

Online Shoe Sales Website Project "KIA SHOP"

**Software Design Specification**

– Can Tho , March 2024 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
| 2/2/2024 | A | All Team | Database |
| 2/2/2024 | A | All Team | Code Designs |
| 4/2/2024 | M | Nguyễn Thị Ngà | Search Product |
| 4/2/2024 | M | Bùi Ngọc Huy | Add To Cart |
| 5/2/2024 | M | Phan Hoàng Phúc | Login |
| 5/2/2024 | M | Nguyễn Thanh Quang Duy | Order |
| 9/3/2024 | M | Nguyễn Thanh Quan Duy | Cancel Order |
| 9/3/2024 | M | Phan Hoàng Phúc | Forgot Password |
| 10/3/2024 | M | Bùi Ngọc Huy | Update Profile |
| 11/3/1024 | M | Nguyễn Thị Ngà | Add New Shoe |
| 12/3/2024 | M | All Team | Database |
| 13/3/2024 | M | All Team | Code Designs |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 5](#_heading=h.gjdgxs)

[1. Code Packages 5](#_heading=h.30j0zll)

[a. Database Schema 6](#_heading=h.3znysh7)

[b. Table Description 6](#_heading=h.2et92p0)

[II. Code Designs 10](#_heading=h.tyjcwt)

[1. SignUp Account 10](#_heading=h.tyfykhkvy6c)

[a. Class Diagram 10](#_heading=h.n4gkkme5hk5b)

[b. Class Specifications 11](#_heading=h.4fev74jcfjve)

[c. Sequence Diagram(s) 12](#_heading=h.x8zh25nuovar)

[d. Database Queries 13](#_heading=h.lyfli4dldxvt)

[2. Login Account 13](#_heading=h.c0qyvf5u99ll)

[a. Class Diagram 13](#_heading=h.5nb12lnb33di)

[b. Class Specifications 14](#_heading=h.bn1clz6730cp)

[c. Sequence Diagram(s) 15](#_heading=h.pulux0jtsxpd)

[d. Database Queries 16](#_heading=h.jqdoum817zs0)

[3. Forgot Password Account 16](#_heading=h.gyom0oafk7ab)

[a. Class Diagram 16](#_heading=h.lkyhcwg8u53q)

[b. Class Specifications 16](#_heading=h.i2yenei4idp7)

[c. Sequence Diagram(s) 17](#_heading=h.j6lhql4kur8a)

[d. Database Queries 18](#_heading=h.edcmn5ictxuq)

[4. Update Product 18](#_heading=h.c9zy6yixzcjy)

[a. Class Diagram 18](#_heading=h.9476q0x47pcr)

[b. Class Specifications 19](#_heading=h.pyryp4mhii00)

[c. Sequence Diagram(s) 20](#_heading=h.h2kvhfjz3bw3)

[d. Database Queries 20](#_heading=h.z1hrib88kkyt)

[5. Add Product Cart 20](#_heading=h.mam81skxta7e)

[a. Class Diagram 20](#_heading=h.uggel398ziua)

[b. Class Specifications 21](#_heading=h.fd90bp9hr2xk)

[c. Sequence Diagram(s) 22](#_heading=h.mj46xygt5n5s)

[d. Database Queries 22](#_heading=h.75nfbp2ie4y4)

[6. Search Product 23](#_heading=h.dcrrjtq1hgs5)

[a. Class Diagram 23](#_heading=h.811byds90gk3)

[b. Class Specifications 23](#_heading=h.664luhvdbh7n)

[c. Sequence Diagram(s) 24](#_heading=h.e29k3y4ctovs)

[d. Database Queries 25](#_heading=h.kk3zrm77pk3i)

[7.Add Cart Product 25](#_heading=h.r3uco3u9f5xg)

[a. Class Diagram 25](#_heading=h.u5t1ekizalz8)

[b. Class Specifications 25](#_heading=h.lyk537l4z0mc)

[c. Sequence Diagram(s) 26](#_heading=h.8tijo1otd8os)

[d. Database Queries 27](#_heading=h.ne35eft1xv7v)

[8.Update Account User 27](#_heading=h.bprhn7kfsvkg)

[a. Class Diagram 27](#_heading=h.qn6u83ez551u)

[b. Class Specifications 28](#_heading=h.cac63f373xny)

[c. Sequence Diagram(s) 29](#_heading=h.hz7g127okfjf)

[d. Database Queries 29](#_heading=h.282sx4hzv8tw)

[9.Delete Product 29](#_heading=h.8yo05krumljh)

[a. Class Diagram 29](#_heading=h.rj9zlihizcyg)

[b. Class Specifications 30](#_heading=h.79wbdwutzvn3)

[c. Sequence Diagram(s) 31](#_heading=h.3mgijc1y4dkh)

[d. Database Queries 31](#_heading=h.xox147e0s5qj)

[10. Sort Product 31](#_heading=h.staysung1xf)

[a. Class Diagram 31](#_heading=h.e19pmwg9el5m)

[b. Class Specifications 32](#_heading=h.79pmvgkscgke)

[c. Sequence Diagram(s) 33](#_heading=h.7m5j27i62x4b)

[d. Database Queries 33](#_heading=h.n380rrkvuqj8)

[11.Changes Password 33](#_heading=h.dwkhnqj9i3m2)

[a. Class Diagram 33](#_heading=h.rihwqow75ir7)

[b. Class Specifications 34](#_heading=h.v2vrae6ke5rk)

[c. Sequence Diagram(s) 35](#_heading=h.eppuh9tk3klj)

[d. Database Queries 36](#_heading=h.hqssqw580mok)

[12.Sort Customer 36](#_heading=h.7sm775rvyw4i)

[a. Class Diagram 36](#_heading=h.l2kzhlisffvr)

[b. Class Specifications 36](#_heading=h.npcxhu6ett35)

[c. Sequence Diagram(s) 36](#_heading=h.2lyb3yon9hjj)

[d. Database Queries 37](#_heading=h.l3oiamruprnr)

[13.Add New Product 37](#_heading=h.u9soss2blkcr)

[a. Class Diagram 37](#_heading=h.o3cr28le2psu)

[b. Class Specifications 38](#_heading=h.2sz047n12vfb)

[c. Sequence Diagram(s) 39](#_heading=h.sjcm3a25i4h4)

[d. Database Queries 39](#_heading=h.kfqxss9ztf9n)

[14.Order 40](#_heading=h.mdugdb1gb191)

[a. Class Diagram 40](#_heading=h.k9nremr078s5)

[b. Class Specifications 40](#_heading=h.yxb0xrpifzg)

[c. Sequence Diagram(s) 41](#_heading=h.n90afbu5tgwm)

[d. Database Queries 42](#_heading=h.p89f2kh0n0xl)

[15.CancelOrder 43](#_heading=h.9rjw8j2pstfu)

[a. Class Diagram 43](#_heading=h.oxoom9has5k3)

[b. Class Specifications 43](#_heading=h.oz233wulgk4r)

[c. Sequence Diagram(s) 44](#_heading=h.z4wk38q8l9uk)

[d. Database Queries 45](#_heading=h.hop3egdqh9v5)

# I. Overview

## 1. Code Packages

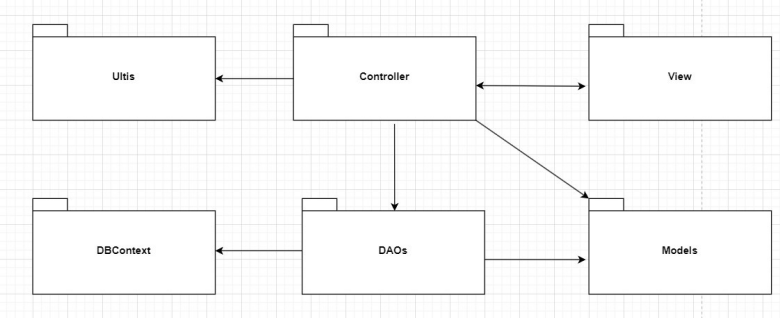


Figure 1. Code Packages

***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | View | This package includes a font-end source with jsp page, css, JavaScript,images,icon,font,bootstrap and jquery library. |
| 02 | Models | This package contains entities(classes) Product,Cart,Oder,User,Staff, etc that can be accessed and used by other classes DAO. |
| 03 | DAOs | Includes class ProductDAO, CartDAO, OderDAO, UserDAO, and etc with detailed functions that can interact directly with the database for data processing to which the Controller can load data for the user. |
| 04 | Controller | Includes Servlet with a responsibility to process, transfer the data collected by the user on web page(view) to DAO and Model in the database to JSP page. |
| 05 | DBContext | This package helps connect database |
| 06 | Ultis | This package contains a Helper class with functions for sending email,random new password, and interacting with users by email. |

