

**Awns’s store**

**(Laptop & Phone)**

**Software Design Specification**

**Team 1**

– Hanoi, Feb 2023 –

Record of changeS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 17/2 | A | GiangNXT | Add Database Design (I.2) |
| 25/2 | A | GiangNXT | Add Code Design (II.1, II.2, II.3) |
| 26/2 | M | GiangNXT | Modified Code Design (II.3, II.4) |
| 2/3 | M | GiangNXT | Modified Code Design (II.1, II.2, II.3, II.4), Database Design (I.2) |
| 3/3 | A | GiangNXT | Add Code Design (II.5, II.6, II.7, II.8, II.9, II.10) |
| 15/3 | A | GiangNXT | Add Code Design (II.11, II.12, II.13, II.15) |
| 21/3 | A | GiangNXT |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_Toc129784039)

[1. Code Packages 4](#_Toc129784040)

[2. Database Design 4](#_Toc129784041)

[II. Code Designs 5](#_Toc129784042)

[1. Home Page – Display Product Lists 5](#_Toc129784043)

[2. Product List - Filter 7](#_Toc129784044)

[3. Search Order 10](#_Toc129784045)

[4. Login for Customer 12](#_Toc129784046)

[5. Login for Employee 15](#_Toc129784047)

[6. Log out 16](#_Toc129784048)

[7. Sign up user 17](#_Toc129784049)

[8. Edit Profile 19](#_Toc129784050)

[9. Product Details 21](#_Toc129784051)

[10. Add to Cart 24](#_Toc129784052)

[11. My Cart (Remove a product, Remove all product) 26](#_Toc129784053)

[12. Update quantity of product on cart 28](#_Toc129784054)

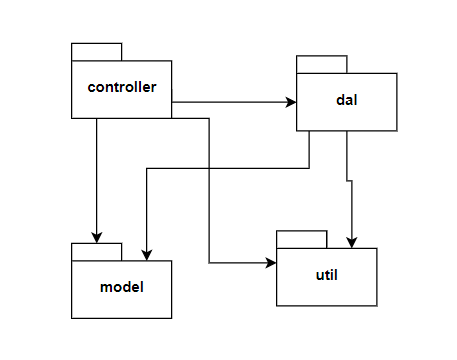
[13. Payment Page 29](#_Toc129784055)

[14. Admin Dashboard 31](#_Toc129784056)

[15. Employee Management 31](#_Toc129784057)

# I. Overview

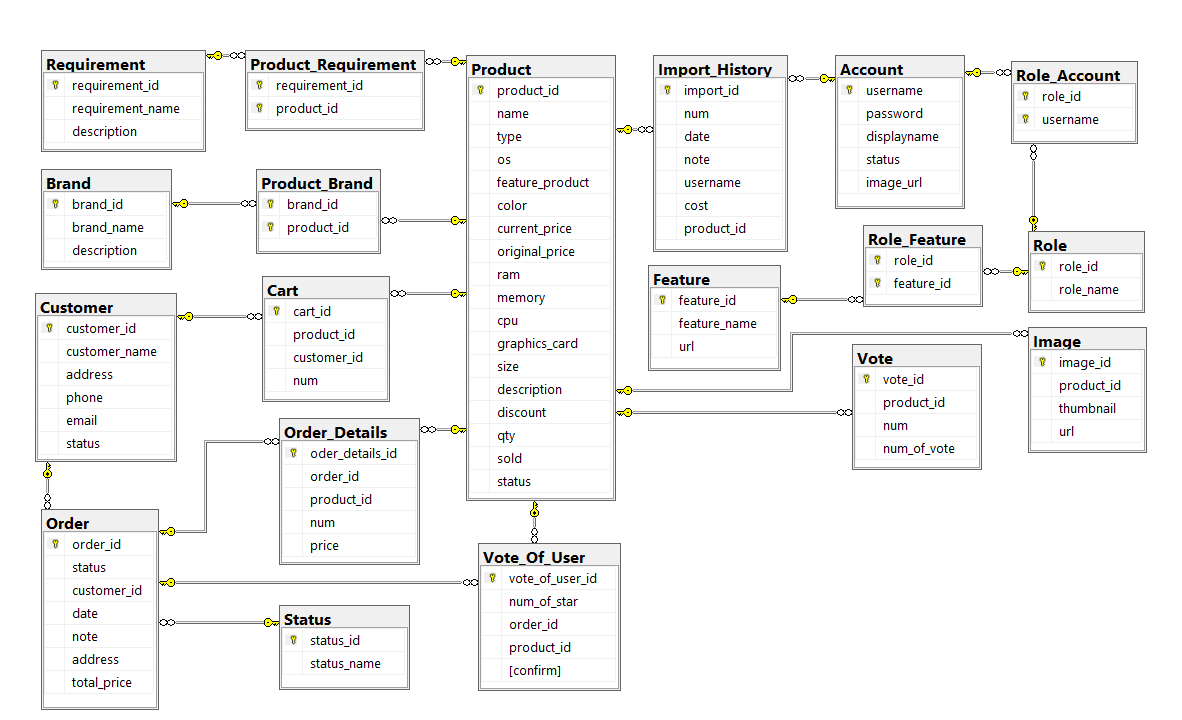
## 1. Code Packages



***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 1 | Controller | Data processing and navigate to jsp |
| 2 | Dal | Interact with data base |
| 3 | Model | Include objectives |
| 4 | Util | Function to support data processing |

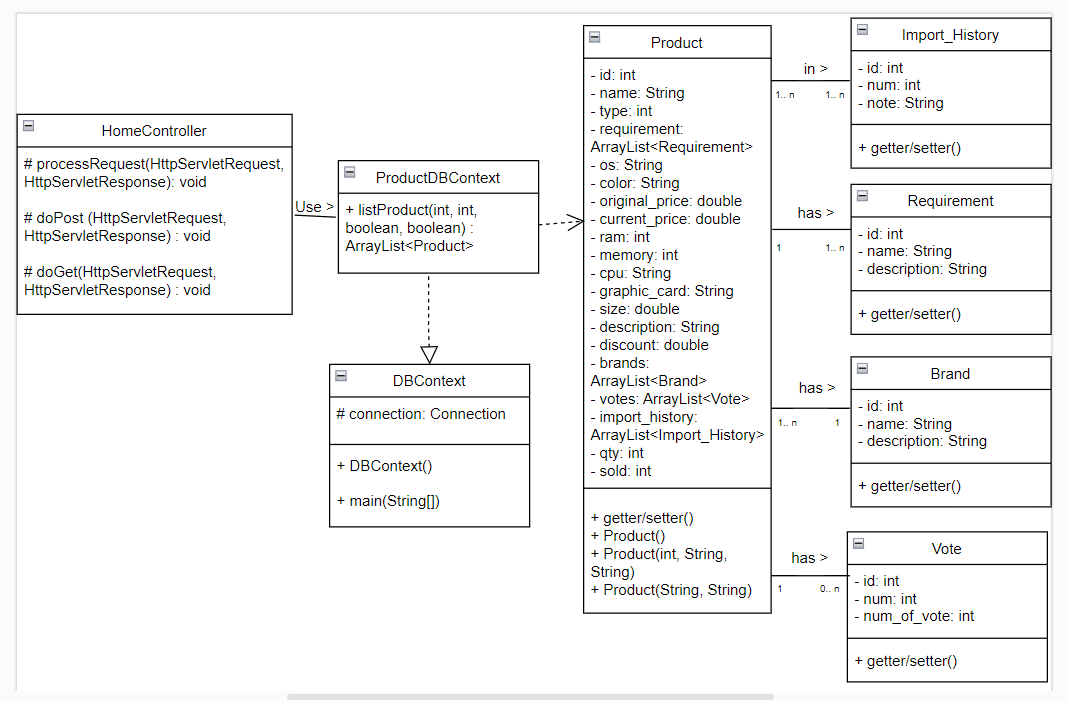
## 2. Database Design



# II. Code Designs

## 1. Home Page – Display Product Lists

a. Class Diagram



b. Class Specifications

#### HomeController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ProductDBContext

***Class Methods***

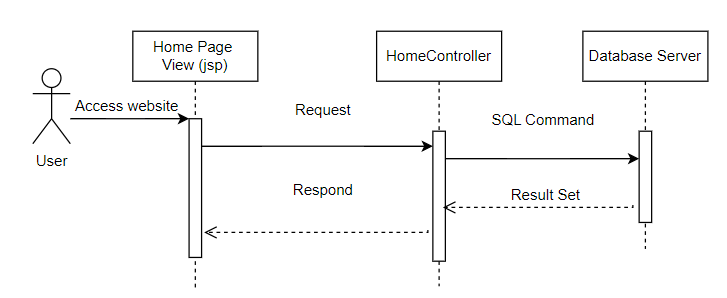
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listProduct(int, int, boolean, boolean): ArrayList<Product> | Get list of product base on: what type? (phone or laptop), how much? (number of product), Is top of sale? Is top of sold? |

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

c. Sequence Diagram(s)



d. Database Queries

//Select top 8 of best-seller

SELECT TOP 8 [PRODUCT\_ID], [NAME], [TYPE], [OS], [COLOR], [CURRENT\_PRICE]

, [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD]

, [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [SOLD], [STATUS]

FROM [PRODUCT] WHERE [STATUS] = 1

ORDER BY [SOLD] DESC

//Select top 8 of promotional product

SELECT TOP 8 [PRODUCT\_ID], [NAME], [TYPE], [OS], [COLOR], [CURRENT\_PRICE]

, [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD]

, [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [SOLD], [STATUS]

FROM [PRODUCT] WHERE [STATUS] = 1

ORDER BY [DISCOUNT] DESC

// Select first of 8 phones

SELECT TOP 8 [PRODUCT\_ID], [NAME], [TYPE], [OS], [COLOR], [CURRENT\_PRICE]

, [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD]

, [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [SOLD], [STATUS]

FROM [PRODUCT] WHERE [STATUS] = 1 AND [TYPE] = 0

//Select first of 8 laptops

SELECT TOP 8 [PRODUCT\_ID], [NAME], [TYPE], [OS], [COLOR], [CURRENT\_PRICE]

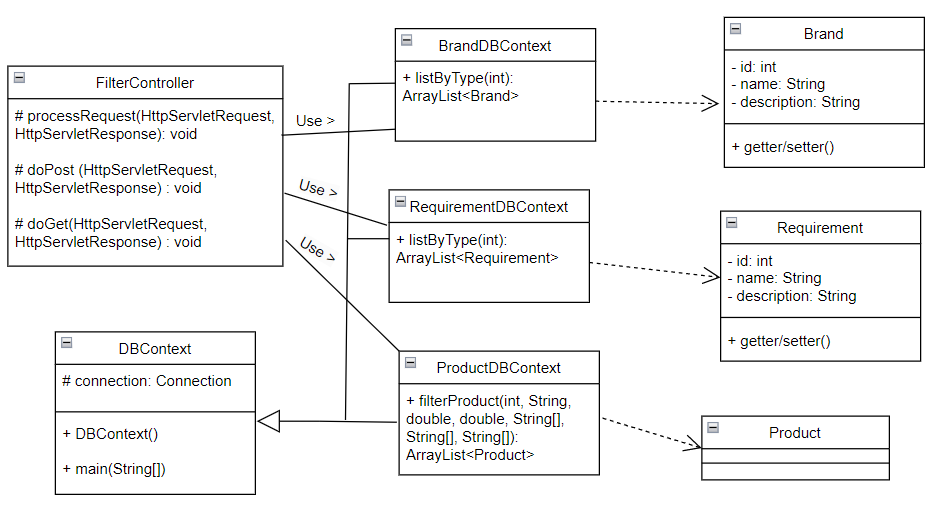
, [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD]

