

**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) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 5](#_Toc128675678)

[1. Code Packages 5](#_Toc128675679)

[2. Database Design 6](#_Toc128675680)

[II. Code Designs 6](#_Toc128675681)

[1. Home Page – Display Product Lists 6](#_Toc128675682)

[2. Product List - Filter 9](#_Toc128675683)

[3. Search Order 12](#_Toc128675684)

[4. Login for Customer 14](#_Toc128675685)

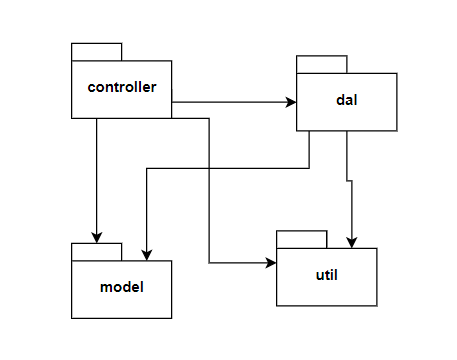
[5. <Feature/Function Name2> 17](#_Toc128675686)

[6. <Feature/Function Name2> 17](#_Toc128675687)

[7. <Feature/Function Name2> 18](#_Toc128675688)

# I. Overview

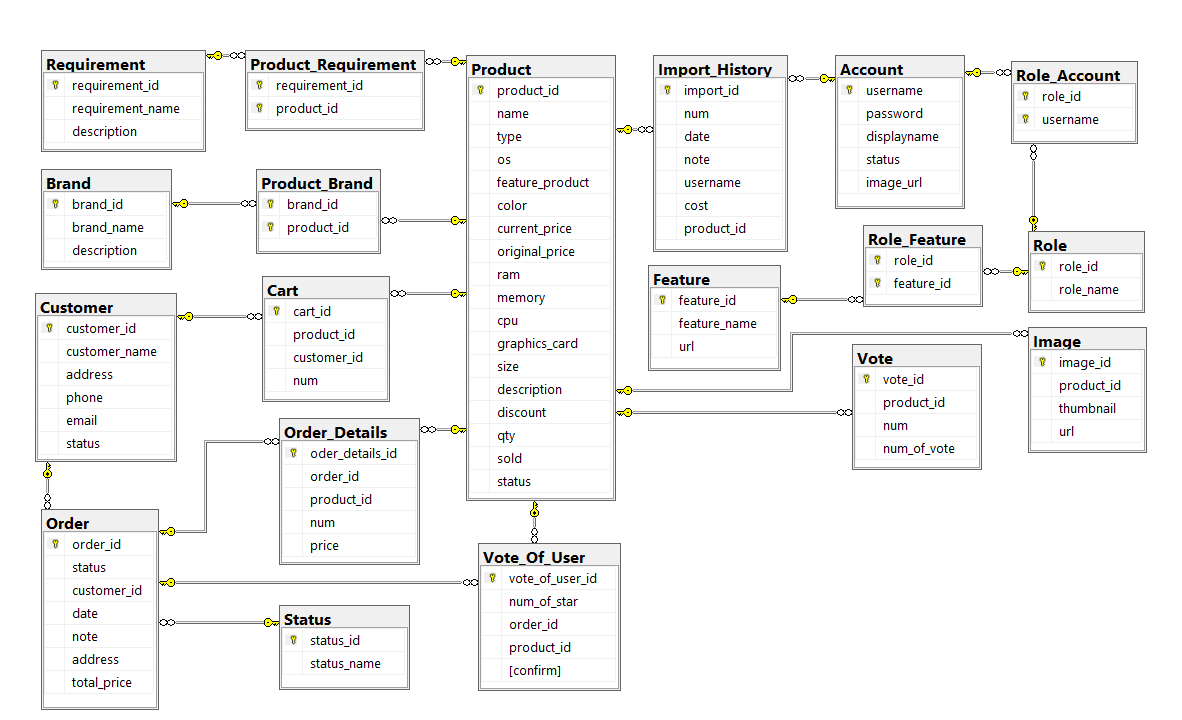
