

**online shop**

**Software Design Document**

– Hanoi, May 2021–

**Table of Contents**

[I. Overview 7](#_Toc77690320)

[1. Code Packages/Namespaces 7](#_Toc77690321)

[2. Database Schema 8](#_Toc77690322)

[II. Code Designs 9](#_Toc77690323)

[1. Feedback 9](#_Toc77690324)

[a. Class Diagram 9](#_Toc77690325)

[b. Class Specifications 10](#_Toc77690326)

[c. Sequence Diagram(s) 13](#_Toc77690327)

[d. Database queries 14](#_Toc77690328)

[2. Feedback List 15](#_Toc77690329)

[a. Class Diagram 15](#_Toc77690330)

[b. Class Specifications 15](#_Toc77690331)

[c. Sequence Diagram(s) 19](#_Toc77690332)

[d. Database queries 19](#_Toc77690333)

[3. Feedback Details 20](#_Toc77690334)

[a. Class Diagram 20](#_Toc77690335)

[b. Class Specifications 20](#_Toc77690336)

[c. Sequence Diagram(s) 24](#_Toc77690337)

[d. Database queries 24](#_Toc77690338)

[4. User Register 25](#_Toc77690339)

[a. Class Diagram 25](#_Toc77690340)

[b. Class Specifications 25](#_Toc77690341)

[c. Sequence Diagram(s) 27](#_Toc77690342)

[d. Database queries 27](#_Toc77690343)

[5. Order Information 28](#_Toc77690344)

[a. Class Diagram 28](#_Toc77690345)

[b. Class Specifications 28](#_Toc77690346)

[c. Sequence Diagram(s) 31](#_Toc77690347)

[d. Database queries 31](#_Toc77690348)

[6. Product Details 32](#_Toc77690349)

[a. Class Diagram 32](#_Toc77690350)

[b. Class Specification 32](#_Toc77690351)

[c. Sequence Diagram 34](#_Toc77690352)

[d. Database queries 35](#_Toc77690353)

[7. Cart Details 36](#_Toc77690354)

[a. Class Diagram 36](#_Toc77690355)

[b. Class Specifications 36](#_Toc77690356)

[c. Sequence Diagram(s) 38](#_Toc77690357)

[d. Database queries 38](#_Toc77690358)

[8. Cart Completion 39](#_Toc77690359)

[a. Class Diagram 39](#_Toc77690360)

[b. Class Specifications 39](#_Toc77690361)

[9. Change Password 43](#_Toc77690362)

[a. Class Diagram 43](#_Toc77690363)

[b. Class Specifications 43](#_Toc77690364)

[d. Database queries 46](#_Toc77690365)

[10. User Profile 47](#_Toc77690366)

[b. Class Specifications 47](#_Toc77690367)

[c. Sequence Diagram(s) 49](#_Toc77690368)

[d. Database queries 49](#_Toc77690369)

[11. Home Page 50](#_Toc77690370)

[a. Class Diagram 50](#_Toc77690371)

[b. Class Specifications 50](#_Toc77690372)

[c. Sequence Diagram(s) 53](#_Toc77690373)

[d. Database queries 53](#_Toc77690374)

[12. Product List 54](#_Toc77690375)

[a. Class Diagram 54](#_Toc77690376)

[b. Class Specifications 54](#_Toc77690377)

[c. Sequence Diagram(s) 57](#_Toc77690378)

[d. Database queries 57](#_Toc77690379)

[13. User Login 58](#_Toc77690380)

[a. Class Diagram 58](#_Toc77690381)

[b. Class Specifications 58](#_Toc77690382)

[c. Sequence Diagram(s) 60](#_Toc77690383)

[d. Database queries 60](#_Toc77690384)

[14. Reset Password 61](#_Toc77690385)

[a. Class Diagram 61](#_Toc77690386)

[b. Class Specifications 61](#_Toc77690387)

[c. Sequence Diagram(s) 63](#_Toc77690388)

[d. Database queries 63](#_Toc77690389)

[15. Product List (Marketing) 64](#_Toc77690390)

[a. Class Diagram 64](#_Toc77690391)

[b. Class Specifications 64](#_Toc77690392)

[c. Sequence Diagram(s) 67](#_Toc77690393)

[d. Database queries 67](#_Toc77690394)

[16. My Order History 68](#_Toc77690395)

[a. Class Diagram 68](#_Toc77690396)

[b. Class Specifications 68](#_Toc77690397)

[c. Sequence Diagram(s) 69](#_Toc77690398)

[d. Database queries 69](#_Toc77690399)

[17. My order details 70](#_Toc77690400)

[a. Class Diagram 70](#_Toc77690401)

[b. Class Specifications 70](#_Toc77690402)

[c. Sequence Diagram(s) 71](#_Toc77690403)

[d. Database queries 71](#_Toc77690404)

[18. Notifications 72](#_Toc77690405)

[a. Class Diagram 72](#_Toc77690406)

[b. Class Specifications 72](#_Toc77690407)

[c. Sequence Diagram(s) 73](#_Toc77690408)

[d. Database queries 73](#_Toc77690409)

[19. View All Invocies 74](#_Toc77690410)

[a. Class Diagram 74](#_Toc77690411)

[b. Class Specifications 74](#_Toc77690412)

[c. Sequence Diagram(s) 75](#_Toc77690413)

[d. Database queries 75](#_Toc77690414)

[20. View Invoice Detail 75](#_Toc77690415)

[a. Class Diagram 75](#_Toc77690416)

[b. Class Specifications 75](#_Toc77690417)

[c. Sequence Diagram(s) 78](#_Toc77690418)

[d. Database queries 78](#_Toc77690419)

[21. Compare 80](#_Toc77690420)

[a. Class Diagram 80](#_Toc77690421)

[b. Class Specifications 80](#_Toc77690422)

[c. Sequence Diagram(s) 82](#_Toc77690423)

[d. Database queries 82](#_Toc77690424)

[22. CompareFinal 83](#_Toc77690425)

[a. Class Diagram 83](#_Toc77690426)

[b. Class Specifications 83](#_Toc77690427)

[c. Sequence Diagram(s) 84](#_Toc77690428)

[d. Database queries 84](#_Toc77690429)

[24. User List 85](#_Toc77690430)

[a. Class Diagram 85](#_Toc77690431)

[b. Class Specifications 85](#_Toc77690432)

[c. Sequence Diagram(s) 87](#_Toc77690433)

[d. Database queries 87](#_Toc77690434)

[25. Blog List 87](#_Toc77690435)

[a. Class Diagram 87](#_Toc77690436)

[b. Class Specifications 87](#_Toc77690437)

[c. Sequence Diagram(s) 89](#_Toc77690438)

[d. Database queries 89](#_Toc77690439)

[26. Blog Detail 90](#_Toc77690440)

[a. Class Diagram 90](#_Toc77690441)

[b. Class Specifications 90](#_Toc77690442)

[c. Sequence Diagram(s) 91](#_Toc77690443)

[d. Database queries 91](#_Toc77690444)

[27. Post List 92](#_Toc77690445)

[a. Class Diagram 92](#_Toc77690446)

[b. Class Specifications 92](#_Toc77690447)

[c. Sequence Diagram(s) 93](#_Toc77690448)

[d. Database queries 93](#_Toc77690449)

[28. Post Detail 94](#_Toc77690450)

[a. Class Diagram 94](#_Toc77690451)

[b. Class Specifications 94](#_Toc77690452)

[c. Sequence Diagram(s) 96](#_Toc77690453)

[d. Database queries 96](#_Toc77690454)

[III. Database Tables 96](#_Toc77690455)

[1. User Status 96](#_Toc77690456)

[2. Users 96](#_Toc77690457)

[3. Category 97](#_Toc77690458)

[4. Product Status 97](#_Toc77690459)

[5. Manufacturer 97](#_Toc77690460)

[6. Product 97](#_Toc77690461)

[7. Information 98](#_Toc77690462)

[8. Cart 98](#_Toc77690463)

[9. Order\_Status 98](#_Toc77690464)

[10. Orders 98](#_Toc77690465)

[11. Order\_Detail 99](#_Toc77690466)

[12. Ship 99](#_Toc77690467)

[13. UserAddress 99](#_Toc77690468)

[14. ShipInfo 99](#_Toc77690469)

[15. Notifications 100](#_Toc77690470)

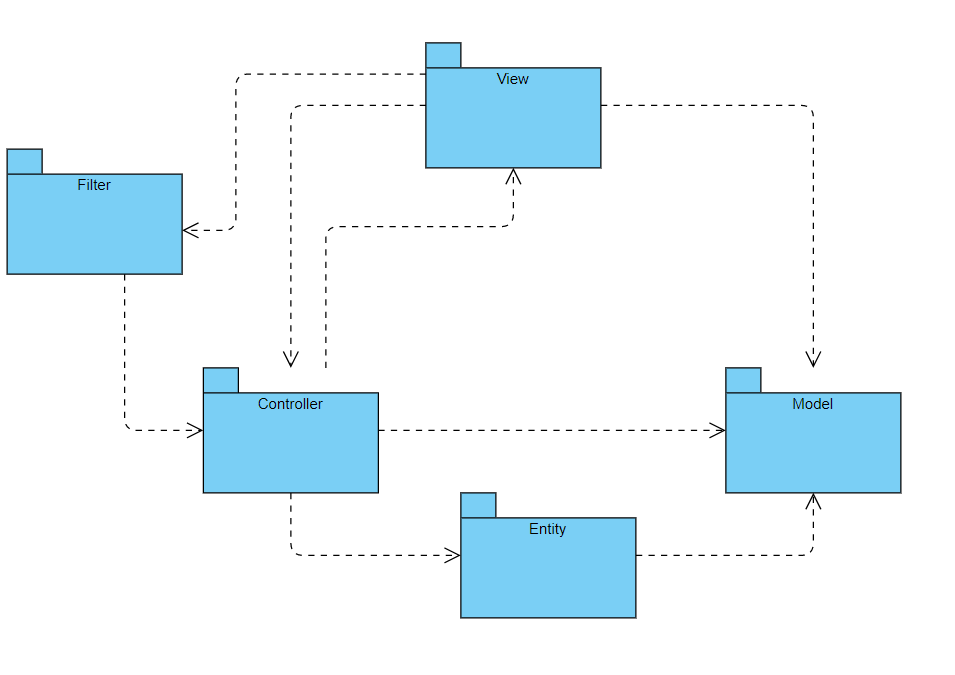
[16. Feedback 100](#_Toc77690471)

[17. Feedback\_Replies 100](#_Toc77690472)

[18. Blog 100](#_Toc77690473)

# I. Overview

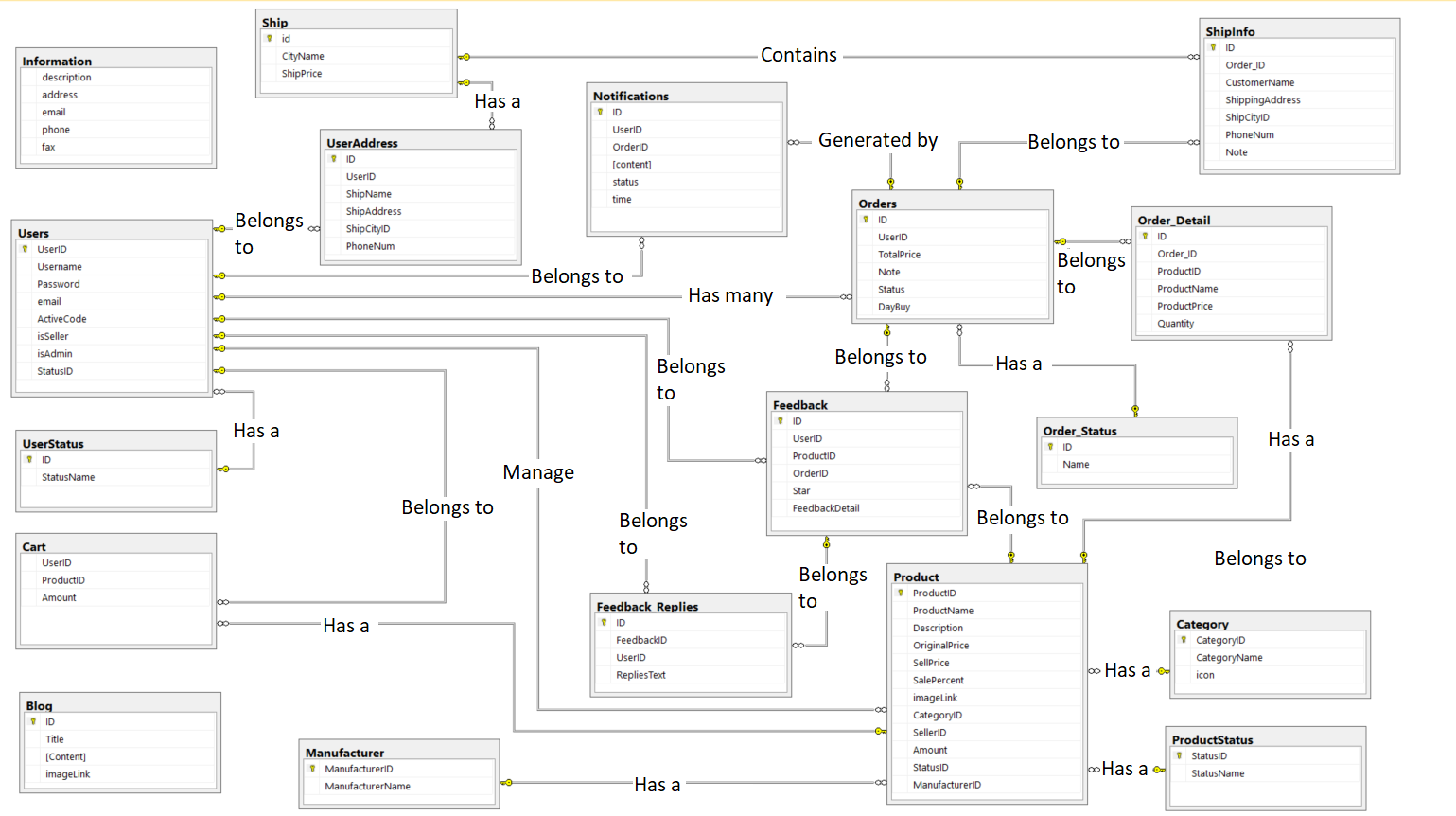
## 1. Code Packages/Namespaces



***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* | *DAO (Model)* | *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* |

## 2. Database Schema



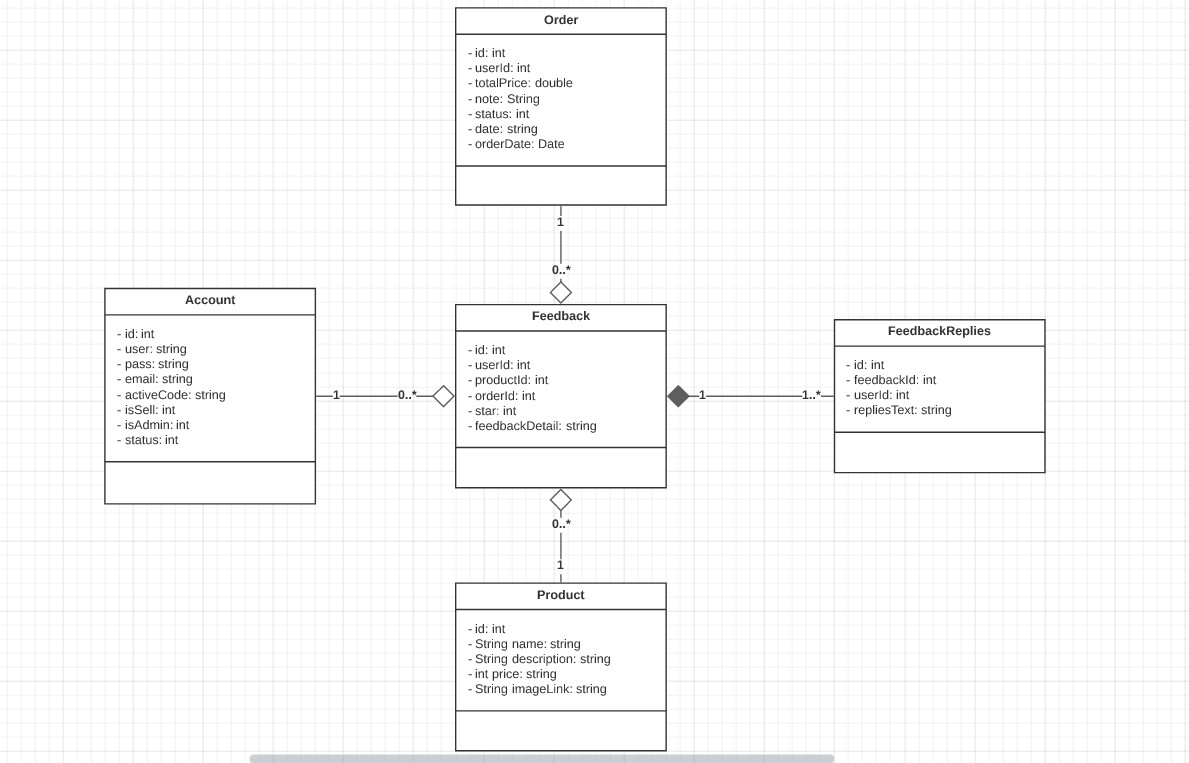
***Table descriptions & package class naming conventions are as below***

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *User Status* | *Store the status of an Account* |
| *02* | *Users* | *Store the information of all users (including customers, sellers and admin)*  *- Primary keys: UserID*  *- Foreign keys: StatusID* |
| *03* | *Category* | *Store the information of Categories of products*  *- Primary keys: CategoryID* |
| *04* | *Product Status* | *Store the status of a Product (out of stock or not)*  *- Primary keys: StatusID* |
| *05* | *Manufacturer* | *Store the information of the Manufacturer of a Product*  *- Primary keys: ManufacturerID* |
| *06* | *Product* | *Store the information of all products*  *- Primary keys: ProductID*  *- Foreign keys: ManufacturerID, StatusID, SellerID, CategoryID* |
| *07* | *Information* | *Store some information of the shop* |
| *08* | *Cart* | *Store information of the Cart of each Customer*  *- Foreign keys: UserID, ProductID* |
| *09* | *Order\_Status* | *Store the status of an Order*  *- Primary keys: ID* |
| *10* | *Orders* | *Store information of all orders*  *- Primary keys: ID*  *- Foreigns keys: UserID* |
| *11* | *Order\_Detail* | *Store products information of all orders*  *- Primary keys: ID*  *- Foreigns keys: Order\_ID, ProductID* |
| *12* | *Ship* | *Store the ship Price of each City*  *- Primary keys: ID* |
| *13* | *UserAddress* | *Store the Shipping Address of Customers*  *- Primary keys: ID*  *- Foreign keys: UserID, ShipCityID* |
| *14* | *ShipInfo* | *Store the ship info of all orders*  *- Primary keys: ID*  *- Foreign keys: UserID, ShipCityID* |
| *15* | *Notifications* | *Store the notifications of a Customer order*  *- Primary keys: ID*  *- Foreign keys: UserID, OrderID* |
| *16* | *Feedback* | *Store all information of Customer’s feedbacks on a Product*  *- Primary keys: ID*  *- Foreign keys: UserID, ProductID, OrderID* |
| *17* | *Feedback*  *\_Replies* | *Store all replies of each feedback*  *- Primary keys: ID*  *- Foreign keys: UserID, FeedbackID* |
| *18* | *Blog* | *Store all blog posts of the system*  *- Primary keys: ID*  *- Foreign keys: SellerID* |

# II. Code Designs

## 1. Feedback

### a. Class Diagram



### b. Class Specifications

#### Feedback Class: This class stores information of a Feedback

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Product ID of a Feedback object |
| 6 | setProductId() | Set the Product ID of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | getProduct() | Get the Product of a Feedback object |
| 14 | setProduct() | Set the Product of a Feedback object |
| 15 | getUser() | Get the User of a Feedback object |
| 16 | setUser() | Set the User of a Feedback object |
| 17 | getOrder() | Get the Order of a Feedback object |
| 18 | setOrder() | Set the Order of a Feedback object |
| 19 | toString() | Convert all information of a Feedback object to a string |

#### FeedbackReplies Class: This class stores information of a Feedback Replies

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of a FeedbackReplies object |
| 2 | setId() | Set the ID of a FeedbackReplies object |
| 3 | getFeedbackId() | Get the Feedback ID of a FeedbackReplies object |
| 4 | setFeedbackId() | Set the Feedback ID of a FeedbackReplies object |
| 5 | getUserId() | Get the User ID of a FeedbackReplies object |
| 6 | setUserId() | Set the User ID of a FeedbackReplies object |
| 7 | getRepliesText() | Get the Replies Text of a FeedbackReplies object |
| 8 | setRepliesText() | Set the Replies Text of a FeedbackReplies object |
| 9 | toString() | Convert all information of a FeedbackReplies object to a string |

#### Order Class: This class stores information of an Order

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Order object |
| 2 | setId() | Set the ID of an Order object |
| 3 | getUserId() | Get the User ID of an Order object |
| 4 | setUserId() | Set the User ID of an Order object |
| 5 | getTotalPrice() | Get the Subtotal of an Order object |
| 6 | setTotalPrice () | Set the Subtotal of an Order object |
| 7 | getNote () | Get the Note of an Order object |
| 8 | setNote() | Set the Note of an Order object |
| 9 | getStatus () | Get the Status of an Order object |
| 10 | setStatus() | Set the Status of an Order object |
| 11 | getDate() | Get the Order Date of an Order object |
| 12 | setDate() | Set the Order Date of an Order object |

#### Account Class: This class stores information of a Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getPass() | Get the Password of an Account |
| 6 | setPass() | Set the Password of an Account |
| 7 | getEmail() | Get the Email of an Account |
| 8 | setEmail() | Set the Email of an Account |
| 9 | getIsSell() | Check if an Account has the seller role |
| 10 | setIsSell() | Add/remove seller role to/from an Account |
| 11 | getIsAdmin() | Check if an Account has the admin role |
| 12 | setIsAdmin() | Add/remove admin role to/from an Account |
| 13 | getStatus() | Get the Status of an Account |
| 14 | setStatus() | Set the Status of an Account |
| 15 | toString() | Convert all information of an Account to a string |

#### FeedbackDAO Class: This class performs CRUD actions to Feedback table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getFeedbacksBy  ProductId() | Get a list of feedback by product id |
| 2 | getFeedbacks() | Get a list all feedback from the database |
| 3 | getFeedbacksById() | Get a feedback by id |
| 4 | getFeedbacks  ByUserId() | Get a list of feedback by user id |
| 5 | getFeedbacksBy  UserIdAndProductId() | Get a list of feedback by user id and product id |
| 6 | getFeedbacksBy  OrderId() | Get a list of feedback by order id |
| 7 | addFeedback() | Add a feedback to database |
| 8 | countTotalFeedback() | Count total number of feedback in the database |

