

**ONLINE SHOPPING SYSTEM**

**Software Design Document**

**(Version 0.7 – 07/11/2022)**

Member:

Trần Trung Việt – HE153481

Nguyễn Trung Hiếu – HE160326

Trần Công Minh – HE161354

Lê Sỹ Thái – HE160614

Vũ Hoàng Anh – HE163109

– Hanoi, November 2022 –

**Table of Contents**

[I. Overview 11](#_Toc118692346)

[1. Code Packages/Namespaces 11](#_Toc118692347)

[2. Coding Conventions 12](#_Toc118692348)

[II. Code Designs 12](#_Toc118692349)

[1. Login 12](#_Toc118692350)

[a. Class Diagram 12](#_Toc118692351)

[b. Class Specifications 12](#_Toc118692352)

[c. Sequence Diagram(s) 12](#_Toc118692353)

[d. Database queries 13](#_Toc118692354)

[2. Register 14](#_Toc118692355)

[a. Class Diagram 14](#_Toc118692356)

[b. Class Specifications 14](#_Toc118692357)

[c. Sequence Diagram(s) 15](#_Toc118692358)

[d. Database queries 15](#_Toc118692359)

[3. Register for seller 16](#_Toc118692360)

[a. Class Diagram 16](#_Toc118692361)

[b. Class Specifications 16](#_Toc118692362)

[c. Sequence Diagram(s) 16](#_Toc118692363)

[d. Database queries 17](#_Toc118692364)

[4. Feature Product 18](#_Toc118692365)

[a. Class Diagram 18](#_Toc118692366)

[b. Class Specification 18](#_Toc118692367)

[c. Sequence Diagram(s) 19](#_Toc118692368)

[d. Database queries 19](#_Toc118692369)

[5. Search and Fillter Product 20](#_Toc118692370)

[a. Class Diagram 20](#_Toc118692371)

[b. Class Specification 20](#_Toc118692372)

[c. Sequence Diagram(s) 21](#_Toc118692373)

[d. Database queries 21](#_Toc118692374)

[6. View Product Detail 22](#_Toc118692375)

[a. Class Diragram 22](#_Toc118692376)

[b. Class Specification 22](#_Toc118692377)

[c. Sequence Diagram(s) 23](#_Toc118692378)

[d. Database queries 23](#_Toc118692379)

[7. Change Pass 23](#_Toc118692380)

[a. Class Diragram 23](#_Toc118692381)

[b. Class Specification 24](#_Toc118692382)

[c. Sequence Diagram(s) 25](#_Toc118692383)

[d. Database queries 25](#_Toc118692384)

[8. Forgot Password 25](#_Toc118692385)

[a. Class Diragram 25](#_Toc118692386)

[b. Class Specification 26](#_Toc118692387)

[c. Sequence Diagram(s) 26](#_Toc118692388)

[d. Database queries 26](#_Toc118692389)

[9. User Profile 26](#_Toc118692390)

[a. Class Diragram 26](#_Toc118692391)

[b. Class Specification 27](#_Toc118692392)

[c. Sequence Diagram(s) 28](#_Toc118692393)

[d. Database queries 28](#_Toc118692394)

[10. Update Profile 28](#_Toc118692395)

[a. Class Diragram 28](#_Toc118692396)

[b. Class Specification 29](#_Toc118692397)

[c. Sequence Diagram(s) 29](#_Toc118692398)

[d. Database queries 29](#_Toc118692399)

[11. Add Address 29](#_Toc118692400)

[a. Class Diragram 29](#_Toc118692401)

[b. Class Specification 30](#_Toc118692402)

[c. Sequence Diagram(s) 31](#_Toc118692403)

[d. Database queries 31](#_Toc118692404)

[12. Edit Address 31](#_Toc118692405)

[a. Class Diragram 31](#_Toc118692406)

[b. Class Specification 32](#_Toc118692407)

[c. Sequence Diagram(s) 32](#_Toc118692408)

[d. Database queries 32](#_Toc118692409)

[13. Delete Address 33](#_Toc118692410)

[a. Class Diragram 33](#_Toc118692411)

[b. Class Specification 33](#_Toc118692412)

[c. Sequence Diagram(s) 34](#_Toc118692413)

[d. Database queries 34](#_Toc118692414)

[14. Show Address 34](#_Toc118692415)

[a. Class Diragram 34](#_Toc118692416)

[b. Class Specification 35](#_Toc118692417)

[c. Sequence Diagram(s) 35](#_Toc118692418)

[d. Database queries 35](#_Toc118692419)

[15. Add to Cart 35](#_Toc118692420)

[a. Class Diragram 35](#_Toc118692421)

[b. Class Specification 36](#_Toc118692422)

[c. Sequence Diagram(s) 37](#_Toc118692423)

[d. Database queries 37](#_Toc118692424)

[16. View Cart 37](#_Toc118692425)

[a. Class Diragram 37](#_Toc118692426)

[b. Class Specification 38](#_Toc118692427)

[c. Sequence Diagram(s) 39](#_Toc118692428)

[d. Database queries 39](#_Toc118692429)

[17. Delete Product in Cart 39](#_Toc118692430)

[a. Class Diragram 39](#_Toc118692431)

[b. Class Specification 40](#_Toc118692432)

[c. Sequence Diagram(s) 40](#_Toc118692433)

[d. Database queries 40](#_Toc118692434)

[18. Create Order 40](#_Toc118692435)

[a. Class Diragram 40](#_Toc118692436)

[b. Class Specification 41](#_Toc118692437)

[c. Sequence Diagram(s) 42](#_Toc118692438)

[d. Database queries 42](#_Toc118692439)

[19. View Order 42](#_Toc118692440)

[a. Class Diragram 42](#_Toc118692441)

[b. Class Specification 43](#_Toc118692442)

[c. Sequence Diagram(s) 44](#_Toc118692443)

[d. Database queries 44](#_Toc118692444)

[20. Cancel Order 44](#_Toc118692445)

[a. Class Diragram 44](#_Toc118692446)

[b. Class Specification 45](#_Toc118692447)

[c. Sequence Diagram(s) 46](#_Toc118692448)

[d. Database queries 46](#_Toc118692449)

[21. Order Detail 46](#_Toc118692450)

[a. Class Diragram 46](#_Toc118692451)

[b. Class Specification 47](#_Toc118692452)

[c. Sequence Diagram(s) 48](#_Toc118692453)

[d. Database queries 48](#_Toc118692454)

[22. Add product to Wishlist 48](#_Toc118692455)

[a. Class Diragram 48](#_Toc118692456)

[b. Class Specification 49](#_Toc118692457)

[c. Sequence Diagram(s) 50](#_Toc118692458)

[d. Database queries 50](#_Toc118692459)

[23. Delete product in Wishlist 50](#_Toc118692460)

[a. Class Diragram 50](#_Toc118692461)

[b. Class Specification 51](#_Toc118692462)

[c. Sequence Diagram(s) 51](#_Toc118692463)

[d. Database queries 51](#_Toc118692464)

[24. Blog List 52](#_Toc118692465)

[a. Class Diagram 52](#_Toc118692466)

[b. Class Specification 52](#_Toc118692467)

[c. Sequence Diagram(s) 53](#_Toc118692468)

[d. Database queries 53](#_Toc118692469)

[25. Blog Detail 53](#_Toc118692470)

[a. Class Diagram 53](#_Toc118692471)

[b. Class Specification 54](#_Toc118692472)

[c. Sequence Diagram(s) 54](#_Toc118692473)

[d. Database queries 54](#_Toc118692474)

[26. Search Blog 55](#_Toc118692475)

[a. Class Diagram 55](#_Toc118692476)

[b. Class Specification 55](#_Toc118692477)

[c. Sequence Diagram(s) 56](#_Toc118692478)

[d. Database queries 56](#_Toc118692479)

[27. Comment Blog 56](#_Toc118692480)

[a. Class Diagram 56](#_Toc118692481)

[b. Class Specification 57](#_Toc118692482)

[c. Sequence Diagram(s) 57](#_Toc118692483)

[d. Database queries 57](#_Toc118692484)

[28. View Comment 58](#_Toc118692485)

[a. Class Diagram 58](#_Toc118692486)

[b. Class Specification 58](#_Toc118692487)

[c. Sequence Diagram(s) 59](#_Toc118692488)

[d. Database queries 59](#_Toc118692489)

[29. Delete Comment 59](#_Toc118692490)

[a. Class Diagram 59](#_Toc118692491)

[b. Class Specification 60](#_Toc118692492)

[c. Sequence Diagram(s) 60](#_Toc118692493)

[d. Database queries 60](#_Toc118692494)

[30. View Service 61](#_Toc118692495)

[a. Class Diagram 61](#_Toc118692496)

[b. Class Specification 61](#_Toc118692497)

[c. Sequence Diagram(s) 62](#_Toc118692498)

[d. Database queries 62](#_Toc118692499)

[31. Search Service 63](#_Toc118692500)

[a. Class Diagram 63](#_Toc118692501)

[b. Class Specification 63](#_Toc118692502)

[c. Sequence Diagram(s) 64](#_Toc118692503)

[d. Database queries 64](#_Toc118692504)

[32. View Feedback 64](#_Toc118692505)

[a. Class Diagram 64](#_Toc118692506)

[b. Class Specification 65](#_Toc118692507)

[c. Sequence Diagram(s) 65](#_Toc118692508)

[d. Database queries 65](#_Toc118692509)

[33. Add Feedback 66](#_Toc118692510)

[a. Class Diagram 66](#_Toc118692511)

[b. Class Specification 66](#_Toc118692512)

[c. Sequence Diagram(s) 67](#_Toc118692513)

[d. Database queries 67](#_Toc118692514)

[34. View Shop 67](#_Toc118692515)

[a. Class Diagram 67](#_Toc118692516)

[b. Class Specification 68](#_Toc118692517)

[c. Sequence Diagram(s) 68](#_Toc118692518)

[d. Database queries 68](#_Toc118692519)

[35. Report Shop 69](#_Toc118692520)

[a. Class Diagram 69](#_Toc118692521)

[b. Class Specification 69](#_Toc118692522)

[c. Sequence Diagram(s) 70](#_Toc118692523)

[d. Database queries 70](#_Toc118692524)

[36. Chat with seller 71](#_Toc118692525)

[a. Class Diagram 71](#_Toc118692526)

[b. Class Specification 71](#_Toc118692527)

[c. Sequence Diagram(s) 72](#_Toc118692528)

[d. Database queries 72](#_Toc118692529)

[37. Admin dashboard 72](#_Toc118692530)

[a. Class Diagram 72](#_Toc118692531)

[b. Class Specification 73](#_Toc118692532)

[c. Sequence Diagram(s) 73](#_Toc118692533)

[d. Database queries 73](#_Toc118692534)

[38. List of Account 74](#_Toc118692535)

[a. Class Diagram 74](#_Toc118692536)

[b. Class Specification 74](#_Toc118692537)

[c. Sequence Diagram(s) 75](#_Toc118692538)

[d. Database queries 75](#_Toc118692539)

[39. Account detail 75](#_Toc118692540)

[a. Class Diagram 75](#_Toc118692541)

[b. Class Specification 76](#_Toc118692542)

[c. Sequence Diagram(s) 76](#_Toc118692543)

[d. Database queries 76](#_Toc118692544)

[40. Search Account 77](#_Toc118692545)

[a. Class Diagram 77](#_Toc118692546)

[b. Class Specification 77](#_Toc118692547)

[c. Sequence Diagram(s) 78](#_Toc118692548)

[d. Database queries 78](#_Toc118692549)

[41. List of Shop 78](#_Toc118692550)

[a. Class Diagram 78](#_Toc118692551)

[b. Class Specification 79](#_Toc118692552)

[c. Sequence Diagram(s) 79](#_Toc118692553)

[d. Database queries 79](#_Toc118692554)

[42. Shop detail 80](#_Toc118692555)

[a. Class Diagram 80](#_Toc118692556)

[b. Class Specification 80](#_Toc118692557)

[c. Sequence Diagram(s) 81](#_Toc118692558)

[d. Database queries 81](#_Toc118692559)

[43. Search Shop 81](#_Toc118692560)

[a. Class Diagram 81](#_Toc118692561)

[b. Class Specification 82](#_Toc118692562)

[c. Sequence Diagram(s) 82](#_Toc118692563)

[d. Database queries 82](#_Toc118692564)

[44. List of Report 83](#_Toc118692565)

[a. Class Diagram 83](#_Toc118692566)

[b. Class Specification 83](#_Toc118692567)

[c. Sequence Diagram(s) 84](#_Toc118692568)

[d. Database queries 84](#_Toc118692569)

[45. Seller dashboard 84](#_Toc118692570)

[a. Class Diagram 84](#_Toc118692571)

[b. Class Specification 85](#_Toc118692572)

[c. Sequence Diagram(s) 85](#_Toc118692573)

[d. Database queries 86](#_Toc118692574)

[46. View all order 86](#_Toc118692575)

[a. Class Diagram 86](#_Toc118692576)

[b. Class Specification 86](#_Toc118692577)

[c. Sequence Diagram(s) 87](#_Toc118692578)

[d. Database queries 87](#_Toc118692579)

[47. Update status order 87](#_Toc118692580)

[a. Class Diagram 87](#_Toc118692581)

[b. Class Specification 88](#_Toc118692582)

[c. Sequence Diagram(s) 88](#_Toc118692583)

[d. Database queries 88](#_Toc118692584)

[48. Delete order 89](#_Toc118692585)

[a. Class Diagram 89](#_Toc118692586)

[b. Class Specification 89](#_Toc118692587)

[c. Sequence Diagram(s) 90](#_Toc118692588)

[d. Database queries 90](#_Toc118692589)

[49. Add product 90](#_Toc118692590)

[a. Class Diagram 90](#_Toc118692591)

[b. Class Specification 91](#_Toc118692592)

[c. Sequence Diagram(s) 91](#_Toc118692593)

[d. Database queries 91](#_Toc118692594)

[50. Update Product 92](#_Toc118692595)

[a. Class Diagram 92](#_Toc118692596)

[b. Class Specification 92](#_Toc118692597)

[c. Sequence Diagram(s) 93](#_Toc118692598)

[d. Database queries 93](#_Toc118692599)

[51. Delete Product 93](#_Toc118692600)

[a. Class Diagram 93](#_Toc118692601)

[b. Class Specification 94](#_Toc118692602)

[c. Sequence Diagram(s) 94](#_Toc118692603)

[d. Database queries 94](#_Toc118692604)

[52. Add Blog 95](#_Toc118692605)

[a. Class Diagram 95](#_Toc118692606)

[b. Class Specification 95](#_Toc118692607)

[c. Sequence Diagram(s) 96](#_Toc118692608)

[d. Database queries 96](#_Toc118692609)

[53. Update Blog 96](#_Toc118692610)

[a. Class Diagram 96](#_Toc118692611)

[b. Class Specification 97](#_Toc118692612)

[c. Sequence Diagram(s) 97](#_Toc118692613)

[d. Database queries 97](#_Toc118692614)

[54. Delete Blog 98](#_Toc118692615)

[a. Class Diagram 98](#_Toc118692616)

[b. Class Specification 98](#_Toc118692617)

[c. Sequence Diagram(s) 99](#_Toc118692618)

[d. Database queries 99](#_Toc118692619)

[55. View all Feedback (Seller) 99](#_Toc118692620)

[a. Class Diagram 99](#_Toc118692621)

[b. Class Specification 100](#_Toc118692622)

[c. Sequence Diagram(s) 100](#_Toc118692623)

[d. Database queries 100](#_Toc118692624)

[56. Reply Feedback 101](#_Toc118692625)

[a. Class Diagram 101](#_Toc118692626)

[b. Class Specification 101](#_Toc118692627)

