

**ONLINE SHOPPING**

**Software Design Document**

– Hanoi, September 2021 –

**Table of Contents**

[I. Overview 10](#_Toc4178)

[Code Packages/Namespaces 10](#_Toc27221)

[II. Code Designs 12](#_Toc8439)

[1. Feature Product 12](#_Toc3583)

[a. Class Diagram 12](#_Toc14155)

[b. Class Specifications 12](#_Toc16559)

[c. Sequence Diagram(s) 13](#_Toc20830)

[d. Database queries 13](#_Toc8883)

[2. Search & Filter Product 14](#_Toc18106)

[a. Class Diagram 14](#_Toc27648)

[b. Class Specifications 14](#_Toc31698)

[c. Sequence Diagram(s) 15](#_Toc29279)

[d. Database queries 15](#_Toc28127)

[a. Class Diagram 16](#_Toc10839)

[b. Class Specifications 16](#_Toc17082)

[c. Sequence Diagram(s) 16](#_Toc16415)

[d. Database queries 17](#_Toc9331)

[4. Change Amount 17](#_Toc4441)

[a. Class Diagram 17](#_Toc11959)

[b. Class Specifications 17](#_Toc21859)

[c. Sequence Diagram(s) 18](#_Toc9276)

[d. Database queries 18](#_Toc29061)

[5. Delete Product From Cart 19](#_Toc5335)

[a. Class Diagram 19](#_Toc8012)

[b. Class Specifications 19](#_Toc8416)

[c. Sequence Diagram(s) 20](#_Toc32445)

[d. Database queries 20](#_Toc11532)

[6. Delete All Cart Item 21](#_Toc15193)

[a. Class Diagram 21](#_Toc12791)

[b. Class Specifications 21](#_Toc25090)

[c. Sequence Diagram(s) 22](#_Toc27869)

[d. Database queries 22](#_Toc14146)

[7. Login 23](#_Toc9059)

[a. Class Diagram 23](#_Toc6272)

[b. Class Specifications 23](#_Toc31324)

[c. Sequence Diagram(s) 24](#_Toc2084)

[d. Database queries 24](#_Toc8164)

[8. Register 25](#_Toc17354)

[a. Class Diagram 25](#_Toc19288)

[b. Class Specifications 25](#_Toc31967)

[c. Sequence Diagram(s) 26](#_Toc27111)

[d. Database queries 26](#_Toc20326)

[9. View User List 27](#_Toc17619)

[a. Class Diagram 27](#_Toc12220)

[b. Class Specifications 27](#_Toc5791)

[c. Sequence Diagram(s) 28](#_Toc3980)

[d. Database queries 28](#_Toc28342)

[10. Edit User 29](#_Toc15188)

[a. Class Diagram 29](#_Toc32599)

[b. Class Specifications 29](#_Toc15659)

[c. Sequence Diagram(s) 31](#_Toc4036)

[d. Database queries 31](#_Toc13997)

[11. Delete User 31](#_Toc23815)

[a. Class Diagram 31](#_Toc29485)

[b. Class Specifications 31](#_Toc17869)

[c. Sequence Diagram(s) 33](#_Toc15525)

[d. Database queries 33](#_Toc16933)

[12. Customers List[Seller] 34](#_Toc4348)

[a. Class Diagram 34](#_Toc28196)

[b. Class Specifications 34](#_Toc660)

[c. Sequence Diagram(s) 35](#_Toc26301)

[d. Database queries 35](#_Toc9797)

[13. Add Ship Information 35](#_Toc29267)

[a. Class Diagram 35](#_Toc16941)

[b. Class Specifications 35](#_Toc31428)

[c. Sequence Diagram(s) 36](#_Toc19452)

[14. Change Ship Information 36](#_Toc19724)

[a. Class Diagram 36](#_Toc31397)

[b. Class Specifications 37](#_Toc29354)

[c. Sequence Diagram(s) 37](#_Toc3744)

[d. Database queries 37](#_Toc23395)

[a. Class Diagram 38](#_Toc19736)

[b. Class Specifications 38](#_Toc558)

[c. Sequence Diagram(s) 39](#_Toc19904)

[16. Change Password 40](#_Toc20027)

[a. Class Diagram 40](#_Toc30760)

[b. Class Specifications 40](#_Toc14710)

[c. Sequence Diagram(s) 41](#_Toc13502)

[d. Database queries 41](#_Toc20478)

[17. User Profile 42](#_Toc14055)

[a. Class Diagram 42](#_Toc30815)

[b. Class Specifications 42](#_Toc12499)

[c. Sequence Diagram(s) 43](#_Toc12946)

[18. My Orders 43](#_Toc1783)

[a. Class Diagram 43](#_Toc11282)

[b. Class Specifications 43](#_Toc2131)

[c. Sequence Diagram(s) 44](#_Toc14728)

[19. Order Information 45](#_Toc6639)

[a. Class Diagram 45](#_Toc2776)

[c. Sequence Diagram(s) 46](#_Toc30514)

[d. Database queries 46](#_Toc21654)

[d. Database queries 49](#_Toc30209)

[22. Order Details 50](#_Toc16943)

[a. Class Diagram 50](#_Toc29896)

[b. Class Specifications 50](#_Toc24135)

[c. Sequence Diagram(s) 51](#_Toc2522)

[d. Database queries 51](#_Toc19229)

[23. View Feedback 52](#_Toc1745)

[a. Class Diagram 52](#_Toc13041)

[b. Class Specifications 52](#_Toc14282)

[c. Sequence Diagram(s) 53](#_Toc22724)

[d. Database queries 53](#_Toc17848)

[24. Send Feedback 53](#_Toc13288)

[a. Class Diagram 53](#_Toc7205)

[b. Class Specifications 53](#_Toc5431)

[c. Sequence Diagram(s) 54](#_Toc30229)

[d. Database queries 54](#_Toc2801)

[25. Blogs List 55](#_Toc4841)

[a. Class Diagram  55](#_Toc8332)

[b. Class Specifications 55](#_Toc23412)

[c. Sequence Diagram(s) 56](#_Toc16889)

[d. Database queries 56](#_Toc16544)

[26. Blog Details 56](#_Toc1586)

[a. Class Diagram 56](#_Toc10328)

[b. Class Specifications 56](#_Toc775)

[c. Sequence Diagram(s) 57](#_Toc8840)

[d. Database queries 57](#_Toc7301)

[27. Blog Management 58](#_Toc12159)

[a. Class Diagram 58](#_Toc5232)

[b. Class Specifications 58](#_Toc27122)

[c. Sequence Diagram(s) 59](#_Toc672)

[d. Database queries 59](#_Toc14952)

[28. View Notification 60](#_Toc25930)

[a. Class Diagram 60](#_Toc9960)

[b. Class Specifications 60](#_Toc12402)

[c. Sequence Diagram(s) 61](#_Toc10515)

[d. Database queries 61](#_Toc15906)

[29. Mark as Read 62](#_Toc1554)

[a. Class Diagram 62](#_Toc19733)

[b. Class Specifications 62](#_Toc16136)

[c. Sequence Diagram(s) 63](#_Toc9691)

[d. Database queries 63](#_Toc16218)

[30. Admin Feature 64](#_Toc16794)

[a. Class Diagram 64](#_Toc27469)

[b. Class Specifications 64](#_Toc18127)

[c. Sequence Diagram(s) 65](#_Toc11433)

[d. Database queries 65](#_Toc19309)

[31. Add Product 66](#_Toc4573)

[a. Class Diagram 66](#_Toc23367)

[b. Class Specifications 66](#_Toc23048)

[c. Sequence Diagram(s) 67](#_Toc31030)

[d. Database queries 67](#_Toc1039)

[32. Delete Product 68](#_Toc14921)

[a. Class Diagram 68](#_Toc9081)

[b. Class Specifications 68](#_Toc21442)

[c. Sequence Diagram(s) 69](#_Toc4365)

[d. Database queries 69](#_Toc31146)

[33. Edit Product 70](#_Toc32137)

[a. Class Diagram 70](#_Toc6970)

[b. Class Specifications 70](#_Toc24301)

[c. Sequence Diagram(s) 71](#_Toc13173)

[d. Database queries 71](#_Toc13606)

[34. Orders Dashboard 71](#_Toc22437)

[d. Database queries 74](#_Toc18727)

[35. PostDetail 75](#_Toc14993)

[b. Class Specifications 75](#_Toc25009)

[PostControl Class: The doGet function is used to get the ID of the post and then use the doPost function to edit the product 75](#_Toc21166)

[DeletepostControll Class: use process request function to get and delete post by ID 76](#_Toc845)

[c. Sequence Diagram(s) 76](#_Toc669)

[d. Database queries 77](#_Toc15700)

[b. Class Specifications 78](#_Toc24231)

[PostManageControl Class: This class process request for list of post create by user ID 78](#_Toc21494)

[d. Database queries 79](#_Toc16159)

[37. Error 80](#_Toc22422)

[a. Class Diagram 80](#_Toc3791)

[b. Class Specifications 80](#_Toc31740)

[c. Sequence Diagram(s) 80](#_Toc32222)

[d. Database queries 80](#_Toc22060)

[38. Manage Feedback [Seller] 81](#_Toc17616)

[a. Class Diagram 81](#_Toc7655)

[b. Class Specifications 81](#_Toc16524)

[c. Sequence Diagram(s) 82](#_Toc14354)

[d. Database queries 82](#_Toc14220)

[39. Manage Feedback [Customer] 82](#_Toc14760)

[a. Class Diagram 82](#_Toc22952)

[b. Class Specifications 82](#_Toc20187)

[c. Sequence Diagram(s) 83](#_Toc22297)

[d. Database queries 83](#_Toc4235)

[40. Manage Banner 84](#_Toc4259)

[a. Class Diagram 84](#_Toc25158)

[b. Class Specifications 84](#_Toc12810)

[c. Sequence Diagram(s) 85](#_Toc1383)

[d. Database queries 85](#_Toc20802)

[41. Chat 86](#_Toc12817)

[a. Class Diagram 86](#_Toc16369)

[b. Class Specifications 86](#_Toc19735)

[c. Sequence Diagram 88](#_Toc27203)

[43. Marketing Dashboard 90](#_Toc18962)

[a. Class Diagram 90](#_Toc13602)

[b. Class Specifications 90](#_Toc814)

[c. Sequence Diagram(s) 91](#_Toc8003)

[d. Database queries 91](#_Toc23683)

[43. DeliveryJob 92](#_Toc17337)

[a. Class Diagram 92](#_Toc30439)

[b. Class Specifications 92](#_Toc4575)

[c. Sequence Diagram(s) 93](#_Toc5790)

[d. Database queries 93](#_Toc28126)

[44. Delivery Details 94](#_Toc17499)

[a. Class Diagram 94](#_Toc17946)

[b. Class Specifications 94](#_Toc9066)

[c. Sequence Diagram(s) 95](#_Toc19783)

[d. Database queries 95](#_Toc3575)

[III. Database Tables 96](#_Toc17918)

[1. Database Schema 96](#_Toc20692)

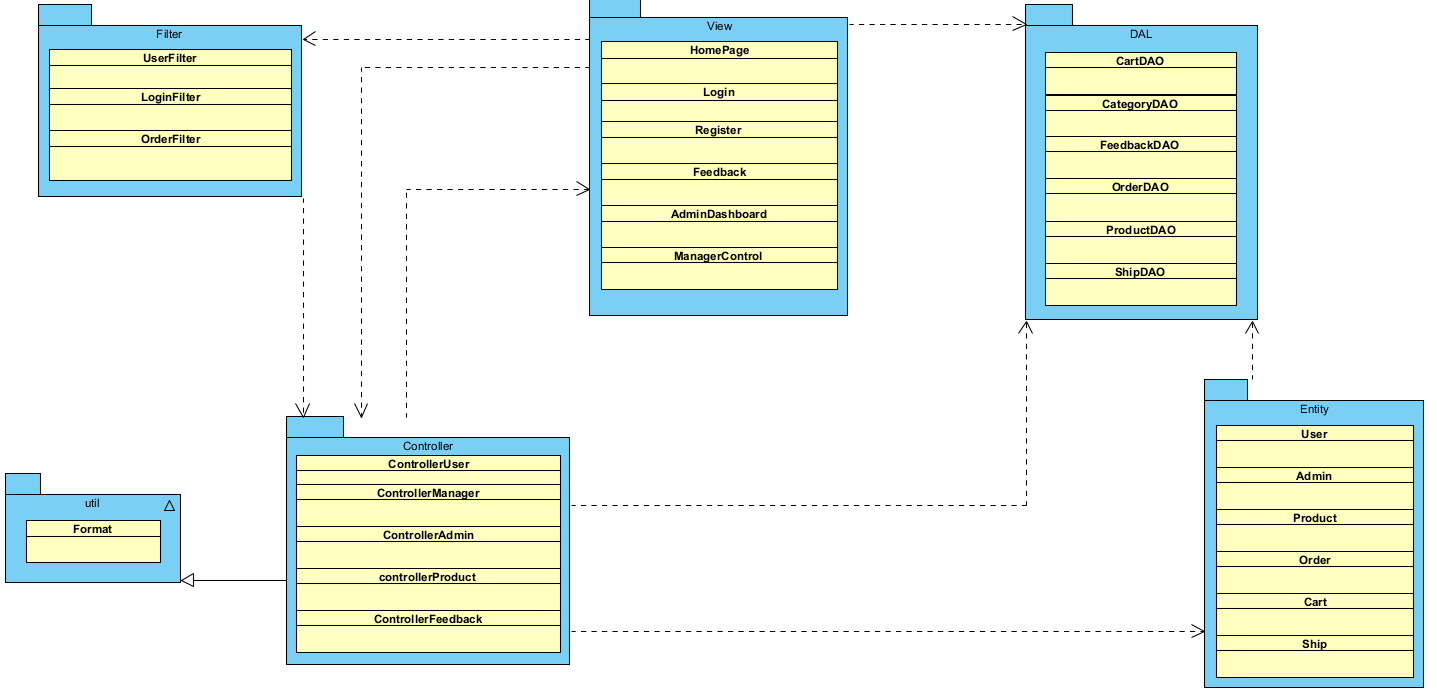
[2. Table Description 96](#_Toc29695)

# I. Overview

Technical

* Tool
* netbeans 8.0
* ms sql server 2018
* Framework
* bootstrap 5.0.2
* jquery

## Code Packages/Namespaces



w

***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *01* | *View(Web Pages)* | *Provide interaction with user ,GUI*  *Java Class Naming Convention* |
| *02* | *Controller* | *Control the data flow between view and model (DAO),Control the logic of the system*  *Java Class Naming Convention* |
| *03* | *DAL* | *Provide the connection to the Database, Extract data for the system* |
| *04* | *Filter* | *Control the redirection between Pages in View and from View to Controller*  *Java Class Naming Convention* |
| *05* | *Entity* | *Define the logic of data extracted from Database*  *Java Class Naming Convention* |
| *06* | *Utity* | *Utility Function* |

# 

# 

# II. Code Designs

## 1. Feature Product

### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAllProduct()* | *Query will get all product from database* |

### c. Sequence Diagram(s)

**

### d. Database queries

* select p.ProductID , ProductName , Description , OriginalPrice , SellPrice , SalePercent ,SubCategoryID , SellerID , Amount , StatusID , StatusID ,BrandID , height , width ,weight , s.ProductImgURL from Product p join ProductImg s on p.ProductID = s.ProductID

## 2. Search & Filter Product

### a. Class Diagram



### b. Class Specifications

Product

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getProductQuery(Query,subcate,brand,price,sortType,sortMode)* | *Query product from database with input parameter* |

### c. Sequence Diagram(s)

**

### d. Database queries

* SELECT \* FROM dbo.ProductImg where ProductID = ?
* SELECT p.ProductID , ProductName , Description , OriginalPrice , SellPrice , SalePercent , SubCategoryID , SellerID , Amount , StatusID , StatusID ,BrandID , height , width ,weight , s.ProductImgURL from Product p join ProductImg s on p.ProductID = s.ProductID WHERE p.ProductName LIKE ? AND p.StatusID!= 2 AND p.Amount>0 [ AND SubCategoryID = ? ] [ AND BrandID = ? ] [AND p.SellPrice< ? AND p.SellPrice>= ? ] [ ORDER BY [ parameter ] ] [ ASC /DESC ]

**3. View Product Detail**

### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetProductDetailsByProductID* | *Traverse from beginning to end all rows in the product table in the database* |

### c. Sequence Diagram(s)



### 

### d. Database queries

SELECT \* from Product

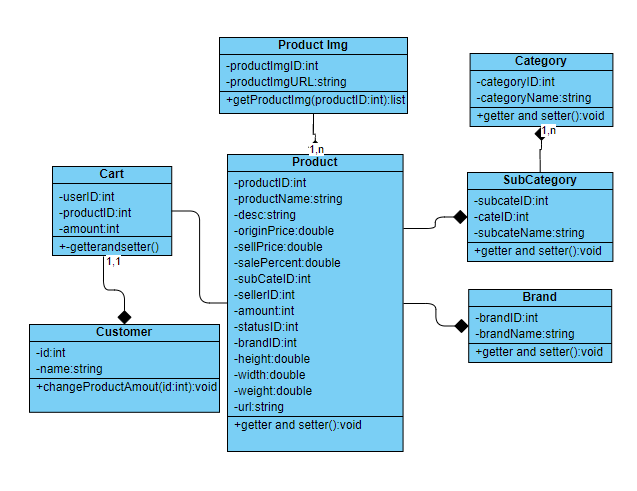
JOIN ProductImg

ON Product.ProductID = ProductImg.ProductID

WHERE Product.ProductID = ?

## 4. Change Amount

### a. Class Diagram

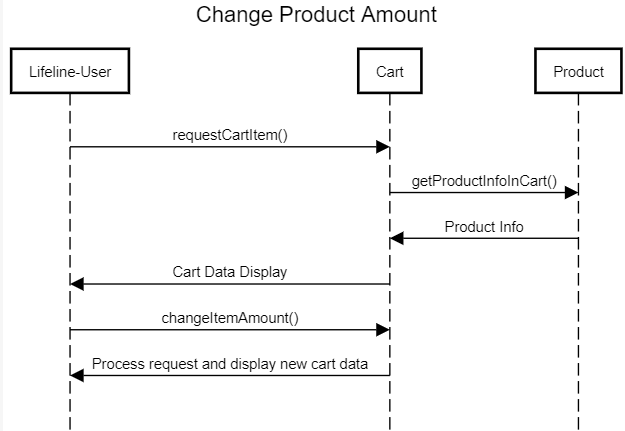


### b. Class Specifications

Customer

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *changeProductAmount()* | *Add or reduce amount of a product from cart* |

### c. Sequence Diagram(s)



### d. Database queries

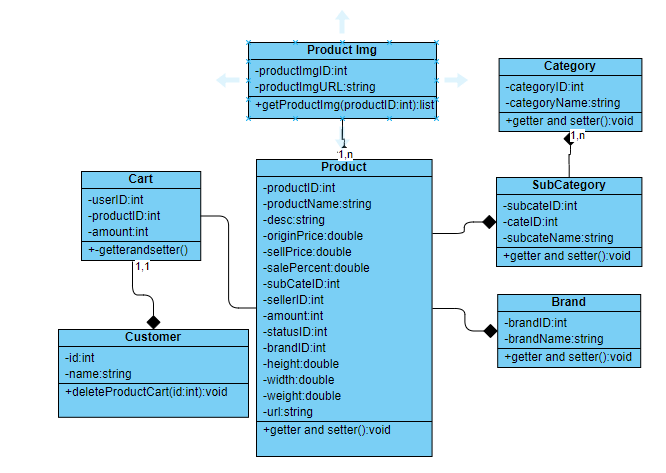
UPDATE dbo.cart

SET amount = ?