#### FeedbackRepliesDAO Class: This class performs CRUD actions to Feedback\_Replies table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getFeedbacksBy  FeedbackId() | Get a list of replies by feedback id |
| 2 | getFeedbacksBy  UserId () | Get a list of replies by user id |
| 3 | addReplies() | Add a replies to a feedback to database |

#### UserDAO Class: This class performs CRUD actions to Users table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | getSellerIdOfAn  Order() | Get the Seller ID of an order |
| 6 | deleteAccount() | Delete an account from the database |
| 7 | editAccount() | Update the information of an account |
| 8 | updatePassword() | Update the password of a user |
| 9 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 10 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 11 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 12 | updateStatus() | Update status of an account and return true if update successful |
| 13 | countAllAccount() | Return the number of account in the database |
| 14 | checkForget  Password() | Get username and email of a user that forget password |
| 15 | searchAccountIn  Manager() | Get accounts by username |

#### OrderDAO Class: This class performs CRUD actions to Orders table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | Adding a new order to database, return the order Id |
| 4 | packaging() | Change the status of an specific order by the order id to packaging |
| 5 | delivering() | Change the status of an specific order by the order id to delivering |
| 6 | canceled() | Change the status of an specific order by the order id to canceled |
| 7 | finished() | Change the status of an specific order by the order id to finished |
| 8 | countOrders() | Count all the number of orders in database and return integer number |
| 9 | getOrderByOrderID() | Get a particular order by its id and return Order |
| 10 | getNewestOrderID() | Get the order id of the newest order |
| 11 | getOrderByUserID() | Get list of orders of a specific user and return List Order |
| 12 | getStatusOfAn  Order() | Get the status of an order by order id |
| 13 | getCompletedOrder  ByUserID() | Get all completed orders of an User by User ID |
| 14 | getRecentOrder() | Get all recent orders |
| 15 | Delete() | Delete a specific order by the order id |

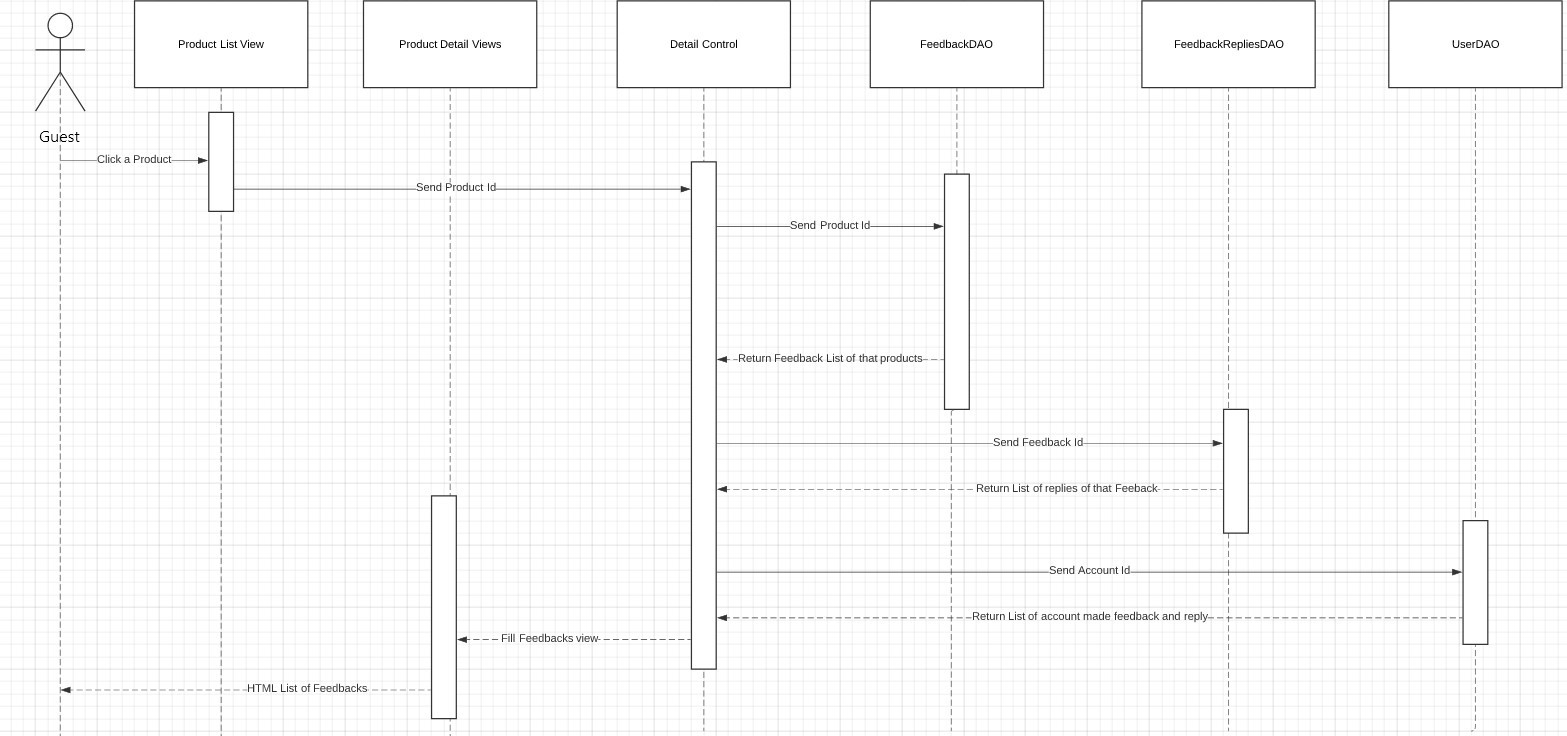
#### OrderDetailDAO Class: This class performs CRUD actions to Order\_Detail table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | addManyOrder  Details() | Adding many oder\_detail to a specific order by the order ID |
| 2 | getOrderDetailBy  OrderID() | getting detailed orders by the order id return OrderDetail |
| 3 | getOrderDetailBy  OrderIDAnd  ProductId() | Getting an order details object by the order id and product id |
| 4 | deleteAmount() | Remove x number of items from a Product by Product ID |

#### DetailControl Class: This class process request for detail of a product

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Return Detail of Product |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

SELECT \* FROM Feedback WHERE ProductID = ?

SELECT o.ID,o.UserId,o.TotalPrice, o.Note, o.DayBuy, os.Name FROM

Orders o INNER JOIN Order\_Status os

ON o.Status = os.ID

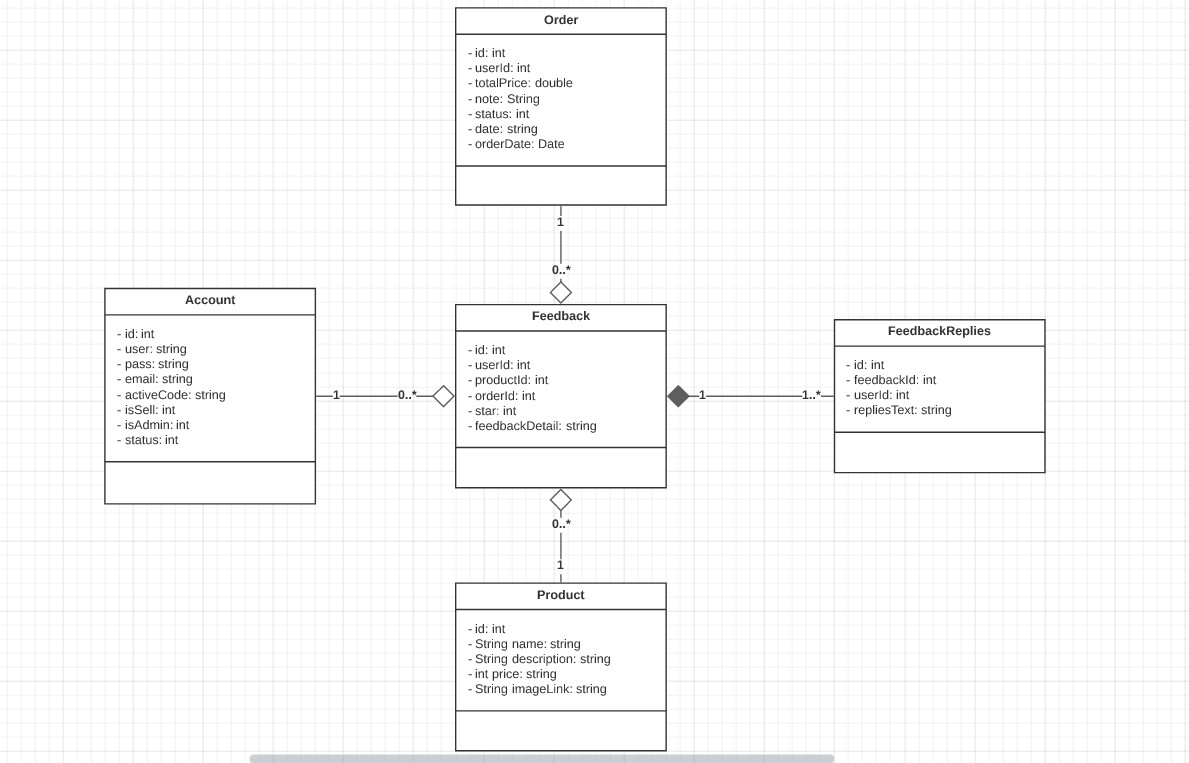
WHERE o.ID = ?

SELECT \* FROM Feedback\_Replies WHERE FeedbackID = ?

SELECT \* FROM Users WHERE UserID = ?

## 2. Feedback List

### a. Class Diagram



### b. Class Specifications

#### Feedback Class: This class stores information of a Feedback

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Product ID of a Feedback object |
| 6 | setProductId() | Set the Product ID of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | getProduct() | Get the Product of a Feedback object |
| 14 | setProduct() | Set the Product of a Feedback object |
| 15 | getUser() | Get the User of a Feedback object |
| 16 | setUser() | Set the User of a Feedback object |
| 17 | getOrder() | Get the Order of a Feedback object |
| 18 | setOrder() | Set the Order of a Feedback object |
| 19 | toString() | Convert all information of a Feedback object to a string |

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### Account Class: This class stores information of a Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getPass() | Get the Password of an Account |
| 6 | setPass() | Set the Password of an Account |
| 7 | getEmail() | Get the Email of an Account |
| 8 | setEmail() | Set the Email of an Account |
| 9 | getIsSell() | Check if an Account has the seller role |
| 10 | setIsSell() | Add/remove seller role to/from an Account |
| 11 | getIsAdmin() | Check if an Account has the admin role |
| 12 | setIsAdmin() | Add/remove admin role to/from an Account |
| 13 | getStatus() | Get the Status of an Account |
| 14 | setStatus() | Set the Status of an Account |
| 15 | toString() | Convert all information of an Account to a string |

#### FeedbackDAO Class: This class performs CRUD actions to Feedback table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getFeedbacksBy  ProductId() | Get a list of feedback by product id |
| 2 | getFeedbacks() | Get a list all feedback from the database |
| 3 | getFeedbacksById() | Get a feedback by id |
| 4 | getFeedbacks  ByUserId() | Get a list of feedback by user id |
| 5 | getFeedbacksBy  UserIdAndProductId() | Get a list of feedback by user id and product id |
| 6 | getFeedbacksBy  OrderId() | Get a list of feedback by order id |
| 7 | addFeedback() | Add a feedback to database |
| 8 | countTotalFeedback() | Count total number of feedback in the database |

#### ProductDAO Class: This class performs CRUD actions to Product table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllProduct() | Get all products from database |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | getProductsByCate  Id() | return the list of products of a particular category |
| 6 | Edit() | Edit the information of a particular product, select by product ID |
| 7 | Delete() | Delete a product, select by its ID |
| 8 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 9 | countProduct() | return the total number of products |
| 10 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 11 | pagingByCategory() | categorizing products and return list of products with same category |
| 12 | pagingManager  Product() | Paging for manage product |
| 13 | countProductBy  Category() | count total product |
| 14 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 15 | getProductByID() | Get data of Product Table searched by UserID |
| 16 | getCateIdOfProductByID() | Get Category ID of a Product by Product ID |
| 17 | getProductDetailBy  ID() | Get Product for Detail |
| 18 | getProductForManager() | Get a particular product and return a product |
| 19 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 20 | searchProductBy  NameAndCateId() | Search for a particular product by filling its name and Category ID |
| 21 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 22 | top3MostSell() | Select top 3 most sold products and return product |
| 23 | top3LeastSell() | Select top 3 least sold products and return product |
| 24 | countProductBy  Category() | Count the number of products by category |
| 25 | getRelatedProduct() | Get 3 related products to a Category by Category ID |

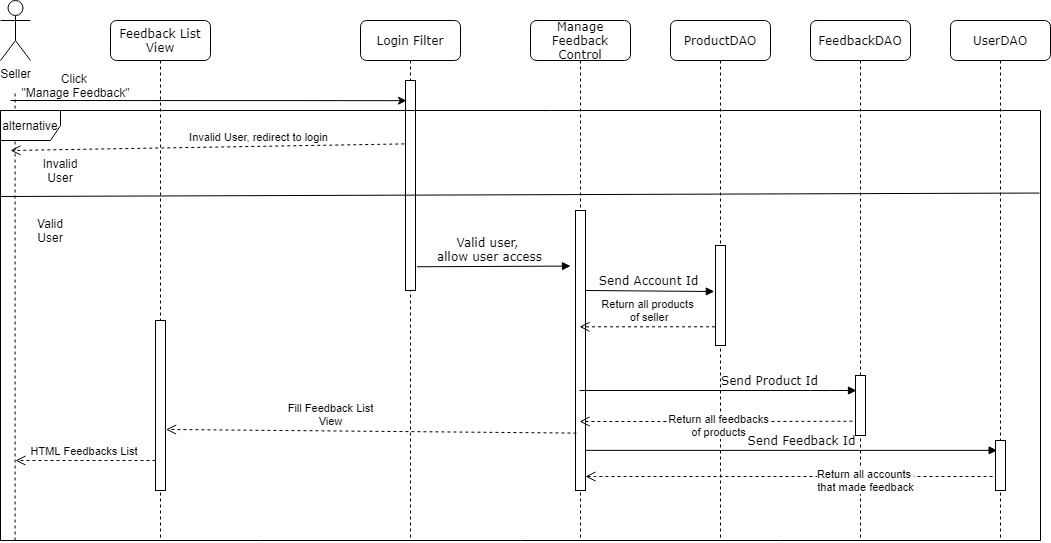
#### UserDAO Class: This class performs CRUD actions to Users table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | getSellerIdOfAn  Order() | Get the Seller ID of an order |
| 6 | deleteAccount() | Delete an account from the database |
| 7 | editAccount() | Update the information of an account |
| 8 | updatePassword() | Update the password of a user |
| 9 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 10 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 11 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 12 | updateStatus() | Update status of an account and return true if update successful |
| 13 | countAllAccount() | Return the number of account in the database |
| 14 | checkForget  Password() | Get username and email of a user that forget password |
| 15 | searchAccountIn  Manager() | Get accounts by username |

#### ManageFeedbackControl Class: This class process request for list of feedbacks for a seller

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Return List of feedback of products of a seller |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

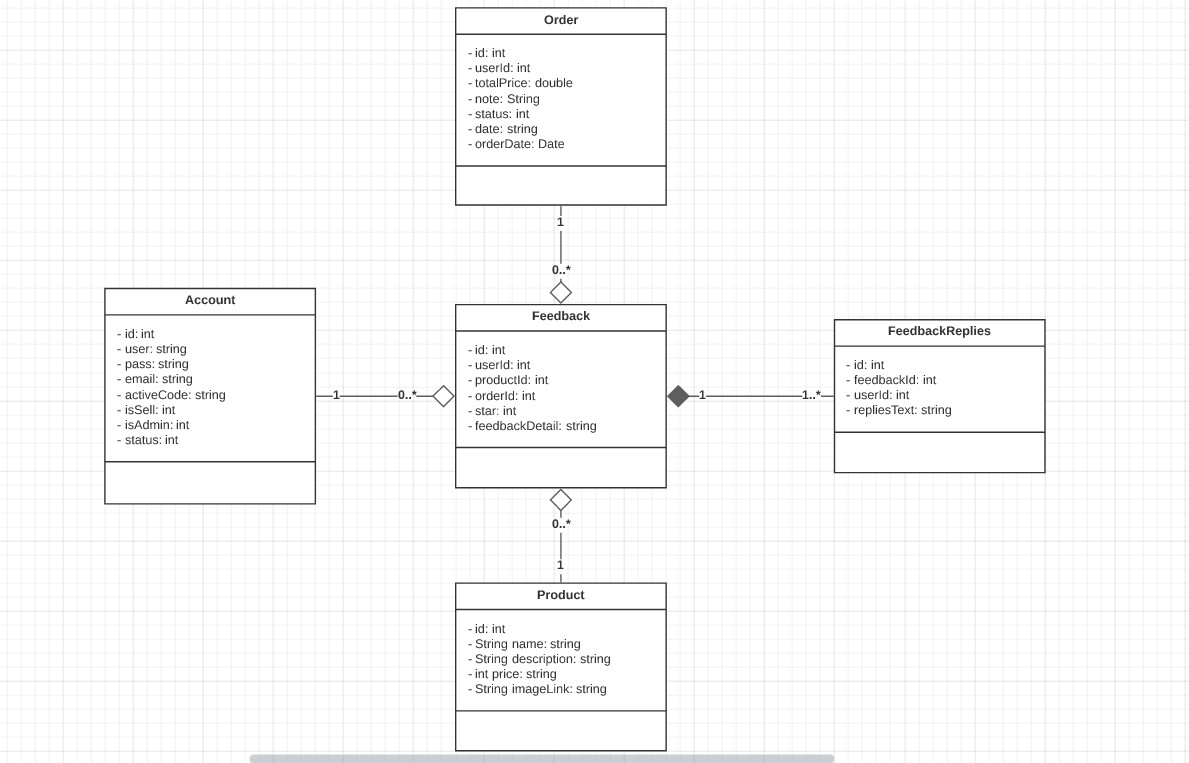
SELECT \* FROM Product WHERE SellerID = ?

SELECT \* FROM Feedback WHERE ProductID = ?

SELECT \* FROM Users WHERE UserID = ?

## 3. Feedback Details

### a. Class Diagram



### b. Class Specifications

#### Feedback Class: This class stores information of a Feedback

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Product ID of a Feedback object |
| 6 | setProductId() | Set the Product ID of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | getProduct() | Get the Product of a Feedback object |
| 14 | setProduct() | Set the Product of a Feedback object |
| 15 | getUser() | Get the User of a Feedback object |
| 16 | setUser() | Set the User of a Feedback object |
| 17 | getOrder() | Get the Order of a Feedback object |
| 18 | setOrder() | Set the Order of a Feedback object |
| 19 | toString() | Convert all information of a Feedback object to a string |

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### Account Class: This class stores information of a Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getPass() | Get the Password of an Account |
| 6 | setPass() | Set the Password of an Account |
| 7 | getEmail() | Get the Email of an Account |
| 8 | setEmail() | Set the Email of an Account |
| 9 | getIsSell() | Check if an Account has the seller role |
| 10 | setIsSell() | Add/remove seller role to/from an Account |
| 11 | getIsAdmin() | Check if an Account has the admin role |
| 12 | setIsAdmin() | Add/remove admin role to/from an Account |
| 13 | getStatus() | Get the Status of an Account |
| 14 | setStatus() | Set the Status of an Account |
| 15 | toString() | Convert all information of an Account to a string |

#### FeedbackDAO Class: This class performs CRUD actions to Feedback table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getFeedbacksBy  ProductId() | Get a list of feedback by product id |
| 2 | getFeedbacks() | Get a list all feedback from the database |
| 3 | getFeedbacksById() | Get a feedback by id |
| 4 | getFeedbacks  ByUserId() | Get a list of feedback by user id |
| 5 | getFeedbacksBy  UserIdAndProductId() | Get a list of feedback by user id and product id |
| 6 | getFeedbacksBy  OrderId() | Get a list of feedback by order id |
| 7 | addFeedback() | Add a feedback to database |
| 8 | countTotalFeedback() | Count total number of feedback in the database |

#### ProductDAO Class: This class performs CRUD actions to Product table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getAllProduct() | Get all products from database |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | getProductsByCate  Id() | return the list of products of a particular category |
| 6 | Edit() | Edit the information of a particular product, select by product ID |
| 7 | Delete() | Delete a product, select by its ID |
| 8 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 9 | countProduct() | return the total number of products |
| 10 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 11 | pagingByCategory() | categorizing products and return list of products with same category |
| 12 | pagingManager  Product() | Paging for manage product |
| 13 | countProductBy  Category() | count total product |
| 14 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 15 | getProductByID() | Get data of Product Table searched by UserID |
| 16 | getCateIdOfProductByID() | Get Category ID of a Product by Product ID |
| 17 | getProductDetailBy  ID() | Get Product for Detail |
| 18 | getProductForManager() | Get a particular product and return a product |
| 19 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 20 | searchProductBy  NameAndCateId() | Search for a particular product by filling its name and Category ID |
| 21 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 22 | top3MostSell() | Select top 3 most sold products and return product |
| 23 | top3LeastSell() | Select top 3 least sold products and return product |
| 24 | countProductBy  Category() | Count the number of products by category |
| 25 | getRelatedProduct() | Get 3 related products to a Category by Category ID |

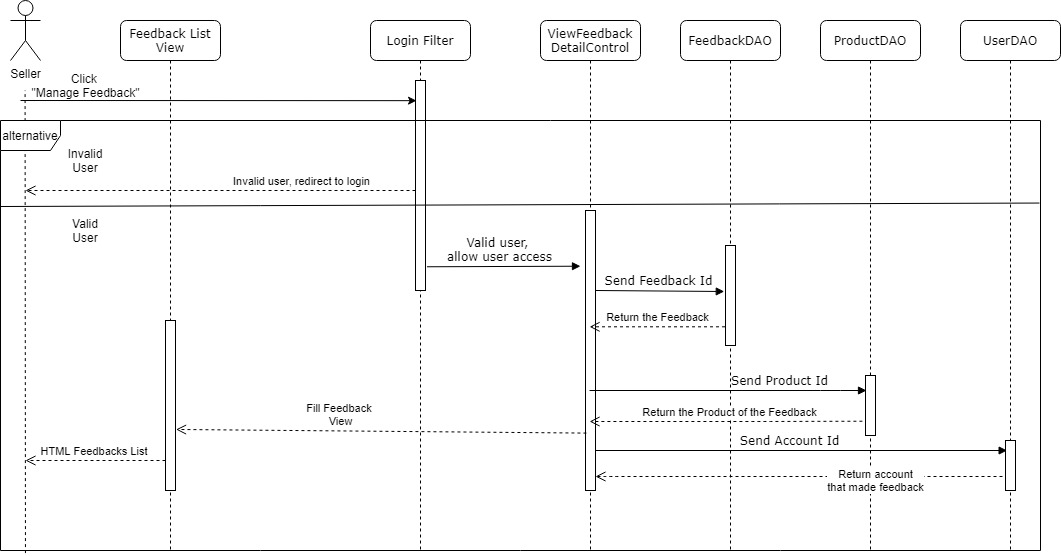
#### UserDAO Class: This class performs CRUD actions to Users table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | getSellerIdOfAn  Order() | Get the Seller ID of an order |
| 6 | deleteAccount() | Delete an account from the database |
| 7 | editAccount() | Update the information of an account |
| 8 | updatePassword() | Update the password of a user |
| 9 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 10 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 11 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 12 | updateStatus() | Update status of an account and return true if update successful |
| 13 | countAllAccount() | Return the number of account in the database |
| 14 | checkForget  Password() | Get username and email of a user that forget password |
| 15 | searchAccountInManager() | Get accounts by username |