## 1. Code Packages



***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 |

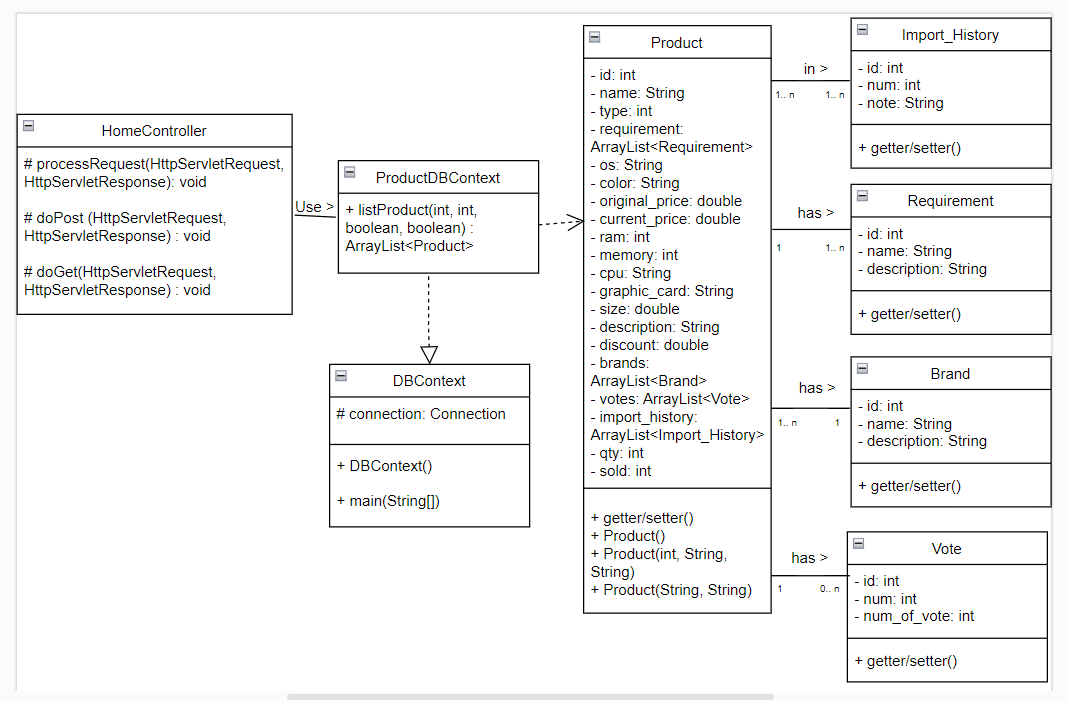
## 2. Database Design



# 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***

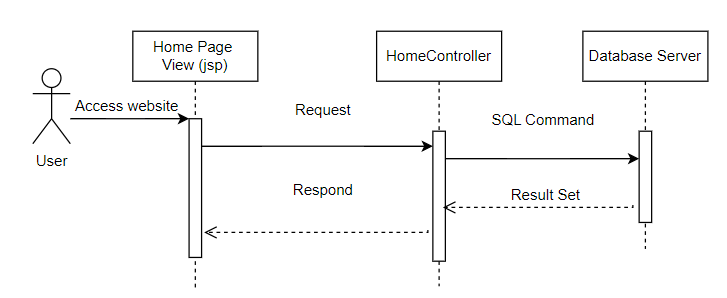
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listProduct(int, int, boolean, boolean): ArrayList<Product> | Get list of product base on: what type? (phone or laptop), how much? (number of product), Is top of sale? Is top of sold? |

#### 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]

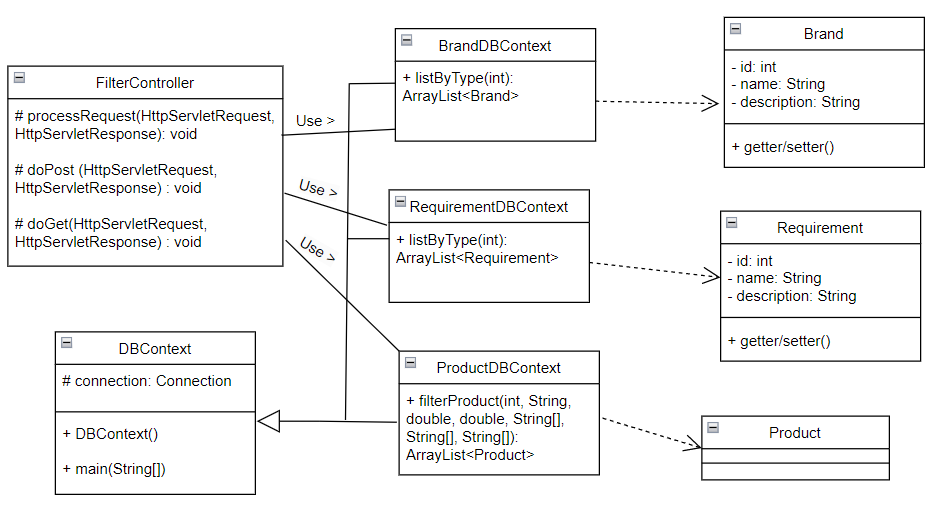
, [ORIGINAL\_PRICE], [RAM], [MEMORY], [CPU], [GRAPHICS\_CARD]

