GÖTÜR Project

Design

Version 1.2

Prepared By:

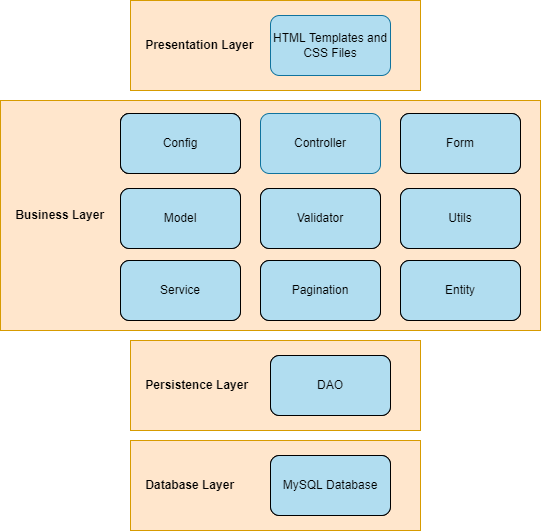
MESE Company

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/04/2022 | 1.0 | Draft 1 | MESE TEAM |
| 20/05/2022 | 1.1 | Review Items Fixed  Iteration 3 Updates | MESE TEAM |
| 11/06/2022 | 1.2 | Iteration 4 Updates | MESE TEAM |

# **Design Structure**

The core of the design is the layered architecture pattern, which is formed by individual layers performing a specific role within the application such as presentation layer, business layer and within each layer application’s components are categorized into. GOTUR application layered architecture consists of presentation, business, persistence and database layers:

* Presentation layer is responsible for handling all user interface and browser communication logic.
* Business layer is responsible for executing specific business rules associated with the request.
* Persistence layer is responsible for serving as an intermediary between the business functions of the application and the data it stores in a database.
* Database layer is responsible for containing the database where the application’s data that need to be saved and retrieved are located.

# **Subsystems**

* Config Package

Responsible for performing web configurations.

* Controller Package

Responsible for handling requests.

* DAO Package

Responsible for saving and retrieving information from the database.

* Entity Package

Responsible for containing persistent domain objects.

* Form Package

Responsible for containing object representations of user interface windows.

* Model Package

Responsible for containing non-persistent, information objects.

* Pagination Package

Responsible for adjusting paginated windows (such as product list window).

* Service Package

Responsible for various service interfaces such as login, payment.

* Utils Package

Responsible for containing utility functions.

* Validator Package

Responsible for user interface validation.

* MySQL Database

Responsible for storing data.

* User Interface (UI)

Responsible for containing visual components.

# **Patterns**

## **Client-Server**

Client-Server approach is followed for interactions between the clients, who are surfing over the GOTUR website and performing some operations and the server, which is handling the requests coming from the clients and update them accordingly. The reason why this approach is utilized is due to the fact that with HTTP request message types the application can understand whether the client is requesting to retrieve, add, delete or update something. The participants are Controllers and Presentation Layer Components and through HTTP protocol these participants interact. There are end-points, i.e., URL paths defined in the controllers and presentation layer interact with the application through these paths in order to send requests. The request sent to an end-point, i.e., URL path is handled by the controllers, processed and then the client is updated accordingly.

## **Model View Controller (MVC)**

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. The Model component corresponds to all the data-related logic that the user works with, the View component is used for all the user interface (UI) logic of the application and the Controller act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. The reason why this approach is utilized is due to the fact that MVC promotes the separation between information, presentation and user interaction. This is achieved through Spring MVC. There are entities, models, controllers and presentation layer components (HTML files) and these are the participants. There is no direct connection from the presentation layer components to the entities and models but through the controllers in the case of which the presentation layer sends a request to an end-point of the controller, controller performs the changes accordingly, then the change done in the model and entities are informed to the presentation layer through the controllers again.

## **Server-Side Rendering**

Server-side rendering is an application's ability to convert HTML files on the server into a fully rendered HTML page for the client. The web browser submits a request for information from the server, which instantly responds by sending a fully rendered page to the client. The reason this approach is utilized is due to the fact that rendering on the server enables more control over responses to different requests in terms of speed, effort and consistency. In the application this is achieved through Thymeleaf and the reason this framework is chosen is due to the fact that it is server-side template engine for Java with built-in support for Spring framework. The participants of this approach are presentation layer components which are Thymeleaf HTML files.

