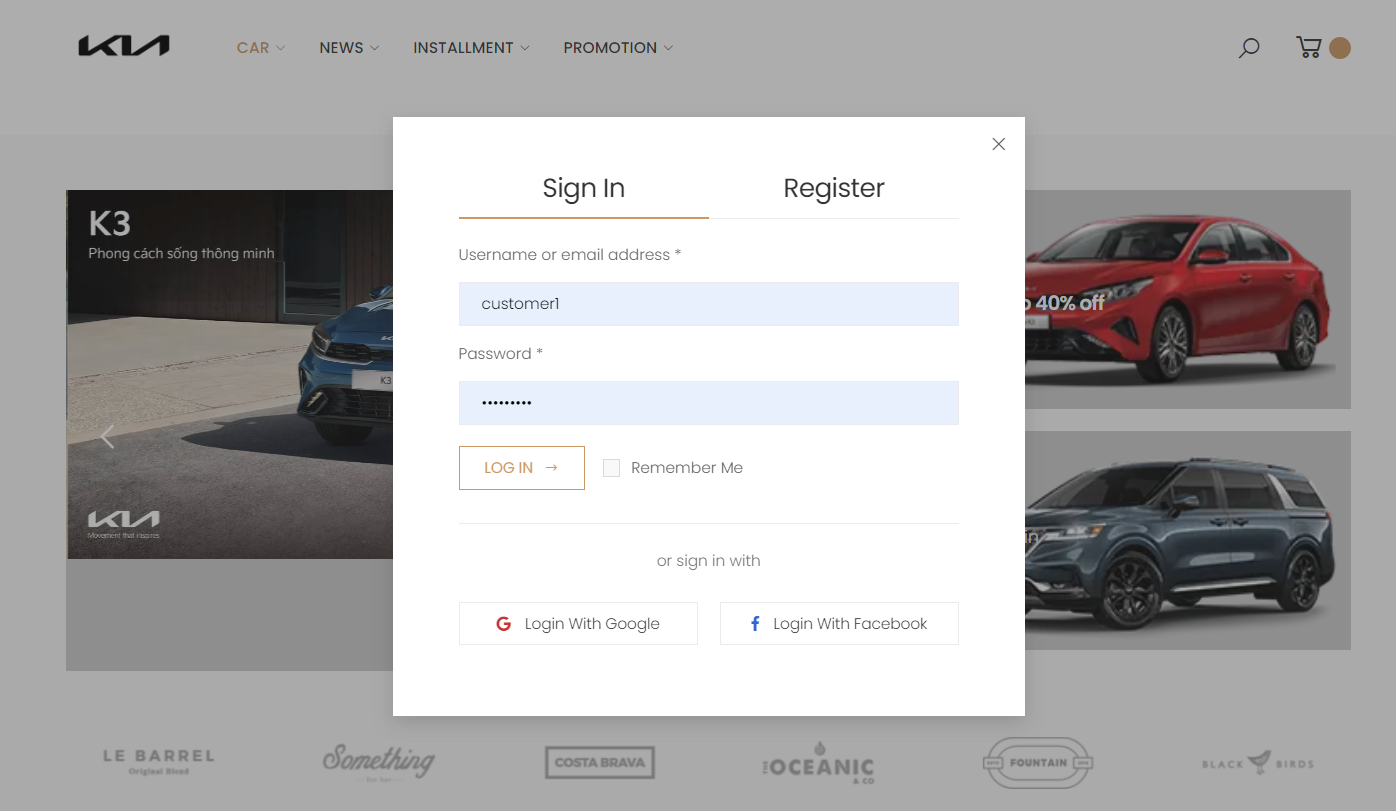
| No | Method | Description |
| --- | --- | --- |
| 01 | Constructors | Entity initialization |
| 02 | Getter | Get value of each attributes |
| 03 | Setter | Set value for each attributes |

**e, Sequence Diagram(s)**

**

## **3. Login**

**a, Screen**

****

**b. Description**

- The purpose of this document is to specify the requirements for a Login Controller, which will be a module in a larger software system. The Login Controller will be responsible for handling user authentication and authorization, allowing users to access secure areas of the software system based on their credentials.

\* Functional Requirements:

· The Login Controller must provide the following functionality:

· Accept a user's login credentials (username and password).

· Verify the user's credentials against the system's user database.

· Grant access to the system's secure areas if the user's credentials are valid.

· Display an error message if the user's credentials are invalid.

· Allow users to reset their passwords if they forget them.

\* Logout Functionality:

· Allow users to log out of the system.

· End the user's session and revoke their access to the system's secure areas.

· Non-Functional Requirements.

\* Security:

· All passwords must be encrypted and stored securely.

- Password reset functionality must follow secure password reset guidelines.