, [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [SOLD], [STATUS]

FROM [PRODUCT] WHERE [STATUS] = 1 AND [TYPE] = 1

## 2. Product List - Filter

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### FilterController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### BrandDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listByType(int): ArrayList<Brand> | Get list of brands by category product (phone or laptop) |

#### RequirementDBContext

***Class Methods***

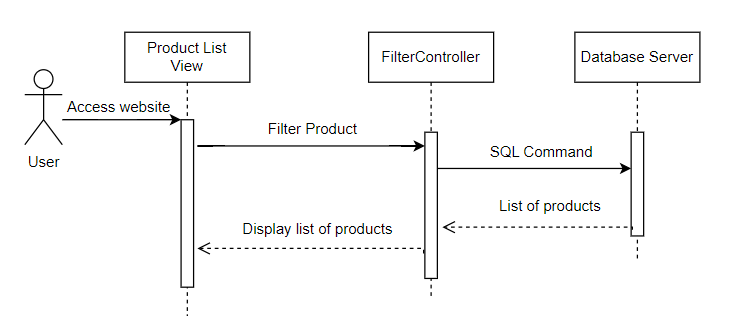
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listByType(int): ArrayList<Requirement> | Get list of requirements by category product (phone or laptop) |

#### ProductDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | filterProduct(int, String, double, double, String[], String[], String[]): ArrayList<Product> | Get list of products by filter |

c. Sequence Diagram(s)



d. Database Queries

//Select all brand of laptop or iphone

SELECT DISTINCT B.[BRAND\_ID], B.[BRAND\_NAME], B.[DESCRIPTION]

FROM [BRAND] B

INNER JOIN PRODUCT\_BRAND PB ON PB.BRAND\_ID = B.BRAND\_ID

INNER JOIN PRODUCT PR ON PR.PRODUCT\_ID = PB.PRODUCT\_ID

WHERE PR.[TYPE] =?

//Select all requirement of laptop or iphone

SELECT DISTINCT P.[REQUIREMENT\_ID], P.[REQUIREMENT\_NAME], P.[DESCRIPTION]

FROM [REQUIREMENT] P

INNER JOIN PRODUCT\_REQUIREMENT PR

ON PR.REQUIREMENT\_ID = P.REQUIREMENT\_ID

INNER JOIN PRODUCT PRD ON PRD.PRODUCT\_ID = PR.PRODUCT\_ID

WHERE PRD.[TYPE] = ?

//Select all product match with filter

SELECT DISTINCT PR.[PRODUCT\_ID], PR.[NAME], PR.[TYPE], PR.[OS], PR.[FEATURE\_PRODUCT]

, PR.[COLOR], PR.[CURRENT\_PRICE], PR.[ORIGINAL\_PRICE], PR.[RAM]

, PR.[MEMORY], PR.[CPU], PR.[GRAPHICS\_CARD], PR.[SIZE], PR.[DESCRIPTION]

, PR.[DISCOUNT], PR.[QTY], PR.[SOLD], PR.[STATUS]

FROM [PRODUCT] PR

INNER JOIN [PRODUCT\_BRAND] PRBR ON PR.PRODUCT\_ID = PRBR.PRODUCT\_ID

INNER JOIN [BRAND] BR ON PRBR.BRAND\_ID = BR.BRAND\_ID

INNER JOIN [PRODUCT\_REQUIREMENT] PRRE

ON PR.PRODUCT\_ID = PRRE.PRODUCT\_ID

INNER JOIN [REQUIREMENT] RE

ON PRRE.REQUIREMENT\_ID =RE.REQUIREMENT\_ID

WHERE PR.[STATUS] = 1

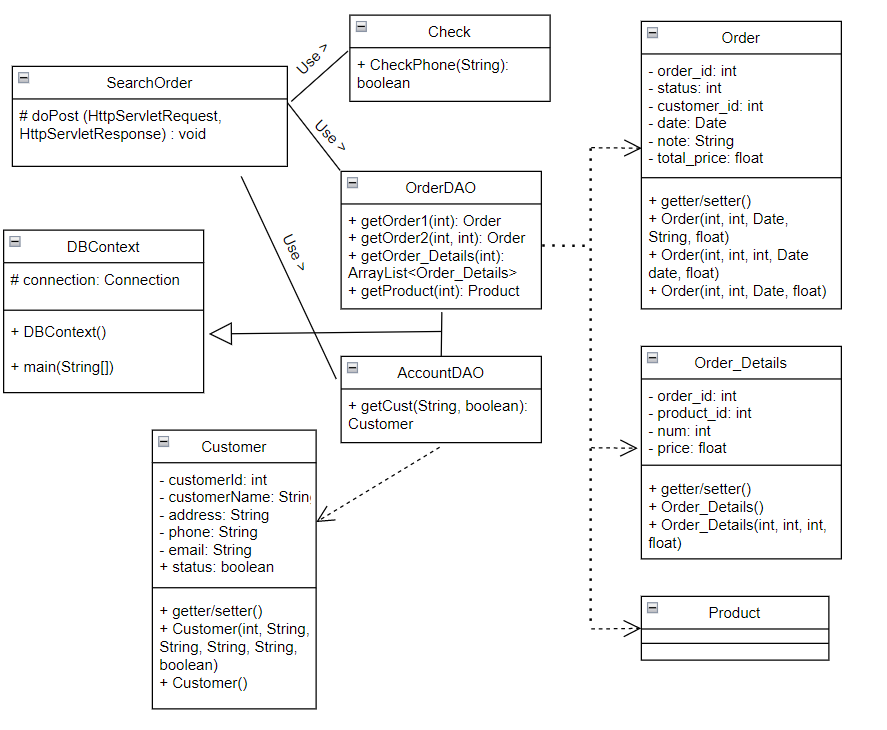
AND PR.[CURRENT\_PRICE] >= ?

AND PR.[CURRENT\_PRICE] <= ?

AND PR.[TYPE] = ?

## 3. Search Order

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### Check

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CheckPhone(String): boolean | Validate input is number or not |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getOrder1(int): Order | Get order by order id |
| 02 | getOrder2(int, int): Order | Get order by order id and customer id |
| 03 | getOrder\_Details(int):  ArrayList<Order\_Details> | Get list of Order Details by order id |
| 04 | getProduct(int): Product | Get product by product id |

AccountDAO

***Class Methods***

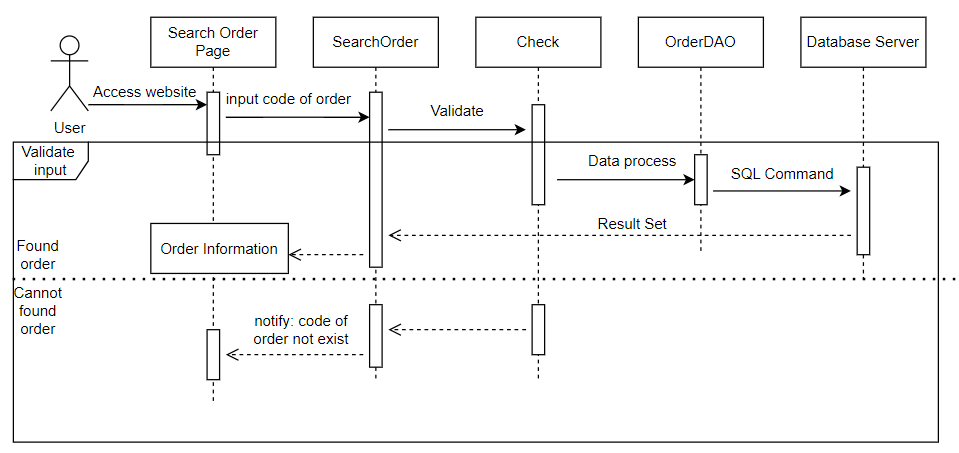
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getCust(String, boolean): Customer | Get Customer by email and status |

#### SearchOrder

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost (HttpServletRequest, HttpServletResponse): void | Handle request by doPost method |

c. Sequence Diagram(s)



d. Database Queries

//Select order by order id

SELECT \* FROM [ORDER] WHERE ORDER\_ID = ?

//Select order by order id and customer id

SELECT \* FROM [ORDER] WHERE ORDER\_ID = ? AND CUSTOMER\_ID = ?

//Select Order Details by order id

SELECT \* FROM ORDER\_DETAILS WHERE ORDER\_ID =?

//Select product by product id

SELECT [NAME],COLOR FROM PRODUCT WHERE PRODUCT\_ID = ?

//Select Customer by email and status

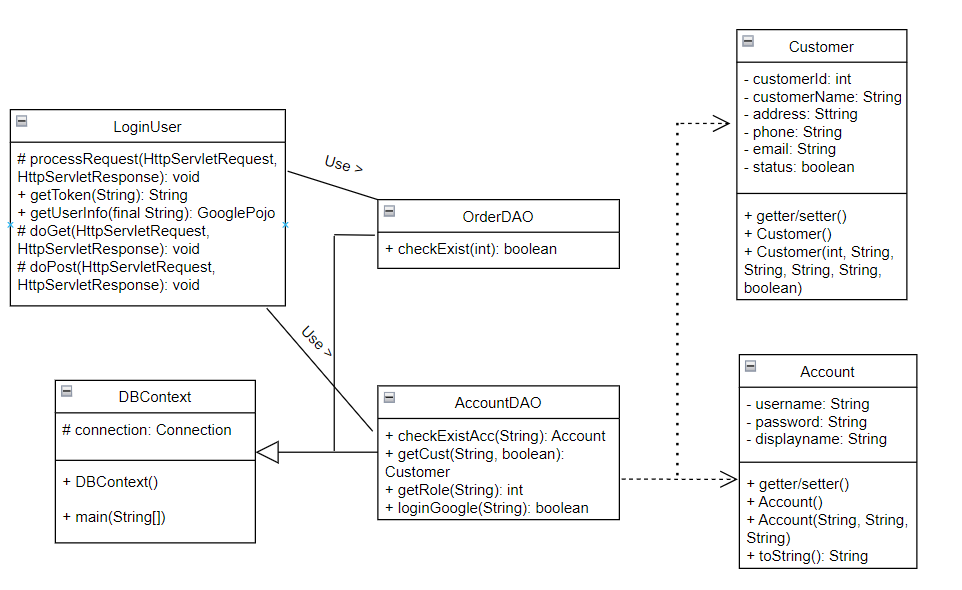
SELECT [CUSTOMER\_ID], [CUSTOMER\_NAME], [ADDRESS], [PHONE], [EMAIL], [STATUS]

FROM [DBO].[CUSTOMER]

WHERE [EMAIL] = ? AND STATUS =?