## **Object Relational Mapping (ORM)**

Object-Relational Mapping (ORM) is an approach for converting data between Java objects (for this application) and relational databases. ORM converts data between two incompatible type systems (Java and MySQL for this application), such that each model class becomes a table in our database and each instance a row of the table. The reason why this approach is followed is due to the fact that in the application the data is represented through models/entities and their life-time existence (unless a modification or deletion is done), i.e., persistency is achieved through storage in a database. There is a need for a functionality in order to retrieve the data from the database through the code as well as manipulate the database from the code without any third-party tool and such. To achieve this, a persistence layer is defined in the application which utilizes JPA and Hibernate. These are tools for Object Relation Mapping (ORM) and are integrated with Java. The participants are persistence layer components, i.e., Data-Access Object (DAO) components that are handling the ORM, models and entities that are the representation of the data in the code, MySQL database which stores the data and the controllers which trigger the operations of DAO’s according to the requests sent by the clients.

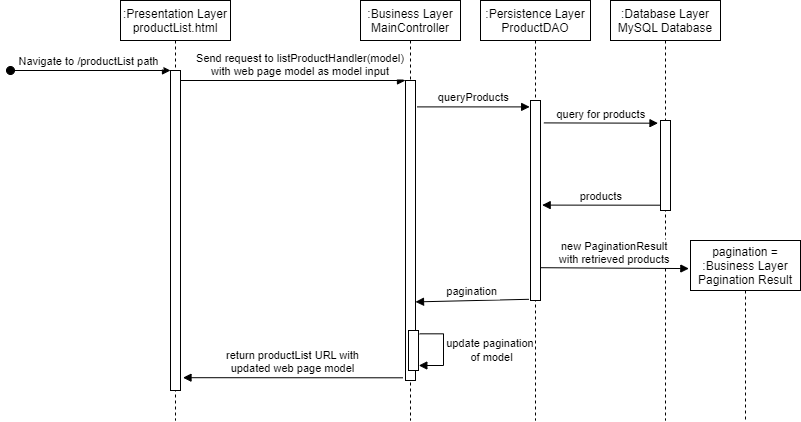
# **Requirement Realizations**

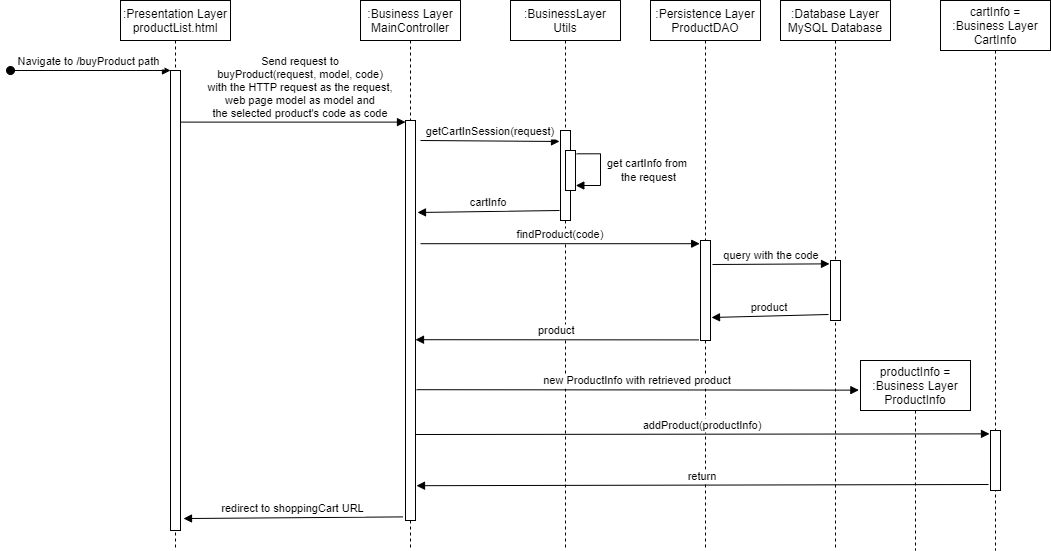
The requirement realizations are described for each use case through UML diagrams. Sequence diagrams are used for describing operations in each use case.