**c. Class diagram**



***d. Class Specifications***

LoginController

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get email and password from Login Form and compare data in database. If user exist in database with correct password, user will login successfully* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

**UserDAO**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *login(String email, String password)* | *Interface abstract method login with email and password* |

#### **UserDAOImpl**

***Class Methods***

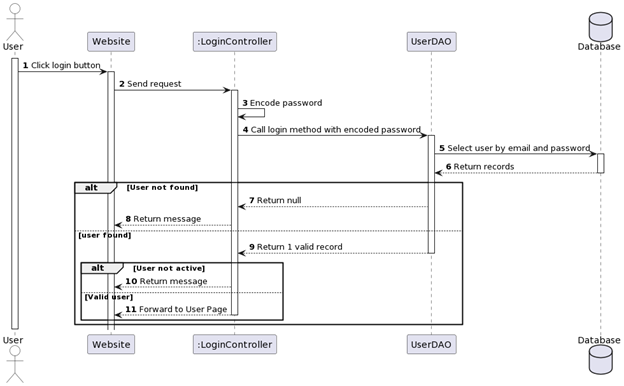
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *login(String email, String password)* | *Login to the system with email and password* |

#### **User**

***Class Methods***

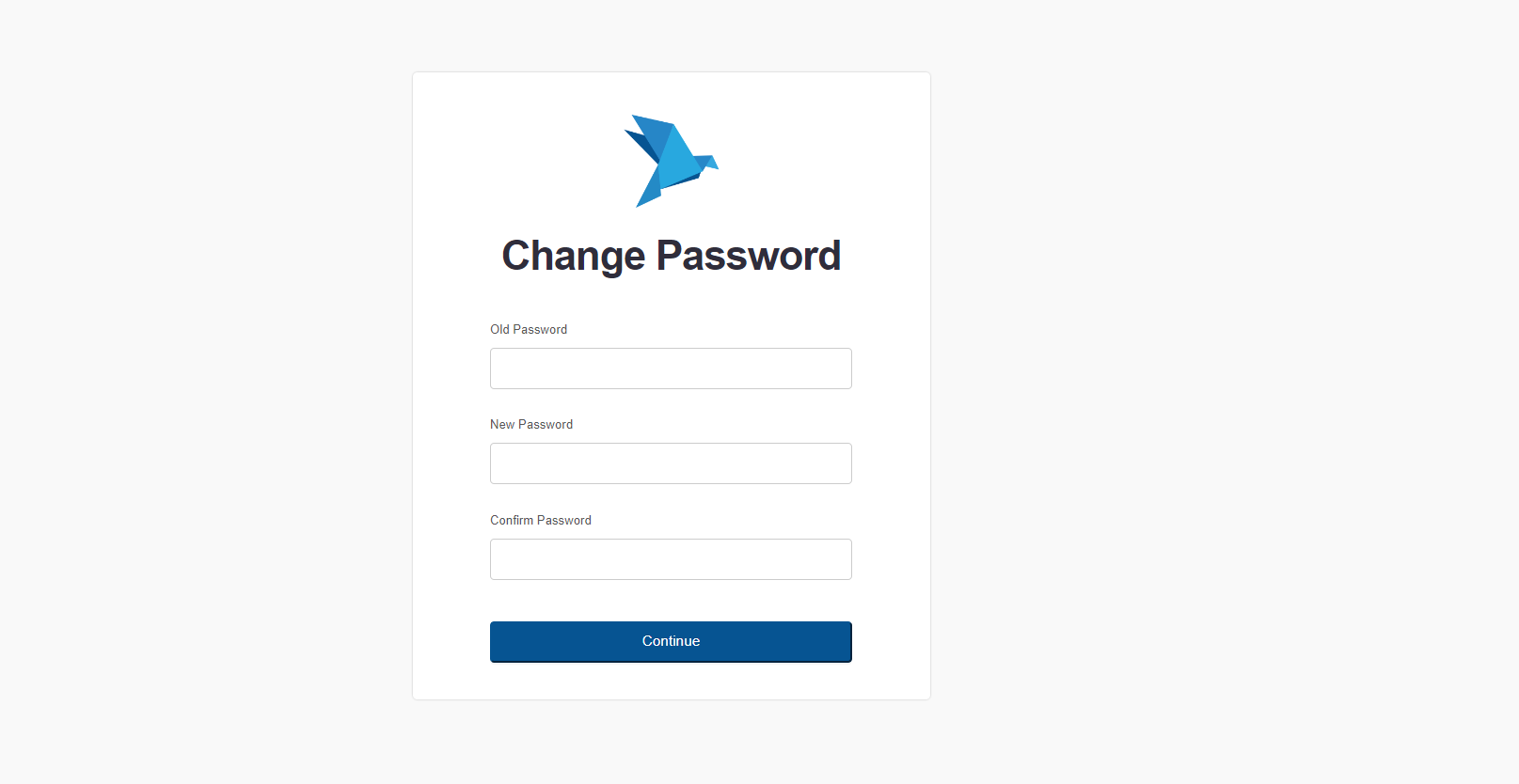
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

**e. Sequence Diagram(s)**



## **4. Change password**

**a. Screen**

****

**b. Description**

- This is the function that can help user to change password is necessary. ChangePassController is the servlet responsible for the function. Connect to database and update user password through username using SQL query.

\* Functional Requirements:

· Verify old password is correct.

· Display an error message if the old password is not correct.

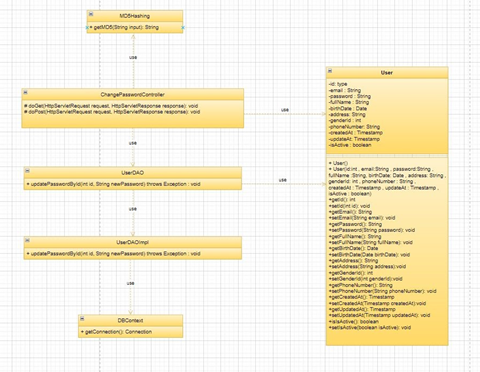
· Validate new password and should be different from old password.

