

**ONLINE SHOPPING**

**Software Design Document**

– Hanoi, September 2021 –

**Table of Contents**

[I. Overview](#_gjdgxs) **9**

[Technical](#_rqt14idk0qov) 9

[Code Packages/Namespaces](#_30j0zll) 9

[**II. Code Designs**](#_arzedbub19nc) **11**

[1. Feature Product](#_2et92p0) 11

[a. Class Diagram](#_tyjcwt) 11

[b. Class Specifications](#_3dy6vkm) 11

[c. Sequence Diagram(s)](#_1t3h5sf) 12

[d. Database queries](#_4d34og8) 12

[2. Search & Filter Product](#_fmffx3f4d8dy) 13

[a. Class Diagram](#_ftkw50ep6ggo) 13

[b. Class Specifications](#_ut2lgfpj220g) 13

[c. Sequence Diagram(s)](#_d046ap1nhzrj) 14

[d. Database queries](#_7k0beimjbxa8) 14

[a. Class Diagram](#_ebr71nt3gqsz) 15

[b. Class Specifications](#_ezpe68u82w9) 15

[c. Sequence Diagram(s)](#_xve4p0oildn6) 15

[d. Database queries](#_op8e1o34zuen) 16

[4. Change Amount](#_mbwqpx8lj17i) 17

[a. Class Diagram](#_dxcc2r9ee4wu) 17

[b. Class Specifications](#_9lh6ewwema0) 17

[c. Sequence Diagram(s)](#_nnsb3nn5j6tq) 18

[d. Database queries](#_8pi6so2rfkr1) 18

[5. Delete Product From Cart](#_w31em9pj27dg) 19

[a. Class Diagram](#_15ud8tdghax5) 19

[b. Class Specifications](#_vbmqfmdqp8e3) 19

[c. Sequence Diagram(s)](#_lz2b46tt2lwz) 20

[d. Database queries](#_wpogvtbck3ee) 20

[6. Delete All Cart Item](#_r3pst5b4ugue) 20

[a. Class Diagram](#_hiuud34sxqir) 21

[b. Class Specifications](#_xtza7p459w77) 21

[c. Sequence Diagram(s)](#_x9tf6lk4ob3) 21

[d. Database queries](#_wtp57vymeeip) 22

[7. Login](#_x0g6lstz4r5f) 22

[a. Class Diagram](#_3peb5hdql8vs) 22

[b. Class Specifications](#_bvza3sr9uutb) 23

[c. Sequence Diagram(s)](#_uk6l7ka8qzk) 24

[d. Database queries](#_b5npzotuk8wk) 24

[8. Register](#_prjh0p8xx0fa) 24

[a. Class Diagram](#_usvmt3nowi) 24

[b. Class Specifications](#_3whwml4) 25

[c. Sequence Diagram(s)](#_gt4h709mx8gd) 26

[d. Database queries](#_hzz3vrxo39gi) 26

[9. View User List](#_t5hi4o41rjqg) 27

[a. Class Diagram](#_1oyu5552ny5i) 27

[b. Class Specifications](#_6jgonxeh1aes) 27

[c. Sequence Diagram(s)](#_traq8b6r9mvz) 28

[d. Database queries](#_blb0j3pxmlvu) 28

[10. Edit User](#_loqng3cs3ju1) 29

[a. Class Diagram](#_r4ekm154b5o4) 29

[b. Class Specifications](#_sokhyrpovkbg) 29

[c. Sequence Diagram(s)](#_my3le0k03fwl) 31

[d. Database queries](#_xyboyj2k8fjz) 31

[11. Delete User](#_jgxjoss5zajj) 31

[a. Class Diagram](#_3cxjlx557jsc) 31

[b. Class Specifications](#_70b7pxkeiahm) 31

[c. Sequence Diagram(s)](#_58vpsuz6n0if) 32

[d. Database queries](#_m7s8ky8l43xo) 32

[12. Customers List[Seller]](#_zg9tynzfaztq) 33

[a. Class Diagram](#_neraeq1g1k96) 33

[b. Class Specifications](#_47wy92gneo6x) 33

[c. Sequence Diagram(s)](#_adnnquhro8pz) 34

[d. Database queries](#_5bzmeugvfryk) 34

[13. Add Ship Information](#_sj8oaxx96dxm) 34

[a. Class Diagram](#_1pk92scu7szj) 34

[b. Class Specifications](#_hujqiyx0abjl) 34

[c. Sequence Diagram(s)](#_gtu7ymvnucvm) 35

[14. Change Ship Information](#_5r5lnajjv952) 35

[a. Class Diagram](#_1i7zo0wgob) 35

[b. Class Specifications](#_g9hssiud3rka) 36

[c. Sequence Diagram(s)](#_nf69x81u2i39) 36

[d. Database queries](#_ulmffnqswlss) 36

[a. Class Diagram](#_ljof3opcc309) 37

[b. Class Specifications](#_ugum98w38ykd) 37

[c. Sequence Diagram(s)](#_7df103g8fsnm) 38

[16. Change Password](#_i3pblysga2ll) 39

[a. Class Diagram](#_202zkr4g8gmy) 39

[b. Class Specifications](#_gxrde6lse1i6) 39

[c. Sequence Diagram(s)](#_qy49vjy1d7jl) 40

[d. Database queries](#_52erbdmtlxv4) 40

[17. User Profile](#_91rexp979ukm) 41

[a. Class Diagram](#_dn07gbeq8oqx) 41

[b. Class Specifications](#_go9fnhfg19il) 41

[c. Sequence Diagram(s)](#_gs1zn9m4xkzv) 42

[18. My Orders](#_6o47ybp751ym) 42

[a. Class Diagram](#_jntcsqopgt0) 42

[b. Class Specifications](#_ngqzlczhx6lw) 42

[c. Sequence Diagram(s)](#_dcsvsc8hzqfc) 43

[19. Order Information](#_ybr4thhe4kb0) 44

[a. Class Diagram](#_y14k9xigftcg) 44

[c. Sequence Diagram(s)](#_yf3x0bfclao7) 45

[d. Database queries](#_j230zpuinqm7) 45

[d. Database queries](#_funhc7g1u18o) 47

[22. Order Details](#_97ucyetbgnes) 48

[a. Class Diagram](#_8yl53fgx0wjx) 48

[b. Class Specifications](#_jzk9rnpaf9fe) 48

[c. Sequence Diagram(s)](#_mt5e8o2wpv6p) 49

[d. Database queries](#_3iwa5vbhkiuy) 49

[23. View Feedback](#_hjzyarr6va9w) 49

[a. Class Diagram](#_lsm03e72umnc) 49

[b. Class Specifications](#_kgd5gn8cbqhi) 50

[c. Sequence Diagram(s)](#_x4u1ej97u40q) 50

[d. Database queries](#_nq41jx9z959k) 51

[24. Send Feedback](#_t8afvtpiwu2v) 51

[a. Class Diagram](#_fpkxjipz6pzd) 51

[b. Class Specifications](#_td2xfnwneeif) 51

[c. Sequence Diagram(s)](#_jpn4h8gewjfr) 52

[d. Database queries](#_fu9zdc9d1zxv) 52

[25. Blogs List](#_3rj0bq5fxxmp) 53

[a. Class Diagram](#_kv0t4guvmihm) 53

[b. Class Specifications](#_8syb3hrc91lf) 53

[c. Sequence Diagram(s)](#_u1znqwjqpo12) 53

[d. Database queries](#_2e2nbxswqud6) 54

[26. Blog Details](#_jbcivhdnucyr) 54

[a. Class Diagram](#_y07veu7531nb) 54

[b. Class Specifications](#_u6gddxntajuc) 54

[c. Sequence Diagram(s)](#_iw8c2o5l0g56) 55

[d. Database queries](#_pkw5oibxw4mt) 55

[27. Blog Management](#_q48lgs5qgw91) 56

[a. Class Diagram](#_3pk5edtg4x83) 56

[b. Class Specifications](#_gbxhsxrz1ola) 56

[c. Sequence Diagram(s)](#_jm97mz2js878) 56

[d. Database queries](#_n8zd2axwqgt1) 57

[28. View Notification](#_5xm5w1ir1o8e) 58

[a. Class Diagram](#_gm35s0fckwxu) 58

[b. Class Specifications](#_jeug5ky84kzx) 58

[c. Sequence Diagram(s)](#_w82de7vzw9t3) 59

[d. Database queries](#_z5zns91r2n2i) 59

[29. Mark as Read](#_9n44amaqerzo) 59

[a. Class Diagram](#_t5gyrkxdqf8q) 59

[b. Class Specifications](#_8nc3bagk5ncw) 60

[c. Sequence Diagram(s)](#_fny6sj308us5) 61

[d. Database queries](#_hhx70h5d0kgc) 61

[30. Admin Feature](#_drxwhhnv73x8) 61

[a. Class Diagram](#_nu7d0pu09y74) 62

[b. Class Specifications](#_gamuu53w76je) 62

[c. Sequence Diagram(s)](#_7x1499yira3f) 62

[d. Database queries](#_8nnokyshs4ng) 62

[31. Add Product](#_bwuay1y70ues) 64

[a. Class Diagram](#_490a2x4onc9c) 64

[b. Class Specifications](#_b7pnn9qkoapw) 64

[c. Sequence Diagram(s)](#_j0vpj0hpwvd1) 64

[d. Database queries](#_7bv97cyc0euq) 64

[32. Delete Product](#_pr20wdqb9oqh) 66

[a. Class Diagram](#_kxok5kysl82y) 66

[b. Class Specifications](#_kyp3s1mkhfeg) 66

[c. Sequence Diagram(s)](#_ka7jnh50914d) 66

[d. Database queries](#_7sevwczq7uy) 66

[33. Edit Product](#_30v23pa1ersb) 68

[a. Class Diagram](#_9nayyqb5022b) 68

[b. Class Specifications](#_j25qfanox6jh) 68

[c. Sequence Diagram(s)](#_bd3jzc82caun) 68

[d. Database queries](#_2w2fhhnncbl8) 68

[34. Orders Dashboard](#_vwjlqzwqdrn) 69

[d. Database queries](#_x5zf9p9gcxvr) 72

[35. PostDetail](#_robkw7ofmls2) 72

[b. Class Specifications](#_othvay7api0b) 73

[PostControl Class: The doGet function is used to get the ID of the post and then use the doPost function to edit the product](#_2wu18netd5gy) 73

[DeletepostControll Class: use process request function to get and delete post by ID](#_czxmyehnj75l) 74

[c. Sequence Diagram(s)](#_s60a3gdo7a8m) 74

[d. Database queries](#_fw2r92aw1jz3) 74

[b. Class Specifications](#_oc8ab241msvt) 76

[PostManageControl Class: This class process request for list of post create by user ID](#_445mnup68phn) 76

[d. Database queries](#_u3jpnkce98tl) 77

[37. Error](#_p2bshxdw7cf) 78

[a. Class Diagram](#_jq044lbw3a4z) 78

[b. Class Specifications](#_tt39w7um789g) 78

[c. Sequence Diagram(s)](#_3iile1ivvu6a) 78

[d. Database queries](#_iwt9qgt9bt4e) 78

[38. Manage Feedback [Seller]](#_4s3t9jdektha) 79

[a. Class Diagram](#_o6ty2ktm2y9g) 79

[b. Class Specifications](#_3vn5mcgmqd9i) 79

[c. Sequence Diagram(s)](#_lzj44mqy91yw) 79

[d. Database queries](#_6xtwsaghu4yj) 80

[39. Manage Feedback [Customer]](#_k3r078t8p16y) 80

[a. Class Diagram](#_hvb1jqiuuoqn) 80

[b. Class Specifications](#_dxch87ts8sbj) 80

[c. Sequence Diagram(s)](#_hctdazumay3h) 81

[d. Database queries](#_t9tuliz8s5gb) 81

[40. Manage Banner](#_j59cq3eye572) 81

[a. Class Diagram](#_p1hs9l5fc2wj) 81

[b. Class Specifications](#_i3d999iwus14) 82

[c. Sequence Diagram(s)](#_7bt1devd1edx) 83

[d. Database queries](#_cxqc93cw9yga) 83

[41. Chat](#_gmnxos91ir56) 83

[a. Class Diagram](#_5n8rlp2tmmrr) 84

[b. Class Specifications](#_6k23z4addfq1) 84

[c. Sequence Diagram](#_pymip5wcxgdt) 86

[43. Marketing Dashboard](#_t47gxw15gk5c) 88

[a. Class Diagram](#_7u0c8dc8ow8x) 88

[b. Class Specifications](#_8sbdxvambdaa) 88

[c. Sequence Diagram(s)](#_kax5nx71muo5) 88

[d. Database queries](#_uzqxdf2etjo5) 88

[43. DeliveryJob](#_5syvfshi1jza) 89

[a. Class Diagram](#_wqxgacerj507) 89

[b. Class Specifications](#_53gd65ct28z) 90

[c. Sequence Diagram(s)](#_klkm2ls864z6) 91

[d. Database queries](#_if2384i0nh29) 91

[44. Delivery Details](#_7ylxpso4ytrp) 91

[a. Class Diagram](#_2x2qwsplz0if) 92

[b. Class Specifications](#_2f0tdff87td8) 92

[c. Sequence Diagram(s)](#_sh1adaws6ztw) 93

[d. Database queries](#_ddw4lqqw8u79) 93

[III. Database Tables](#_17dp8vu) **93**

[1. Database Schema](#_o8zg3ltgd1h3) 93

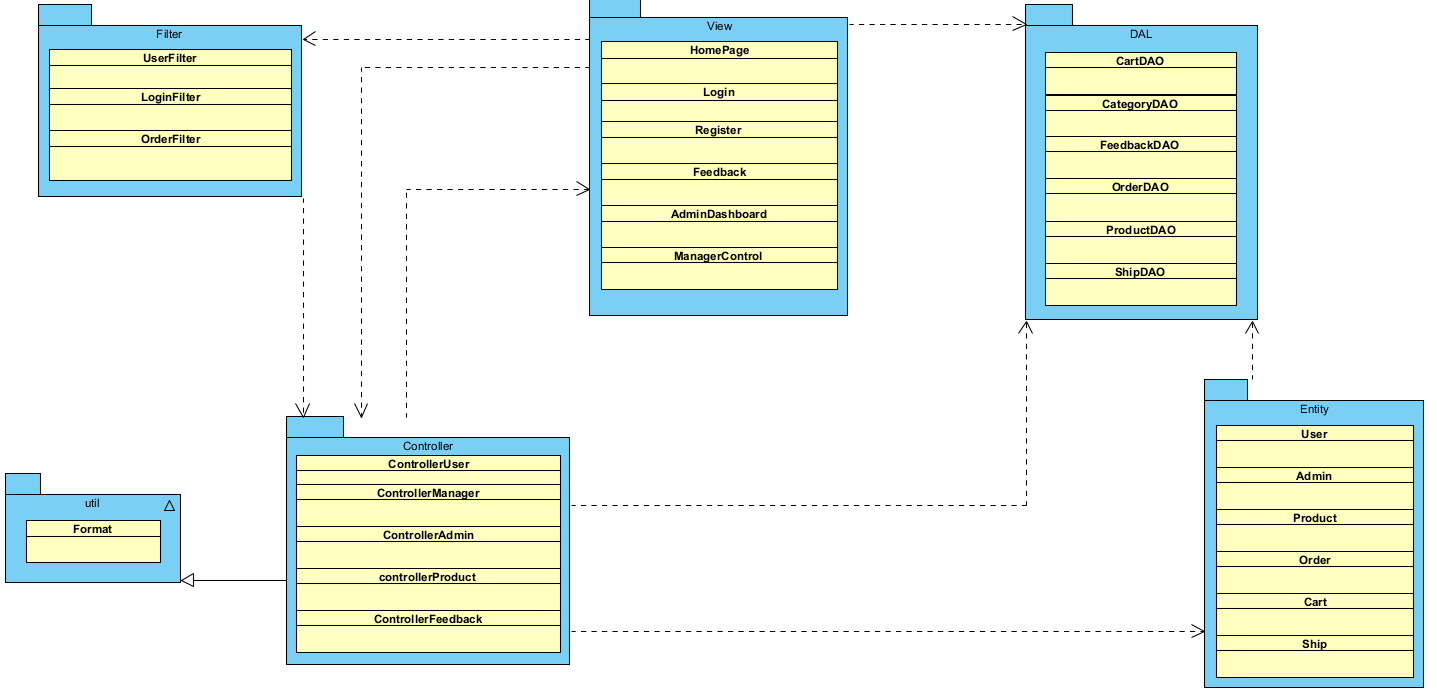
[2. Table Description](#_uxafy98nib3t) 94

# I. Overview

## Technical

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

## Code Packages/Namespaces



***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* | *Util* | *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

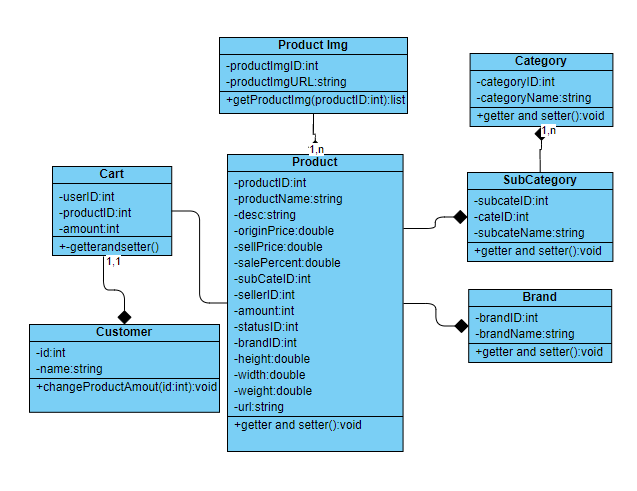
JOIN ProductImg

ON Product.ProductID = ProductImg.ProductID

WHERE Product.ProductID = ?