## **Manage Products**

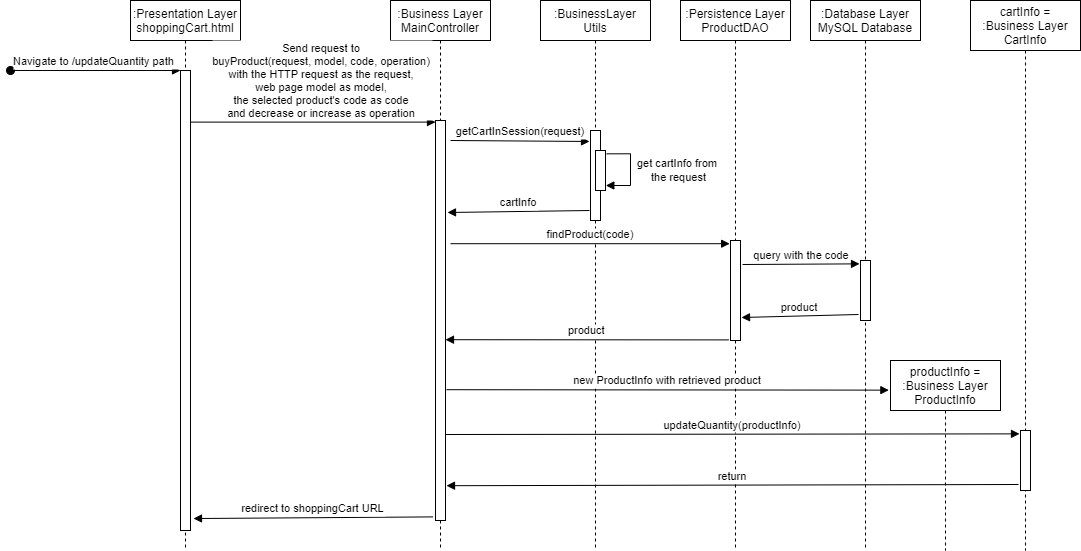
Listing products, adding to cart, updating products added in the cart, removing products from the cart operations are realized in the scope of Manage Products.

Listing products:

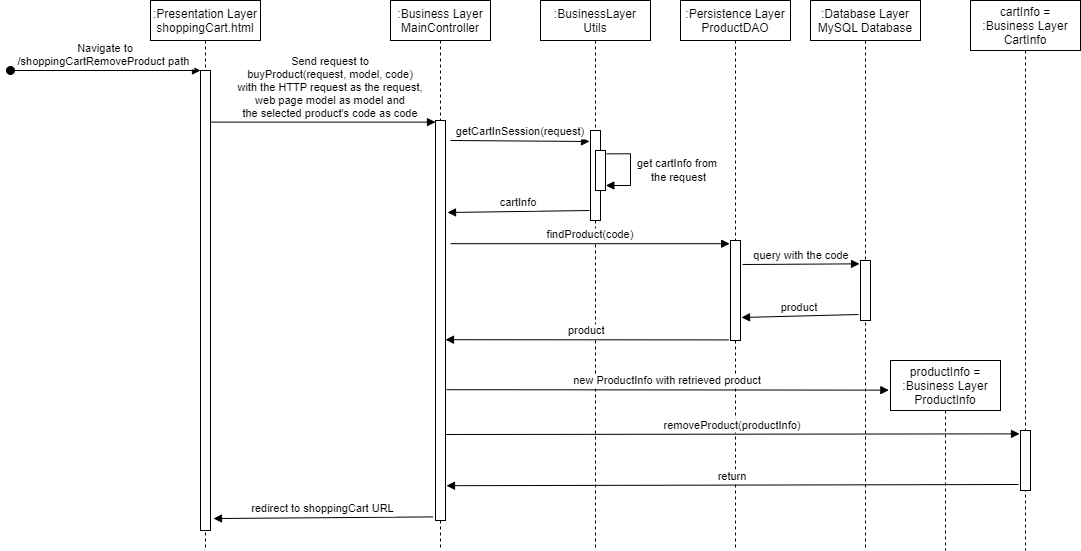


Adding to cart:

Update products added to cart:

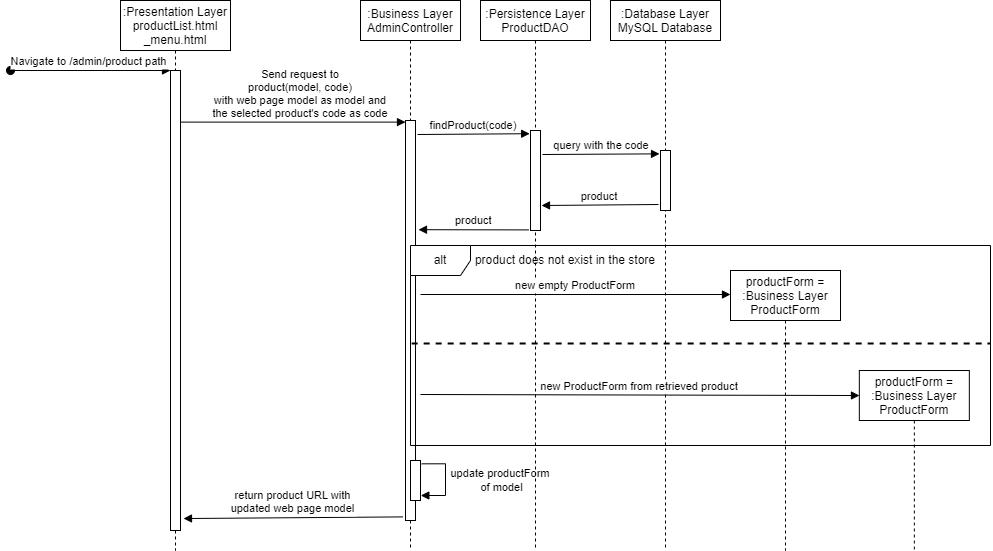


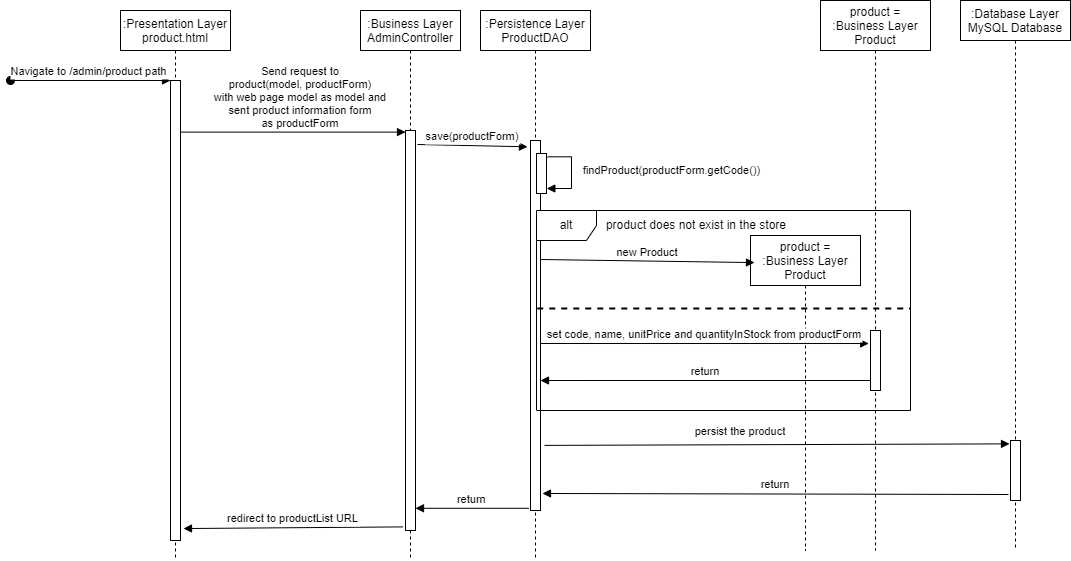
Remove products from the cart:

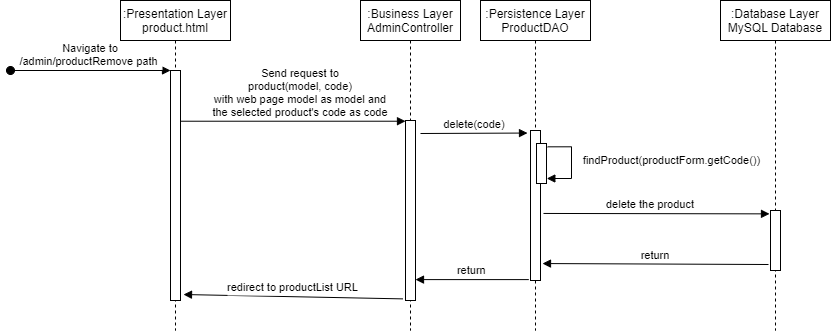


## **Manage Product Specifications**

Retrieve product information/Retrieve form for new product addition, Add new product/Update product information and Remove product from the store requests are realized in the scope of Manage Product Specifications.