WHERE userID = ? and productID = ?

## 5. Delete Product From Cart

### a. Class Diagram

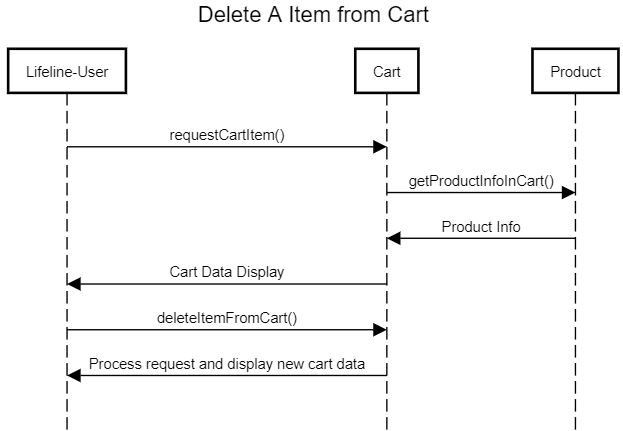


### b. Class Specifications

Customer

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *deleteProductCart()* | *Remove a product() from cart* |

### c. Sequence Diagram(s)

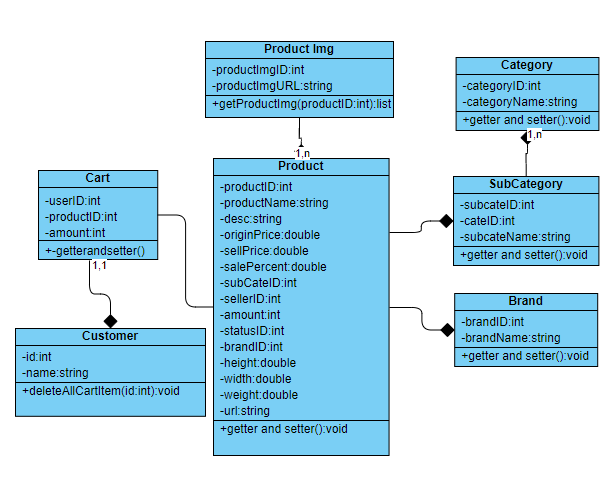


### d. Database queries

DELETE FROM *dbo.cart* WHERE *userID= ? and* productID = ?

## 6. Delete All Cart Item

### a. Class Diagram

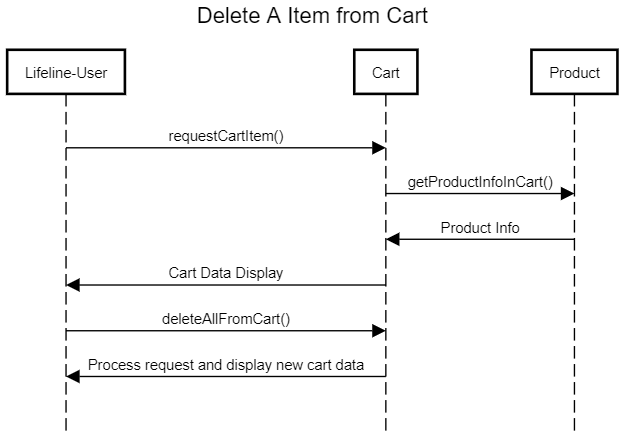


### b. Class Specifications

Customer

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *deleteAllCartItem()* | *Remove every item from cart* |

### c. Sequence Diagram(s)

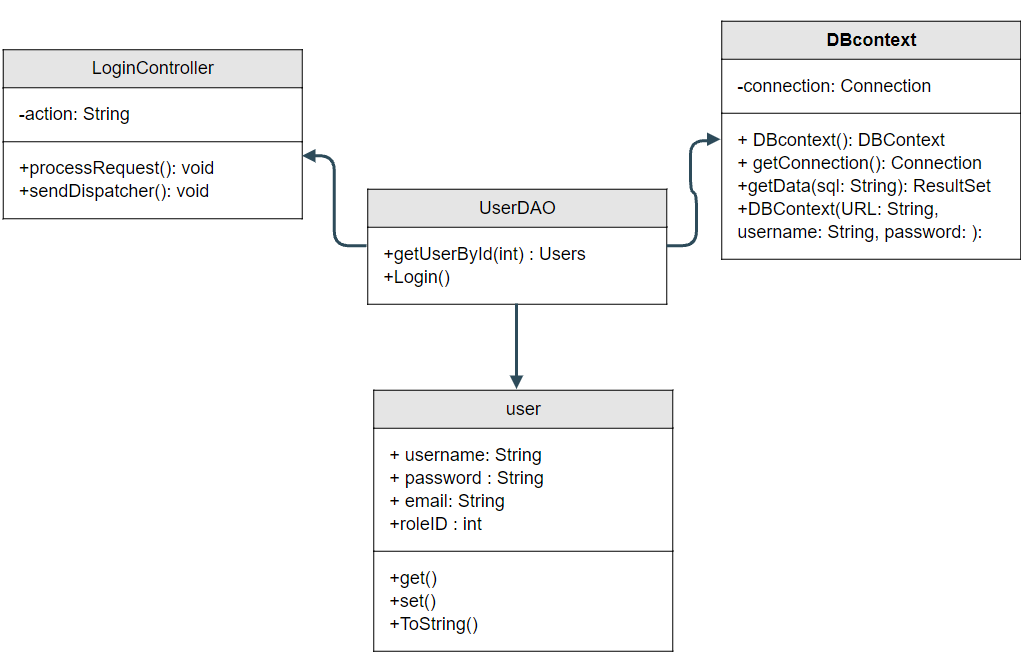


### d. Database queries

DELETE FROM *dbo.Cart* WHERE *UserID = ?*;

## 7.

### a. Class Diagram



### b. Class Specifications

UserController

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | get all new information from user Form and update the database |
|  |  |  |

UserDAO

[Provide the detailed description for the class methods]

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getUserList() | get the list of user |
| 02 | getUserById(int uId) | get the user where the user id is |

### 

### c. Sequence Diagram(s)

****

### d. Database queries

**"select \* from [User] where username=? and password=?";**

## 8. Register

### a. Class Diagram



### b. Class Specifications

UserDAO

[Provide the detailed description for the class methods]

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | Register( String username,String password, String mai) | insert a new user with Name , Pass, Mail) |
| 02 | checkAccount(String xName) | - check if the username is already exist  - return null |

UserController

Class Methods

[Provide the detailed description for the class methods]

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest | get all new information from user Form and update the database |

### 

### c. Sequence Diagram(s)

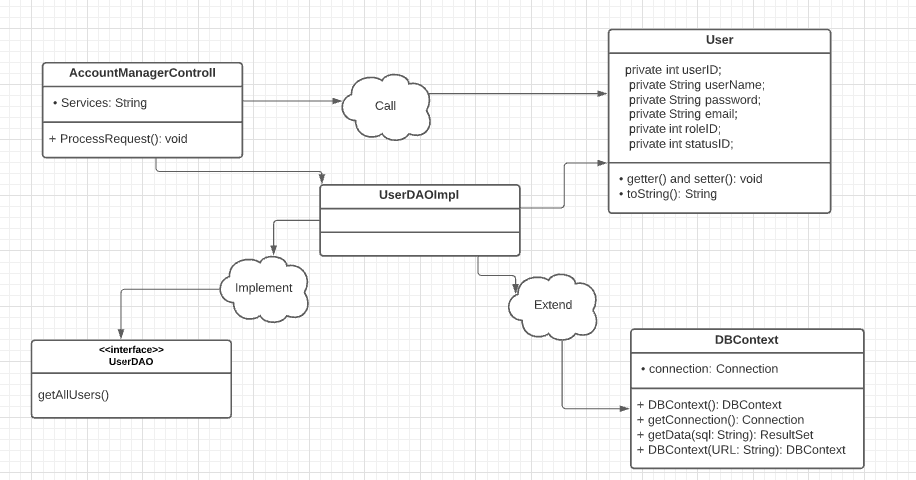
****

### d. Database queries

**+insert into [User] value(?,?,?,?,3,1)**

## 9. View User List

### a. Class Diagram



### b. Class Specifications

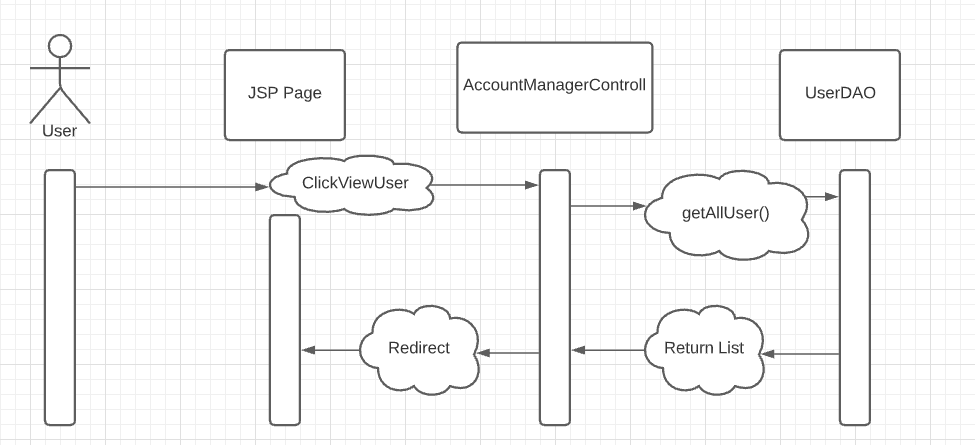
* AccountManagerControll:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing all users and put that data to jsp page* |

* UserDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getAllUsers() | - Return: List<User>  - Parameter: No  - Get all the users in the database and return a list of users |

### c. Sequence Diagram(s)

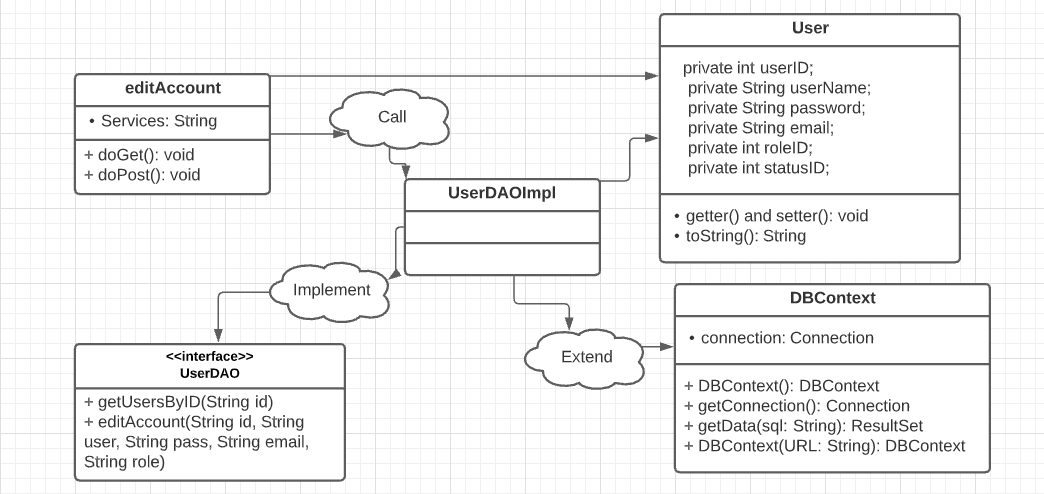


### d. Database queries

* select \* from Users

## 10. Edit User

### a. Class Diagram



### b. Class Specifications

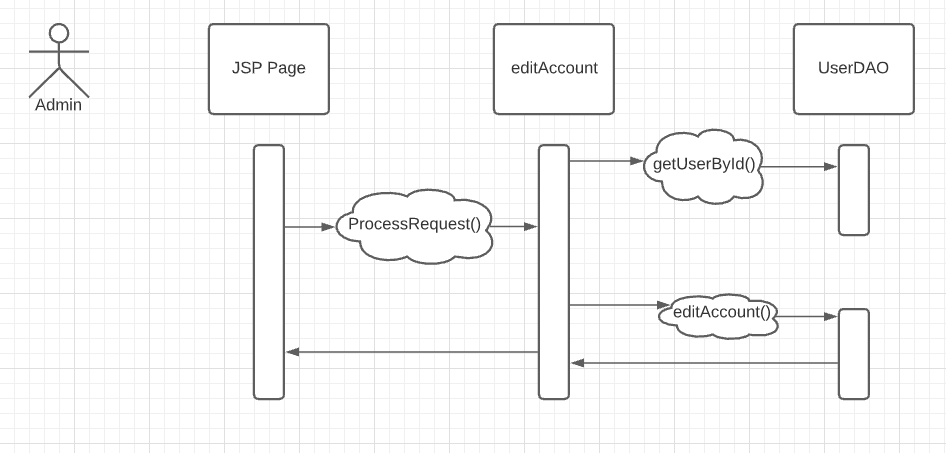
* editAccount:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Input: Service (createUser)*  - *Get data from Form in AccountManager.JSP page and process them, and update that User into DB and direct to the AccountManager.JSP page* |
| 02 | doGet() | * Get user by UserID from the database, and load current his/her information to textbox |
| 03 | doPost() | * Update User’s information in to database |

* UserDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | editAccount | - *Return: new User*  - *Input: All properties of entity User*  - *Update the User entered on AccountManager.JSP page to database* |
| 02 | getUsersByID | - *Return: ‘A user’ or ‘Null’*  - *Input: UserID*  - *Find User by UserID* |

### c. Sequence Diagram(s)

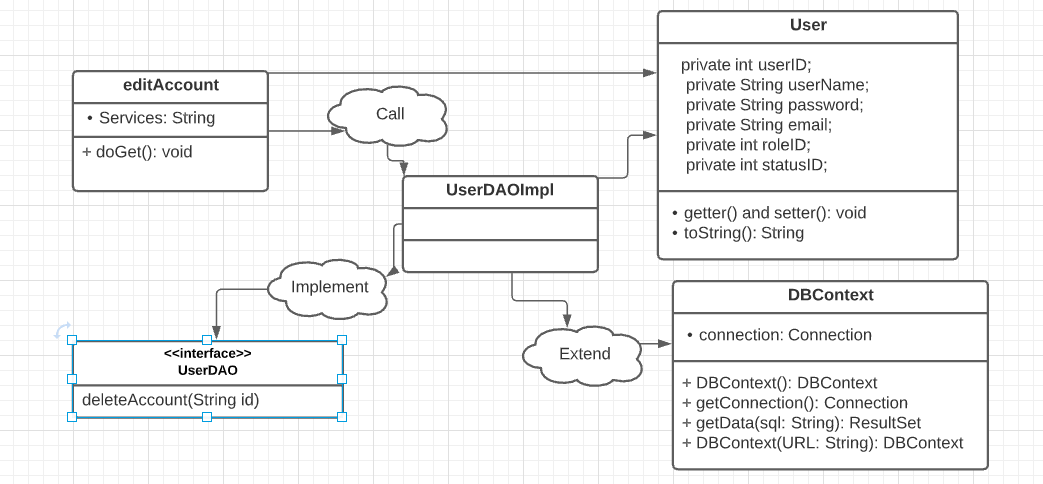


### d. Database queries

* select \* from Users where UserID = ?
* UPDATE Users SET Username = ?, Password = ?, email = ?, RoleID = ? WHERE UserID = ?

## 11. Delete User

### a. Class Diagram



### b. Class Specifications

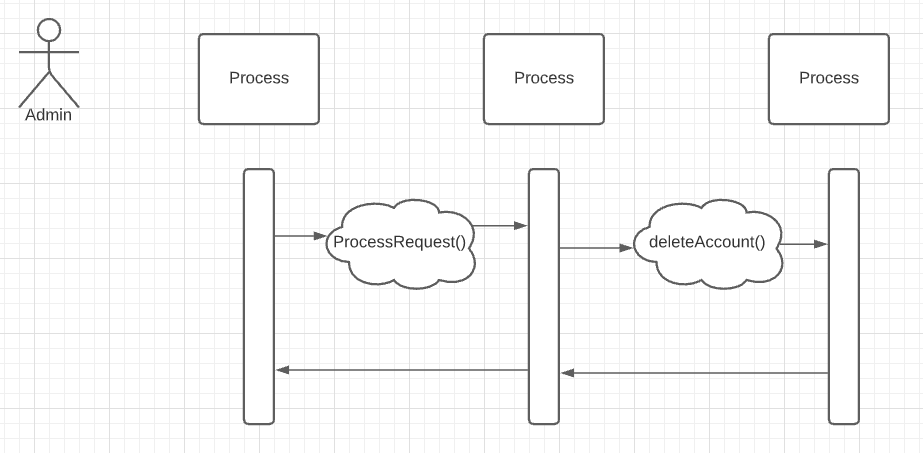
* deleteAccount:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Input: Service (deleteUser)*  - *Get data from Form in AccountManager.JSP page and process them, and delete that User into DB and direct to the AccountManager.JSP page* |

* UserDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | deleteAccount | - *Return: New list of user*  - *Input: UserID of a user*  - *Delete the User entered on AccountManager.JSP page to database* |

### c. Sequence Diagram(s)

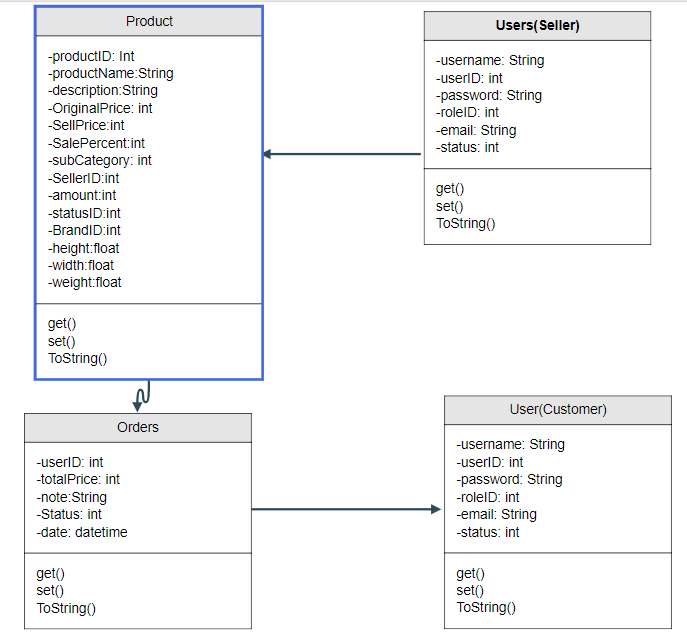


### d. Database queries

* delete from Users where UserID = ?