## 4. Change Amount

### a. Class Diagram

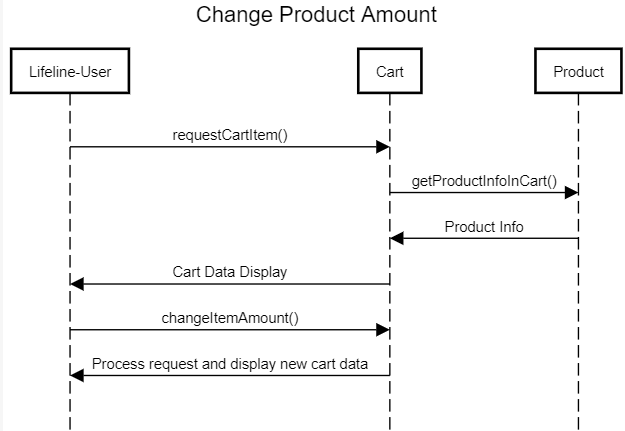


### b. Class Specifications

Customer

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

### c. Sequence Diagram(s)



### d. Database queries

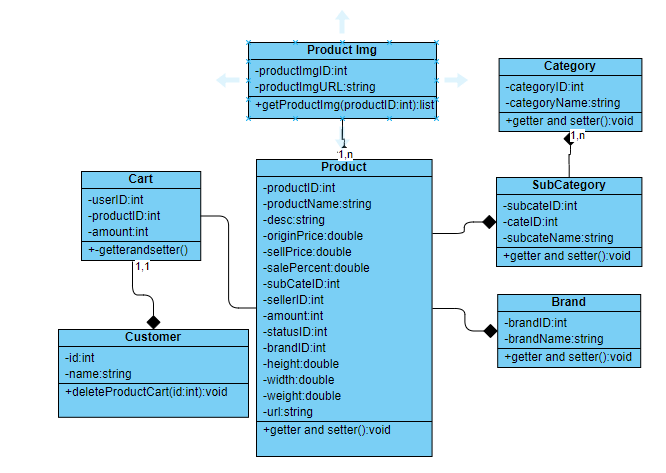
UPDATE dbo.cart

SET amount = ?

WHERE userID = ? and productID = ?

## 5. Delete Product From Cart

### a. Class Diagram

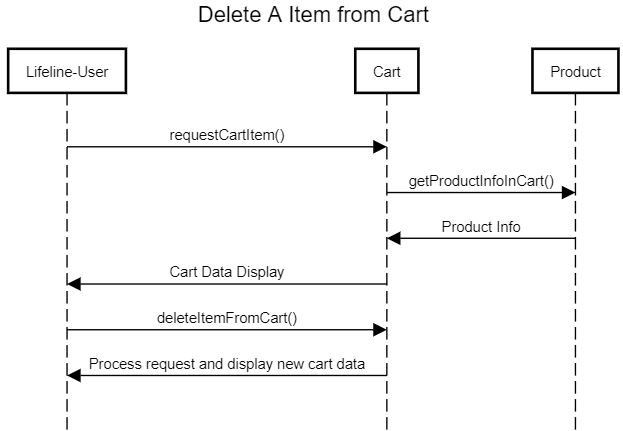


### b. Class Specifications

Customer

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

### c. Sequence Diagram(s)

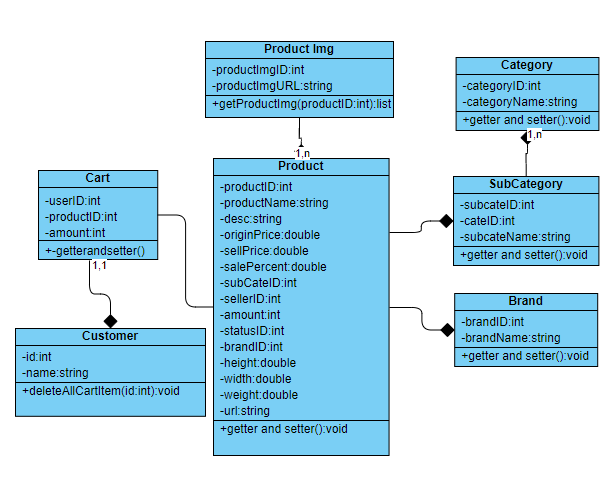


### d. Database queries

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

## 6. Delete All Cart Item

### a. Class Diagram

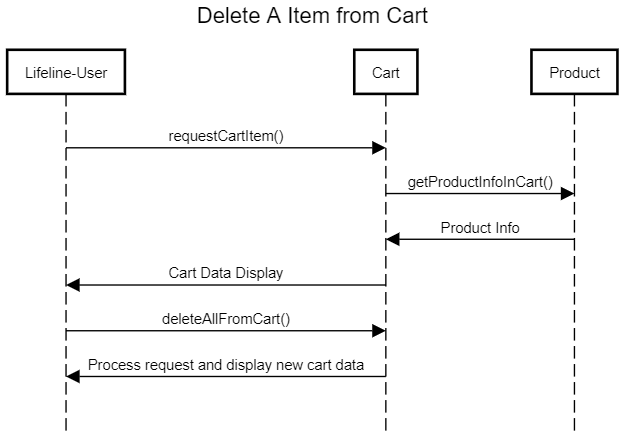


### b. Class Specifications

Customer

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

### c. Sequence Diagram(s)

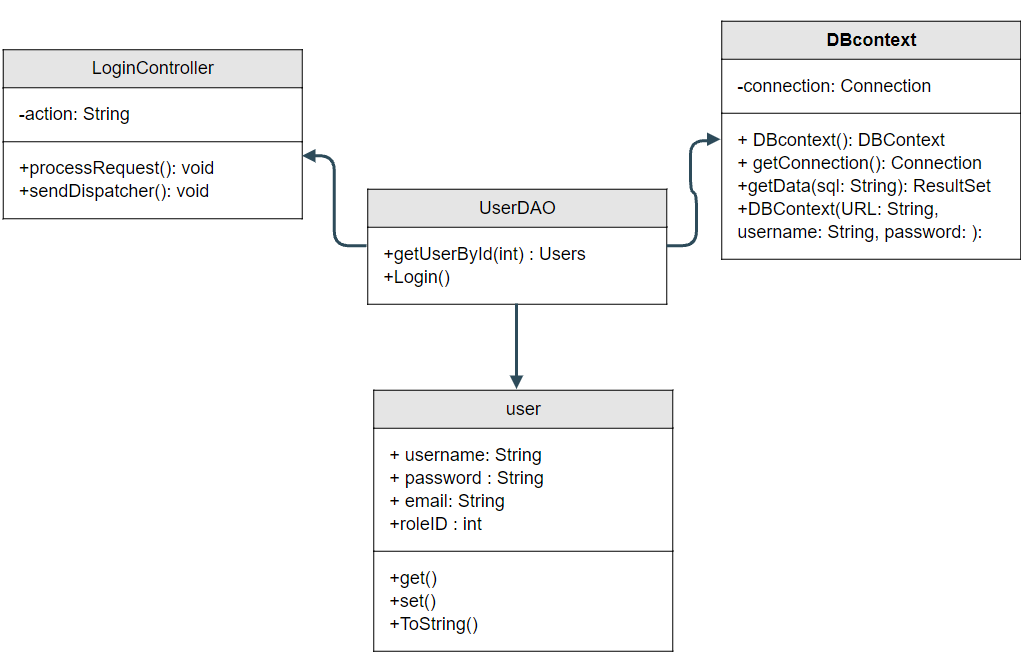


### d. Database queries

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

## 7. Login

### 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)

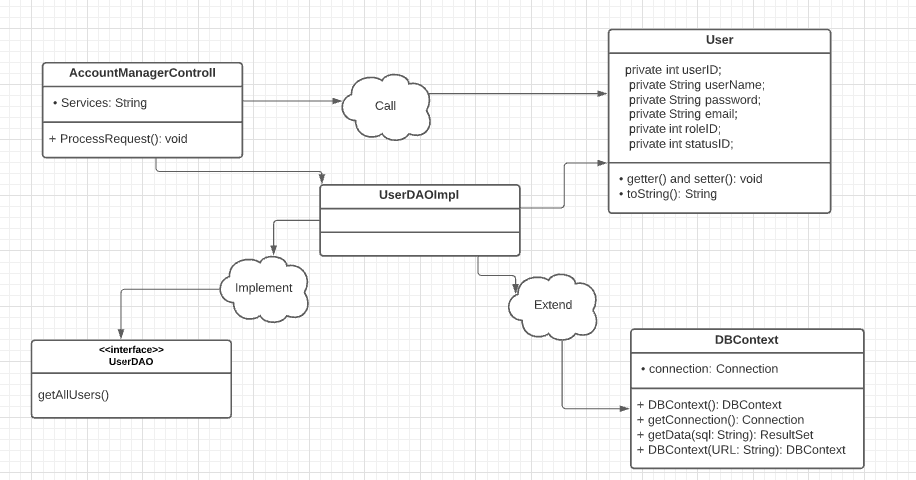
****

### d. Database queries

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

## 9. View User List

### a. Class Diagram



### b. Class Specifications

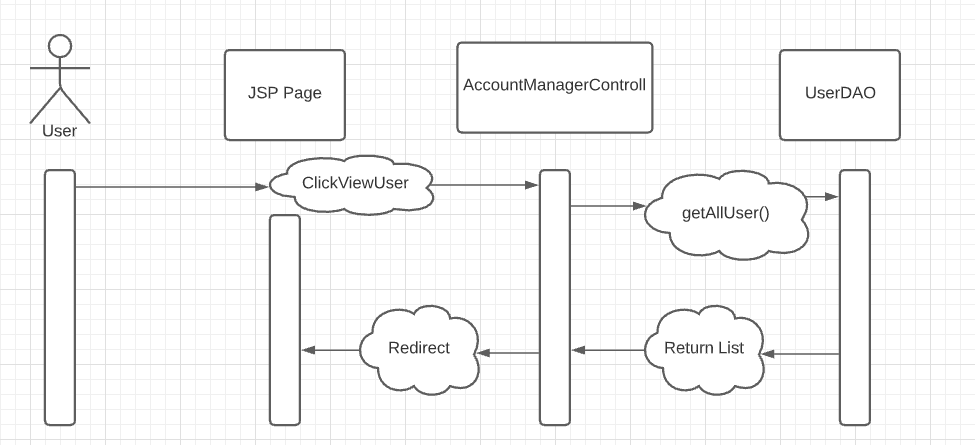
* AccountManagerControll:

|  |  |  |
| --- | --- | --- |
| 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)

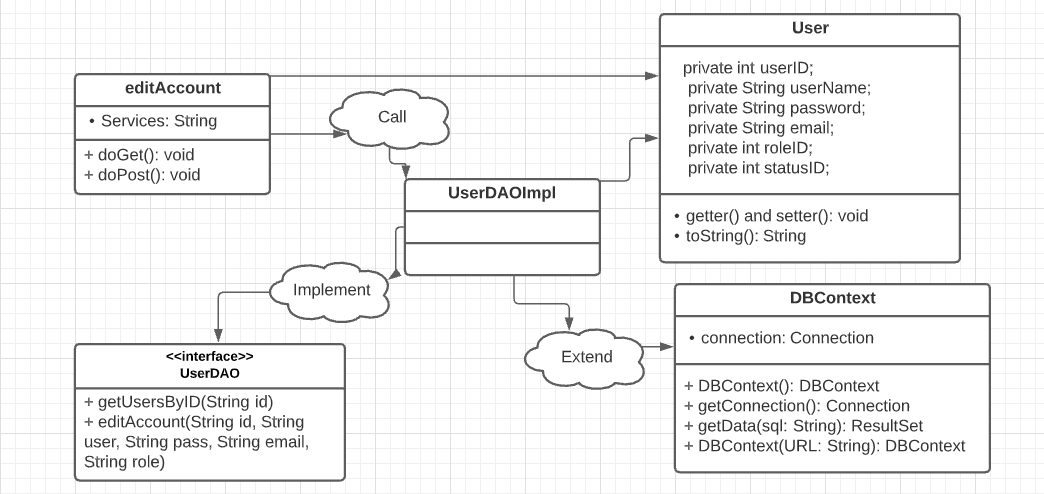


### d. Database queries

* select \* from Users

## 10. Edit User

### a. Class Diagram



### b. Class Specifications

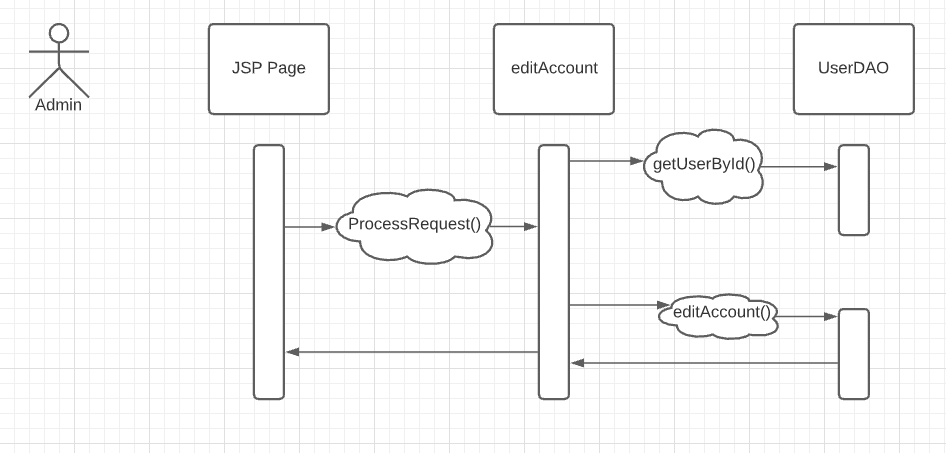
* editAccount:

|  |  |  |
| --- | --- | --- |
| 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)

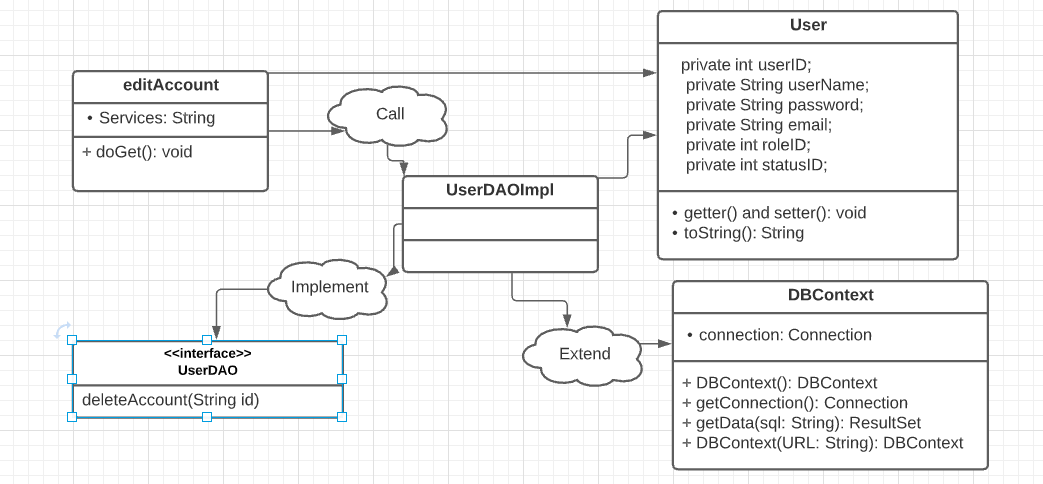


### d. Database queries

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

## 11. Delete User

### a. Class Diagram



### b. Class Specifications

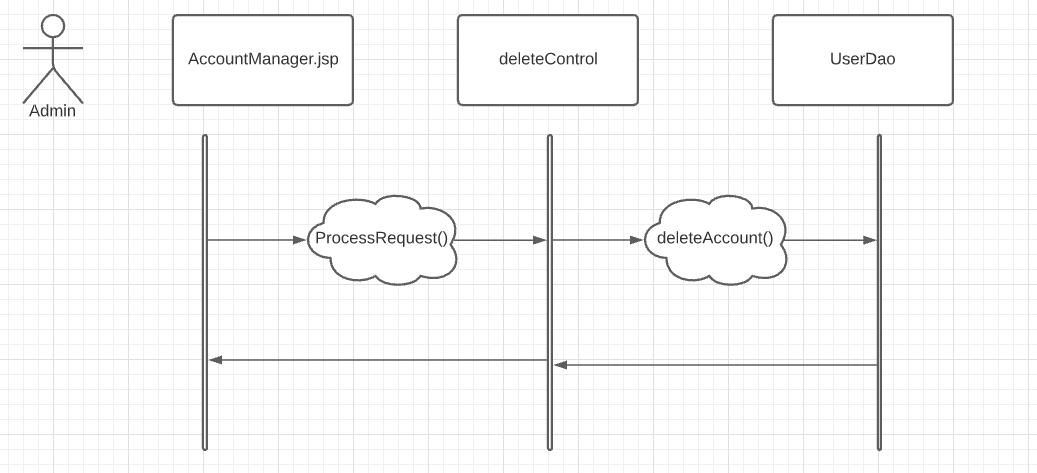
* deleteAccount:

|  |  |  |
| --- | --- | --- |
| 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)