Retrieve product information/Retrieve form for new product addition:

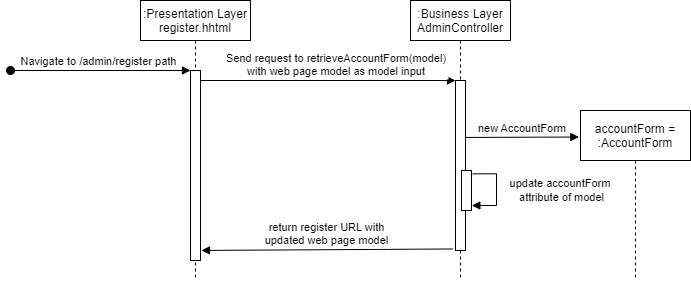
Add new product/Update product information:

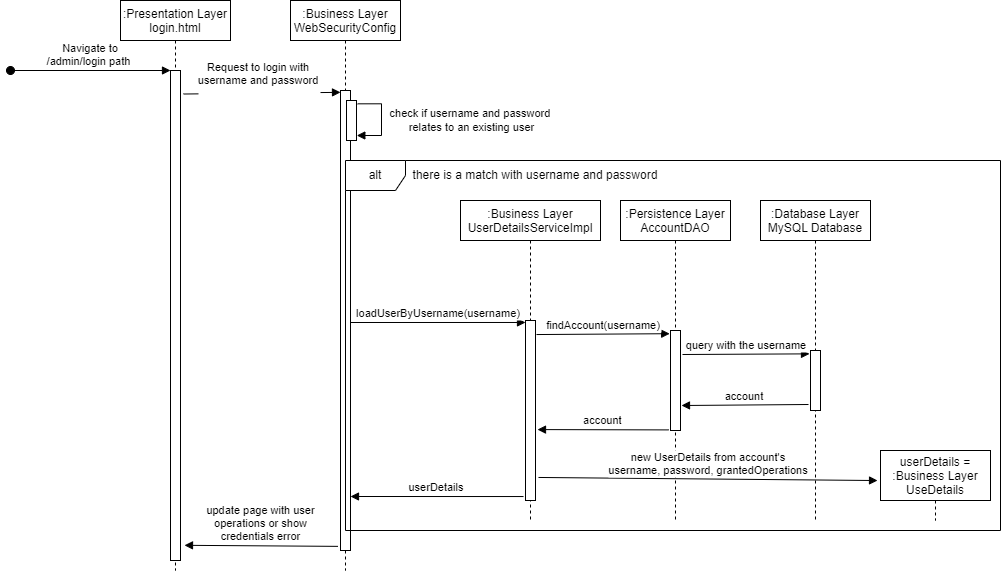
Remove product from the store:

## **Register To System**

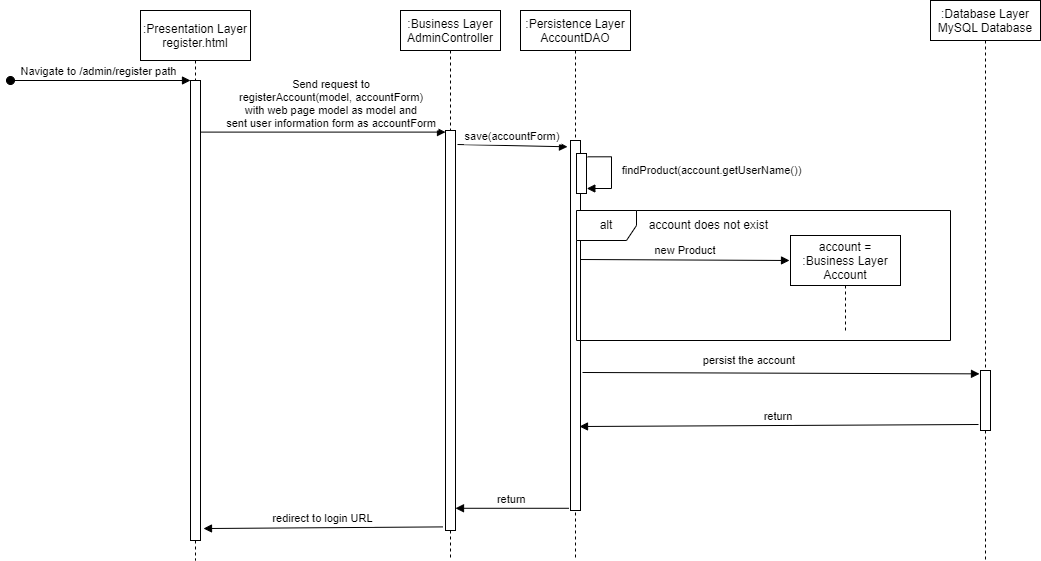
In the scope of Register To System, a user creation request through username, password and role type inputs is handled. Upon a successful input, a user is created and otherwise, the application warns the user of false inputs.