## 12. Customers List[Seller]

### a. Class Diagram

****

### b. Class Specifications

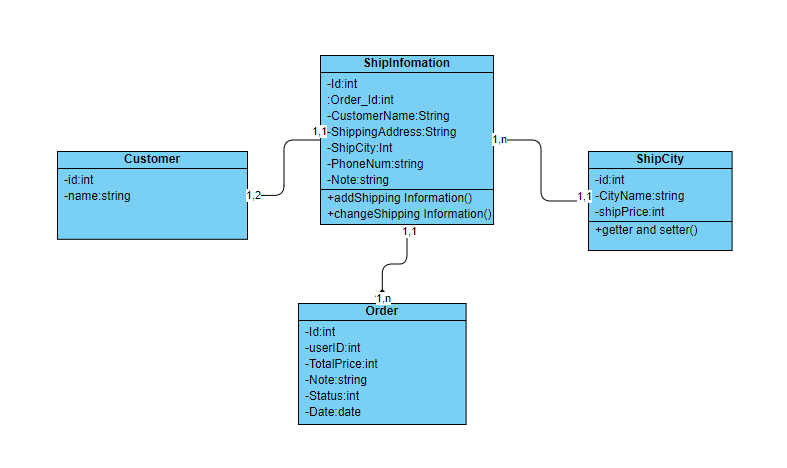
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getUserByUsername() | *get user from dtb by username* |

### c. Sequence Diagram(s)

### d. Database queries

## 13. Add Ship Information

### a. Class Diagram

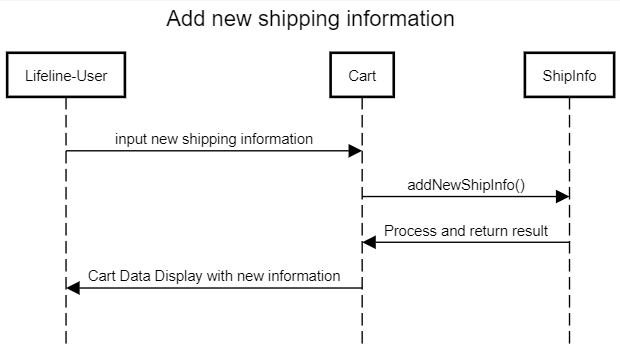


### b. Class Specifications

Ship Information Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *addShipping information()* | *Add New Shipping information for account* |

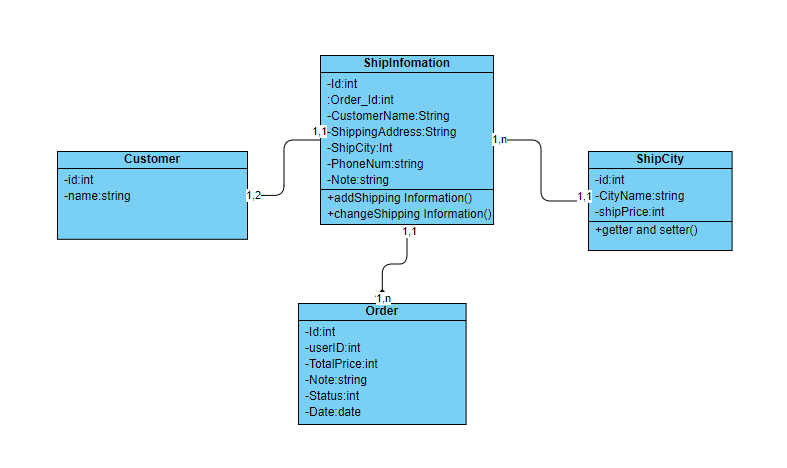
### c. Sequence Diagram(s)

Prod. Database queries

INSERT INTO dbo.ShipInfo(order\_id,customerName,shippingAddress,ShipCity,PhoneNum,Note) VALUES(?,?,?,?,?,?)

## 14. Change Ship Information

### a. Class Diagram

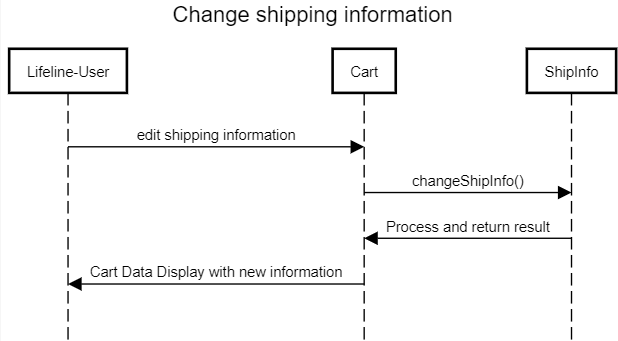


### b. Class Specifications

Ship Information Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *changeShipping information()* | *Change existing shipping information for current order* |

### c. Sequence Diagram(s)



### d. Database queries

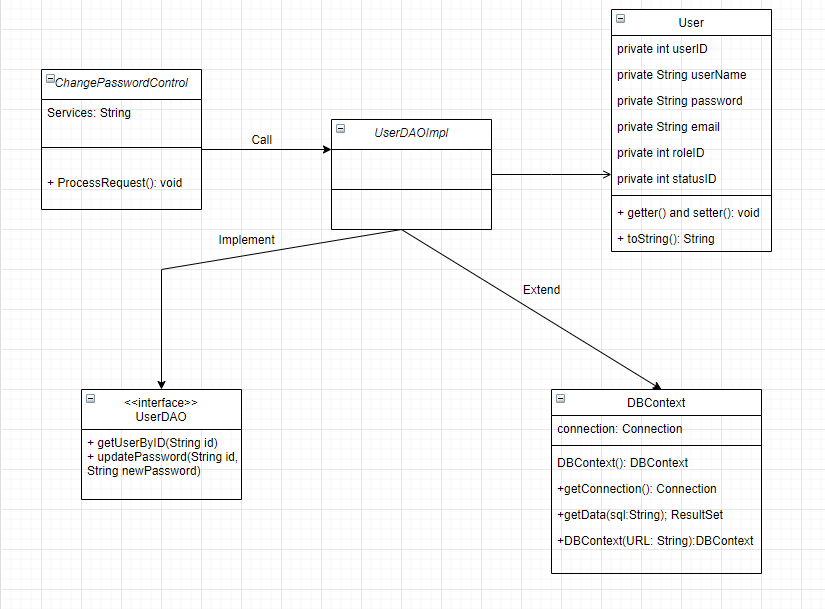
Update dbo.ShipInfomation

SET CustomerName=?,ShippingAddress=?,ShipCity=?,PhoneNum=?,Note=?

Where orderID = ?

15. Reset Password

### a. Class Diagram



### b. Class Specifications

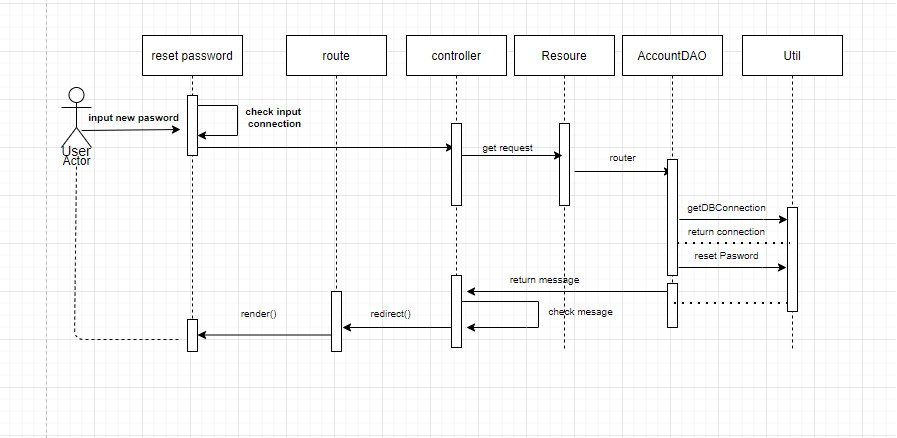
* ResetPasswordControl:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  **- Parameter: HttpServletRequest request, HttpServletResponse response** |

* UserDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getUsersByID() | - *Return: ‘A user’ or ‘Null’*  - *Input: UserID*  - *Find User by UserID* |
| 02 | updatePassword() | - *Return: void*  - Parameter: input string  - *Input: UserID, newPassword*  - *Update this new password into database* |

### c. Sequence Diagram(s)

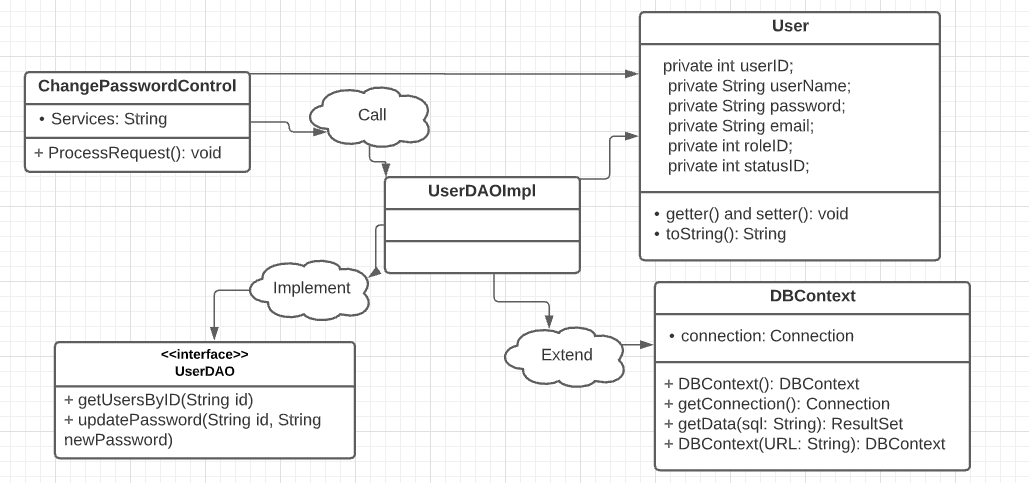


**d. Database queries**

* select \* from Users where UserID = ?
* UPDATE Users SET Password = ? WHERE UserID = ?

## 16. Change Password

### a. Class Diagram



### b. Class Specifications

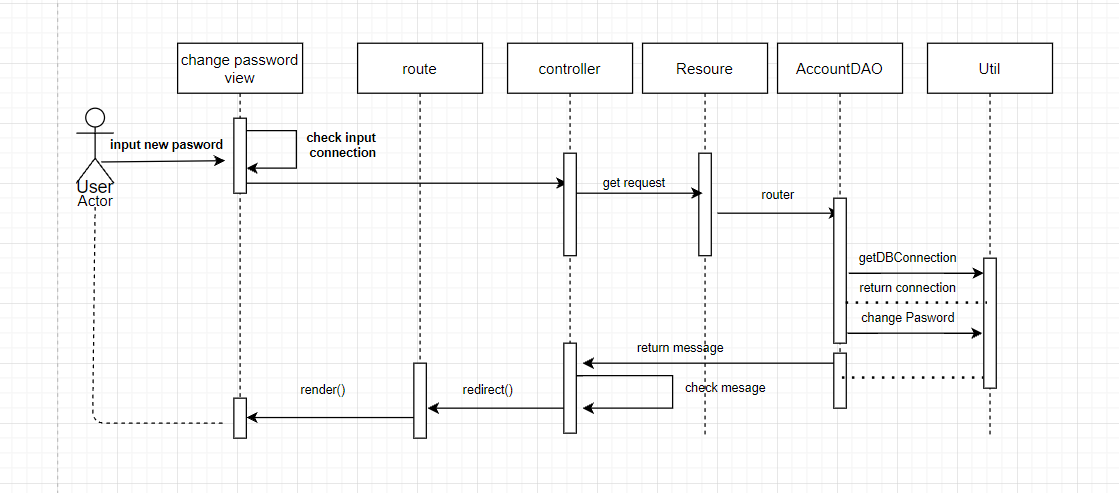
* ChangePasswordControl:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  **- Parameter: HttpServletRequest request, HttpServletResponse response** |

* UserDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getUsersByID() | - *Return: ‘A user’ or ‘Null’*  - *Input: UserID*  - *Find User by UserID* |
| 02 | updatePassword() | - *Return: void*  - Parameter: input string  - *Input: UserID, newPassword*  - *Update this new password into database* |

### c. Sequence Diagram(s)

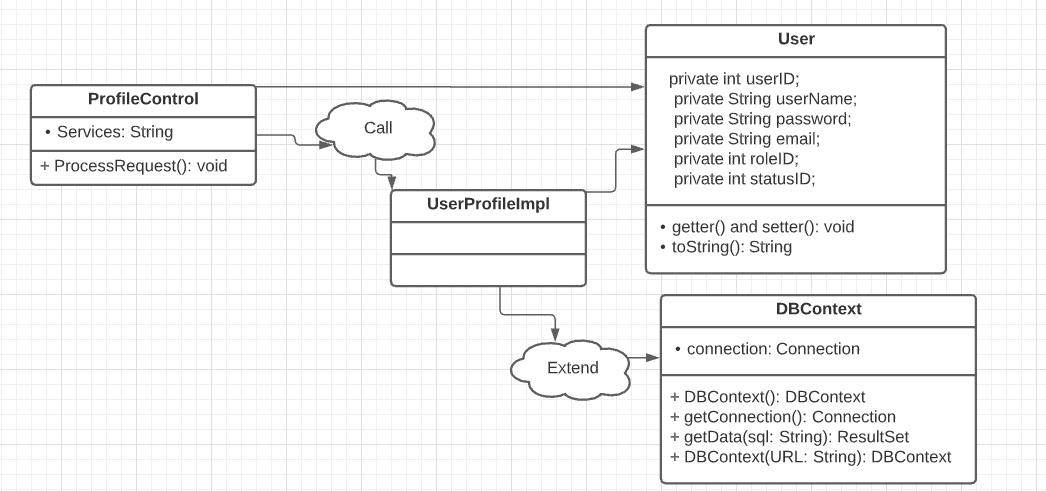


### d. Database queries

* select \* from Users where UserID = ?
* UPDATE Users SET Password = ? WHERE UserID = ?

## 17. User Profile

### a. Class Diagram

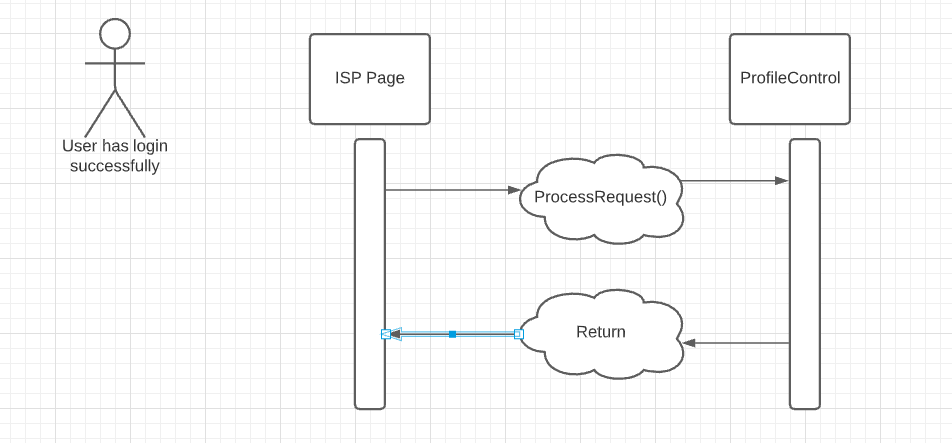


### b. Class Specifications

* ProfileControll:

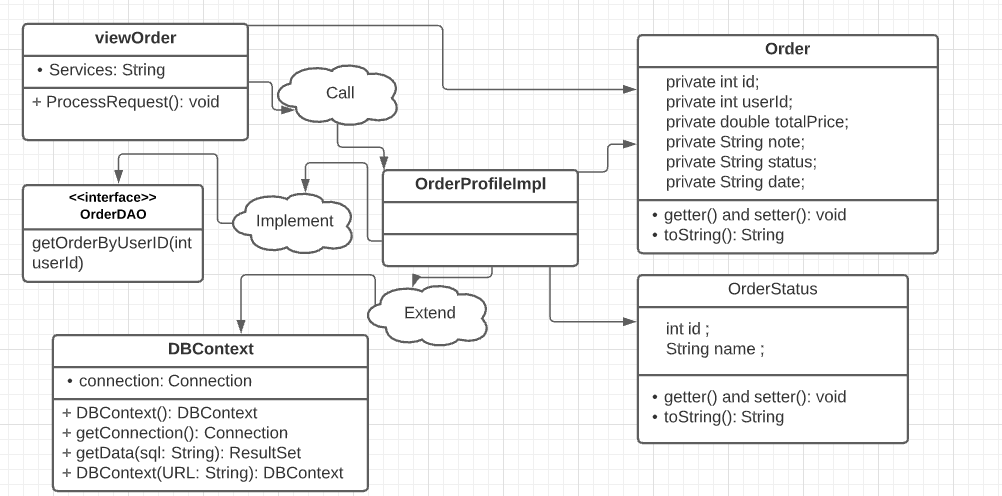
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | get all user information from the database |

### c. Sequence Diagram(s)



## 18. My Orders

### a. Class Diagram



### b. Class Specifications

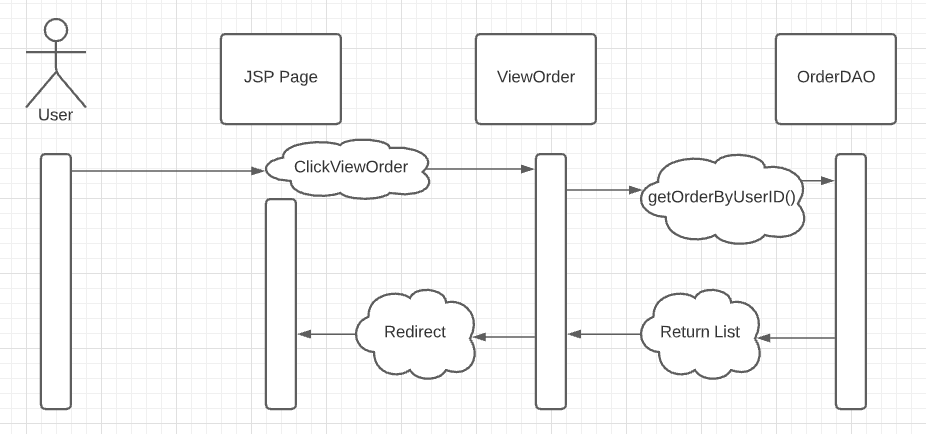
* viewOrderr:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing all orders and put that data to jsp page* |

* OrderDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getAllUsers() | - *Return: List<Order>*  - ***Parameter: UserID***  - *Get all the orders of a user in the database and return a list of orders* |

### c. Sequence Diagram(s)

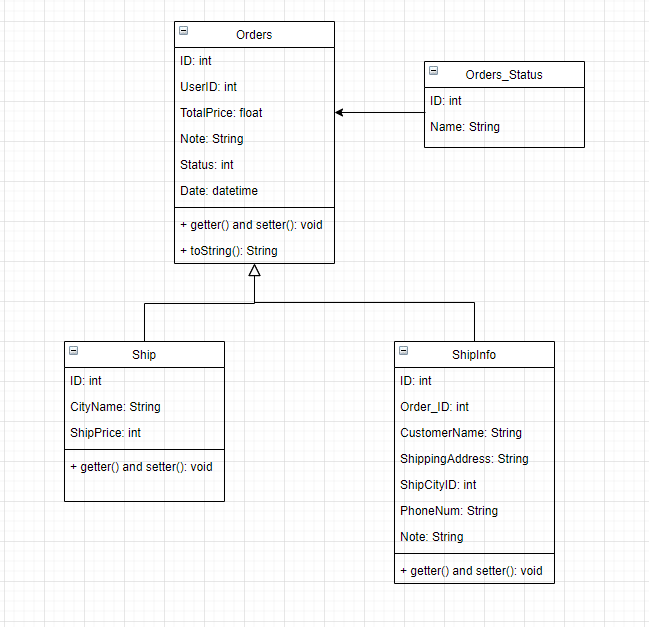


**d. Database queries**

* SELECT o.ID, o.UserID, o.TotalPrice, o.Note, os.Name, o.DayBuy FROM Orders o INNER JOIN Order\_Status os ON o.Status = os.ID WHERE o.UserId = ?