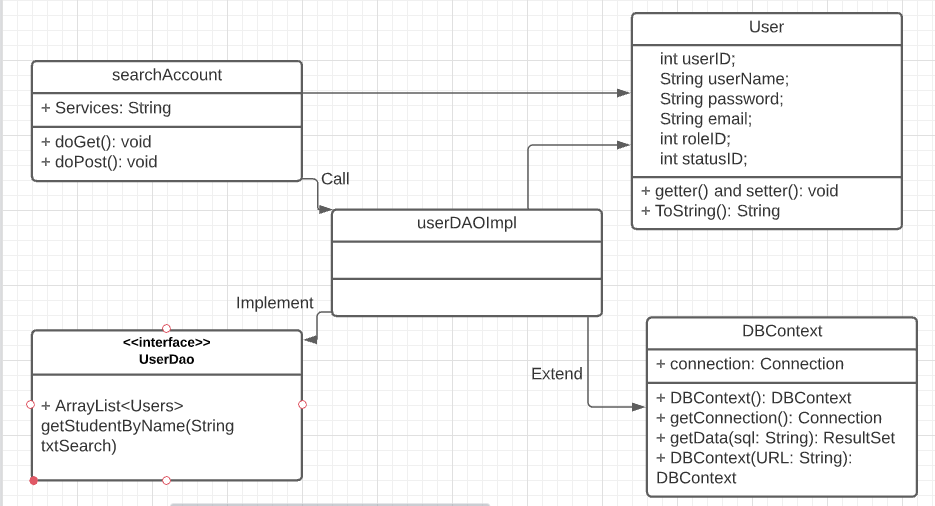


### d. Database queries

* delete from Users where UserID = ?

## 11. Search User

**a. Class Diagram**

****

### b. Class Specifications

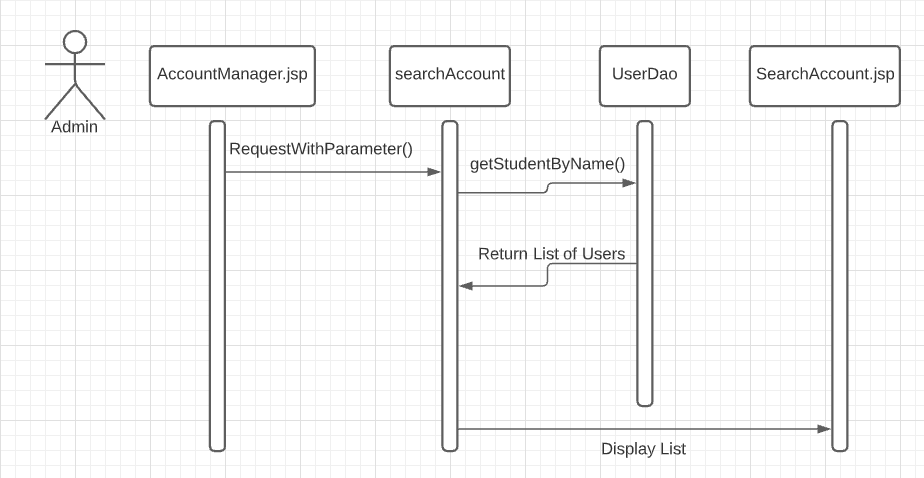
- searchAccount:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | doGet() | * Navigate to the SearchAccount.jsp . page |
| 02 | doPost() | * Get List of Users from database by the first letter of his or her name * Navigate to the SearchAccount.jsp . page |

- UserDao:

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | ArrayList<Users> getStudentByName(String txtSearch) | - *Return: List of user*  *- Input:* The first letter of his or her name  - *Search for a user whose first letter is the same as the one entered by the admin and entered on SearchAccount.JSP page* |

### c. Sequence Diagram(s)

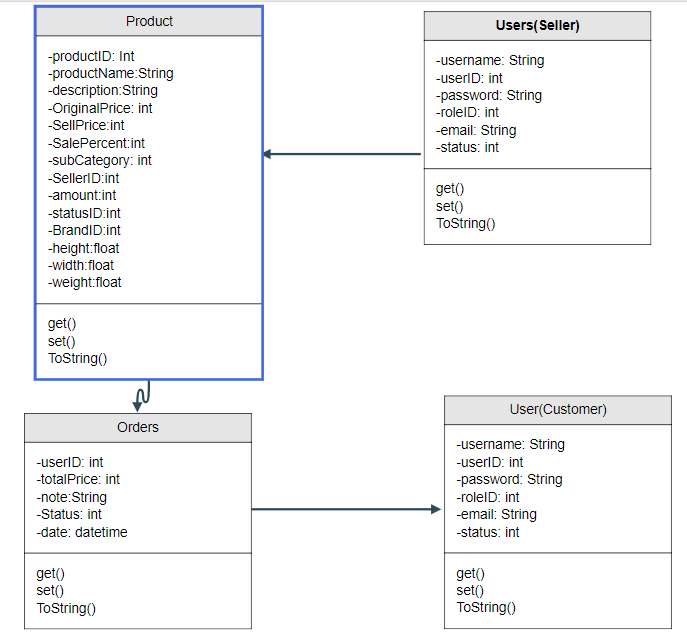


### d. Database queries

+ SELECT \* FROM Users where UserName like '" + txtSearch + "%'

## 12. Customers List[Seller]

### a. Class Diagram

****

### b. Class Specifications

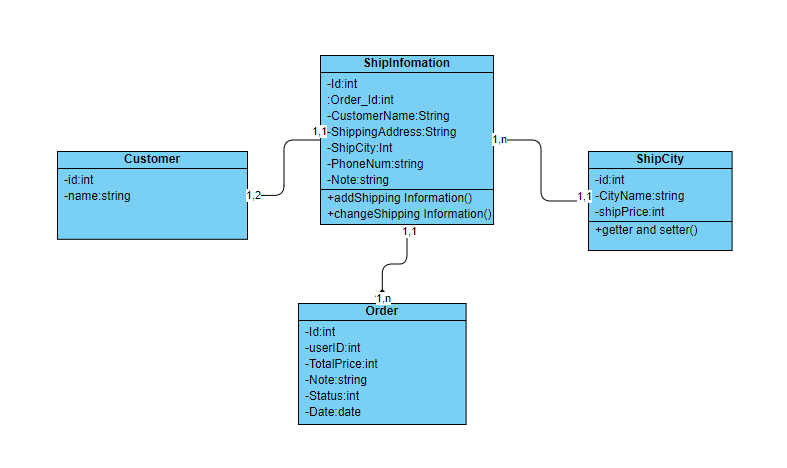
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getUserByUsername() | *get user from dtb by username* |

### c. Sequence Diagram(s)

### d. Database queries

## 13. Add Ship Information

### a. Class Diagram

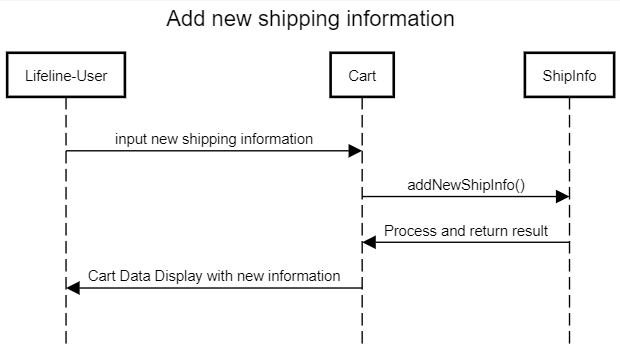


### b. Class Specifications

Ship Information Class

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

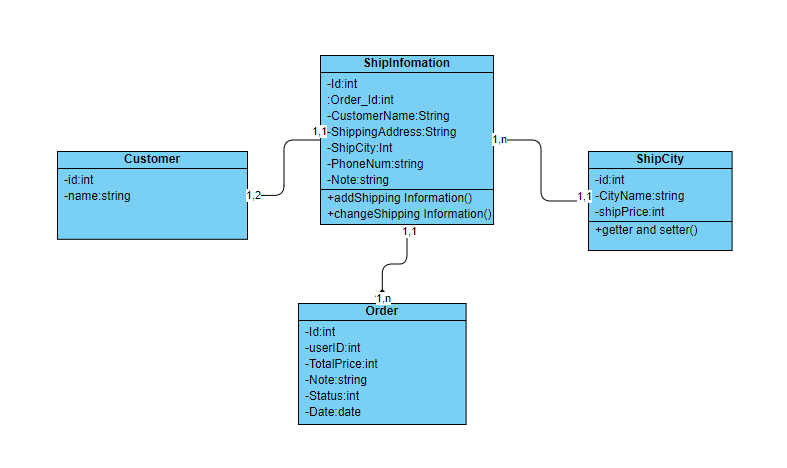
### c. Sequence Diagram(s)

Prod. Database queries

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

## 14. Change Ship Information

### a. Class Diagram

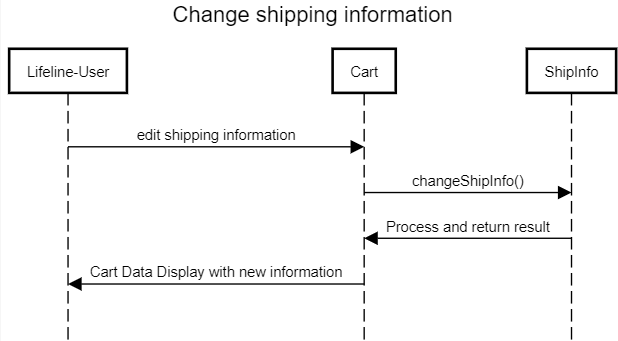


### b. Class Specifications

Ship Information Class

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

### c. Sequence Diagram(s)



### d. Database queries

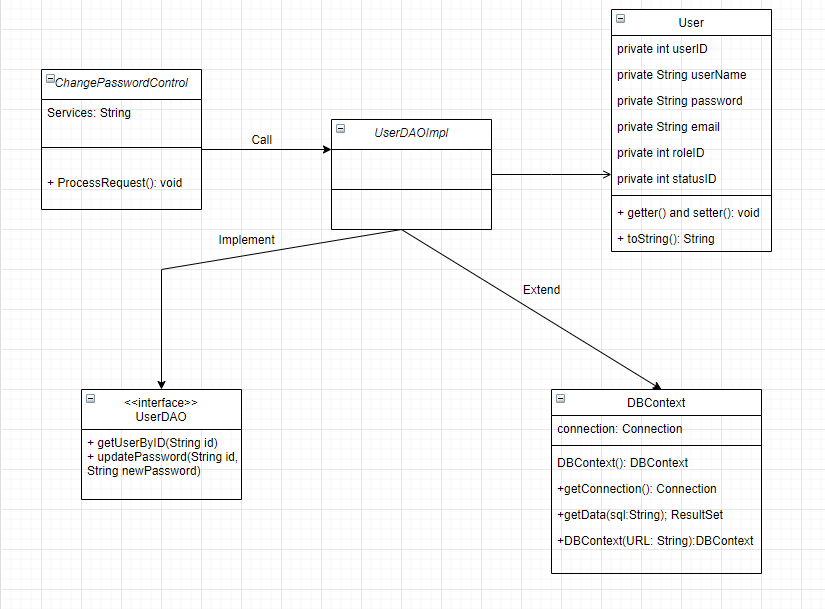
Update dbo.ShipInfomation

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

Where orderID = ?

15. Reset Password

### a. Class Diagram



### b. Class Specifications

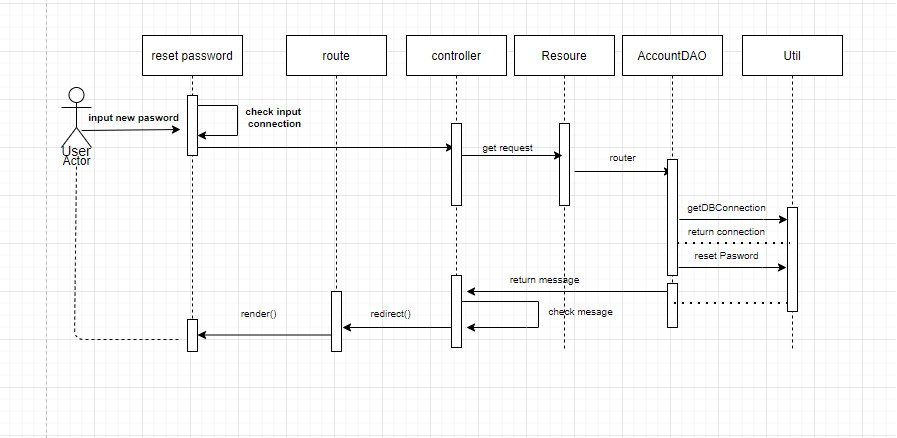
* ResetPasswordControl:

|  |  |  |
| --- | --- | --- |
| 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)

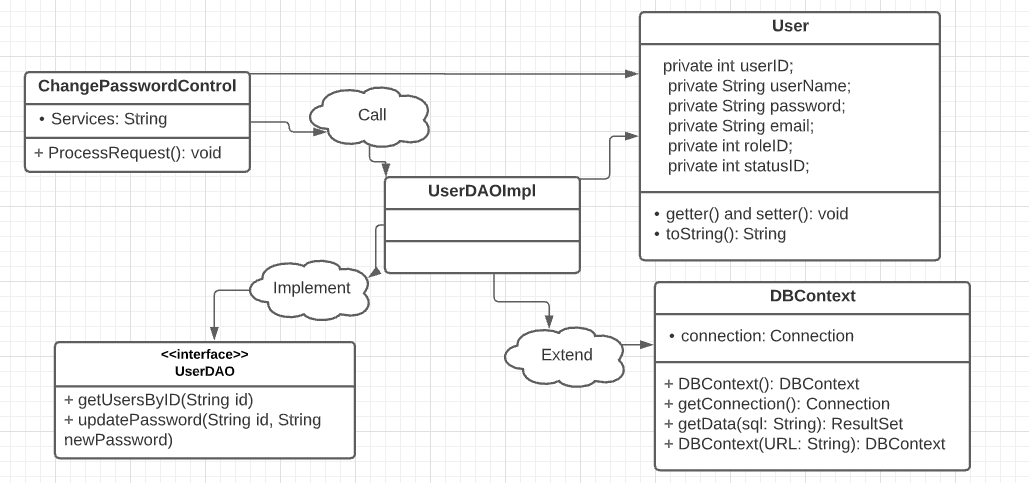


**d. Database queries**

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

## 16. Change Password

### a. Class Diagram



### b. Class Specifications

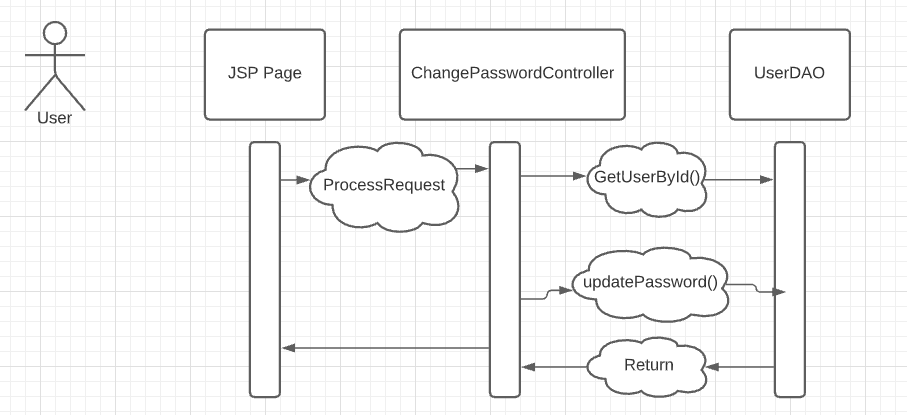
* ChangePasswordControl:

|  |  |  |
| --- | --- | --- |
| 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)