#### ViewFeedbackDetail Class: : This class process request for detail of a feedback by seller

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Return Detail of feedback |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

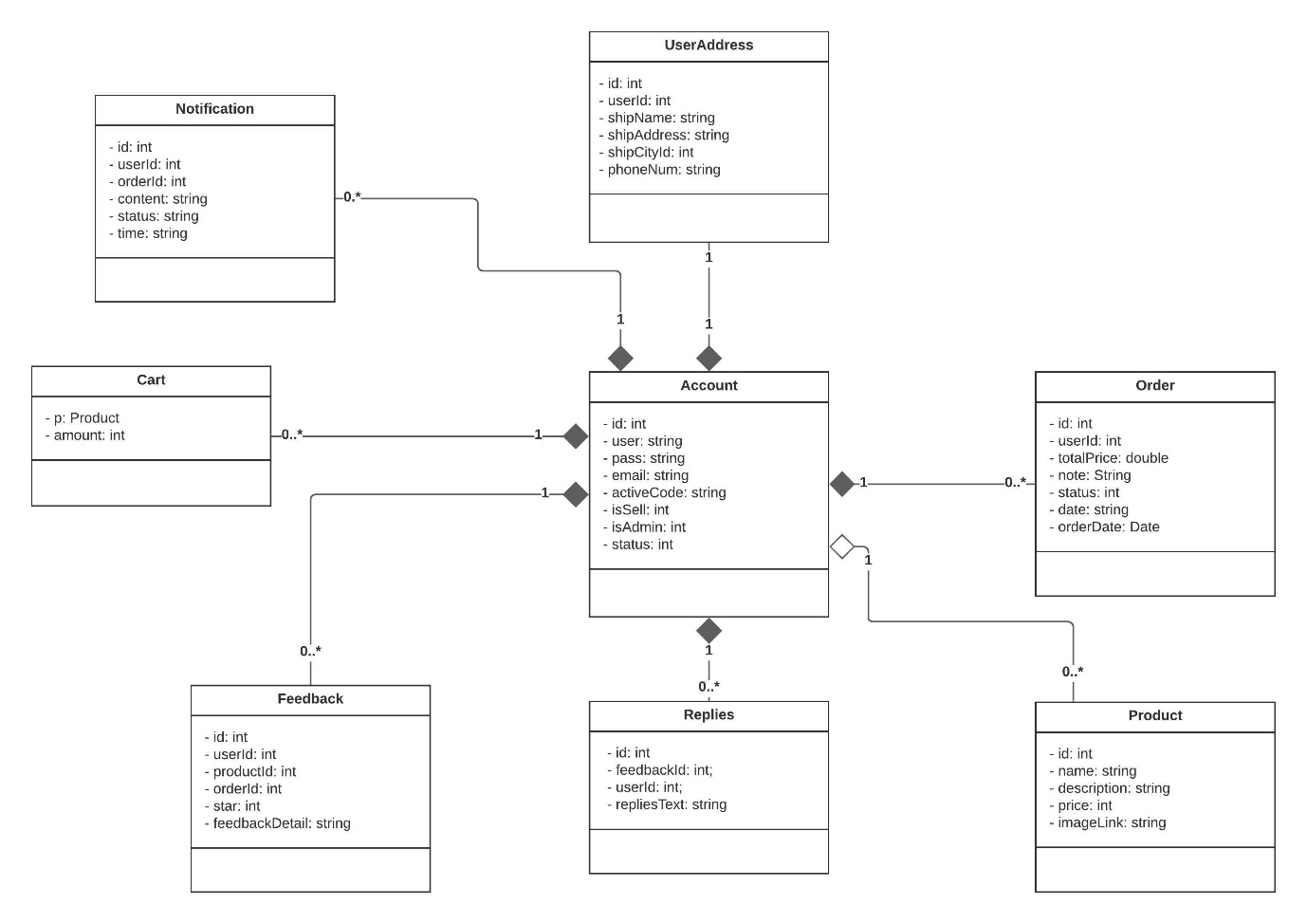
SELECT \* FROM Feedback WHERE ID = ?

SELECT \* FROM Product WHERE ProductID = ?

SELECT \* FROM Users WHERE UserID = ?

## 4. User Register

### a. Class Diagram



### b. Class Specifications

#### Account Class: This class stores information of a Account

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getPass() | Get the Password of an Account |
| 6 | setPass() | Set the Password of an Account |
| 7 | getEmail() | Get the Email of an Account |
| 8 | setEmail() | Set the Email of an Account |
| 9 | getIsSell() | Check if an Account has the seller role |
| 10 | setIsSell() | Add/remove seller role to/from an Account |
| 11 | getIsAdmin() | Check if an Account has the admin role |
| 12 | setIsAdmin() | Add/remove admin role to/from an Account |
| 13 | getStatus() | Get the Status of an Account |
| 14 | setStatus() | Set the Status of an Account |
| 15 | toString() | Convert all information of an Account to a string |

#### UserDAO Class: This class performs CRUD actions to Users table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | getSellerIdOfAn  Order() | Get the Seller ID of an order |
| 6 | deleteAccount() | Delete an account from the database |
| 7 | editAccount() | Update the information of an account |
| 8 | updatePassword() | Update the password of a user |
| 9 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 10 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 11 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 12 | updateStatus() | Update status of an account and return true if update successful |
| 13 | countAllAccount() | Return the number of account in the database |
| 14 | checkForget  Password() | Get username and email of a user that forget password |
| 15 | searchAccountInManager() | Get accounts by username |

#### SendEmail Class: This class performs send email action

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | sentEmail() | Send an email |

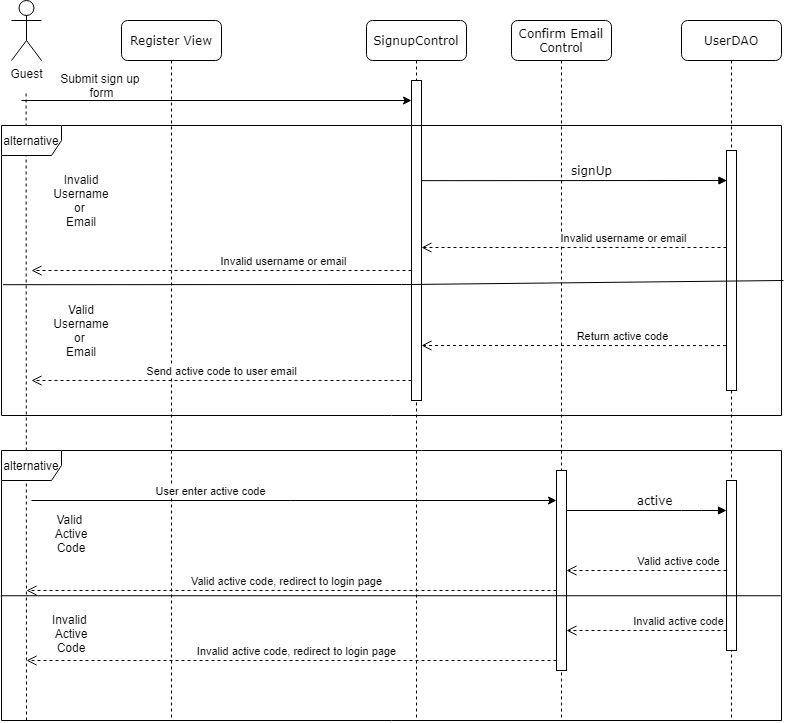
#### SignUpControl Class: This class process request for register

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Get information from user register form, register user account, redirect to code page |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

#### ConfirmEmailControl Class: This class process request for active email

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Get inputted code from user, active email |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

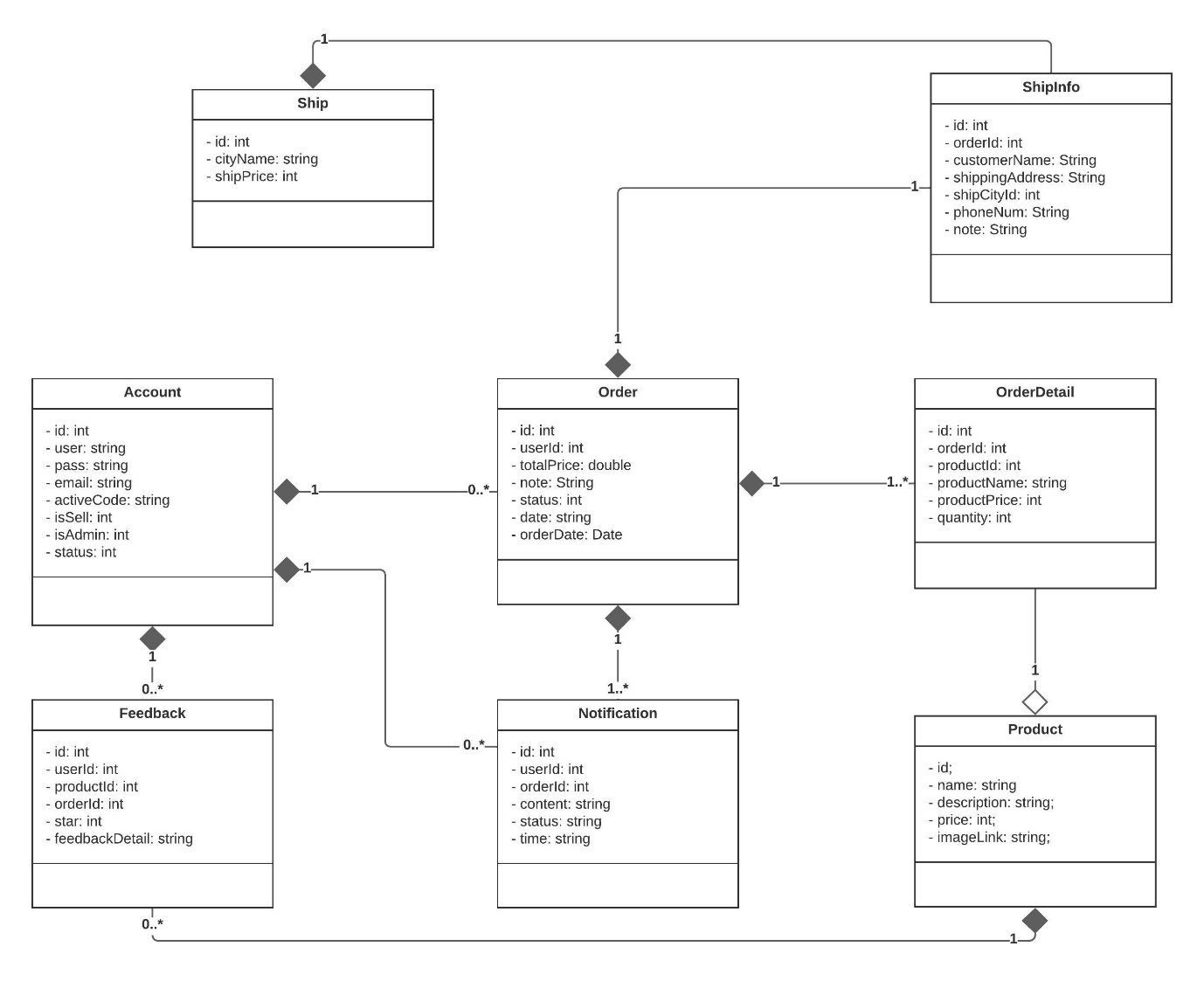
SELECT \* FROM Users WHERE email = ?

SELECT \* FROM Users WHERE Username = ?

INSERT INTO Users VALUES (?, ?, ?, ?, ?, ?,?)

## 5. Order Information

### a. Class Diagram



### b. Class Specifications

#### Feedback Class: This class stores information of a Feedback

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Product ID of a Feedback object |
| 6 | setProductId() | Set the Product ID of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | getProduct() | Get the Product of a Feedback object |
| 14 | setProduct() | Set the Product of a Feedback object |
| 15 | getUser() | Get the User of a Feedback object |
| 16 | setUser() | Set the User of a Feedback object |
| 17 | getOrder() | Get the Order of a Feedback object |
| 18 | setOrder() | Set the Order of a Feedback object |
| 19 | toString() | Convert all information of a Feedback object to a string |

#### Order Class: This class stores information of an Order

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Order object |
| 2 | setId() | Set the ID of an Order object |
| 3 | getUserId() | Get the User ID of an Order object |
| 4 | setUserId() | Set the User ID of an Order object |
| 5 | getTotalPrice() | Get the Subtotal of an Order object |
| 6 | setTotalPrice () | Set the Subtotal of an Order object |
| 7 | getNote () | Get the Note of an Order object |
| 8 | setNote() | Set the Note of an Order object |
| 9 | getStatus () | Get the Status of an Order object |
| 10 | setStatus() | Set the Status of an Order object |
| 11 | getDate() | Get the Order Date of an Order object |
| 12 | setDate() | Set the Order Date of an Order object |

#### OrderDetailWithImage Class: This class stores information of an Order Detail with Image

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of an order detail object |
| 2 | setId() | Set the ID of an order detail object |
| 3 | getOrderId() | Get the Order ID of an order detail object |
| 4 | setOrderId() | Set the Order ID of an order detail object |
| 5 | getProductId() | Get the Product ID of an order detail object |
| 6 | setProductId() | Set the Product ID of an order detail object |
| 7 | getProductName() | Get the Product Name of an order detail object |
| 8 | setProductName() | Set the Product Name of an order detail object |
| 9 | getProductPrice() | Get the Product Price of an order detail object |
| 10 | setProductPrice() | Set the Product Price of an order detail object |
| 11 | getImageLink() | Get the Image Link of an order detail object |
| 12 | setImageLink() | Set the Image Link of an order detail object |
| 13 | toString() | Convert all information of an order detail object to a string |

#### FeedbackDAO Class: This class performs CRUD actions to Feedback table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getFeedbacksBy  ProductId() | Get a list of feedback by product id |
| 2 | getFeedbacks() | Get a list all feedback from the database |
| 3 | getFeedbacksById() | Get a feedback by id |
| 4 | getFeedbacks  ByUserId() | Get a list of feedback by user id |
| 5 | getFeedbacksBy  UserIdAndProductId() | Get a list of feedback by user id and product id |
| 6 | getFeedbacksByOrderId() | Get a list of feedback by order id |
| 7 | addFeedback() | Add a feedback to database |
| 8 | countTotalFeedback() | Count total number of feedback in the database |

#### OrderDAO Class: This class performs CRUD actions to Order table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | Adding a new order to database, return the order Id |
| 4 | packaging() | Change the status of an specific order by the order id to packaging |
| 5 | delivering() | Change the status of an specific order by the order id to delivering |
| 6 | canceled() | Change the status of an specific order by the order id to canceled |
| 7 | finished() | Change the status of an specific order by the order id to finished |
| 8 | countOrders() | Count all the number of orders in database and return integer number |
| 9 | getOrderByOrderID() | Get a particular order by its id and return Order |
| 10 | getNewestOrderID() | Get the order id of the newest order |
| 11 | getOrderByUserID() | Get list of orders of a specific user and return List Order |
| 12 | getStatusOfAn  Order() | Get the status of an order by order id |
| 13 | getCompletedOrder  ByUserID() | Get all completed orders of an User by User ID |
| 14 | getRecentOrder() | Get all recent orders |
| 15 | Delete() | Delete a specific order by the order id |

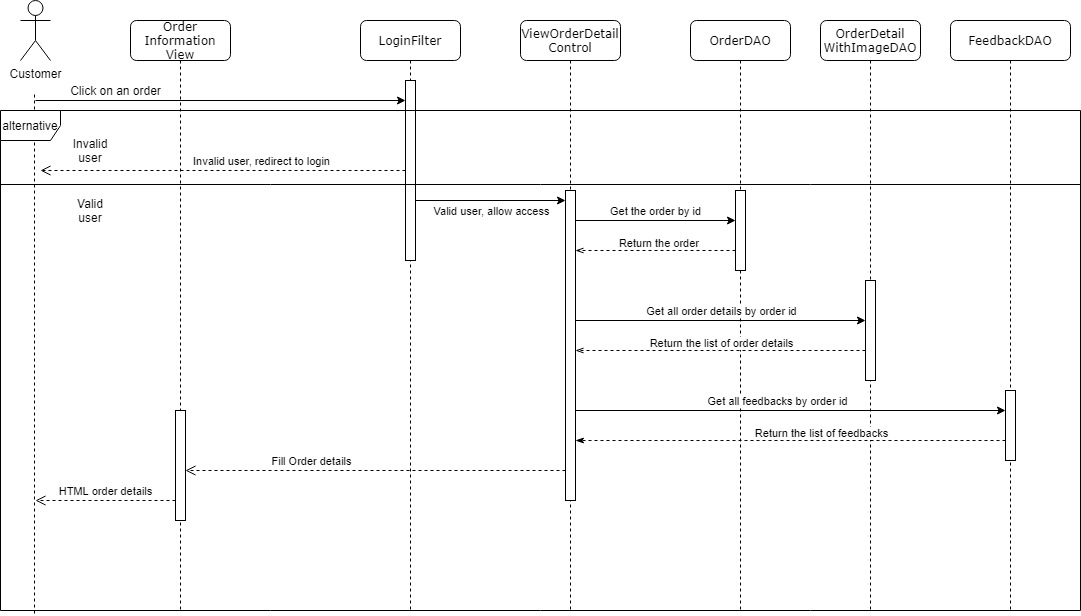
#### OrderDetailWithImageDAO Class: This class performs CRUD actions to Order\_Detail table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getOrderDetail() | Get all order details with image by Order ID |

#### ViewOrderDetailClass: This class process request for detail of an order by customer

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Return Detail of an order |
| 2 | doGet() | Handle GET request |
| 3 | doPost() | Handle POST request |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

SELECT od.id, od.order\_id, od.productId, od.productName, od.quantity, od.productPrice, p.imageLink FROM Order\_Detail od INNER JOIN Product p

ON od.ProductID = p.ProductID

WHERE od.Order\_ID = ?

SELECT o.ID,o.UserId,o.TotalPrice, o.Note, o.DayBuy, os.Name

FROM Orders o INNER JOIN Order\_Status os

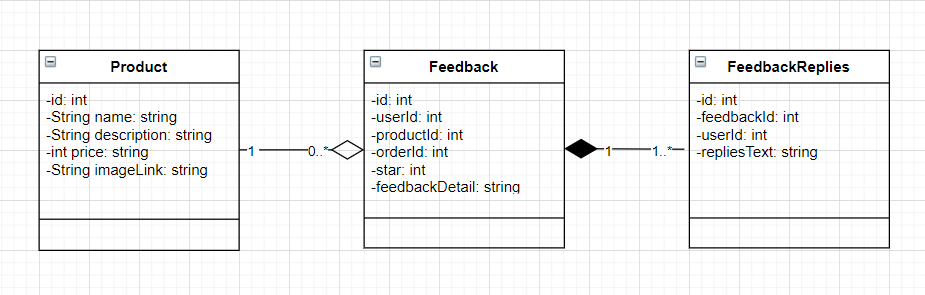
ON o.Status = os.ID

WHERE o.ID = ?

SELECT \* FROM Feedback WHERE OrderID = ?