[c. Sequence Diagram(s) 102](#_Toc118692628)

[d. Database queries 102](#_Toc118692629)

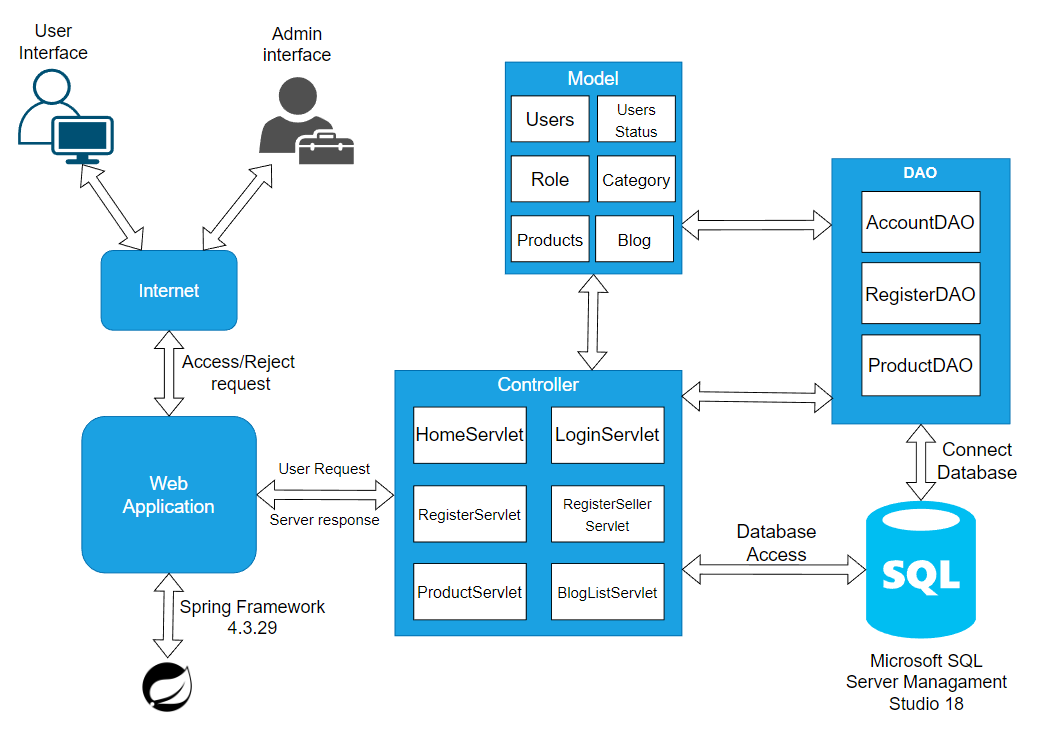
[III. Database Design 102](#_Toc118692630)

[1. Database Schema 102](#_Toc118692631)

[2. Table Description 103](#_Toc118692632)

# I. Overview

## 1. Code Packages/Namespaces



*Figure 1.1. Software Architecture Design*

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

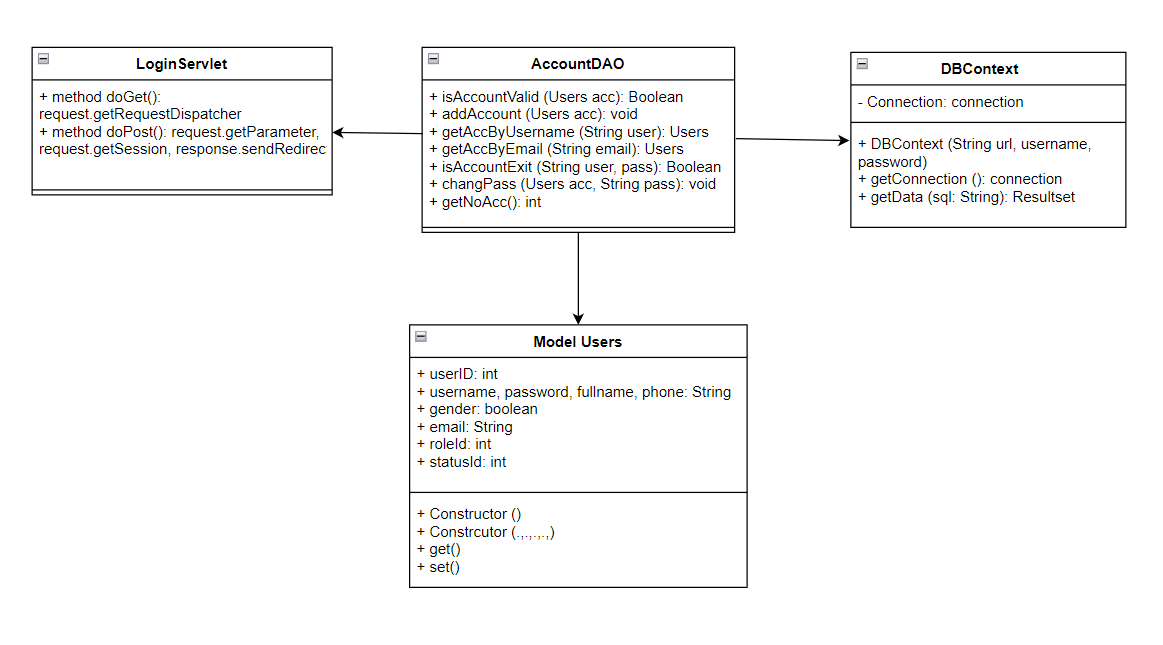
|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| *1* | *View(Web Pages)* | *Provide interaction with user* |
| *2* | *Controller* | *Control the data flow between View (Frontend) and Model (DAO), control the logic of the system, Servlet naming convention* |
| *3* | *DAO* | *Provide the connection to the Database, Extract data for the system* |
| *4* | *Filter* | *Control the redirection between Pages in View (Frontend) and from View (Frontend) to Controller, Java Class naming convention* |
| *5* | *Model* | *Define the logic of data extracted from Database, Java Class naming convention* |
| *6* | *Context* | *Connect DBContext* |
| *7* | *Libraries* | *Contains libraries used in the project* |

## 2. Coding Conventions

# II. Code Designs

## 1. Login

### a. Class Diagram



*Figure 2.1.1. Class Login Diagram*

### b. Class Specifications

#### LoginServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get all information from Login Form and compare data in database* |

#### AdminDAO, RegisterDAO, Users

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAllAccount()* | *Get all user accounts list* |
| *02* | *getAccount(String username)* | *Get user account with username* |
| *03* | *Public class Users {},*  *Public user()*  *Getter(), setter(),* | *Create a class that contains the properties of the Users table* |

### c. Sequence Diagram(s)

****

*Figure 2.1.2. Sequence Diagram*

### d. Database queries

Get Account by username:

- **Select \* from Users where username = ?;**

## 2. Register

### a. Class Diagram



*Figure 2.2.1. Class Register Diagram*

### b. Class Specifications

#### RegisterServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get all new information from register Form and update the database* |

#### RegisterDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *checkAccountExits(String username)* | *Check if the username is already exits*  *Return null* |
| *02* | *void create(String username, String password, String fullname, String phone, String sex, String email)* | *Insert a new user with username, password, fullname, phone, gender, email.* |

### c. Sequence Diagram(s)

****

*Figure 2.2.2. Sequence Diagram*

### d. Database queries

Check Account Exits:

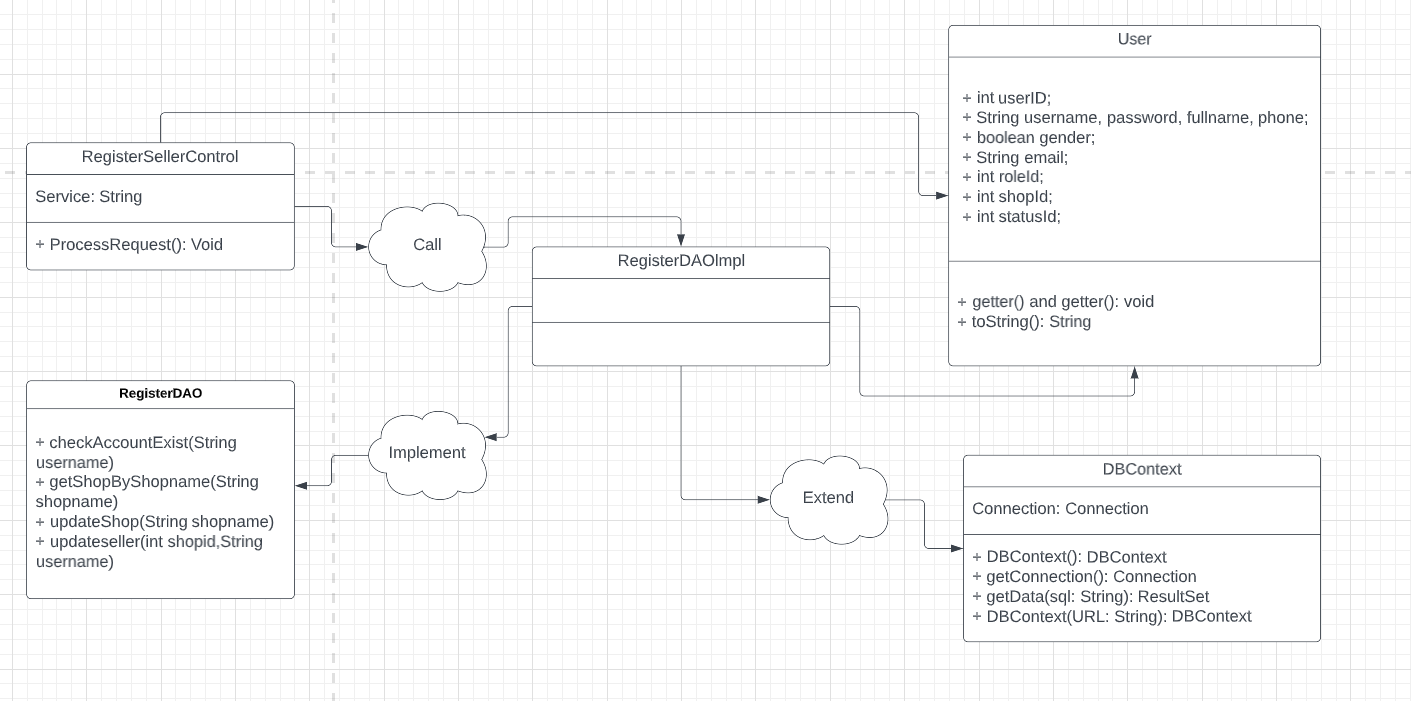
- **SELECT \* FROM [dbo].[Users] WHERE [username] = ?**

Create new account:

**- Insert into Users values(?, ?, ?, ?, ?, ?, 3, NULL, 1);**

## 3. Register for seller

### a. Class Diagram



*Figure 2.3.1. Class Register For Seller Diagram*

### b. Class Specifications

#### RegisterSellerServlet

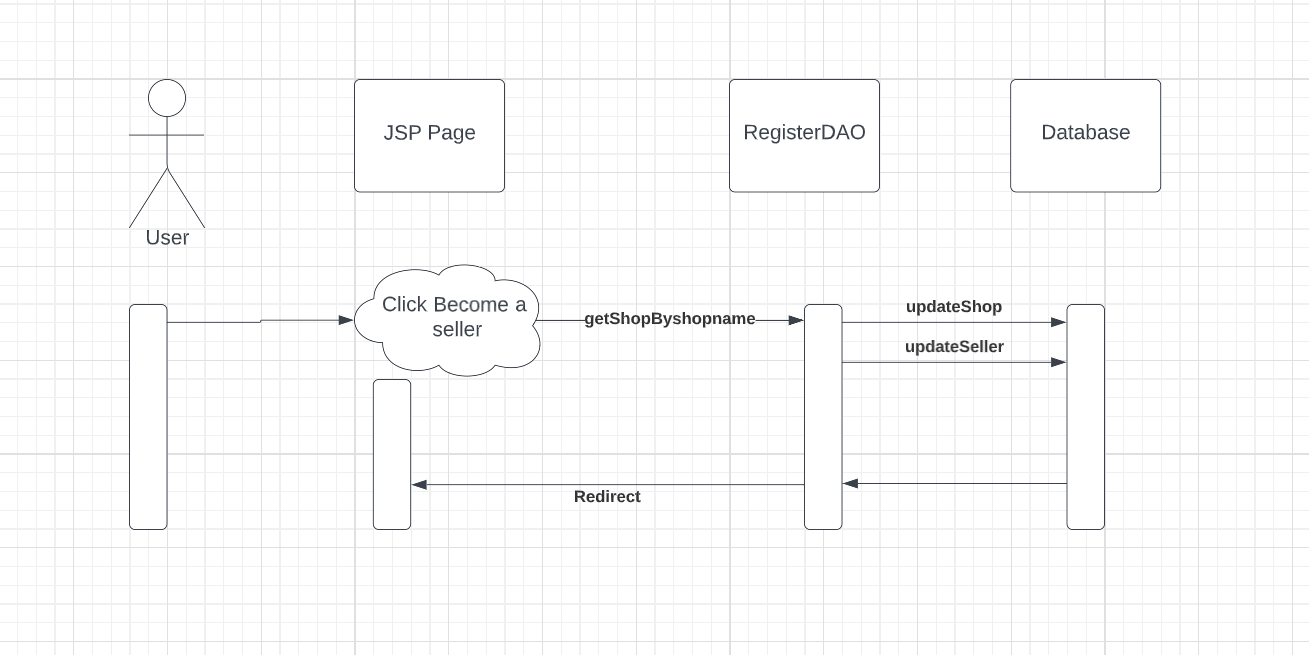
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get all new information from Seller Registration Form and update to database* |

#### RegisterDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *checkAccountExits(String username)* | *Check if the username is already exits*  *Return null* |
| *02* | *getShopByShopName(String shopname)* | *Get the shop by name in the database* |
| *03* | *void updateShop(String shopname)* | *Insert into the Shop table in the database with shop name* |
| *02* | *void updateseller(int shopid, String username)* | *Update users with RoleID = 3 to 2* |

### c. Sequence Diagram(s)



*Figure 2.3.2. Sequence Diagram*

### d. Database queries

Check Account Exits:

- **SELECT \* FROM [dbo].[Users] WHERE [username] = ?**

Get Shop by ShopName:

**- SELECT ID, ShopName FROM Shop WHERE ShopName = ?**

Create a new Shop:

**- INSERT INTO Shop VALUES (?)**

Update User Account Roles:

**- UPDATE Users SET RoleID = 2, ShopID = ? WHERE username = ?**

## 4. Feature Product

### a. Class Diagram



*Figure 2.4.1. Class Diagram*

### b. Class Specification

#### ProductsServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### CategoryDAO, SubCategoryDAO, ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *List<Category> getAll()* | *Get all the data in the Category table* |
| *02* | *List<SubCategory> getAllSubCategory()* | *Get all the data in the SubCategory table* |
| *03* | *List<Products> getProductByCid(String key, int cid, int sid, int sortType, int sortMode)* | *Get all products in the Product table by Cid, Sid* |

### c. Sequence Diagram(s)

**

*Figure 2.4.2 Sequence Diagram*

### d. Database queries

Get All Category:

**- select \* from Category;**

Get All SubCategory:

**- SELECT [SubCategoryID], [SubCategoryName], [CategoryID] FROM [dbo].[SubCategory]**

Get All Product by Cid, Sid:

**-** SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,

MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,

MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,

MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.SellerID) AS SellerID,

MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID, MIN(p.BrandID) AS BrandID,

MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID

FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID

JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID

WHERE p.ProductName LIKE ? AND p.StatusID!= 2 AND p.Amount>0 AND Sub.CategoryID= ?

AND Sub.SubCategoryID= ?