## 19. Order Information

### a. Class Diagram



**b. Class Specifications**

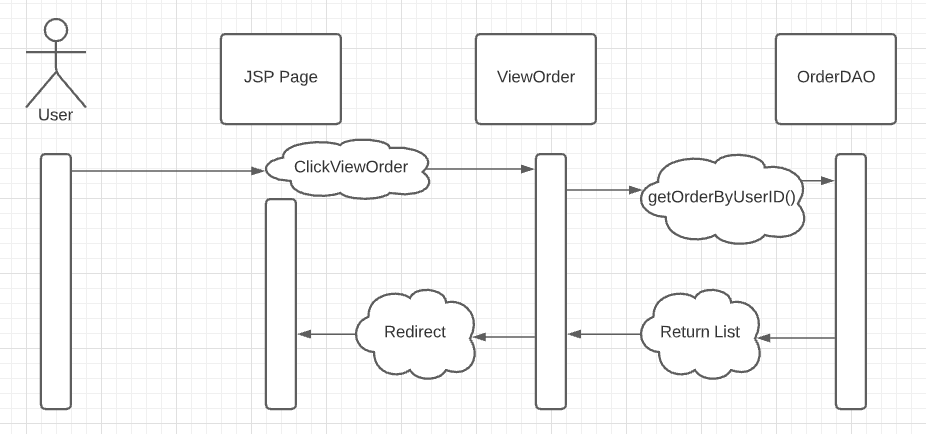
* viewOrderr:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing all orders and put that data to jsp page* |

* OrderDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getAllUsers() | - *Return: List<Order>*  - ***Parameter: UserID***  - *Get all the orders of a user in the database and return a list of orders* |

### c. Sequence Diagram(s)



### d. Database queries

SELECT o.ID, o.UserID, o.TotalPrice, o.Note, os.Name, o.DayBuy FROM Orders o INNER JOIN Order\_Status os ON o.Status = os.ID WHERE o.UserId = ?

**20. View Product List**

**a. Class Diagram**

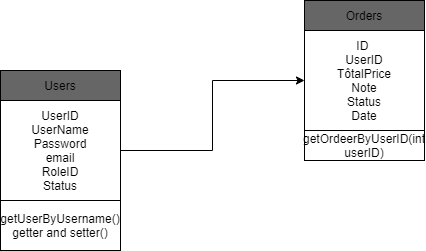
**b. Class Specifications**

**c. Sequence Diagram(s)**

**d. Database queries**

**21. Orders List**

**a. Class Diagram**

****

**b. Class Specifications**

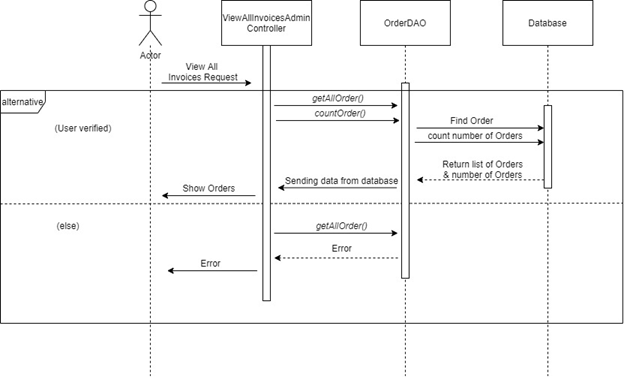
**Dashboard**

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | doget() | - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing orders by list Product and put that data to jsp page* |

**OrderDAO**

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getOrderByUserID(int userId) | - *Return: list Order*  - *Parameter:* userID  - Get list Order by userID |

**c. Sequence Diagram(s)**

****

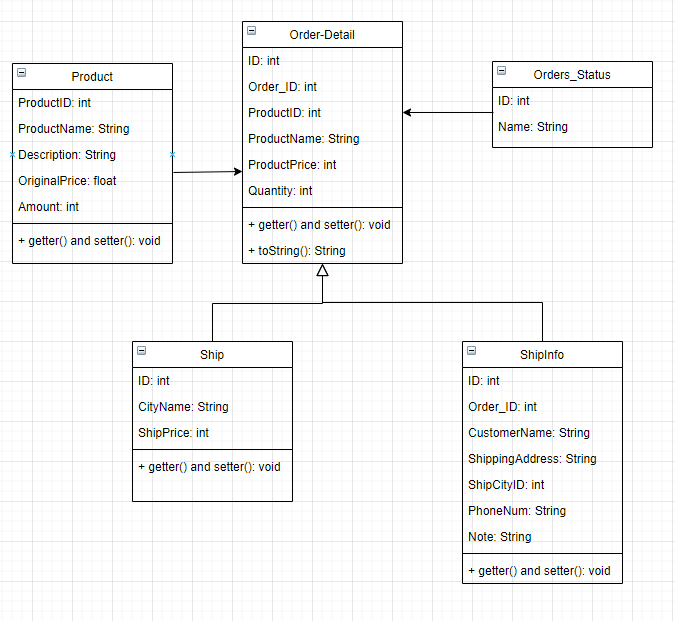
### d. Database queries

* SELECT o.ID, o.UserID, o.TotalPrice, o.Note, os.Name, o.DayBuy FROM Orders o INNER JOIN Order\_Status os

ON o.Status = os.ID WHERE o.UserId = ?

## 22. Order Details

### a. Class Diagram



### b. Class Specifications

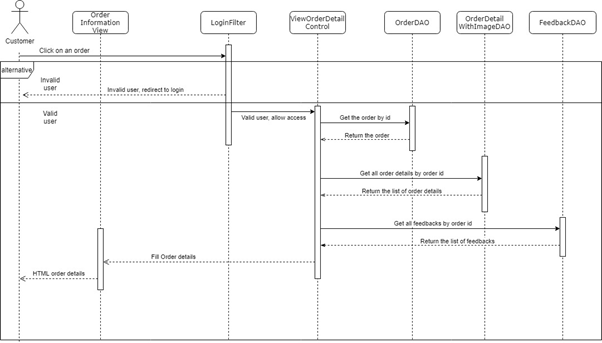
* viewOrder Details:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | - *Return: Null*  - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing all orders and put that data to jsp page* |

* OrderDAO:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getAllUsers() | - *Return: List<Order>*  - ***Parameter: UserID***  - *Get all the orders of a user in the database and return a list of orders* |

### c. Sequence Diagram(s)

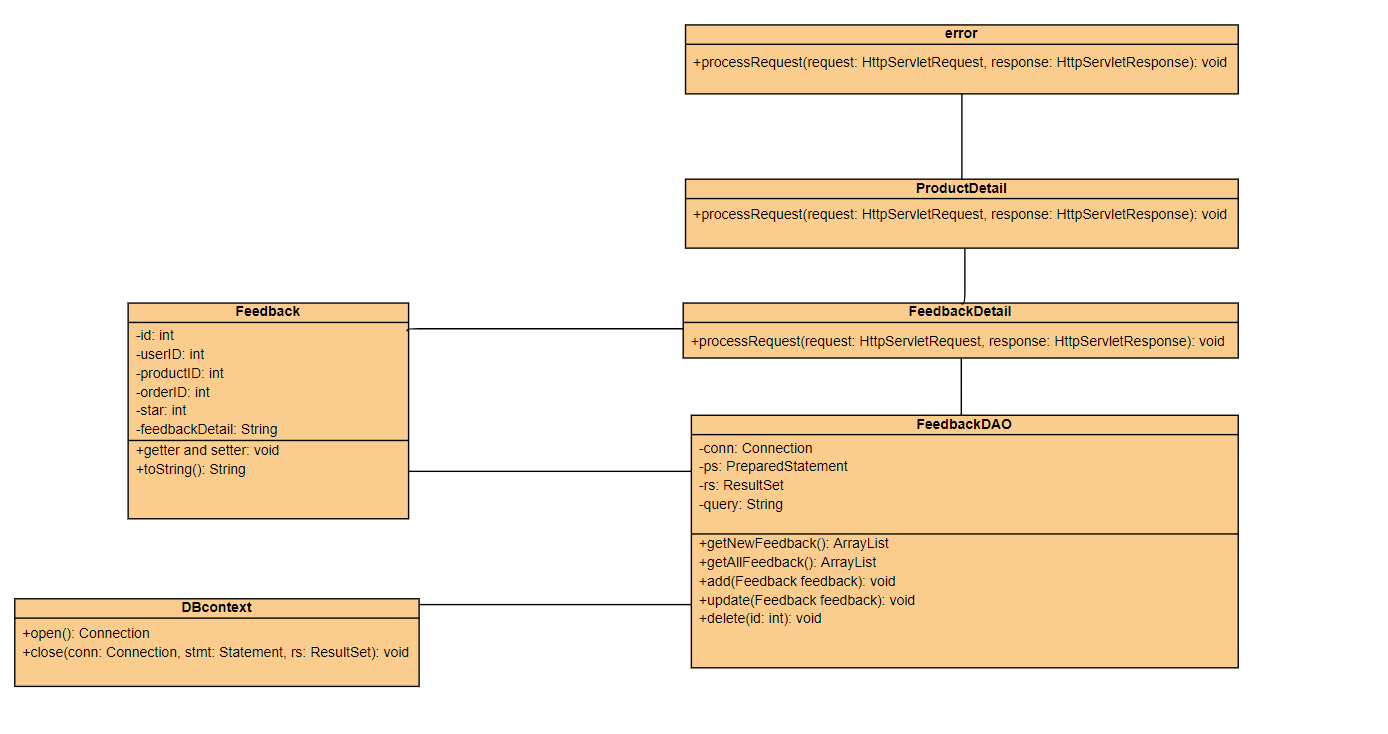
****

### d. Database queries

SELECT o.ID, o.Order\_ID, o.ProductID, o.ProductName, o.ProductPrice os.Name, o.DayBuy FROM Orders o INNER JOIN Order\_Status os ON o.Status = os.ID WHERE o.UserId = ?

## 23. View Feedback

### a. Class Diagram



### b. Class Specifications

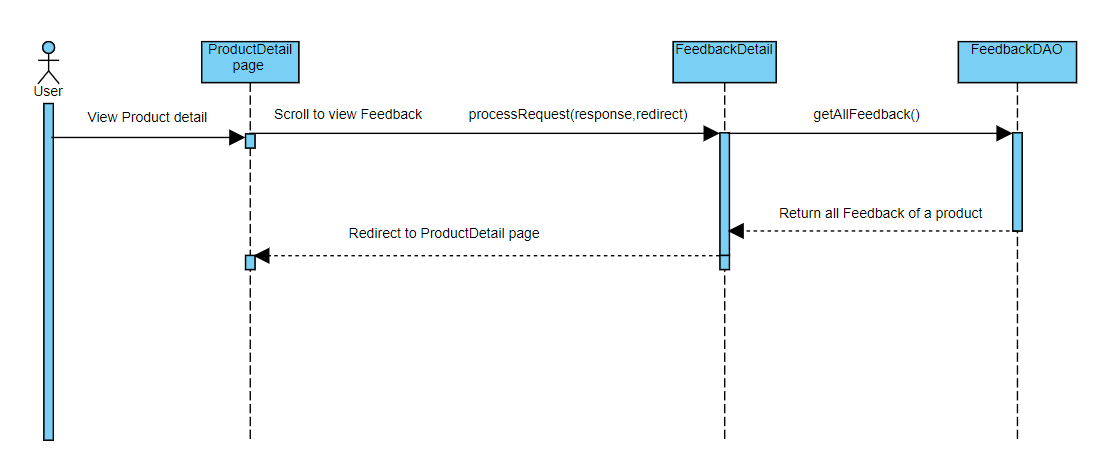
* FeedbackDetail class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get blog detail from the database to display on ProductDetail.jsp* |

* FeedbackDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getNewFeedback()* | *Return: top 5 newest Feedback from database* |
| *03* | *getAllFeedback (int id)* | *Return: All Feedback of a product in database*  *input: id* |
| *04* | *add(Feedback feedback)* | *Add feedback with detail entered in addFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *05* | *update(Feedback feedback)* | *Update feedback with detail entered in updateFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *06* | *delete(int id)* | *Update feedback with id entered in updateFeedback.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)

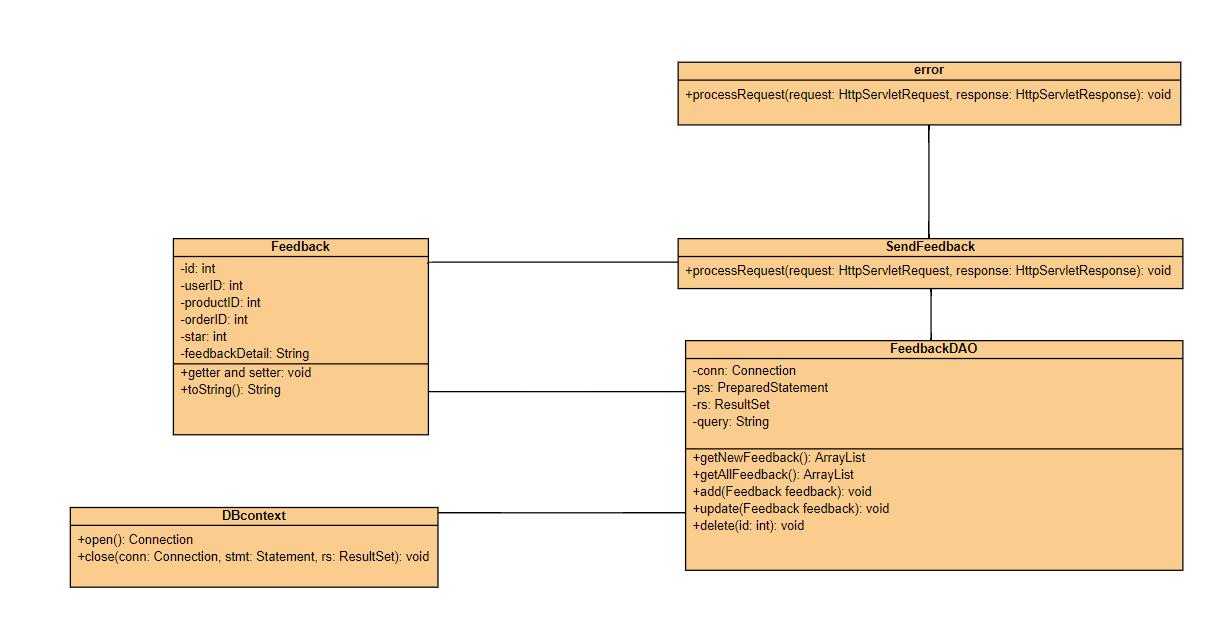


### d. Database queries

Select \* from Feedback where productID=?

## 24. Send Feedback

### a. Class Diagram



### b. Class Specifications

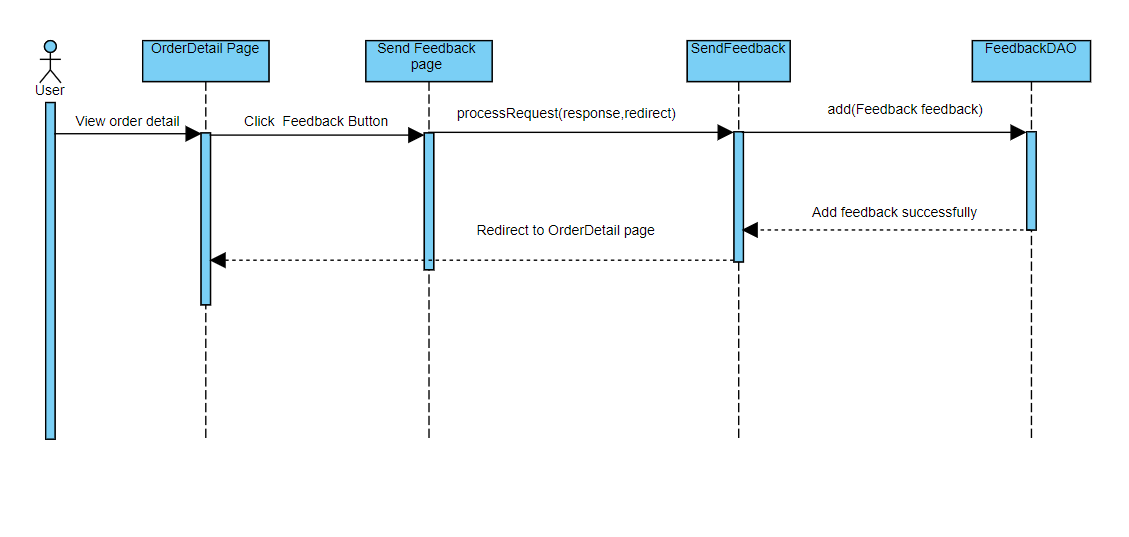
* SendFeedback class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get Form Feedback to display on SendFeedback.jsp* |

* FeedbackDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getNewFeedback()* | *Return: top 5 newest Feedback from database* |
| *03* | *getAllFeedback (int id)* | *Return: All Feedback of a product in database*  *input: id* |
| *04* | *add(Feedback feedback)* | *Add feedback with detail entered in addFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *05* | *update(Feedback feedback)* | *Update feedback with detail entered in updateFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *06* | *delete(int id)* | *Update feedback with id entered in updateFeedback.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)



### d. Database queries

Insert into Feedback value (?,?,?,?,?)