2. Database Design

### a. Database Schema

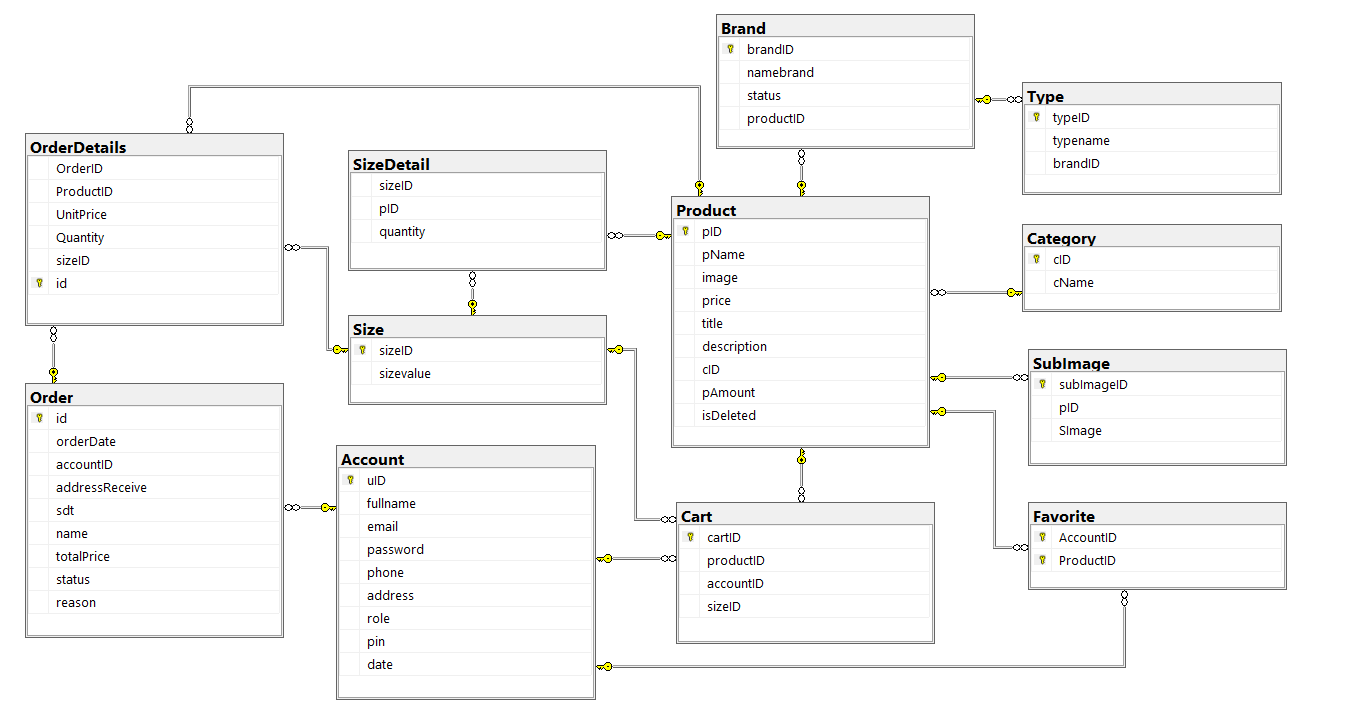


Figure 2.1 Database

### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | brand | Description  Stores information about brands associated with products.  - Primary keys: brandID  - Foreign keys: productID (linked to the product table)  Columns:  - brandID (int): Primary key, unique identifier for each brand.  - productID (int): Foreign key, linking to the product table.  -name (nvarchar(255)): The name of the brand (up to 255 characters).  - status (int): Represents the status of the brand. |
| 02 | product | Description:  - Primary key: pID  - Foreign key: productID (linked to cart table)  Column:  - pID(int): Primary key, unique identifier for each product.  - pName(nvarchar(MAX)): Product name (MAX characters).  - title(nvarchar(MAX)): product title (max characters)  - description (text): Text description of the product.  - cID(int): Product description text.  - image(nvarchar(MAX)): URL or path to product image.  - price (decimal(10,2)): Price of the product.  -amount(int): Amount of products available. |
| 03 | account | Store information about customer accounts.  - Primary key: uID  - Foreign keys: accountID (linked to cart table)  Column:  - uID(int): Primary key, unique identifier for each customer account.  - fullname (nvarchar(255)): Customer name.  - email (nvarchar(255)): Customer's email address.  - password (nvarchar(255)): Password of the account.  - address(nvarchar(255)): address of the account.  - phone(int): phone number of the account.  - role (int): Represents the account's role (eg customer, administrator).  - pin(int): save pin code when creating account  - date(date): save the date the account was created |
| 04 | orderdetail | Store detailed product information in orders.  - Primary key: id  - Foreign key: ProductID (linked to product table), order\_id  - Foreign key: SizeID (linked to Size table), SizeID  Column:  - orderID (int): Primary key, unique identifier for each order item.  - ProductID(int): Foreign key, link to product table.  - orderID(int): Foreign key, link to order table.  - quantity (int): Number of products in the order.  - sizeID(int): shoe size. |
| 05 | cart | Stores information about items in a customer's shopping cart.  - Primary key: cartID  - Foreign keys: ProductID (linked to product table), orderID,  - Foreign key: SizeID (linked to Size table), SizeID  - Foreign key: accountID (linked to Account table), accountID  Column:  - cartID(int): Primary key, unique identifier for each item in the cart.  - ProductID(int): Foreign key, link to product table.  - orderID(int): Foreign key, link to order table.  - accountID(int): Foreign key, associated with the customer account. |
| 06 | order | Store information about customer orders.  - Primary key: id  - Foreign key: AccountID (linked to account table)  Column:  - id(int): Primary key, unique identifier for each order.  - accountID(int): Foreign key, associated with the customer account.  - orderDate(datetime): Date and time of order.  - TotalPrice(decimal(10,2)): Total order value.  - status(int): Represents the status of the order.  - addressReceive(nvarchar(255)): Delivery address for the order.  - phone (int): Contact phone number for ordering.  -reason(nvarchar(MAX)): reason for canceling the order |
| 07 | size | Stores information about the product's size  - Primary key: sizeID  Column:  - sizeID(int): Primary key, identifier for size.  - sizeValue(int): stores the value of size. |
| 08 | Favorite | Stores favorite products  - Primary key: AccountID.  - foreign key: ProductID  Column:  - AccountID(int): Primary key, identification for the customer.  - Product(int): subkey identifying the favorite product. |
| 09 | subImage | Stores favorite products  - Primary key: subImageID  - foreign key: pID, associated with key pID of the Product table  Column:  - subImageID(int): Primary key, identifier for the image link.  - pID(int): subkey identifying the image of any product. |
| 10 | Category | Stores the product's category  - Primary key: cID  - foreign key: cName, linked to key pName of Product table  Column:  - cID(int): Primary key, identifier for the product type.  - cName(int): subkey identifies any name of that type. |
| 11 | Type | Stores type of product  - Primary key: typeID  -foreign key: bandID  Column:  -typeID (int): Primary key  -typeName(nvarchar(255)): name type of product  -bandID(int): Foreign key |

# II. Code Designs

## 1. SignUp Account

### a. Class Diagram

***Class SignUp***

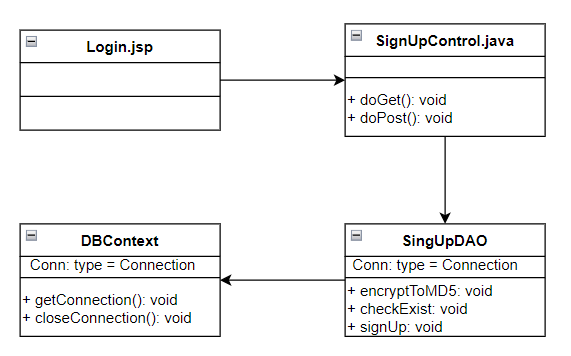


Figure 1.0 Class Diagram SignUp Function

### b. Class Specifications

SignUpController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's SignUp() method to update the information into the database. If the process is successful, it will guide the User to the Pin.jsp page to continue verification and login |

SignUpDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | encryptToMD5(): | Hashes the password the user enters |
| 02 | CheckExit(): | Check the entered email is already in the database |
| 03 | signup(): | Save account information to database |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

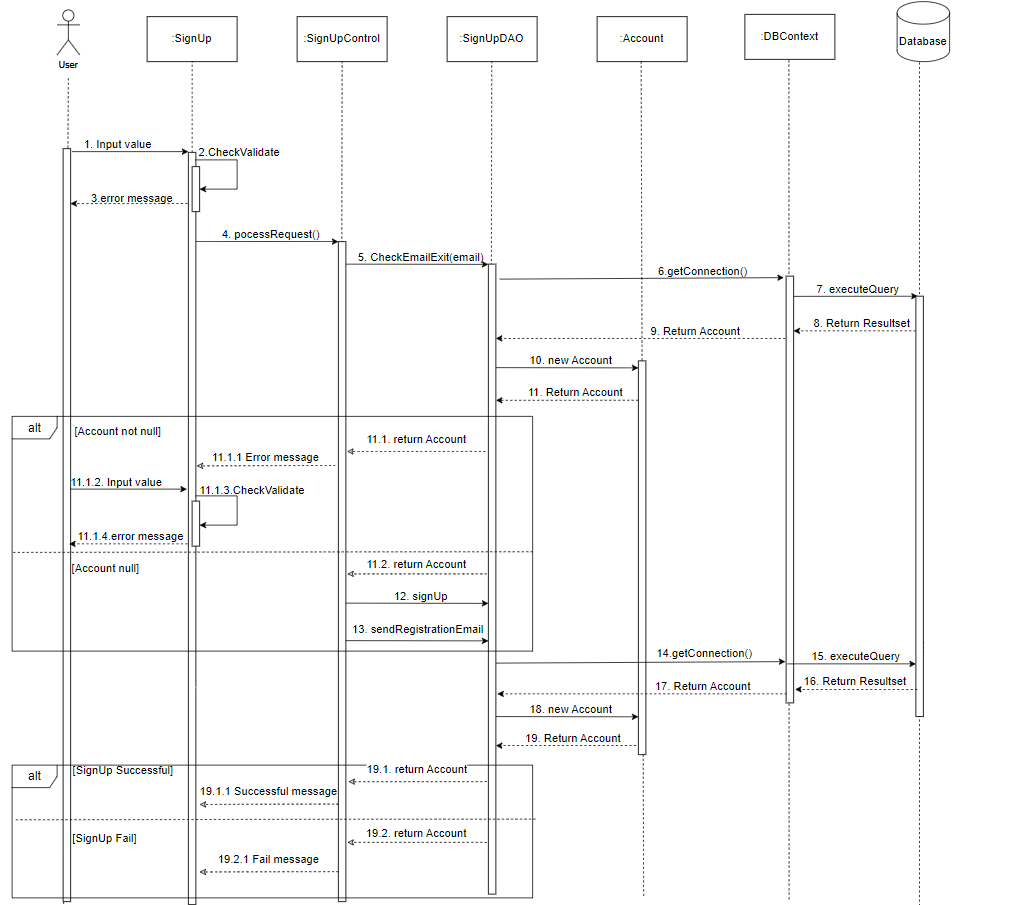


Figure 1.1 Sequence Diagram Sign Up

### d. Database Queries

insert into Account values(?,?,?, null, null,0,?,CURRENT\_TIMESTAMP)