## 4. Login for Customer

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### LoginUser

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | Handle request |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 04 | getToken(String): String | Compile code from respond of google and send Verification Code back |
| 05 | getUserInfo(final String): GooglePojo | Get information of user from google |

#### OrderDAO

***Class Methods***

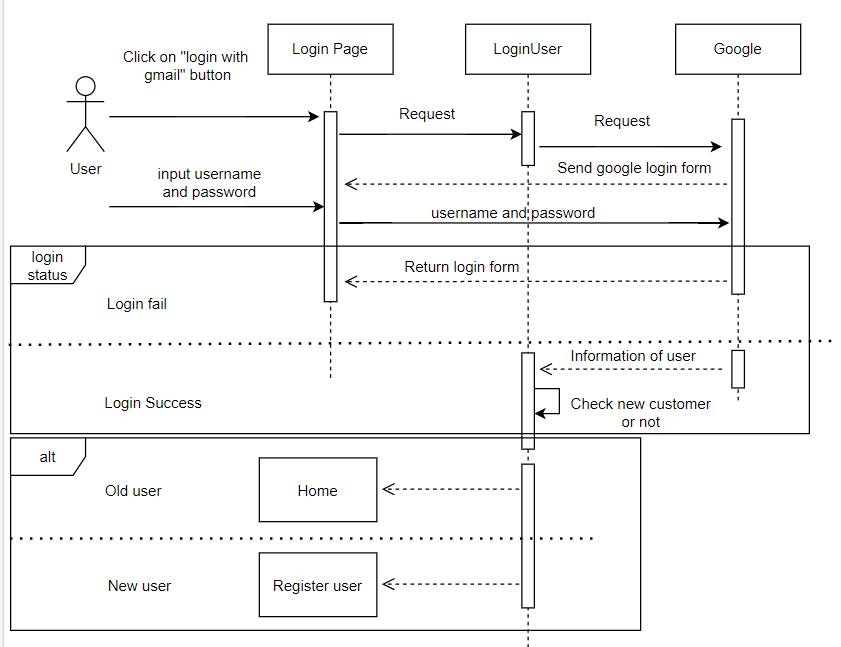
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkExist(int): boolean | Check user account has cart or not |

#### AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkExistAcc(String): Account | Check and get Account by username |
| 02 | getCust(String, boolean): Customer | Get customer information by email and status |
| 03 | getRole(String): int | Get role of account |
| 04 | loginGoogle(String): boolean | Check email exist in database or not |

c. Sequence Diagram(s)



d. Database Queries

//Select account by username

SELECT \* FROM ACCOUNT WHERE USERNAME =?

//Select customer by email and status

SELECT [CUSTOMER\_ID], [CUSTOMER\_NAME], [ADDRESS], [PHONE], [EMAIL], [STATUS]

FROM [DBO].[CUSTOMER]

WHERE [EMAIL] = ? AND STATUS =?

//Select role id by username

SELECT ROLE\_ID FROM ROLE\_ACCOUNT WHERE USERNAME = ?

//Select cart id by customer id

SELECT CART\_ID FROM CART WHERE CUSTOMER\_ID = ?

//Select customer information by email

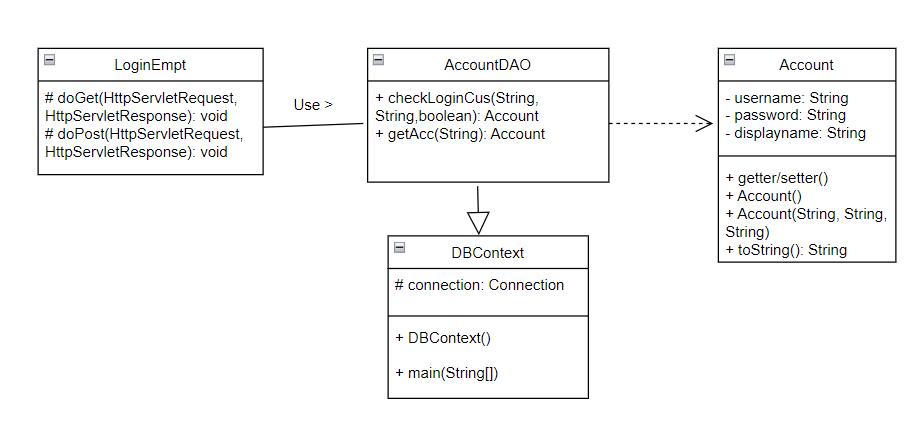
SELECT [CUSTOMER\_ID], [CUSTOMER\_NAME], [ADDRESS], [PHONE], [EMAIL], [STATUS]

FROM [DBO].[CUSTOMER]

WHERE [EMAIL] = ?

## 5. Login for Employee

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### AccountDAO

***Class Methods***

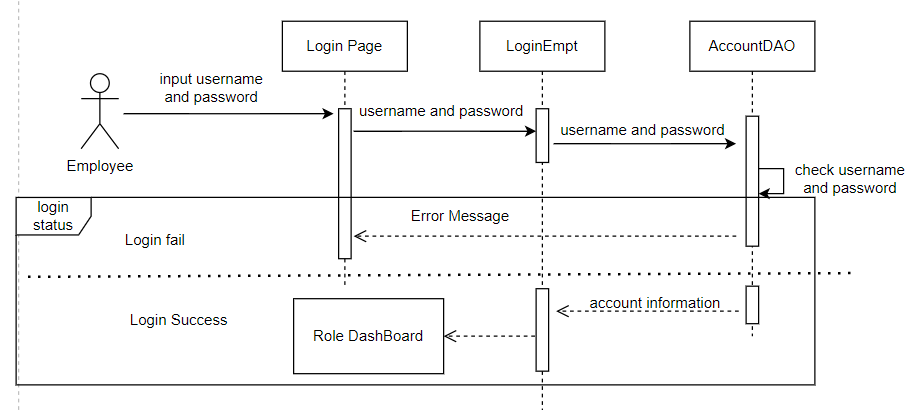
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkLoginCus(String, String,boolean): Account | Check username and password |
| 02 | getAcc(String): Account | Get account has specific username |

#### LoginEmpt

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request with Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request with Get method |

c. Sequence Diagram(s)



d. Database Queries

//select account has specific username and password

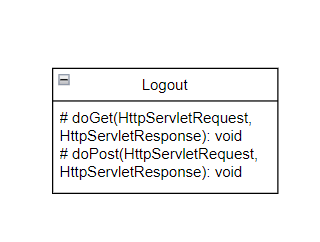
SELECT \* FROM ACCOUNT WHERE USERNAME =? AND PASSWORD=? AND [STATUS] =1

//select information of account has specific username

SELECT \* FROM ACCOUNT INNER JOIN ROLE\_ACCOUNT ON ACCOUNT.USERNAME = ROLE\_ACCOUNT.USERNAME WHERE ACCOUNT.USERNAME= ? AND ACCOUNT.STATUS = 1

## 6. Log out

a. Class Diagram



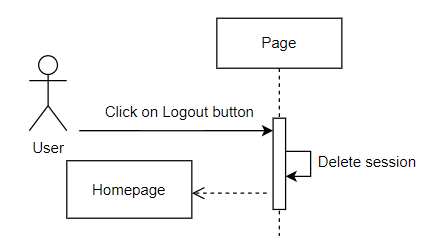
b. Class Specifications

#### Logout

***Class Methods***

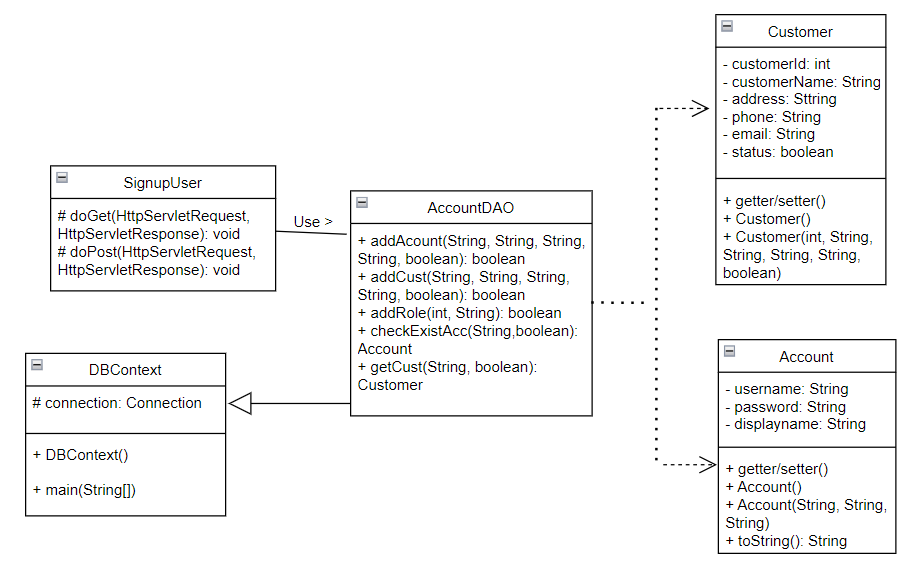
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request with Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request with Get method |

c. Sequence Diagram(s)



## 7. Sign up user

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### SignupUser

***Class Methods***

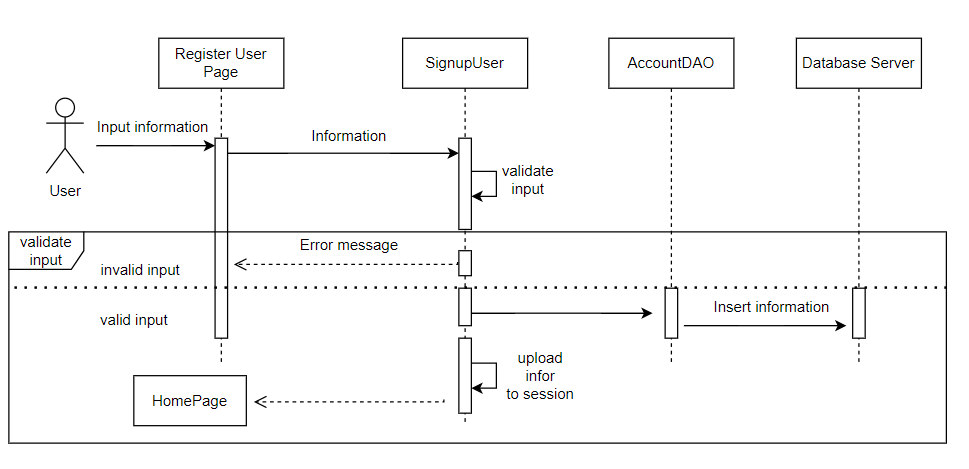
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request with Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request with Get method |

#### AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addAcount(String, String, String, String, boolean): boolean | Add account information to database |
| 02 | addCust(String, String, String, String, boolean): boolean | Add customer information to database |
| 03 | addRole(int, String): boolean | Add role for specific account |
| 04 | checkExistAcc(String,boolean): Account | Get specific account from database |
| 05 | getCust(String, boolean): Customer | Get customer by account |

c. Sequence Diagram(s)



d. Database Queries

//insert new account

INSERT INTO ACCOUNT

(USERNAME,PASSWORD,DISPLAYNAME,[STATUS],IMAGE\_URL) VALUES (?,?,?,?,?)

//insert new customer

INSERT INTO CUSTOMER

(CUSTOMER\_NAME,ADDRESS,PHONE,EMAIL,[STATUS]) VALUES (?,?,?,?,?)

//add role to account

INSERT INTO ROLE\_ACCOUNT

(ROLE\_ID,USERNAME) VALUES(?,?)