, [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***

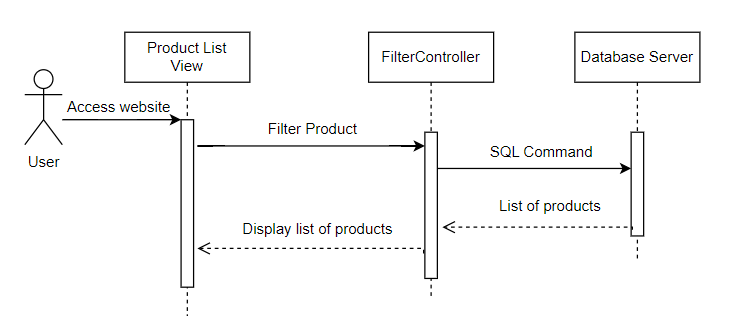
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | listByType(int): ArrayList<Requirement> | Get list of requirements by category product (phone or laptop) |

#### 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

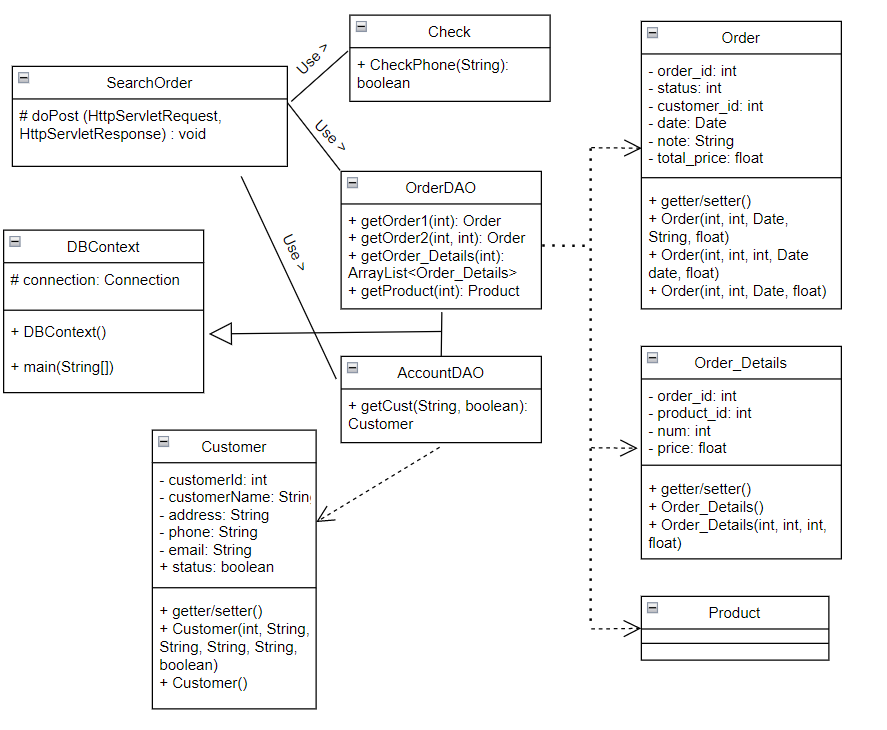
AND PR.[CURRENT\_PRICE] >= ?

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***

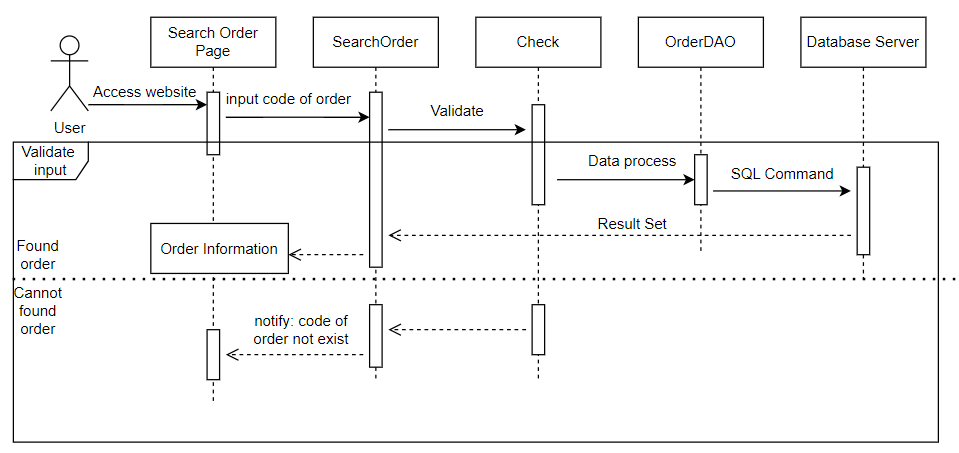
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getCust(String, boolean): Customer | Get Customer by email and status |