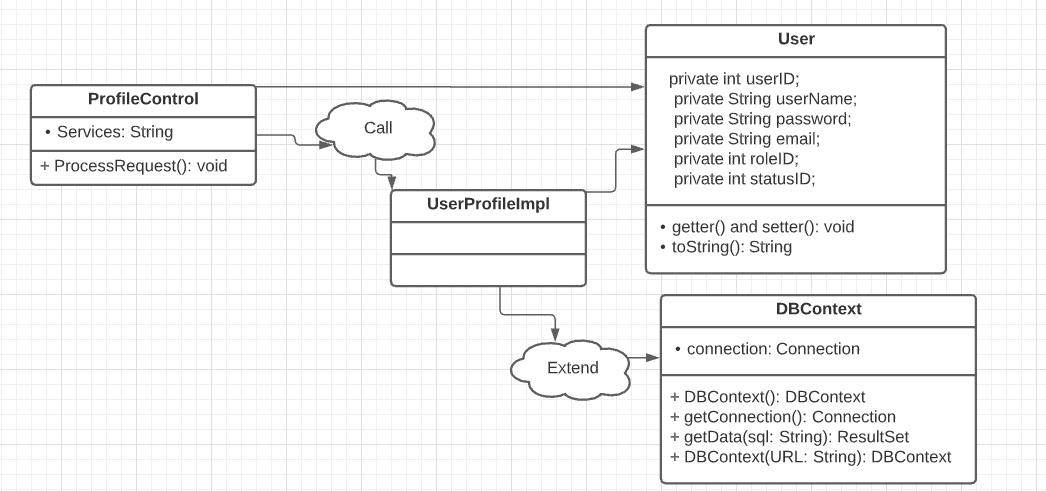


### d. Database queries

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

## 17. User Profile

### a. Class Diagram

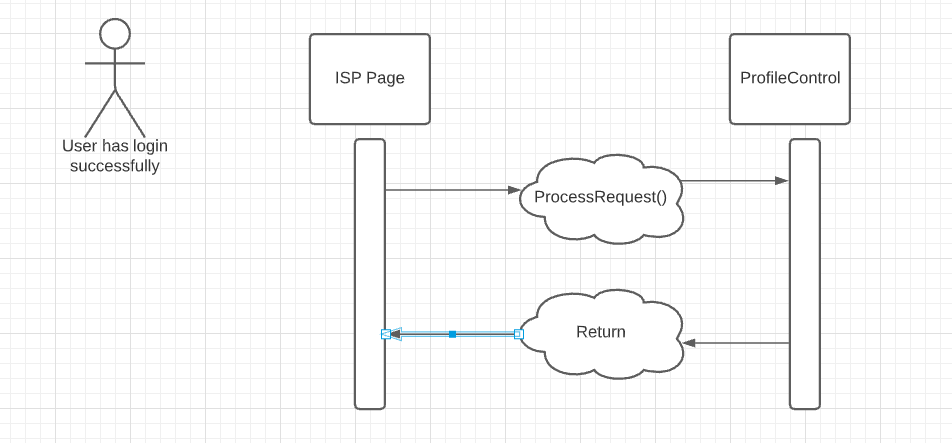


### b. Class Specifications

* ProfileControll:

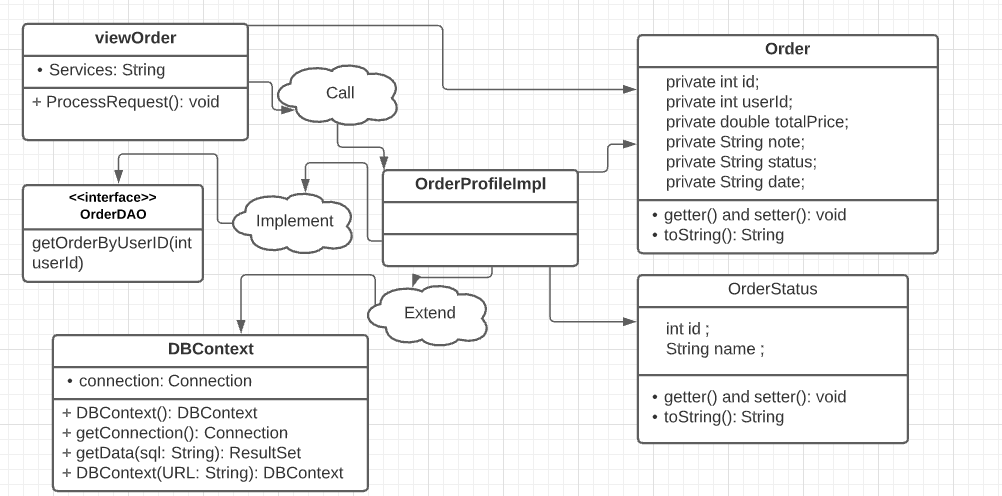
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | processRequest() | get all user information from the database |

### c. Sequence Diagram(s)



## 18. My Orders

### a. Class Diagram



### b. Class Specifications

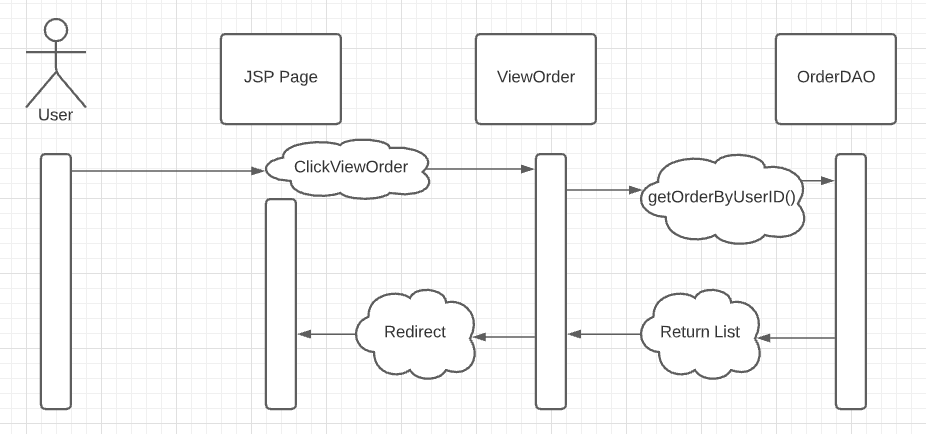
* viewOrderr:

|  |  |  |
| --- | --- | --- |
| 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)

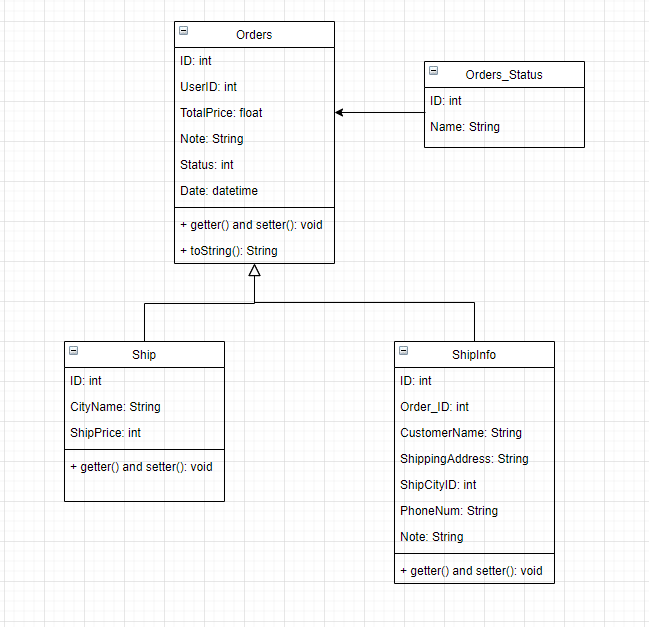


**d. Database queries**

* 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**

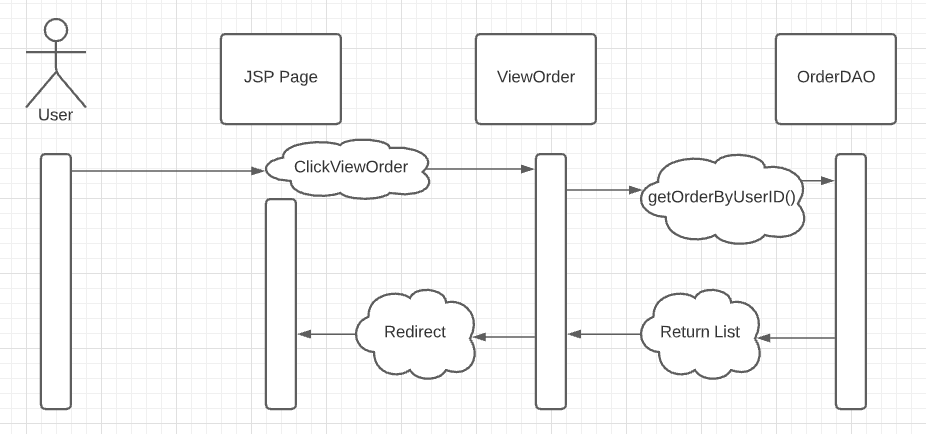
* viewOrderr:

|  |  |  |
| --- | --- | --- |
| 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**

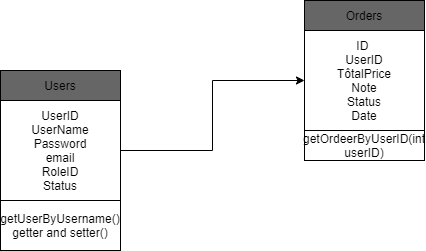
**b. Class Specifications**

**c. Sequence Diagram(s)**

**d. Database queries**

**21. Orders List**

**a. Class Diagram**

****

**b. Class Specifications**

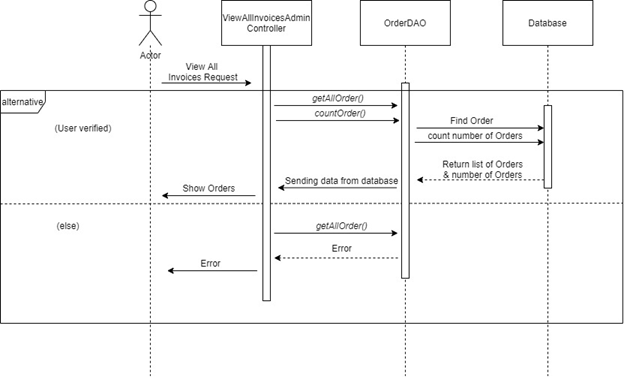
**Dashboard**

|  |  |  |
| --- | --- | --- |
| 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)**

****

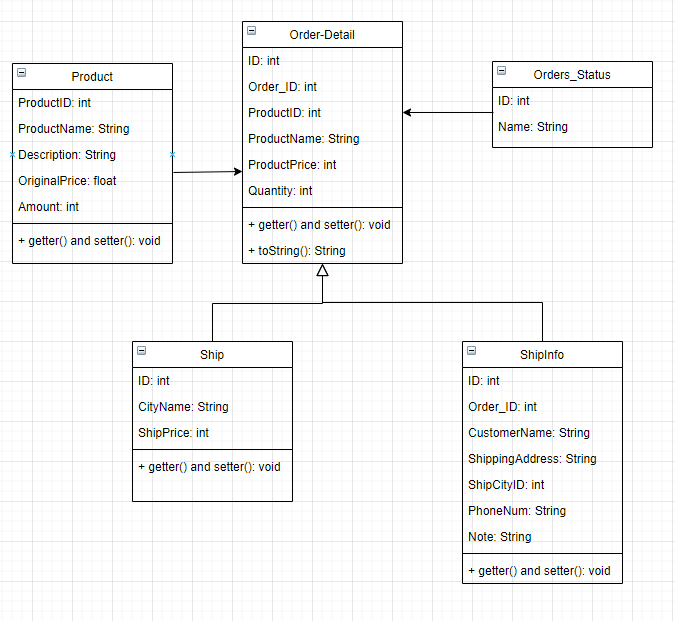
### d. Database queries

* 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

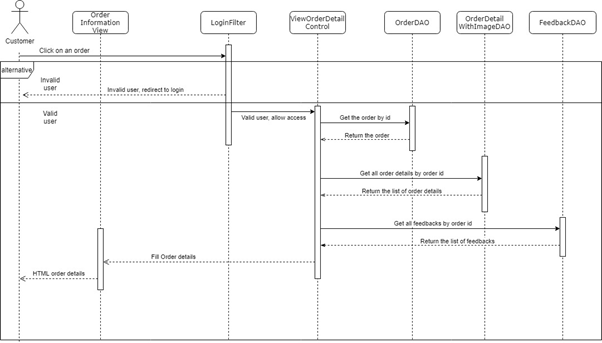
* viewOrder Details:

|  |  |  |
| --- | --- | --- |
| 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)

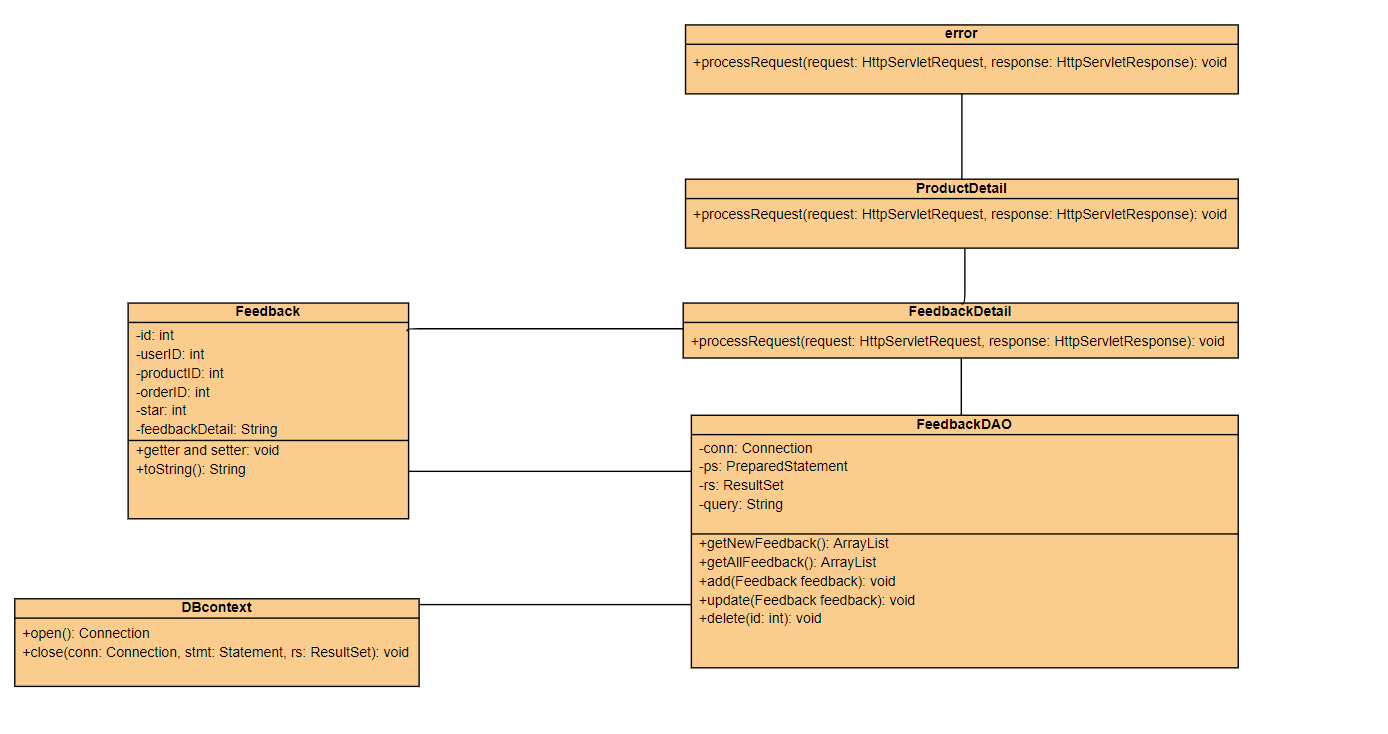
****

### 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

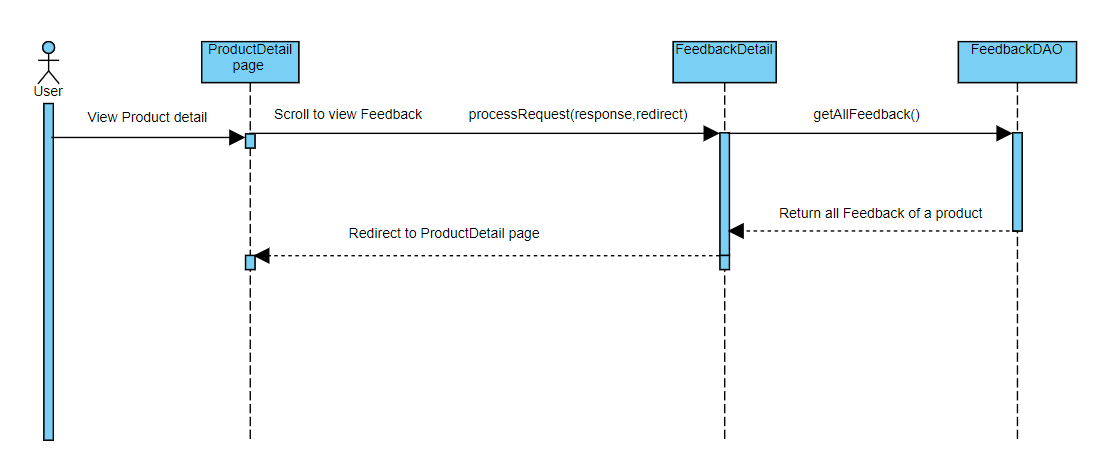
* FeedbackDetail class

|  |  |  |
| --- | --- | --- |
| **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)