//select account by specific username

SELECT \* FROM ACCOUNT

WHERE USERNAME =? AND [STATUS] = 1

//select customer information by email

SELECT [CUSTOMER\_ID]

,[CUSTOMER\_NAME]

,[ADDRESS]

,[PHONE]

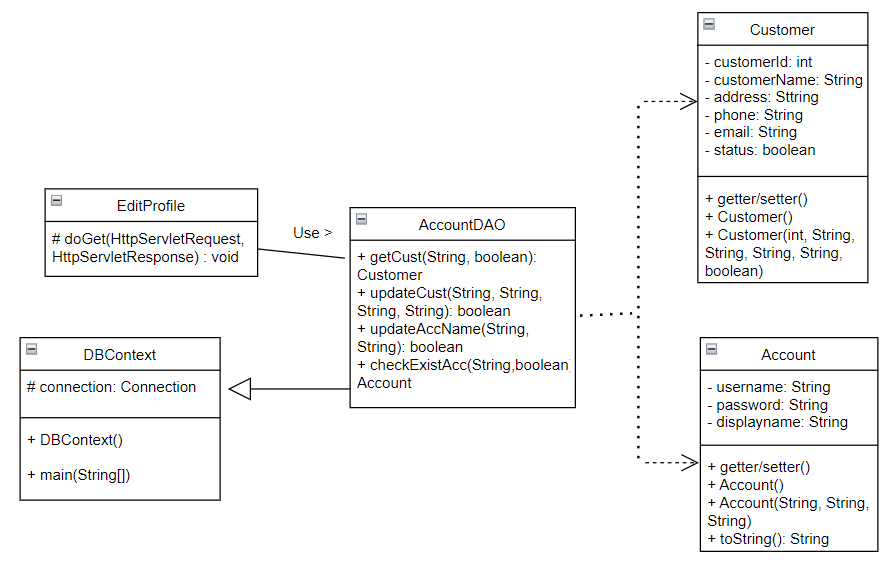
,[EMAIL],[STATUS]

FROM [DBO].[CUSTOMER]

WHERE [EMAIL] = ? AND STATUS =?

## 8. Edit Profile

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### EditProfile

***Class Methods***

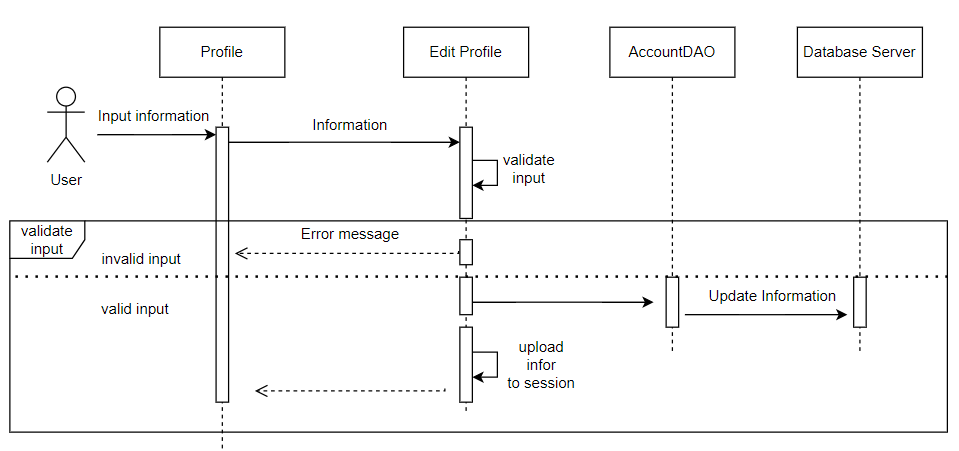
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request with Get method |

#### AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getCust(String, boolean): Customer | Get customer by email |
| 02 | updateCust(String, String, String, String): boolean | Update information of customer |
| 03 | updateAccName(String, String): boolean | Update display name of account |
| 04 | checkExistAcc(String,boolean): Account | Get account by username |

c. Sequence Diagram(s)



d. Database Queries

//Update customer information

UPDATE CUSTOMER

SET CUSTOMER\_NAME = ? ,ADDRESS = ? ,PHONE = ? WHERE EMAIL = ?

//Update displayname of account

UPDATE ACCOUNT SET DISPLAYNAME = ? WHERE USERNAME = ?

//select customer information by email

SELECT [CUSTOMER\_ID]

,[CUSTOMER\_NAME]

,[ADDRESS]

,[PHONE]

,[EMAIL],[STATUS]

FROM [DBO].[CUSTOMER]

WHERE [EMAIL] = ? AND STATUS =?

//select account information by username

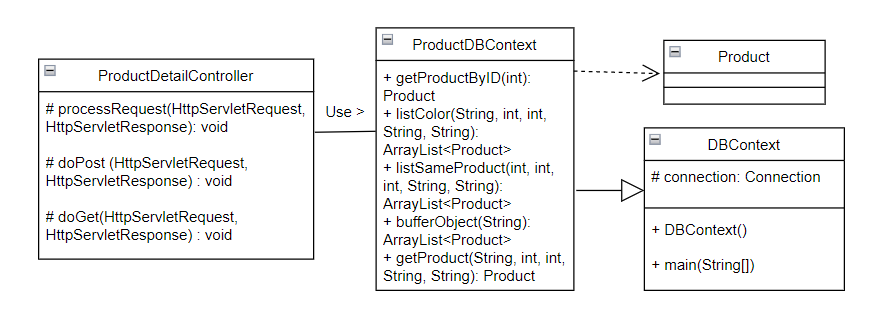
SELECT \*

FROM ACCOUNT

WHERE USERNAME =? AND [STATUS] = 1

## 9. Product Details

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### ProductDetailController

***Class Methods***

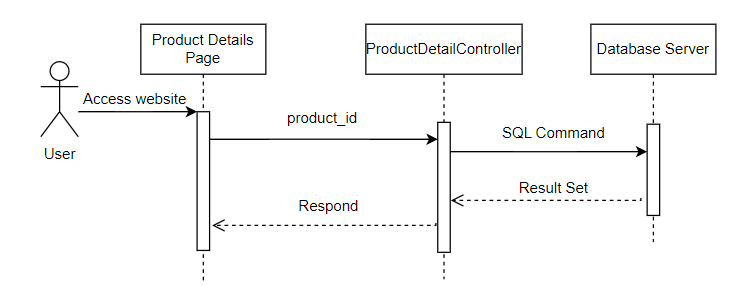
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ProductDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getProductByID(int): Product | Get Product by Product id |
| 02 | listColor(String, int, int, String, String): ArrayList<Product> | Get list of same products with difference color |
| 03 | listSameProduct(int, int, int, String, String): ArrayList<Product> | Get list of product same with specific product |
| 04 | bufferObject(String): ArrayList<Product> | Get list of product have same name with specific product |
| 05 | getProduct(String, int, int, String, String): Product | Get product with specific information |

c. Sequence Diagram(s)



d. Database Queries

//select product by product id

SELECT [PRODUCT\_ID],[NAME],[TYPE],[OS],[COLOR],[CURRENT\_PRICE],[ORIGINAL\_PRICE],[RAM]

,[MEMORY],[CPU],[GRAPHICS\_CARD],[SIZE],[DESCRIPTION],[DISCOUNT],[QTY]

,[SOLD],[STATUS]

FROM [PRODUCT] WHERE [STATUS] = 1 AND [PRODUCT\_ID] = ?

//select same product with difference color

SELECT DISTINCT PR.[PRODUCT\_ID],PR.[NAME],PR.[TYPE],PR.[OS],PR.[FEATURE\_PRODUCT],PR.[COLOR]

,PR.[CURRENT\_PRICE],PR.[ORIGINAL\_PRICE],PR.[RAM],PR.[MEMORY],PR.[CPU]

,PR.[GRAPHICS\_CARD] ,PR.[SIZE],PR.[DESCRIPTION],PR.[DISCOUNT],PR.[QTY],PR.[SOLD]

,PR.[STATUS]

FROM [PRODUCT] PR

WHERE PR.[STATUS] = 1 AND PR.[NAME] LIKE ?

AND PR.[RAM] LIKE ? AND PR.[MEMORY] LIKE ? AND PR.[CPU] LIKE ? AND PR.[GRAPHICS\_CARD] LIKE ?

//select same product

SELECT DISTINCT TOP (?) PR.[PRODUCT\_ID],PR.[NAME],PR.[TYPE],PR.[OS],PR.[FEATURE\_PRODUCT],PR.[COLOR]

,PR.[CURRENT\_PRICE],PR.[ORIGINAL\_PRICE],PR.[RAM],PR.[MEMORY],PR.[CPU]

,PR.[GRAPHICS\_CARD],PR.[SIZE],PR.[DESCRIPTION],PR.[DISCOUNT],PR.[QTY],PR.[SOLD]

,PR.[STATUS]

FROM [PRODUCT] PR

WHERE PR.[STATUS] = 1

AND PR.[RAM] LIKE ? AND PR.[MEMORY] LIKE ? AND PR.[CPU] LIKE ? AND PR.[GRAPHICS\_CARD] LIKE ?

//select product with same name

SELECT DISTINCT

PR.[NAME] ,PR.[RAM],PR.[MEMORY]

,PR.[CPU],PR.[GRAPHICS\_CARD]

FROM [PRODUCT] PR

WHERE PR.[STATUS] = 1 AND PR.[NAME] LIKE ?

//select product have specific information

SELECT TOP (1) [PRODUCT\_ID],[NAME],[TYPE],[OS],[COLOR],[CURRENT\_PRICE],[ORIGINAL\_PRICE],[RAM],[MEMORY]

,[CPU],GRAPHICS\_CARD],[SIZE],[DESCRIPTION],[DISCOUNT],[QTY],[SOLD],[STATUS]

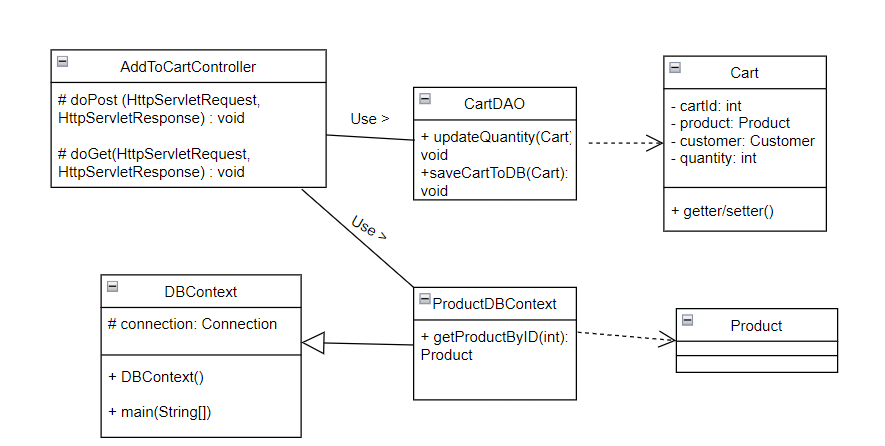
FROM [PRODUCT]

WHERE [STATUS] = 1 AND [NAME] LIKE ?

AND [RAM] LIKE ? AND [MEMORY] LIKE ? AND [CPU] LIKE ? AND [GRAPHICS\_CARD] LIKE ?