## 2. Login Account

### a. Class Diagram

***Class Login***

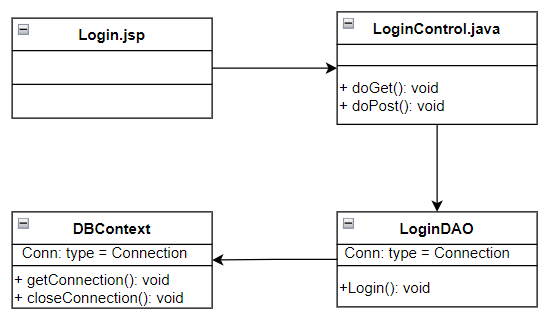
******

Figure 2.0 Class Diagram Login Function

### b. Class Specifications

SignUpController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's Login() method to check if the account is correct, if so, switch to the Shop.jsp page. |

LoginDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | Login(): | Login to the created account |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

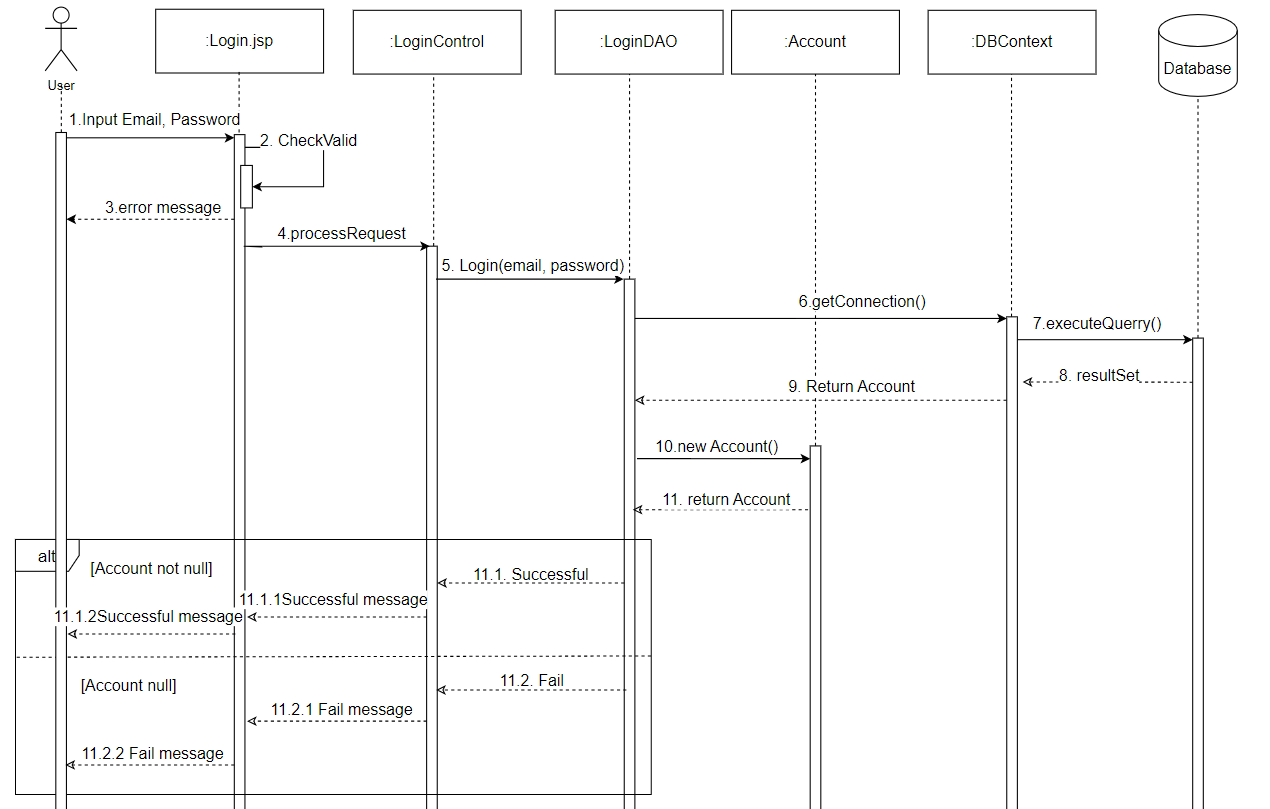
**

Figure 2.1 Sequence Diagram Login

### d. Database Queries

select \* from Account where email = ? and password = ?";

## 3. Forgot Password Account

### a. Class Diagram

***Class Forgot Password***

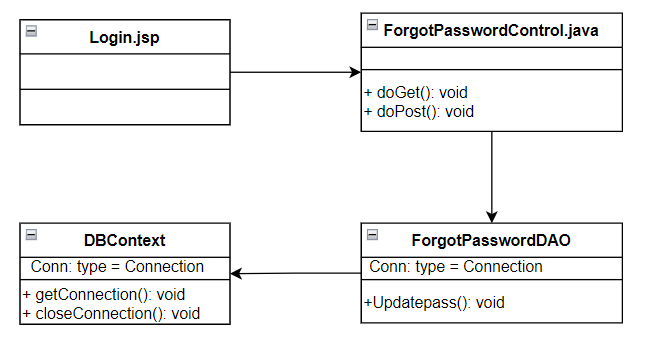


Figure 3.0 Class Diagram Forgot Password Function

### b. Class Specifications

ForgotPasswordController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's Updatepass() method to update the new password and pass it to the Pin.jsp page |

ForgotPasswordDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | Updatepass(): | Update your password |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

### 

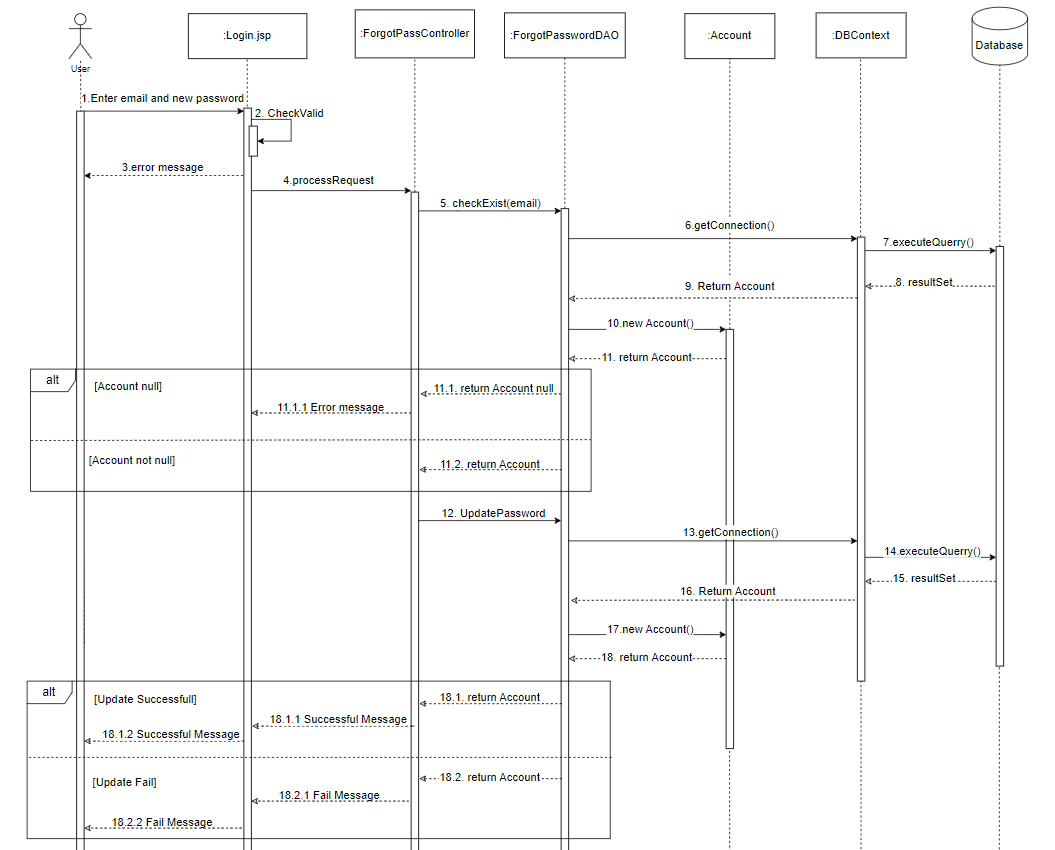


Figure 3.1 Sequence Diagram Forgot Password

### d. Database Queries

UPDATE Account SET password = ?, pin = ? WHERE email = ?

## 4. Update Product

### a. Class Diagram

***Class Update Product***

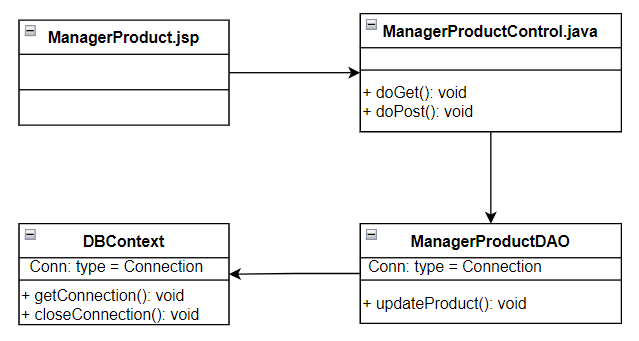


Figure 4.0 Class Diagram Update Product Function

### b. Class Specifications

UpdateProductController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's UpdateProduct() method to update the information it to the ManagerProduct.jsp page |

UpdateProductDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateProduct(): | Update product information quantity |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