#### 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

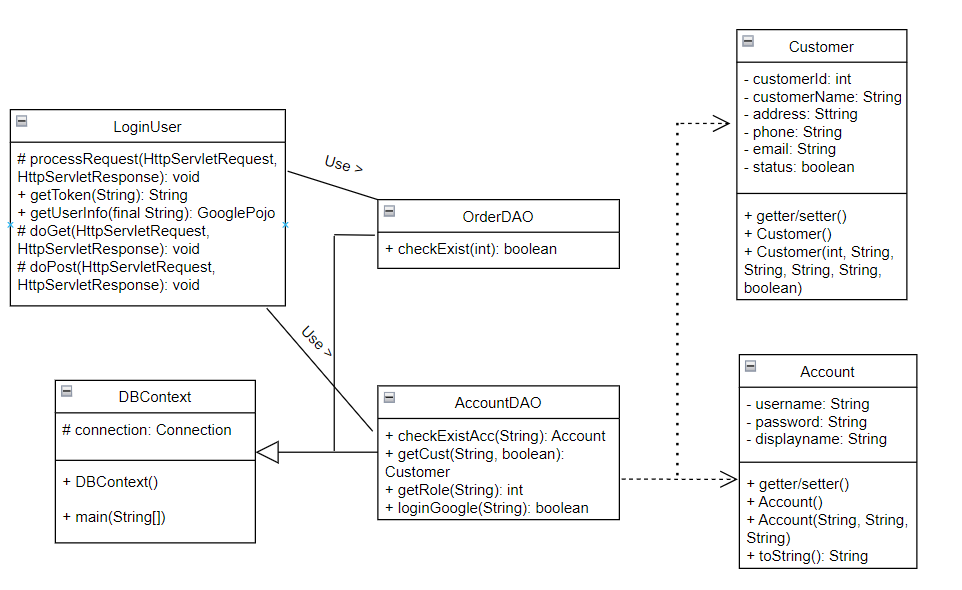
SELECT [CUSTOMER\_ID], [CUSTOMER\_NAME], [ADDRESS], [PHONE], [EMAIL], [STATUS]

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***

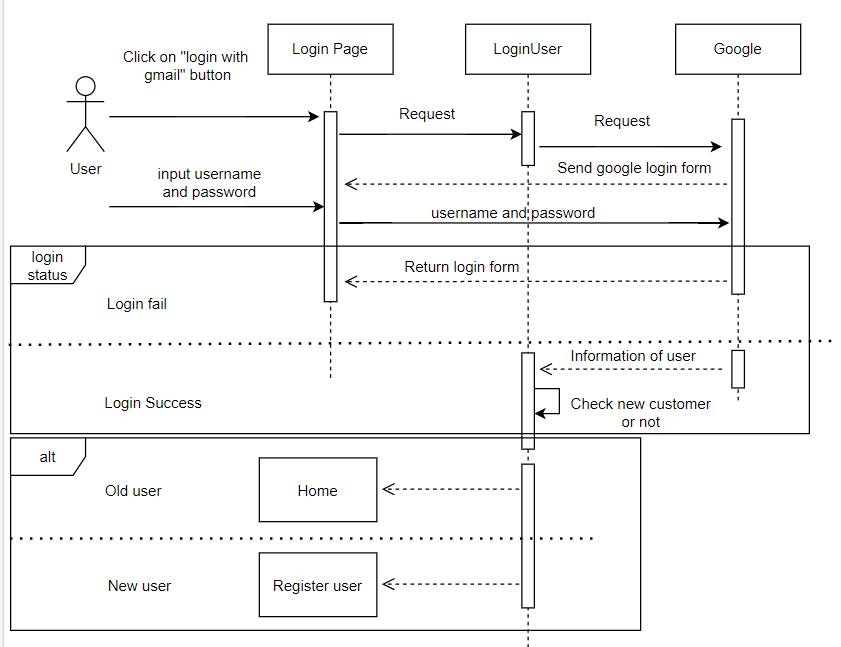
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | checkExist(int): boolean | Check user account has cart or not |

#### 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. <Feature/Function Name2>

a. Class Diagram

b. Class Specifications

#### DBContext

***Class Methods***

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

c. Sequence Diagram(s)

d. Database Queries

## 6. <Feature/Function Name2>

a. Class Diagram

b. Class Specifications

#### DBContext

***Class Methods***

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

c. Sequence Diagram(s)

d. Database Queries

## 7. <Feature/Function Name2>

a. Class Diagram

b. Class Specifications

#### DBContext

***Class Methods***

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

c. Sequence Diagram(s)

d. Database Queries