GROUP BY p.ProductID) t

ORDER BY t.SellPrice, t.SalePercent, t.ProductName ASC

## 5. Search and Fillter Product

### a. Class Diagram



*Figure 2.5.1. Class Diagram*

### b. Class Specification

#### ProductsServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### CategoryDAO, SubCategoryDAO, ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *List<Products> getProductByCid(String key, int cid, int sid, int sortType, int sortMode)* | *Search products in Products table by key and filter products by sortType, sortMode* |

### c. Sequence Diagram(s)

**

*Figure 2.5.2 Sequence Diagram*

### d. Database queries

Search and fillter Product by key (ProductName), sortType (SellPrice, SalePercent, ProductName) and sortMode (ASC, DESC):

**-** SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,

MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,

MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,

MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.SellerID) AS SellerID,

MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID, MIN(p.BrandID) AS BrandID,

MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID

FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID

JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID

WHERE p.ProductName LIKE ? AND p.StatusID!= 2 AND p.Amount>0 AND Sub.CategoryID= ?

AND Sub.SubCategoryID= ?

GROUP BY p.ProductID) t

ORDER BY t.SellPrice, t.SalePercent, t.ProductName ASC

## 6. View Product Detail

### a. Class Diragram



*Figure 2.6.1 Class Diagram*

### b. Class Specification

#### DetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### CategoryDAO, SubCategoryDAO, ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getProductByid(int id)* | *Get product in Product table using ProductID* |

### c. Sequence Diagram(s)



*Figure 2.6.2 Sequence Specification*

### d. Database queries

Get Product by ProductID:

**-** SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,

MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,

MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,

MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.SellerID) AS SellerID,

MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID, MIN(p.BrandID) AS BrandID,

MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID

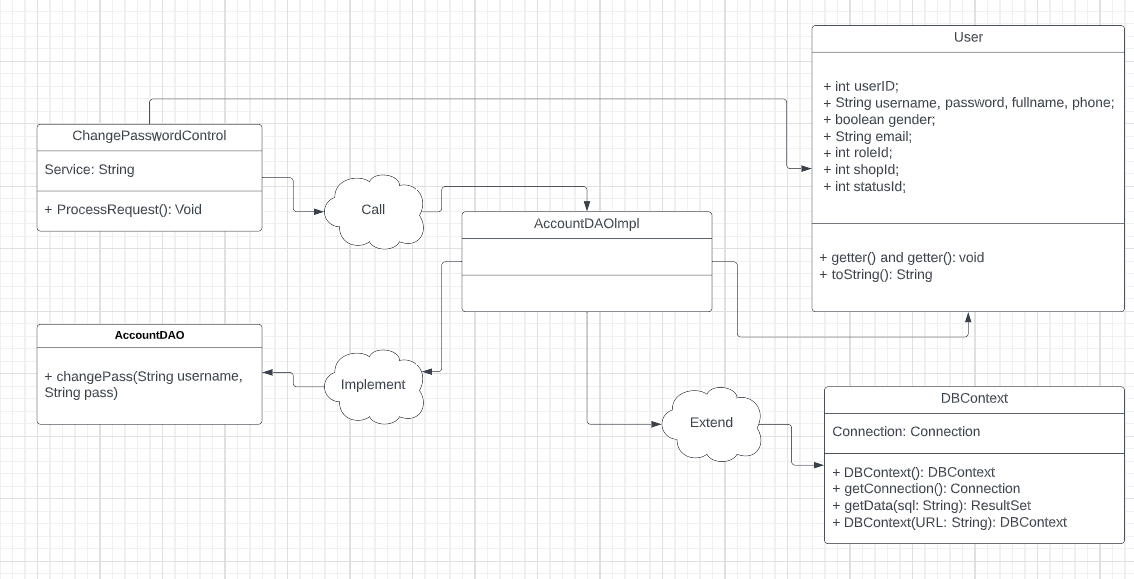
FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID

JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID

GROUP BY p.ProductID ) t where t.ProductID = ?

## 7. Change Pass

### a. Class Diragram



*Figure 2.7.1 Class Diagram*

### b. Class Specification

#### ChangePassServlet

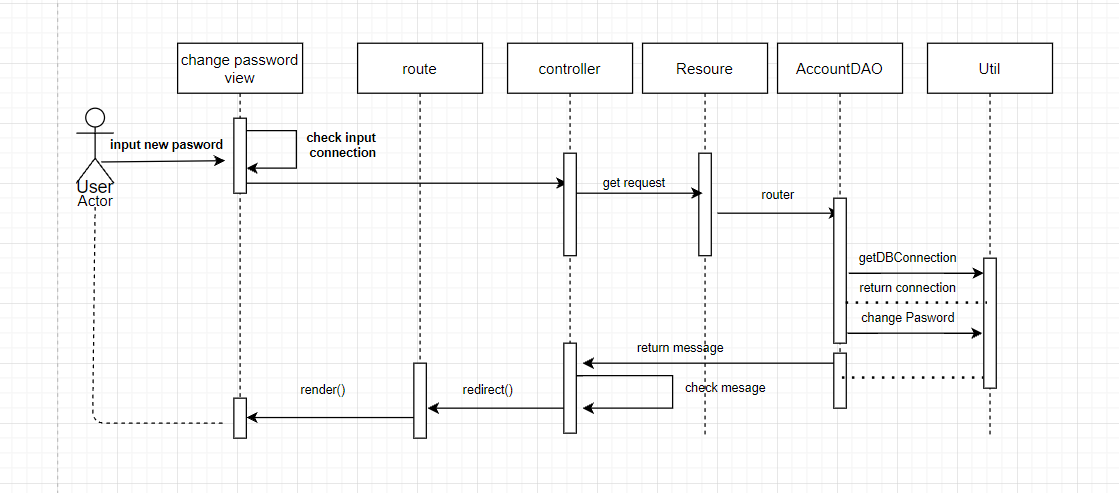
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void changePass(String username, String pass){}* | *Get data from the form and update it in the database* |

### c. Sequence Diagram(s)



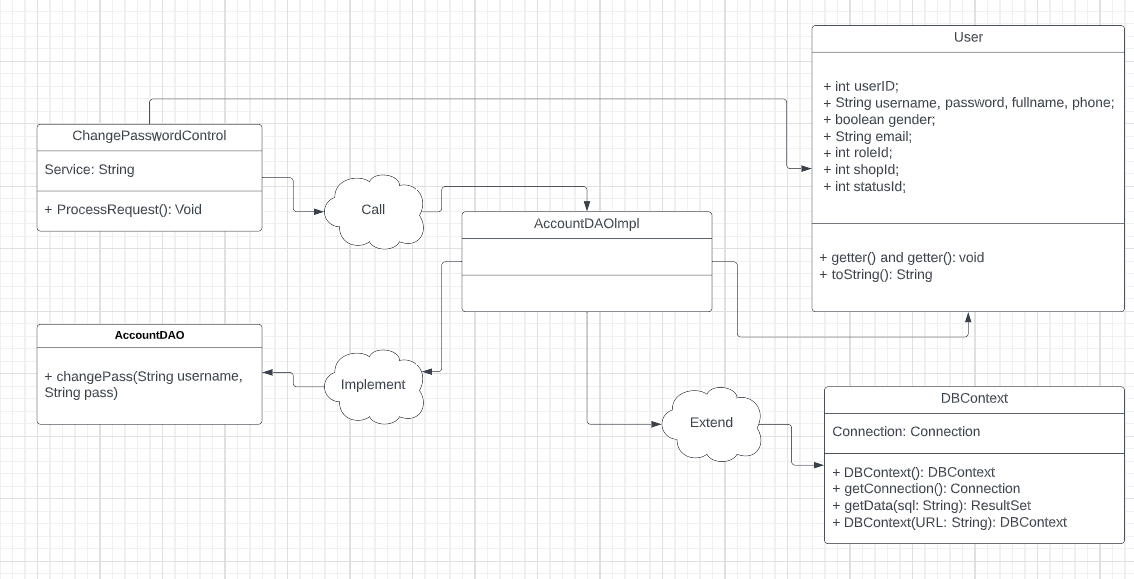
*Figure 2.7.2 Sequence Specification*

### d. Database queries

**-** update Users set Password = ? where Username = ?

## 8. Forgot Password

### a. Class Diragram



*Figure 2.8.1 Class Diagram*

### b. Class Specification

#### ChangePassServlet

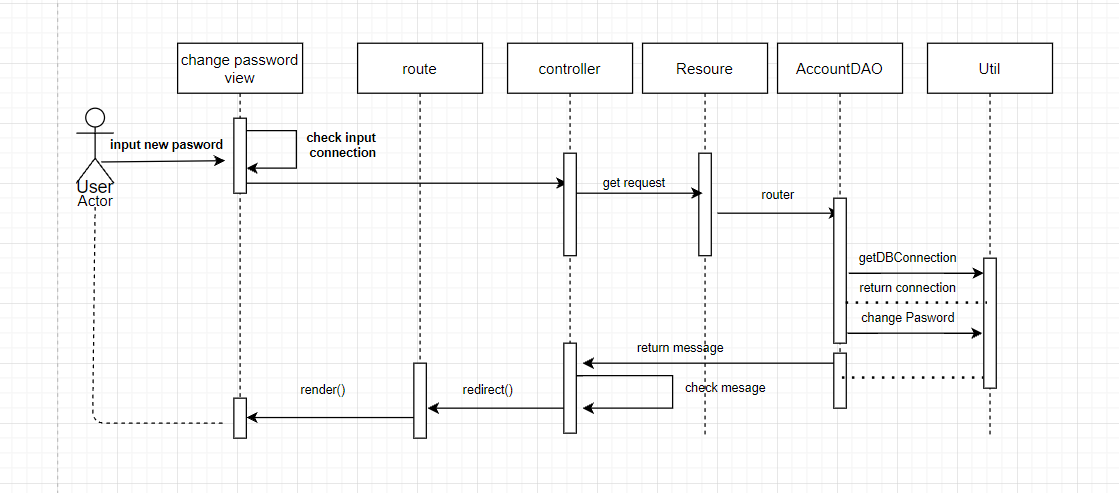
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void changePass(String username, String pass){}* | *Get data from the form and update it in the database* |

### c. Sequence Diagram(s)



*Figure 2.8.2 Sequence Specification*

### d. Database queries

**-** update Users set Password = ? where Username = ?

## 9. User Profile

### a. Class Diragram



*Figure 2.9.1 Class Diagram*

### b. Class Specification

#### UserProfileServlet

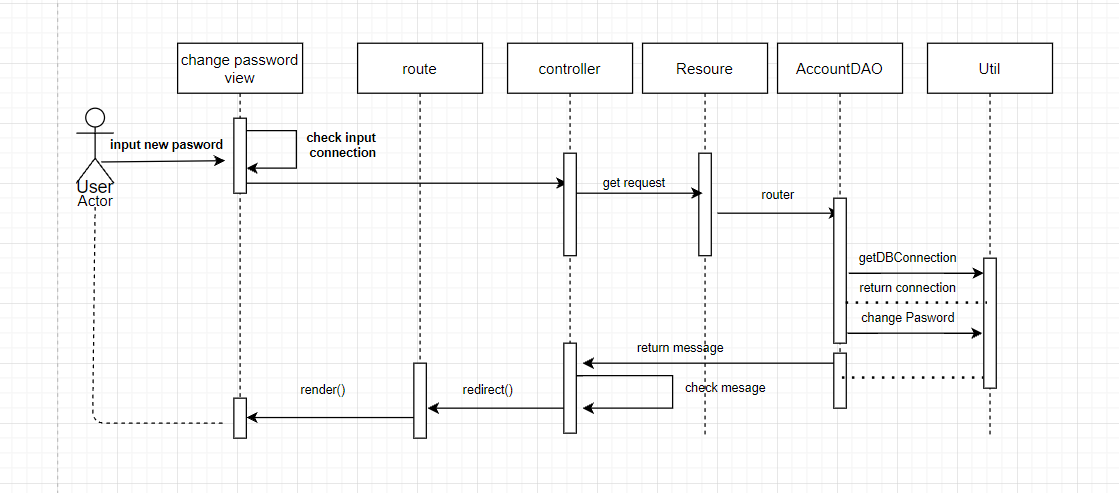
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAccount(String username)* | *Get user account in Users table using username* |

### c. Sequence Diagram(s)



*Figure 2.9.2 Sequence Specification*

### d. Database queries

**-** select \* from Users where username = ?

## 10. Update Profile

### a. Class Diragram



*Figure 2.10.1 Class Diagram*

### b. Class Specification

#### UserProfileServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAccount(String username)* | *Get user account in Users table using username* |
| *02* | *void updateUserProfile(int id, String fullname, String phone, String email, int gender)* | *Get the data from the User Profile form and update it to the Users table in the database* |

### c. Sequence Diagram(s)



*Figure 2.10.2 Sequence Specification*

### d. Database queries

Update profile:

**- update Users set fullname = ?, phone = ?, email = ?, gender = ? where UserID = ?**

## 11. Add Address

### a. Class Diragram



*Figure 2.11.1 Class Diagram*

### b. Class Specification

#### UserAddressServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO, UserAddressDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getUserByID(int id)* | *Get user account in Users table using UserID* |
| *02* | *void addAddress(int id, String fullname, String phone, int inputCity)* | *Get the data from the Address form and update it to the UserAddress table in the database* |

### c. Sequence Diagram(s)



*Figure 2.11.2 Sequence Specification*

### d. Database queries

Add new Address:

**- INSERT INTO [UserAddress] VALUES (?, ?, ?, ?, ?)**

## 12. Edit Address

### a. Class Diragram



*Figure 2.12.1 Class Diagram*

### b. Class Specification

#### addressUPServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO, UserAddressDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getUserByID(int id)* | *Get user account in Users table using UserID* |
| *02* | *void updateAddress(int addressid, int id, String fullname, String phone, int inputCity, String note)* | *Get the data from the Update Address form and update it to the UserAddress table in the database* |

### c. Sequence Diagram(s)



*Figure 2.12.2 Sequence Specification*

### d. Database queries

Update Address:

**- UPDATE [dbo].[UserAddress] SET [UserID] = ?, [ShipName] = ?, [PhoneNum] = ?,**

**[ShipCityID] = ?, [NoteDetail] = ? WHERE ID = ?**

## 13. Delete Address

### a. Class Diragram



*Figure 2.13.1 Class Diagram*

### b. Class Specification

#### DeleteAddressServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get the id of the address and execute the delete command in the database* |
| *02* | *doPost()* | *Redirect to address page* |

#### UserAddressDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *void delete(int id)* | *Delete the address in the UserAddress table in the database* |

### c. Sequence Diagram(s)



*Figure 2.13.2 Sequence Specification*

### d. Database queries

Delete Address:

**- DELETE FROM [UserAddress] WHERE ID = ?**

## 14. Show Address

### a. Class Diragram



*Figure 2.14.1 Class Diagram*

### b. Class Specification

#### UserAddressServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ShipDAO, UserAddressDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *List<UserAddress> getUserAddress(int id)* | *Get the list of addresses in the UserAddress table by UserID* |
| *02* | *List<Ship> getShip()* | *Get the list of cities in the Ship table* |

### c. Sequence Diagram(s)



*Figure 2.14.2 Sequence Specification*

### d. Database queries

List all Address:

**- SELECT \* FROM UserAddress WHERE UserID = ?**

List all City:

**- SELECT \* FROM [Ship]**

## 15. Add to Cart

### a. Class Diragram



*Figure 2.15.1 Class Diagram*

### b. Class Specification

#### cartServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ProductDAO, CartDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getProductById(int id)* | *Get products by ProductID* |
| *02* | *void SaveAllItem(Cart a, int uid)* | *Save data to Cart* |