· Password changed in database if successful.

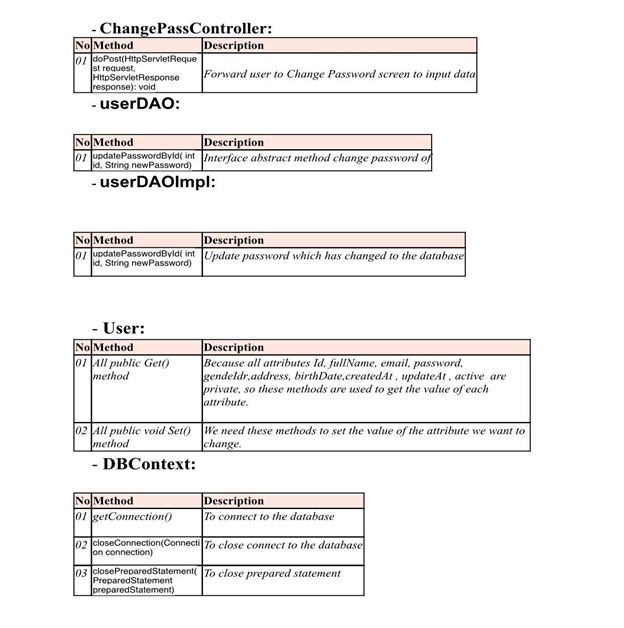
\* Security:

· All passwords must be encrypted and stored securely.