## 25. Blogs List

### a. Class Diagram

### b. Class Specifications

* BlogList class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get blog list from the database to display on BlogList.jsp* |

* BlogDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getBlogByID(int id)* | *Return: Blog Detail where blog id contain parameter*  *Input: ID of blog* |
| *02* | *getHotBlogs()* | *Return: top 3 newest blog from database* |
| *03* | *getAllBlog()* | *Return: All blog in database* |
| *04* | *add(String author, String title, String content, String imageLink)* | *Add blog with detail entered in addBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink* |
| *05* | *update(String author, String title, String content, String imageLink, String id)* | *Update blog with detail entered in updateBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink, id* |
| *06* | *delete(int id)* | *Update blog with id entered in blogManage.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)

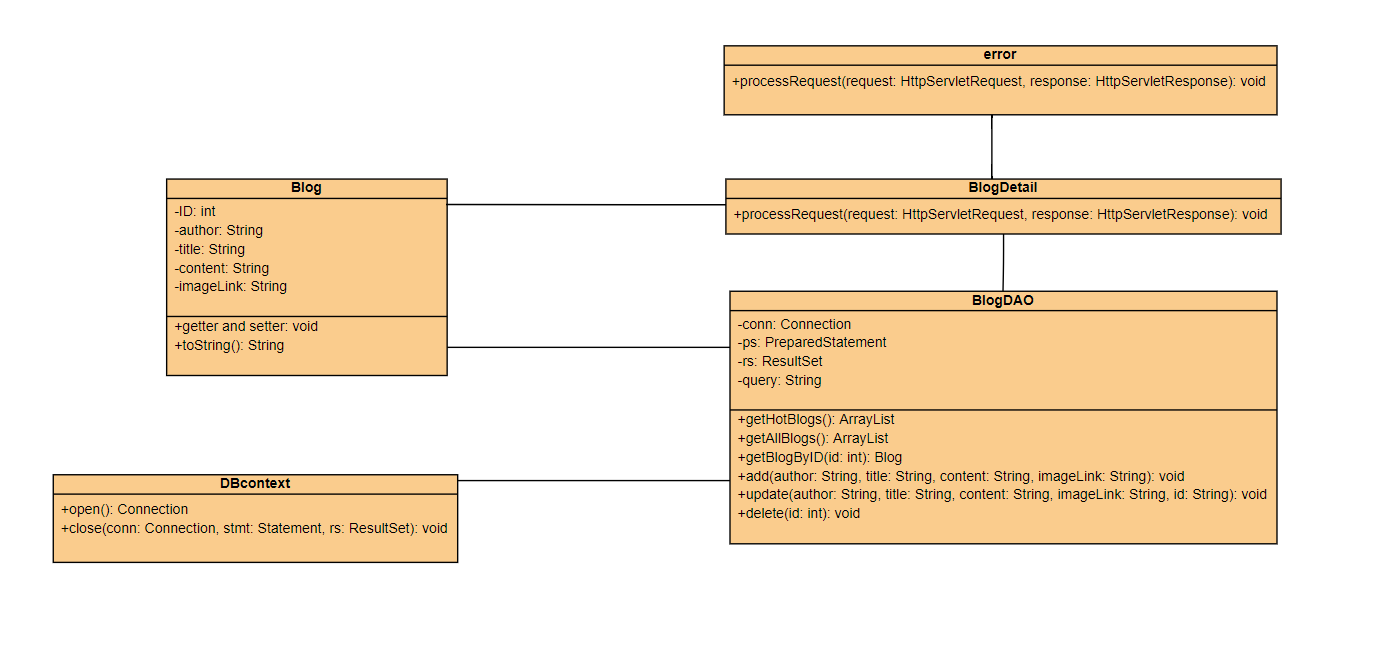


### d. Database queries

Select \* from Blog desc

## 26. Blog Details

### a. Class Diagram



### b. Class Specifications

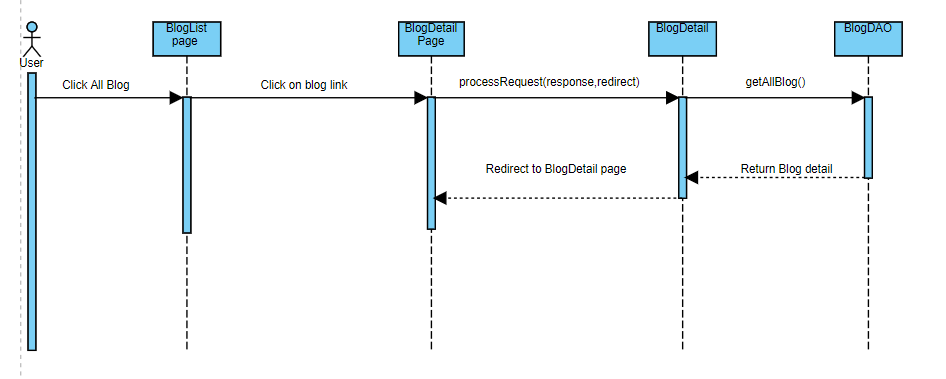
* BlogDetail class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get blog detail from the database to display on BlogDetail.jsp* |

* BlogDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getBlogByID(int id)* | *Return: Blog Detail where blog id contain parameter*  *Input: ID of blog* |
| *02* | *getHotBlogs()* | *Return: top 3 newest blog from database* |
| *03* | *getAllBlog()* | *Return: All blog in database* |
| *04* | *add(String author, String title, String content, String imageLink)* | *Add blog with detail entered in addBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink* |
| *05* | *update(String author, String title, String content, String imageLink, String id)* | *Update blog with detail entered in updateBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink, id* |
| *06* | *delete(int id)* | *Update blog with id entered in blogManage.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)

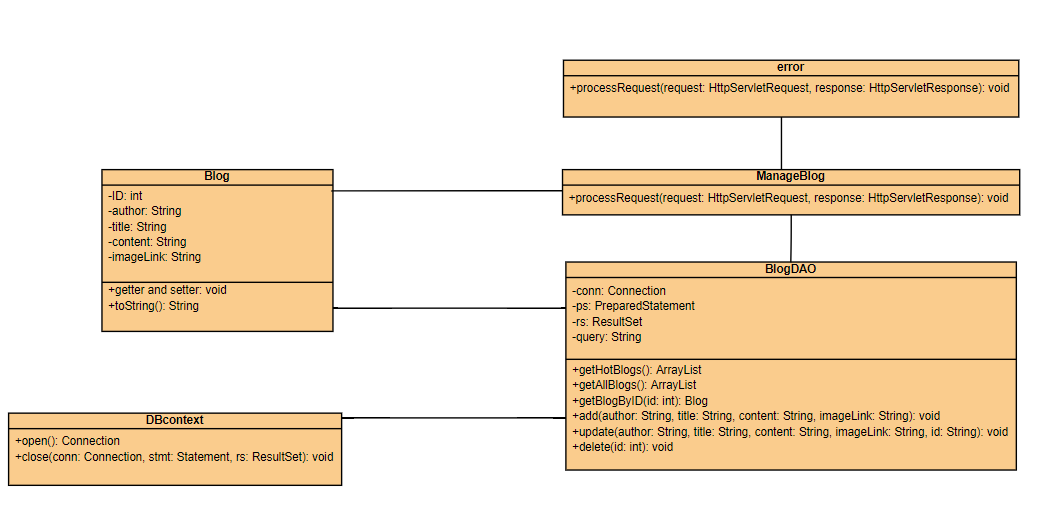


### d. Database queries

SELECT \* FROM Blog WHERE ID = ?

## 27. Blog Management

### a. Class Diagram



### b. Class Specifications

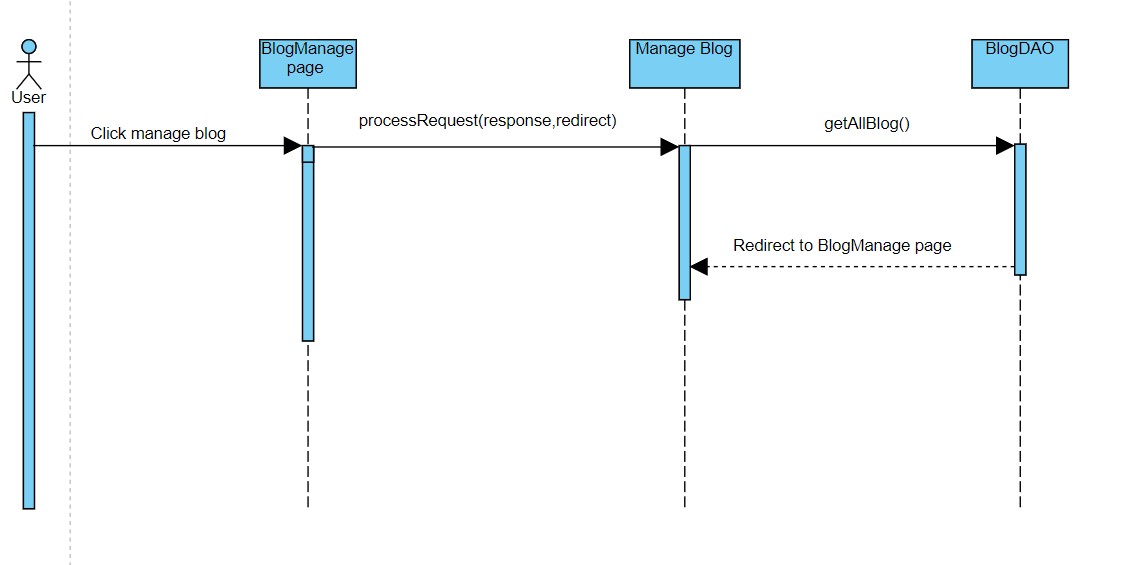
* ManageBlog class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get blog detail from the database to display on ManageBlog.jsp* |

* BlogDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getBlogByID(int id)* | *Return: Blog Detail where blog id contain parameter*  *Input: ID of blog* |
| *02* | *getHotBlogs()* | *Return: top 3 newest blog from database* |
| *03* | *getAllBlog()* | *Return: All blog in database* |
| *04* | *add(String author, String title, String content, String imageLink)* | *Add blog with detail entered in addBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink* |
| *05* | *update(String author, String title, String content, String imageLink, String id)* | *Update blog with detail entered in updateBlog.jsp page*  *Return: null*  *Input: author, title, content, imageLink, id* |
| *06* | *delete(int id)* | *Update blog with id entered in blogManage.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)

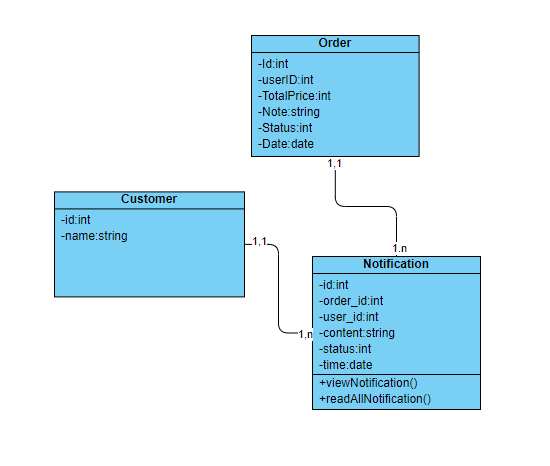


### d. Database queries

Select \* from blog

## 28. View Notification

### a. Class Diagram

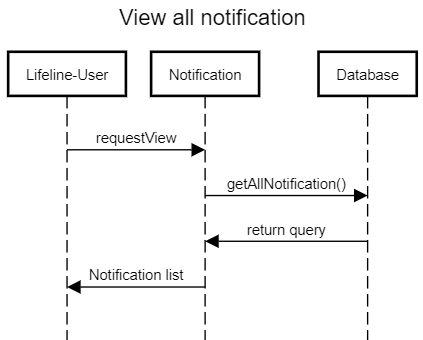


### b. Class Specifications

Notification class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *viewNotification()* | *get all notification from database for a user* |

### c. Sequence Diagram(s)

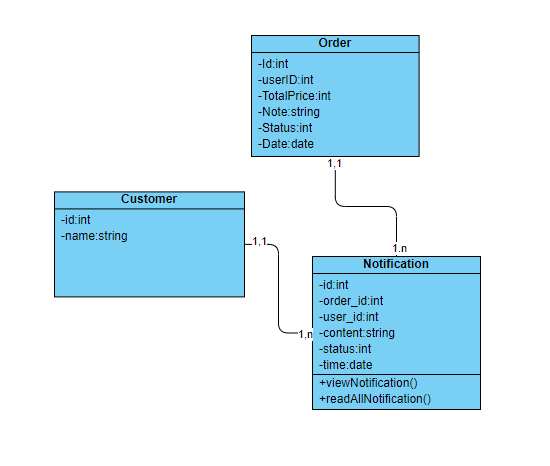


### d. Database queries

SELECT \* from dbo.Notification where user\_id = ?

## 29. Mark as Read

### a. Class Diagram

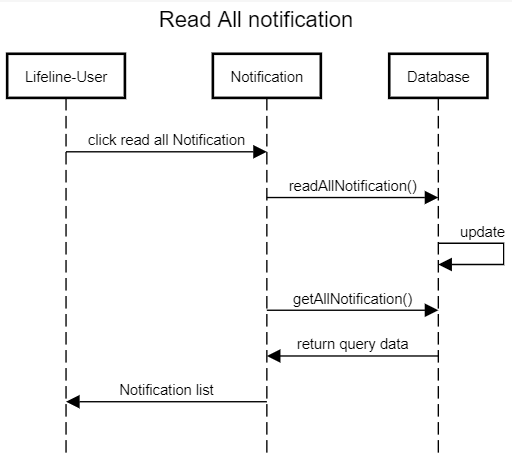


### b. Class Specifications

Notification class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *readAllNotification()* | *mark all notification as read* |

### c. Sequence Diagram(s)



### d. Database queries

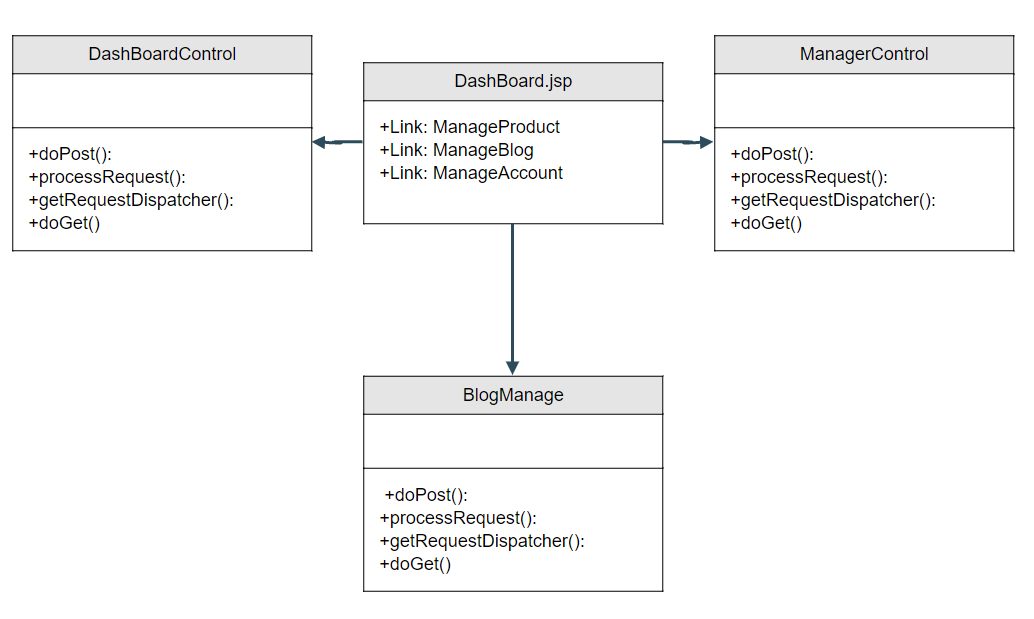
Update dbo.Notification

set status=1

where user\_id=?

## 30. Admin Feature

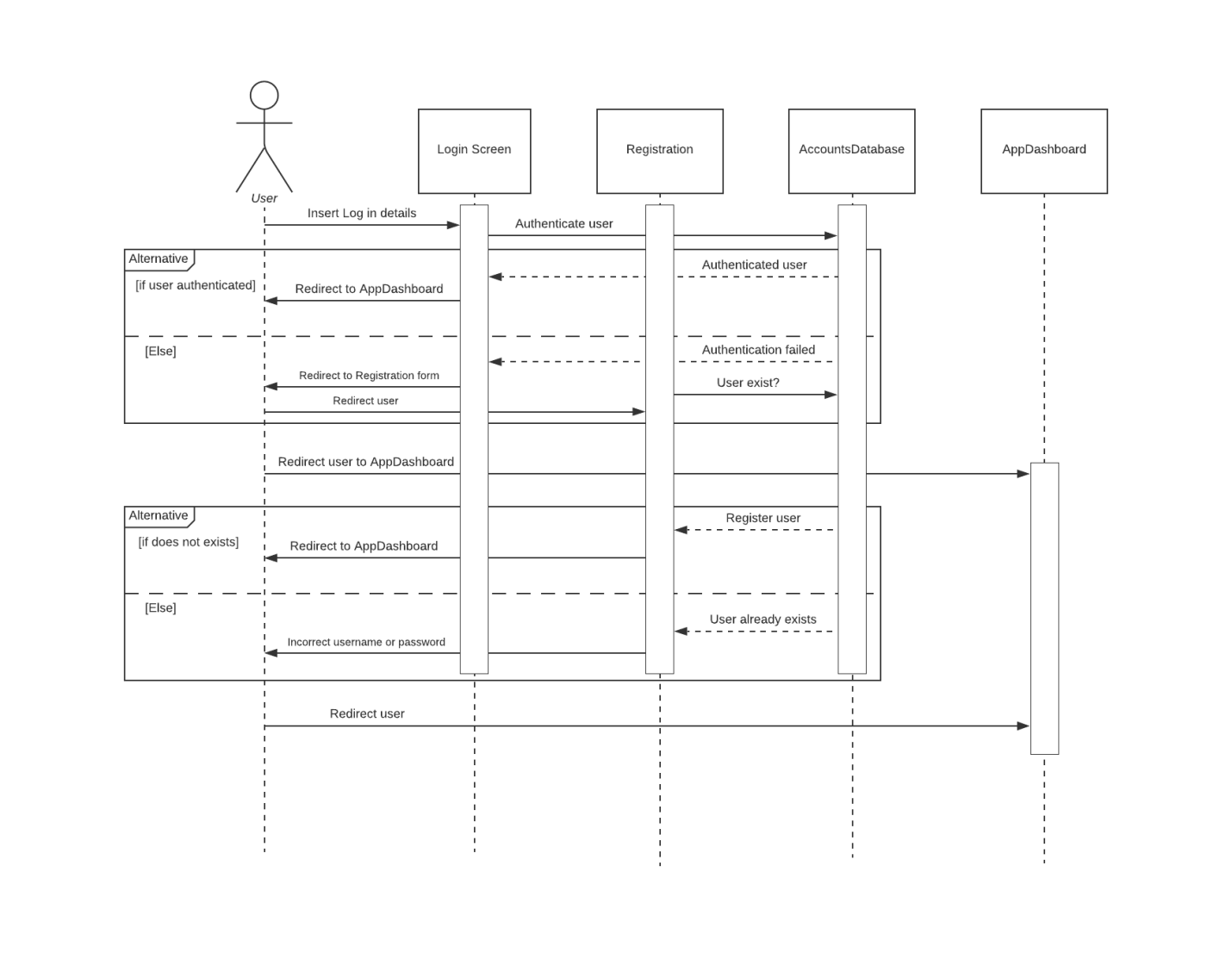
### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get blog detail from the database to display on ManageProduct.jsp* |
| *02* | *processRequest()* | *get blog detail from the database to display on ManageAccount.jsp* |
| *03* | *processRequest()* | *get blog detail from the database to display on ManageBlog.jsp* |