### c. Sequence Diagram(s)



*Figure 2.15.2 Sequence Specification*

### d. Database queries

Get Product by ProductID:

**- SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,**

**MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,**

**MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,**

**MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.ShopID) AS ShopID,**

**MIN(s.ShopName) as ShopName, MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID,**

**MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID**

**FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID**

**JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID**

**JOIN dbo.Shop s ON s.ID = p.ShopID**

**GROUP BY p.ProductID ) t where t.ProductID = ?**

Add to cart:

**-INSERT INTO [dbo].[Cart] UserID] , [ProductID], [Amount]) VALUES (?,?,?)**

## 16. View Cart

### a. Class Diragram



*Figure 2.16.1 Class Diagram*

### b. Class Specification

#### ViewCartServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send data from cartServlet to .jsp file* |

#### CartDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Cart getAll(int Userid, Cart c)* | *Get the list of product in the Cart table by UserID* |

### c. Sequence Diagram(s)



*Figure 2.16.2 Sequence Specification*

### d. Database queries

Show all product in cart:

**- select \* from cart where UserID = ?**

## 17. Delete Product in Cart

### a. Class Diragram



*Figure 2.17.1 Class Diagram*

### b. Class Specification

#### DeleteInCartServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ProductDAO, CartDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getProductById(int id)* | *Get products by ProductID* |
| *02* | *void makeEdit(int pid, int uid, int amount)* | *Update prices after increasing or decreasing the number of products in the cart, or when removing products* |
| *03* | *void delAnItem(int iid, int uid)* | *Remove product from cart* |

### c. Sequence Diagram(s)



*Figure 2.17.2 Sequence Specification*

### d. Database queries

**-** UPDATE [dbo].[Cart] SET [Amount] = ? WHERE UserID = ? and ProductID = ?

**-** DELETE FROM [dbo].[Cart] WHERE UserID = ? and ProductID = ?

## 18. Create Order

### a. Class Diragram



*Figure 2.18.1 Class Diagram*

### b. Class Specification

#### CheckOutServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDAO, OrderDetailDAO, UserAddressDAO, ShipDAO, CartDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Int getShippingFee (Users u)* | *Update shipping price for each address* |
| *02* | *void createOrder(Order o)* | *Create an order and save the data in the Order table* |
| *03* | *Order getCheckOutOrder(int id)* | *Create a status for the order the user just placed* |
| *04* | *void insertOrderDetail(Cart c, int id)* | *Save data to OrderDetail table* |
| *05* | *UserAddress getDefaultAddress(int id)* | *Set default shipping address* |
| *06* | *void insertShipInfo(int id, String note ,UserAddress ud)* | *Save the user's information, shipping address, order number in the ShipInfo table* |
| *07* | *void removeCart(int uid)* | *Remove the products in the order from the cart* |

### c. Sequence Diagram(s)



*Figure 2.18.2 Sequence Specification*

### d. Database queries

**-** SELECT Ship.ShipPrice, UserAddress.ShipName, Ship.CityName

FROM Ship INNER JOIN UserAddress

ON Ship.id = UserAddress.ShipCityID

AND UserAddress.UserID = ?

**-** INSERT INTO [dbo].[Orders] ([UserID], [TotalPrice], [Note], [Status], [Date]) VALUES(?,?,?,1,?)

- SELECT TOP(1) Order\_Status.Name, Orders.ID, Orders.UserID, Orders.TotalPrice, Orders.Note, Orders.Date

FROM Order\_Status INNER JOIN Orders ON Order\_Status.ID = Orders.Status

WHERE UserID = ? ORDER BY Date DESC

- INSERT INTO [dbo].[Order\_Detail] ([Order\_ID],[ProductID],[ProductName],[ProductPrice],[Quantity]) VALUES (?,?,?,?,?)

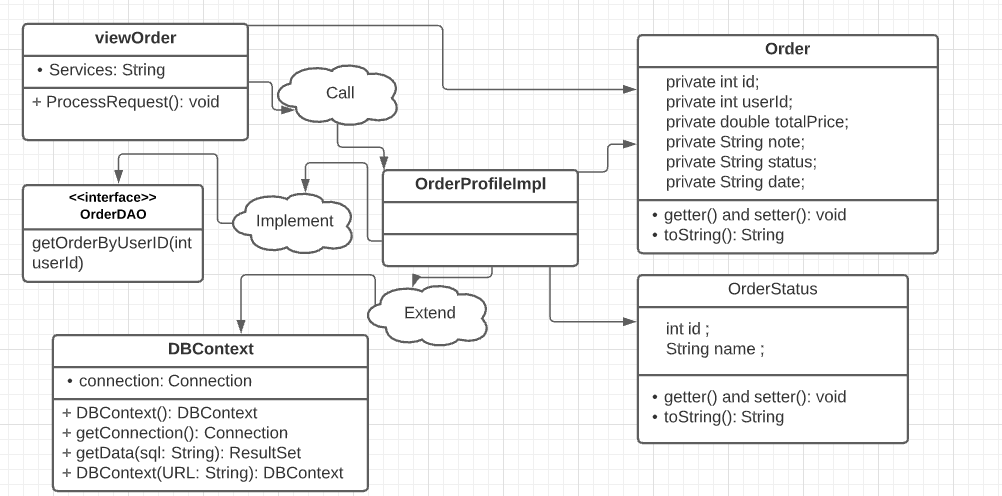
- SELECT TOP(1)\* FROM [dbo].[UserAddress] WHERE UserID = ? ORDER BY ID DESC

- INSERT INTO [dbo].[ShipInfo] ([Order\_ID], [CustomerName], [ShippingAddress], [ShipCityID], [PhoneNum], [Note]) VALUES (?,?,?,?,?,?)

- DELETE FROM [dbo].[Cart] WHERE UserID = ?

## 19. View Order

### a. Class Diragram



*Figure 2.19.1 Class Diagram*

### b. Class Specification

#### ViewOrderServlet

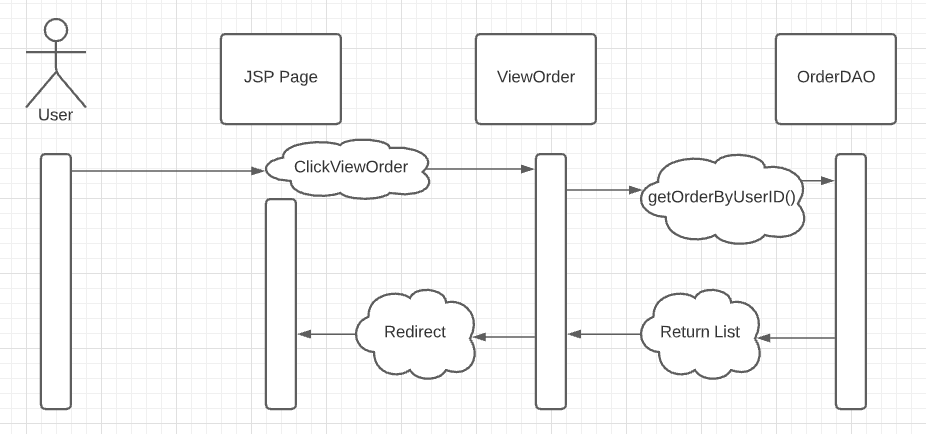
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *List<Order> getOrderByUserID(int userId)* | *Get a list of orders by UserID* |

### c. Sequence Diagram(s)



*Figure 2.19.2 Sequence Specification*

### d. Database queries

**-** SELECT o.ID, os.Name, o.TotalPrice, o.Date

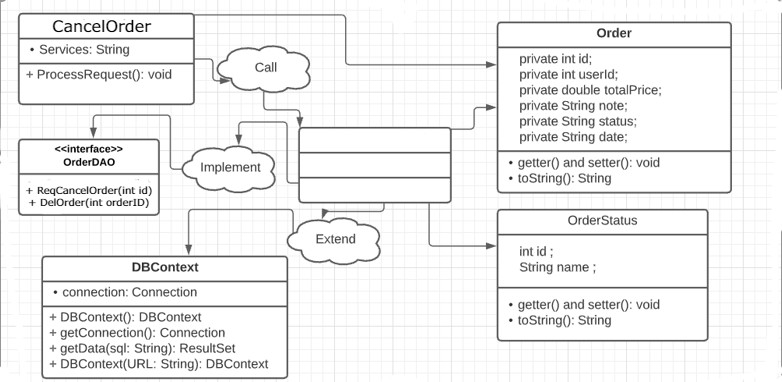
FROM Orders o INNER JOIN Order\_Status os

ON o.Status = os.ID

WHERE o.UserId = ?

## 20. Cancel Order

### a. Class Diragram



*Figure 2.20.1 Class Diagram*

### b. Class Specification

#### CancelOrderServlet

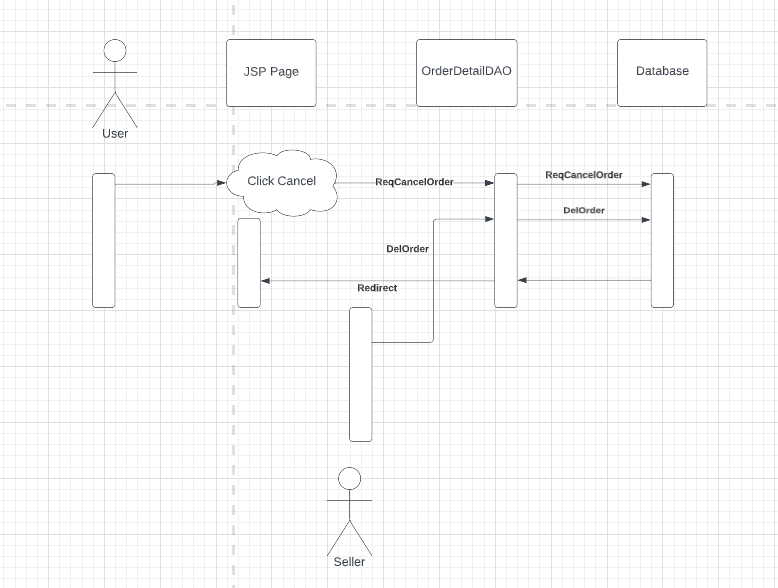
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Update order status to Canceled* |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *void ReqCancelOrder(int id)* | *Proceed to cancel the order, in the database the order status will change to Canceled* |

### c. Sequence Diagram(s)



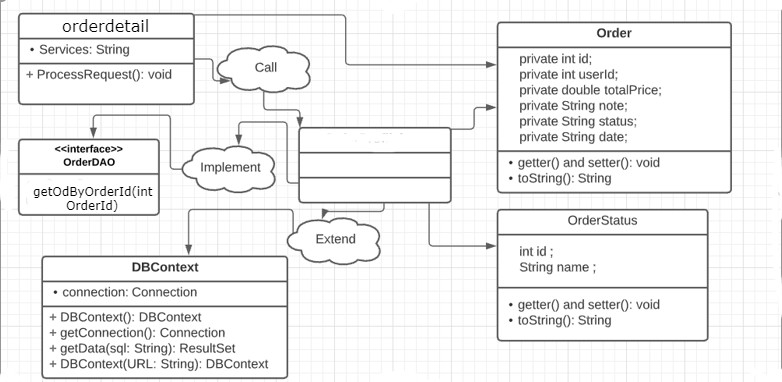
*Figure 2.20.2 Sequence Specification*

### d. Database queries

**-** UPDATE [dbo].[Orders] SET [Status] = 4 WHERE ID = ?

## 21. Order Detail

### a. Class Diragram



*Figure 2.21.1 Class Diagram*

### b. Class Specification

#### OrderDetailServlet

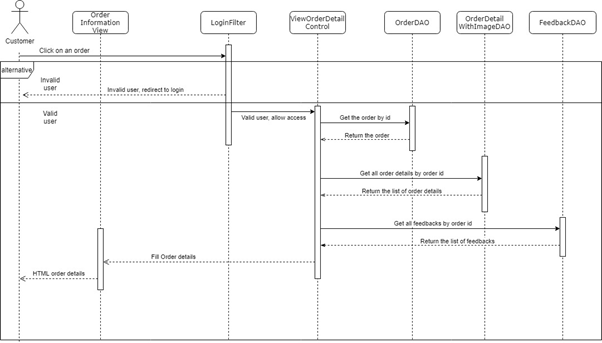
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDetailDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *List<OrderDetail> getOdByOrderId(int OrderId)* | *Get a list of products in an order by OrderID* |

### c. Sequence Diagram(s)

****

*Figure 2.21.2 Sequence Specification*

### d. Database queries

**-** select o.ProductID, o.ProductName, p.ProductImgURL, o.ProductPrice, o.Quantity

from Order\_Detail o inner join ProductImg p on o.ProductID=p.ProductID

where o.Order\_ID=?

## 22. Add product to Wishlist

### a. Class Diragram



*Figure 2.22.1 Class Diagram*

### b. Class Specification

#### AddToWishlistServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ProductDAO, FavoriteProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Products getProductById(int id){}* | *Get product in Product table by ProductID* |
| *02* | *public void addProductToWishlish(int uid, int pid){}* | *Insert data into Favorite\_Product table with UserID, ProductID* |

### c. Sequence Diagram(s)



*Figure 2.22.2 Sequence Specification*

### d. Database queries

**- INSERT INTO [dbo].[Favorite\_Product] ([UserID], [ProductID]) VALUES(?, ?)**

## 23. Delete product in Wishlist

### a. Class Diragram



*Figure 2.23.1 Class Diagram*

### b. Class Specification

#### DeleteFavoriteProductServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get id parameter to perform product removal from wishlist* |

#### ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void deleteFromWishlist(int id){}* | *Delete the data of the product in the wishlist from the Favorite\_Product table by ProductID* |

### c. Sequence Diagram(s)



*Figure 2.23.2 Sequence Specification*

### d. Database queries

**- DELETE FROM [Favorite\_Product] WHERE ProductID = ?**

## 24. Blog List

### a. Class Diagram



*Figure 2.24.1 Class Diagram*

### b. Class Specification

#### BlogListServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### BlogDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Blog> getAllBlogs(){}* | *Get all Blogs in Blogs table* |
| *02* | *public List<Users> getAllAccount(){}* | *Get the user who is the author of the Blog* |

### c. Sequence Diagram(s)



*Figure 2.24.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [dbo].[Blog]**

## 25. Blog Detail

### a. Class Diagram



*Figure 2.25.1 Class Diagram*

### b. Class Specification

#### BlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### BlogDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<BlogDetail> getBlogDetailById(int id){}* | *Retrieve detailed Blog posts by BlogID in BlogDetail table* |

### c. Sequence Diagram(s)



*Figure 2.25.2 Sequence Diagram*

### d. Database queries

**- select \* from BlogDetail, Blog where BlogDetail.BlogID = Blog.ID and Blog.ID = ?**

## 26. Search Blog

### a. Class Diagram



*Figure 2.26.1 Class Diagram*

### b. Class Specification

#### BlogListServlet, BlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### BlogDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Blog> getBlogBySearch(String key){}* | *Search in the Blog table by keyword* |
| *02* | *public List<BlogDetail> getBlogDetailBySearch(String key, int bid){}* | *Search in BlogDetail table by keyword, with BlogID* |

### c. Sequence Diagram(s)



*Figure 2.26.2 Sequence Diagram*

### d. Database queries

**- select \* from BlogDetail bd where bd.BlogID = ? and (bd.Title like ? or bd.Content like ?)**

**- select \* from Blog b JOIN Users u ON b.AuthorID = u.UserID where b.BlogTitle like ? or b.BlogContent like ? or u.fullname like ?**