## 10. Add to Cart

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### AddToCartController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request with Get method |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request with Post method |

#### CartDAO

***Class Methods***

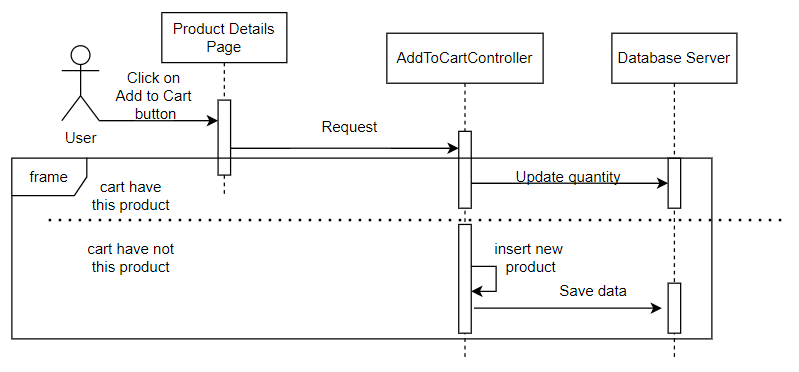
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateQuantity(Cart): void | Update quantity in specific cart |
| 02 | saveCartToDB(Cart): void | Save cart to database |

#### ProductDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getProductByID(int): Product | Get Product by Product ID |

c. Sequence Diagram(s)



d. Database Queries

//update quantity of product in specific cart

UPDATE [DBO].[CART]

SET [PRODUCT\_ID] = ?

,[CUSTOMER\_ID] = ?

,[NUM] = ?

WHERE PRODUCT\_ID = ? AND CUSTOMER\_ID = ?

//insert new cart

INSERT INTO [DBO].[CART]

([PRODUCT\_ID],[CUSTOMER\_ID],[NUM])

VALUES (?, ?, ?)

//select product by specific information

SELECT TOP (1) [PRODUCT\_ID],[NAME],[TYPE],[OS],[COLOR],[CURRENT\_PRICE],[ORIGINAL\_PRICE],[RAM],[MEMORY]

,[CPU],GRAPHICS\_CARD],[SIZE],[DESCRIPTION],[DISCOUNT],[QTY],[SOLD],[STATUS]

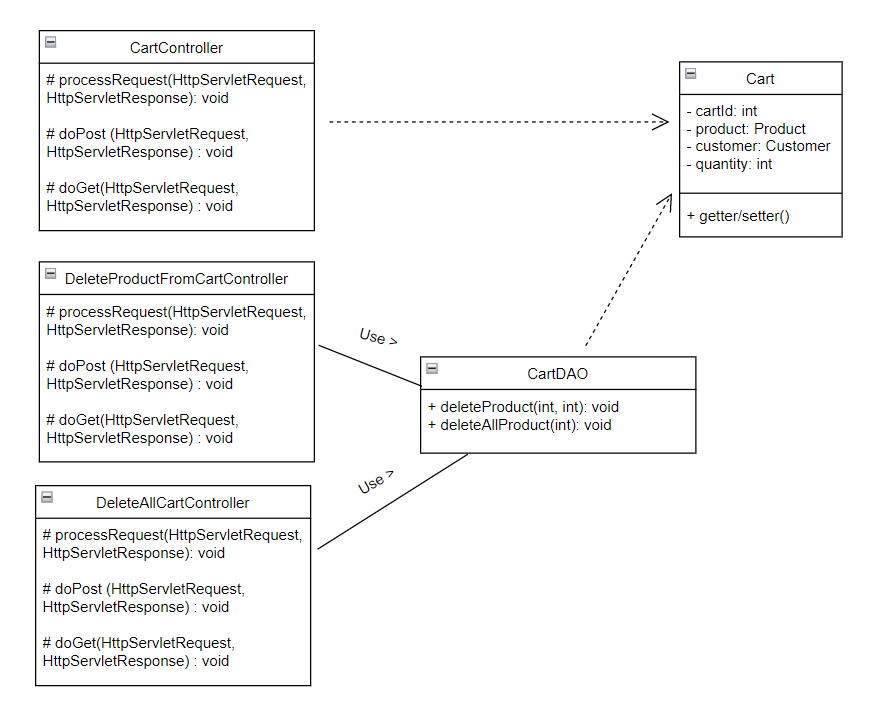
FROM [PRODUCT]

WHERE [STATUS] = 1 AND [NAME] LIKE ?

AND [RAM] LIKE ? AND [MEMORY] LIKE ? AND [CPU] LIKE ? AND [GRAPHICS\_CARD] LIKE ?

## 11. My Cart (Remove a product, Remove all product)

a. Class Diagram



b. Class Specifications

#### CartController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

**DeleteProductFromCartController**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

**DeleteAllCartController**

***Class Methods***

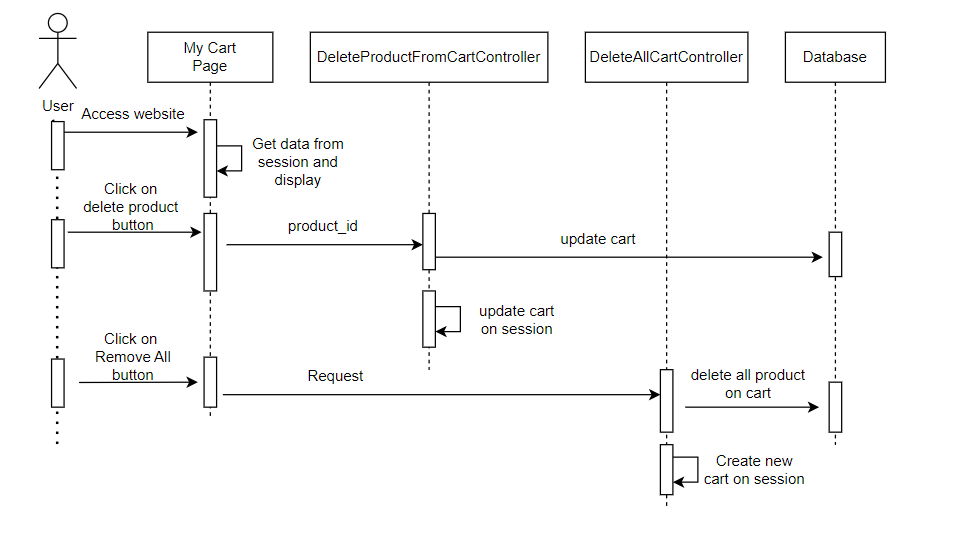
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

**CartDAO**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | deleteProduct(int, int): void | Delete specific product on cart |
| 02 | deleteAllProduct(int): void | Delete all product on cart |

c. Sequence Diagram(s)



d. Database Queries

//delete specific product on cart

DELETE FROM [DBO].[CART]

WHERE PRODUCT\_ID = ? AND CUSTOMER\_ID = ?

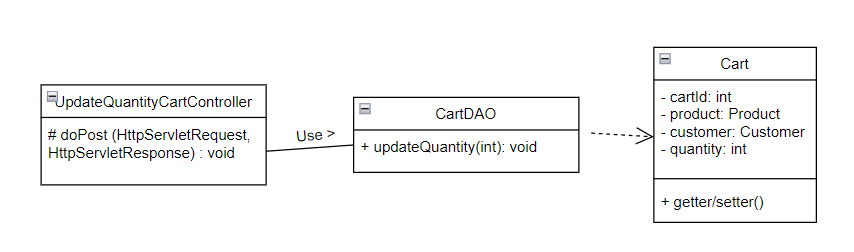
//delete cart by customer id

DELETE FROM [DBO].[CART]

WHERE CUSTOMER\_ID = ?

## 12. Update quantity of product on cart

a. Class Diagram



b. Class Specifications

#### CartDAO

***Class Methods***

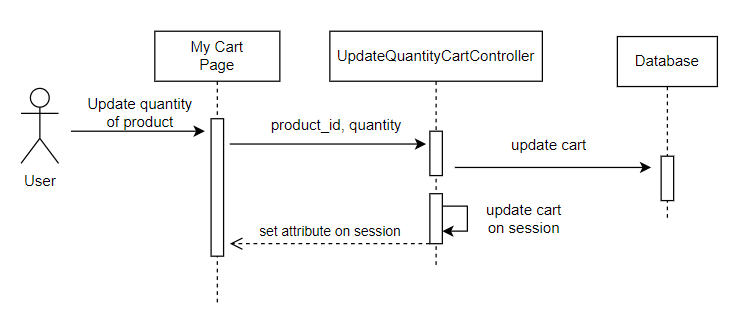
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateQuantity(int): void | Update quantity of product on cart |

**UpdateQuantityCartController**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by Post method |

c. Sequence Diagram(s)



d. Database Queries

//update quantity of product on cart in database

UPDATE [DBO].[CART]

SET [PRODUCT\_ID] = ?

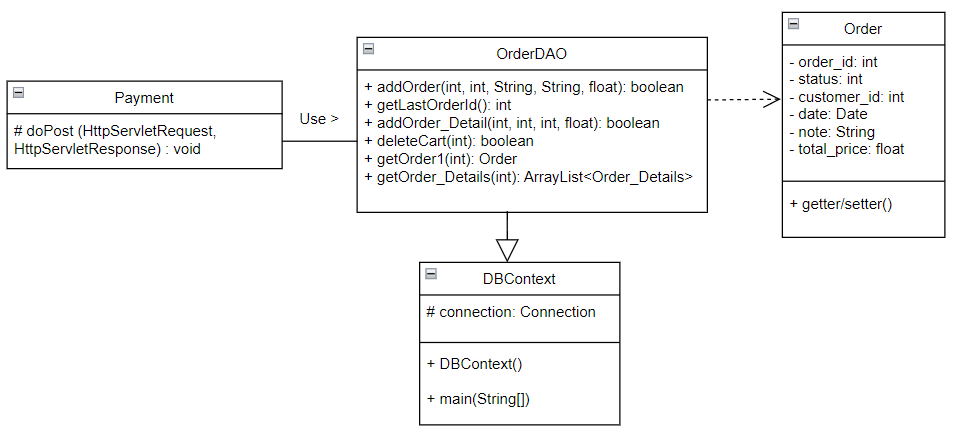
,[CUSTOMER\_ID] = ?

,[NUM] = ?

WHERE PRODUCT\_ID = ? AND CUSTOMER\_ID = ?

## 13. Payment Page

a. Class Diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

**Payment**

***Class Methods***

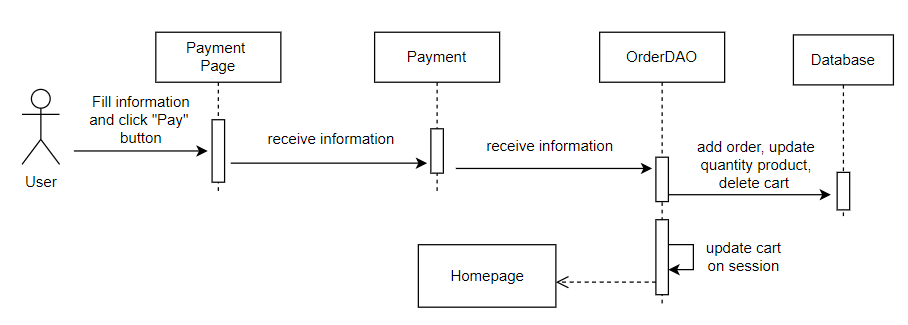
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by Post method |

**OrderDAO**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addOrder(int, int, String, String, float): boolean | Create new order in database |
| 02 | getLastOrderId(): int | Get last order |
| 03 | addOrder\_Detail(int, int, int, float): boolean | Create new order details in database |
| 04 | deleteCart(int): boolean | Delete cart |
| 05 | getOrder1(int): Order | Get order by order id |
| 06 | getOrder\_Details(int): ArrayList<Order\_Details> | Get order details by order id |

c. Sequence Diagram(s)



d. Database Queries

//insert new order

INSERT INTO [ORDER]

(STATUS,CUSTOMER\_ID,DATE,NOTE,TOTAL\_PRICE) VALUES (?,?,?,?,?)