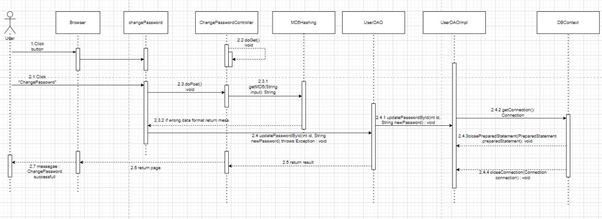
**c. Class diagram**



**d. Class Specifications**

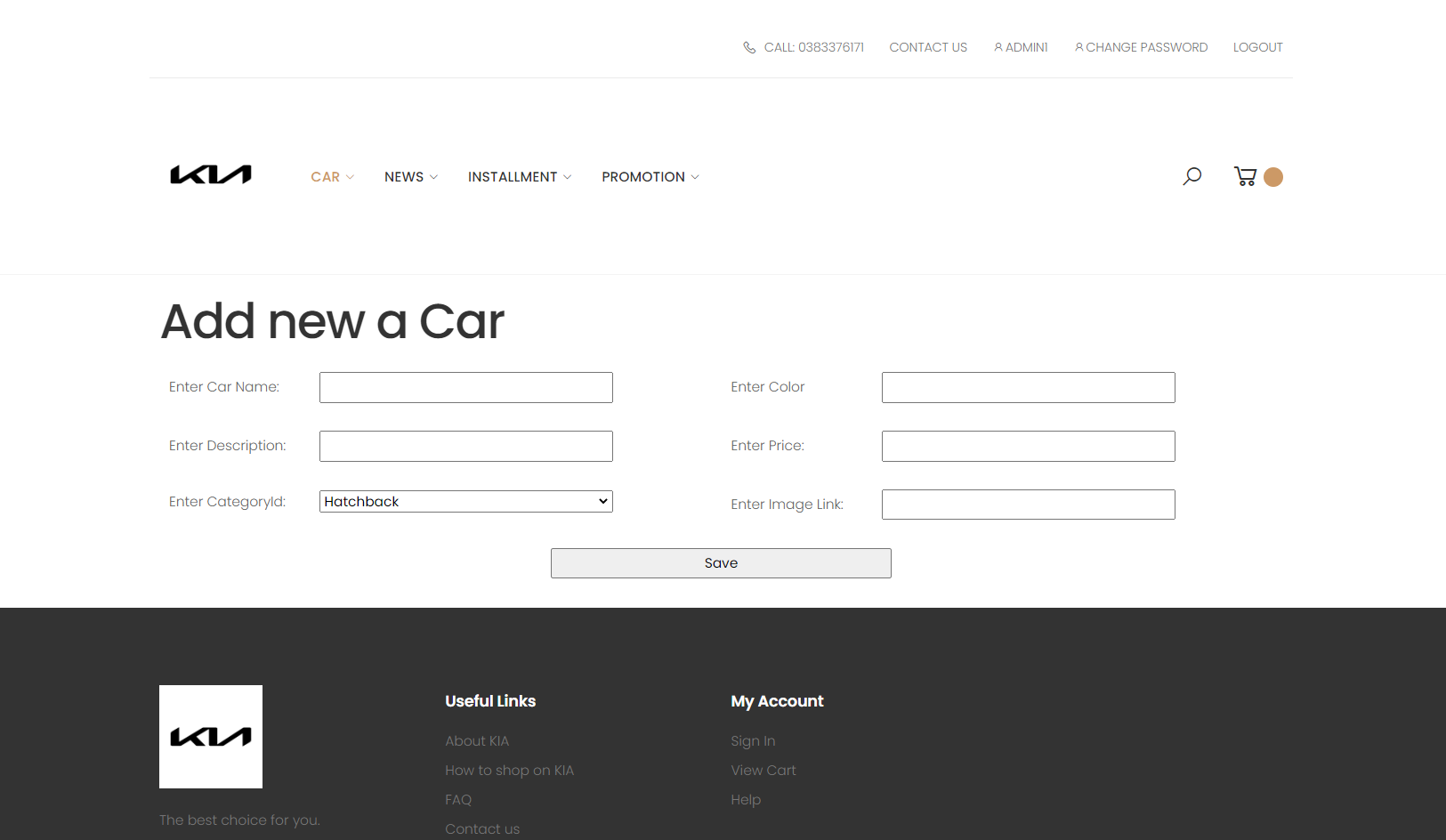


**e. Sequence Diagram(s)**

****

## **5. Create Car**

**a. Screen**

****

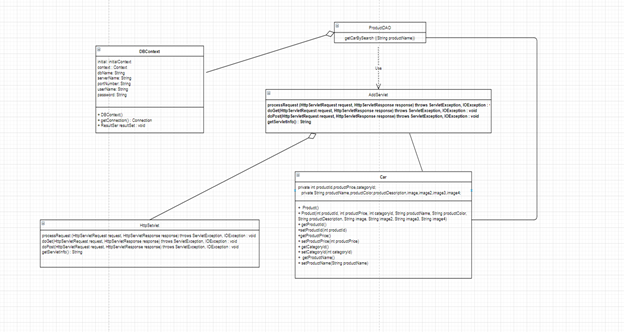
**b. Description**

- When the user is interested in a certain product, the system will allow the user to add the product to the item of interest.

\* Require function:

• View all products

**c. Class diagram**



**d. Class Specifications**

#### **AddCarController**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *with id, name, quantity, description, price,...*  *and create a car from it. Then save new debtor to database.* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

#### **CarDAO**

***Class Methods***

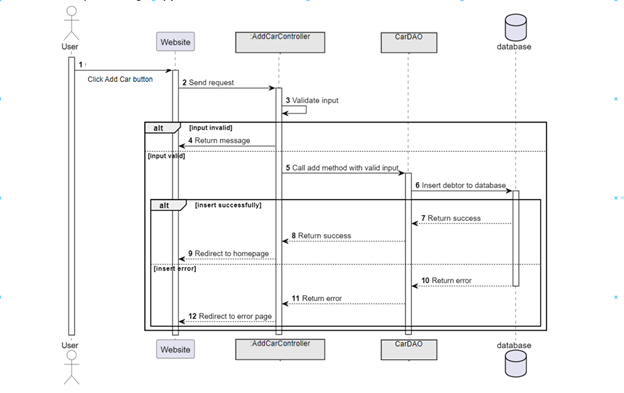
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *add(StringfullName, price,String Description, Quantity)* | *Interface abstract method add car with id, name, quantity, description, price,...* |

Car

***Class Methods***

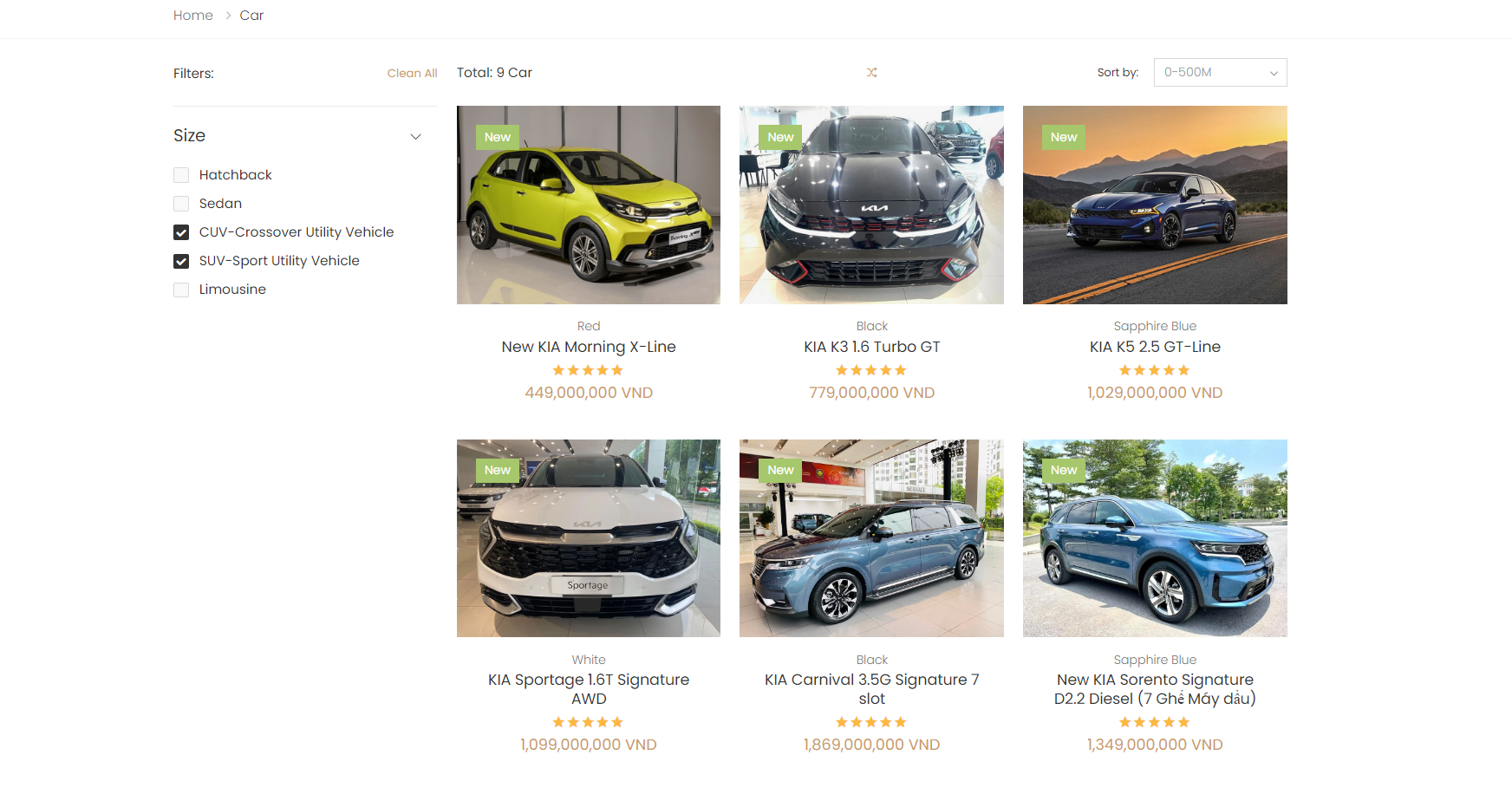
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