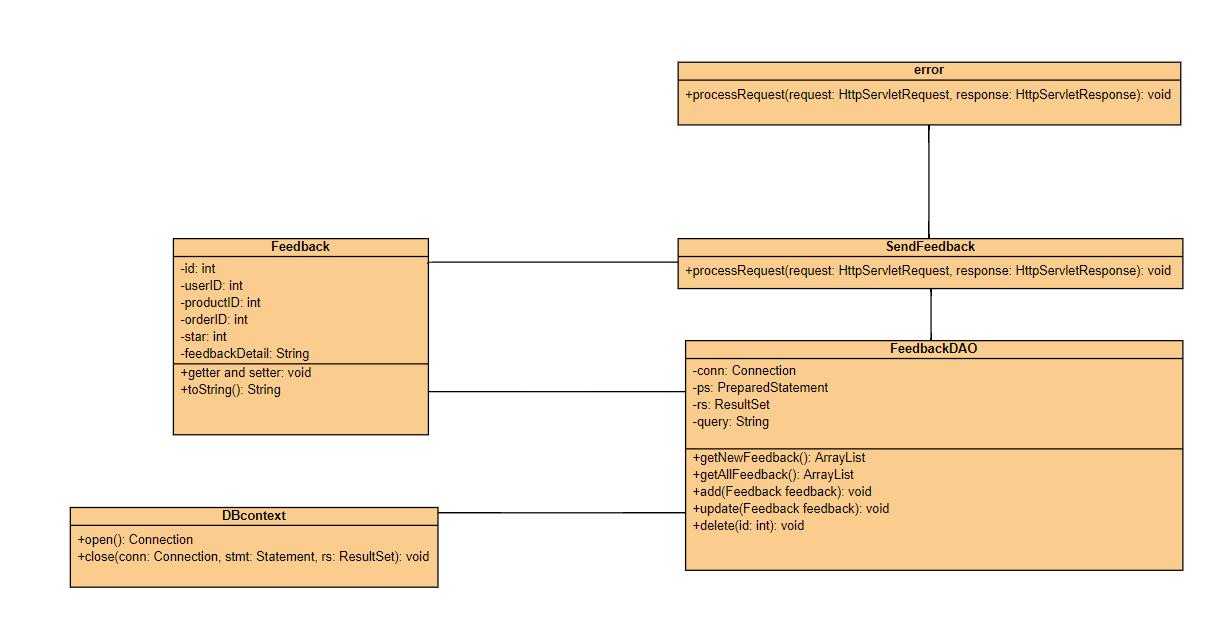


### d. Database queries

Select \* from Feedback where productID=?

## 24. Send Feedback

### a. Class Diagram



### b. Class Specifications

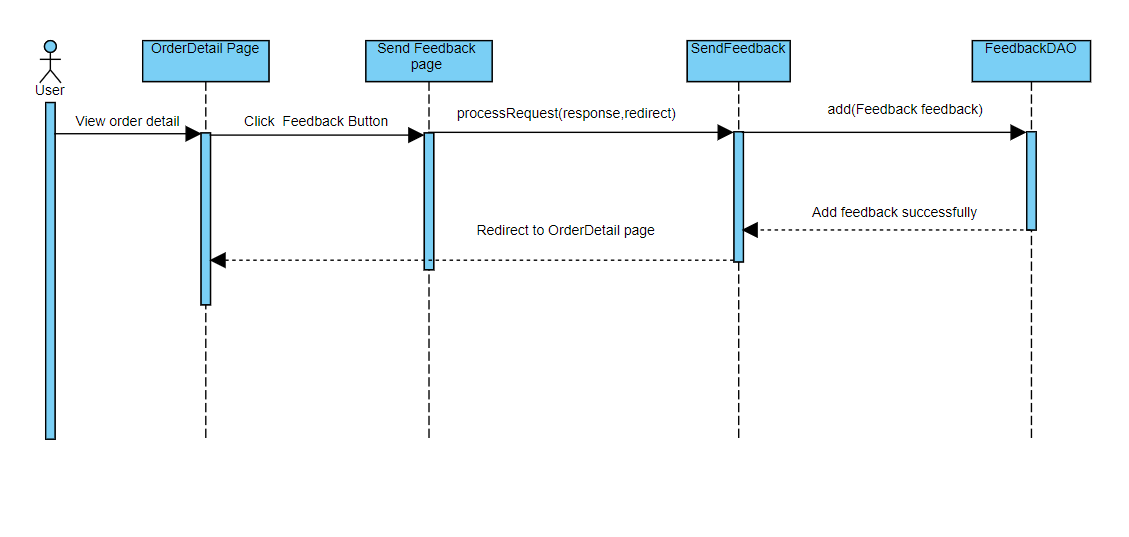
* SendFeedback class

|  |  |  |
| --- | --- | --- |
| **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)

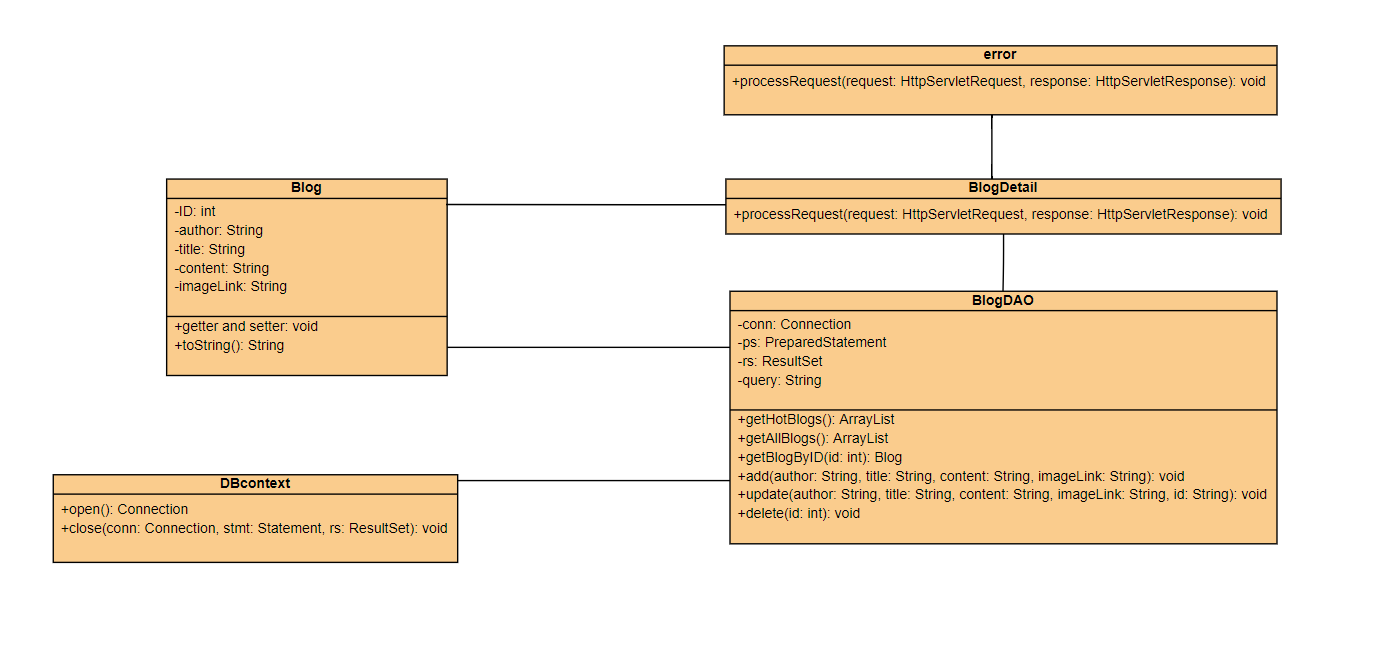


### d. Database queries

Select \* from Blog desc

## 26. Blog Details

### a. Class Diagram



### b. Class Specifications

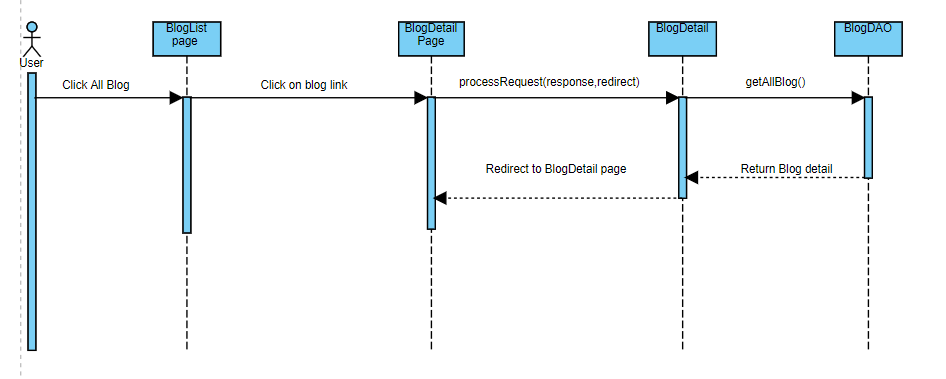
* BlogDetail class

|  |  |  |
| --- | --- | --- |
| **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)

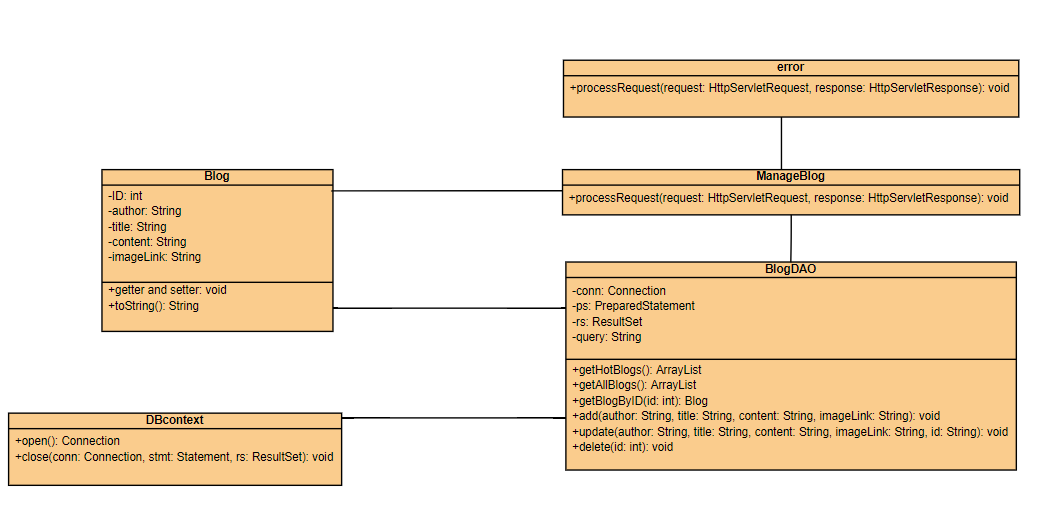


### d. Database queries

SELECT \* FROM Blog WHERE ID = ?

## 27. Blog Management

### a. Class Diagram



### b. Class Specifications

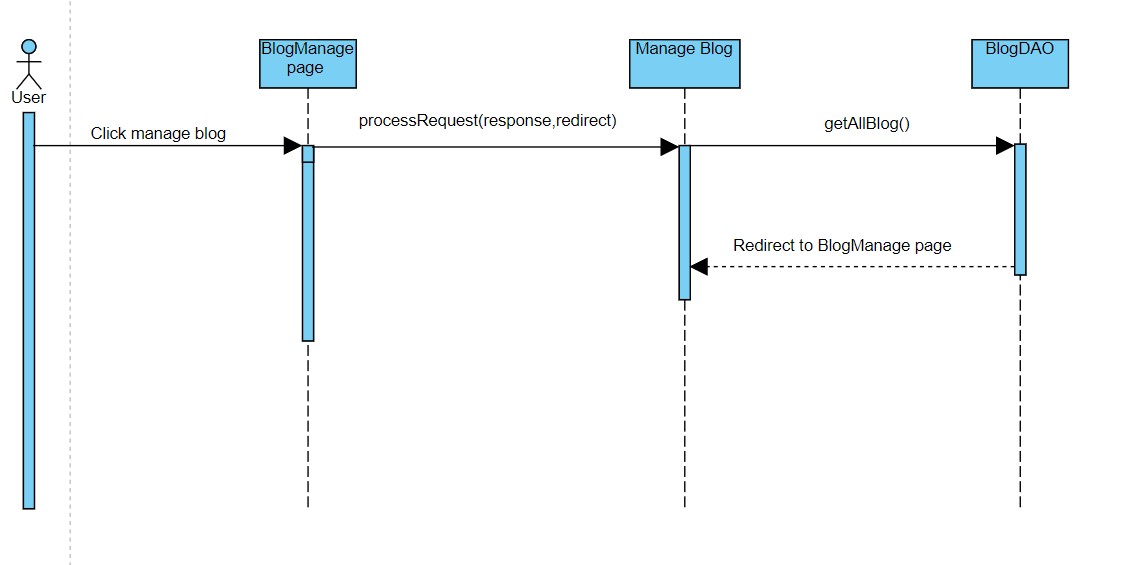
* ManageBlog class

|  |  |  |
| --- | --- | --- |
| **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)

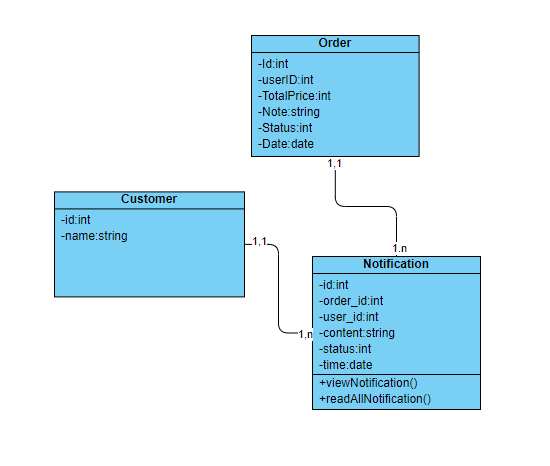


### d. Database queries

Select \* from blog

## 28. View Notification

### a. Class Diagram

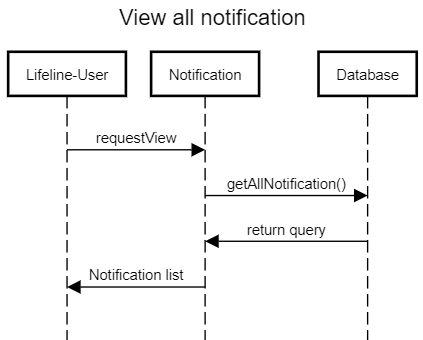


### b. Class Specifications

Notification class

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

### c. Sequence Diagram(s)

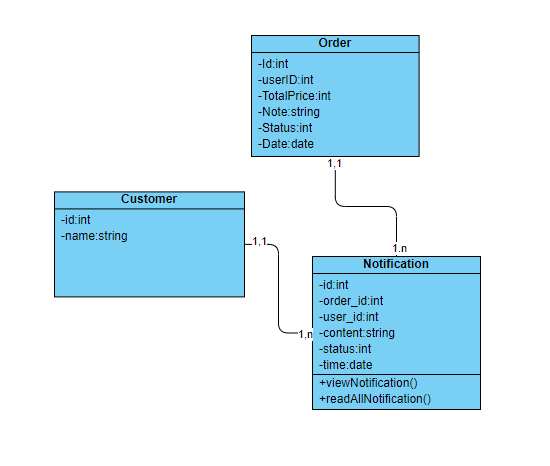


### d. Database queries

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

## 29. Mark as Read

### a. Class Diagram

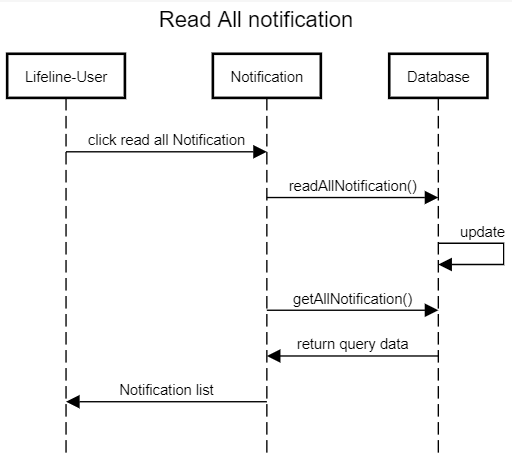


### b. Class Specifications

Notification class

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

### c. Sequence Diagram(s)



### d. Database queries

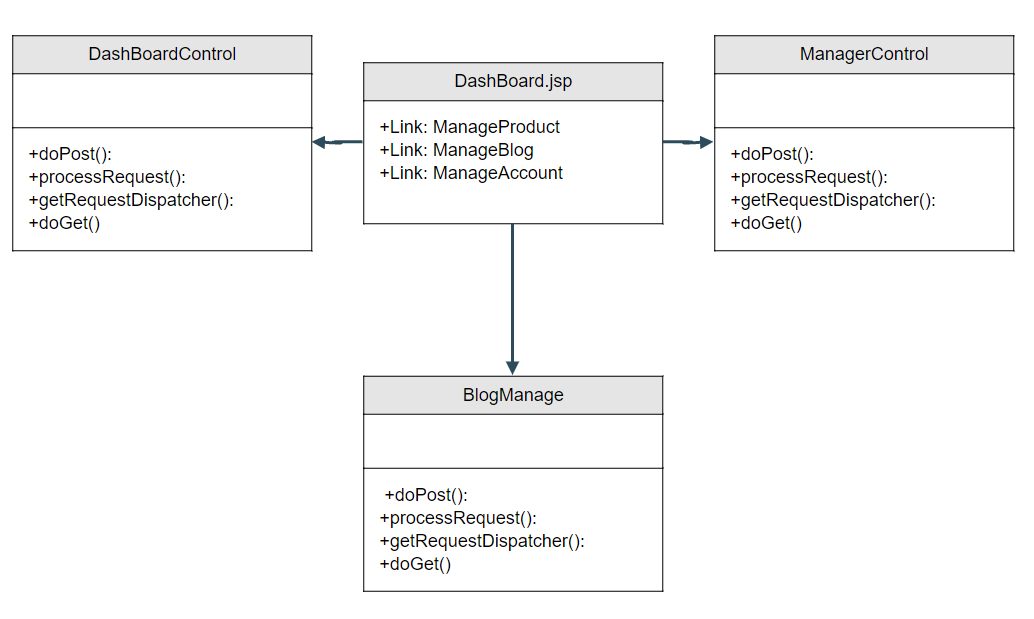
Update dbo.Notification

set status=1

where user\_id=?