//select last order

SELECT TOP 1 ORDER\_ID FROM [ORDER] ORDER BY ORDER\_ID DESC

//insert new order details

INSERT INTO ORDER\_DETAILS "

+ "(ORDER\_ID,PRODUCT\_ID,NUM,PRICE) VALUES (?,?,?,?)

//delete specific cart

DELETE FROM CART WHERE CART\_ID =?

//select specific order by order id

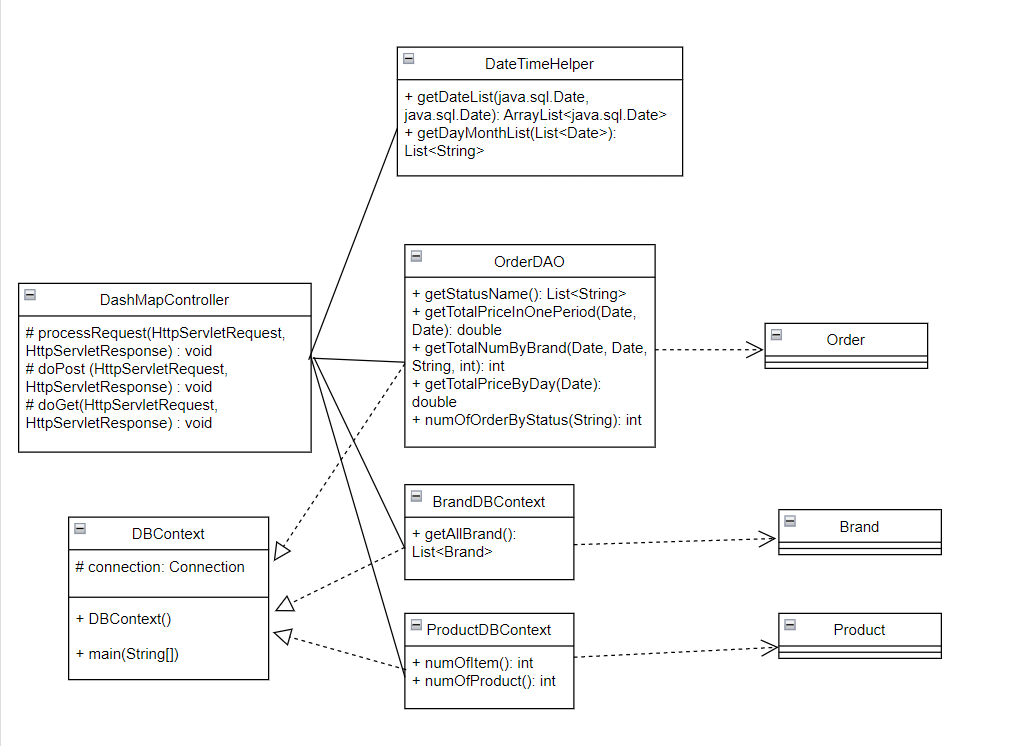
SELECT \* FROM [ORDER] WHERE ORDER\_ID = ?

//select order details by order id

SELECT \* FROM ORDER\_DETAILS WHERE ORDER\_ID =?

## 14. Admin Dashboard

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### DashMapController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### BrandDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllBrand(): List<Brand> | Get all brand of product |

#### ProductDBContext

***Class Methods***

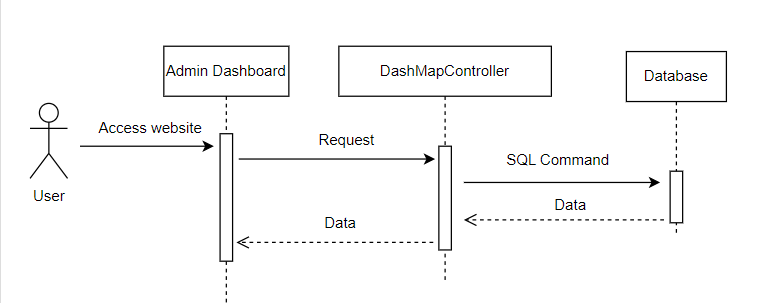
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | numOfItem(): int | Get all brand of product |
| 02 | numOfProduct(): int | Get total quantity of product |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getStatusName(): List<String> | Get list of status name |
| 02 | getTotalPriceInOnePeriod(Date, Date): double | Get Total price of all order on specific date |
| 03 | getTotalNumByBrand(Date, Date, String, int): int | Get number of each type product |
| 04 | getTotalPriceByDay(Date): double | Get total price of all order on a day |
| 05 | numOfOrderByStatus(String): int | Get number of order with specific status order |

c. Sequences Diagram(s)



d. Database query

//select all brand

SELECT [BRAND\_ID], [BRAND\_NAME], [DESCRIPTION] FROM [BRAND]

//count number of product

SELECT COUNT(\*) AS X FROM [PRODUCT]

//select total quantity of product

SELECT SUM(QTY) AS TOTAL FROM [PRODUCT]

//select status name

SELECT [STATUS\_NAME] FROM [STATUS]

//select sum of total price with specific date

SELECT SUM([TOTAL\_PRICE]) AS [TOTAL] FROM [ORDER] WHERE [DATE] BETWEEN ? AND ?

//select total price of order with specific brand

SELECT SUM(ORDDET.[NUM]) AS [TOTAL\_NUM]

FROM [ORDER\_DETAILS] ORDDET

INNER JOIN [ORDER] ORD ON ORDDET.ORDER\_ID = ORD.ORDER\_ID

INNER JOIN [PRODUCT] PR ON PR.PRODUCT\_ID = ORDDET.PRODUCT\_ID

INNER JOIN [PRODUCT\_BRAND] PRBR ON PR.PRODUCT\_ID = PRBR.PRODUCT\_ID

INNER JOIN [BRAND] BR ON PRBR.BRAND\_ID = BR.BRAND\_ID

WHERE BR.BRAND\_NAME = ? AND PR.TYPE = ? AND ORD.DATE BETWEEN ? AND ?

GROUP BY BR.BRAND\_ID

//select total price on specific day

SELECT SUM([TOTAL\_PRICE]) AS [TOTAL] FROM [ORDER] WHERE [DATE] = ? GROUP BY [DATE]

//select number of order by status

SELECT COUNT(ORD.[STATUS]) AS C

FROM [ORDER] ORD

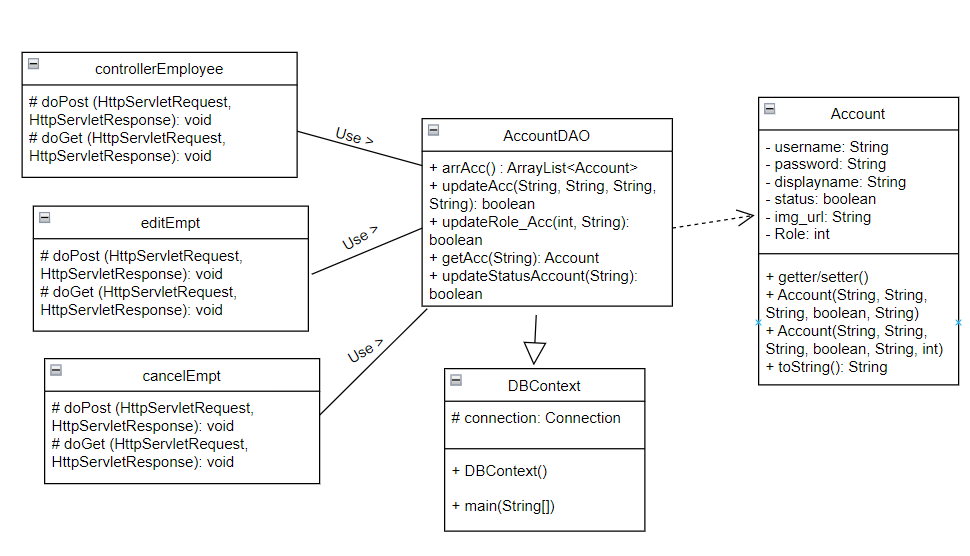
INNER JOIN [STATUS] STA ON STA.STATUS\_ID = ORD.[STATUS]

WHERE STA.STATUS\_NAME = ?

GROUP BY ORD.[STATUS] ORDER BY ORD.[STATUS] ASC

## 15. Employee Management

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

**editEmpt**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request by Get method |

**controllerEmployee**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request by Get method |

**cancelEmpt**

***Class Methods***

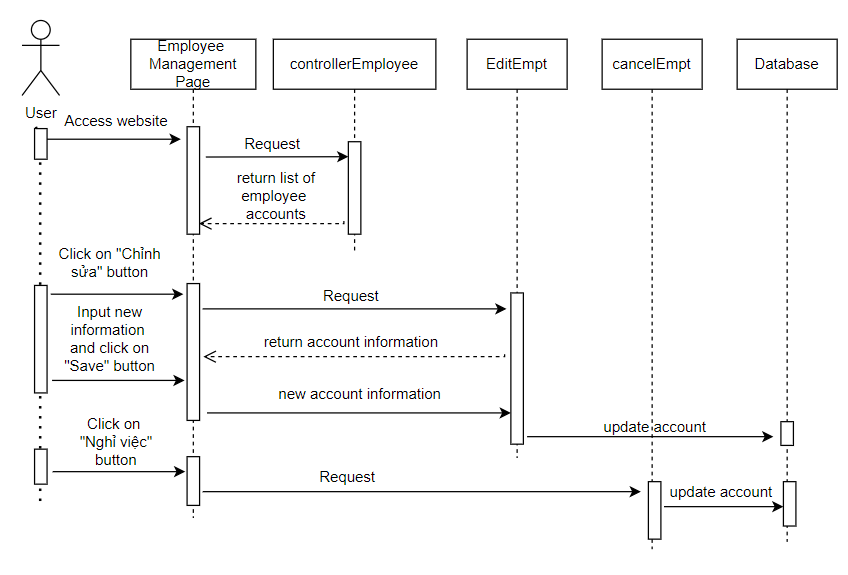
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by Post method |
| 02 | doGet(HttpServletRequest, HttpServletResponse): void | Handle request by Get method |

**AccountDAO**

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | arrAcc(): ArrayList<Account> | Get list of active employee accounts |
| 02 | updateAcc(String, String, String, String): boolean | Update account with new information |
| 03 | updateRole\_Acc(int, String): boolean | Update role of specific account |
| 04 | getAcc(String): Account | Get Account by username |
| 05 | updateStatusAccount(String): boolean | Update status of account (active/deactive) |

c. Sequences Diagram(s)



d. Database query

//select account has id is 2 or 3 and status account is 1

SELECT \* FROM ACCOUNT

INNER JOIN ROLE\_ACCOUNT ON ACCOUNT.USERNAME = ROLE\_ACCOUNT.USERNAME

WHERE (ROLE\_ACCOUNT.ROLE\_ID =2 OR ROLE\_ACCOUNT.ROLE\_ID =3)

AND ACCOUNT.STATUS = 1

//update account information

UPDATE ACCOUNT

SET [PASSWORD] = ?, DISPLAYNAME = ?, IMAGE\_URL=? WHERE USERNAME =?

//update role account

UPDATE ROLE\_ACCOUNT

SET ROLE\_ID = ? WHERE USERNAME = ?