### **e. Sequence Diagram(s)**



## **6.View List Car**

**a. Screen**

****

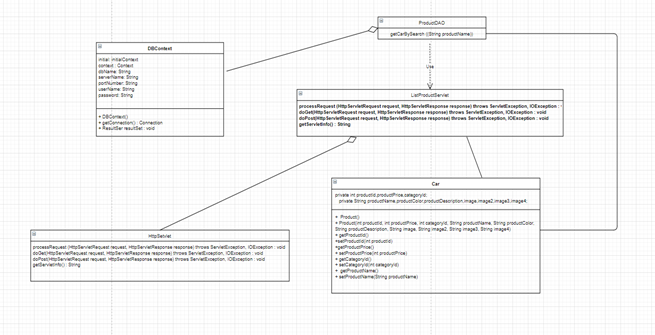
**b. Description**

- when the user has added a product to favorites, the user, manager or admin will see the list of added products from which to make decisions

- Require function:

• View all products in the Cart view

**c. Class diagram**



### **d. Class Specifications**

#### 

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get user id from session and return list car of user who logged in.* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

#### **CarDAO**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCarByProductId(int userId)* | *Interface abstract method getAllProductById.* |

#### **CarDAO**

***Class Methods***

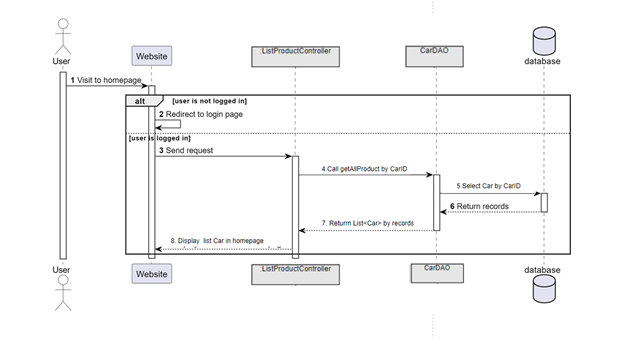
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllProductByUserId(int userId)* | *Return List car are created by user with id is user id.* |

***Car***

***Class Methods***

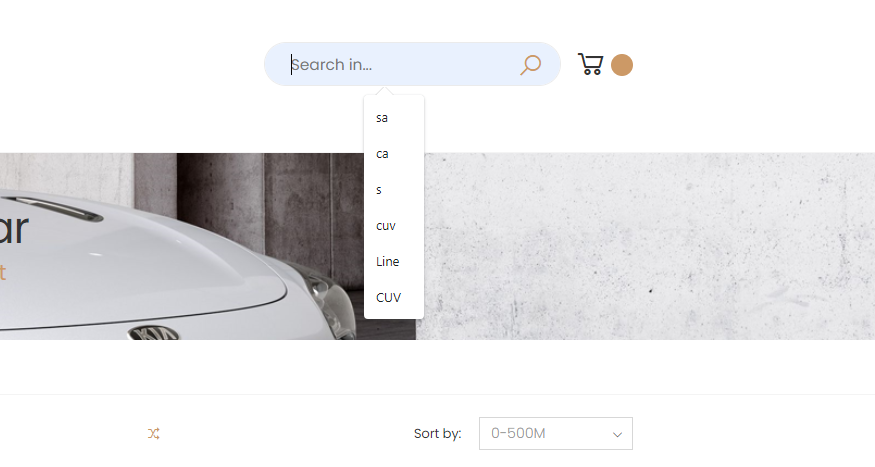
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

### **e. Sequence Diagram(s)**



## **7. Search Car**

**a. Screen**

****

**b. Description**

- This function helps customers easily search for products by name, price, and segment they want

- Require function:

• Log in to the website

### **c.** **Class Diagram**



### **d. Class Specifications**

#### **SearchCarController**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get name, address, email, gender, total debt from Search Form. Return all debtors satisfy the condition.* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

#### **CarDAO**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getProductBySearch(int id, String fullName, String address, String email, int genderId, int userId, float totalDebt )* | *Interface abstract method search debtor with id, name, address, email, gender, user id, total debt.* |

#### **CarDAO**

***Class Methods***

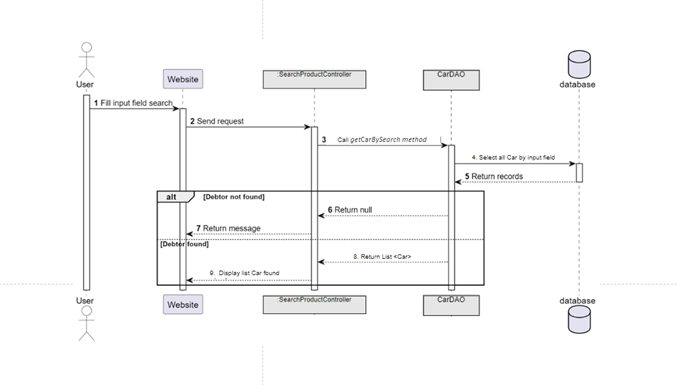
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *get*Car*BySearch (int id, String fullName, String address, String email, int genderId, int userId, Timestamp createdAt, Timestamp updatedAt )* | *Find* Car *with id, fullName, address, email, genderId, userId, createdAt, updatedAt in database.* |

Car

***Class Methods***

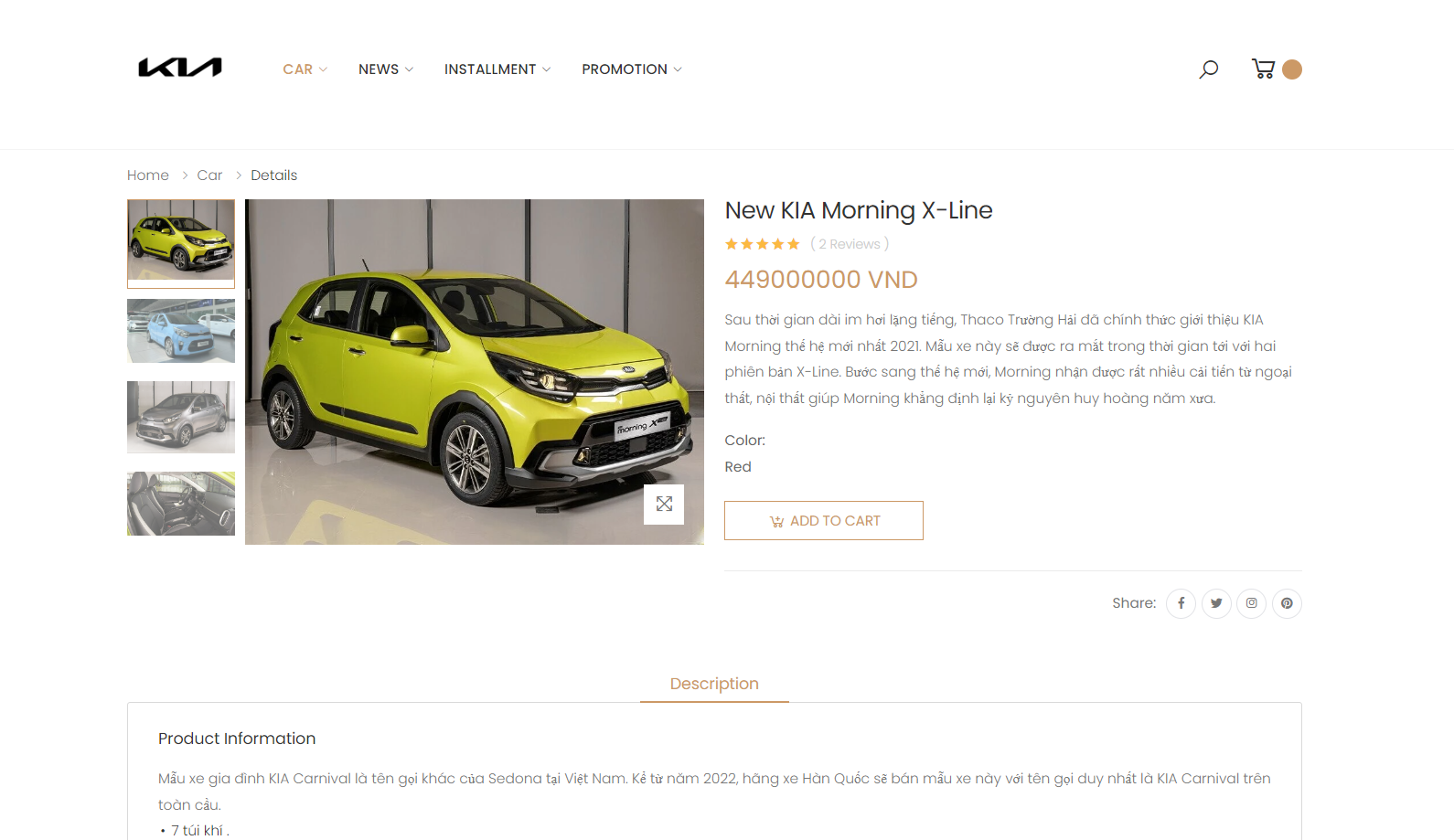
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