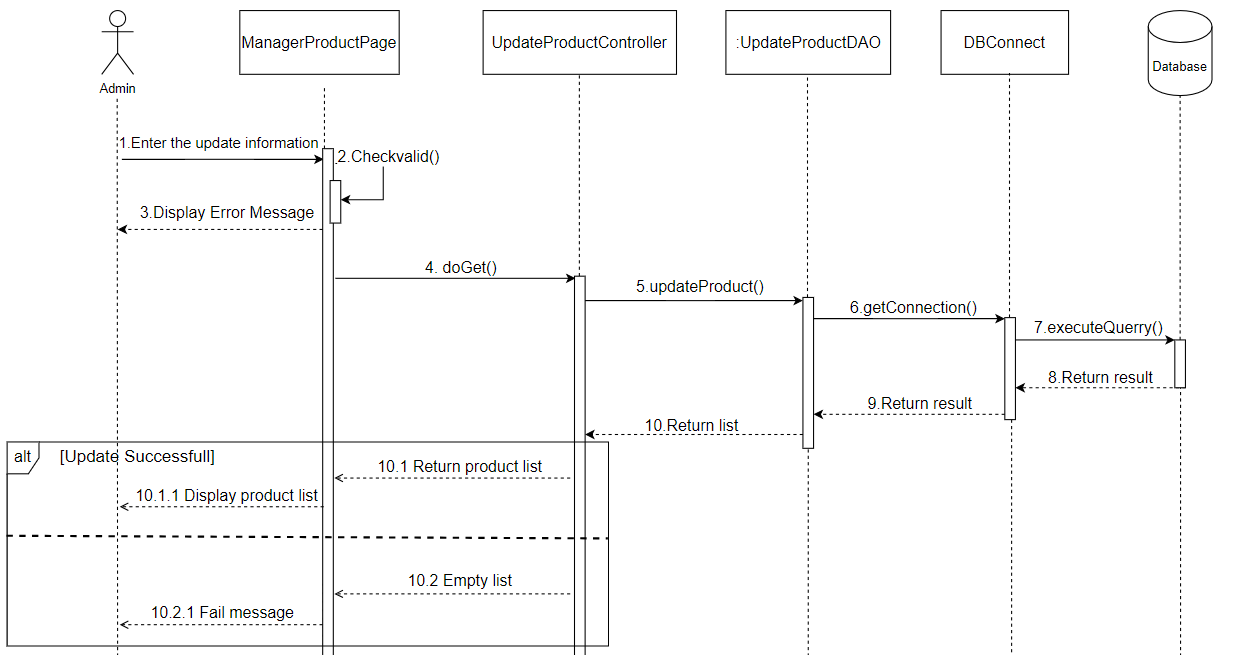


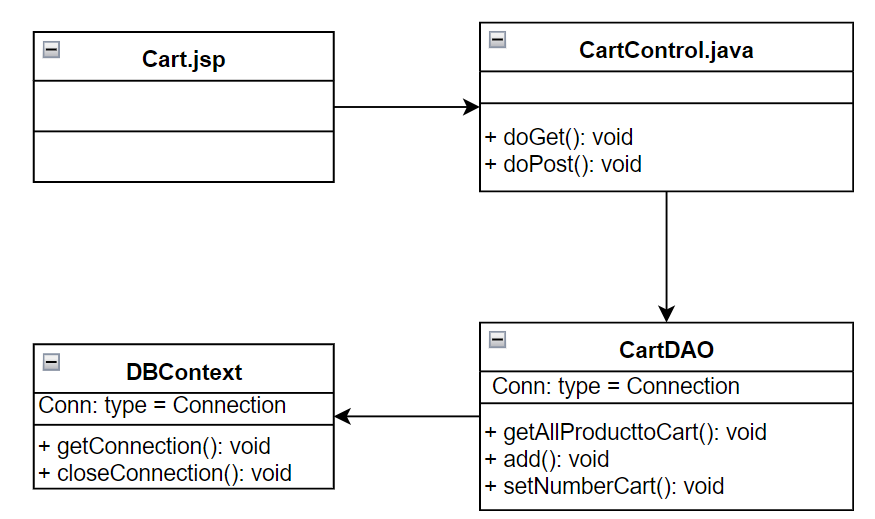
Figure 4.1 Sequence Diagram Update Product

### d. Database Queries

UPDATE Product set pName = ?, image = ?, price = ?, title = ?, description = ?, cID = ?, pAmount = ? where pID = ?

## 5. Add Product Cart

### a. Class Diagram



### b. Class Specifications

CartControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call CartDAO's getAllProductToCart() and add(Product) methods to update the information to the Cart.jsp page |

CartDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAddProductToCart(): | Show all products in the cart |
| 02 | add(Product): | add product to cart |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |
| 03 | setNumberCart() | Update the quantity of the product |

### c. Sequence Diagram(s)

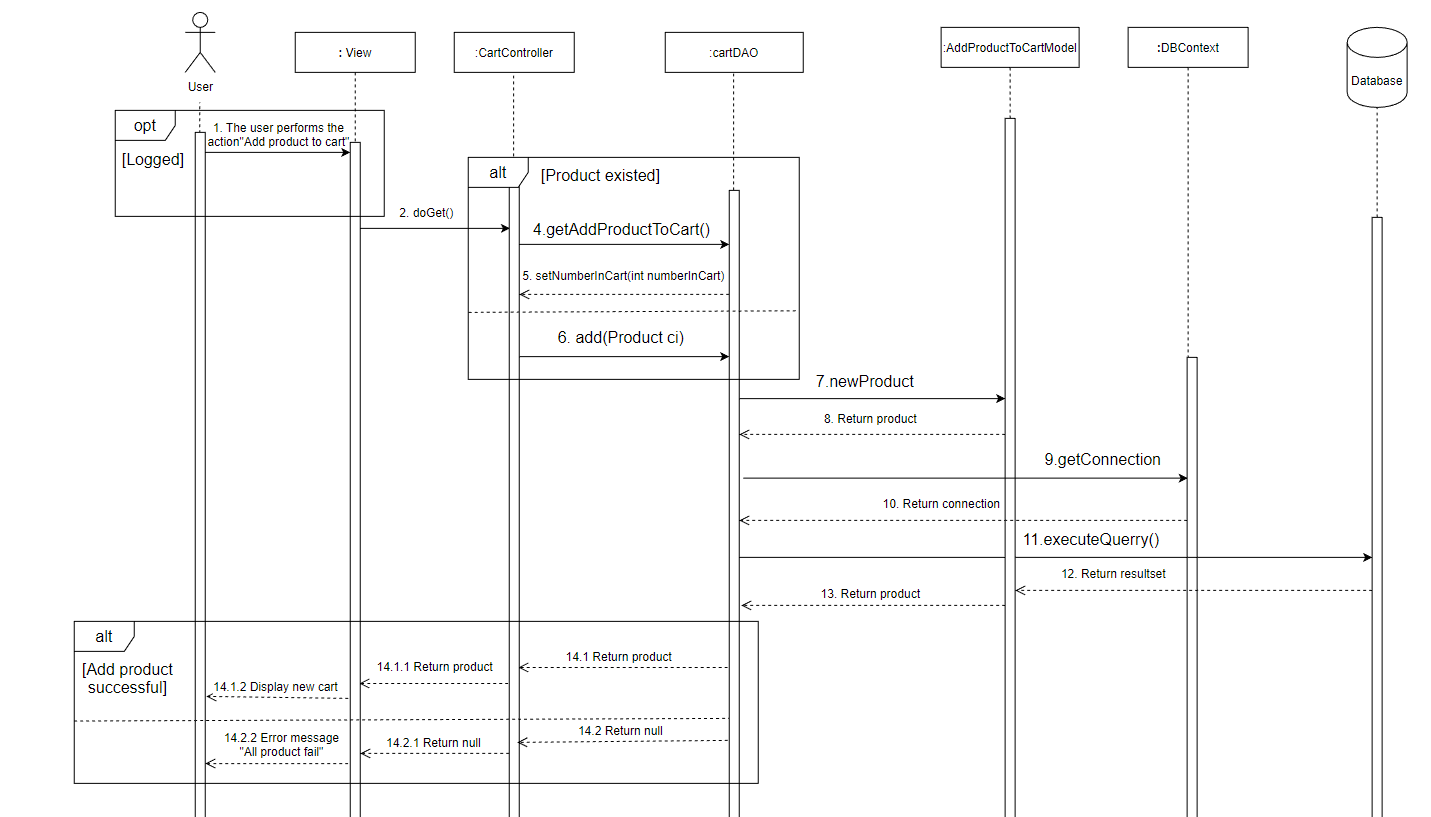


Figure 5.1 Sequence Diagram Add new product

### d. Database Queries

INSERT into Product (pName, [image], price, title, [description], cID, pAmount, isDeleted) VALUES (?, ?, ?, ?, ?, ?, ?, 0)

## 6. Search Product

### a. Class Diagram

***Class Search Product***

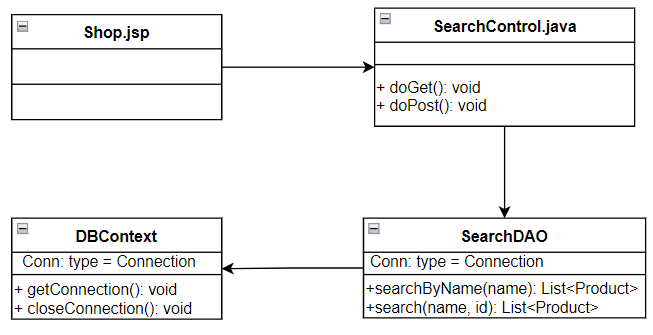


Figure 6.0 Class Diagram Search Product

### b. Class Specifications

SearchControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's search() and searchByName() methods to search products by name or by id |

SearchDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchByName(): | Search products by name |
| 02 | search(): | Search products by name and ID |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