//select account by specific username and status account is 1

SELECT \* FROM ACCOUNT

INNER JOIN ROLE\_ACCOUNT ON ACCOUNT.USERNAME = ROLE\_ACCOUNT.USERNAME

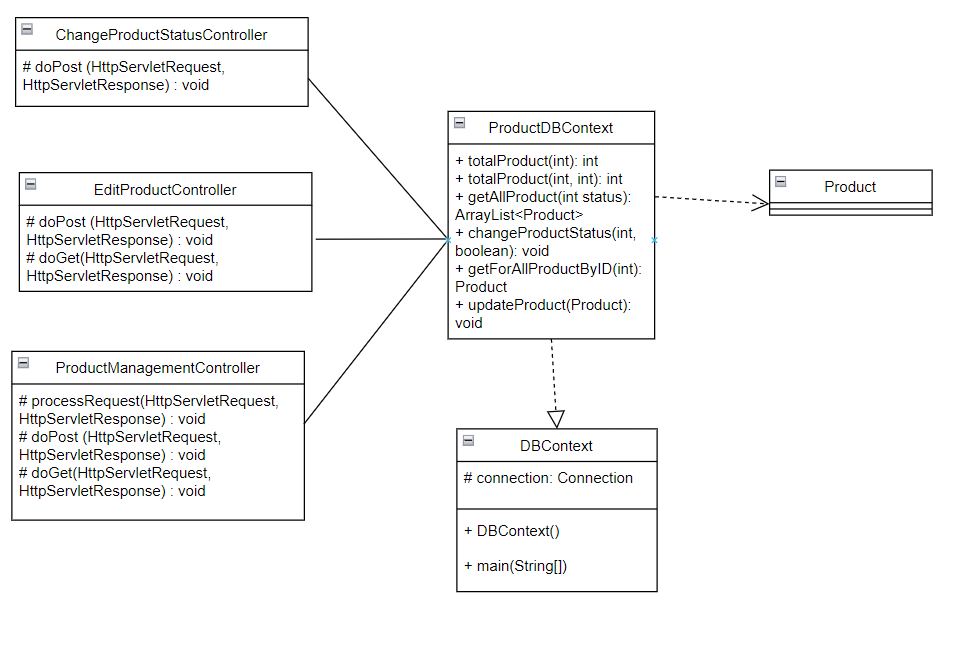
WHERE ACCOUNT.USERNAME = ? AND ACCOUNT.STATUS = 1

//update status account

UPDATE ACCOUNT SET [STATUS] = 0 WHERE USERNAME = ?

## 16. Product Management

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### ProductManagementController

***Class Methods***

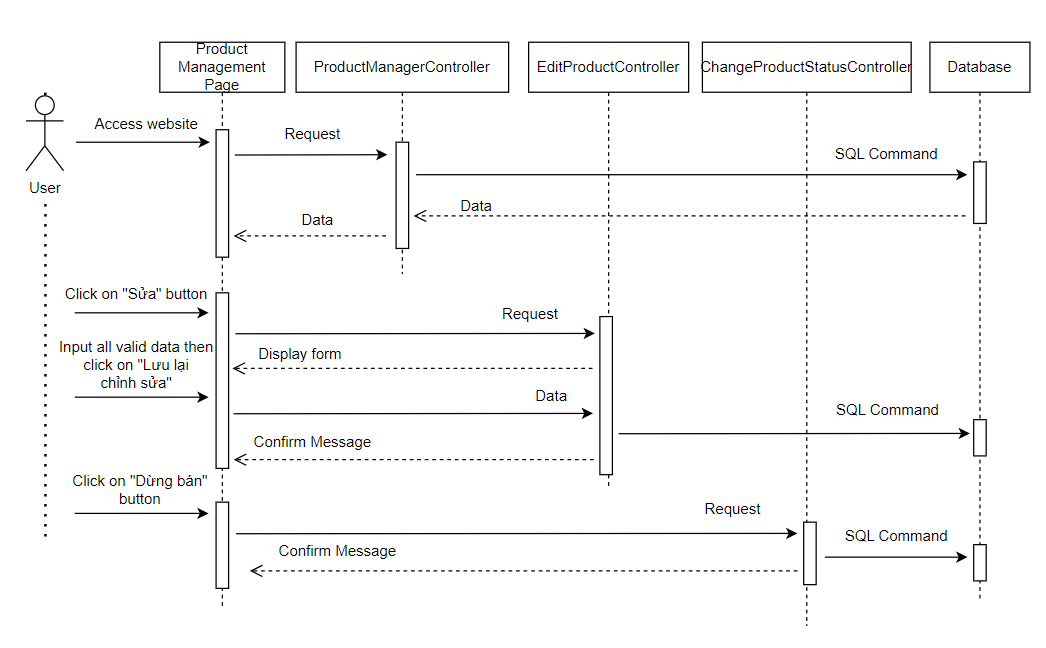
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ProductDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | totalProduct(int): int | Get number of product with specific status |
| 02 | totalProduct(int, int): int | Get number of product with specific status and type |
| 03 | getAllProduct(int status): ArrayList<Product> | Get all product |
| 04 | changeProductStatus(int, boolean): void | Change product status |
| 05 | getForAllProductByID(int): Product | Get Product by product id |
| 06 | updateProduct(Product): void | Update Product |

c. Sequences Diagram(s)



d. Database query

//count product with specific status

SELECT COUNT(PRODUCT\_ID) [TOTAL] FROM PRODUCT WHERE [STATUS] = ?

//count product with specific status and type

SELECT COUNT(PRODUCT\_ID) [TOTAL] FROM PRODUCT WHERE [TYPE] = ? AND [STATUS] = ?

//select all product

SELECT [PRODUCT\_ID], [NAME], [TYPE], [OS], [FEATURE\_PRODUCT], [COLOR],

[CURRENT\_PRICE], [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD], [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [STATUS]

FROM [DBO].[PRODUCT] WHERE STATUS = ?

//update product status with specific product

UPDATE [DBO].[PRODUCT]

SET [STATUS] = ?

WHERE PRODUCT\_ID = ?

//select all information of product

SELECT [PRODUCT\_ID], [NAME], [TYPE], [OS], [FEATURE\_PRODUCT], [COLOR],

[CURRENT\_PRICE], [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD], [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [STATUS]

FROM [DBO].[PRODUCT] WHERE [PRODUCT\_ID] = ?

//Update product information

UPDATE [DBO].[PRODUCT] SET [NAME] = ?, [TYPE] = ?, [OS] = ?, [COLOR] = ?, [CURRENT\_PRICE] = ?

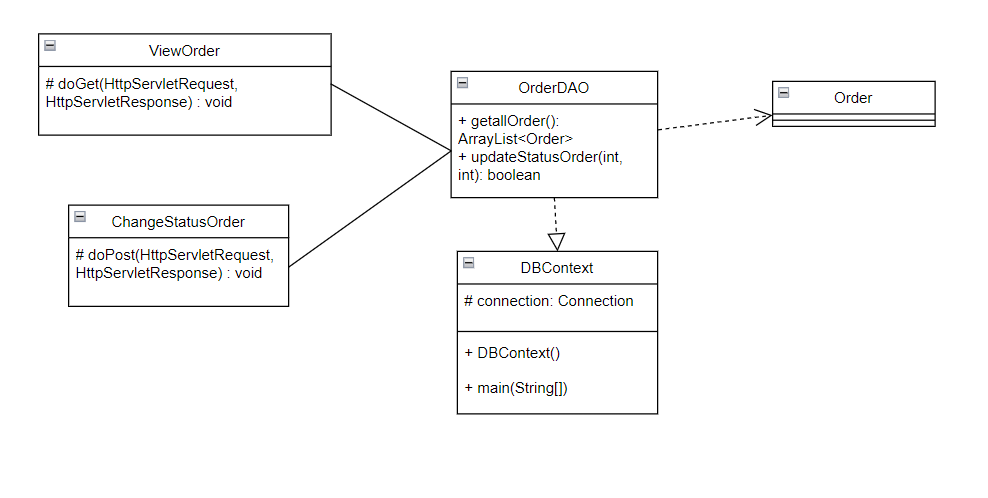
, [ORIGINAL\_PRICE] = ?, [RAM] = ?, [MEMORY] = ?, [CPU] = ?, [GRAPHICS\_CARD] = ?, [SIZE] = ?

, [DESCRIPTION] = ?, [DISCOUNT] = ?

WHERE PRODUCT\_ID = ?

## 17. Order Management

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### ViewOrder

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet(HttpServletRequest, HttpServletResponse): void | To handle request by method GET |

#### ChangeStatusOrder

***Class Methods***

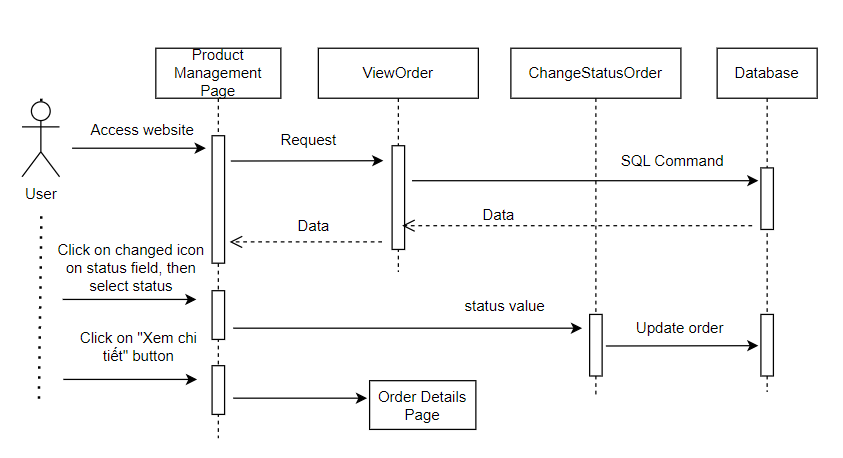
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | To handle request by method POST |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getallOrder(): ArrayList<Order> | Get all order |
| 02 | updateStatusOrder(int, int): boolean | Update order status |

c. Sequences Diagram(s)



d. Database query

//select all order

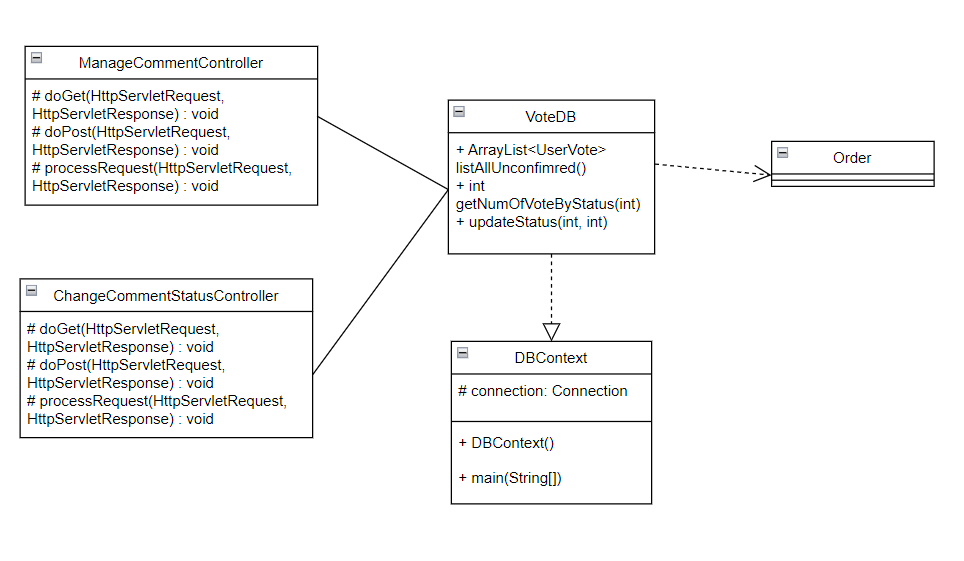
SELECT \* FROM [ORDER]

//update order status

UPDATE [ORDER] SET [STATUS] = ? WHERE ORDER\_ID = ?