## 30. Admin Feature

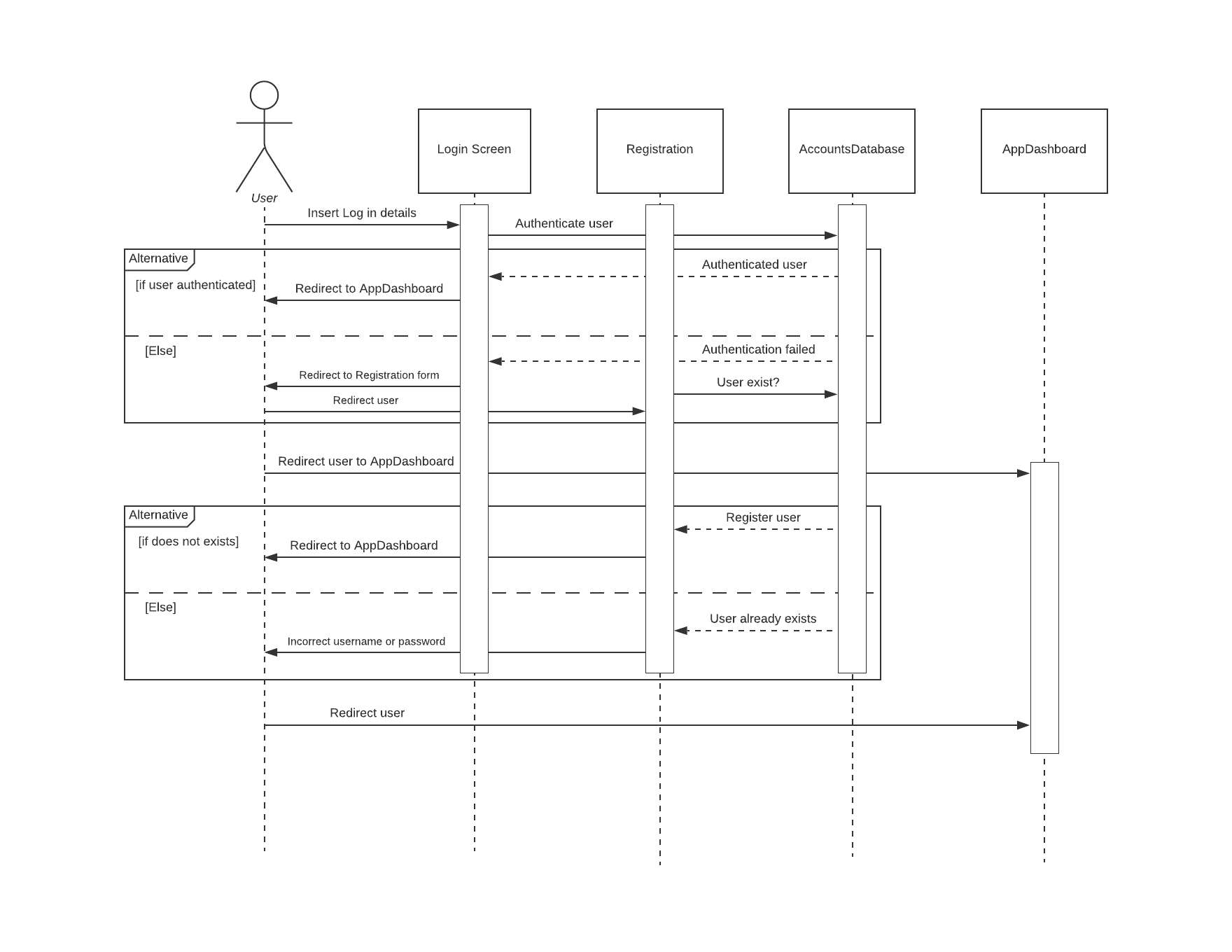
### a. Class Diagram



### 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)

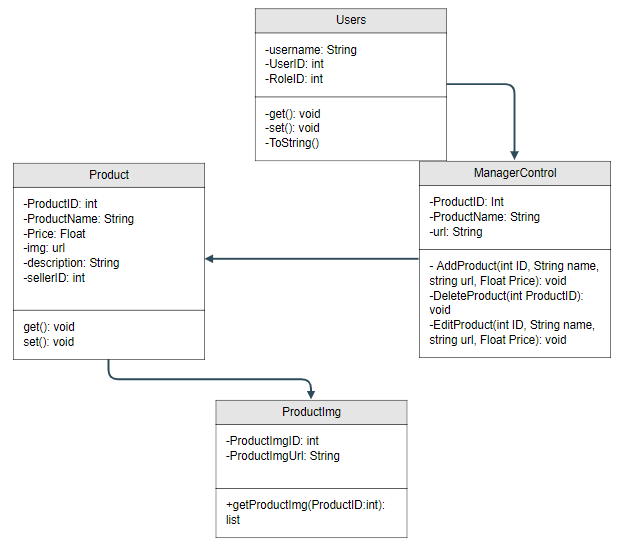


### d. Database queries

+select \* from …

## 31. Add Product

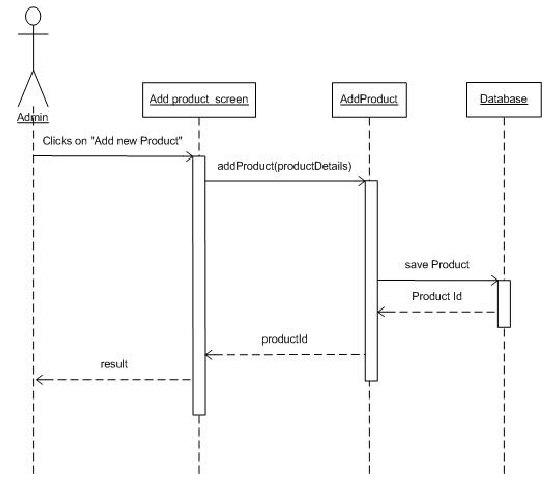
### a. Class Diagram



### b. Class Specifications

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

### c. Sequence Diagram(s)

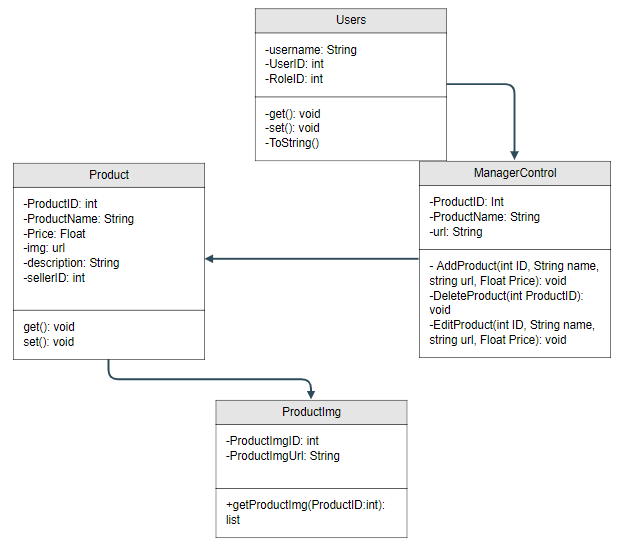


### d. Database queries

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

## 32. Delete Product

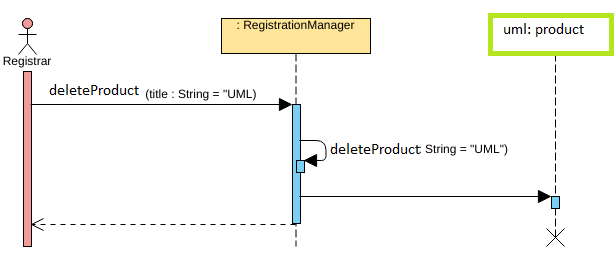
### a. Class Diagram



### b. Class Specifications

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

### c. Sequence Diagram(s)

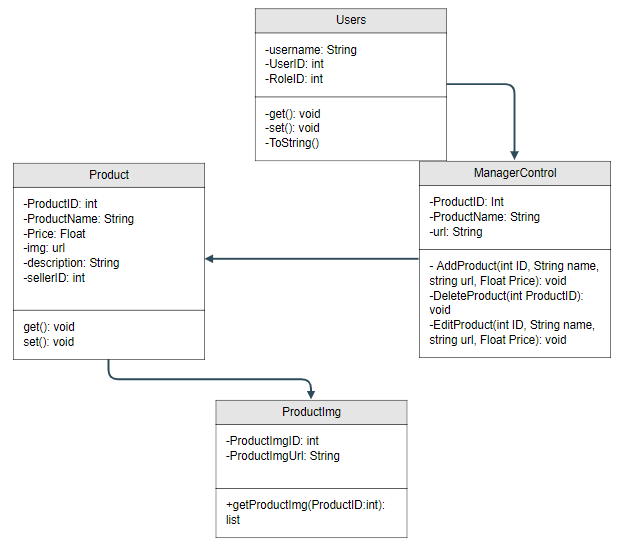


### d. Database queries

+delete from Product where ProductID = …

## 33. Edit Product

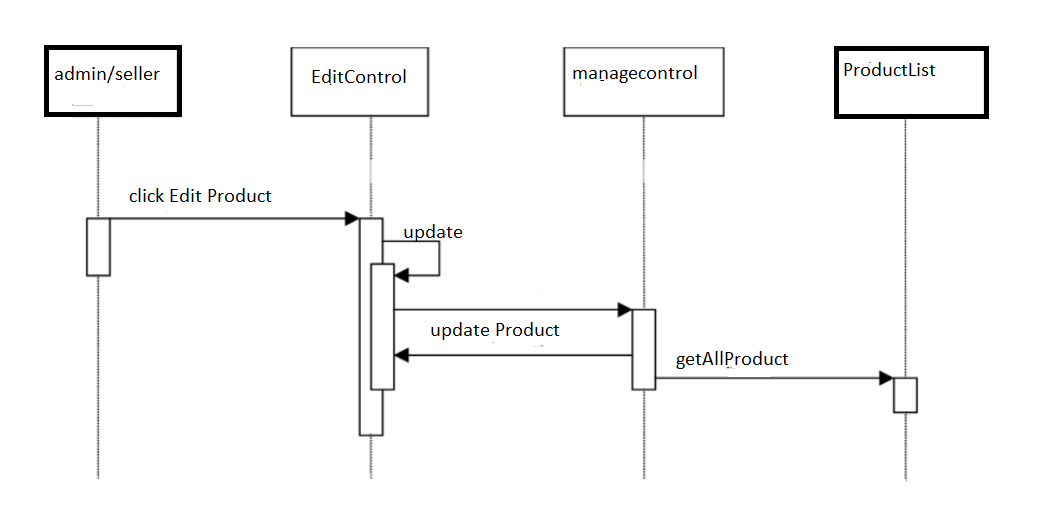
### a. Class Diagram



### b. Class Specifications

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

### c. Sequence Diagram(s)



### d. Database queries

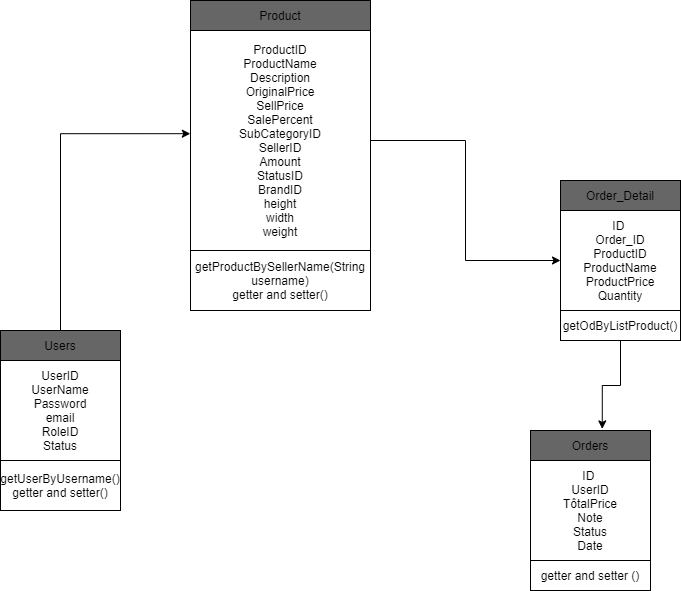
+UPDATE datatable

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* |

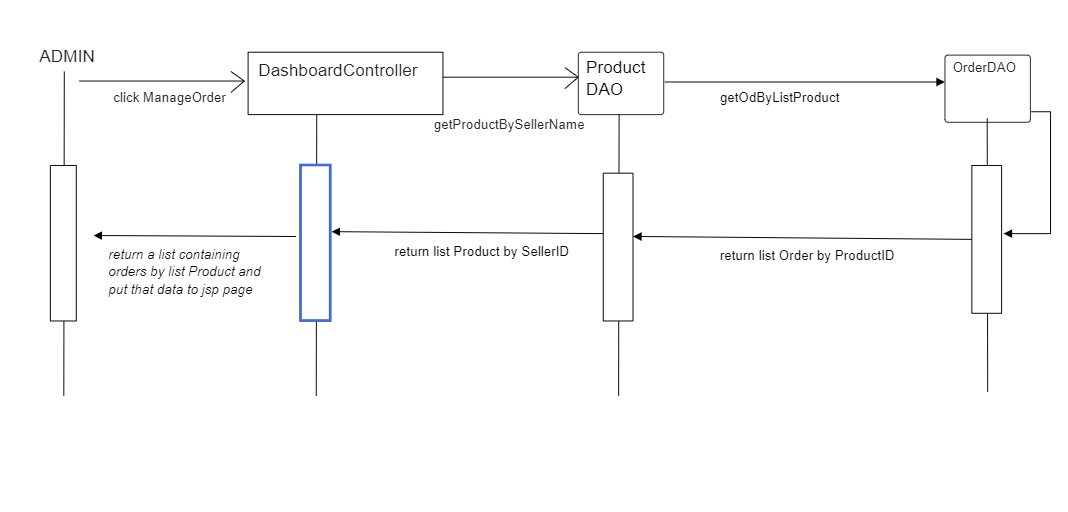
**Product DAO**

|  |  |  |
| --- | --- | --- |
| 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**