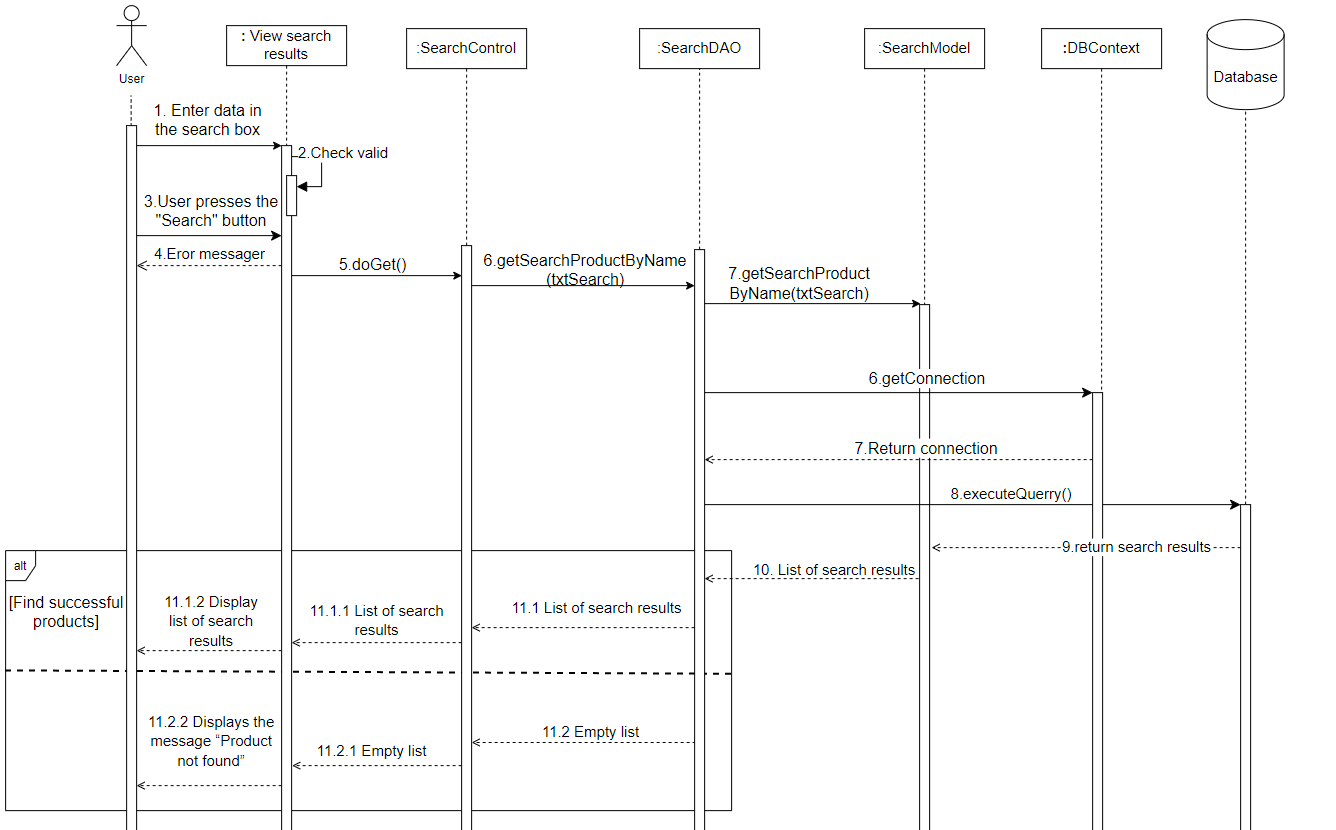


Figure 6.1 Sequence Diagram Search product

### d. Database Queries

SELECT \* FROM Product where pName like ?

SELECT \* FROM Product WHERE isDeleted != 1

## 7.Add Cart Product

### a. Class Diagram

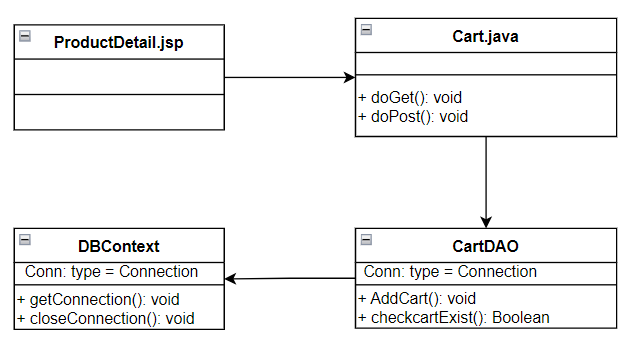


Figure 7.0 Class Diagram Search Product

### b. Class Specifications

CartControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's AddCart() and checkcartExits() methods to add the product to the cart and check if it exists in the cart. |

CartDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | AddCart(): | Add product to cart |
| 02 | CheckCartExits(): | Check the products in the cart |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

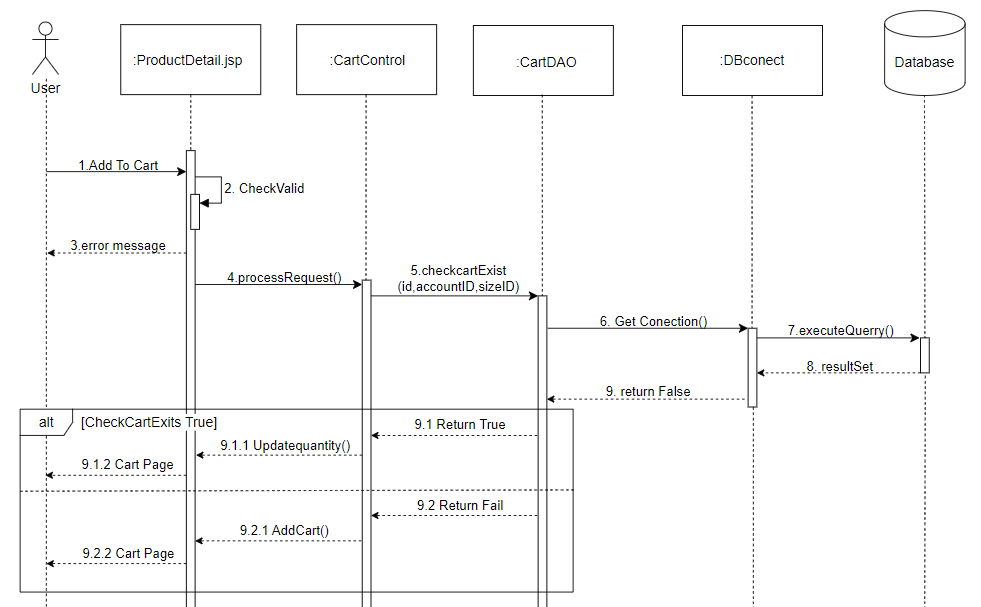


Figure 7.1 Sequence Diagram Add Cart Product

### d. Database Queries

SELECT \* FROM [Cart] WHERE productID = ? AND accountID = ? AND sizeID = ?

INSERT INTO [Cart](productID, accountID, sizeID, quantity) VALUES (?, ?, ?, 1)

## 8.Update Account User

### a. Class Diagram

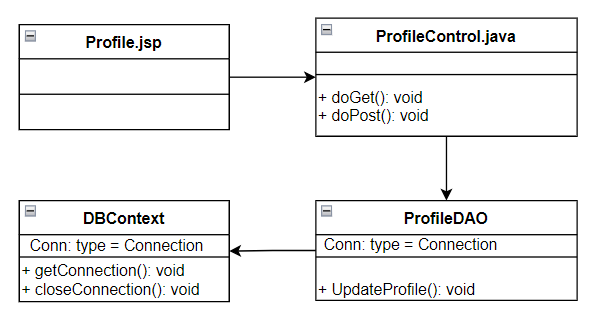


Figure 8.0 Class Diagram Update Profile

### b. Class Specifications

ProfileControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's updateProfile() methods to update profile user’s. |

ProfileDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateProfile(): | update profile user’s. |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### 

### c. Sequence Diagram(s)

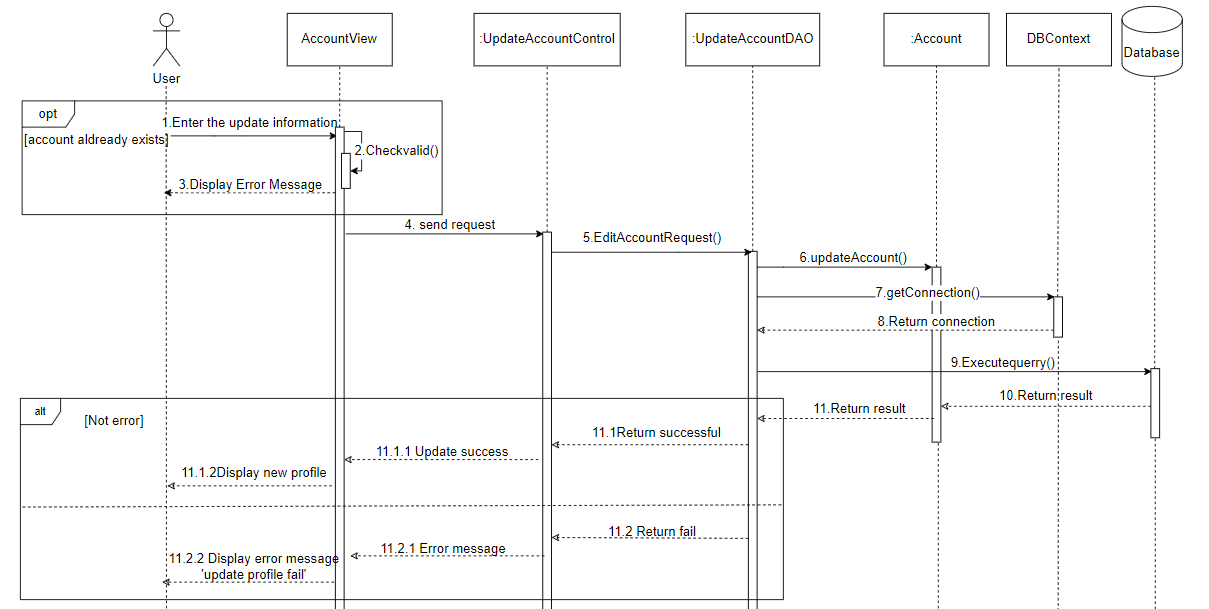


Figure 8.1 Sequence Diagram Update Account User

### d. Database Queries

update Account set fullname = ?, email = ?, phone = ?, address= ? where uID = ?"