## 27. Comment Blog

### a. Class Diagram



*Figure 2.27.1 Class Diagram*

### b. Class Specification

#### BlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### CommetDAO, BlogDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Blog getBlogs(int id){}* | *Get data from Blog table using BlogID* |
| *02* | *public void addComment(int BlogID, int UserID, String Comment){}* | *Add Comments to CommentBlogs table with BlogID, UserID and Comment* |

### c. Sequence Diagram(s)



*Figure 2.27.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [dbo].[Blog] WHERE ID = ?**

**- INSERT INTO [dbo].[CommentBlogs] ([BlogID], [UserID], [Comment]) VALUES (?, ?, ?)**

## 28. View Comment

### a. Class Diagram



*Figure 2.28.1 Class Diagram*

### b. Class Specification

#### BlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### CommetDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Comment> getCommentByBlogID(int id){}* | *Get all data in CommentBlogs table using BlogID* |
| *02* | *public List<Users> getAllAccount(){}* | *Get all user data in Users table* |

### c. Sequence Diagram(s)



*Figure 2.28.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [CommentBlogs] WHERE BlogID = ?**

## 29. Delete Comment

### a. Class Diagram



*Figure 2.29.1 Class Diagram*

### b. Class Specification

#### DeleteCommentServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### CommetDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void deleteComment(int id){}* | *Delete 1 comment in the CommentBlogs table with CommentID* |

### c. Sequence Diagram(s)



*Figure 2.29.2 Sequence Diagram*

### d. Database queries

**- DELETE FROM CommentBlogs WHERE ID = ?**

## 30. View Service

### a. Class Diagram



*Figure 2.30.1 Class Diagram*

### b. Class Specification

#### HelpCenterServlet, ServiceListServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### HelpDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<HelpCenter> getAllHelpCenter(){}* | *Get data from the HelpCategory table* |
| *02* | *public List<HelpTitle> getAllHelpTitle(){}* | *Get data from the HelpTitle table* |
| *03* | *public HelpTitle getHelpTitle(int stid){}* | *Get data from HelpTitle table with TitleID* |
| *04* | *public List<HelpContent> getHelpContentByID(int seid, int stid){}* | *Get data from HelpContent table with TitleID* |

### c. Sequence Diagram(s)



*Figure 2.30.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [dbo].[HelpCategory]**

**- SELECT \* FROM [dbo].[HelpTitle]**

**- SELECT \* FROM [HelpTitle] WHERE TitleID = ?**

**- SELECT \* FROM (SELECT co.ContentID, MIN(co.Content) AS Content,**

**MIN(co.TitleID) AS TitleID, MIN(ht.Title) AS Title, MIN(ht.CategoryID) AS CategoryID**

**FROM HelpContent co JOIN HelpTitle ht ON co.TitleID = ht.TitleID**

**AND ht.CategoryID = ? AND ht.TitleID = ? GROUP BY co.ContentID) t**

## 31. Search Service

### a. Class Diagram



*Figure 2.31.1 Class Diagram*

### b. Class Specification

#### SearchServiceServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### HelpDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<HelpContent> getHelpContentBySearch(String key) {}* | *Search in HelpContent table by keyword* |

### c. Sequence Diagram(s)



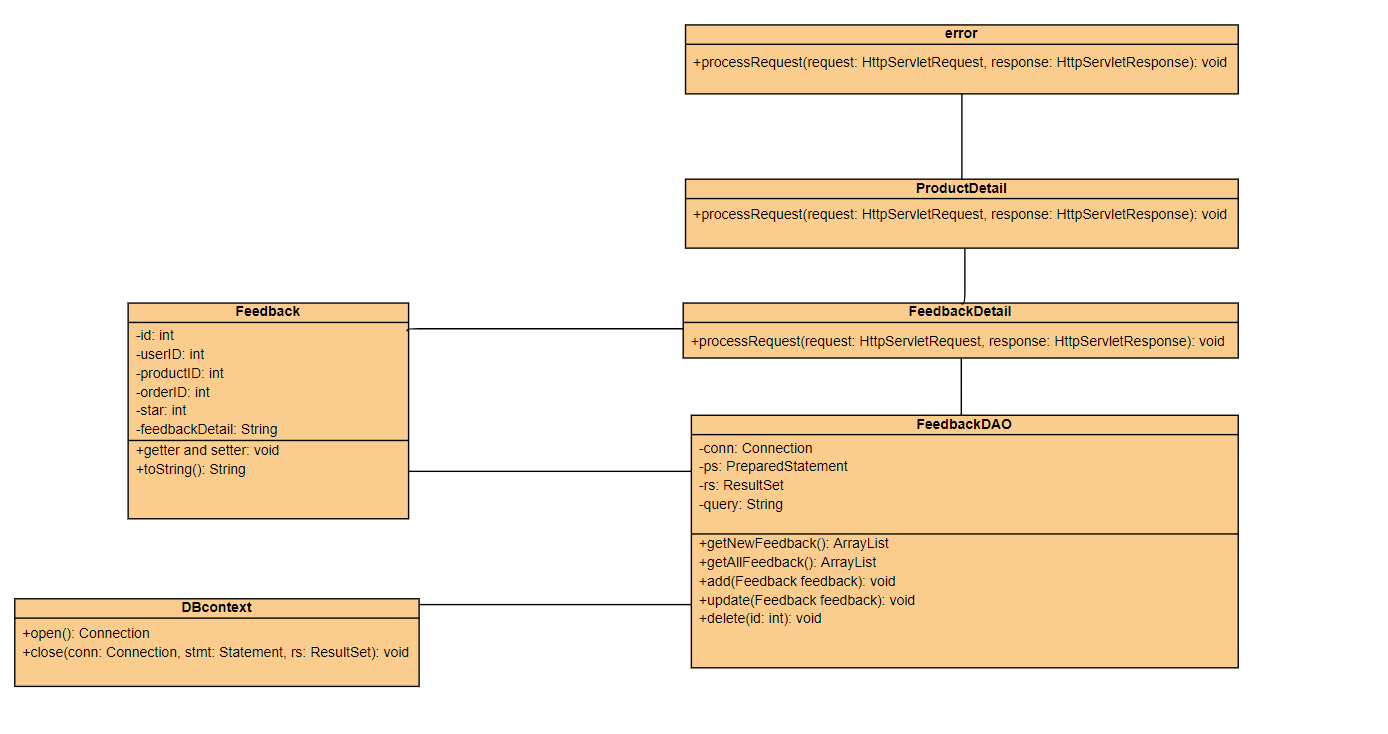
*Figure 2.31.2 Sequence Diagram*

### d. Database queries

**- select h.\* from HelpContent h where h.Content like ?**

## 32. View Feedback

### a. Class Diagram



*Figure 2.32.1 Class Diagram*

### b. Class Specification

#### DetailServlet

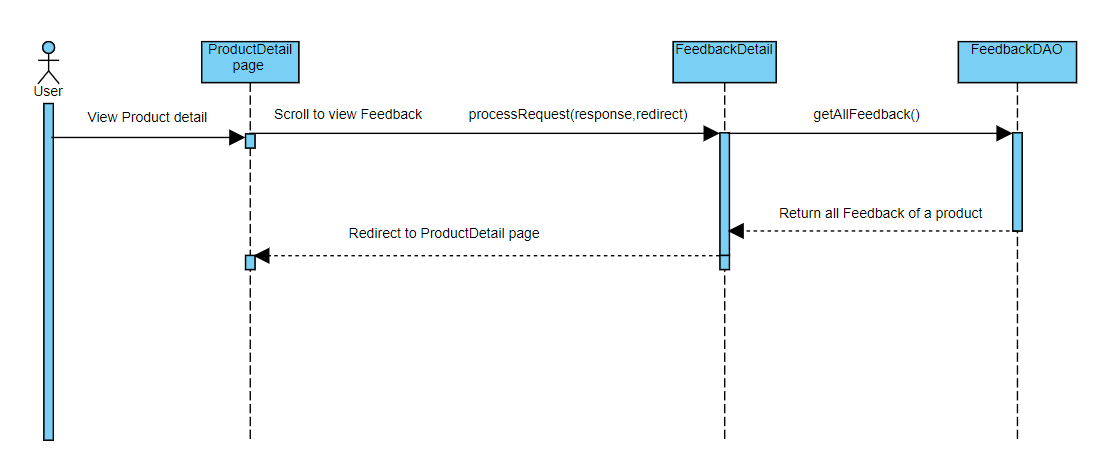
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### FeedbackDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Feedback> getFeedbackbyProductID(int productId){}* | *All feedback of a product in database* |

### c. Sequence Diagram(s)



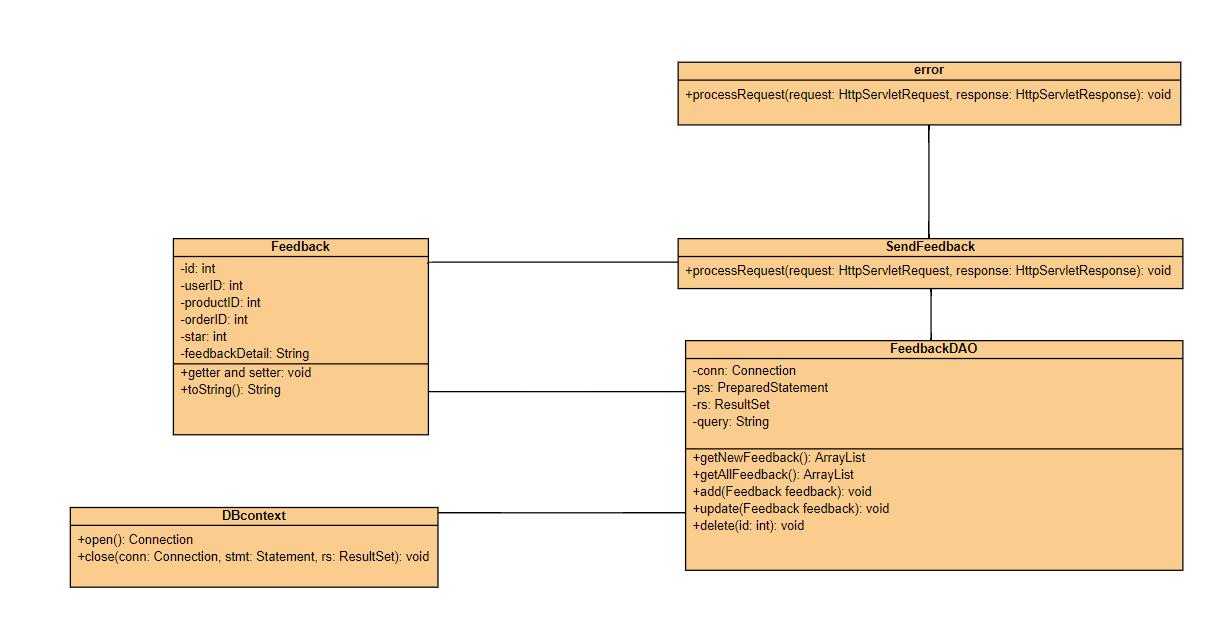
*Figure 2.32.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM Feedback WHERE ProductID = ?**

## 33. Add Feedback

### a. Class Diagram



*Figure 2.33.1 Class Diagram*

### b. Class Specification

#### FeedBackDetailServlet

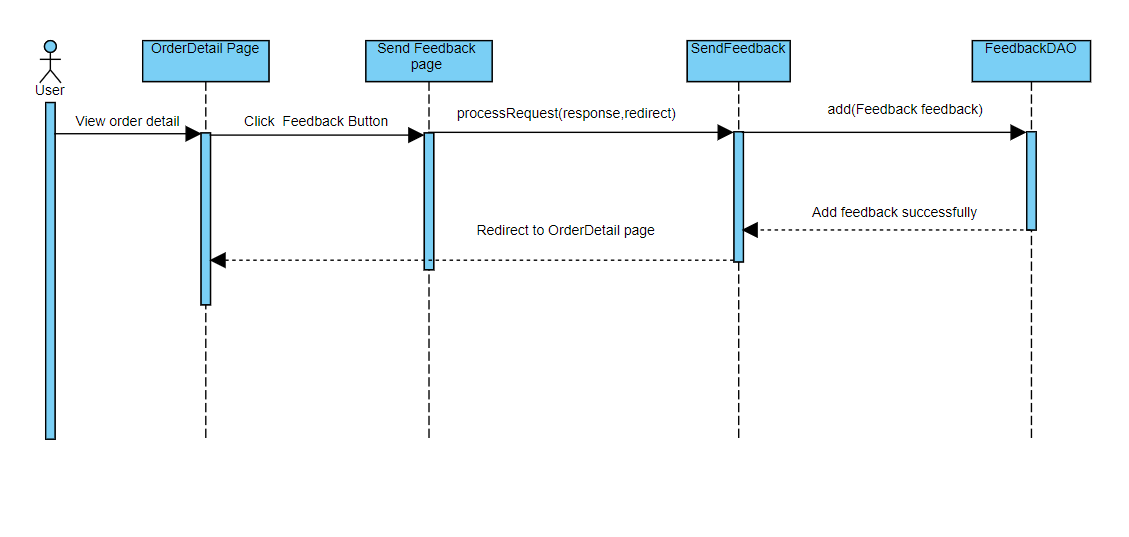
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### FeedbackDAO, OrderDetailDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void insertFeedback(int userid, int productid, int orderid, int star, String review){}* | *Add feedback with detail entered in addFeedback.jsp page* |

### c. Sequence Diagram(s)



*Figure 2.33.2 Sequence Diagram*

### d. Database queries

**- insert into Feedback values(?,?,?,?,?)**

**- select o.ProductID, o.ProductName, p.ProductImgURL, o.ProductPrice, o.Quantity**

**from Order\_Detail o inner join ProductImg p on o.ProductID=p.ProductID where o.Order\_ID=?**

## 34. View Shop

### a. Class Diagram



*Figure 2.34.1 Class Diagram*

### b. Class Specification

#### ViewShopServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ShopDAO, AccountDAO, CategoryDAO, ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Shop getShopById(int id) {}* | *Get data from Shop table using ShopID* |
| *02* | *public Users getSellerByShopID(int id){}* | *Get user data in Users table with ShopID* |
| *03* | *public List<Category> getAll(){}* | *Get all the data in the Category table* |
| *04* | *public List<Products> getProductsbyShopid(int id, String key, int cid, int sortType, int sortMode){}* | *Get all product data in Products table by ShopID, keyword to search, CategoryID, sort by attribute, sort ascending or descending* |

### c. Sequence Diagram(s)



*Figure 2.34.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM Shop WHERE ID = ?**

**- SELECT \* FROM [dbo].[Users] WHERE ShopID = ?**

**- select \* from Category**

**- SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,**

**MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,**

**MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,**

**MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.ShopID) AS ShopID,**

**MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID,**

**MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID**

**FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID**

**JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID**

**WHERE p.ShopID = ? AND p.ProductName LIKE ? AND p.StatusID!= 2 AND p.Amount>0**

**AND Sub.CategoryID = ? GROUP BY p.ProductID) t**

**ORDER BY (t.SellPrice/ t.SalePercent/ t.ProductName) ASC/DESC**

## 35. Report Shop

### a. Class Diagram



*Figure 2.35.1 Class Diagram*

### b. Class Specification

#### ReportPageServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send data from database to .jsp page* |
| *02* | *doPost()* | *Get data from the form to add to the Report table in the database* |

#### ShopDAO, ReportDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Shop> getAllShop() {}* | *Get all shop for users to report* |
| *02* | *public void addReport(int id, String title, int inputShop, String content){}* | *Insert a report into the Report table in the database* |

### c. Sequence Diagram(s)



*Figure 2.35.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [Shop]**

**- INSERT INTO [dbo].[Report] ([UserID], [ReportTitle], [ShopID], [ReportContent], [Date]) VALUES (?, ?, ?, ?, getDATE())**