### **e. Sequence Diagram(s)**



## **8.View Detail Car**

**a. Screen**

****

**b. Description**

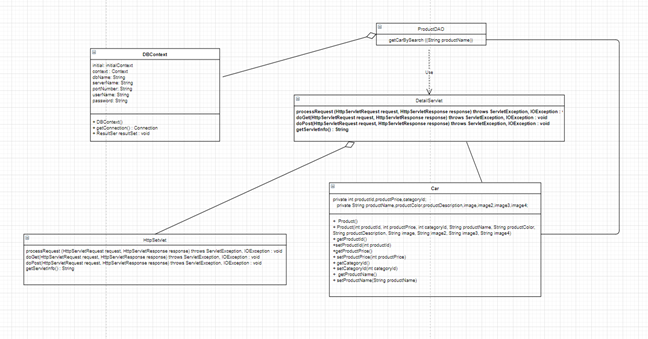
- This function helps customers see product details (product parameters)

- Require function:

• Log in to the website

• view product list

### **c. Class Diagram**



### **d.** **Class Specifications**

#### **ViewCarController**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get id from debtor button and return debtor with that id.* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

#### **CarDAO**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCarById(int id)* | *Interface abstract method view debtor with car id.* |

#### **CarDAOImpl**

***Class Methods***

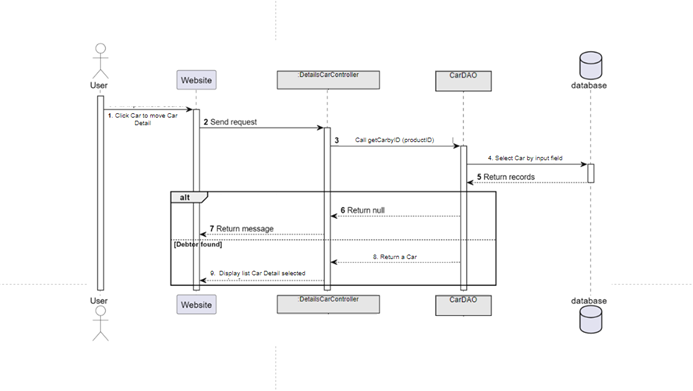
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCarById(int id)* | *Find car by car id.* |

#### **Car**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

### **e.** **Sequence Diagram(s)**



## **9. Update Car**

**a. Screen**

**b. Description**

- This function helps the manager or admin to edit the details of the product as desired

- Require function:

• Log in to the website

• view product list

### **c. Class Diagram**

### **d. Class Specifications**

*UpdateCarController*

*Class Methods*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *Get data from Update Car Form and add it into database* |

#### **DBContext**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getConnection()* | *To connect with database* |
| *02* | *closeResultSet(ResultSet resultSet)* | *To close result set* |
| *03* | *closePreparedStatement(PreparedStatement preparedStatement)* | *To close prepared statement* |
| *04* | *closeConnection(Connection connection)* | *To close connection* |

#### 

***CarDAO***

***Class Methods***

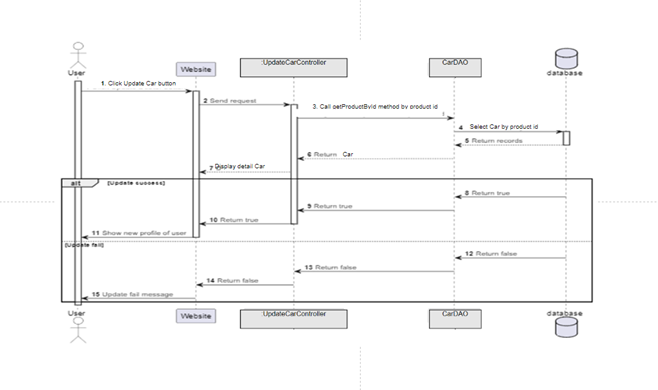
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *add(String fullName, String address, String mobilePhone, String email, int genderId, int userId, float totalDebt, Timestamp createdDate, Timestamp updatedDate)* | *Interface abstract method add debtor with name, address, mobile phone, email, gender and total debt* |
| *02* | *getAllProductByUserId(int userId)* | *Interface abstract method get product by car ID* |
| *03* | *getCarById(int debtorId)* | *Interface abstract method get Car by ID* |