## 9.Delete Product

### a. Class Diagram

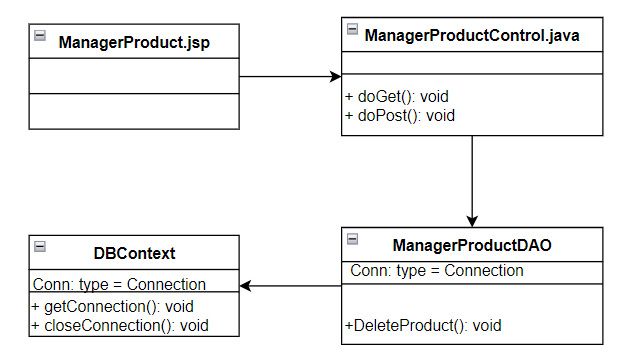


Figure 9.0 Class Diagram Delete Product

### b. Class Specifications

AdminDeleteProductControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's deleteProduct () methods to delete product. |

ManagerProductDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteProduct(): | delete product. |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

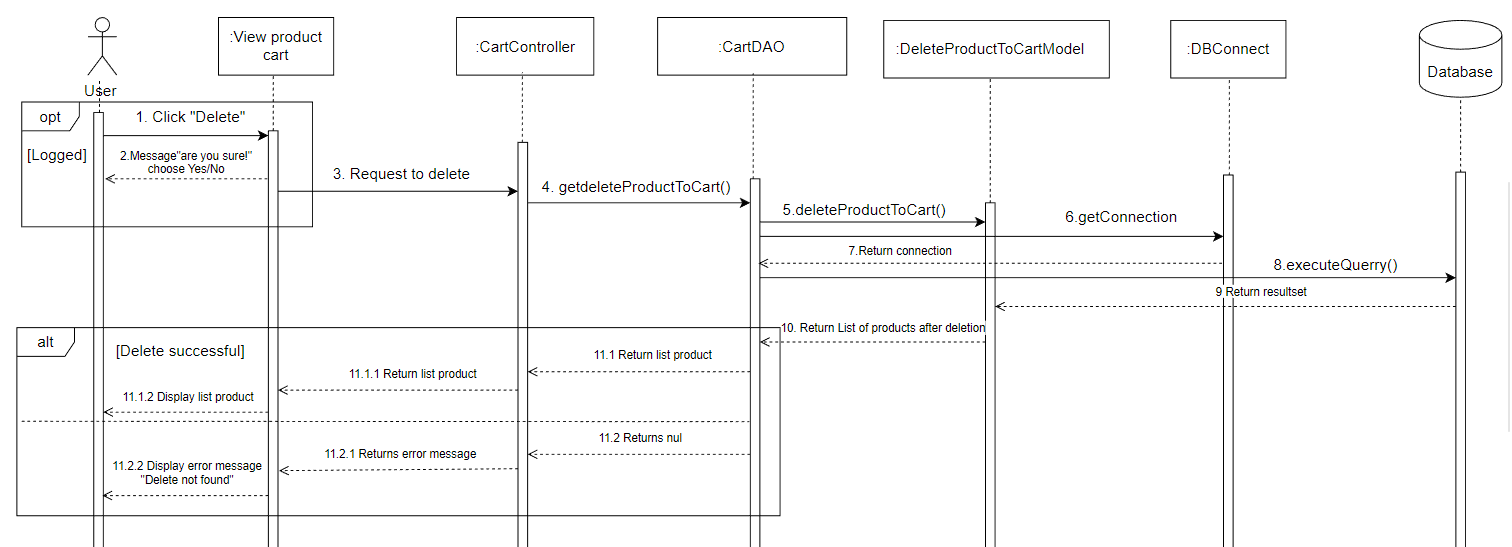


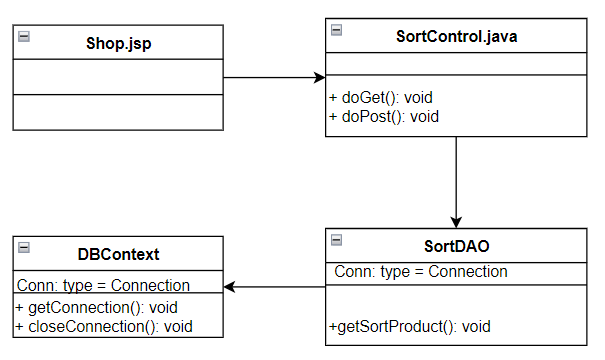
Figure 9.1 Sequence Diagram Delete product

### d. Database Queries

Update Product set isDeleted = 1 WHERE pID = ?

## 10. Sort Product

### a. Class Diagram



### b. Class Specifications

SortControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call SortDAO's getSortProduct() method to update the information to the Shop.jsp page |

SortDAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getSortProduct(): | Sort products as desired |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### c. Sequence Diagram(s)

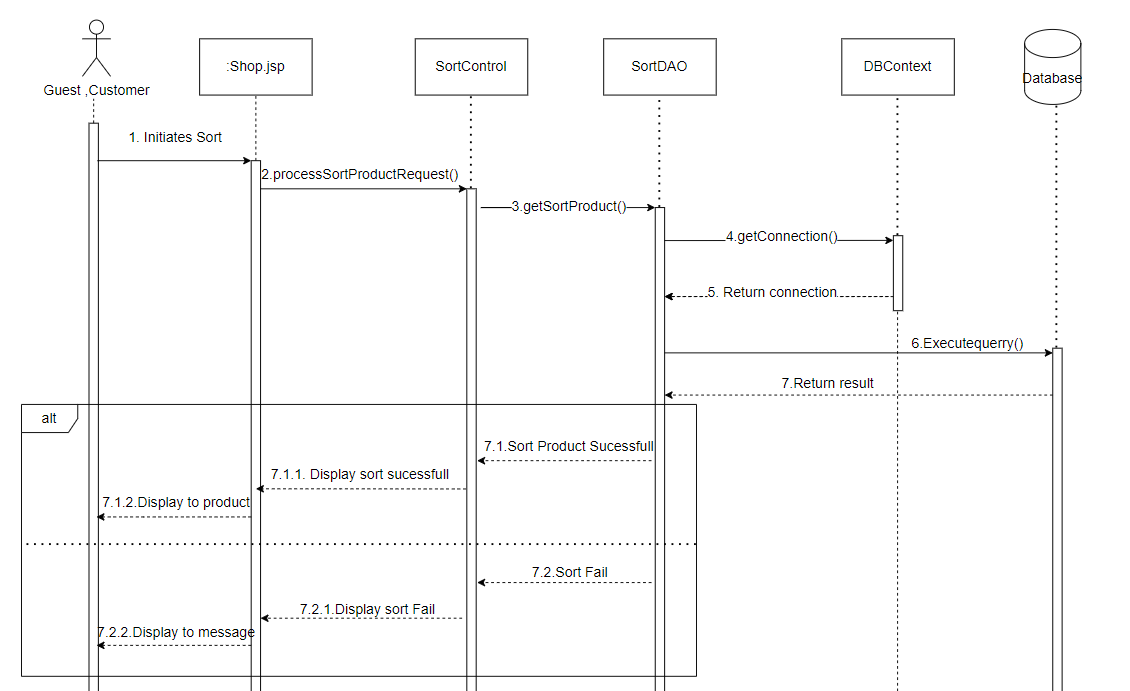


Figure 10.1 Sequence Diagram Sort Product

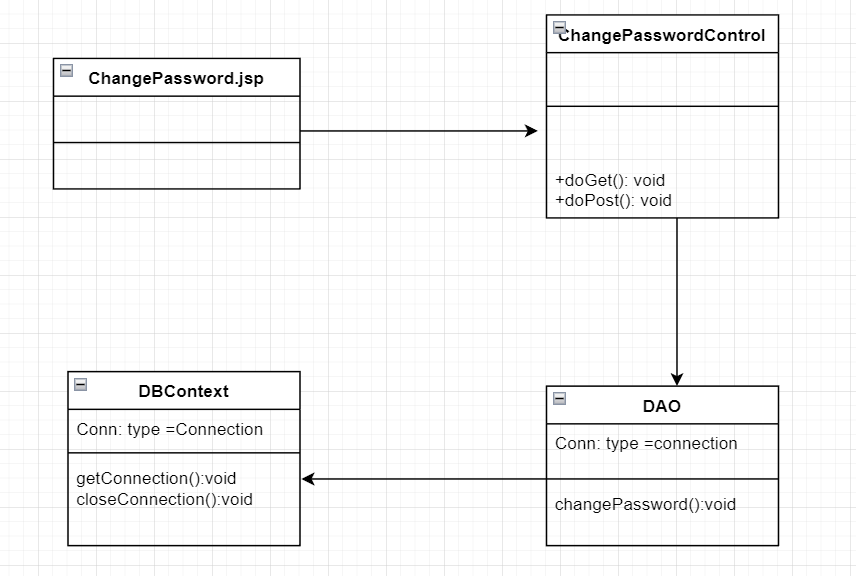
### d. Database Queries

select \* from Category

select \* from Product where isDeleted != 1

## 11.Changes Password

### a. Class Diagram



### b. Class Specifications

ChangePasswordControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's updatePassword() methods to change user’s password. |

DAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | ChangePassword() | Change user’s password. |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### 