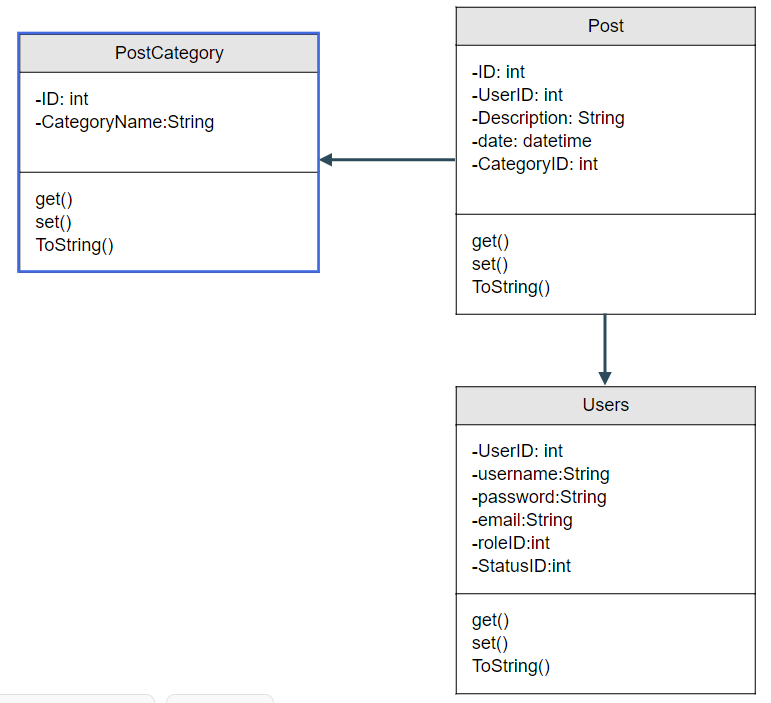
**on p.ProductID = s.ProductID where p.SellerID = (select UserID from Users where Username = ? )**

* **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

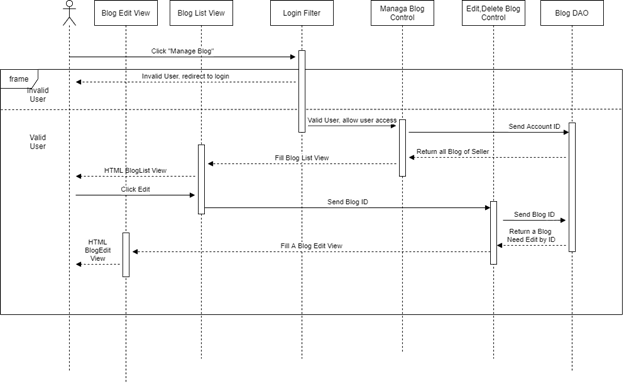
#### PostControl Class: The doGet function is used to get the ID of the post and then use the doPost function to edit the product

|  |  |  |
| --- | --- | --- |
| **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

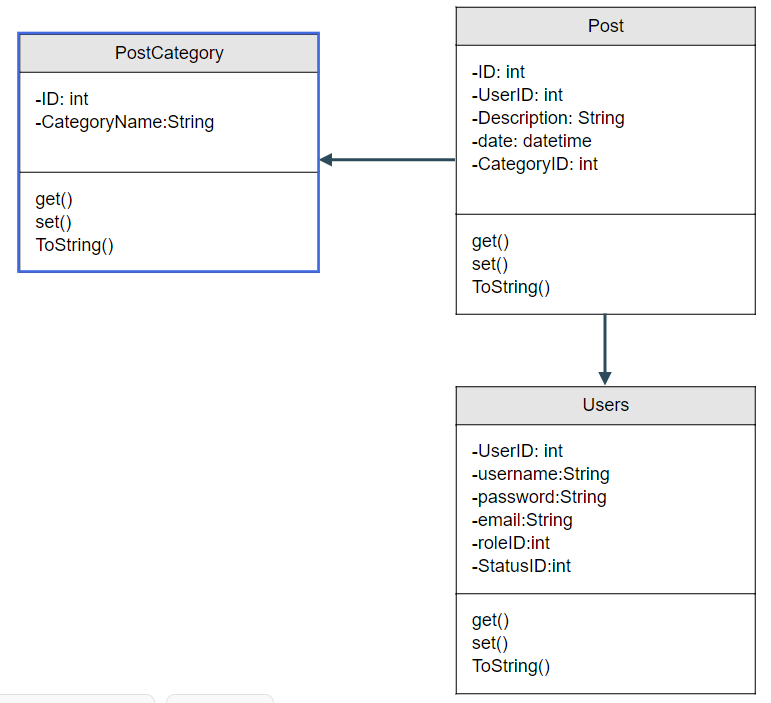
- "Delete FROM Post WHERE ID = ?"

- "Update …= ? “

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

**36. PostList**

**a.class diagram**

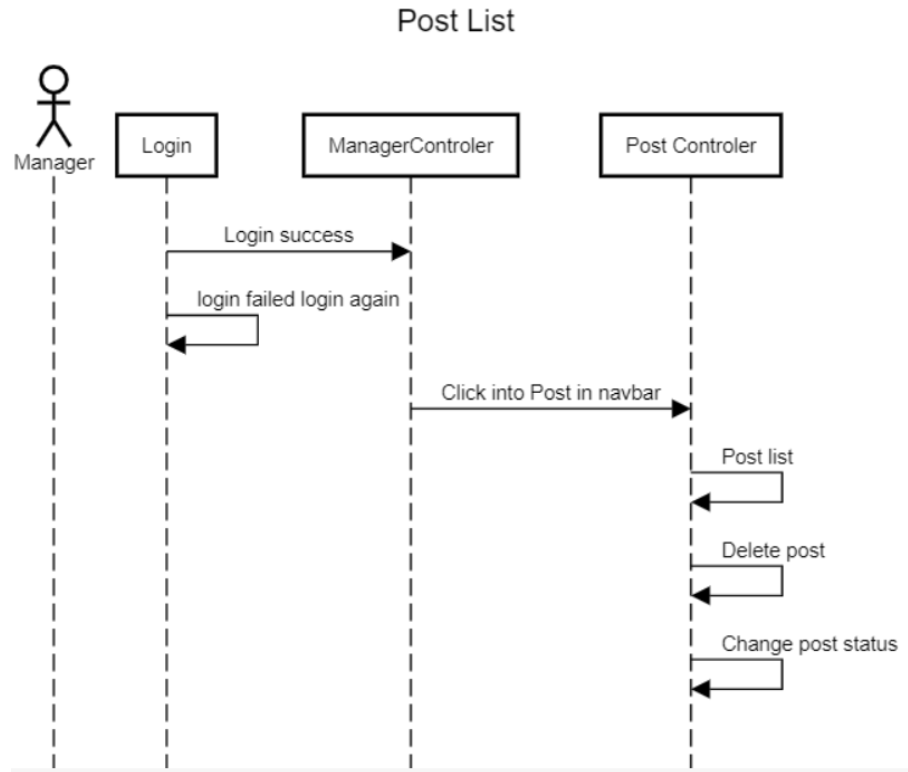
****

### 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**

****

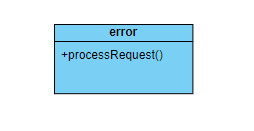
### d. Database queries

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

## 

## 37. Error

### a. Class Diagram

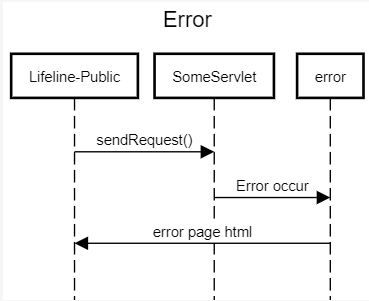
**

### b. Class Specifications

error Class

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

### c. Sequence Diagram(s)

**

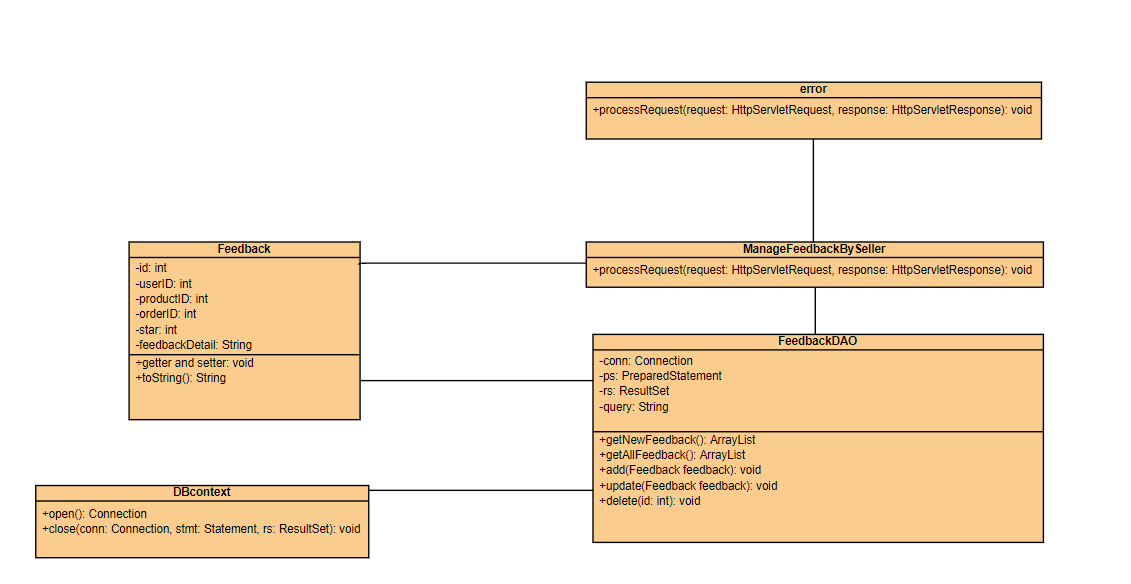
### d. Database queries

None

## 

## 38. Manage Feedback [Seller]

### a. Class Diagram



### b. Class Specifications

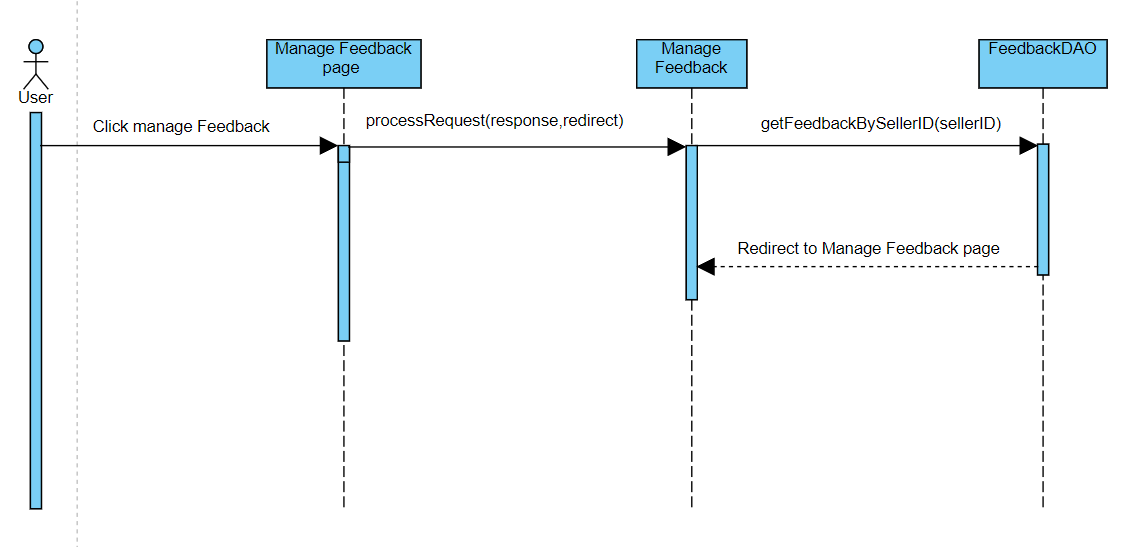
* ManageFeedbackBySeller class

|  |  |  |
| --- | --- | --- |
| **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)

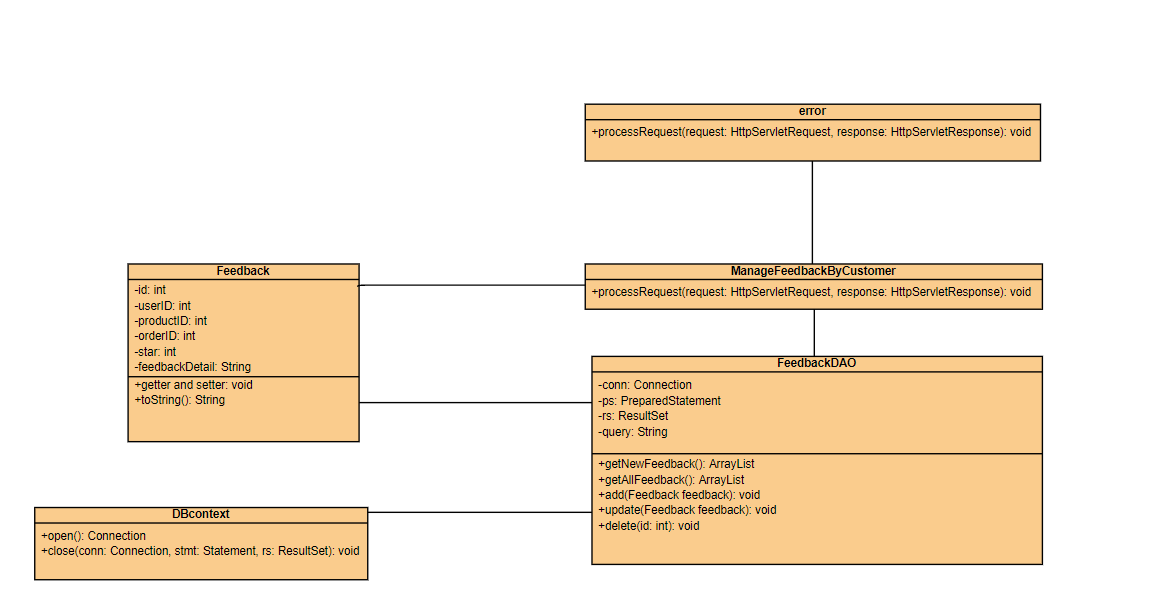


### d. Database queries

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

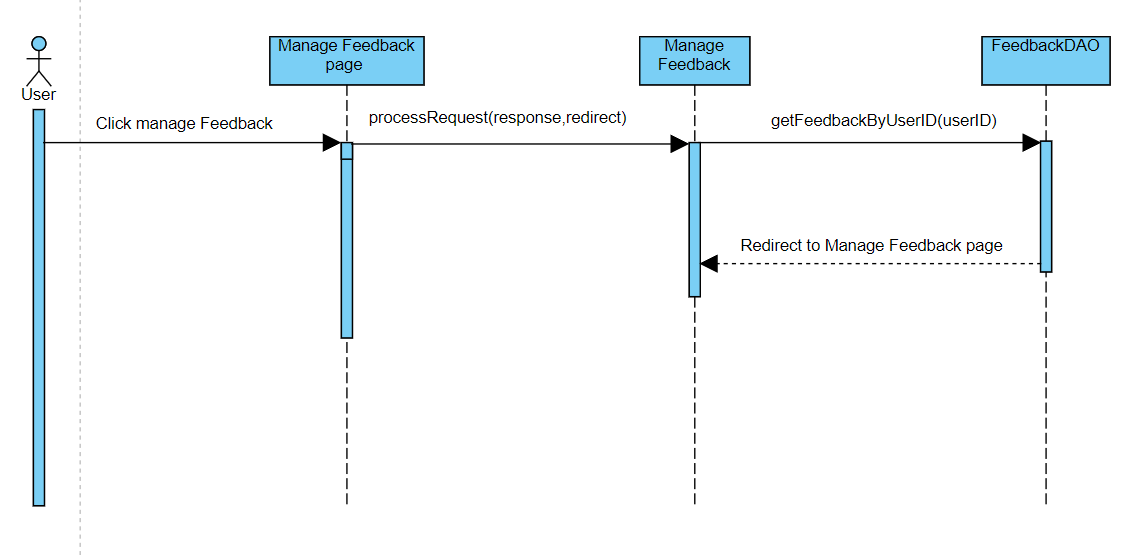
* ManageFeedbackBySeller class

|  |  |  |
| --- | --- | --- |
| **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)