## 6. Product Details

### a. Class Diagram



### b. Class Specification

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### Feedback Class: This class stores information of a Feedback

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Subtotal of a Feedback object |
| 6 | setProductId() | Set the Subtotal of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | toString() | Convert all information of a Feedback object to a string |

#### FeedbackReplies Class: This class stores information of a Feedback Replies

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a FeedbackReplies object |
| 2 | setId() | Set the ID of a FeedbackReplies object |
| 3 | getFeedbackId () | Get the Feedback ID of a FeedbackReplies object |
| 4 | setFeedbackId () | Set the Feedback ID of a FeedbackReplies object |
| 5 | getUserId () | Get the User ID of a FeedbackReplies object |
| 6 | setUserId () | Set the User ID of a FeedbackReplies object |
| 7 | getRepliesText() | Get the Replies Text of a FeedbackReplies object |
| 8 | setRepliesText() | Set the Replies Text of a FeedbackReplies object |
| 9 | toString () | Convert all information of a FeedbackReplies object to a string |

#### ProductDAO Class: This class performs CRUD actions to Product table in the database

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getAllProduct() | … |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | Edit() | Edit the information of a particular product, select by product ID |
| 6 | Delete() | Delete a product, select by its ID |
| 7 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 8 | countProduct() | return the total number of products |
| 9 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 10 | pagingByCategory() | categorizing products and return list of products with same category |
| 11 | pagingManager  Product() | … |
| 12 | countProductBy  Category() | count total product |
| 13 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 14 | getProductByID() | Get data of Product Table searched by UserID |
| 15 | getProductDetailBy  ID() | Get Product for Detail |
| 16 | getProductForManager() | Get a particular product and return a product |
| 17 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 18 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 19 | top3MostSell() | Select top 3 most sold products and return product |
| 20 | top3LeastSell() | Select top 3 least sold products and return product |

#### FeedbackDAO Class: This class performs CRUD actions to Feedback table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getFeedbacksBy  ProductId() | Get a list of feedback by product id |
| 2 | getFeedbacksBy  UserId() | Get a list of feedback by user id |

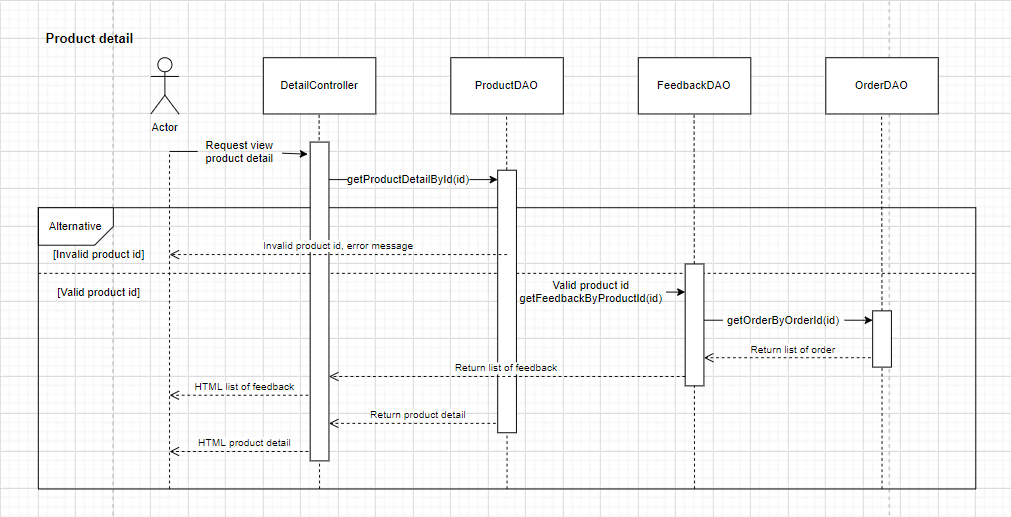
#### OrderDAO Class: This class performs CRUD actions to Order table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | adding a new order to database, return the order Id |
| 4 | packaging() | change the status of an specific order by the order id |
| 5 | delivering() | … |
| 6 | cancled() | … |
| 7 | countOrders() | Count all the number of orders in database and return integer number |
| 8 | getOrderByOrderID() | get a particular order by its id and return Order |
| 9 | getNewestOrderID() | … |
| 10 | getOrderByUserID() | get list of orders of a specific user and return List Order |
| 11 | Delete() | delete a specific order by the order id |

#### DetailControl Class: This class process request for detail of a product

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Return Detail of Product |
| 2 | doGet() | … |
| 3 | doPost() | … |
| 4 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram



### d. Database queries

* SELECT \*

FROM Product INNER JOIN Manufacturer

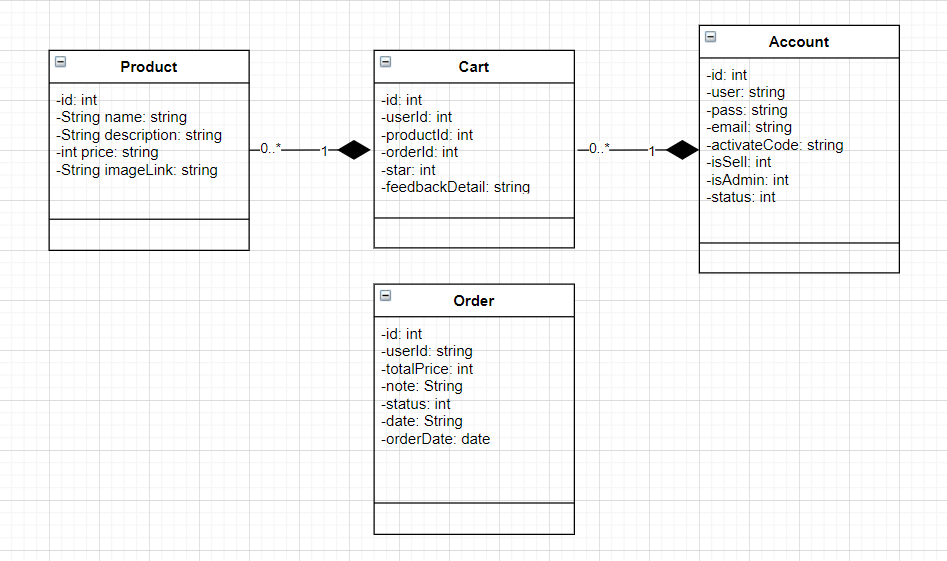
ON Product.ManufacturerID = Manufacturer.ManufacturerID

WHERE ProductID = ?

* SELECT \* FROM Product WHERE ProductID = ?

## 7. Cart Details

### a. Class Diagram



### b. Class Specifications

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |

#### Account Class: This class stores information of an Account

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

#### Order Class: This class stores information of an Order

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Order object |
| 2 | setId() | Set the ID of an Order object |
| 3 | getUserId() | Get the User ID of an Order object |
| 4 | setUserId() | Set the User ID of an Order object |
| 5 | getTotalPrice() | Get the Subtotal of an Order object |
| 6 | setTotalPrice () | Set the Subtotal of an Order object |
| 7 | getNote () | Get the Note of an Order object |
| 8 | setNote() | Set the Note of an Order object |
| 9 | getStatus () | Get the Status of an Order object |
| 10 | setStatus() | Set the Status of an Order object |
| 11 | getDate() | Get the Order Date of an Order object |
| 12 | setDate() | Set the Order Date of an Order object |

#### Cart Class: This class stores information of a Cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getP() | Get the Product of a Cart object |
| 2 | setP() | Set the Product of a Cart object |
| 3 | getAmount() | Get the Amount of a Cart object |
| 4 | setAmount() | Set the Amount of a Cart object |
| 5 | toString() | Convert all information of a Cart object to a string |

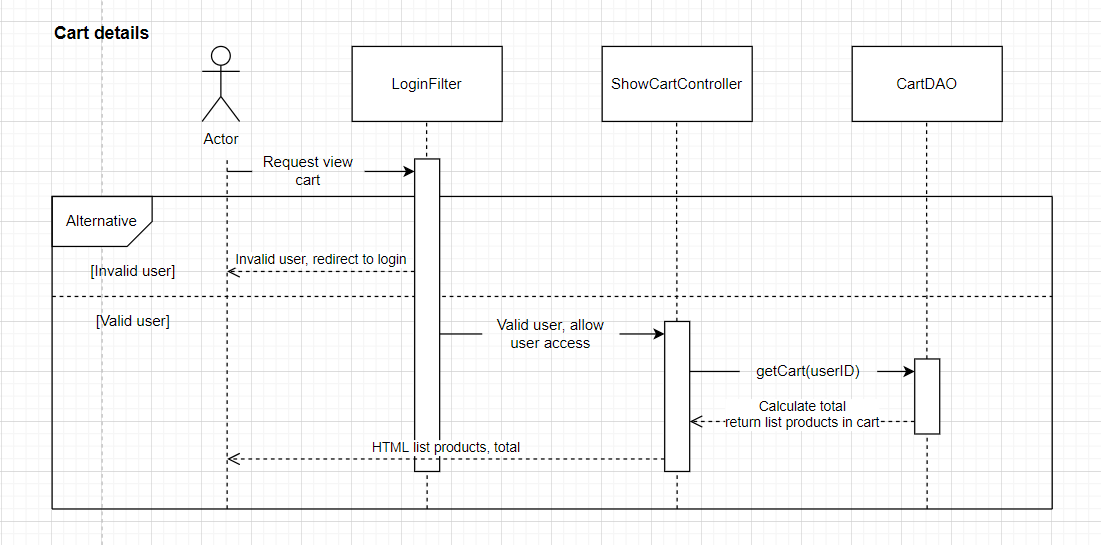
#### CartDAO Class: This class performs CRUD actions to Cart table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getCart() | Return list of products in cart of a particular customer |
| 2 | countCart() | Return number of different items in cart |
| 3 | countAllCart() | The number of all products in cart |
| 4 | addToCart() | Function returns a boolean to inform whether the product is added to cart or not |
| 5 | delete1Amount() | Minus 1 amount from Product after 1 Customer Add to cart |
| 6 | countAmountProduct() | Count the amount of a product to see if it is out of stock |
| 7 | countNumCart() | Return number of products in cart of a particular user |
| 8 | deleteCart() | Delete all products in cart of a particular user |
| 9 | add1ProductToCart() | Add 1 Product to Cart, Use when User Click "+" in ListCart |

#### ShowCartControlClass: This class process request for detail of a cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | … |
| 2 | getPriceWithDot() | … |
| 3 | doGet() | … |
| 4 | doPost() | … |
| 5 | getServletInfo() | return a String containing servlet description |

### c. Sequence Diagram(s)



### d. Database queries

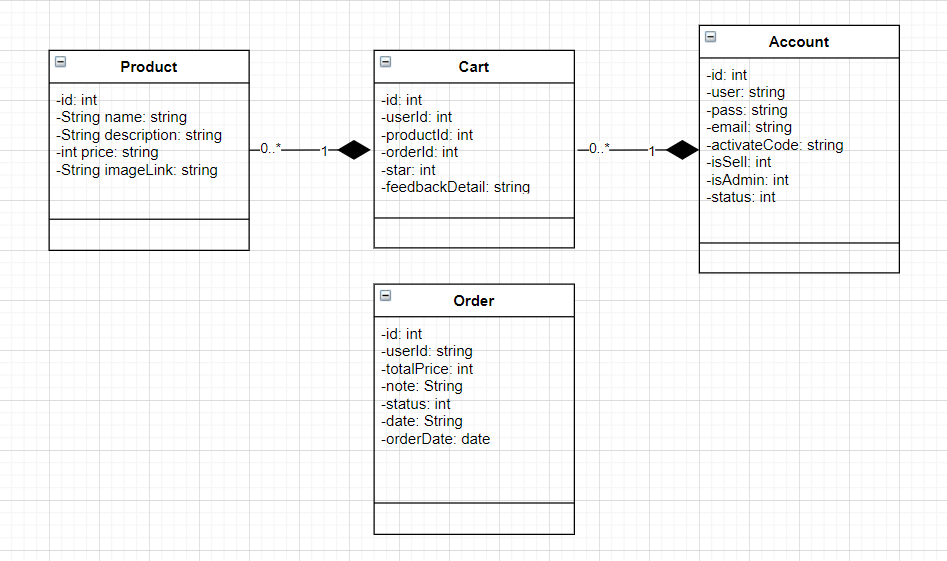
* UPDATE Cart

SET Amount = ?

WHERE UserID = ? AND ProductID = ?

## 8. Cart Completion

### a. Class Diagram



### b. Class Specifications

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |

#### Account Class: This class stores information of an Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

#### Order Class: This class stores information of an Order

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Order object |
| 2 | setId() | Set the ID of an Order object |
| 3 | getUserId() | Get the User ID of an Order object |
| 4 | setUserId() | Set the User ID of an Order object |
| 5 | getTotalPrice() | Get the Subtotal of an Order object |
| 6 | setTotalPrice () | Set the Subtotal of an Order object |
| 7 | getNote () | Get the Note of an Order object |
| 8 | setNote() | Set the Note of an Order object |
| 9 | getStatus () | Get the Status of an Order object |
| 10 | setStatus() | Set the Status of an Order object |
| 11 | getDate() | Get the Order Date of an Order object |
| 12 | setDate() | Set the Order Date of an Order object |

#### Cart Class: This class stores information of a Cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getP() | Get the Product of a Cart object |
| 2 | setP() | Set the Product of a Cart object |
| 3 | getAmount() | Get the Amount of a Cart object |
| 4 | setAmount() | Set the Amount of a Cart object |
| 5 | toString() | Convert all information of a Cart object to a string |

#### CartDAO Class: This class performs CRUD actions to Cart table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getCart() | Return list of products in cart of a particular customer |
| 2 | countCart() | Return number of different items in cart |
| 3 | countAllCart() | The number of all products in cart |
| 4 | addToCart() | Function returns a boolean to inform whether the product is added to cart or not |
| 5 | delete1Amount() | Minus 1 amount from Product after 1 Customer Add to cart |
| 6 | countAmountProduct() | Count the amount of a product to see if it is out of stock |
| 7 | countNumCart() | Return number of products in cart of a particular user |
| 8 | deleteCart() | Delete all products in cart of a particular user |
| 9 | add1ProductToCart() | Add 1 Product to Cart, Use when User Click "+" in ListCart |

#### ShipDAO Class: This class performs CRUD actions to Shipping table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllShip() | Get every shipping orders and return list of ship |
| 2 | getShipPriceBy  CityName() | Get a shipping price from db using cityName and return a int number |
| 3 | getShipPriceByCId() | Get a shipping price from db using the id and return an int number |
| 4 | getCityByCId() | Get a shipping price from db using the id and return an int number |

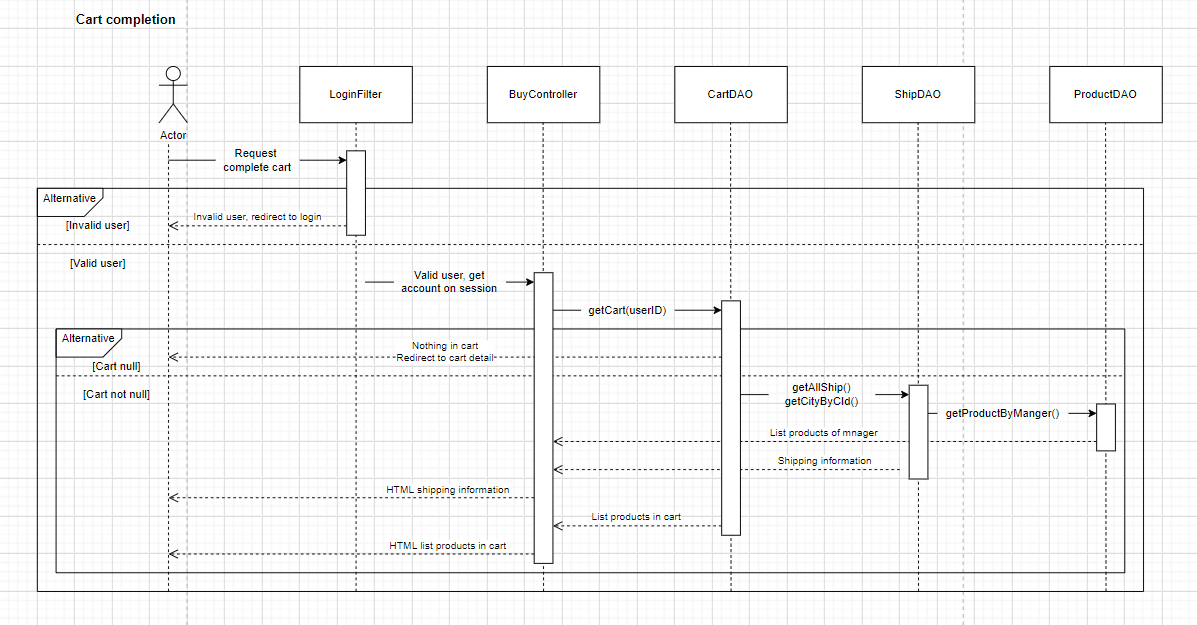
#### ProductDAO Class: This class performs CRUD actions to Product table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllProduct() | … |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | Edit() | Edit the information of a particular product, select by product ID |
| 6 | Delete() | Delete a product, select by its ID |
| 7 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 8 | countProduct() | return the total number of products |
| 9 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 10 | pagingByCategory() | categorizing products and return list of products with same category |
| 11 | pagingManager  Product() | … |
| 12 | countProductBy  Category() | count total product |
| 13 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 14 | getProductByID() | Get data of Product Table searched by UserID |
| 15 | getProductDetailBy  ID() | Get Product for Detail |
| 16 | getProductForManager() | Get a particular product and return a product |
| 17 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 18 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 19 | top3MostSell() | Select top 3 most sold products and return product |
| 20 | top3LeastSell() | Select top 3 least sold products and return product |

#### BuyControl Class: This class process request for buying products in a cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | … |
| 2 | doGet() | Use session to call id -> create necessary DAO -> get products in user's cart -> Get All Ships information -> calculate total sum -> Set attribute, send to Buy.jsp |
| 3 | doPost() | … |
| 4 | getPriceWithDot() | …. |

**c. Sequence Diagram(s)**



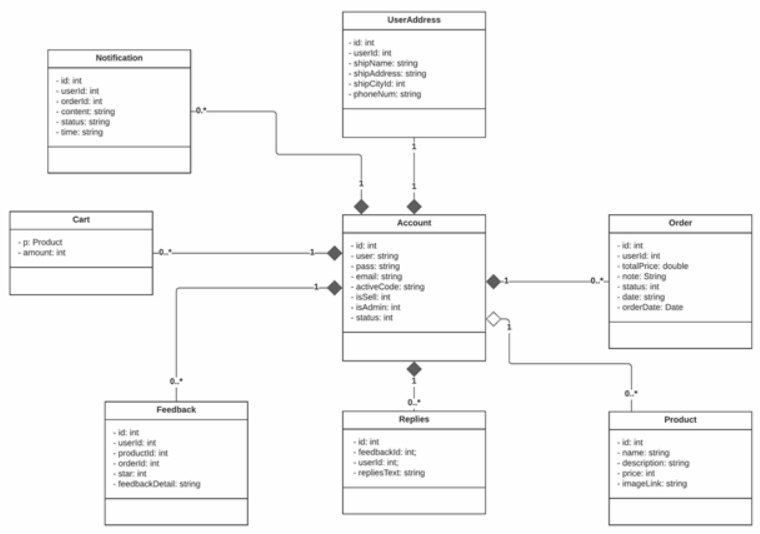
**d. Database queries**

* SELECT \* FROM Product WHERE ProductID = ?
* SELECT Product.ProductID, Product.ProductName, Product.Description, Product.SellPrice, Product.imageLink, Cart.Amount FROM Cart INNER JOIN Product

ON Cart.ProductID = Product.ProductID WHERE Cart.UserID = ?

## 9. Change Password

### a. Class Diagram



### b. Class Specifications

#### Account Class: This class stores information of an Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

#### Feedback Class: This class stores information of a Feedback

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Subtotal of a Feedback object |
| 6 | setProductId() | Set the Subtotal of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | toString() | Convert all information of a Feedback object to a string |

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### Cart Class: This class stores information of a Cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getP() | Get the Product of a Cart object |
| 2 | setP() | Set the Product of a Cart object |
| 3 | getAmount() | Get the Amount of a Cart object |
| 4 | setAmount() | Set the Amount of a Cart object |
| 5 | toString() | Convert all information of a Cart object to a string |

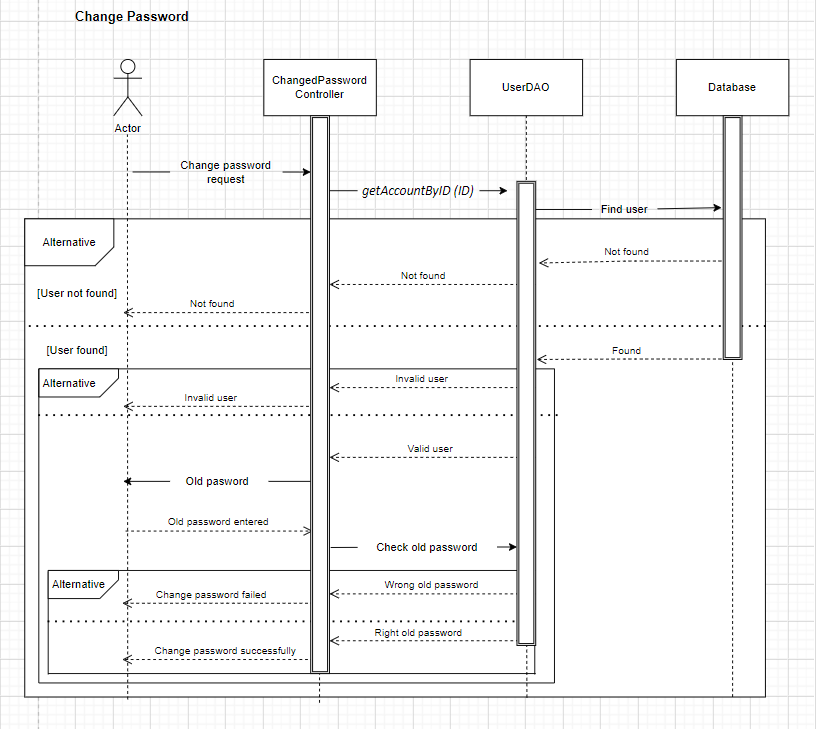
#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | deleteAccount() | Delete an account from the database |
| 6 | editAccount() | Update the information of an account |
| 7 | updatePassword() | Update the password of a user |
| 8 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 9 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 10 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 11 | updateStatus() | Update status of an account and return true if update successful |
| 12 | countAllAccount() | Return the number of account in the database |
| 13 | checkForget  Password() | … |

#### ChangePasswordControl Class: This class process request for change user password

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | … |
| 2 | doGet() | … |
| 3 | doPost() | … |
| 4 | getServletInfo() | return a String containing servlet description |

**c. Sequence Diagram(s)**

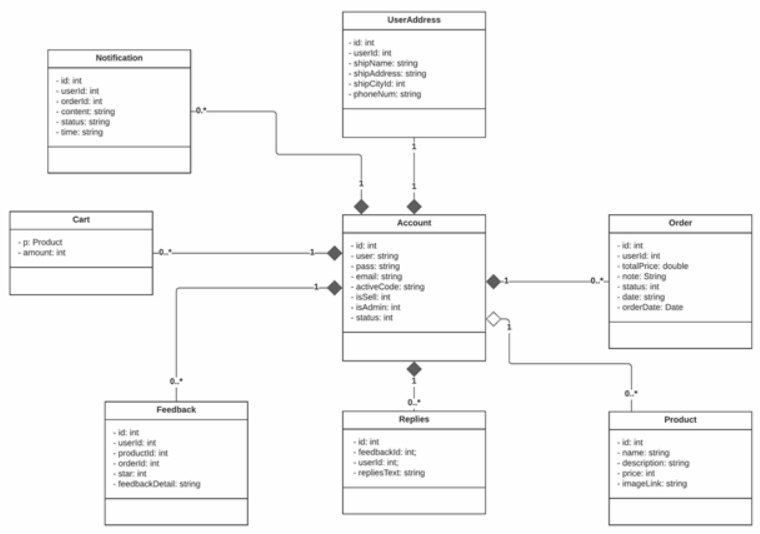


### d. Database queries

* SELECT \* FROM Users WHERE UserID = ?
* UPDATE Users SET Password = ? WHERE UserID = ?