## 36. Chat with seller

### a. Class Diagram



*Figure 2.36.1 Class Diagram*

### b. Class Specification

#### ChatServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *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()* | *This post method will get message which user post and insert it into database* |

#### MessagesDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void SaveNewMessage(int room, String message, int Uid){}* | *This method will insert new message into database* |

### c. Sequence Diagram(s)



*Figure 2.36.2 Sequence Diagram*

### d. Database queries

**- INSERT INTO [dbo].[messsages] ([room\_id], [UserID], [Message], [Date]) VALUES (?,?,?,?)**

## 37. Admin dashboard

### a. Class Diagram



*Figure 2.37.1 Class Diagram*

### b. Class Specification

#### DashboardServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDAO, AccountDAO, ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Integer> getALL(){}* | *Get data in Tracking table to graph* |
| *02* | *public int[] getStatis(){}* | *Get data in Order table to graph* |
| *03* | *public int countProducts(){}* | *Count the total number of products* |
| *04* | *public int getNoAcc(){}* | *Count the number of accounts that are customers* |
| *05* | *public int getSeller(){}* | *Count the number of accounts that are sellers* |

### c. Sequence Diagram(s)



*Figure 2.37.2 Sequence Diagram*

### d. Database queries

**- select \* from tracking**

**- SELECT COUNT(ProductID) AS Count FROM dbo.Product**

**- SELECT COUNT(UserID) AS Count,RoleID FROM dbo.Users WHERE RoleID=3 GROUP BY RoleID**

- **SELECT COUNT(UserID) AS Count, RoleID FROM dbo.Users WHERE RoleID=2 GROUP BY RoleID**

**- select distinct a.ID as OrderID, a.UserID,a.TotalPrice,a.Note,d.Name as StatusName,a.Date as Date from Orders as a,Order\_Detail as b, Product as c, Order\_Status as d**

**where a.ID = b.Order\_ID and b.ProductID = c.ProductID and d.ID = a.Status**

## 38. List of Account

### a. Class Diagram



*Figure 2.38.1 Class Diagram*

### b. Class Specification

#### AccountServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Users> getAllAccount(){}* | *Get all data in User table in database* |
| *02* | *public int countAllUserBySearch(String key){}* | *Count the number of accounts in the database* |

### c. Sequence Diagram(s)



*Figure 2.38.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [dbo].[Users]**

**- select count(u.UserID) from Users u JOIN Role r ON u.RoleID = r.ID where u.username like ? or u.fullname like ? or r.RoleName like ?**

## 39. Account detail

### a. Class Diagram



*Figure 2.39.1 Class Diagram*

### b. Class Specification

#### ProfileServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with id parameters* |

#### AdminDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Users getUserByID(int id) {}* | *Get data in Users table using UserID* |

### c. Sequence Diagram(s)



*Figure 2.39.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [Users] WHERE UserID = ?**

## 40. Search Account

### a. Class Diagram



*Figure 2.40.1 Class Diagram*

### b. Class Specification

#### AccountServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### AdminDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Users> getUserBySearch(String key) {}* | *Search for accounts in the Users table by Username, fullname, role* |

### c. Sequence Diagram(s)



*Figure 2.40.2 Sequence Diagram*

### d. Database queries

**- select \* from Users u JOIN Role r ON u.RoleID = r.ID where u.username like ? or u.fullname like ? or r.RoleName like ?**

## 41. List of Shop

### a. Class Diagram



*Figure 2.41.1 Class Diagram*

### b. Class Specification

#### ShopServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ShopDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Shop> getAllShop() {}* | *Get all the data in the Shop table in the database* |
| *02* | *public List<Users> getAllAccount(){}* | *Get all the data in the Users table in the database* |

### c. Sequence Diagram(s)



*Figure 2.41.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [Shop]**

**- SELECT \* FROM [dbo].[Users]**

## 42. Shop detail

### a. Class Diagram



*Figure 2.42.1 Class Diagram*

### b. Class Specification

#### ShopDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ShopDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Shop getShopById(int id) {}* | *Get the data of a shop in the Shop table using ShopID* |
| *02* | *public Users getSellerByShopID(int id){}* | *Get the seller's data in the User table using ShopID* |
| *03* | *public int getTotalProduct(int id){}* | *To count of product of a shop* |

### c. Sequence Diagram(s)



*Figure 2.42.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM Shop WHERE ID = ?**

**- SELECT \* FROM [dbo].[Users] WHERE ShopID = ?**

**- SELECT count(ProductID) FROM [Product] WHERE ShopID = ?**

## 43. Search Shop

### a. Class Diagram



*Figure 2.43.1 Class Diagram*

### b. Class Specification

#### ShopServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ShopDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Shop> getUserBySearch(String key) {}* | *Search in Shop table by keyword* |

### c. Sequence Diagram(s)



*Figure 2.43.2 Sequence Diagram*

### d. Database queries

**- select \* from Shop s JOIN Users u ON s.ID = u.ShopID where s.ShopName like ? or u.fullname like ?**

## 44. List of Report

### a. Class Diagram



*Figure 2.44.1 Class Diagram*

### b. Class Specification

#### ViewReportServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### ReportDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Users> getAllAccount() {}* | *Get all user data in the Users table in the database* |
| *02* | *public List<Report> getAllReport(){}* | *Get all Report data in Report table in database* |
| *03* | *public int countReport(){}* | *Count the total number of reports in the Report table* |

### c. Sequence Diagram(s)



*Figure 2.44.2 Sequence Diagram*

### d. Database queries

**- SELECT \* FROM [dbo].[Users]**

- **SELECT \* FROM Report**

**- SELECT COUNT(ReportID) AS Count FROM dbo.Report**

## 45. Seller dashboard

### a. Class Diagram



*Figure 2.45.1 Class Diagram*

### b. Class Specification

#### SellerDashboardServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### SellerDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<OrderSeller> getAllProductOrderBySeller(int id) {}* | *Get the data of all orders in a shop (ShopID)* |
| *02* | *public int countTotalCustomerBySeller(int id){}* | *Count the number of people who have ordered from the shop* |
| *03* | *public int countTotalProductBySeller(int id) {}* | *Count the total number of products of the shop* |
| *04* | *public int countTotalOrderBySeller(int id){}* | *Count the total number of customer orders with products from the shop* |

### c. Sequence Diagram(s)



*Figure 2.45.2 Sequence Diagram*

### d. Database queries

**- select count(distinct o.UserID) from Orders o , Order\_Detail od, Product p where o.ID = od.Order\_ID and p.ProductID=od.ProductID and p.ShopID = ?**

**- select sum(p.Amount) from Product p where p.ShopID = ?**

**- select count(distinct od.Order\_ID) from Order\_Detail od, Product p where od.ProductID = p.ProductID and p.ShopID = ?**

**- select distinct o.ID, od.ProductName,od.ProductPrice,o.Status, o.Date, od.Quantity from Orders o, Order\_Detail od, Product p, Order\_Status os where p.ShopID = ? and o.ID = od.Order\_ID and p.ProductID = od.ProductID**

## 46. View all order

### a. Class Diagram



*Figure 2.46.1 Class Diagram*

### b. Class Specification

#### ManageOrderServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Send attribute to .jsp page with keyword parameters* |

#### AccountDAO, OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Order> getAllByShopID(int id) {}* | *Get data of all orders of a shop by ShopID* |
| *02* | *public String getUserByID(int id){}* | *Retrieve the person's order data with UserID* |

### c. Sequence Diagram(s)



*Figure 2.46.2 Sequence Diagram*

### d. Database queries

**- select distinct a.ID as OrderID, a.UserID,a.TotalPrice,a.Note,d.Name as StatusName,a.Date as Date from Orders as a,Order\_Detail as b, Product as c, Order\_Status as d**

**where a.ID = b.Order\_ID and b.ProductID = c.ProductID and c.ShopID = ? and d.ID = a.Status**

## 47. Update status order

### a. Class Diagram



*Figure 2.47.1 Class Diagram*

### b. Class Specification

#### ManageOrderServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void upDateOrderStatus(String note, String status, int id) {}* | *Update order status and save to database* |

### c. Sequence Diagram(s)



*Figure 2.47.2 Sequence Diagram*

### d. Database queries

**- UPDATE [dbo].[Orders] SET [Note] = ?, [Status] = ? WHERE ID = ?**

## 48. Delete order

### a. Class Diagram



*Figure 2.48.1 Class Diagram*

### b. Class Specification

#### ManageOrderServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Send attribute to .jsp page with keyword parameters* |

#### OrderDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void DelOrder(int orderID){}* | *Delete an order data in the database* |

### c. Sequence Diagram(s)



*Figure 2.48.2 Sequence Diagram*

### d. Database queries

**- delete from Order\_Detail where Order\_ID = ?**

**delete from Feedback where OrderID = ?**

**delete from ShipInfo where Order\_ID = ?**

**delete Orders where ID = ?**

## 49. Add product

### a. Class Diagram



*Figure 2.49.1 Class Diagram*

### b. Class Specification

#### AddNewProduct

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get all the data in the SubCategory table to put on the website* |
| *02* | *doPost()* | *Get data from the form and insert it into the Products table in the database* |

#### ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Products getInsertProduct(int id){}* | *Save the data in the Products table in the database* |
| *02* | *public void insertProductImg(int id, String image){}* | *Save the image path in the ProductImg table in the database* |

### c. Sequence Diagram(s)



*Figure 2.49.2 Sequence Diagram*

### d. Database queries

**- SELECT TOP(1) Product.ProductID, Product.ProductName, Product.Description,**

**Product.OriginalPrice, Product.SellPrice, Product.SalePercent,**

**Product.SubCategoryID, Product.ShopID, Shop.ShopName, Product.Amount, Product.StatusID, SubCategory.CategoryID FROM Product INNER JOIN SubCategory**

**ON Product.SubCategoryID = SubCategory.SubCategoryID CROSS JOIN Shop**

**WHERE Product.ShopID = ? order by Product.ProductID desc**

**- INSERT INTO [dbo].[ProductImg] ([ProductID], [ProductImgURL]) VALUES (?,?)**

## 50. Update Product

### a. Class Diagram



*Figure 2.50.1 Class Diagram*

### b. Class Specification

#### UpdateProduct

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get the data of a product in the Product table using ProductID* |
| *02* | *doPost()* | *Get new data from form and update to database* |

#### ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public Products getProductById(int id){}* | *Get the data of a product in the Product table using ProductID* |
| *02* | *public void update(Products p){}* | *Store data in the Products table in the database* |
| *03* | *public void updateImg(Products p){}* | *Save image path to ProductImg table in database* |

### c. Sequence Diagram(s)



*Figure 2.50.2 Sequence Diagram*

### d. Database queries

**- UPDATE [dbo].[Product] SET [ProductName] = ?, [Description] = ?, [OriginalPrice] = ?, [SellPrice] = ?, [SalePercent] = ?, [SubCategoryID] = ?, [ShopID] = ?, [Amount] = ?,**

**[StatusID] = 1 WHERE ProductID =?**

**- UPDATE [dbo].[ProductImg] SET [ProductImgURL] = ? WHERE [ProductID] = ?**

## 51. Delete Product

### a. Class Diagram



*Figure 2.51.1 Class Diagram*

### b. Class Specification

#### DeleteProduct

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get the data of a product in the Product table using id parameter* |

#### ProductDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void deleteProduct(int id) {}* | *Delete product data in the Product table in the database* |

### c. Sequence Diagram(s)



*Figure 2.51.2 Sequence Diagram*

### d. Database queries

**- DELETE FROM Order\_Detail WHERE ProductID = ?**

**DELETE FROM Cart WHERE ProductID = ?**

**DELETE FROM Feedback WHERE ProductID = ?**

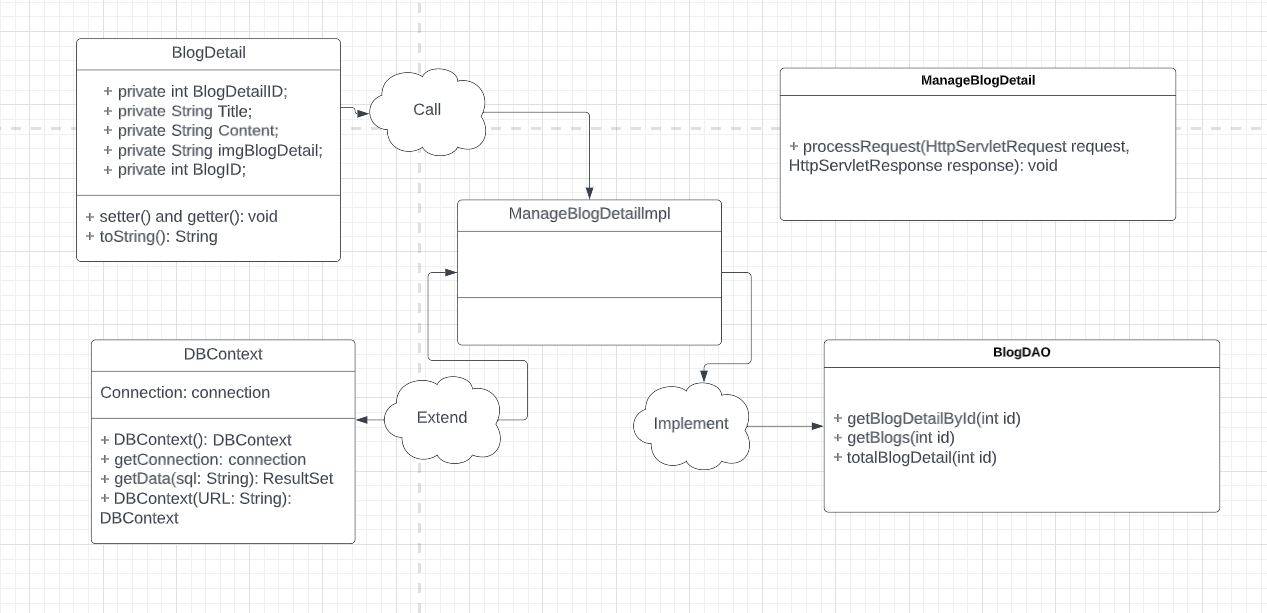
**DELETE FROM Feedback\_Replies WHERE ProductID = ?**

**DELETE FROM ProductImg WHERE ProductID = ?**

**DELETE FROM Product WHERE ProductID = ?**

## 52. Add Blog

### a. Class Diagram



*Figure 2.52.1 Class Diagram*

### b. Class Specification

#### AddBlogServlet, AddBlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doPost()* | *Get data from form* |

#### BlogDAO, AccountDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void addBlog1(String content, String title, String imageLink, int id){}* | *Insert data into Blog table with UserID* |
| *02* | *public Users getAccById(int id){}* | *Get UserID to perform data insertion* |
| *03* | *public void addBlogDetail(String content, String title, String imgBlogDetail, int id){}* | *Insert data into BlogDetail table with BlogID and UserID* |