### c. Sequence Diagram(s)

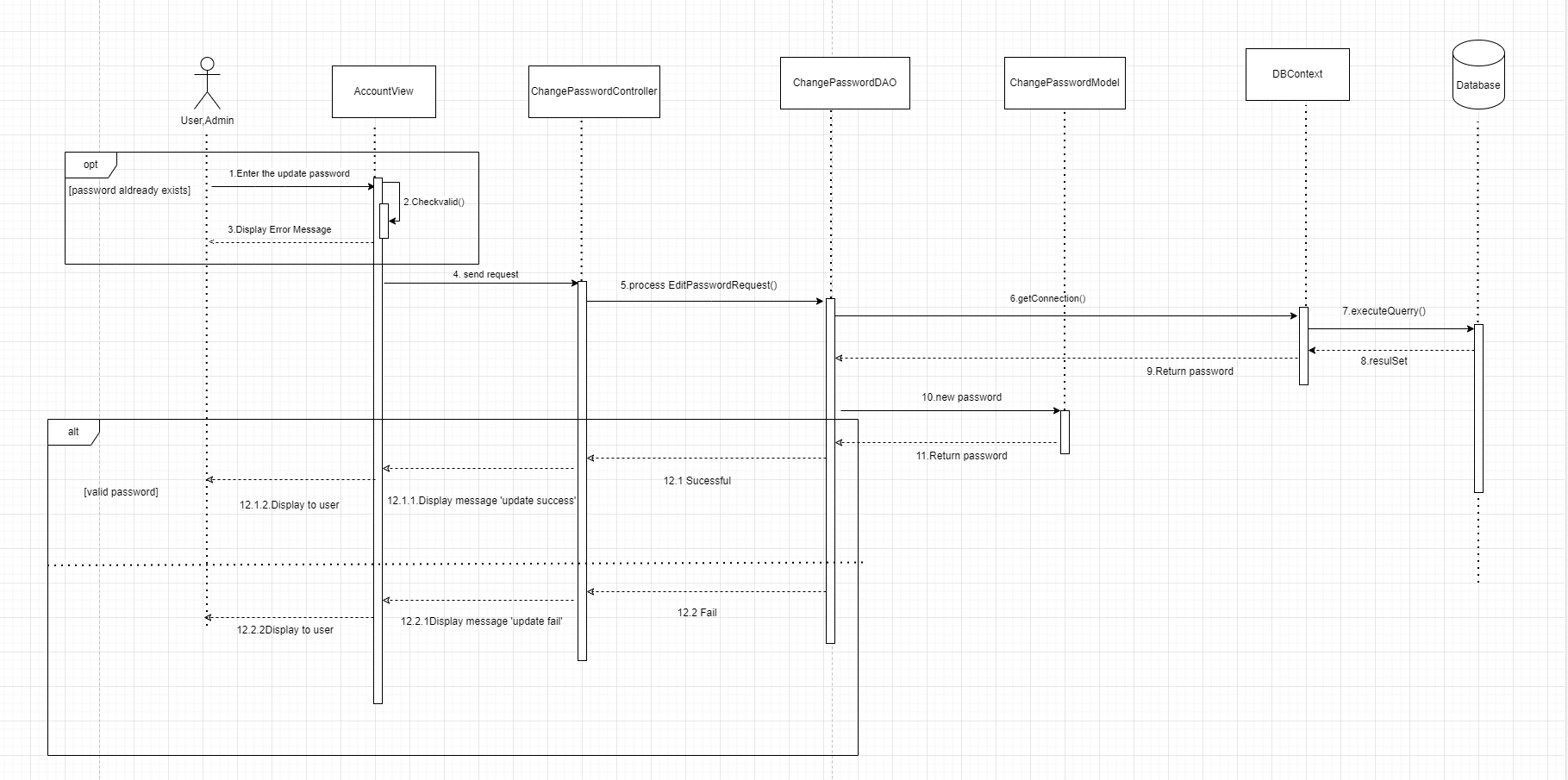


Figure 11.1 Sequence Diagram Change Password

### d. Database Queries

"update Account\n"

+ "set password = ? where uID = ?"

## 

## 12.Sort Customer

### a. Class Diagram

### b. Class Specifications

### c. Sequence Diagram(s)

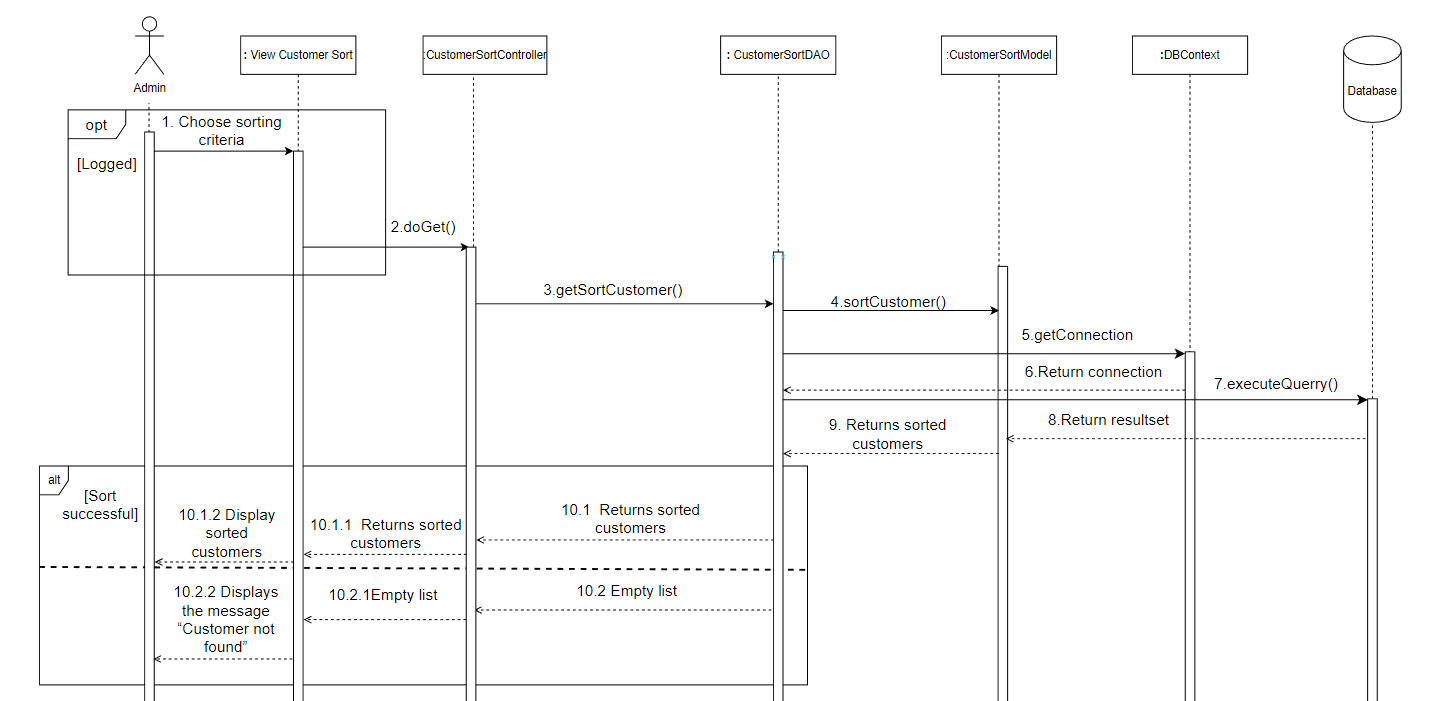


Figure 12.1 Sequence Diagram Sort customer

### d. Database Queries

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 13.Add New Product

### a. Class Diagram

***Class Add New Product***

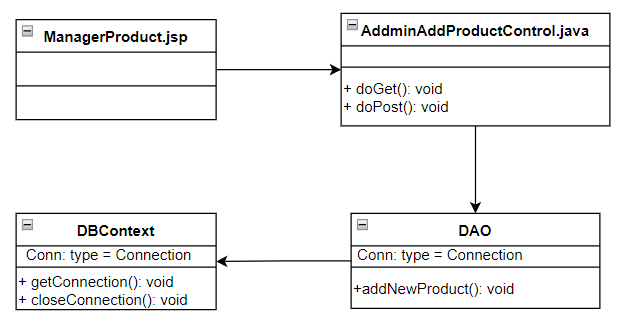


Figure 13.0 Class Diagram Update Product Function

### b. Class Specifications

AdminAddProductControl:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's addNewProduct() method to add the product to the system and pass it to ManagerProduct.jsp page |

DAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addNewProduct(): | Add new product information in web |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### 