## 10. User Profile

**a. Class Diagram**



### b. Class Specifications

#### Account Class: This class stores information of an Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

#### Feedback Class: This class stores information of a Feedback

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Subtotal of a Feedback object |
| 6 | setProductId() | Set the Subtotal of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | toString() | Convert all information of a Feedback object to a string |

#### Product Class: This class stores information of a Product

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

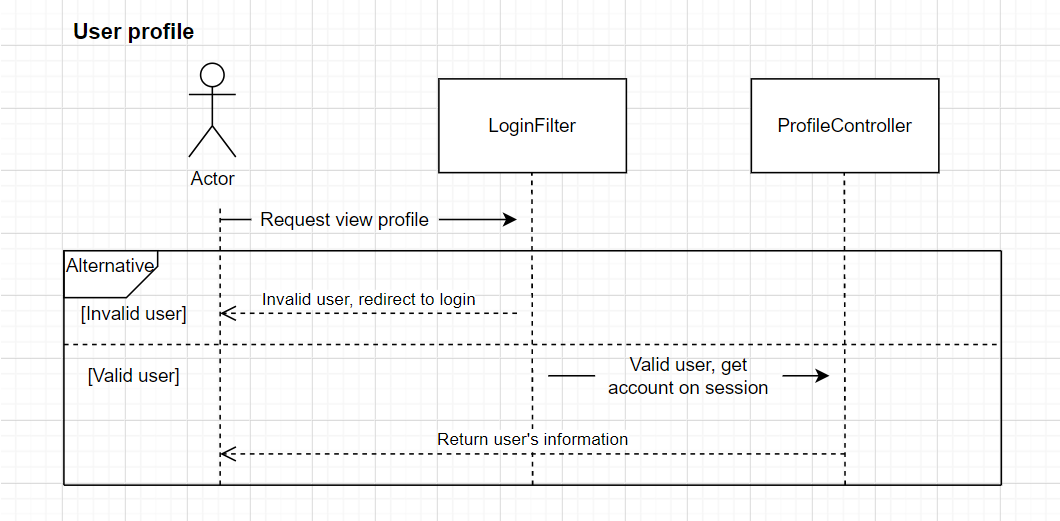
#### Cart Class: This class stores information of a Cart

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getP() | Get the Product of a Cart object |
| 2 | setP() | Set the Product of a Cart object |
| 3 | getAmount() | Get the Amount of a Cart object |
| 4 | setAmount() | Set the Amount of a Cart object |
| 5 | toString() | Convert all information of a Cart object to a string |

#### ProfileControlClass: This class process request for detail of a user’s profile

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Get current account from session and push to UserProfile.jsp |
| 2 | doGet() | … |
| 3 | doPost() | … |
| 4 | getServletInfo() | return a String containing servlet description |

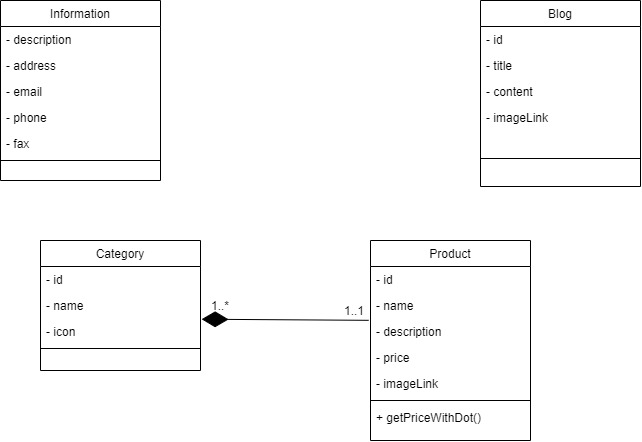
### c. Sequence Diagram(s)



### d. Database queries

## 11. Home Page

### a. Class Diagram



### b. Class Specifications

#### HomeControl Class

*This class takes the data from DAO and push it to JSP to display*

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | This method contains all logic execution of the class |

#### ProductDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getTop4SalePercent | This method get the Top 4 Products that has the highest Sale Percent   * Output: List<Product> |
| 2 | getAllProduct() | Get all Products in Database |
| 3 | getHotProduct() | Return Product with the highest sold amount |
| 4 | getFavoriteProduct  () | Return get the first and the second product with most sold amount |
| 5 | getProductBy  SellID() | Return the list of products of a particular seller |
| 6 | Edit() | Edit the information of a particular product, select by product ID |
| 7 | Delete() | Delete a product, select by its ID |
| 8 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 9 | countProduct() | Return the total number of products |
| 10 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 11 | pagingByCategory() | Categorizing products and return list of products with same category |
| 12 | pagingManager  Product() | Return 6 Product for 1 Page in List Product of Seller |
| 13 | countProductBy  Category() | Count total product |
| 14 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 15 | getProductByID() | Get data of Product Table searched by UserID |
| 16 | getProductDetailBy  ID() | Get Product for Detail |
| 17 | getProductForManager() | Get a particular product and return a product |
| 18 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 19 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 20 | top3MostSell() | Select top 3 most sold products and return product |
| 21 | top3LeastSell() | Select top 3 least sold products and return product |

#### Product Class

This class store the information of a Product

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### InforDAO Class

This class performs action to Data (CRUD) in Information Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getInfor() | Return information of the shop |

#### Information Class

This class store the information about the Information Table

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getDescription() | Get the Description of an Information object |
| 2 | setDescription() | Set the Description of an Information object |
| 3 | getAddress() | Get the Address of an Information object |
| 4 | setAddress() | Set the Address of an Information object |
| 5 | getEmail() | Get the Email of an Information object |
| 6 | setEmail() | Set the Email of an Information object |
| 7 | getPhone() | Get the Phone of an Information object |
| 8 | setPhone() | Set the Phone of an Information object |
| 9 | getFax() | Get the Fax of an Information object |
| 10 | setFax() | Set the Fax of an Information object |
| 11 | toString() | Convert all information of an Information object to a string |

#### CategoryDAO Class

This class performs action to Data (CRUD) in Category Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllCategory() | Return all list Category |
| 2 | getCateNameByID() | Select a specified category by its ID, used for categorizing products in homepage |

#### Category Class

This class store the information about a Category

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Category object |
| 2 | setId() | Set the ID of a Category object |
| 3 | getName() | Get the Name of a Category object |
| 4 | setName() | Set the Name of a Category object |
| 5 | getIcon() | Get the Icon of a Category object |
| 6 | setIcon() | Set the Icon of a Category object |
| 7 | toString() | Convert all information of a Category object to a string |

#### BlogDAO Class

This class performs action to Data (CRUD) in Blog Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getHotBlog () | Get the First Blog in Database |
| 2 | getAllBlog () | Get all Blogs in Database |
| 3 | getBlogByID() | Get a Blog by its ID |
| 4 | add () | Add a Blog to Database |
| 5 | edit () | Edit a Blog |
| 6 | delete () | Delete a Blog in Database |
| 7 | pagingBlog () | Get Blogs for Paging |

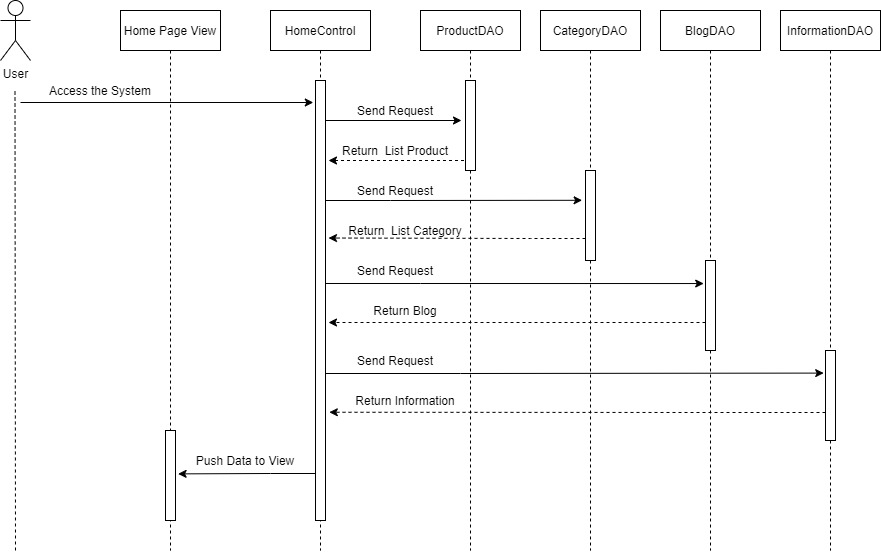
#### Blog Class

This class store the information about a Blog

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId () | Get the ID of a Blog object |
| 2 | setId () | Set the ID of a Blog object |
| 3 | getTitle () | Get the Title of a Blog object |
| 4 | setTitle () | Set the Title of a Blog object |
| 5 | getContent () | Get the Content of a Blog object |
| 6 | setContent () | Set the Content of a Blog object |
| 7 | getImageLink () | Get the ImageLink of a Blog object |
| 8 | setImageLink | Set the ImageLink of a Blog object |
| 9 | toString | Convert all information of a Category object to a string |

### c. Sequence Diagram(s)



### d. Database queries

*- SELECT \* FROM Category*

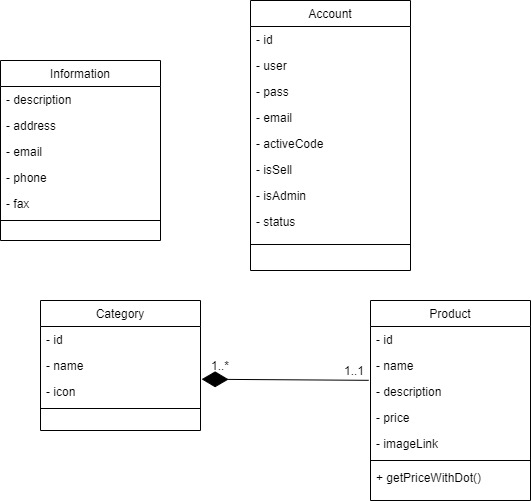
*- SELECT \* FROM Information*

*- SELECT TOP 1 \* FROM Blog ORDER BY id DESC*

*- SELECT TOP 4 \* FROM Product ORDER BY SalePercent DESC*

## 12. Product List

### a. Class Diagram



### b. Class Specifications

#### ProductListControl Class

*This class takes the data from DAO and push it to JSP to display*

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | This method contains all logic execution of the class |

#### ProductDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getTop4SalePercent | This method get the Top 4 Products that has the highest Sale Percent   * Output: List<Product> |
| 2 | getAllProduct() | Get all Products in Database |
| 3 | getHotProduct() | Return Product with the highest sold amount |
| 4 | getFavoriteProduct  () | Return get the first and the second product with most sold amount |
| 5 | getProductBy  SellID() | Return the list of products of a particular seller |
| 6 | Edit() | Edit the information of a particular product, select by product ID |
| 7 | Delete() | Delete a product, select by its ID |
| 8 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 9 | countProduct() | Return the total number of products |
| 10 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 11 | pagingByCategory() | Categorizing products and return list of products with same category |
| 12 | pagingManager  Product() | Return 6 Product for 1 Page in List Product of Seller |
| 13 | countProductBy  Category() | Count total product |
| 14 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 15 | getProductByID() | Get data of Product Table searched by UserID |
| 16 | getProductDetailBy  ID() | Get Product for Detail |
| 17 | getProductForManager() | Get a particular product and return a product |
| 18 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 19 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 20 | top3MostSell() | Select top 3 most sold products and return product |
| 21 | top3LeastSell() | Select top 3 least sold products and return product |

#### Product Class

This class store the information of a Product

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### InforDAO Class

This class performs action to Data (CRUD) in Information Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getInfor() | Return information of the shop |

#### Information Class

This class store the information about the Information Table

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getDescription() | Get the Description of an Information object |
| 2 | setDescription() | Set the Description of an Information object |
| 3 | getAddress() | Get the Address of an Information object |
| 4 | setAddress() | Set the Address of an Information object |
| 5 | getEmail() | Get the Email of an Information object |
| 6 | setEmail() | Set the Email of an Information object |
| 7 | getPhone() | Get the Phone of an Information object |
| 8 | setPhone() | Set the Phone of an Information object |
| 9 | getFax() | Get the Fax of an Information object |
| 10 | setFax() | Set the Fax of an Information object |
| 11 | toString() | Convert all information of an Information object to a string |

#### CategoryDAO Class

This class performs action to Data (CRUD) in Category Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllCategory() | Return all list Category |
| 2 | getCateNameByID() | Select a specified category by its ID, used for categorizing products in homepage |

#### Category Class

This class store the information about a Category

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Category object |
| 2 | setId() | Set the ID of a Category object |
| 3 | getName() | Get the Name of a Category object |
| 4 | setName() | Set the Name of a Category object |
| 5 | getIcon() | Get the Icon of a Category object |
| 6 | setIcon() | Set the Icon of a Category object |
| 7 | toString() | Convert all information of a Category object to a string |

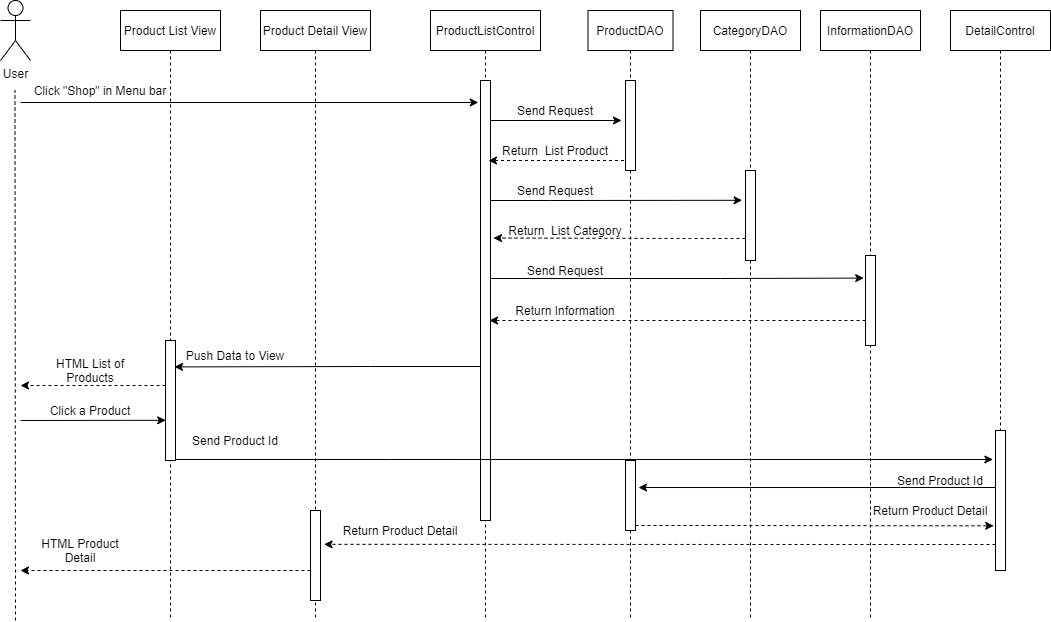
#### Account Class

This class store the information about an Account

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

### c. Sequence Diagram(s)



### d. Database queries

*- SELECT \* FROM Category*

*- SELECT \* FROM Information*

*- SELECT TOP 1 \* FROM Product ORDER BY Amount DESC*

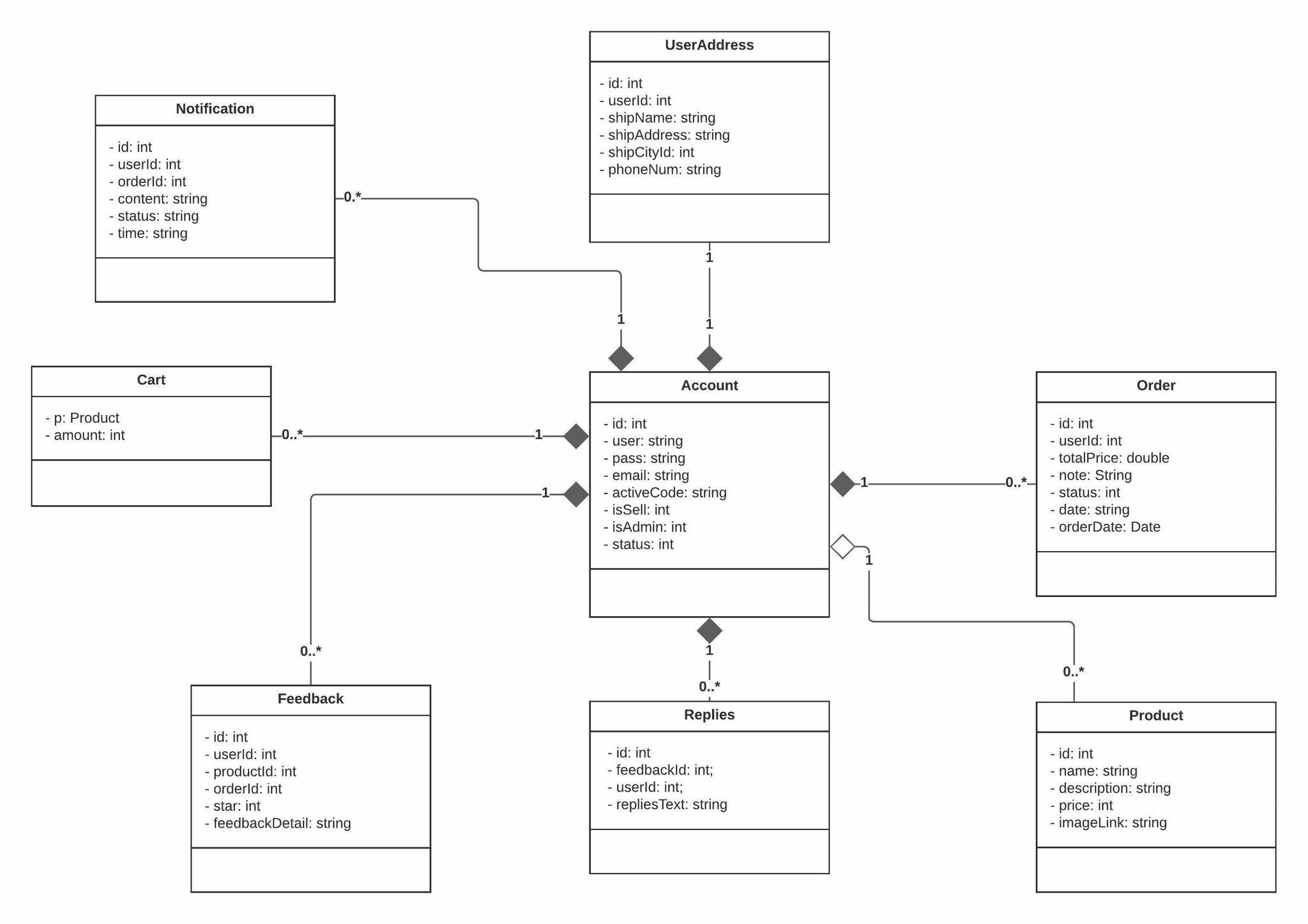
*- SELECT TOP 2 \* FROM Product ORDER BY Amount DESC*

*- SELECT COUNT(\*) FROM Product WHERE CategoryID = ?*

*- SELECT \* FROM Product WHERE CategoryID = ? ORDER BY ProductID OFFSET ? ROWS FETCH NEXT 6 ROWS ONLY*

## 13. User Login

### a. Class Diagram



### b. Class Specifications

#### LoginControl Class

*This class takes the data from DAO and push it to JSP to display, as well as execute the Login function*

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | doGet | This method get the information from Cookie to display in JSP |
| 2 | doPost | This method get the information from Cookie to execute Function |

#### UserDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | deleteAccount() | Delete an account from the database |
| 6 | editAccount() | Update the information of an account |
| 7 | updatePassword() | Update the password of a user |
| 8 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 9 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 10 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 11 | updateStatus() | Update status of an account and return true if update successful |
| 12 | countAllAccount() | Return the number of account in the database |
| 13 | checkForget  Password() | … |

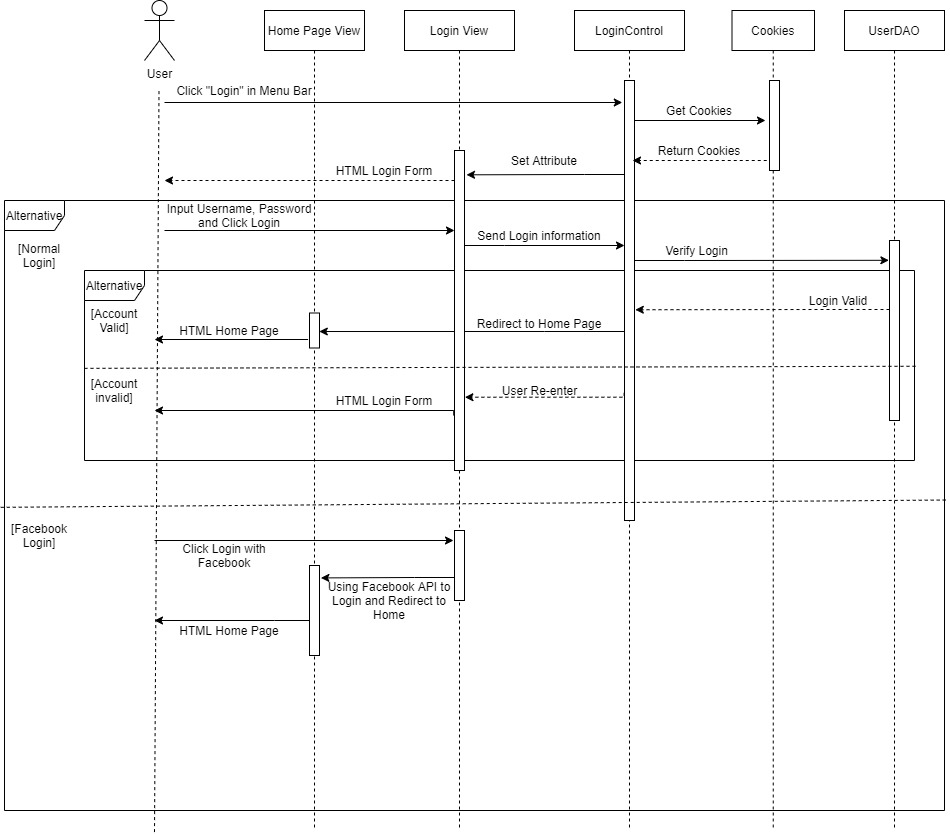
#### Account Class

This class store the information of an Account

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

### c. Sequence Diagram(s)



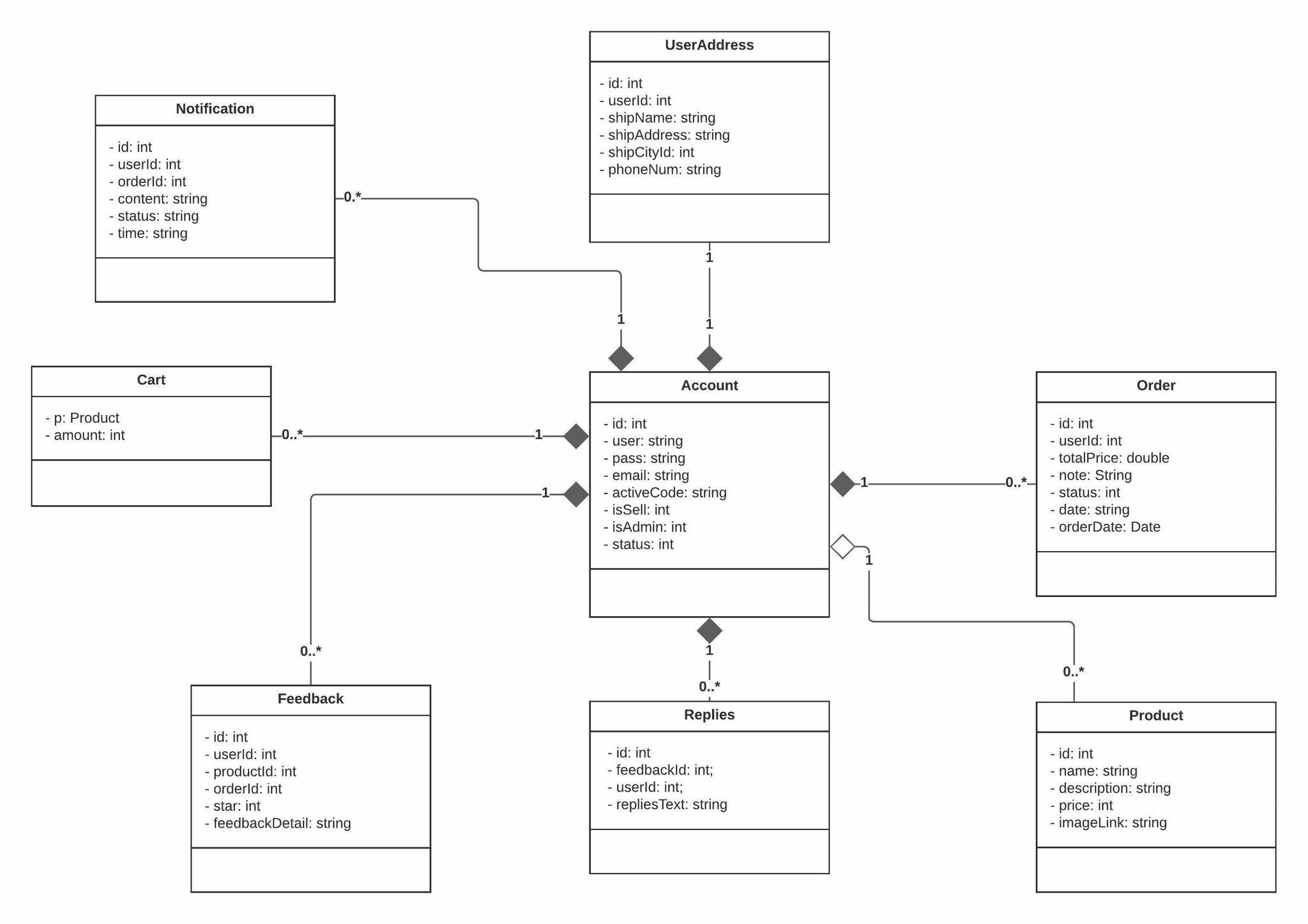
### d. Database queries

*- SELECT \* FROM Users WHERE email = ?*

*- SELECT \* FROM Users WHERE Username = ? AND Password = ?*

## 14. Reset Password

### a. Class Diagram



### b. Class Specifications

#### ForgetPasswordControl Class

*This class takes the information from JSP to send Verification Code*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | This method get the information from JSP, check it with Database and Send Verification Code through Email |
| 2 | doPost | This method get the Verification Code from JSP and Compare to make sure it’s correct |