## 18. Feedback Management

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### ManageCommentController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ChangeCommentStatusController

***Class Methods***

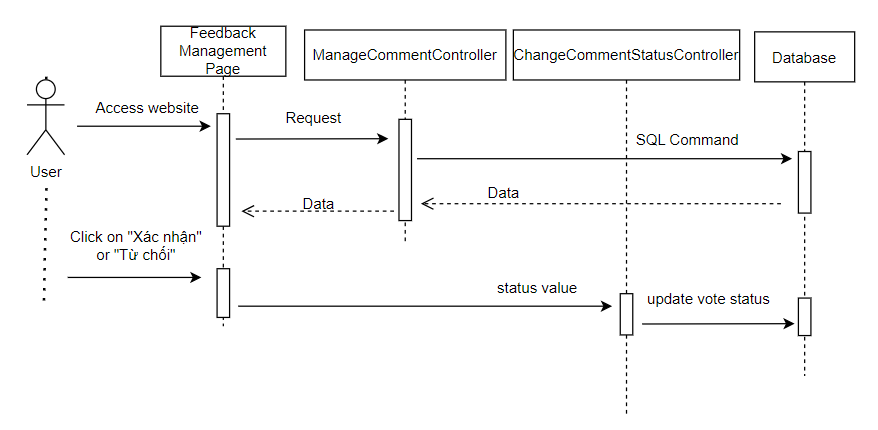
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### VoteDB

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | ArrayList<UserVote> listAllUnconfimred() | Get List of unconfirmed votes |
| 02 | int getNumOfVoteByStatus(int) | Get number of votes by status vote |
| 03 | updateStatus(int, int) | Update status of vote |

c. Sequences Diagram(s)

****

d. Database query

//select all unconfirmed vote

SELECT [VOTE\_OF\_USER\_ID], [NUM\_OF\_STAR], [PRODUCT\_ID], [COMMENT], [USERNAME]

FROM [VOTE\_OF\_USER] WHERE [CONFIRM] = 0

//select number of vote by status

SELECT COUNT(VOTE\_OF\_USER\_ID) AS NUM

FROM [VOTE\_OF\_USER] WHERE CONFIRM = ? GROUP BY [CONFIRM]

//update status of vote

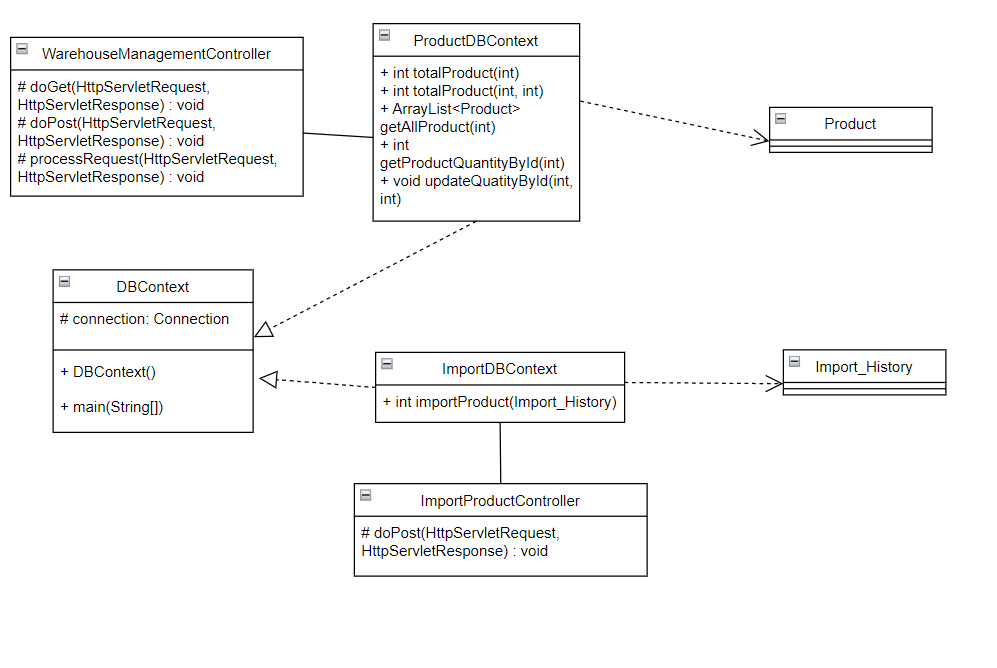
UPDATE [VOTE\_OF\_USER]

SET CONFIRM = ?

WHERE VOTE\_OF\_USER\_ID = ?

## 19. Warehouse Management

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### WarehouseManagementController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ImportProductController

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost(HttpServletRequest, HttpServletResponse): void | Handle request by POST method |

#### ProductDBContext

***Class Methods***

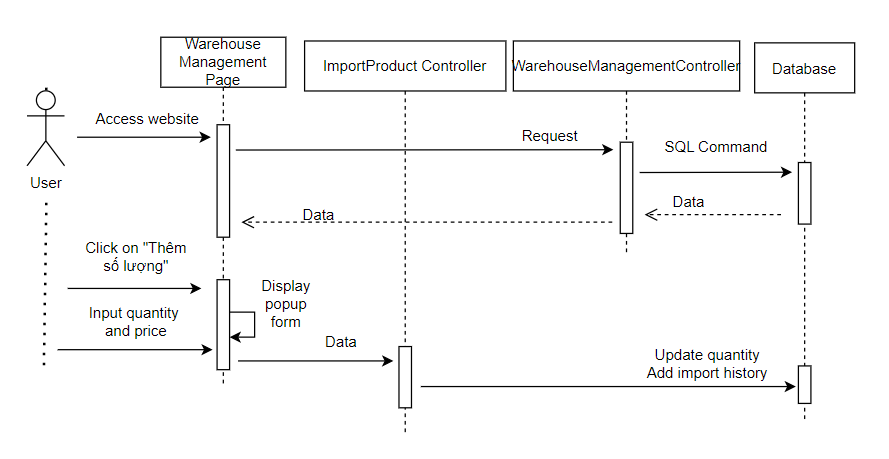
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | int totalProduct(int) | Get number of product with specific status |
| 02 | int totalProduct(int, int) | Get number of product with specific status and type |
| 03 | ArrayList<Product> getAllProduct(int) | Get all product |
| 04 | int getProductQuantityById(int) | Get product quantity by product id |
| 05 | void updateQuatityById(int, int) | Update product quantity |

#### ImportDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Int importProduct(Import\_History) | Add import history to database |

c. Sequences Diagram(s)



d. Database query

//count product with specific status

SELECT COUNT(PRODUCT\_ID) [TOTAL] FROM PRODUCT WHERE [STATUS] = ?

//count product with specific status and type

SELECT COUNT(PRODUCT\_ID) [TOTAL] FROM PRODUCT WHERE [TYPE] = ? AND [STATUS] = ?

//select all product

SELECT [PRODUCT\_ID], [NAME], [TYPE], [OS], [FEATURE\_PRODUCT], [COLOR],

[CURRENT\_PRICE], [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD], [SIZE], [DESCRIPTION], [DISCOUNT], [QTY], [STATUS]

FROM [DBO].[PRODUCT] WHERE STATUS = ?

//select quantity of product by product id

SELECT [QTY] FROM [DBO].[PRODUCT] WHERE PRODUCT\_ID = ?

//update quantity of product by product id

UPDATE [DBO].[PRODUCT] SET [QTY] = ? WHERE PRODUCT\_ID = ?

//insert new import history

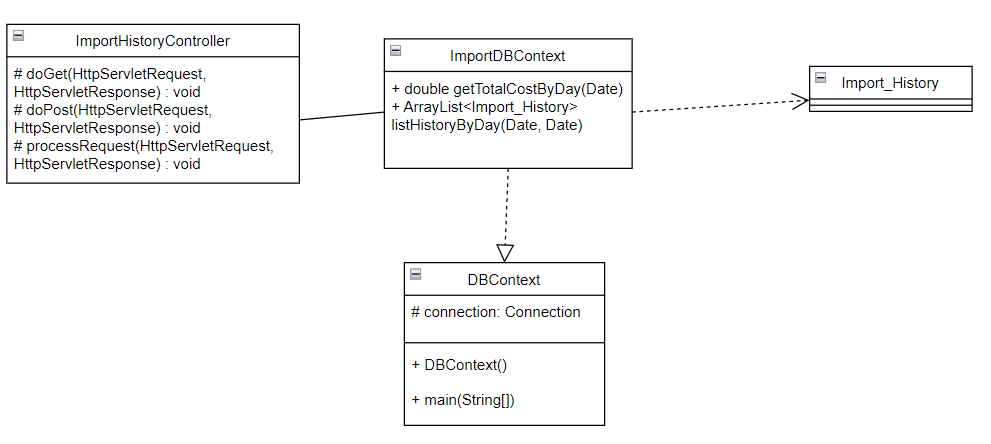
INSERT INTO [DBO].[IMPORT\_HISTORY]

([NUM], [DATE], [USERNAME], [COST], [PRODUCT\_ID])

VALUES (?,?,?,?,?)

## 20. Import History

a. Class diagram



b. Class Specifications

#### DBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext() | To connect with database |
| 02 | main(String[]) | Call DBContext() |

#### ImportHistoryController

***Class Methods***

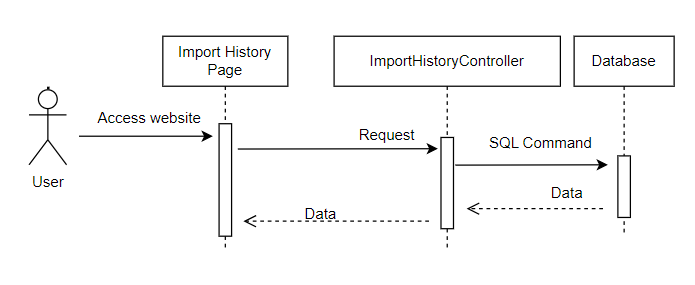
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest(HttpServletRequest, HttpServletResponse): void | To handle request, get data from database and display to jsp |
| 02 | doPost(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |
| 03 | doGet(HttpServletRequest, HttpServletResponse): void | Call processRequest(HttpServletRequest, HttpServletResponse) |

#### ImportDBContext

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | double getTotalCostByDay(Date) | Get Total Cost by day |
| 02 | ArrayList<Import\_History> listHistoryByDay(Date, Date) | Get list of import history |

c. Sequences Diagram(s)



d. Database query

//select import history by day

SELECT [IMPORT\_ID], [NUM], [DATE], [NOTE], [USERNAME], [COST], [PRODUCT\_ID]

FROM [IMPORT\_HISTORY] WHERE [DATE] BETWEEN ? AND ? ORDER BY [DATE] DESC

//select cost of import history by day

SELECT [num], [cost] FROM [Import\_History] WHERE [date] = ?