### c. Sequence Diagram(s)

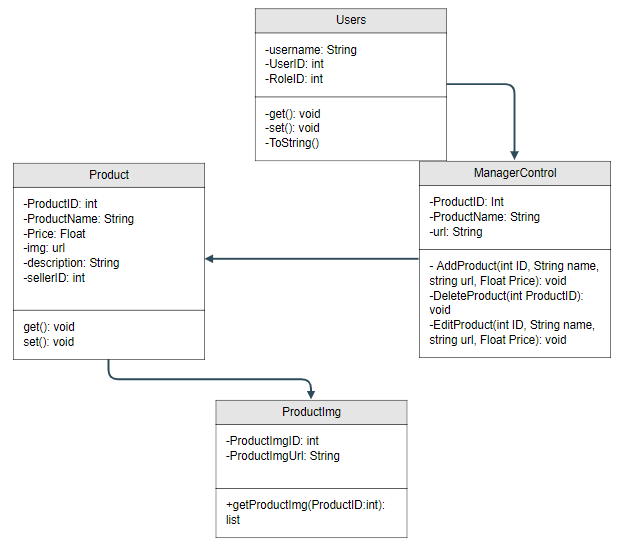


### d. Database queries

+select \* from …

## 31. Add Product

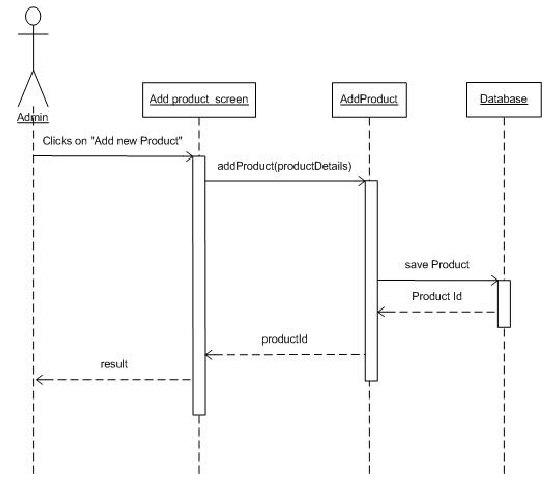
### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *AddProduct from the database to display on ManageProduct.jsp* |

### c. Sequence Diagram(s)

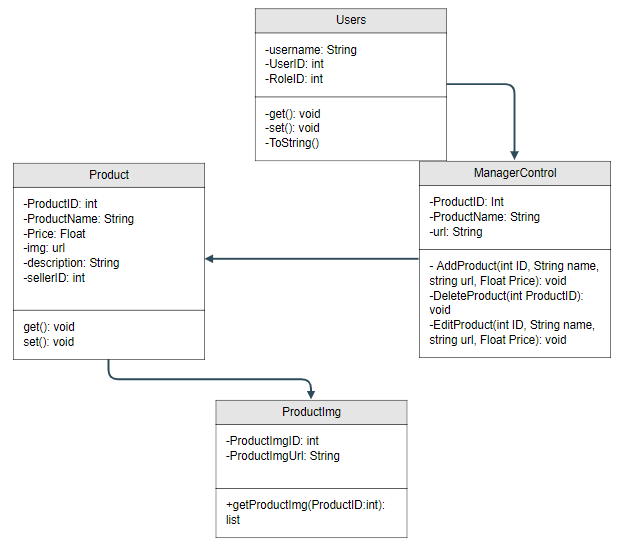


### d. Database queries

+insert into datatable (....)values (....)

## 32. Delete Product

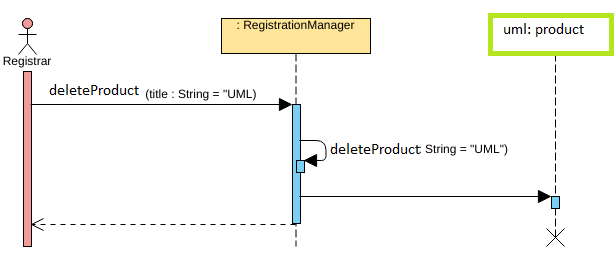
### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *DeleteProduct from the database to display on ManageProduct.jsp* |

### c. Sequence Diagram(s)

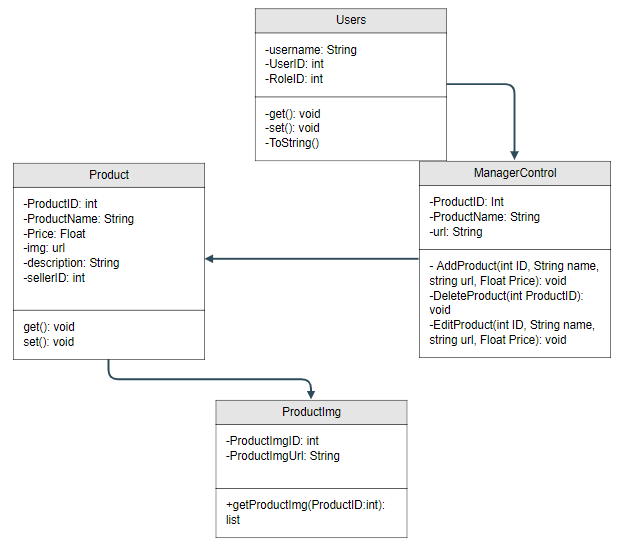


### d. Database queries

+delete from Product where ProductID = …

## 33. Edit Product

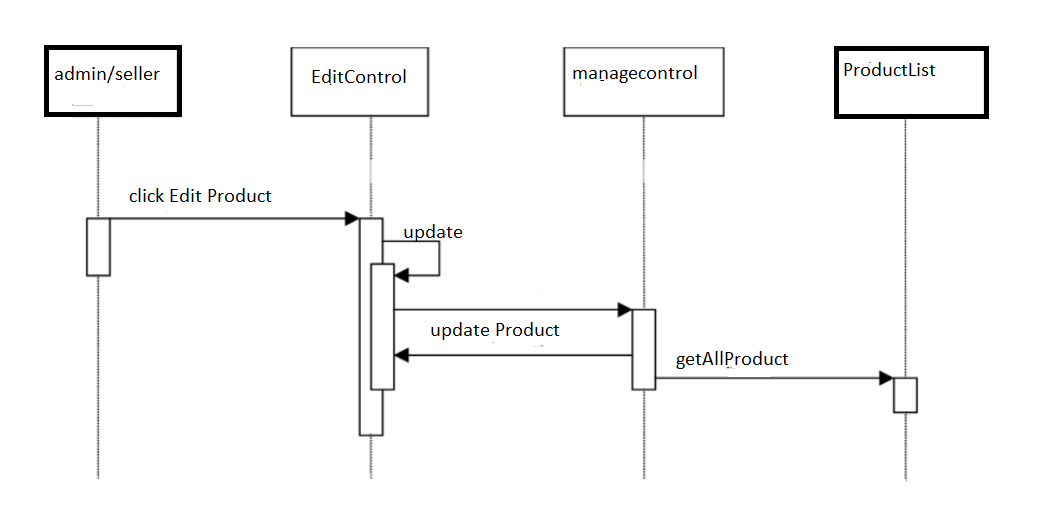
### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *Update from the database to display on ManageProduct.jsp* |

### c. Sequence Diagram(s)



### d. Database queries

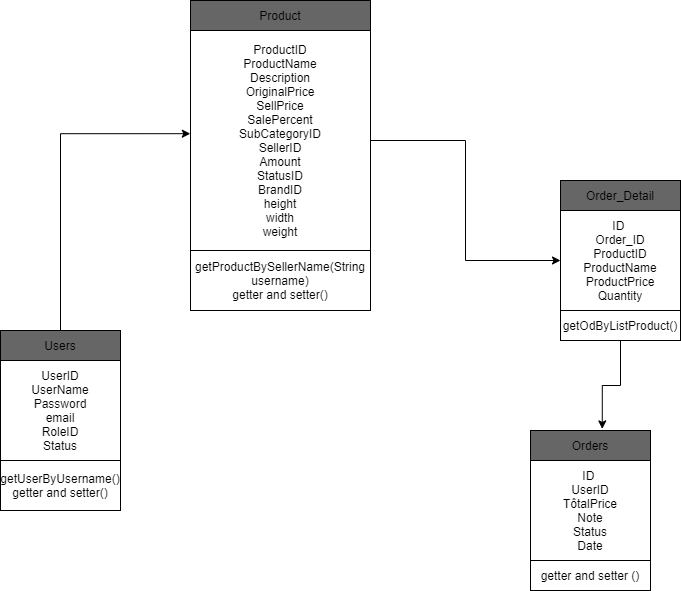
+UPDATE datatable

SET ... = '......'

WHERE …... = '.......'

## 34. Orders Dashboard

**a. Class Diagram**

****

**b. Class Specifications**

**Dashboard**

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | doget() | -  - *Parameter: HttpServletRequest request, HttpServletResponse response*  - *Receive request from client and process data with Database, then return a list containing orders by list Product and put that data to jsp page* |

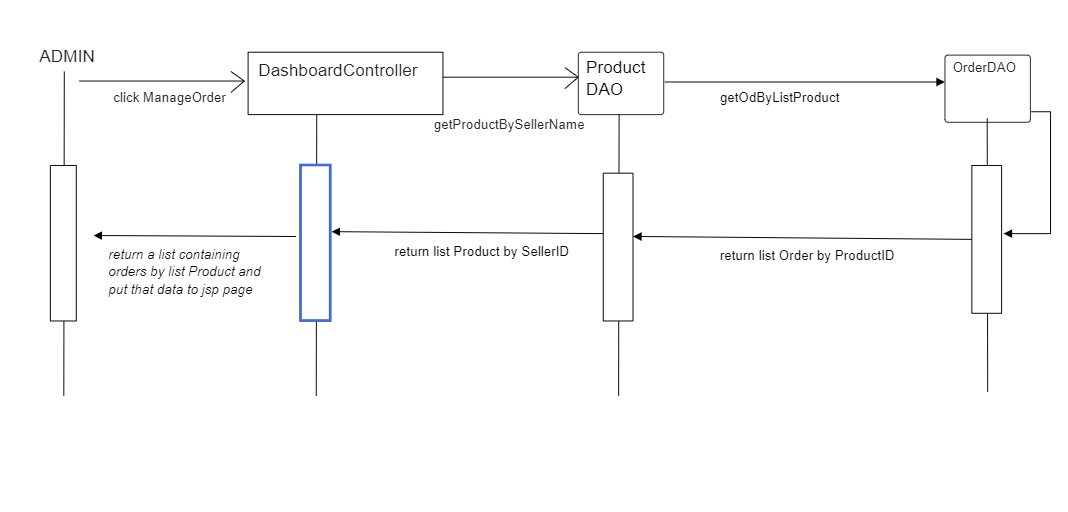
**Product DAO**

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getProductBySellerName(String username) | - *Return: listProduct*  - *Parameter: String username*  - Get list product by sellerID |

**OrderDAO**

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getOdByListProduct(ArrayList<Product> listP) | - *Return: list Order*  - *Parameter:* ArrayList<Product> listP  - Get list Order by ProductID |

**c. Sequence Diagram(s)**

****

### d. Database queries

* **select \* from Users where Username = ?**
* **select p.ProductID , ProductName , Description , OriginalPrice , SellPrice , SalePercent , SubCategoryID , SellerID , Amount , StatusID , StatusID ,BrandID , height , width ,weight , s.ProductImgURL from Product p join ProductImg s**

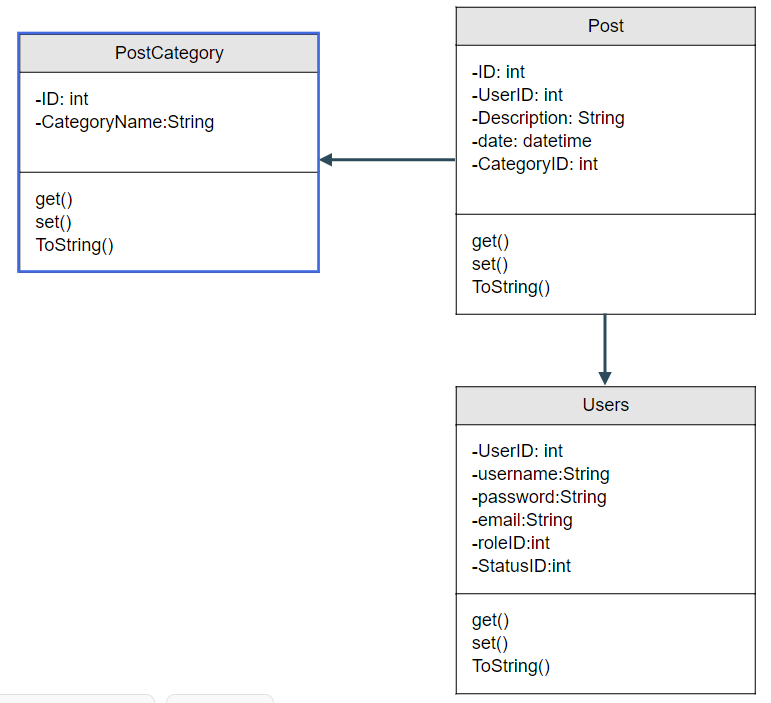
**on p.ProductID = s.ProductID where p.SellerID = (select UserID from Users where Username = ? )**

* **select o.ID , o.UserID , o.TotalPrice , o.Note , o.Status , o.Date , d.Order\_ID , d.ProductID , d.ProductName , d.ProductPrice ,d.ProductPrice , d.Quantity from Orders o**

**join Order\_Detail d on d.Order\_ID = o.ID where ProductID = ?**

## 35. PostDetail

**a.class diagram**



### b. Class Specifications

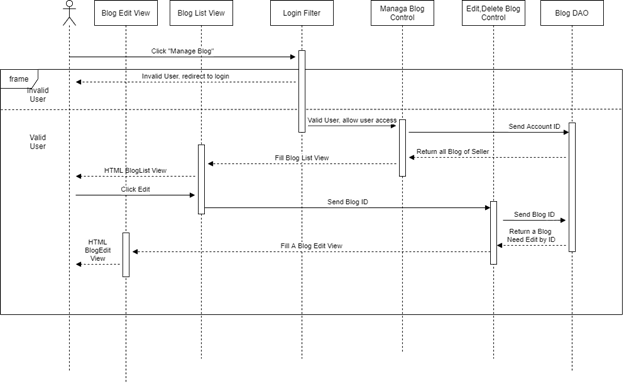
#### PostControl Class: The doGet function is used to get the ID of the post and then use the doPost function to edit the product

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **processRequest()** | **Nothing** |
| **2** | **doGet()** | **Get ID of post and push ID, title, description, imageLink, Seller ID to EditBlog.jsp to edit** |
| **3** | **doPost()** | **Get ID, title, description, imageLink, userID in PostDetailjsp when changed, use edit function in postDAO to edit post and combackManageControl Class** |
| **4** | **getServletInfo()** | **Return a String containing servlet description** |

#### DeletepostControll Class: use process request function to get and delete post by ID

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **processRequest()** | **Get ID of post need to Delete and use function delete in post DAO to delete post by ID and comeback Error.jsp** |
| **2** | **doGet()** | **Nothing** |
| **3** | **doPost()** | **Nothing** |
| **4** | **getServletInfo()** | **Return a String containing servlet description** |

### c. Sequence Diagram(s)



### d. Database queries

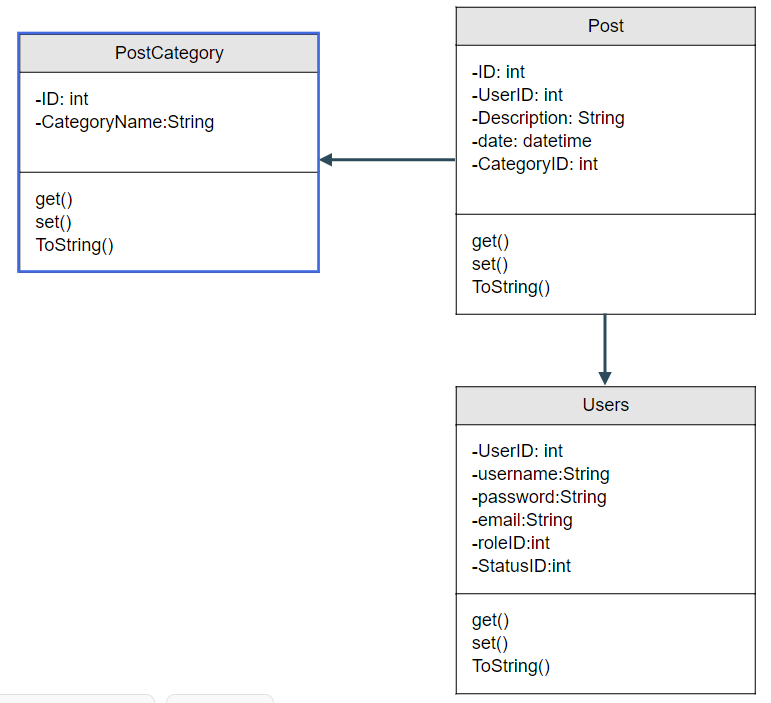
- "Delete FROM Post WHERE ID = ?"

- "Update …= ? “

- "INSERT INTO Post VALUES (?,?,?,?)"