#### Forget\_ChangePasswordControl Class

*This class allows user to enter new Password and update it in Database*

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | This method get the password from JSP to update in Database |

#### UserDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | deleteAccount() | Delete an account from the database |
| 6 | editAccount() | Update the information of an account |
| 7 | updatePassword() | Update the password of a user |
| 8 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 9 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 10 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 11 | updateStatus() | Update status of an account and return true if update successful |
| 12 | countAllAccount() | Return the number of account in the database |
| 13 | checkForget  Password() | … |

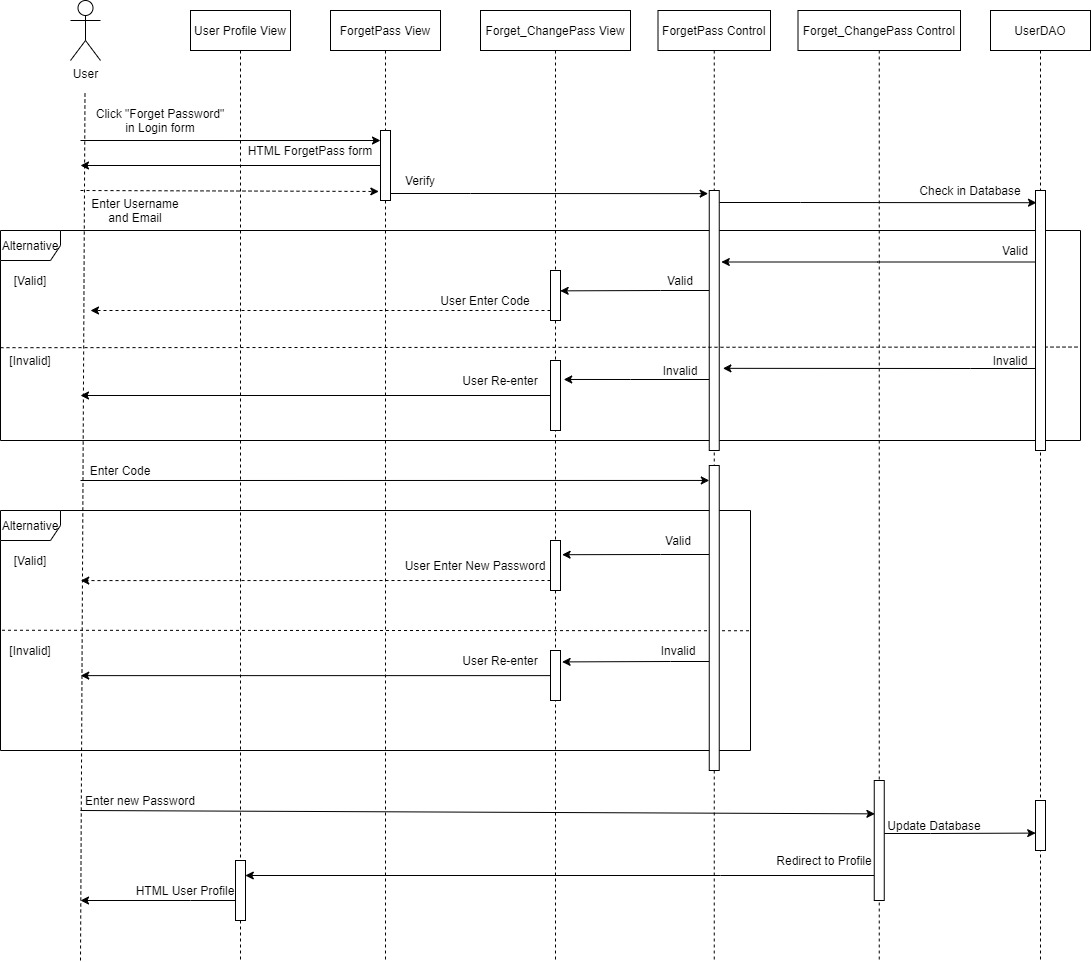
#### Account Class

This class store the information of an Account

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

### c. Sequence Diagram(s)



### d. Database queries

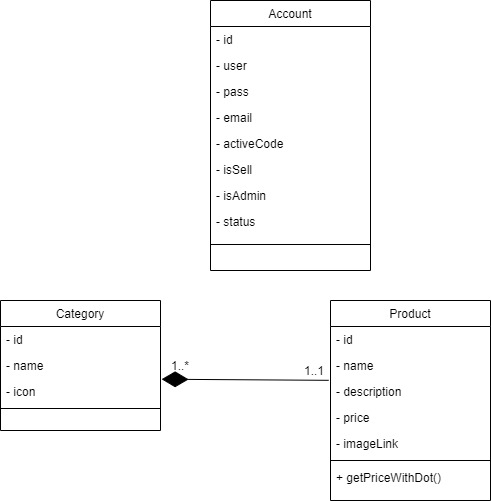
*- SELECT UserID FROM Users WHERE Username = ? AND email = ?*

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

*- UPDATE Users SET Password = ? WHERE UserID = ?*

## 15. Product List (Marketing)

### a. Class Diagram



### b. Class Specifications

#### ManagerControl Class

*This class takes the List Product from Database and Display it in JSP*

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest | This method contains all the logic execution of the Class |

#### ProductDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllProduct() | … |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | Edit() | Edit the information of a particular product, select by product ID |
| 6 | Delete() | Delete a product, select by its ID |
| 7 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 8 | countProduct() | return the total number of products |
| 9 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 10 | pagingByCategory() | categorizing products and return list of products with same category |
| 11 | pagingManager  Product() | … |
| 12 | countProductBy  Category() | count total product |
| 13 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 14 | getProductByID() | Get data of Product Table searched by UserID |
| 15 | getProductDetailBy  ID() | Get Product for Detail |
| 16 | getProductForManager() | Get a particular product and return a product |
| 17 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 18 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 19 | top3MostSell() | Select top 3 most sold products and return product |
| 20 | top3LeastSell() | Select top 3 least sold products and return product |

#### Account Class

This class store the information of an Account

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

#### Product Class

This class store the information of a Product

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getID | get ID Products |
| 02 | setID | set ID Products |
| 03 | getName() | get Name Products |
| 04 | setName() | set Name Products |
| 05 | getPrice() | get Price Products |
| 06 | setPrice() | set Price Products |
| 07 | getImageLink() | get ImageLink Products |
| 08 | setImageLink | set ImageLink Products |
| 09 | getDescription() | get Description Products |
| 10 | setDescription() | set Description Products |
| 11 | toString() | Output information |
| 12 | getPriceWithDot() | Function to write the Product's Price in proper form |

#### CategoryDAO Class

This class performs action to Data (CRUD) in Category Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllCategory() | Return all list Category |
| 2 | getCateNameByID() | Select a specified category by its ID, used for categorizing products in homepage |

#### Category Class

This class store the information about a Category

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Category object |
| 2 | setId() | Set the ID of a Category object |
| 3 | getName() | Get the Name of a Category object |
| 4 | setName() | Set the Name of a Category object |
| 5 | getIcon() | Get the Icon of a Category object |
| 6 | setIcon() | Set the Icon of a Category object |
| 7 | toString() | Convert all information of a Category object to a string |

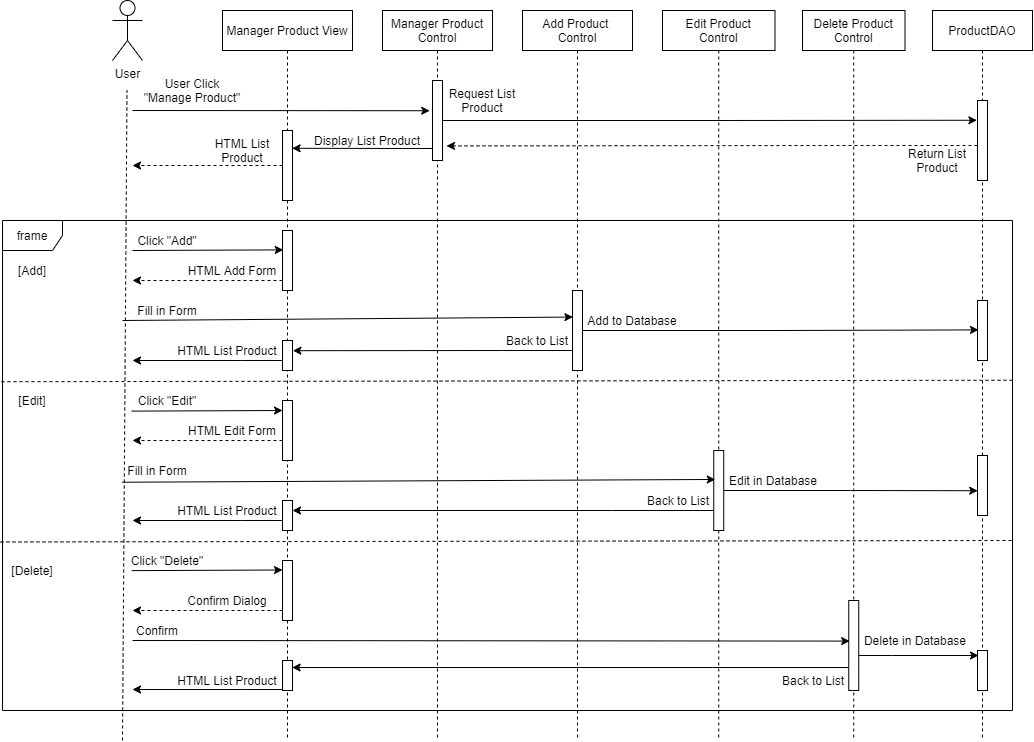
#### UserDAO Class

This class performs action to Data (CRUD) in Product Table in Database

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | deleteAccount() | Delete an account from the database |
| 6 | editAccount() | Update the information of an account |
| 7 | updatePassword() | Update the password of a user |
| 8 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 9 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 10 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 11 | updateStatus() | Update status of an account and return true if update successful |
| 12 | countAllAccount() | Return the number of account in the database |
| 13 | checkForget  Password() | … |

### c. Sequence Diagram(s)



### d. Database queries

*- SELECT \* FROM Product WHERE SellerID = ? ORDER BY ProductID OFFSET ? ROWS FETCH NEXT 6 ROWS ONLY*

*- SELECT COUNT(\*) FROM Product WHERE SellerID = ?*

*- SELECT \* FROM Category*

*- SELECT \* FROM Users*

*- INSERT INTO Product VALUES (?, ?, 0, ?, 0, ?, ?, ?, ?, 1, 1);*

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

*- UPDATE Product\n"*

*+ "SET ProductName = ?,\n"*

*+ "Description = ?,\n"*

*+ "SellPrice = ?,\n"*

*+ "imageLink = ?,\n"*

*+ "CategoryID = ?,\n"*

*+ "SellerID = ?,\n"*

*+ "Amount = ?\n"*

*+ "WHERE ProductID = ?*

*- DELETE FROM Cart WHERE ProductID = ?\n"*

*+ "DELETE FROM Product WHERE ProductID = ?*

## 16. My Order History

### a. Class Diagram

### b. Class Specifications

***ViewOrder***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the data from the database and dump it on the order history page to display the orders view’s user interface* |

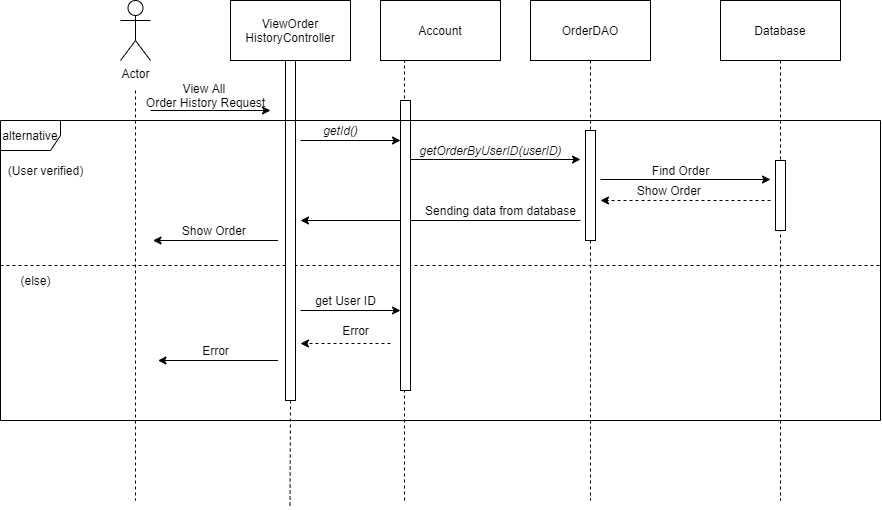
***OrderDAO***

***Class Methods***

This class is used performs CRUD with Order table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | adding a new order to database, return the order Id |
| 4 | packaging() | change the status of an order to packaging |
| 5 | delivering() | change the status of an order to delivering |
| 6 | cancled() | change the status of an order to canceled |
| 7 | countOrders() | Count all the number of orders in database and return integer number |
| 8 | getOrderByOrderID() | get a particular order by its id and return Order |
| 9 | getNewestOrderID() | Get the newest order ID |
| 10 | getOrderByUserID() | get list of orders of a specific user and return List Order |
| 11 | Delete() | delete a specific order by the order id |

### 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 = ?

## 17. My order details

### a. Class Diagram

### b. Class Specifications

***ViewOrderDetail***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the data from the database and dump it on the view order detail page to display the order detail’s user interface* |

***OrderDAO***

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | adding a new order to database, return the order Id |
| 4 | packaging() | change the status of an specific order by the order id |
| 5 | delivering() | … |
| 6 | cancled() | … |
| 7 | countOrders() | Count all the number of orders in database and return integer number |
| 8 | getOrderByOrderID() | get a particular order by its id and return Order |
| 9 | getNewestOrderID() | … |
| 10 | getOrderByUserID() | get list of orders of a specific user and return List Order |
| 11 | Delete() | delete a specific order by the order id |

***OrderWithImageDAO***

***Class Methods***

This class is used to perform CRUD with Order table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getOrderDetail(int Id)* | *Get a specific order detail by its ID* |

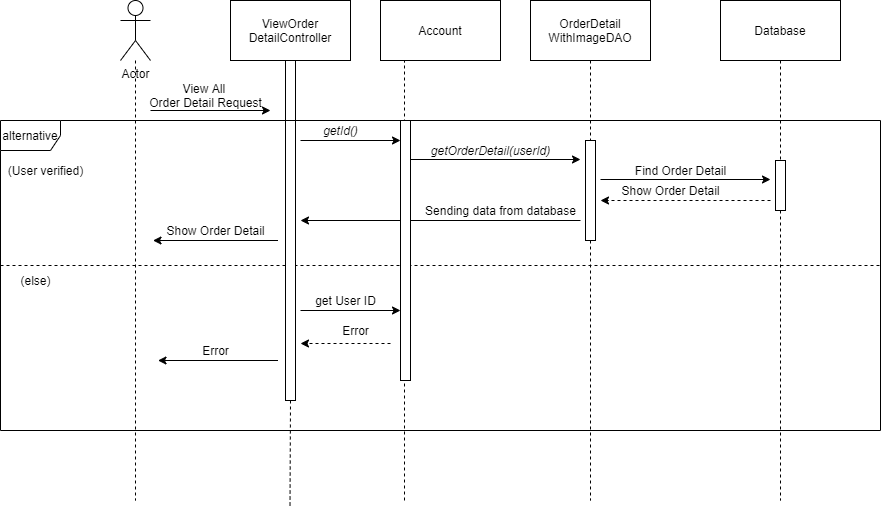
***FeedbackDAO***

***Class Methods***

This class is used to perform CRUD with Feedback table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Feedback object |
| 2 | setId() | Set the ID of a Feedback object |
| 3 | getUserId() | Get the User ID of a Feedback object |
| 4 | setUserId() | Set the User ID of a Feedback object |
| 5 | getProductId() | Get the Subtotal of a Feedback object |
| 6 | setProductId() | Set the Subtotal of a Feedback object |
| 7 | getStar() | Get the Star of a Feedback object |
| 8 | setStar() | Set the Star of a Feedback object |
| 9 | getFeedbackDetail() | Get the Feedback Detail of a Feedback object |
| 10 | setFeedbackDetail() | Set the Feedback Detail of a Feedback object |
| 11 | getListReplies () | Get the list of Replies of a Feedback object |
| 12 | setListReplies () | Set the list of Replies of a Feedback object |
| 13 | toString() | Convert all information of a Feedback object to a string |

### c. Sequence Diagram(s)



### d. Database queries

SELECT od.id, od.order\_id, od.productId, od.productName,od.quantity, od.productPrice, p.imageLink

FROM Order\_Detail od INNER JOIN Product p

ON od.ProductID = p.ProductID

WHERE od.Order\_ID = ?

## 18. Notifications

### a. Class Diagram

### b. Class Specifications

***ViewAllNotifications***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the data from the database and dump it on the view all notifications page to display the notification’s user interface* |

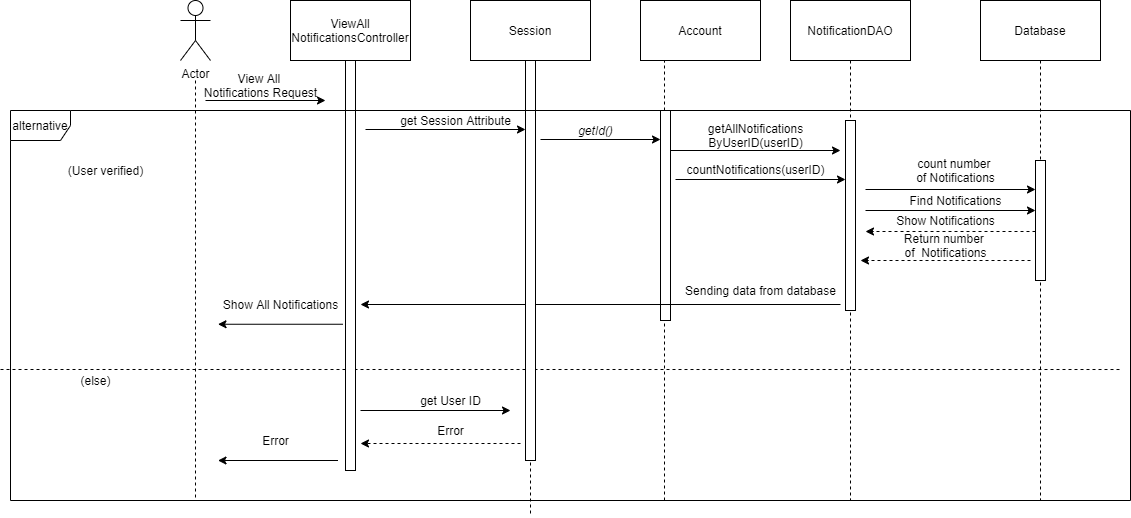
***NotificationDAO***

***Class Methods***

This class performs CRUD with Notifications table in databse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *countNotifications(userId)* | *Get number of notifications received from a specific user* |
| *02* | *getAllNotificationsByUserID(id)* | *Get all notifications of an user by their ID* |
| *03* | *approveOrderAdminNoti(userId, orderId)* | *Send a notification to a specific user that their order has been approved* |
| *04* | *finishOrderUserNoti(userId, orderId)* | *Send a notification to admin that the customer order has been shipped successfully* |
| *05* | *deliverOrderAdminNoti(userId, orderId)* | *Send a notification to a specific user that their order has been packaged and now being delivered* |
| *06* | *userReceivedOrderNoti(userId, orderId, <adminSellerId>)* | *Inform to the seller and admin that the order has been paid successfully!* |
| *07* | *userBuyNoti(int userId, int orderId, <sellerId>)* | *Inform to seller of the product and admin that an customer has placed an order* |
| *08* | *userBuyNotiAdmin(userId, orderId)* | *Inform to admin that an customer has placed an order* |
| *09* | *cancelOrderNoti(userId, OrderId)* | *Adding a notification to user by admin that their order has been canceled* |
| *10* | *getTop5NotificationsByUserID(userId)* | *Get 5 most recently notifications of an user* |
| *11* | *getAllNotificationsByUserID(userID)* | *Get all notifications of an user* |
| *12* | *getUnreadNotificationsByUserID(userID)* | *Get all unread notifications of an user* |
| *13* | *read(userId)* | *Set all notifications to status read* |
| *14* | *readOneNoti(userId, orderID)* | *Set a specific notifications of an user to read* |
| *15* | *countUnreadNotifications(userId)* | *Count unread notifications of an user* |
| *16* | *countNotifications(userId)* | *Count all notifications of an user* |
| *17* | *adminApproveNotiSeller(userId, orderId, <sellerId>)* | *Inform seller of the product that their product has been placed in an order* |
| *18* | *userReceivedNotiSeller(userId, orderId, <sellerID>)* | *Inform seller of the product that their product has been purchased successfully* |

### c. Sequence Diagram(s)



### d. Database queries

+ SELECT COUNT(\*) FROM Notifications WHERE userId=?