Retrieve form for new user creation:



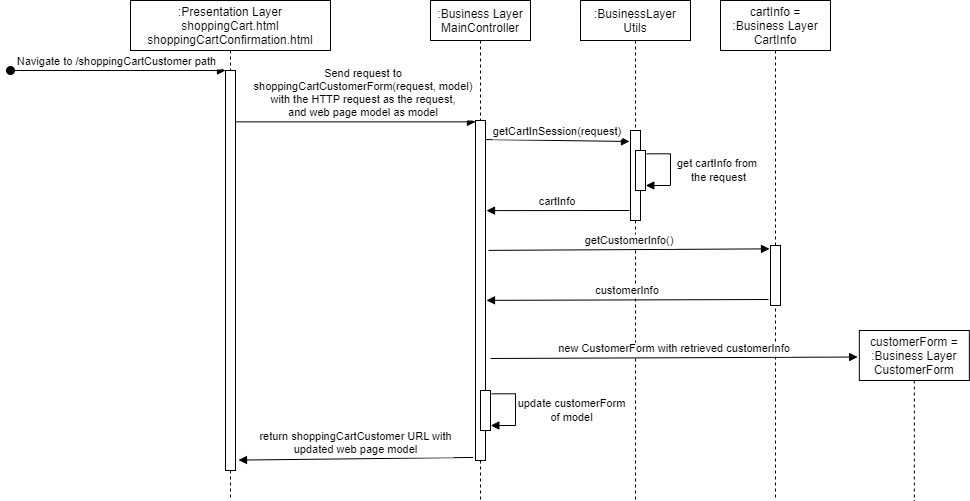
Add user/Register operation:

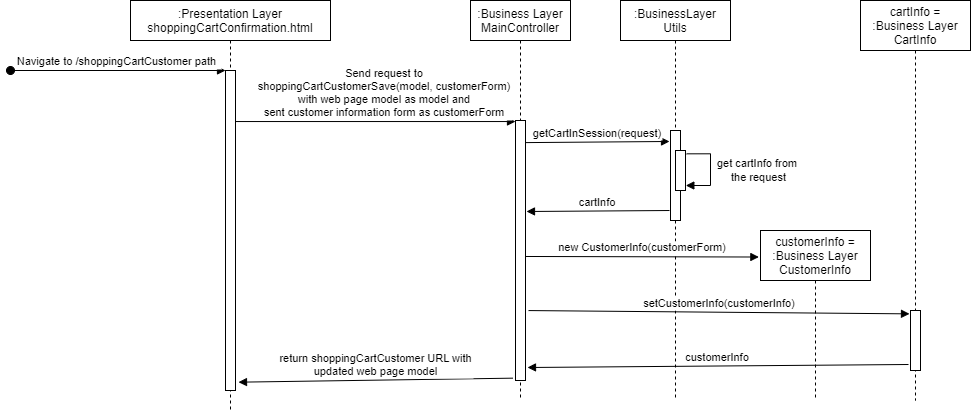
## **Log In To System**

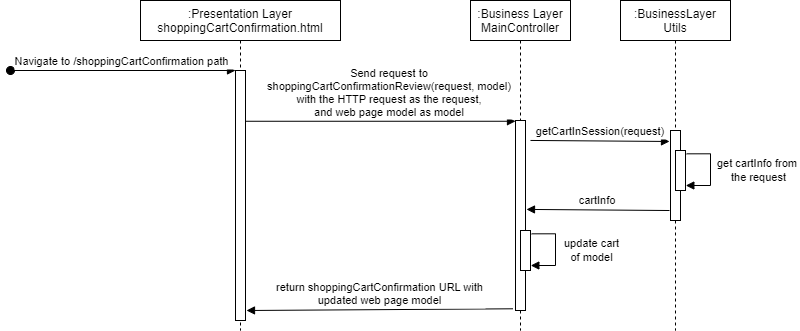
In the scope of Log In To System, a user’s authentication request through his/her username and password is handled. Upon a successful account match, the user logs in and otherwise, the application warns the user of unknown credentials.