#### **Car**

***Class Methods***

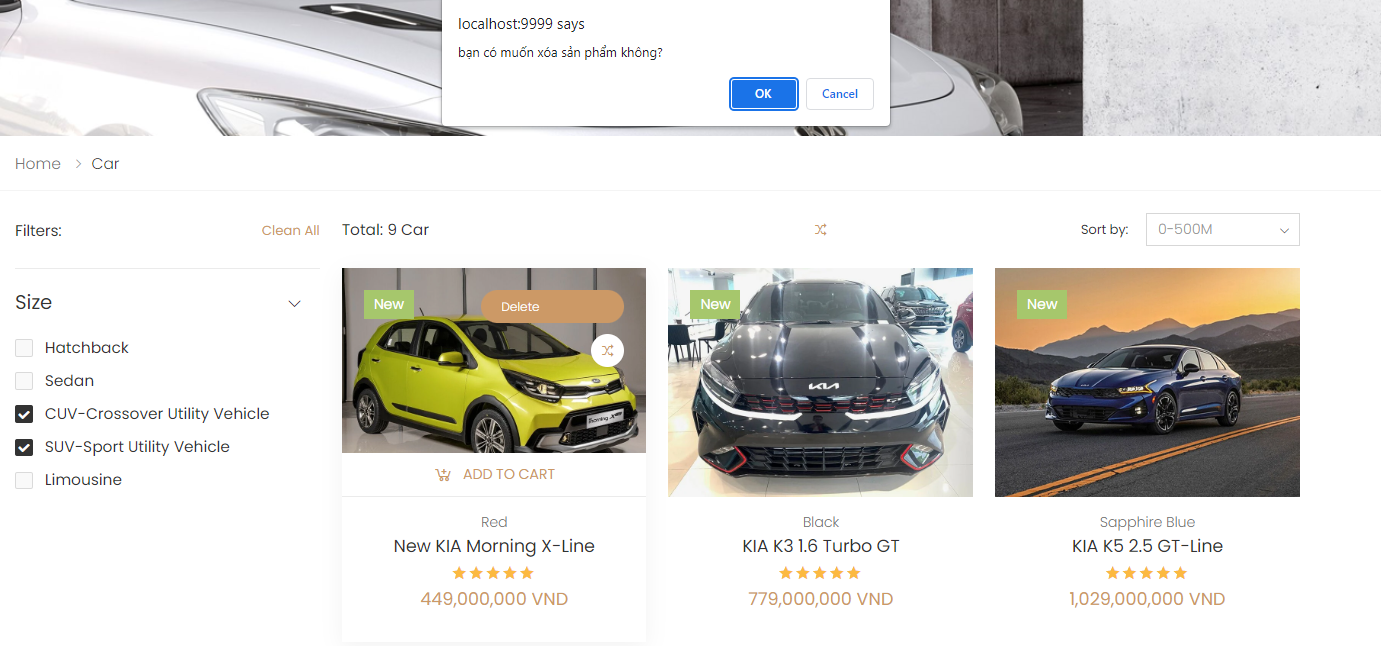
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Constructors* | *Entity initialization* |
| *02* | *Getter* | *Get value of each attributes* |
| *03* | *Setter* | *Set value for each attributes* |

### **e. Sequence Diagram(s)**



## **10.** **Delete Cars**

**a. Screen**

****

**b. Description**

"Delete Car" refers to the action of removing a car from the web page or database of a car dealership or car selling website. This action can be performed by a user who has appropriate access rights, such as an administrator or a seller.

**c. Class diagram**

****

**d. Class Specifications**

DeleteProductController

| No | Method | Description |
| --- | --- | --- |
| 01 | doPost(HttpServletRequest request, HttpServletResponse response) | Get data from Delete Car Form and add it into database |

#### DBContext

Class Methods

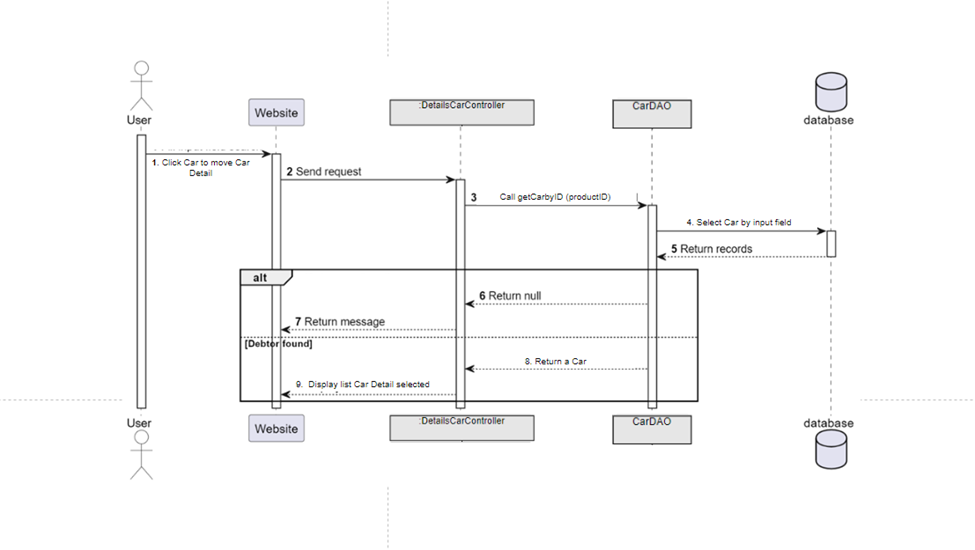
| No | Method | Description |
| --- | --- | --- |
| 01 | getConnection() | To connect with database |
| 02 | closeResultSet(ResultSet resultSet) | To close result set |
| 03 | closePreparedStatement(PreparedStatement preparedStatement) | To close prepared statement |
| 04 | closeConnection(Connection connection) | To close connection |

#### Car

Class Methods

| No | Method | Description |
| --- | --- | --- |
| 01 | Constructors | Entity initialization |
| 02 | Getter | Get value of each attributes |
| 03 | Setter | Set value for each attributes |

**e. Sequence Diagram(s)**

****