+ SELECT \* FROM Notifications WHERE userID=? ORDER BY id DESC

## 19. View All Invocies

### a. Class Diagram

### b. Class Specifications

***ViewAllInvoiceAdmin***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the data from the database and dump it on the view all invoices admin page to display all invoices with user interface* |

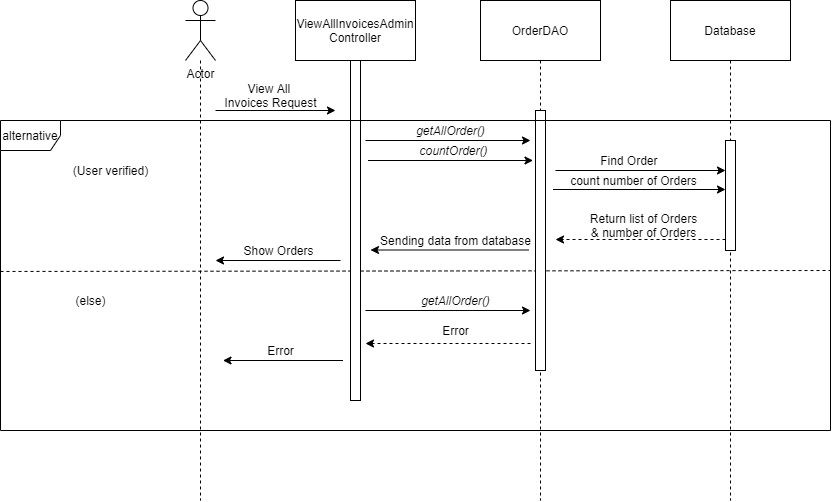
***OrderDAO***

***Class Methods***

This class performs CRUD with Order table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | adding a new order to database, return the order Id |
| 4 | packaging() | change the status of an specific order by the order id |
| 5 | delivering() | … |
| 6 | cancled() | … |
| 7 | countOrders() | Count all the number of orders in database and return integer number |
| 8 | getOrderByOrderID() | get a particular order by its id and return Order |
| 9 | getNewestOrderID() | … |
| 10 | getOrderByUserID() | get list of orders of a specific user and return List Order |
| 11 | Delete() | delete a specific order by the order id |

### 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

+ SELECT COUNT(\*) FROM Orders

## 20. View Invoice Detail

### a. Class Diagram

### b. Class Specifications

***ViewAllInvoiceAdmin***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the data from the database and dump it on the view invoice detail admin page to display detailed invoice with user interface* |

***NotificationDAO***

***Class Methods***

This class performs CRUD with Notifications table in databse

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *countNotifications(userId)* | *Get number of notifications received from a specific user* |
| *02* | *getAllNotificationsByUserID(id)* | *Get all notifications of an user by their ID* |
| *03* | *approveOrderAdminNoti(userId, orderId)* | *Send a notification to a specific user that their order has been approved* |
| *04* | *finishOrderUserNoti(userId, orderId)* | *Send a notification to admin that the customer order has been shipped successfully* |
| *05* | *deliverOrderAdminNoti(userId, orderId)* | *Send a notification to a specific user that their order has been packaged and now being delivered* |
| *06* | *userReceivedOrderNoti(userId, orderId, <adminSellerId>)* | *Inform to the seller and admin that the order has been paid successfully!* |
| *07* | *userBuyNoti(int userId, int orderId, <sellerId>)* | *Inform to seller of the product and admin that an customer has placed an order* |
| *08* | *userBuyNotiAdmin(userId, orderId)* | *Inform to admin that an customer has placed an order* |
| *09* | *cancelOrderNoti(userId, OrderId)* | *Adding a notification to user by admin that their order has been canceled* |
| *10* | *getTop5NotificationsByUserID(userId)* | *Get 5 most recently notifications of an user* |
| *11* | *getAllNotificationsByUserID(userID)* | *Get all notifications of an user* |
| *12* | *getUnreadNotificationsByUserID(userID)* | *Get all unread notifications of an user* |
| *13* | *read(userId)* | *Set all notifications to status read* |
| *14* | *readOneNoti(userId, orderID)* | *Set a specific notifications of an user to read* |
| *15* | *countUnreadNotifications(userId)* | *Count unread notifications of an user* |
| *16* | *countNotifications(userId)* | *Count all notifications of an user* |
| *17* | *adminApproveNotiSeller(userId, orderId, <sellerId>)* | *Inform seller of the product that their product has been placed in an order* |
| *18* | *userReceivedNotiSeller(userId, orderId, <sellerID>)* | *Inform seller of the product that their product has been purchased successfully* |

***OrderDAO***

***Class Methods***

This class performs CRUD with Order table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllOrder() | Get all the current orders in database and return list order |
| 2 | add() | Adding a new order to database |
| 3 | addOrder() | adding a new order to database, return the order Id |
| 4 | packaging() | change the status of an specific order by the order id |
| 5 | delivering() | … |
| 6 | cancled() | … |
| 7 | countOrders() | Count all the number of orders in database and return integer number |
| 8 | getOrderByOrderID() | get a particular order by its id and return Order |
| 9 | getNewestOrderID() | … |
| 10 | getOrderByUserID() | get list of orders of a specific user and return List Order |
| 11 | Delete() | delete a specific order by the order id |

***InvoiceDAO***

***Class Methods***

This class performs CRUD with Order table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getInvoiceDetailBy  OrderID(orderId) | Get detailed invoice of an order ID |
| 2 | getInvoiceDetailBySellerID(sellerID) | Get all sold products of a seller |
| 3 | countProductSoldOfSeller(sellerID) | Count number of product sold of a seller |

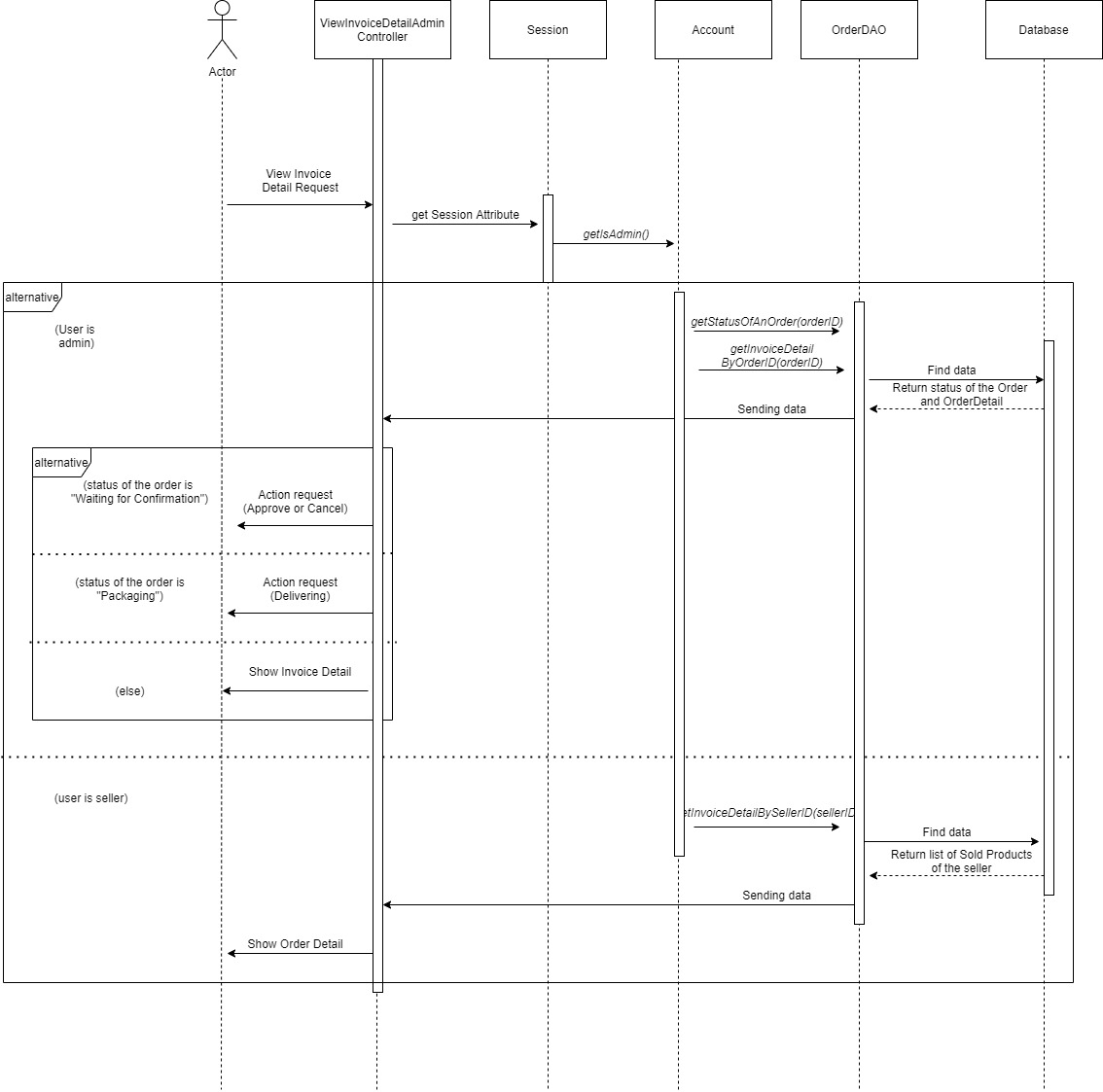
***Order***

***Class Methods***

This class represents Order entity

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Order object |
| 2 | setId() | Set the ID of an Order object |
| 3 | getUserId() | Get the User ID of an Order object |
| 4 | setUserId() | Set the User ID of an Order object |
| 5 | getTotalPrice() | Get the Subtotal of an Order object |
| 6 | setTotalPrice () | Set the Subtotal of an Order object |
| 7 | getNote () | Get the Note of an Order object |
| 8 | setNote() | Set the Note of an Order object |
| 9 | getStatus () | Get the Status of an Order object |
| 10 | setStatus() | Set the Status of an Order object |
| 11 | getDate() | Get the Order Date of an Order object |
| 12 | setDate() | Set the Order Date of an Order object |

### c. Sequence Diagram(s)



### d. Database queries

+ SELECT status FROM Orders WHERE ID=?

+ SELECT COUNT(\*) FROM Orders

+ SELECT od.id,u.UserId, od.Order\_ID, u.username,od.productId,

p.productName,p.imageLink,p.SellPrice,od.quantity,

si.shippingAddress, si.phoneNum

FROM Order\_Detail od INNER JOIN Orders o

ON od.Order\_ID = o.ID

INNER JOIN Users u

ON o.UserID = u.UserID

INNER JOIN Product p

ON od.ProductID = p.ProductID

INNER JOIN ShipInfo si

ON o.ID = si.Order\_ID

WHERE od.Order\_ID = ?

ORDER BY od.ID DESC

+ SELECT o.ID,o.UserId,o.TotalPrice, o.Note, o.DayBuy, os.Name

FROM Orders o INNER JOIN Order\_Status os

ON o.Status = os.ID

WHERE o.ID = ?

+ SELECT od.id,u.userId, od.Order\_ID, u.username,od.productId,

p.productName,p.imageLink,

p.SellPrice, si.shippingAddress,

si.phoneNum, od.quantity

FROM Order\_Detail od INNER JOIN Orders o

ON od.Order\_ID = o.ID

INNER JOIN Users u

ON o.UserID = u.UserID

INNER JOIN Product p

ON od.ProductID = p.ProductID

INNER JOIN ShipInfo si

ON o.ID = si.Order\_ID

WHERE p.SellerID = ? and o.status <> 1 and o.Status<>4

ORDER BY od.ID DESC

+ SELECT COUNT(\*) FROM Order\_Detail od INNER JOIN Orders o

ON od.Order\_ID = o.ID

INNER JOIN Users u

ON o.UserID = u.UserID

INNER JOIN Product p

ON od.ProductID = p.ProductID

INNER JOIN ShipInfo si

ON o.ID = si.Order\_ID

WHERE p.SellerID = ? and o.status <> 1 and o.Status<>4

+ UPDATE Notifications SET Status = 'read' WHERE userID = ? AND orderID=?

## 21. Compare

### a. Class Diagram

### b. Class Specifications

***Compare***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the first chosen product and send it to the comparing page with user interface* |

***ProductDAO***

***Class Methods***

This class performs CRUD with Product table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllProduct() | … |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | Edit() | Edit the information of a particular product, select by product ID |
| 6 | Delete() | Delete a product, select by its ID |
| 7 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 8 | countProduct() | return the total number of products |
| 9 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 10 | pagingByCategory() | categorizing products and return list of products with same category |
| 11 | pagingManager  Product() | … |
| 12 | countProductBy  Category() | count total product |
| 13 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 14 | getProductByID() | Get data of Product Table searched by UserID |
| 15 | getProductDetailBy  ID() | Get Product for Detail |
| 16 | getProductForManager() | Get a particular product and return a product |
| 17 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 18 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 19 | top3MostSell() | Select top 3 most sold products and return product |
| 20 | top3LeastSell() | Select top 3 least sold products and return product |

***InforDAO***

***Class Methods***

This class interact with Information table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getInfor() | Return information of the administrator's place (There's only one) |

***CategoryDAO***

***Class Methods***

This class interact with Category table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllCategory() | Return all list Category |
| 2 | getCateNameByID() | Select a specified category by its ID, used for categorizing products in homepage |

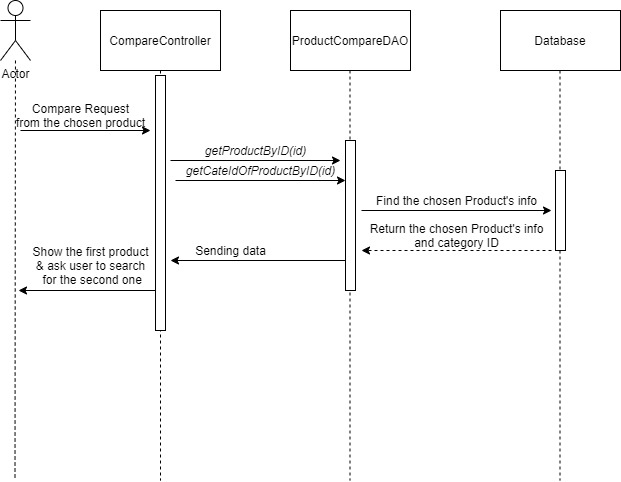
***ProductCompareDAO***

***Class Methods***

This class interact with Product table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getProductByID(id) | Return information of the product |

### c. Sequence Diagram(s)



### d. Database queries

+ SELECT TOP 1 \* FROM Product

ORDER BY Amount DESC

+ SELECT CategoryId FROM Product WHERE ProductID = ?

+ SELECT TOP 2 \* FROM Product

ORDER BY Amount DESC

+ SELECT \* FROM Information

+ SELECT \* FROM Product WHERE ProductID = ?

+ SELECT \* FROM Product

+ SELECT \* FROM Category

## 22. CompareFinal

### a. Class Diagram

### b. Class Specifications

***Compare***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest* | *Get the second chosen product and send it to the comparing page with user interface* |

***ProductDAO***

***Class Methods***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getAllProduct() | … |
| 2 | getHotProduct() | return Product with the highest sold amount |
| 3 | getFavoriteProduct  () | return get the first and the second product with most sold amount |
| 4 | getProductBy  SellID() | return the list of products of a particular seller |
| 5 | Edit() | Edit the information of a particular product, select by product ID |
| 6 | Delete() | Delete a product, select by its ID |
| 7 | Add() | Adding a new product to the database, the ID is no need to be added because it is automatically added |
| 8 | countProduct() | return the total number of products |
| 9 | pagingProduct() | Every 6 products will be displayed in a single page and return list of 6 products |
| 10 | pagingByCategory() | categorizing products and return list of products with same category |
| 11 | pagingManager  Product() | … |
| 12 | countProductBy  Category() | count total product |
| 13 | countProductBy  Seller() | Count the products of a particular seller and return the number of product |
| 14 | getProductByID() | Get data of Product Table searched by UserID |
| 15 | getProductDetailBy  ID() | Get Product for Detail |
| 16 | getProductForManager() | Get a particular product and return a product |
| 17 | SearchProductBy  Name() | Search for a particular product by filling its name |
| 18 | searchProductIn  Manager() | Search for a product in admin/moderator mode by filling its name |
| 19 | top3MostSell() | Select top 3 most sold products and return product |
| 20 | top3LeastSell() | Select top 3 least sold products and return product |

***InforDAO***

***Class Methods***

This class interact with Information table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getInfor() | Return information of the administrator's place (There's only one) |

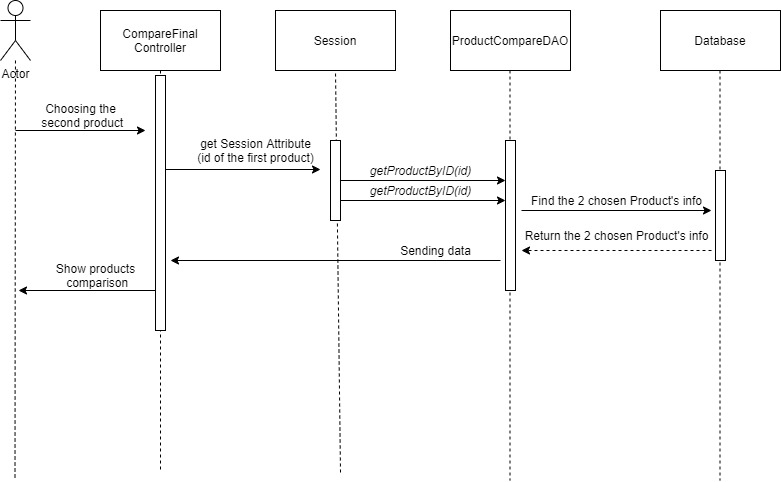
***ProductCompareDAO***

***Class Methods***

This class interact with Product table in database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getProductByID(id) | Return information of the product |

### c. Sequence Diagram(s)



### d. Database queries

+ SELECT TOP 1 \* FROM Product

ORDER BY Amount DESC

+ SELECT TOP 2 \* FROM Product

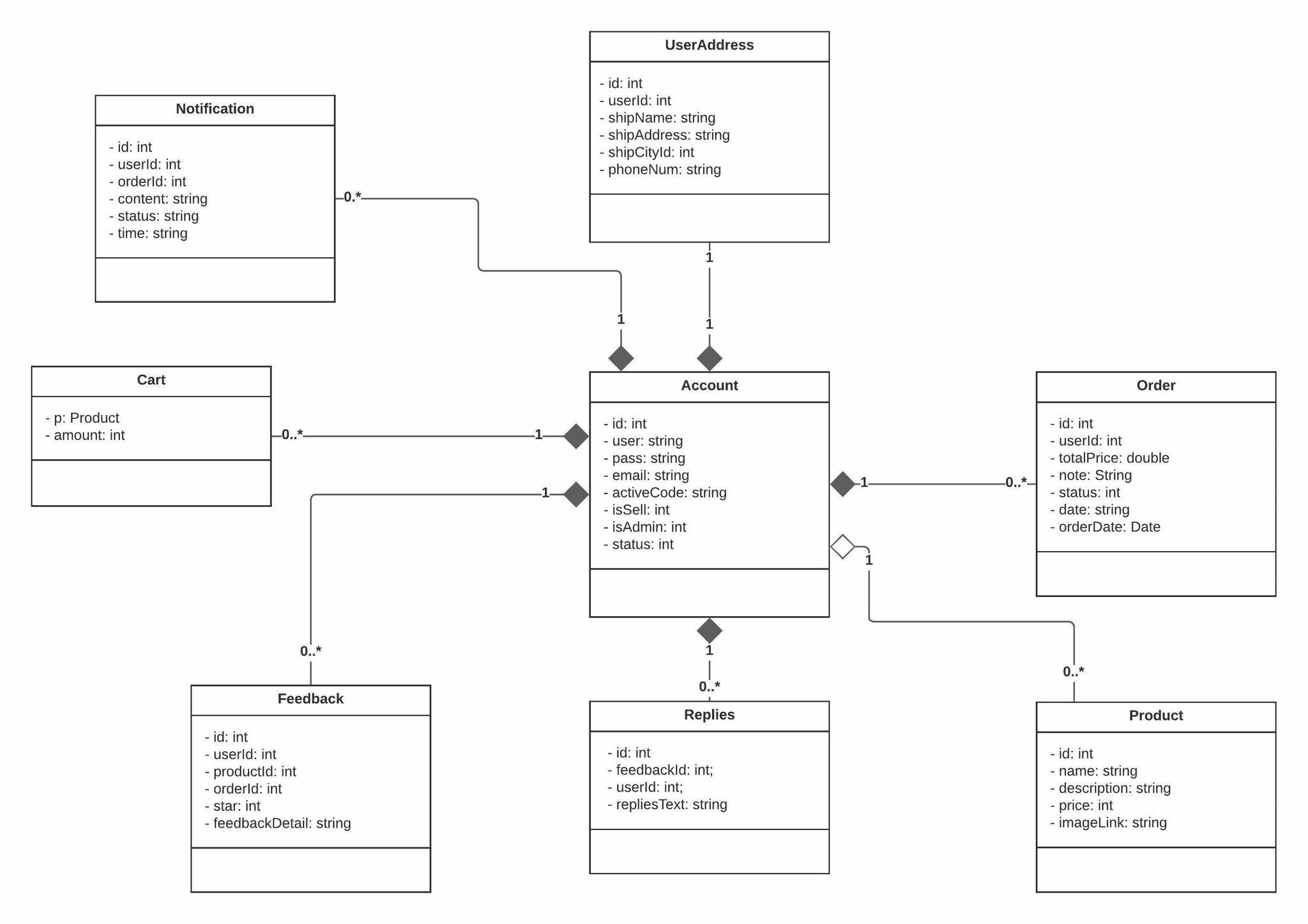
ORDER BY Amount DESC

+ SELECT \* FROM Information

+ SELECT \* FROM Product WHERE ProductID = ?