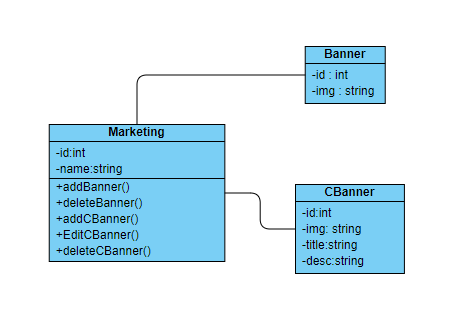
### d. Database queries

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

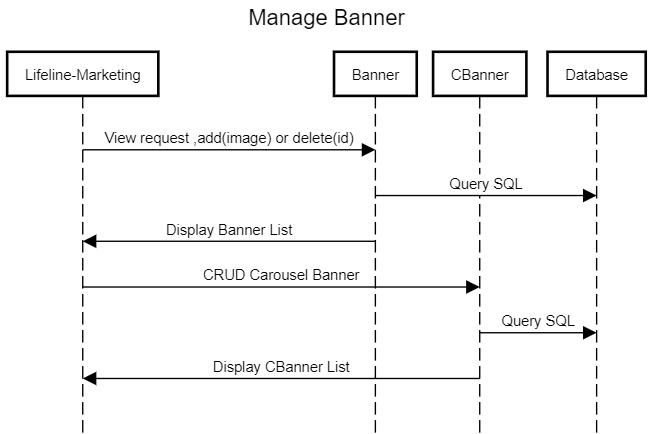
### a. Class Diagram

**

### 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

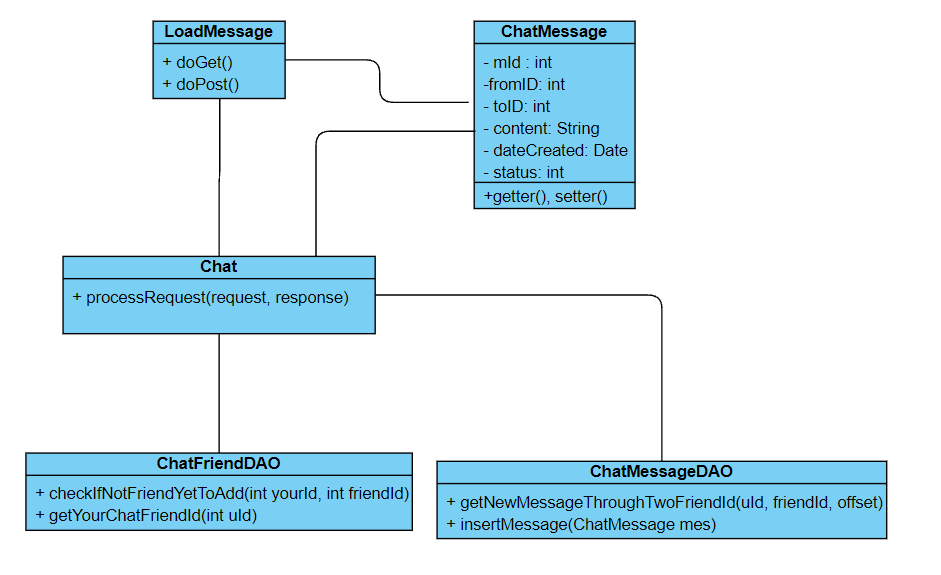
INSERT INTO dbo.CBanner VALUES ( ? , ? , ? )

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 |

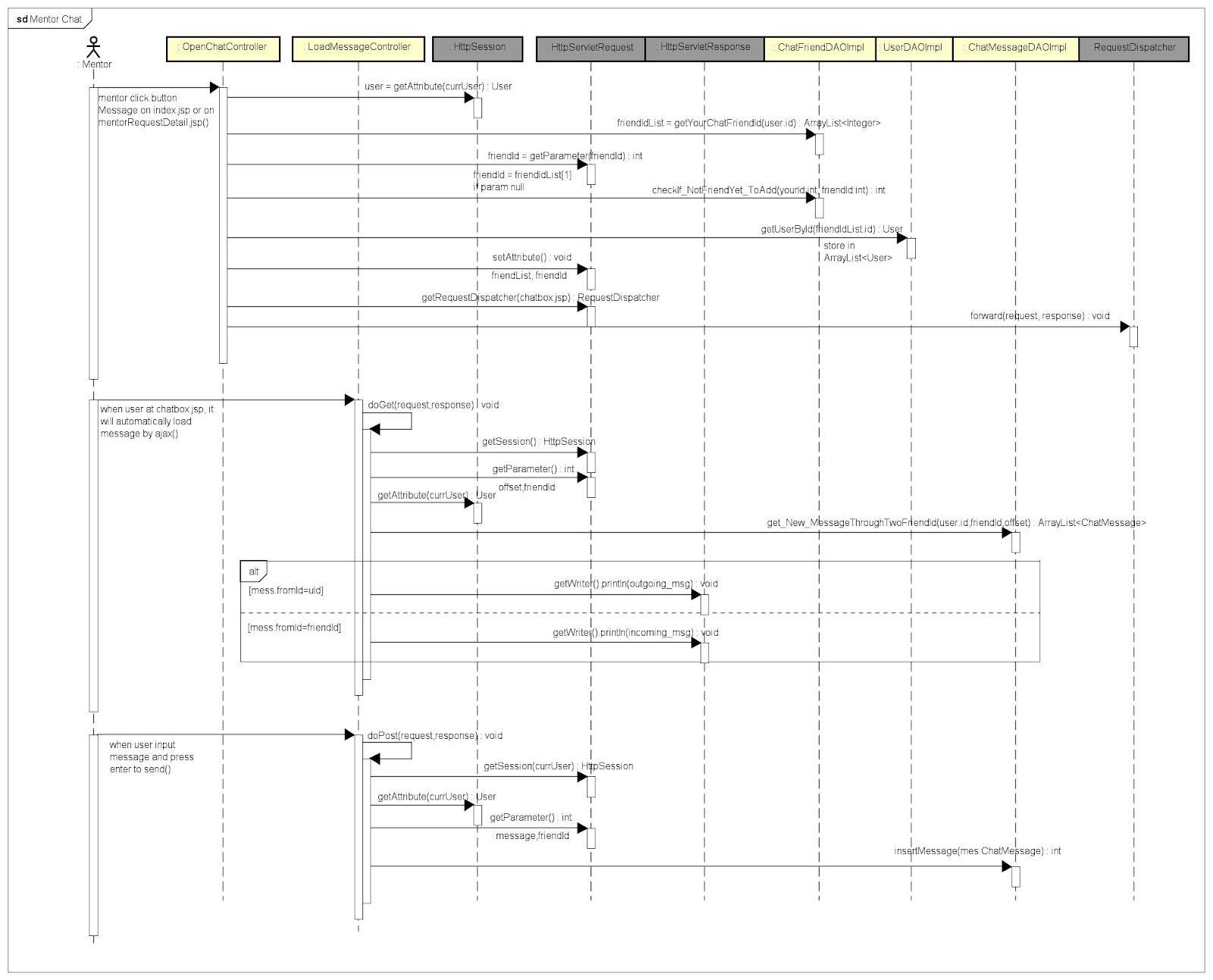
### 

ChatMessageDAO

### 

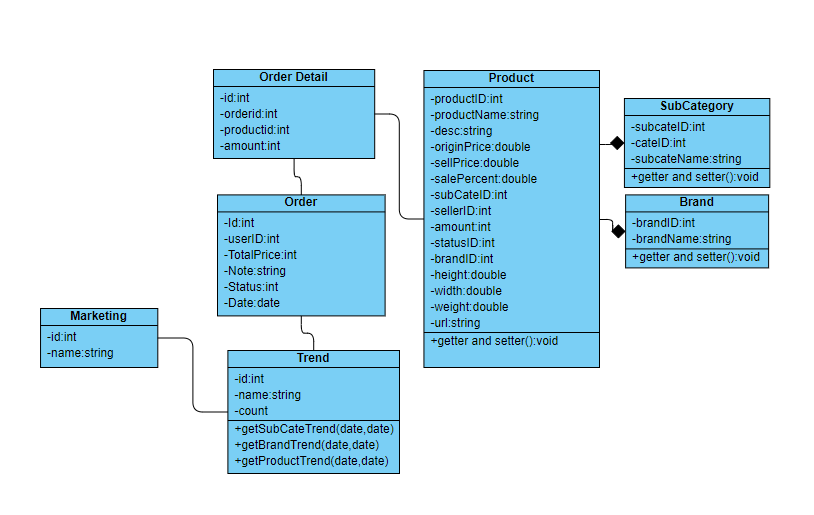
|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getNewMessageThroughTwoFriendId(uId, friendId, offset) | * This method will ignore some first messages and load new message between two friend * param: uId (your id) * friendId (friend id) * offset (the number of message loaded and not 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

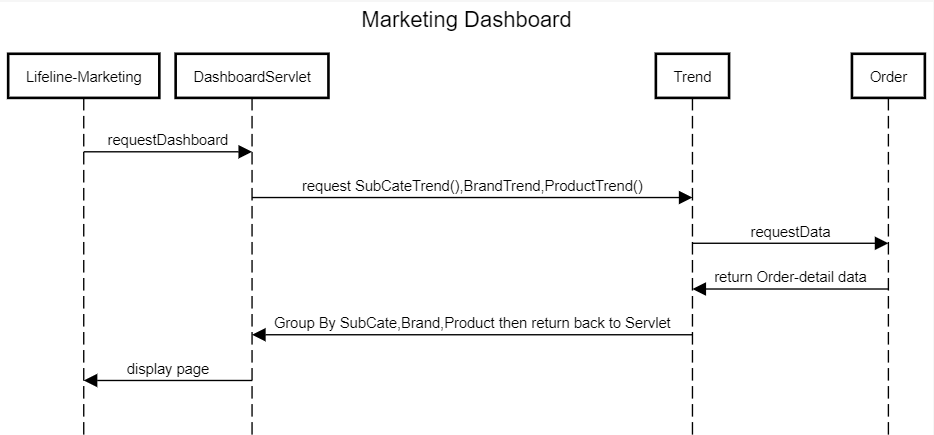
### a. Class Diagram

**

### 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

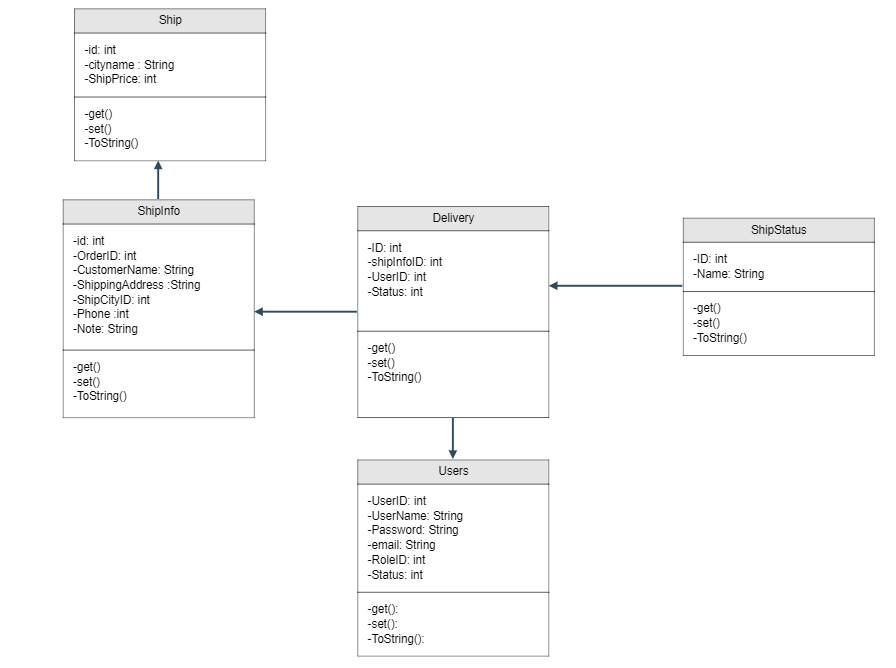
SELECT odr.SubCategoryID,SubCategoryName,SUM(odr.Quantity) AS Count FROM ( SELECT od.ProductID,od.ProductName,od.Quantity,od.Date,SubCategoryID,BrandID FROM (SELECT ProductID,ProductName,Quantity,Date FROM dbo.Orders JOIN dbo.Order\_Detail ON Order\_Detail.Order\_ID = Orders.ID ) od JOIN dbo.Product 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* |

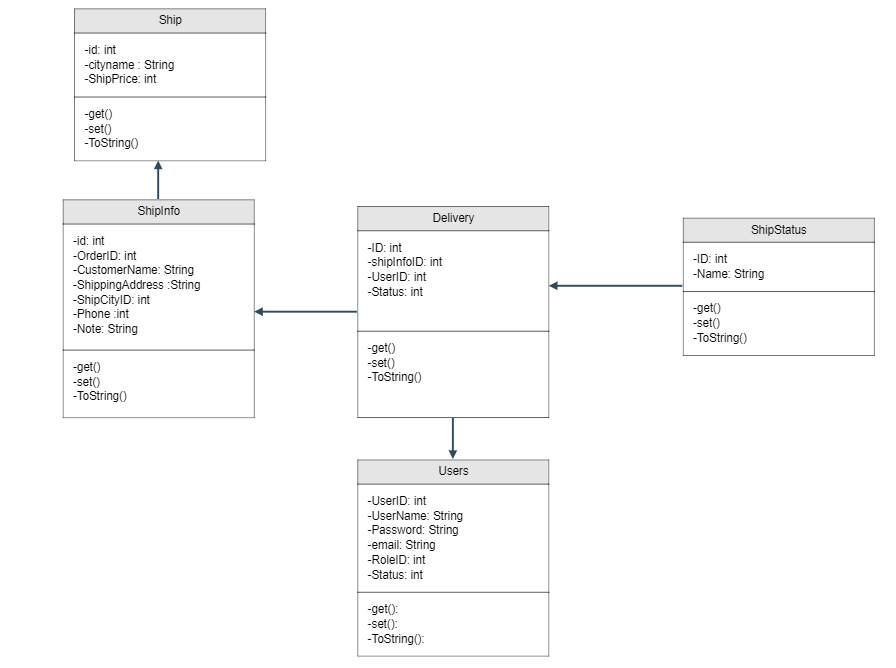
### c. Sequence Diagram(s)

### 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* |

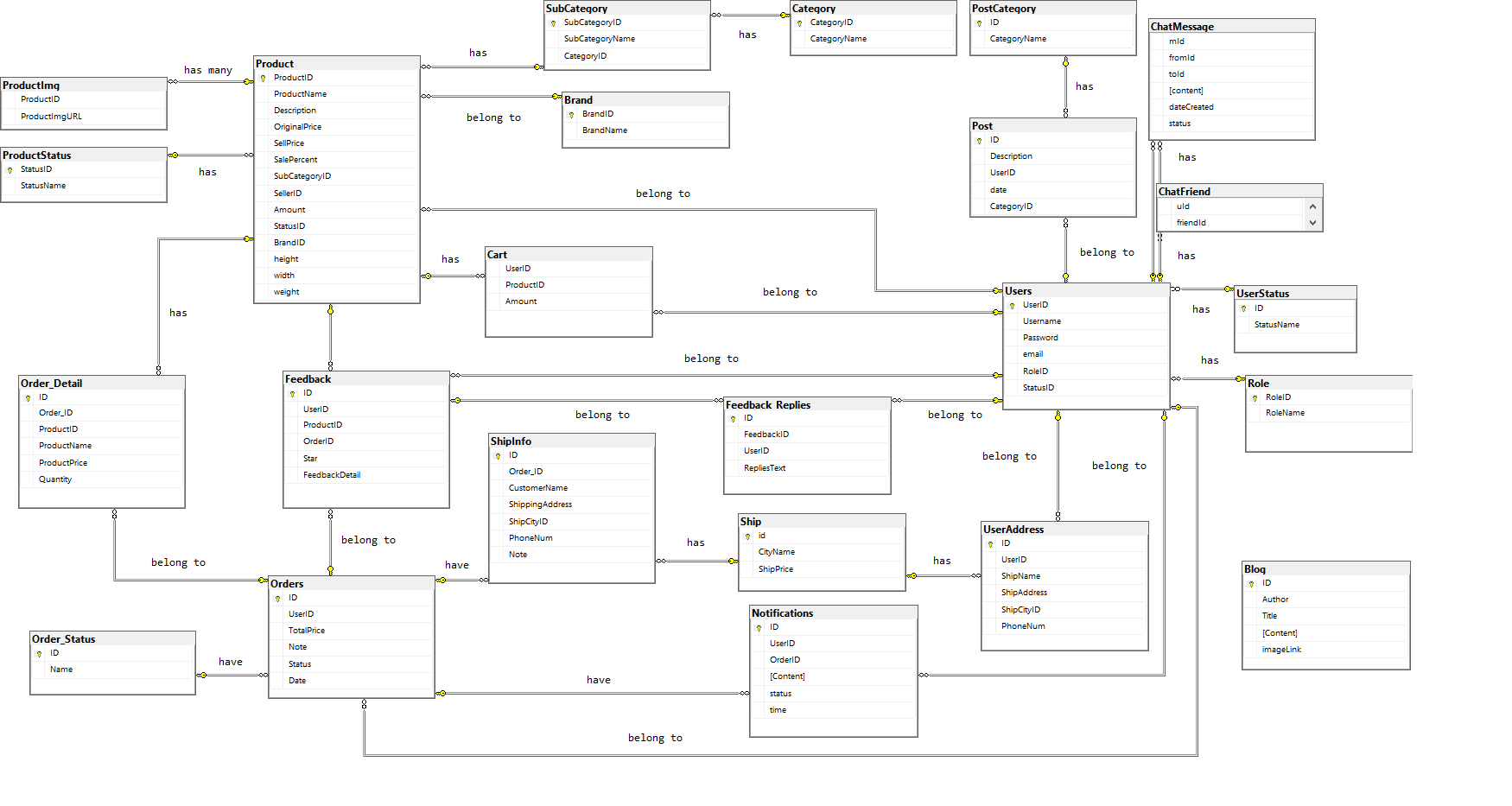
### c. Sequence Diagram(s)

### 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* |

## 