### c. Sequence Diagram(s)

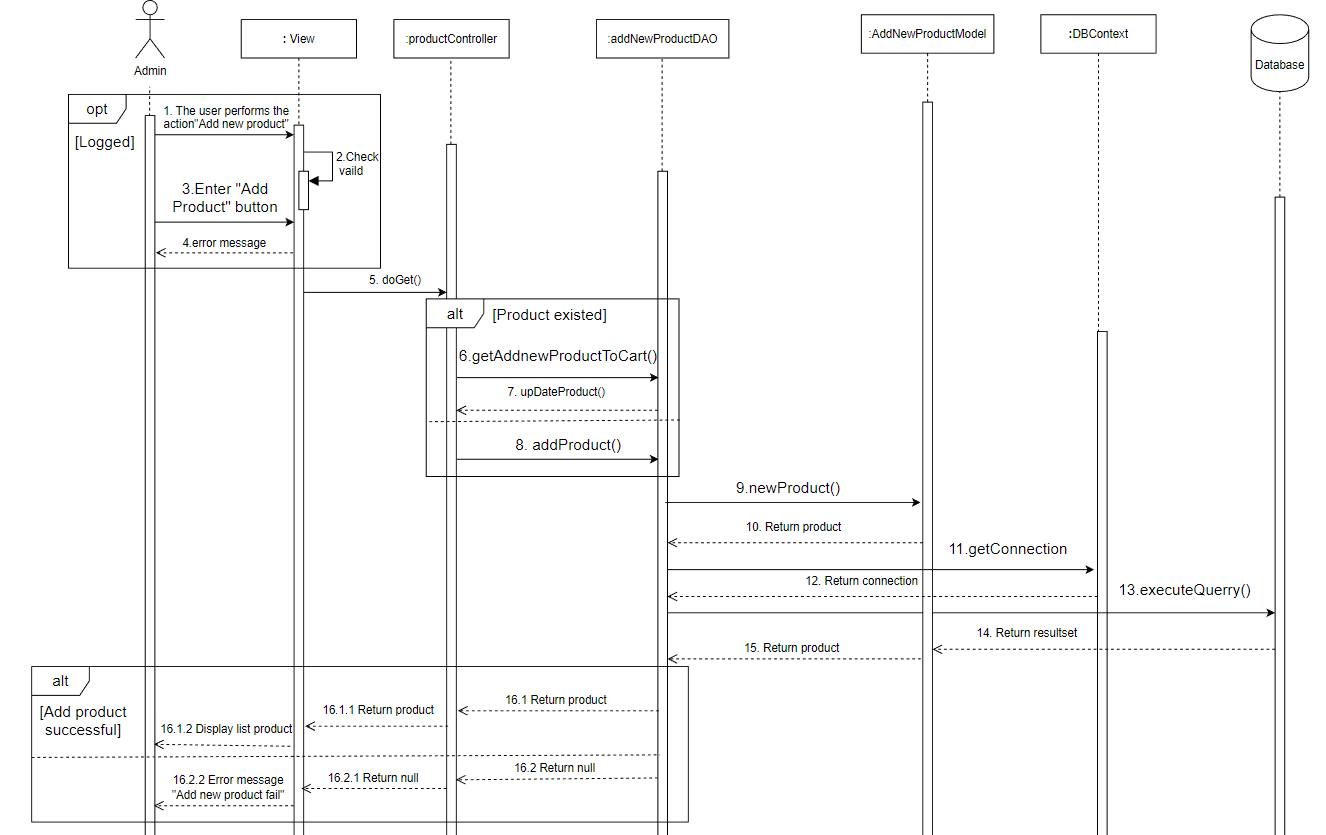


Figure 16.0 Sequence Diagram Add new product

### d. Database Queries

## 14.Order

### a. Class Diagram

***Class Create New Order***

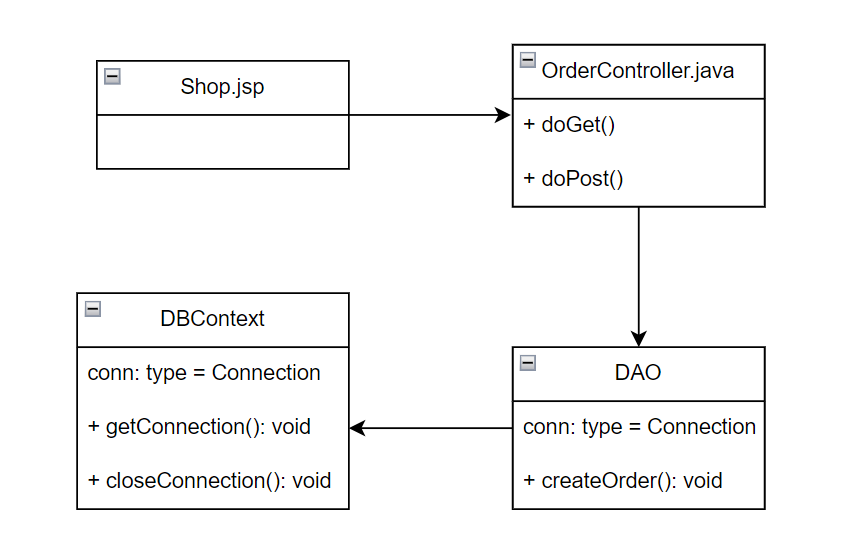


Figure 13.0 Class Diagram Create Order Function

### b. Class Specifications

### 

OrderController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's addNewProduct() method to add the product to the system and pass it to ManagerProduct.jsp page |

DAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createOrder(): | Create new order and store information to database. |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### 

### c. Sequence Diagram(s)

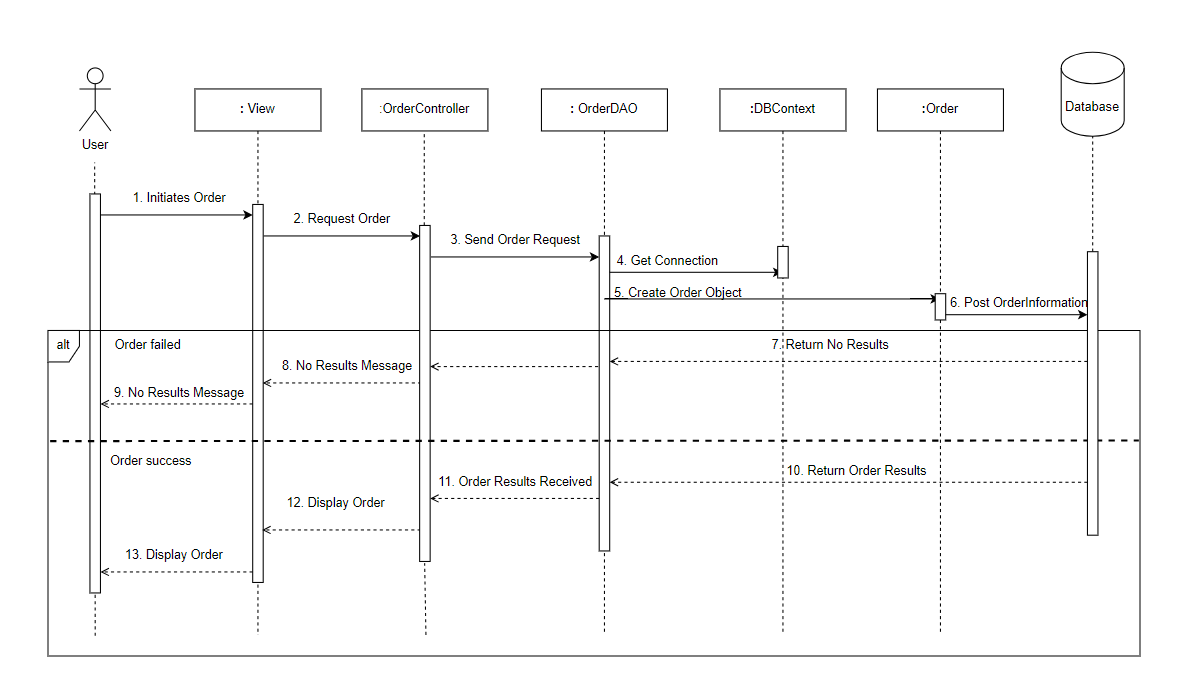


Figure 16.0 Sequence Diagram Order

### d. Database Queries

INSERT INTO [Order](orderDate, accountID,addressReceive,sdt,name,totalPrice,status)

## 15.CancelOrder

### a. Class Diagram

***Class Cancel Order***

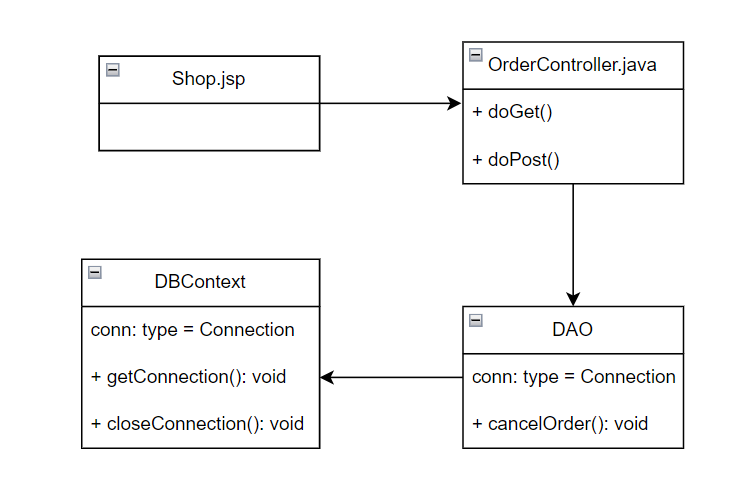


Figure 13.0 Class Diagram Cancel Order Function

### b. Class Specifications

### 

OrderController:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet(HttpServletRequest, HttpServletResponse) | A void method. It receives 2 inputs (request, response) sent from jsp pages. |
| 02 | doPost(HttpServletRequest, HttpServletResponse) | An empty method. It receives 2 inputs (request, response) sent from jsp pages. This method will add information from the jsp page and call DAO's addNewProduct() method to add the product to the system and pass it to ManagerProduct.jsp page |

DAO:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | cancelOrder(): | Cancel this order and change its status. |

DBContext:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getConnection() | Connect to the database and return the connection with its properties. |
| 02 | closeConnection() | Close the connection to the database |

### 

### c. Sequence Diagram(s)

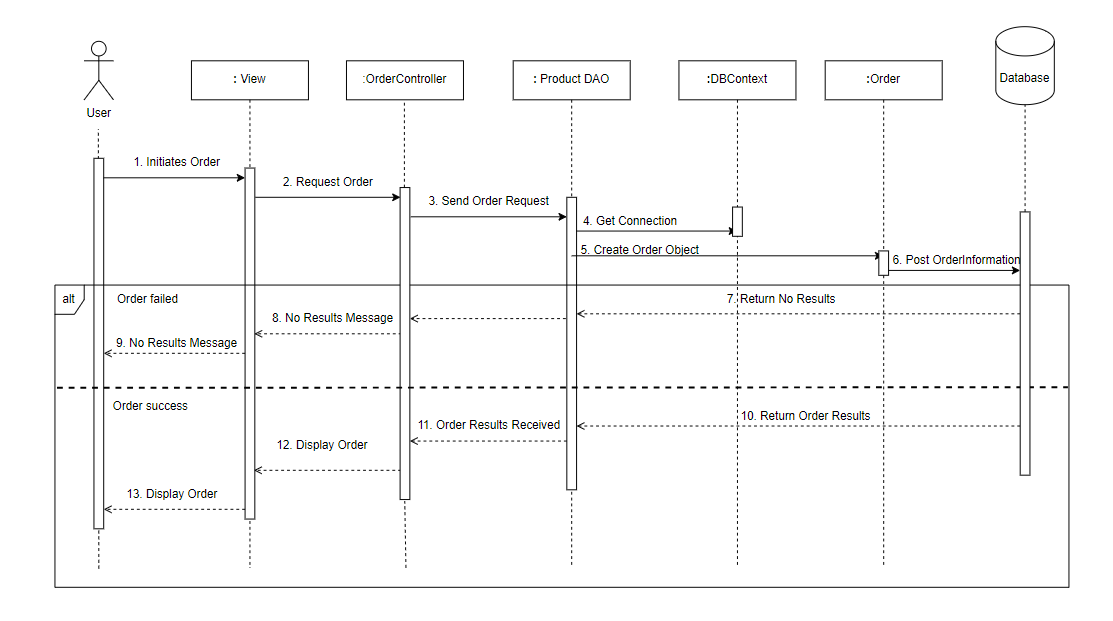


Figure 16.0 Sequence Diagram Cancel Order

### d. Database Queries

Update [Order] set status = 'Canceled' WHERE id = ?