## 24. User List

### a. Class Diagram



### b. Class Specifications

#### AccountManagerControl Class: This class process request for all list in Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Select all list of Account in SQL server and return AccountManager.jsp |
| 2 | doGet() | Nothing |
| 3 | doPost() | Nothing |
| 4 | getServletInfo() | Return a String containing servlet description |

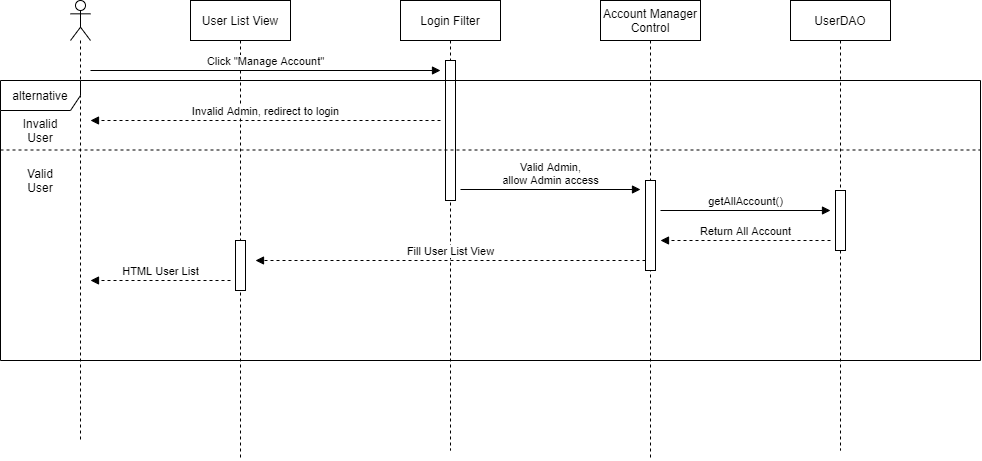
#### UserDAO Class

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | Login() | For login an account to the website and return the account information of the account logging in, null if username or password does not exist |
| 2 | signUp() | Use for signup an account to the website |
| 3 | signUpFB() | Used when user try to login via FB, the password will be set automatically and the user will be added to database |
| 4 | getAllAccounts() | Get all account from database and return the list of account from the database |
| 5 | deleteAccount() | Delete an account from the database |
| 6 | editAccount() | Update the information of an account |
| 7 | updatePassword() | Update the password of a user |
| 8 | getAccountByID() | Find an account by id and eturn the found account, null if the account does not exist in the database |
| 9 | getAccountByEmail  () | Find an account by email return the found account, null if the account does not exist in the database |
| 10 | getAccountBy  Username() | Find an account by username and return the found account, null if the account does not exist in the database |
| 11 | updateStatus() | Update status of an account and return true if update successful |
| 12 | countAllAccount() | Return the number of account in the database |

#### Account Class: This class stores information of a Account

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of an Account |
| 2 | setId() | Set the ID of an Account |
| 3 | getUser() | Get the Username of an Account |
| 4 | setUser() | Set the Username of an Account |
| 5 | getEmail() | Get the Email of an Account |
| 6 | setEmail() | Set the Email of an Account |
| 7 | getIsSell() | Check if an Account has the seller role |
| 8 | setIsSell() | Add/remove seller role to/from an Account |
| 9 | getIsAdmin() | Check if an Account has the admin role |
| 10 | setIsAdmin() | Add/remove admin role to/from an Account |
| 11 | getStatus() | Get the Status of an Account |
| 12 | setStatus() | Set the Status of an Account |
| 13 | toString() | Convert all information of an Account to a string |

### c. Sequence Diagram(s)

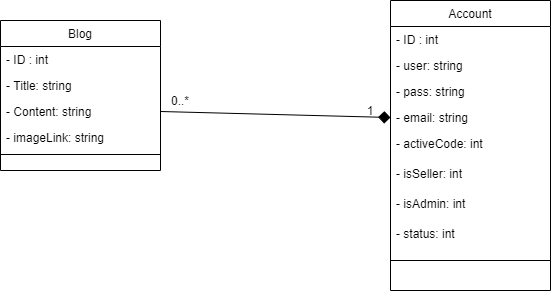


### d. Database queries

- " select \* from Users";

## 25. Blog List

### a. Class Diagram



### b. Class Specifications

#### BlogListControl Class: This class process request for all list in Blog

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Select all list of Blog in SQL server and return BlogList.jsp |
| 2 | doGet() | Nothing |
| 3 | doPost() | Nothing |
| 4 | getServletInfo() | Return a String containing servlet description |

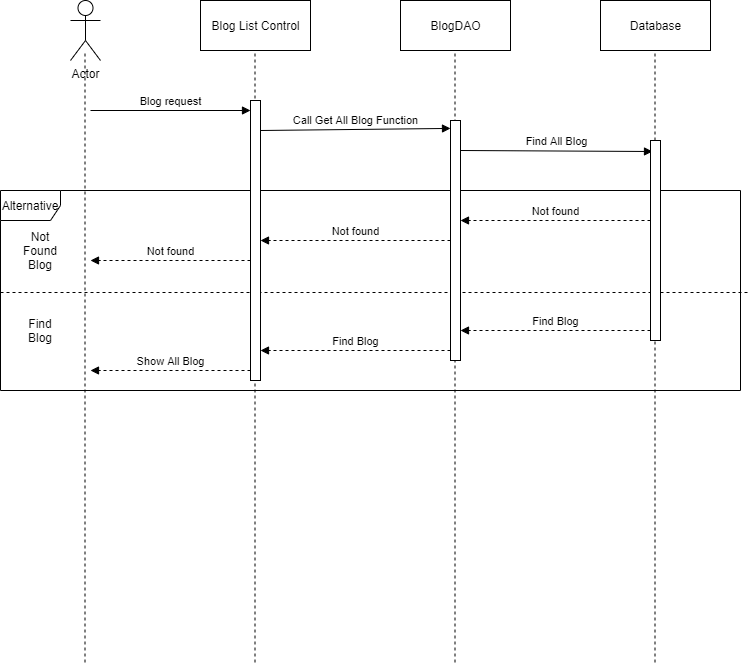
#### BlogDAO Class: This class performs CRUD actions to Blog table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getHotBlog() | Get First Blog in Database |
| 2 | getAllBlog() | Get All Blog In Database |
| 3 | getBlogByID() | Get Blog Need to View Detail |
| 4 | add() | Add new Blog |
| 5 | edit() | edit blog by ID |
| 6 | delete() | delete blog by ID |
| 7 | getBlogForManager() | Get Post List In Manage Blog For Seller |

**Blog Class: This class stores information of a Blog**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Blog |
| 2 | setId() | Set the ID of an Blog |
| 3 | getTitle() | Get the Title of a Blog |
| 4 | setTitle() | Set the Title of a Blog |
| 5 | getContent() | Get the Content of a Blog |
| 6 | setContent() | Set the Content of a Blog |
| 7 | getImageLink() | Get the ImageLink of a Blog |
| 8 | setImageLink() | Set the ImageLink of a Blog |

### c. Sequence Diagram(s)

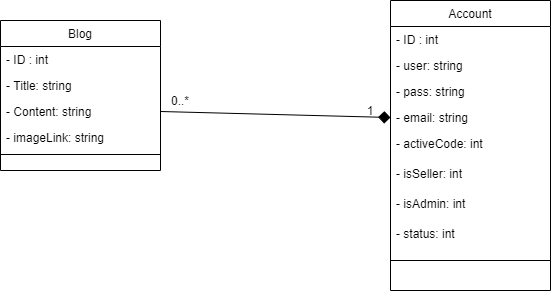


### d. Database queries

-"SELECT \* FROM Blog"

## 26. Blog Detail

### a. Class Diagram



### b. Class Specifications

#### DetailBlogControll Class: This class process request for a detail of Blog

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Get ID of Blog need to see and use function getBlogByID to view Detail of Blog in Database and push to BlogDetail.jsp |
| 2 | doGet() | Nothing |
| 3 | doPost() | Nothing |
| 4 | getServletInfo() | Return a String containing servlet description |

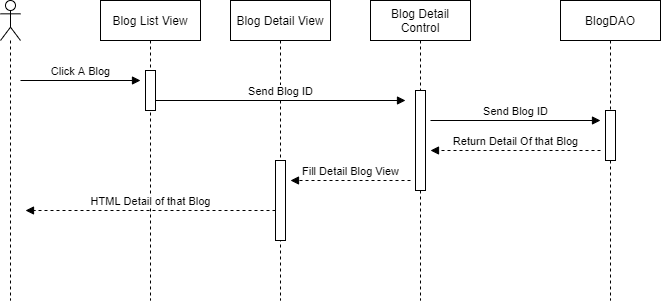
#### BlogDAO Class: This class performs CRUD actions to Blog table in the database

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getHotBlog() | Get First Blog in Database |
| 2 | getAllBlog() | Get All Blog In Database |
| 3 | getBlogByID() | Get Blog Need to View Detail |
| 4 | add() | Add new Blog |
| 5 | edit() | edit blog by ID |
| 6 | delete() | delete blog by ID |
| 7 | getBlogForManager() | Get Post List In Manage Blog For Seller |

**Blog Class: This class stores information of a Blog**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Blog |
| 2 | setId() | Set the ID of an Blog |
| 3 | getTitle() | Get the Title of a Blog |
| 4 | setTitle() | Set the Title of a Blog |
| 5 | getContent() | Get the Content of a Blog |
| 6 | setContent() | Set the Content of a Blog |
| 7 | getImageLink() | Get the ImageLink of a Blog |
| 8 | setImageLink() | Set the ImageLink of a Blog |

### c. Sequence Diagram(s)

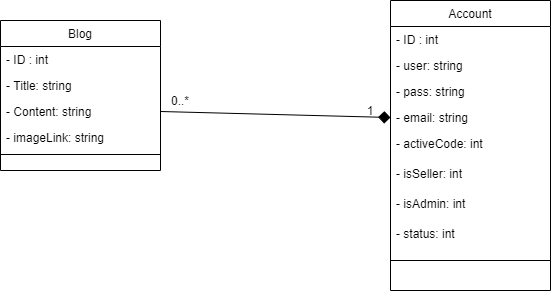


### d. Database queries

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

## 27. Post List

### a. Class Diagram



### b. Class Specifications

#### BlogManagerControl Class: This class process request for list of Blog create by SellerID

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Select list Blog of Seller in SQL server and return ManageBlog.jsp |
| 2 | doGet() | Nothing |
| 3 | doPost() | Nothing |
| 4 | getServletInfo() | Return a String containing servlet description |

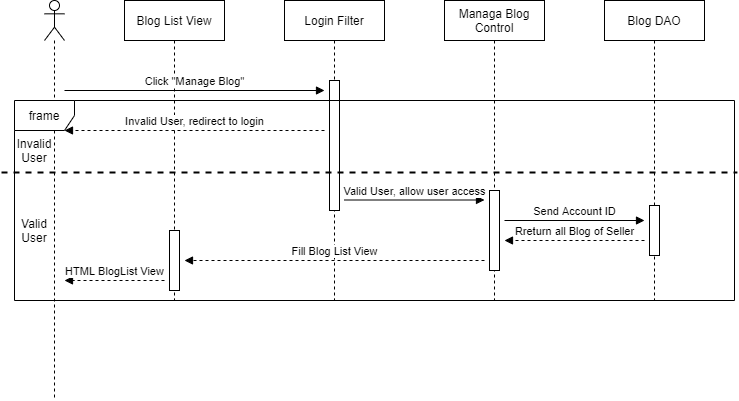
**BlogDAO Class: : This class performs CRUD actions to Blog table in the database**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getHotBlog() | Get First Blog in Database |
| 2 | getAllBlog() | Get All Blog In Database |
| 3 | getBlogByID() | Get Blog Need to View Detail |
| 4 | add() | Add new Blog |
| 5 | edit() | edit blog by ID |
| 6 | delete() | delete blog by ID |
| 7 | getBlogForManager() | Get Post List In Manage Blog For Seller |

**BlogInManager Class: This class stores information of a Blog create by Seller**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Blog |
| 2 | setId() | Set the ID of an Blog |
| 3 | getTitle() | Get the Title of a Blog |
| 4 | setTitle() | Set the Title of a Blog |
| 5 | getContent() | Get the Content of a Blog |
| 6 | setContent() | Set the Content of a Blog |
| 7 | getImageLink() | Get the ImageLink of a Blog |
| 8 | setImageLink() | Set the ImageLink of a Blog |
| 9 | getSellerID() | Get the ID of Seller to Create Blog |
| 10 | setSellerID() | Set the ID of Seller to Create Blog |

### c. Sequence Diagram(s)

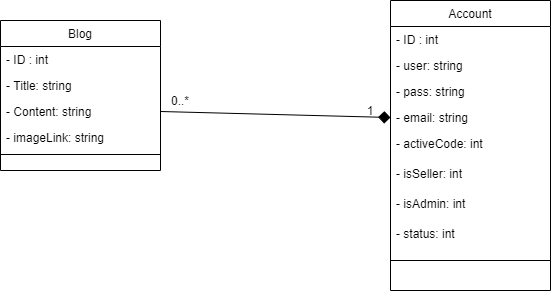


### d. Database queries

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

## 28. Post Detail

### a. Class Diagram



### b. Class Specifications

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

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Nothing |
| 2 | doGet() | Get ID of Blog and push ID, title, content, imageLink, Seller ID to EditBlog.jsp to edit |
| 3 | doPost() | Get ID, title, content, imageLink, Seller ID in EditBlog.jsp when changed, use edit function in BlogDAO to edit blog and comback BlogManageControl Class |
| 4 | getServletInfo() | Return a String containing servlet description |

#### DeleteBlogControll Class: use process request function to get and delete blogs by ID

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

#### AddBlogControll Class: Use the process request function to add more Blogs

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | processRequest() | Get title, content, imageLink, SellerID and use function add in Blog DAO to create new Blog In Database |
| 2 | doGet() | Nothing |
| 3 | doPost() | Nothing |
| 4 | getServletInfo() | Return a String containing servlet description |

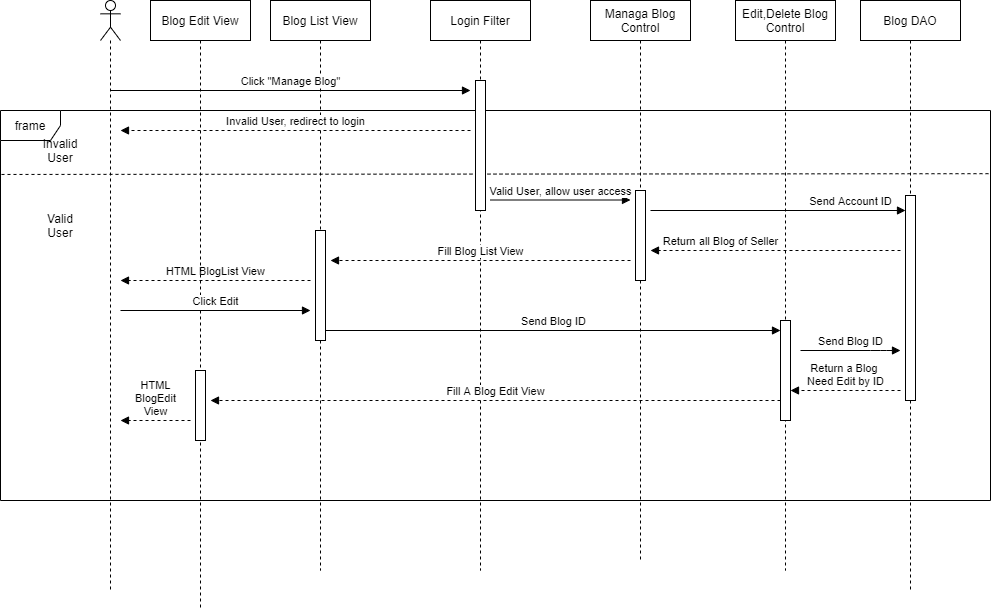
**BlogDAO Class: This class performs CRUD actions to Blog table in the database**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getHotBlog() | Get First Blog in Database |
| 2 | getAllBlog() | Get All Blog In Database |
| 3 | getBlogByID() | Get Blog Need to View Detail |
| 4 | add() | Add new Blog |
| 5 | edit() | edit blog by ID |
| 6 | delete() | delete blog by ID |
| 7 | getBlogForManager() | Get Post List In Manage Blog For Seller |

**BlogInManager Class: This class stores information of a Blog create by Seller**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getId() | Get the ID of a Blog |
| 2 | setId() | Set the ID of an Blog |
| 3 | getTitle() | Get the Title of a Blog |
| 4 | setTitle() | Set the Title of a Blog |
| 5 | getContent() | Get the Content of a Blog |
| 6 | setContent() | Set the Content of a Blog |
| 7 | getImageLink() | Get the ImageLink of a Blog |
| 8 | setImageLink() | Set the ImageLink of a Blog |
| 9 | getSellerID() | Get the ID of Seller to Create Blog |
| 10 | setSellerID() | Set the ID of Seller to Create Blog |

### c. Sequence Diagram(s)



### d. Database queries

- "Delete FROM Blog WHERE ID = ?"

- "Update Blog SET Title =?, Content=?, imageLink=?, SellerID= ? Where ID =?""

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

# III. Database Tables

## 1. User Status

*Store the status of an Account*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | StatusName | nvarchar | 1000 |  |  |  |  |

## 2. Users

*Store the information of all users*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | UserID | int |  |  | Yes | PK |  |
| 2 | Username | nvarchar | 500 |  |  |  |  |
| 3 | Password | nvarchar | 1000 |  |  |  |  |
| 4 | email | nvarchar | 1000 |  |  |  |  |
| 5 | ActiveCode | nvarchar | 11 |  |  |  |  |
| 4 | isSeller | int |  |  |  |  |  |
| 5 | isAdmin | int |  |  |  |  |  |
| 6 | StatusID | int |  |  |  | FK |  |

## 3. Category

*Store the information of all categories of products*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | CategoryID | int |  |  | Yes | PK |  |
| 2 | CategoryName | nvarchar | 1000 |  |  |  |  |
| 3 | icon | nvarchar | 1000 |  |  |  |  |

## 4. Product Status

*Store the status of a Product (out of stock or not)*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | StatusID | int |  |  | Yes | PK |  |
| 2 | StatusName | nvarchar | 1000 |  |  |  |  |

## 5. Manufacturer

*Store the information of the Manufacturer of a Product*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ManufacturerID | int |  |  | Yes | PK |  |
| 2 | Manufacturer  Name | nvarchar | 1000 |  |  |  |  |

## 6. Product

*Store the information of all products*

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | ProductID | int |  |  | Yes | PK |  |
| 2 | ProductName | nvarchar | 1000 |  |  |  |  |
| 3 | Description | nvarchar | 2000 |  |  |  |  |
| 4 | OriginalPrice | int |  |  |  |  |  |
| 5 | SellPrice | int |  |  |  |  |  |
| 6 | SalePercent | int |  |  |  |  | SalePercent>=1 AND SalePercent<=100 |
| 5 | imageLink | nvarchar | 1000 |  |  |  |  |
| 6 | CategoryID | int |  |  |  | FK |  |
| 7 | SellerID | int |  |  |  | FK |  |
| 8 | Amount | int |  |  |  |  |  |
| 9 | StatusID | int |  |  |  | FK |  |
| 10 | ManufacturerID | int |  |  |  | FK |  |
| 11 | height | float |  |  |  |  |  |
| 12 | width | float |  |  |  |  |  |
| 13 | weight | float |  |  |  |  |  |

## 7. Information

*Store some information of the shop*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | description | nvarchar | max |  |  |  |  |
| 2 | address | nvarchar | max |  |  |  |  |
| 3 | email | nvarchar | max |  |  |  |  |
| 4 | phone | nvarchar | max |  |  |  |  |
| 5 | fax | nvarchar | max |  |  |  |  |

## 8. Cart

*Store information of the Cart of each Customer*

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | UserID | int |  |  | Yes | FK |  |
| 2 | ProductID | int |  |  |  | FK |  |
| 3 | Amount | int |  |  |  |  |  |

## 9. Order\_Status

*Store the status of an Order*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | Name | nvarchar |  |  |  |  |  |

## 10. Orders

*Store information of all orders*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | UserID | int |  |  |  | FK |  |
| 3 | TotalPrice | float |  |  |  |  |  |
| 4 | Note | nvarchar | 2000 |  |  |  |  |
| 5 | Status | int |  |  |  |  |  |
| 6 | Daybuy | Datetime |  |  |  |  |  |

## 11. Order\_Detail

*Store products information of all orders*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | Order\_ID | int |  |  |  | FK |  |
| 3 | ProductID | Int |  |  |  | FK |  |
| 4 | ProductName | nvarchar | 1000 |  |  |  |  |
| 5 | ProductPrice | int |  |  |  |  |  |
| 6 | Quantity | int |  |  |  |  |  |

## 12. Ship

*Store the ship Price of each City*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | CityName | nvarchar | 1000 |  |  |  |  |
| 3 | ShipPrice | int |  |  |  |  |  |

## 13. UserAddress

*Store the Shipping Address of Customers*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | UserID | Int |  |  |  | FK |  |
| 3 | ShipName | nvarchar | 500 |  |  |  |  |
| 4 | ShipAddress | nvarchar | 1000 |  |  |  |  |
| 5 | ShipCityID | int |  |  |  | FK |  |
| 6 | PhoneNum | varchar | 20 |  |  |  |  |

## 14. ShipInfo

*Store the ship info of all orders*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | Order\_ID | int |  |  |  | FK |  |
| 3 | CustomerName | int |  |  |  |  |  |
| 4 | ShippingAddress | nvarchar | 1000 |  |  |  |  |
| 5 | ShipCityID | int |  |  |  | FK |  |
| 6 | PhoneNum | Varchar | 20 |  |  |  |  |
| 7 | Note | nvarchar | 2000 |  |  |  |  |

## 15. Notifications

*Store the notifications of a Customer order*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | UserID | int |  |  |  | FK |  |
| 3 | OrderID | int |  |  |  | FK |  |
| 4 | content | nvarchar | 1000 |  |  |  |  |

## 16. Feedback

*Store all information of Customer’s feedbacks on a Product*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | UserID | int |  |  |  | FK |  |
| 3 | ProductID | int |  |  |  | FK |  |
| 4 | OrderID | int |  |  |  | FK |  |
| 5 | Star | int |  |  |  |  | Star < 6 AND Star > 0 |
| 6 | FeedbackDetail | nvarchar | 2000 |  |  |  |  |

## 17. Feedback\_Replies

*Store all replies of each feedback*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | FeedbackID | int |  |  |  | FK |  |
| 3 | UserID | int |  |  |  | FK |  |
| 4 | RepliesText | nvarchar | 2000 |  |  |  |  |

## 18. Blog

*Store all blog posts of the system*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | ID | int |  |  | Yes | PK |  |
| 2 | Title | nvarchar | 2500 |  |  |  |  |
| 3 | Content | nvarchar | 2500 |  |  |  |  |
| 4 | imageLink | nvarchar | 2500 |  |  |  |  |
| 5 | SellerID | int |  |  |  | FK |  |