**36. PostList**

**a.class diagram**

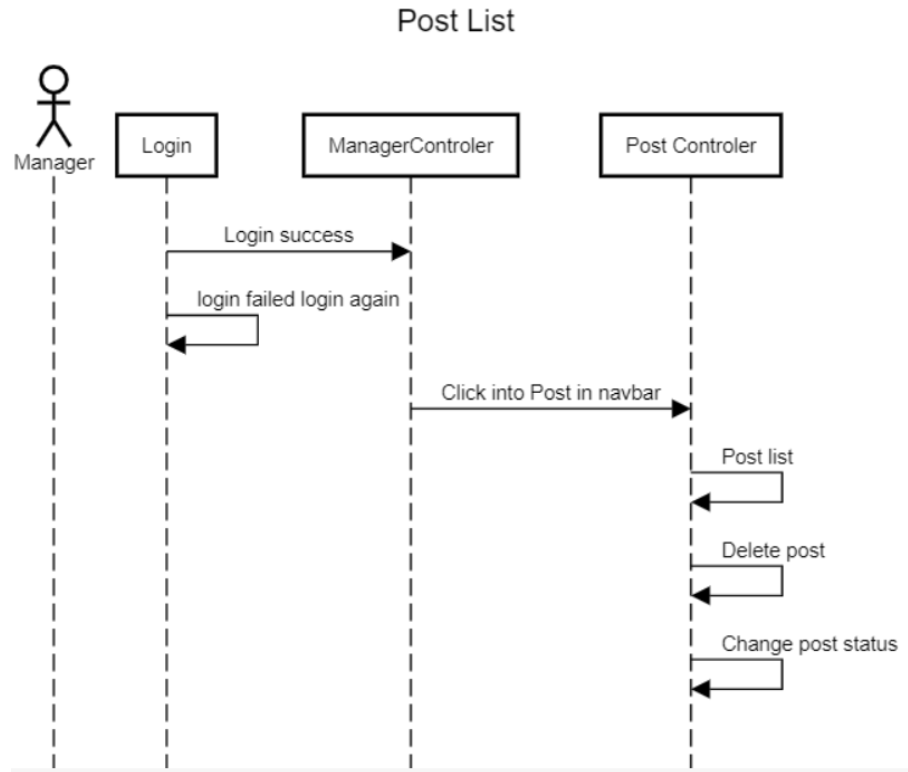
****

### b. Class Specifications

#### PostManageControl Class: This class process request for list of post create by user ID

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | **processRequest()** | **Select list Post of user in SQL server and return Postlist.jsp** |
| **2** | **doGet()** | **Nothing** |
| **3** | **doPost()** | **Nothing** |

**c.sequences diagram**

****

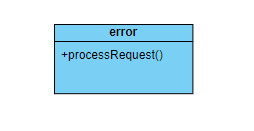
### d. Database queries

**- "SELECT \* FROM Post WHERE ID = ?"**

## 

## 37. Error

### a. Class Diagram

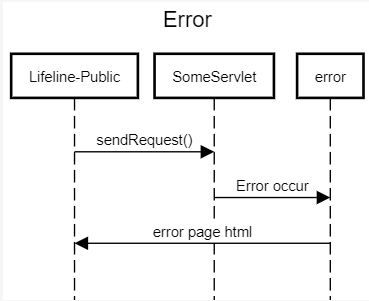
**

### b. Class Specifications

error Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *Process and return error page html back to user* |

### c. Sequence Diagram(s)

**

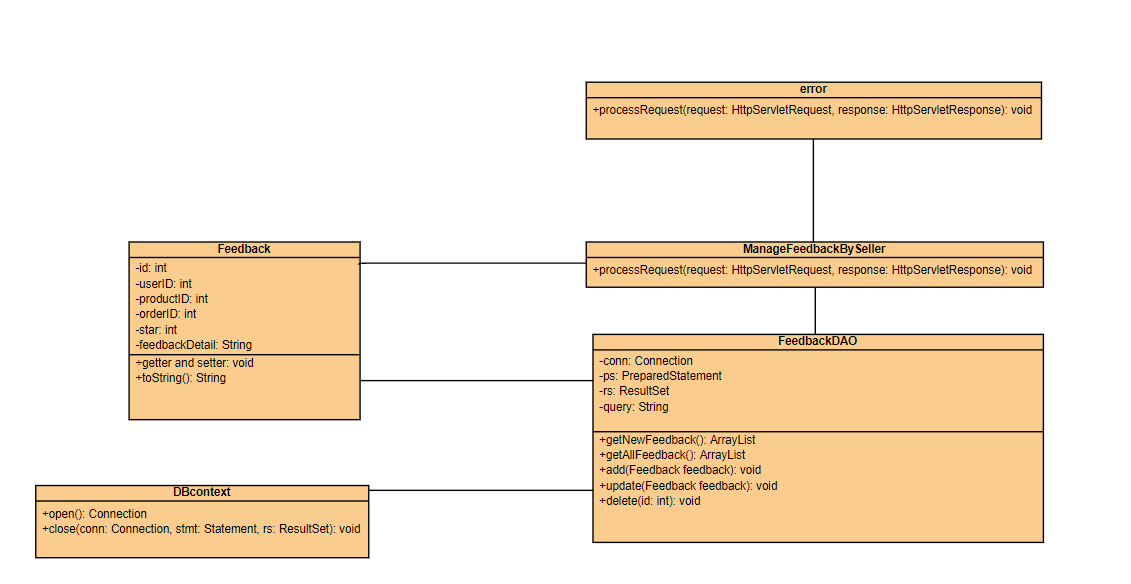
### d. Database queries

None

## 

## 38. Manage Feedback [Seller]

### a. Class Diagram



### b. Class Specifications

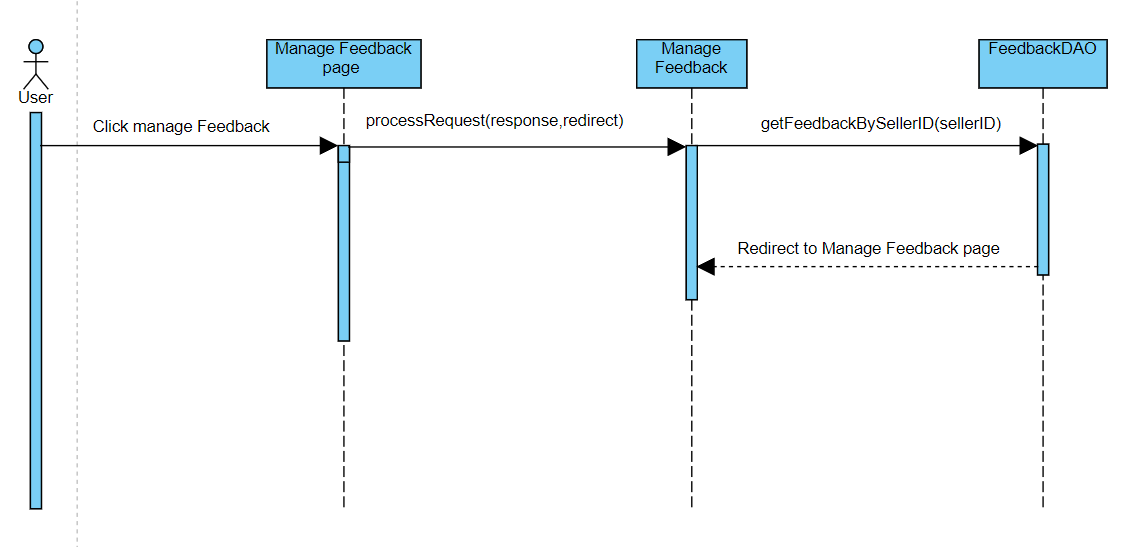
* ManageFeedbackBySeller class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get Feedback by Seller to display on ManageFeedbackBySeller.jsp* |

* FeedbackDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getNewFeedback()* | *Return: top 5 newest Feedback from database* |
| *03* | *getAllFeedback (int id)* | *Return: All Feedback of a product in database*  *input: id* |
| *04* | *add(Feedback feedback)* | *Add feedback with detail entered in addFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *05* | *update(Feedback feedback)* | *Update feedback with detail entered in updateFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *06* | *delete(int id)* | *Update feedback with id entered in updateFeedback.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)

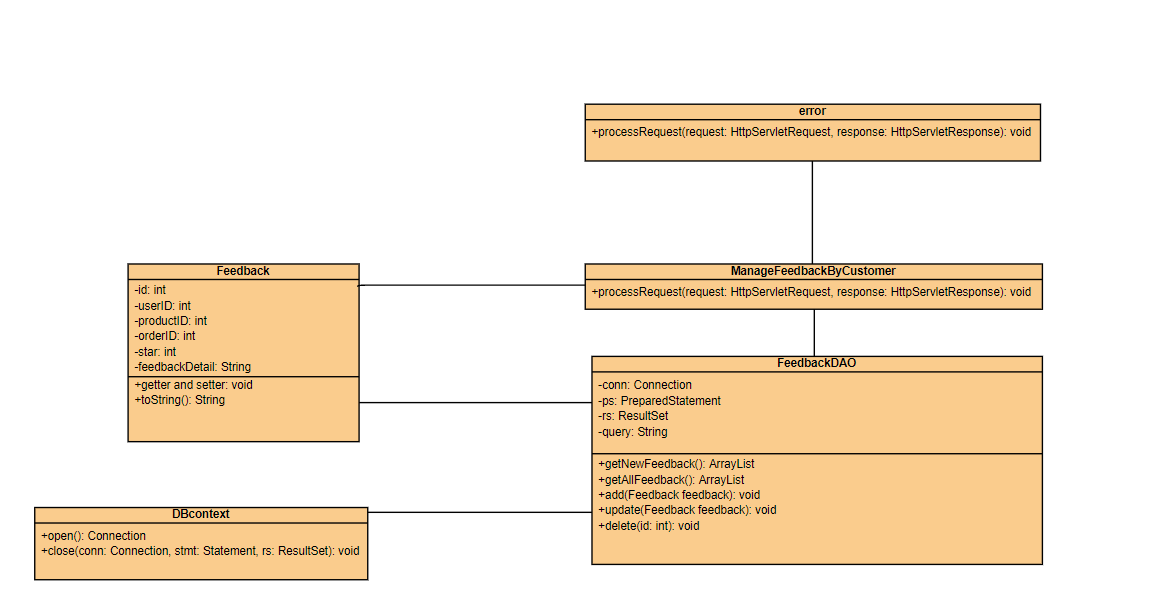


### d. Database queries

SELECT \* FROM (dbo.Feedback f JOIN dbo.Product p ON p.ProductID=f.ProductID) JOIN dbo.Users u ON u.UserID=p.SellerID WHERE f.UserID=?

## 39. Manage Feedback [Customer]

### a. Class Diagram



### b. Class Specifications

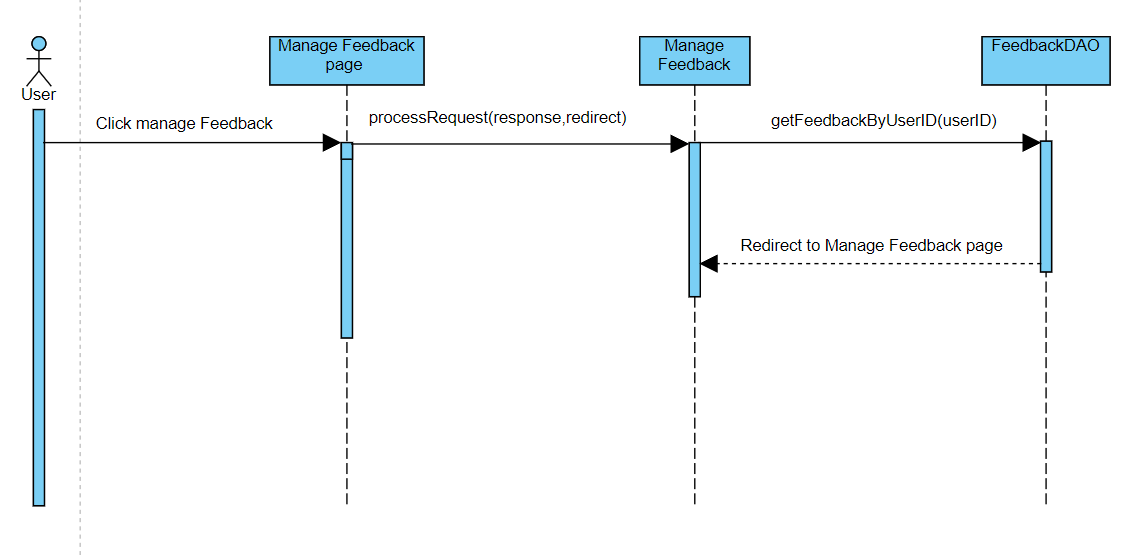
* ManageFeedbackBySeller class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest()* | *get Feedback by Customer to display on ManageFeedbackByCustomer.jsp* |

* FeedbackDAO class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getNewFeedback()* | *Return: top 5 newest Feedback from database* |
| *03* | *getAllFeedback (int id)* | *Return: All Feedback of a product in database*  *input: id* |
| *04* | *add(Feedback feedback)* | *Add feedback with detail entered in addFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *05* | *update(Feedback feedback)* | *Update feedback with detail entered in updateFeedback.jsp page*  *Return: null*  *Input: feedback* |
| *06* | *delete(int id)* | *Update feedback with id entered in updateFeedback.jsp*  *Return: null*  *Input: id* |

### c. Sequence Diagram(s)



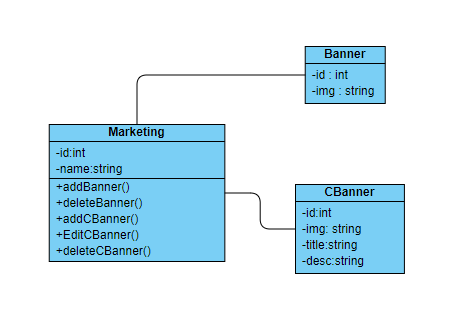
### d. Database queries

SELECT \* FROM (dbo.Feedback f JOIN dbo.Product p ON p.ProductID=f.ProductID) JOIN dbo.Users u ON u.UserID=p.SellerID WHERE f.UserID=?

### 

## 40. Manage Banner

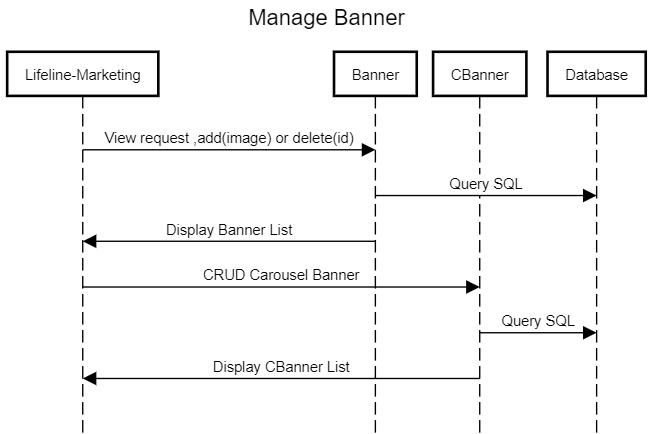
### a. Class Diagram

**

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *addBanner()* | *add new Banner to database* |
| *02* | *deleteBanner()* | *delete Banner from database* |
| *03* | *addCBanner()* | *add new Homepage Carousel Banner* |
| *04* | *editCBanner()* | *edit existing Carousel Banner* |
| *05* | *deleteCBanner()* | *delete old Carousel Banner* |

### c. Sequence Diagram(s)

**

### d. Database queries

SELECT \* FROM dbo.Banner WHERE ID = ?

SELECT \* FROM dbo.Banner

INSERT INTO dbo.Banner VALUES ( ? )

DELETE FROM dbo.Banner WHERE ID = ?

SELECT \* FROM dbo.CBanner WHERE ID = ?

SELECT \* FROM dbo.CBanner

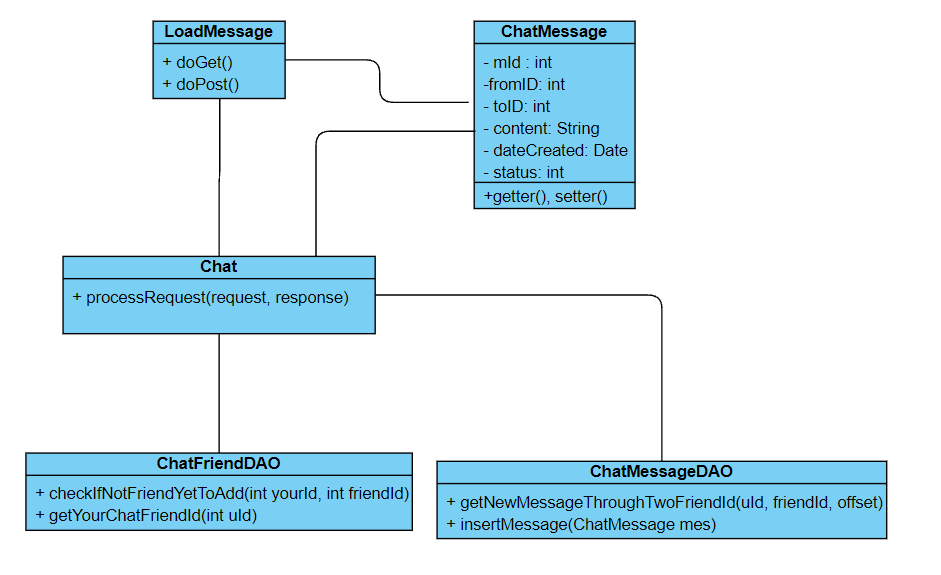
INSERT INTO dbo.CBanner VALUES ( ? , ? , ? )

UPDATE dbo.CBanner SET Img = ?,Title = ? , [desc] = ? WHERE ID = ?

DELETE FROM dbo.CBanner WHERE ID = ?

## 41. Chat

### a. Class Diagram



### b. Class Specifications

Chat

### 

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest(request, response) | * Press Request Statistic on header will call this controller * This class will get list of friend from data for user and send it to chatbox jsp * Will run to mentorRequestStatistic jsp with attribute data * Will throw Exception and run to error jsp if there any error |

### 

LoadMessage

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | doGet(HttpServletRequest request, HttpServletResponse response) | This get method will load new message for the user and send to ajax and load newly inserted messages if they have not been shown |
| 02 | doPost(HttpServletRequest request, HttpServletResponse response) | This post method will get message which user post and insert it into database |

### 

ChatFriendDAO

### 

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | checkIfNotFriendYetToAdd(int yourId, int friendId) | * This method will check two argument id if they are not friend yet, if not it will add into ChatFriend table) * param: yourId (your id) * friendId (friend id) * return: int (1 if add chat friend success) * throws Exception |
| 02 | getYourChatFriendId(int uId) | * This method will get all your chat friend id from database * param: uId (your id) * return: ArrayList<Integer> (list of friend id) * throws Exception |