### c. Sequence Diagram(s)



*Figure 2.52.2 Sequence Diagram*

### d. Database queries

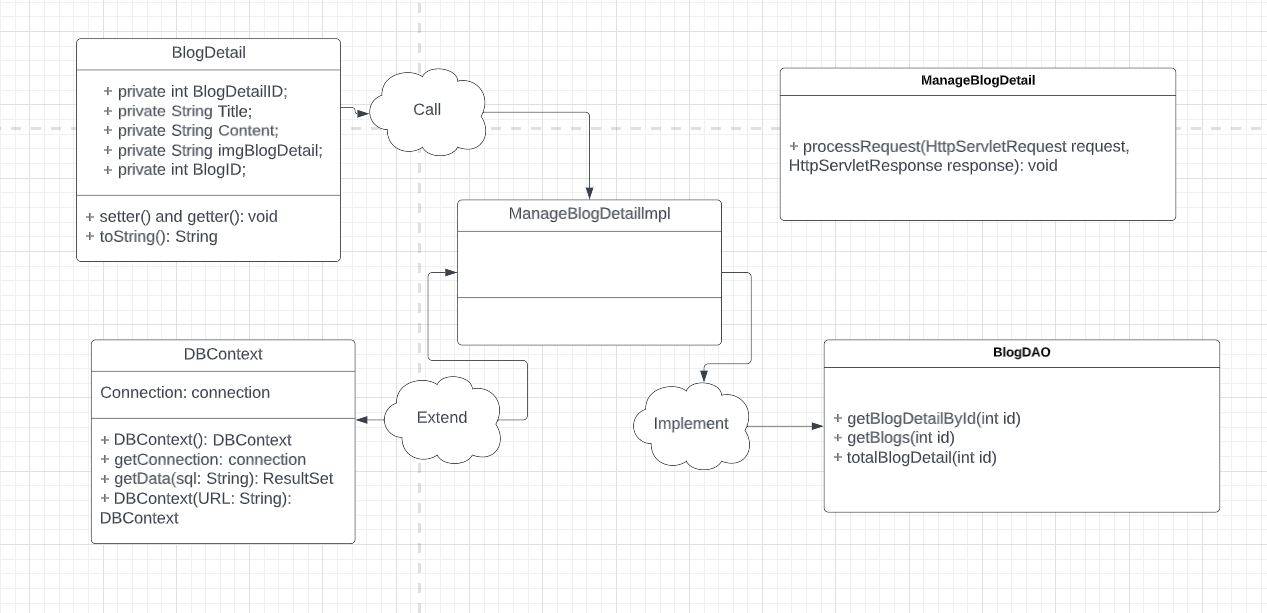
**- INSERT INTO dbo.Blog([Day], [Month], [Year], BlogTitle, BlogContent, imageLink, AuthorID) VALUES(1, 'November', 2022, ?, ?, ?, ?)**

**- select \* from dbo.[Users] where UserID=?**

**- INSERT INTO dbo.BlogDetail(Title, Content, imgBlogDetail, BlogID) VALUES(?,?,?,?)**

## 53. Update Blog

### a. Class Diagram



*Figure 2.53.1 Class Diagram*

### b. Class Specification

#### UpdateBlogServlet, UpdateBlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get data from the database to put on the website* |
| *02* | *doPost()* | *Get data from the form and update it in the database* |

#### BlogDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void updateBlog(int id, String title, String content, String imageLink){}* | *Update new data in Blog table* |
| *02* | *public void updateBlogdetail(int id, String title, String content, String imgBlogDetail){}* | *Update new data in BlogDetail table* |

### c. Sequence Diagram(s)



*Figure 2.53.2 Sequence Diagram*

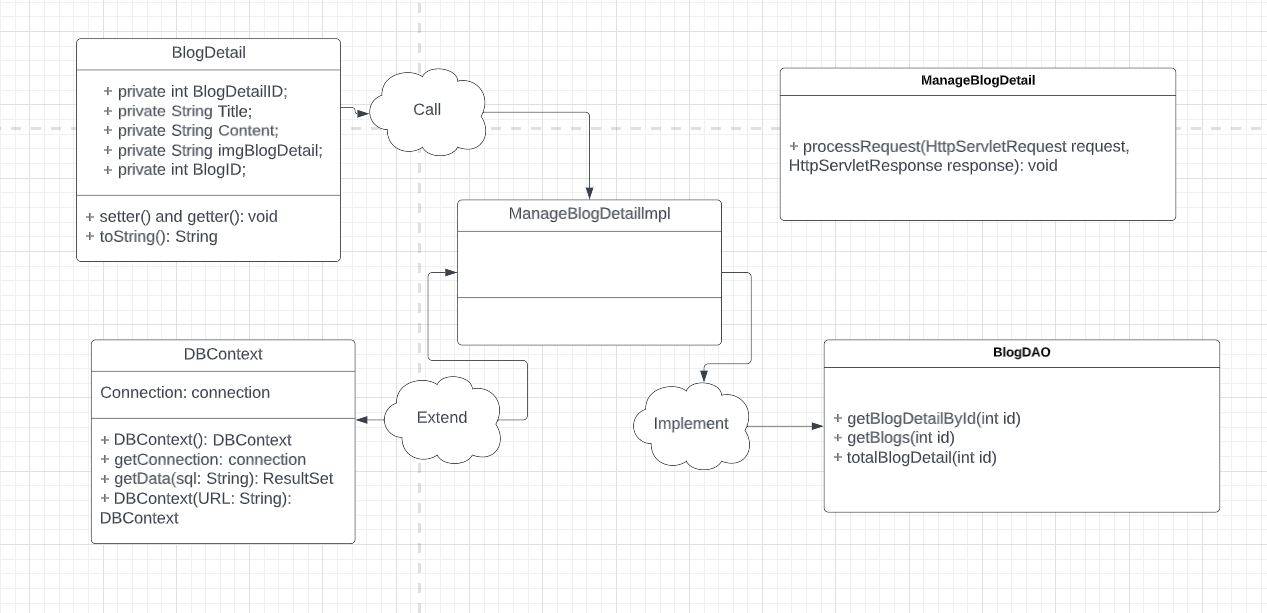
### d. Database queries

**- UPDATE [Blog] SET [BlogTitle] = ? ,[BlogContent] = ?, [imageLink] = ? WHERE ID = ?**

**- UPDATE [BlogDetail] SET [Title] = ? ,[Content] = ? ,[imgBlogDetail] = ? WHERE BlogDetailID = ?**

## 54. Delete Blog

### a. Class Diagram



*Figure 2.54.1 Class Diagram*

### b. Class Specification

#### DeleteBlogServlet, DeleteBlogDetailServlet

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get paramater id from jsp page to delete Blog in database* |

#### BlogDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void delete(int id) {}* | *Delete data in Blog table by BlogID* |
| *02* | *public void deleteDetail(int id){}* | *Delete data in BlogDetail table by BlogDetailID* |

### c. Sequence Diagram(s)



*Figure 2.54.2 Sequence Diagram*

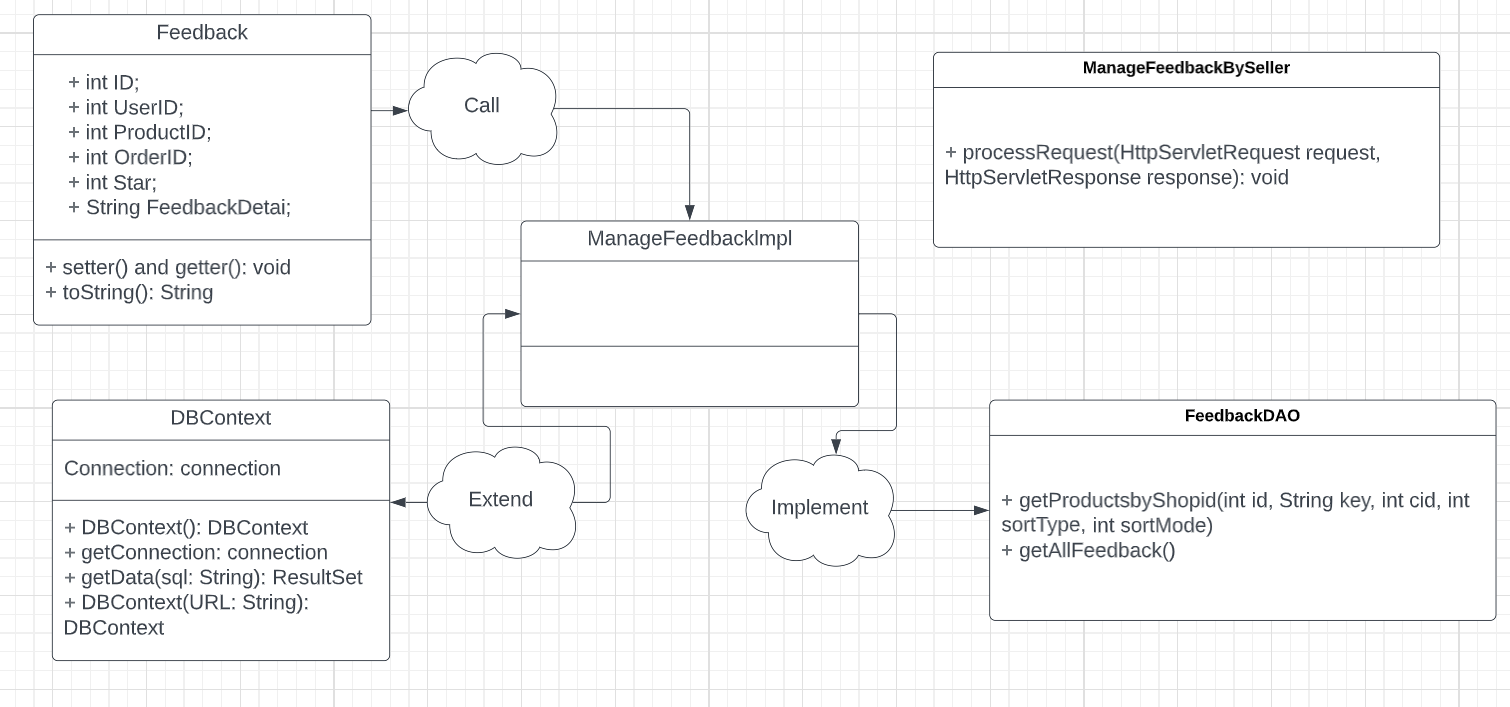
### d. Database queries

**- DELETE FROM [Blog] WHERE ID = ?**

**- DELETE FROM [BlogDetail] WHERE BlogDetailID = ?**

## 55. View all Feedback (Seller)

### a. Class Diagram



*Figure 2.55.1 Class Diagram*

### b. Class Specification

#### ManageFeedbackServlet

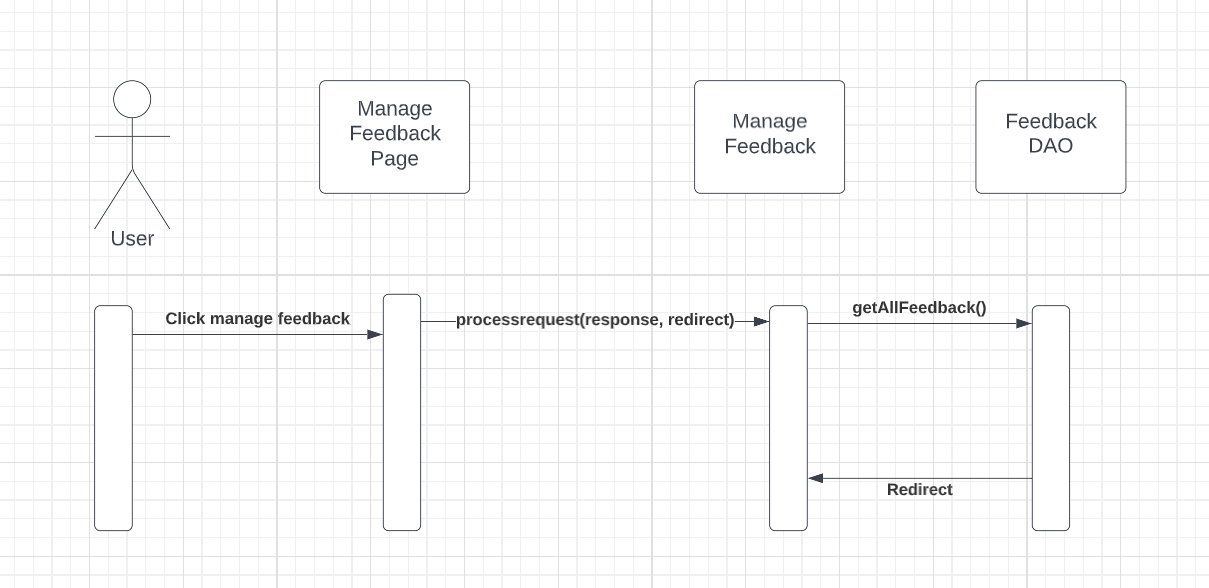
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Retrieve data from the database and put it on the website* |

#### ProductDAO, FeedbackDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public List<Products> getProductsbyShopid(int id, String key, int cid, int sortType, int sortMode) {}* | *Get all product data of a shop* |
| *02* | *public List<Feedback> getAllFeedback(){}* | *Get all feedback data* |

### c. Sequence Diagram(s)



*Figure 2.55.2 Sequence Diagram*

### d. Database queries

**-** SELECT \* FROM (SELECT p.ProductID,MIN(p.ProductName) AS ProductName,

MIN(p.Description) AS Description, MIN(p.OriginalPrice) AS OriginalPrice,

MIN(p.SellPrice) AS SellPrice, MIN(p.SalePercent) AS SalePercent,

MIN(p.SubCategoryID) AS SubCategoryID, MIN(p.SellerID) AS SellerID,

MIN(p.Amount) AS Amount, MIN(p.StatusID) AS StatusID, MIN(p.BrandID) AS BrandID,

MIN(ProI.ProductImgURL) AS ProductImgURL, MIN(Sub.CategoryID) AS CategoryID

FROM dbo.Product p JOIN dbo.ProductImg ProI ON ProI.ProductID = p.ProductID

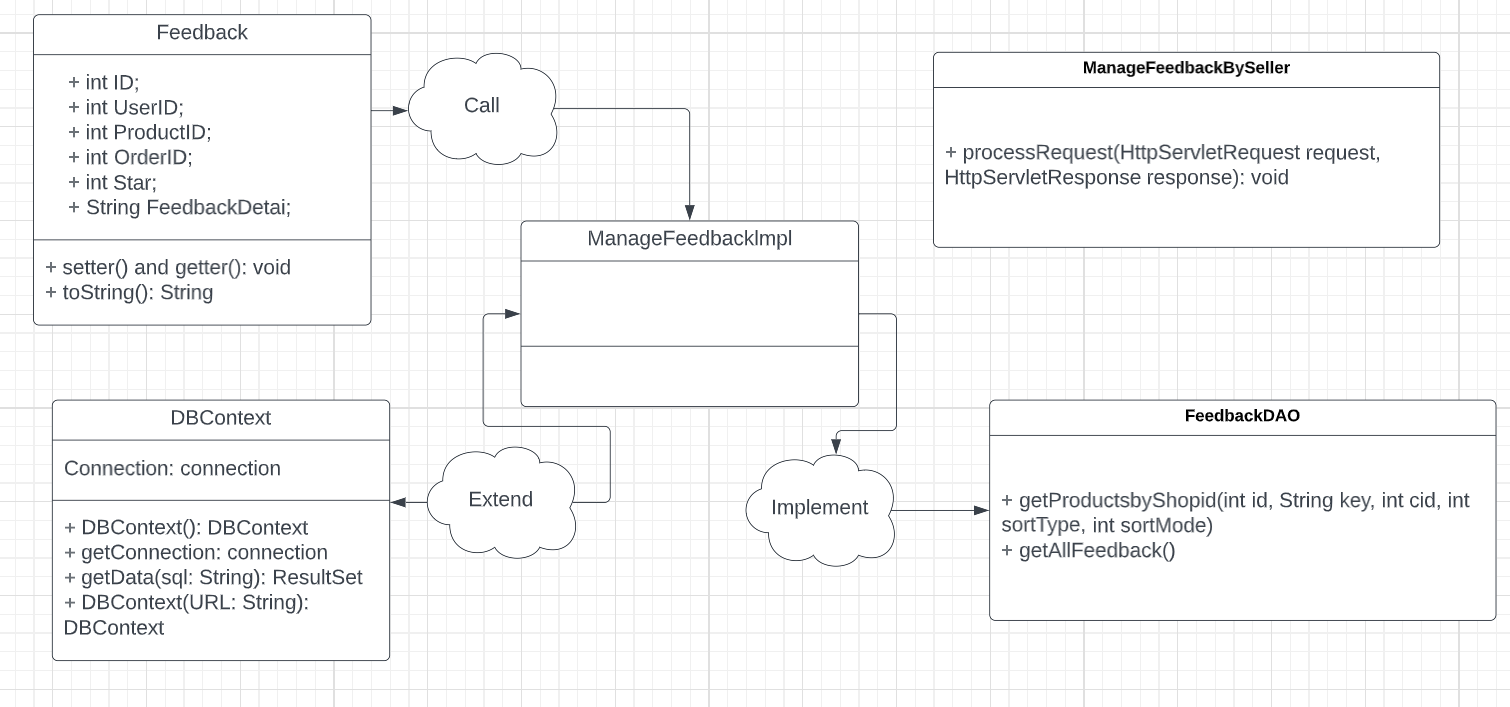
JOIN dbo.SubCategory Sub ON Sub.SubCategoryID = p.SubCategoryID

GROUP BY p.ProductID ) t where t.ProductID = ?