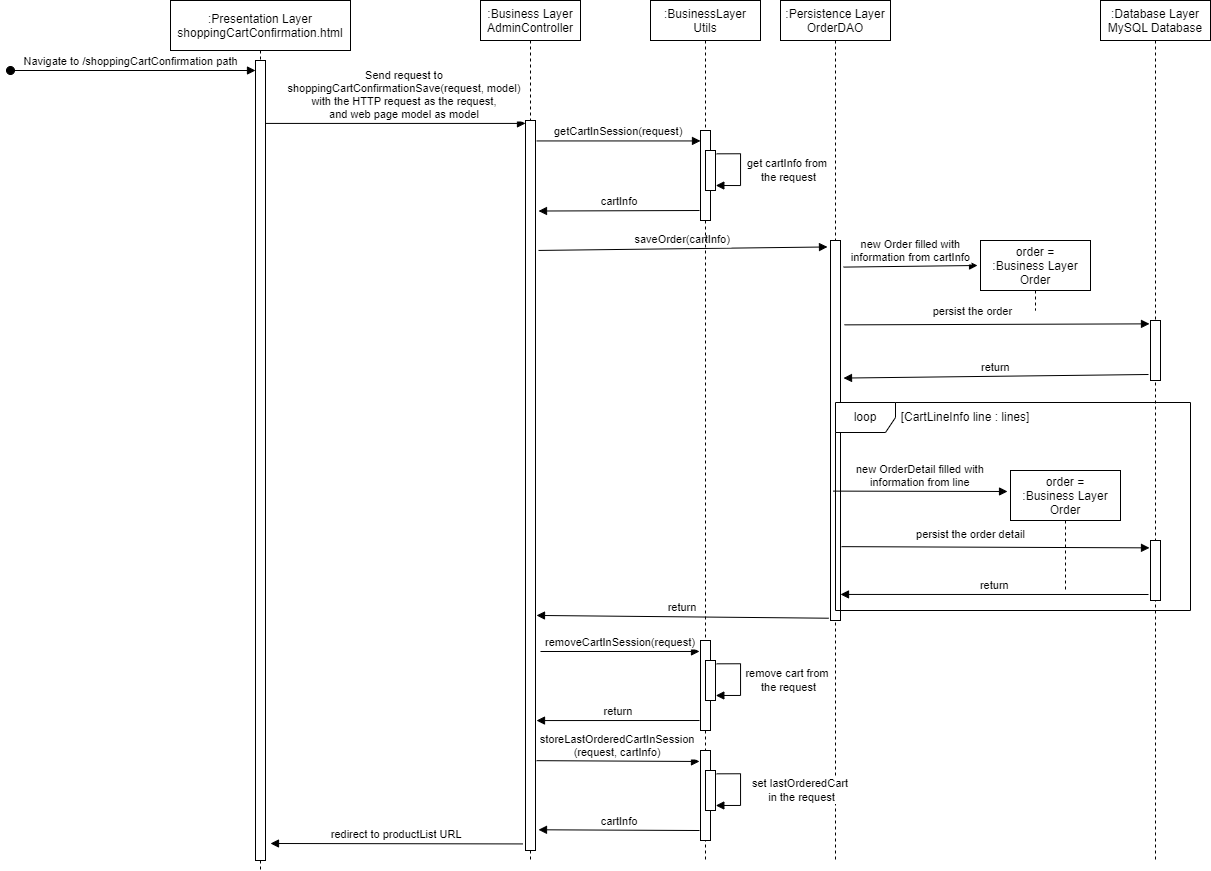
## **Place Order**

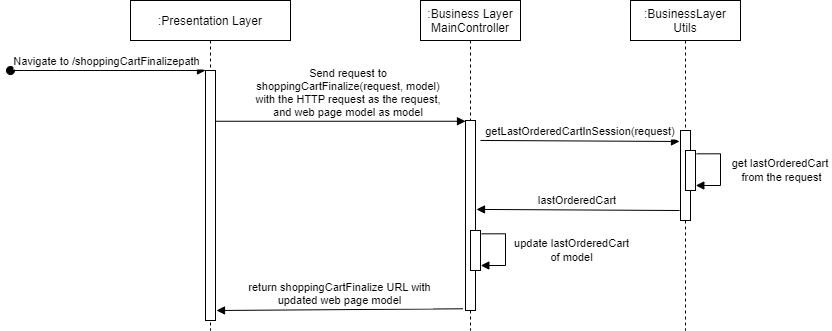
Retrieve customer information/Retrieve form for new customer information, Add new customer information/Update customer information, Retrieve order confirmation, Post order confirmation and Retrieve order finalization are realized in the scope of Place Order.