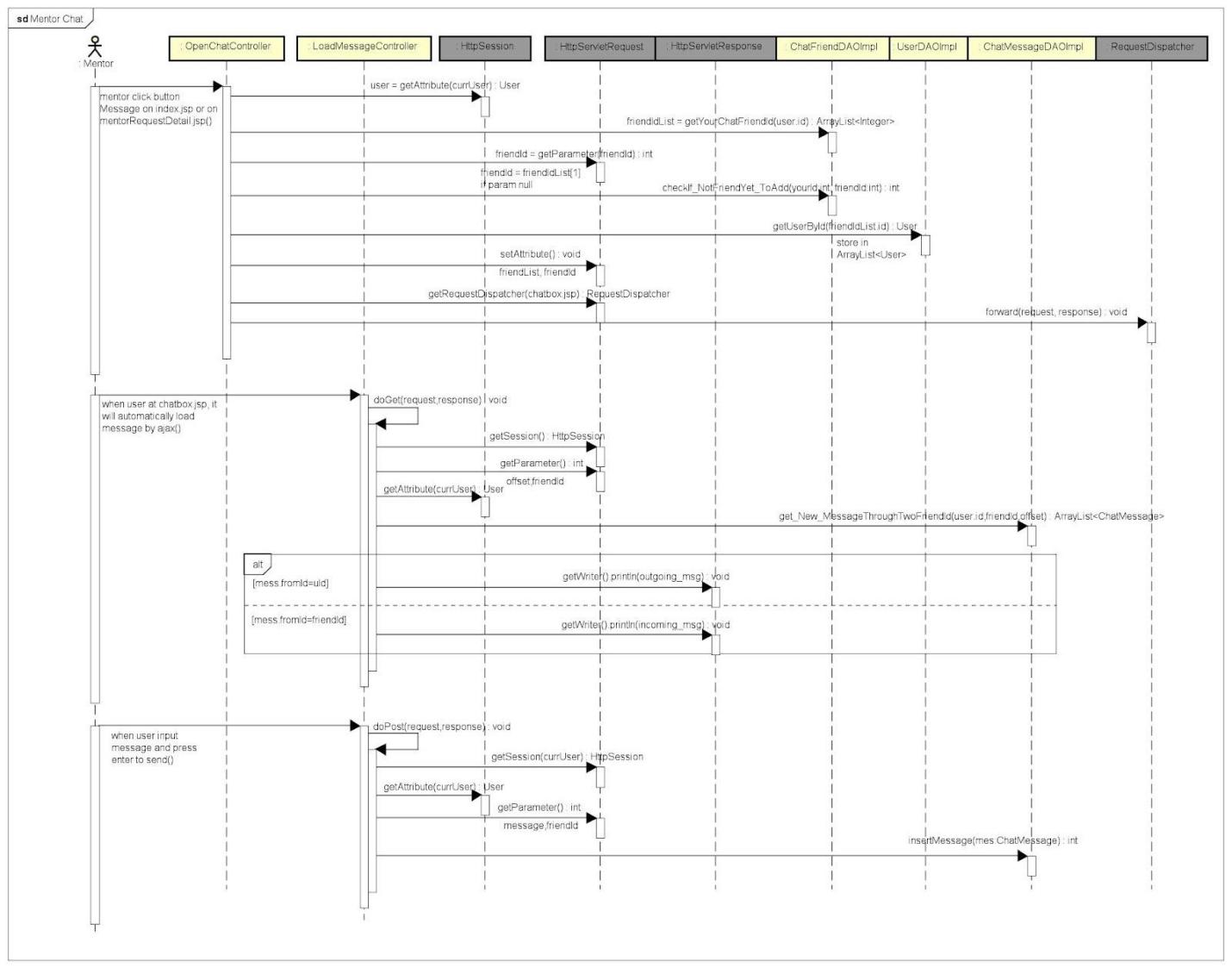
### 

ChatMessageDAO

### 

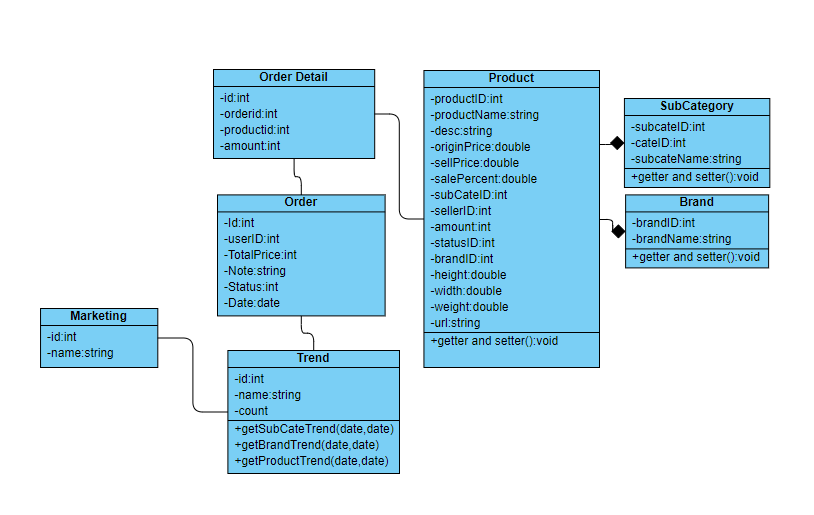
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getNewMessageThroughTwoFriendId(uId, friendId, offset) | * This method will ignore some first messages and load new message between two friend * param: uId (your id) * friendId (friend id) * offset (the number of message loaded and not need loaded again) * return: ArrayList<ChatMessage> * throws Exception |
| 02 | insertMessage(ChatMessage mes) | * This method will insert new message into database * param: mess( ChatMessage object ) * return: int (1 if add mess success) * throws Exception |

### c. Sequence Diagram



## 43. Marketing Dashboard

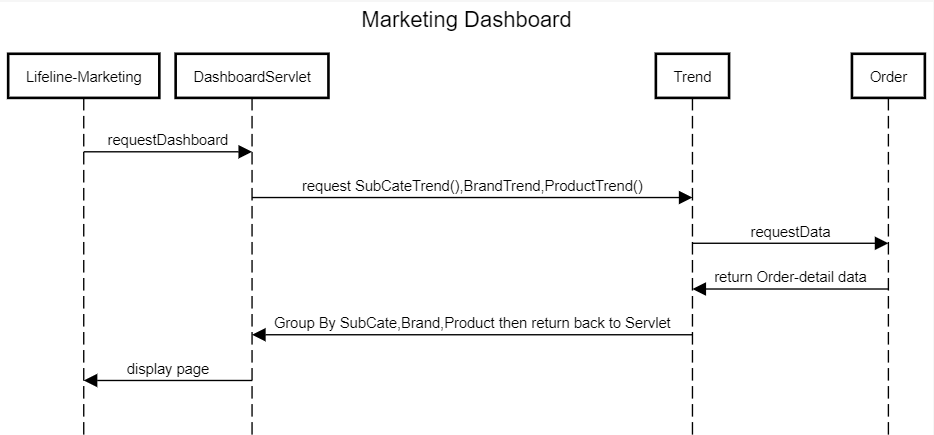
### a. Class Diagram

**

### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getSubCateTrend(date,date)* | *get Sub Category Ordered Trend between 2 date* |
| *02* | *getBrandTrend(date,date)* | *get Brand Ordered Trend between 2 date* |
| *03* | *getProductTrend(date,date)* | *get Top 10 Product Ordered Trend between 2 date* |

### c. Sequence Diagram(s)

**

### d. Database queries

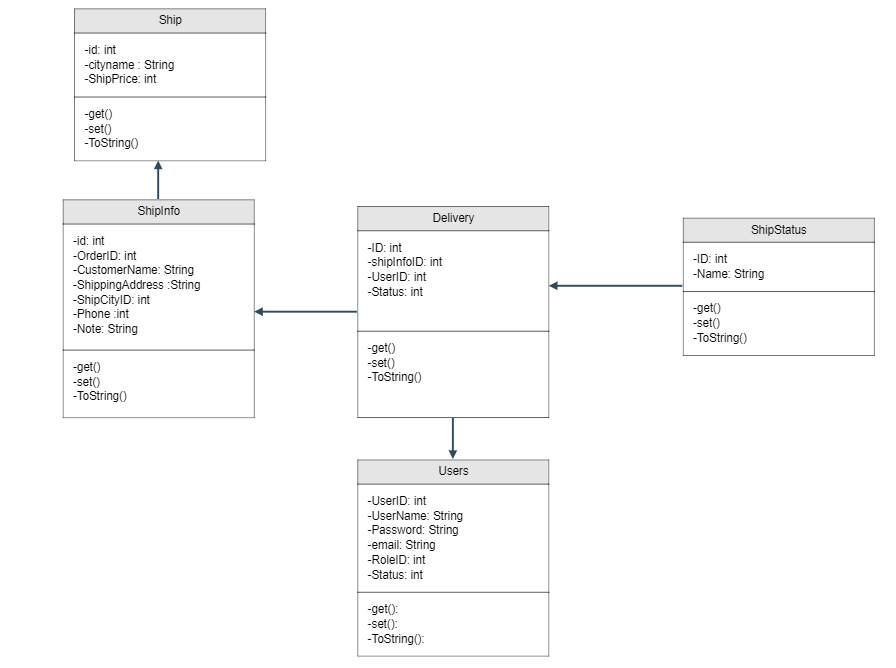
SELECT odr.SubCategoryID,SubCategoryName,SUM(odr.Quantity) AS Count FROM ( SELECT od.ProductID,od.ProductName,od.Quantity,od.Date,SubCategoryID,BrandID FROM (SELECT ProductID,ProductName,Quantity,Date FROM dbo.Orders JOIN dbo.Order\_Detail ON Order\_Detail.Order\_ID = Orders.ID ) od JOIN dbo.Product ON Product.ProductID = od.ProductID WHERE od.Date BETWEEN CAST( ? AS DATE) AND CAST( ? AS DATE) ) odr JOIN dbo.SubCategory ON SubCategory.SubCategoryID = odr.SubCategoryID GROUP BY odr.SubCategoryID,SubCategoryName ORDER BY COUNT DESC

SELECT odr.BrandID,BrandName,SUM(odr.Quantity) AS Count FROM (SELECT od.ProductID,od.ProductName,od.Quantity,od.Date,SubCategoryID,BrandID FROM (SELECT ProductID,ProductName,Quantity,Date FROM dbo.Orders JOIN dbo.Order\_Detail ON Order\_Detail.Order\_ID = Orders.ID ) od JOIN dbo.Product ON Product.ProductID = od.ProductID WHERE od.Date BETWEEN CAST( ? AS DATE) AND CAST( ? AS DATE) ) odr JOIN dbo.Brand ON Brand.BrandID = odr.BrandID GROUP BY odr.BrandID,BrandName ORDER BY COUNT DESC

SELECT TOP 10 odr.ProductID,odr.ProductName,SUM(odr.Quantity) AS Count FROM ( SELECT od.ProductID,od.ProductName,od.Quantity,od.Date,SubCategoryID,BrandID FROM (SELECT ProductID,ProductName,Quantity,Date FROM dbo.Orders JOIN dbo.Order\_Detail ON Order\_Detail.Order\_ID = Orders.ID ) od JOIN dbo.Product ON Product.ProductID = od.ProductID WHERE od.Date BETWEEN CAST(? AS DATE) AND CAST( ? AS DATE) ) odr GROUP BY odr.ProductID, odr.ProductName ORDER BY COUNT DESC

## 43. DeliveryJob

### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getRoleByUserName()* | *get user by user id* |
| *02* | *getJobByShipper(id)* | *get list job by shipper id* |
| *03* | *getShipInfoByJob(listjob)* | *get list shipInfo by list job* |
| *04* | *getShip()* | *get all ship* |

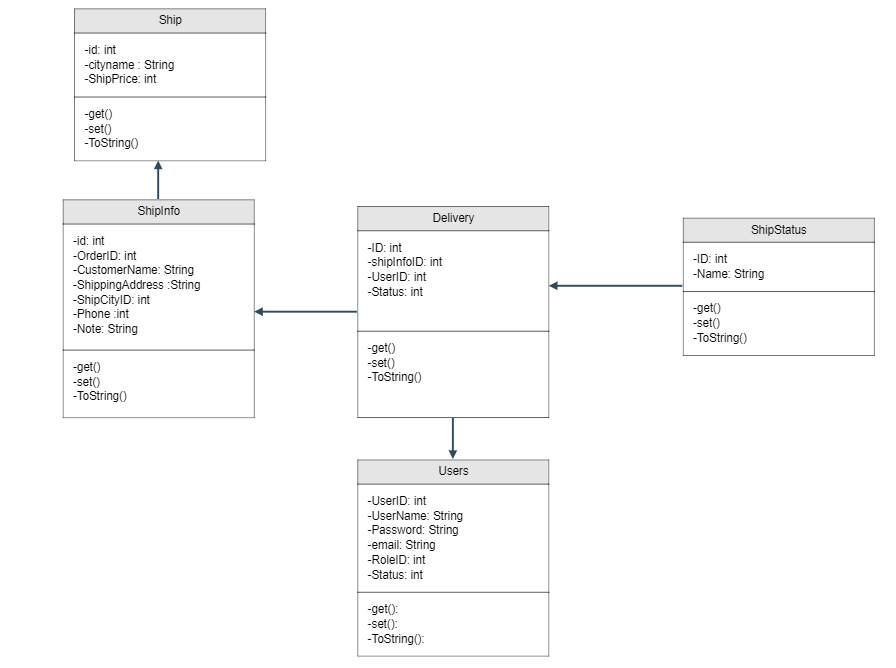
### c. Sequence Diagram(s)

### d. Database queries

* select \* From delivery where UserID = ?
* select \* from ShipInfo where ID = ?
* SELECT \* from dbo.Ship

## 44. Delivery Details

### a. Class Diagram



### b. Class Specifications

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getOrderPackaging()* | *get order packaging()* |
| *02* | *CheckOrderExist( orderID, listo)* | *Check Order is Exist in list order* |
| *03* | *getShipInfoByOdID(id)* | *get ShipInfo by orderID* |
| *04* | *getOrderByOrderID(oid)* | *get Order By order ID* |
| *05* | *getShip()* | *get all ship* |

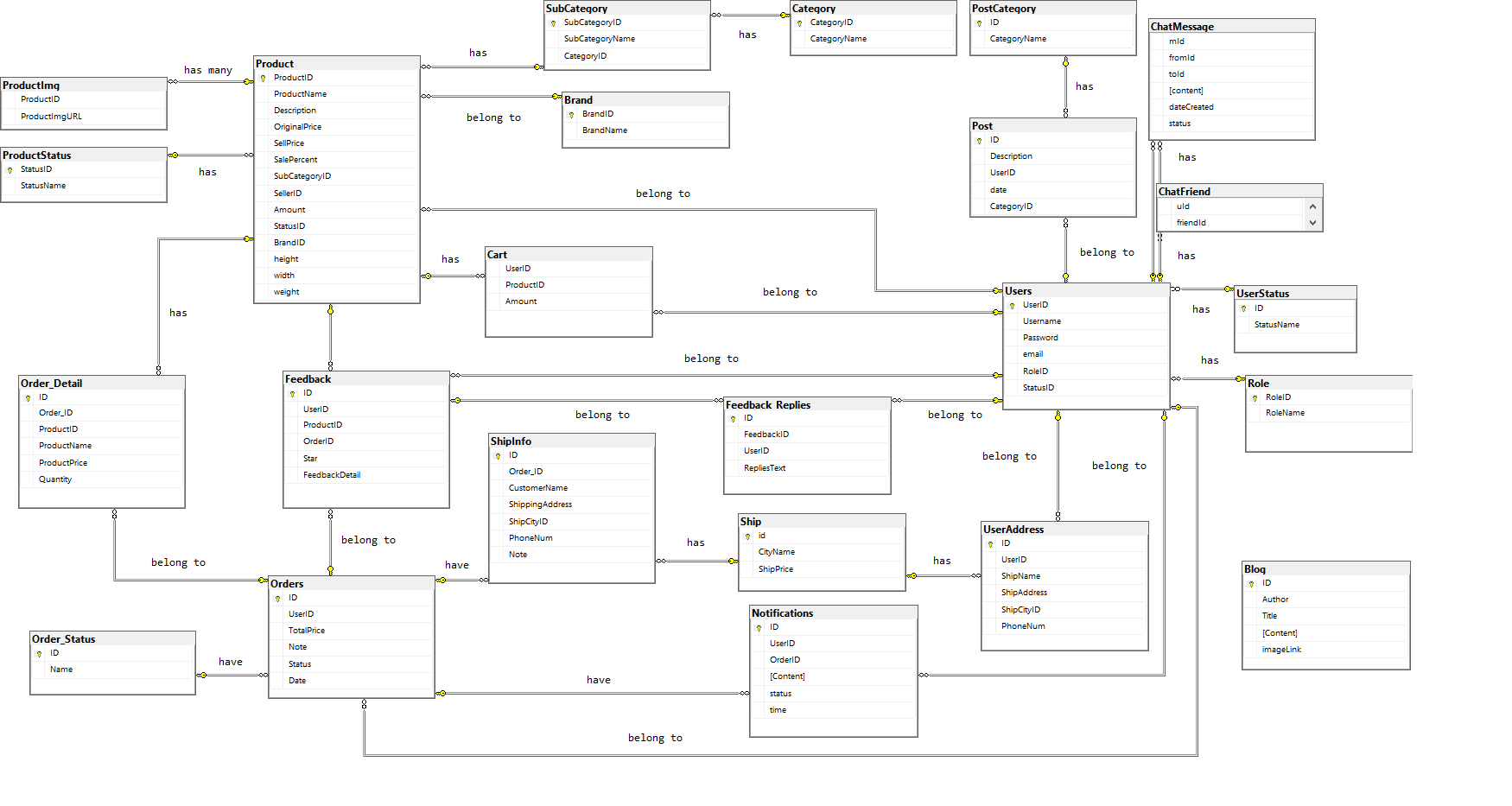
### c. Sequence Diagram(s)

### d. Database queries

* select \* from ShipInfo where Order\_ID = ?
* select o.ID , o.UserID , o.TotalPrice , o.Note , o.Status , o.Date , d.Order\_ID , d.ProductID , d.ProductName ,d.ProductPrice , d.Quantity from Orders o join Order\_Detail d on d.Order\_ID = o.ID where o.ID = ?
* SELECT \* from dbo.Ship

# III. Database Tables

## 1. Database Schema



## 2. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *Blog* | *Blog content + image*  *- Primary keys: ID*  *- Foreign keys:* |
| *02* | *Role* | *Role of a user ( Admin,Marketing,Sale,Customer )*  *- Primary keys: RoleID*  *- Foreign keys:* |
| *03* | *UserStatus* | *Account activation status*  *- Primary keys: ID*  *- Foreign keys:* |
| *04* | *Ship* | *Shipping price for each city*  *- Primary keys: id*  *- Foreign keys:* |
| *05* | *Order\_Detail* | *Detail of each product in order*  *- Primary keys:*  *- Foreign keys: OrderID, ProductID* |
| *06* | *UserAddress* | *Address for each user*  *- Primary keys:ID*  *- Foreign keys: UserID, ShipCityID* |
| *07* | *Users* | *Users*  *- Primary keys: UserID*  *- Foreign keys: RoleID,StatusID* |
| *08* | *Notification* | *Notification data*  *- Primary keys: ID*  *- Foreign keys: OrderID, UserID* |
| *09* | *ShipInfo* | *Shipping info for each order*  *- Primary keys: ID*  *- Foreign keys: Order\_ID, ShipCityID* |
| *10* | *Order* | *Order data*  *- Primary keys: ID*  *- Foreign keys: UserID,Status* |
| *11* | *Feedback* | *Feedback of a product order*  *- Primary keys: ID*  *- Foreign keys: UserID,ProductID, OrderID* |
| *12* | *Feedback\_Replies* | *Replies of feedbacks*  *- Primary keys: id*  *- Foreign keys: FeedbackID,UserID* |
| *13* | *Order\_Status* | *Status of a order*  *- Primary keys: ID*  *- Foreign keys:* |
| *14* | *Product* | *Product Data*  *- Primary keys: ProductID*  *- Foreign keys: SubCategoryID,SellerID,StatusID,BrandID* |
| *15* | *Category* | *Category Data*  *- Primary keys: CategoryID*  *- Foreign keys:* |
| *16* | *SubCategory* | *Sub Category ID*  *- Primary keys: SubCategoryID*  *- Foreign keys: CategoryID* |
| *17* | *Productimg* | *Image of products*  *- Primary keys:*  *- Foreign keys: ProductID* |
| *18* | *ProductStatus* | *Status of a product*  *- Primary keys: StatusID*  *- Foreign keys: StatusName* |
| *19* | *Brand* | *Brand data*  *- Primary keys: BrandID*  *- Foreign keys:* |
| *20* | *Cart* | *Cart info for each user*  *- Primary keys: UserID*  *- Foreign keys: ProductID* |
| *21* | *CBanner* | *Carousel Banner*  *-Primary keys: BannerID* |
| *22* | *Banner* | *Ads Banner*  *-Primary keys: BannerID* |
| *23* | *ChatMessage* | *Message*  *- Primary keys:*  *- Foreign keys: fromId, toId* |
| *24* | *ChatFriend* | *Chat’s Members*  *- Primary keys:*  *- Foreign keys: uId,FriendId* |

## 