- **SELECT ID,UserID,ProductID,OrderID,Star,FeedbackDetail FROM Feedback**

## 56. Reply Feedback

### a. Class Diagram



*Figure 2.56.1 Class Diagram*

### b. Class Specification

#### ReplyFeedback

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *doGet()* | *Get the product's data and put it on the website* |
| *02* | *doPost()* | *Insert reply feedback into the Feedback\_reply table in the database* |

#### FeedbackDAO

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *public void insertReply(FeedbackReply fr){}* | *Insert reply feedback into the Feedback\_reply table in the database* |

### c. Sequence Diagram(s)



*Figure 2.56.2 Sequence Diagram*

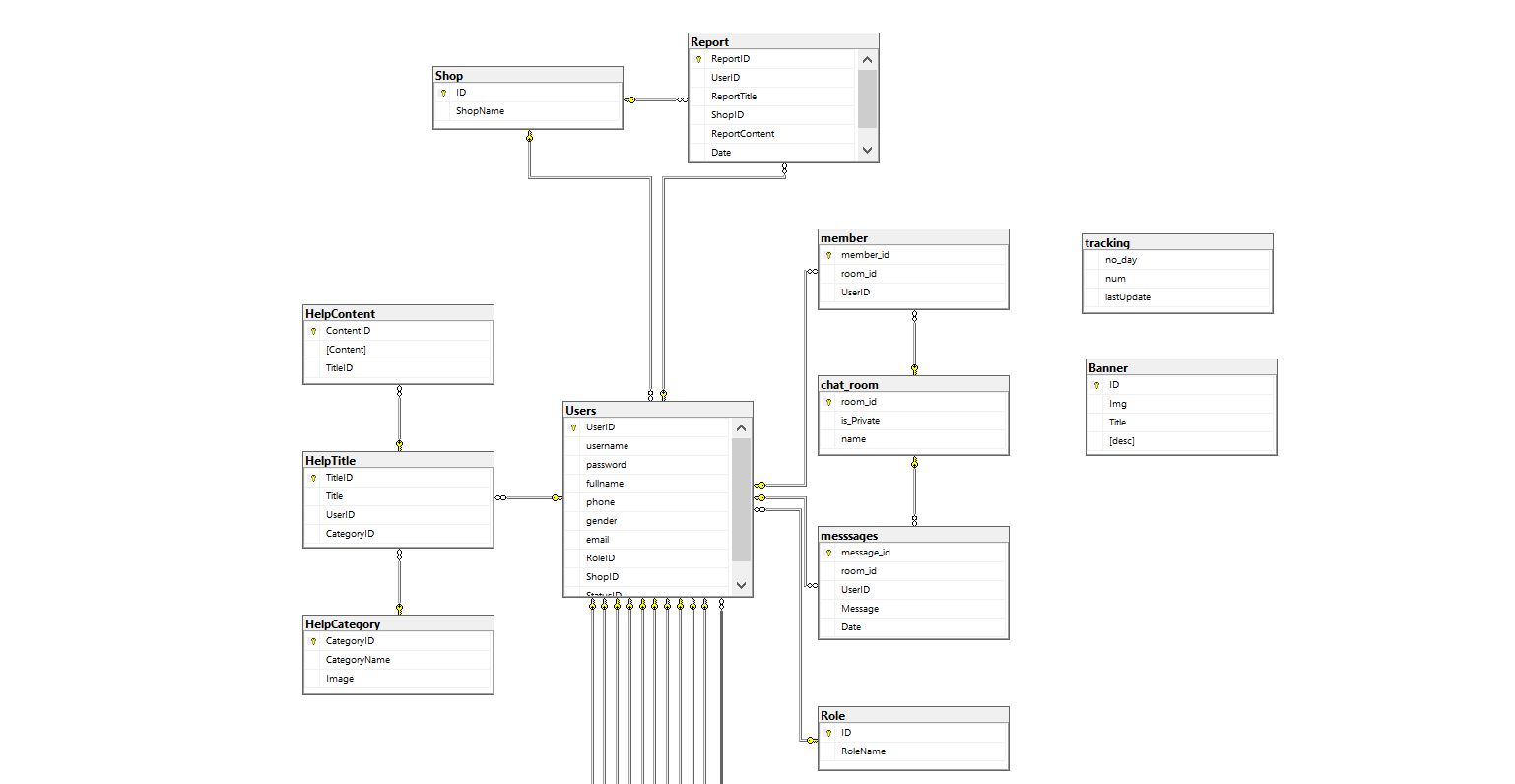
### d. Database queries

**- INSERT INTO [dbo].[Feedback\_Replies] ([FeedbackID], [UserID], [ProductID], [RepliesText])**

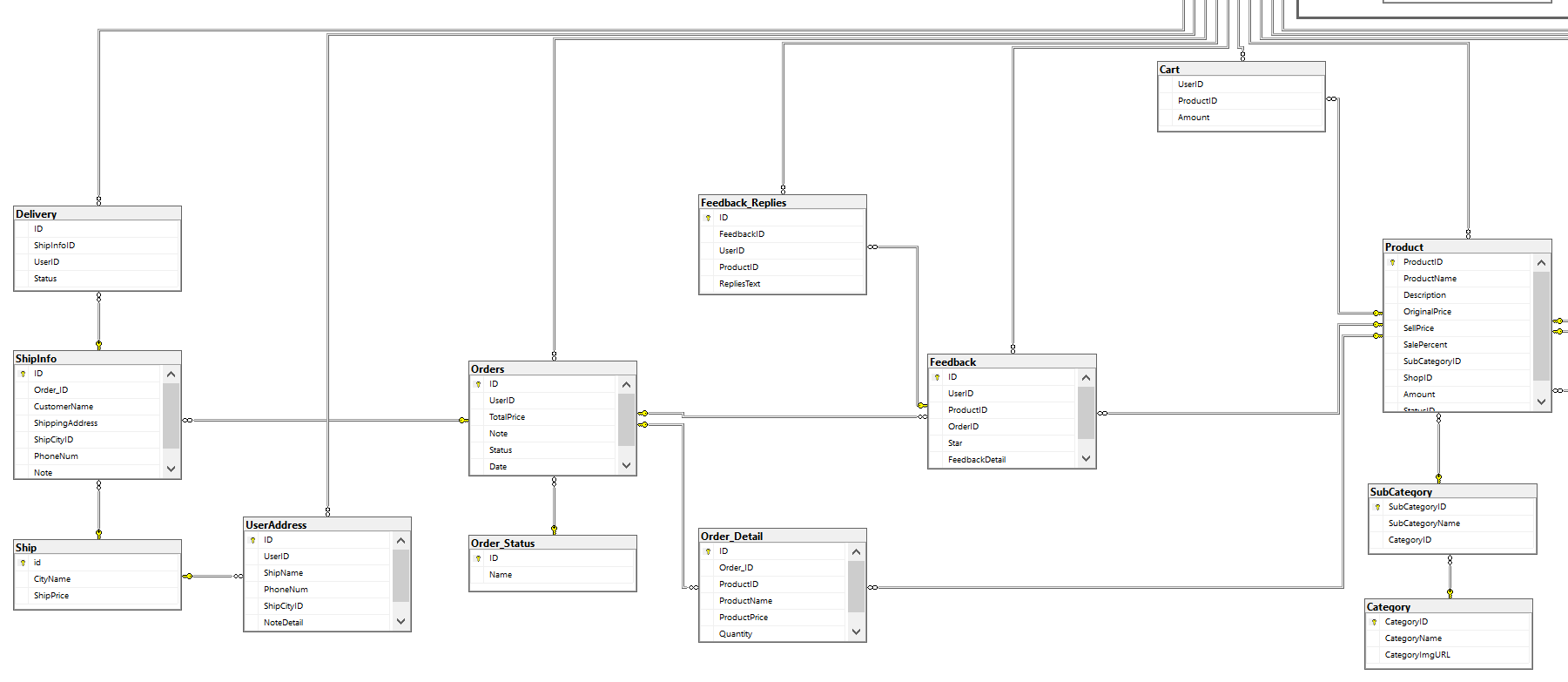
**VALUES (?,?,?,?)**

# III. Database Design

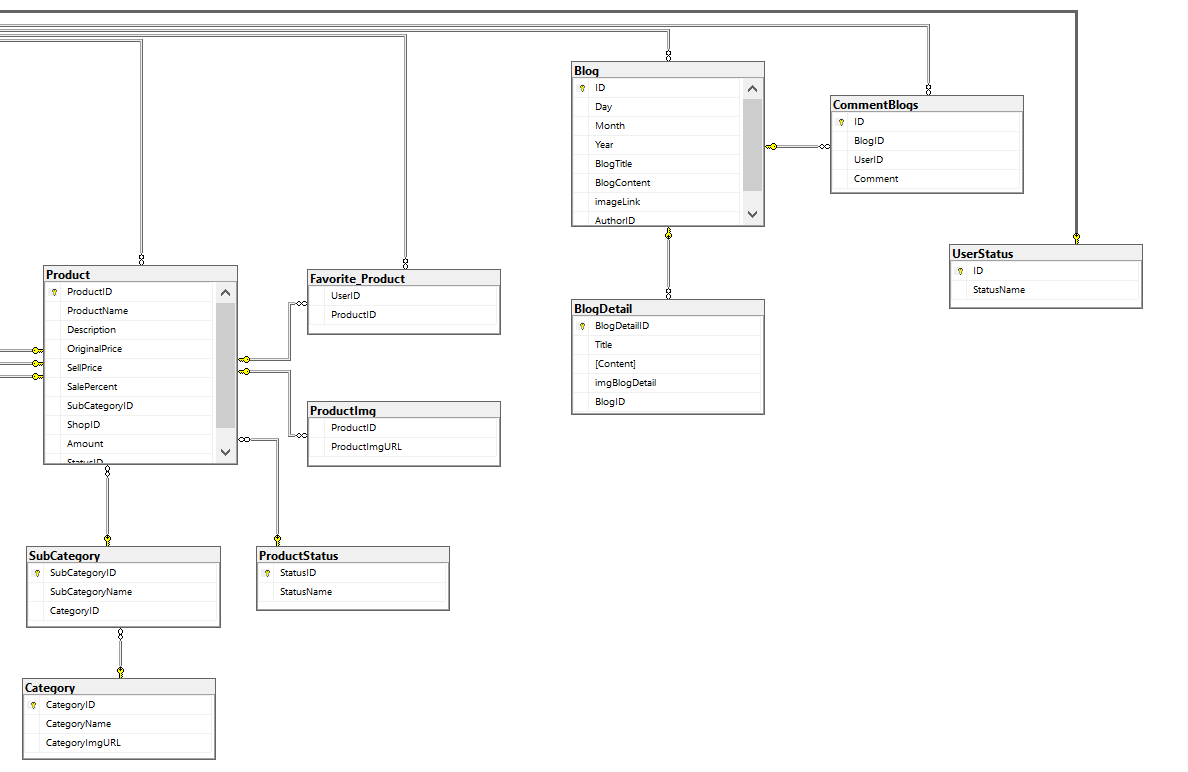
## 1. Database Schema



*Figure 3.1.1 Database diagram*

**

*Figure 3.1.2 Database diagram*

**

*Figure 3.1.3 Database diagram*

## 2. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 1 | Shop | Shop information  - Primary keys: ID  - Foreign keys: none |
| 2 | UserStatus | Account activation status  - Primary keys: ID  - Foreign keys: none |
| 3 | Role | Role of the user (Admin, Seller, Customer, Shipper)  - Primary keys: ID  - Foreign keys: none |
| 4 | Users | User  - Primary keys: UserID  - Foreign keys: StatusID, ShopID, RoleID |
| 5 | Ship | <Description of the table>  - Primary keys: id  - Foreign keys: none |
| 6 | UserAddress | Address for each user  - Primary keys: ID  - Foreign keys: UserID, ShipCityID |
| 7 | Category | <Description of the table>  - Primary keys: CategoryID  - Foreign keys: none |
| 8 | SubCategory | Sub Categories  - Primary keys: SubCategoryID  - Foreign keys: CategoryID |
| 9 | ProductStatus | Status of a product  - Primary keys: StatusID  - Foreign keys: none |
| 10 | Product | Product data  - Primary keys: ProductID  - Foreign keys: SubCategoryID, SellerID, StatusID, BrandID |
| 11 | ProductImg | Image of products  - Primary keys: none  - Foreign keys: ProductID |
| 12 | Cart | Cart info for each user  - Primary keys: UserID  - Foreign keys: UserID, ProductID |
| 13 | Favorite\_Product | List of favorite products of each person  - Primary keys: UserID  - Foreign keys: UserID, ProductID |
| 14 | Order\_Status | Status of a order  - Primary keys: ID  - Foreign keys: none |
| 15 | Order | Order Data  - Primary keys: ID  - Foreign keys: UserID, Status |
| 16 | Order\_Detail | <Description of the table>  - Primary keys: ID  - Foreign keys: Order\_ID, ProductID |
| 17 | Report | <Description of the table>  - Primary keys: ReportID  - Foreign keys: UserID, ShopID |
| 18 | ShipInfo | Shipping info for each order  - Primary keys: ID  - Foreign keys: Order\_ID, ShipCityID |
| 19 | Feedback | Feedback of a product order  - Primary keys: ID  - Foreign keys: UserID, ProductID, OrderID |
| 20 | Feedback\_Replies | Replies of feedbacks  - Primary keys: ID  - Foreign keys: FeedbackID, UserID |
| 21 | Blog | Blog content + image  - Primary keys: ID  - Foreign keys: AuthorID |
| 22 | CommentBlogs | User comments on Blog  - Primary keys: ID  - Foreign keys: BlogID, UserID |
| 23 | BlogDetail | Detailed content of the blog + image  -Primary key: BlogDetailID  -Foreign key: BlogID |
| 24 | HelpCategory | <Description of the table>  - Primary keys: CategoryID  - Foreign keys: none |
| 25 | HelpTitle | <Description of the table>  - Primary keys: TitleID  - Foreign keys: UserID, CategoryID |
| 26 | HelpContent | <Description of the table>  - Primary keys: ContentID  - Foreign keys: TitleID |
| 27 | Delivery | Delevery methos  - Primary keys: ID  - Foreign keys: ShipInfo, UserID |
| 28 | chat\_room | <Description of the table>  - Primary keys: room\_id  - Forrign keys: None |
| 29 | messages | <Description of the table>  - Primary keys: message\_id  - Forrign keys: room\_id, UserID |
| 30 | member | <Description of the table>  - Primary keys: member\_id  - Forrign keys: room\_id, UserID |
| 31 | Tracking | <Description of the table>  - Primary keys: no\_day  - Forrign keys: None |