Retrieve customer information/Retrieve form for new customer information:

Add new customer information/Update customer information:

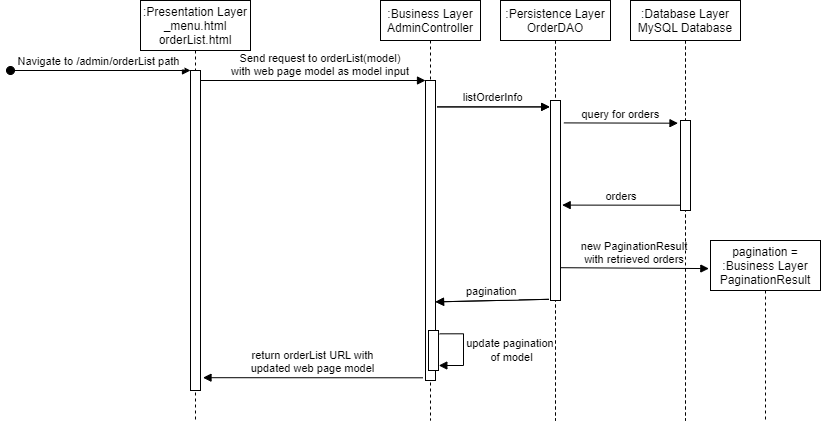
Retrieve order confirmation:

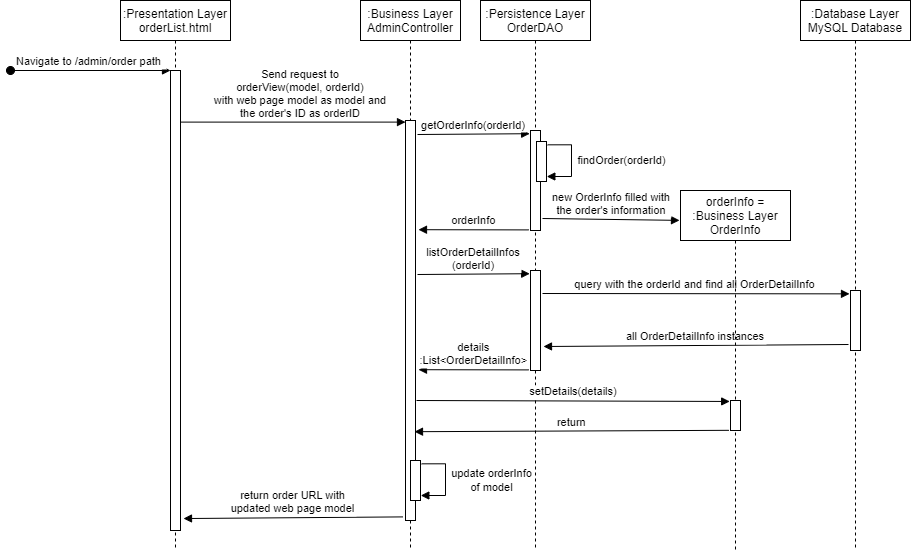
Post order confirmation:

Retrieve order finalization:

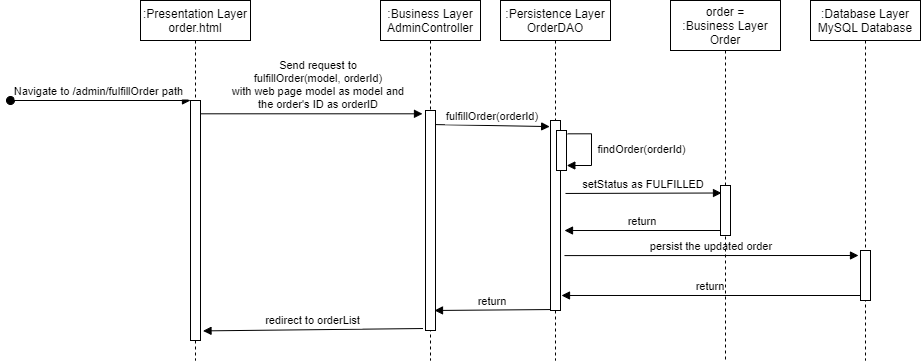
## **View Order Details**

Retrieve order list and Retrieve the details of an order are realized in the scope of Manage Product Specifications.

Retrieve order list:

Retrieve the details of an order:

## **Deliver Order**

In the scope of Deliver Order, a given order is initially marked as ON THE WAY and will be FULFILLED whenever that order’s Fulfill button is pressed from the application.

## **Cancel/Return Order**

In the scope of Cancel/Return Order, a given order can be cancelled by the customer through pressing that order’s Cancel button if it is marked as ON THE WAY and can be returned by the customer through pressing that order’s Return button if it is marked as FULFILLED. The status is then marked as CANCELLED.

