

**ONLINE SHOP**

**Software Design Document**

– Hanoi, August 2019 –

**Table of Contents**

[I. Overview 3](#_heading=h.gjdgxs)

[1. Code Packages 3](#_heading=h.1fob9te)

[2. Database Schema 3](#_heading=h.23muvy2)

[II. Code Designs 4](#_heading=h.17dp8vu)

[1. <Feature/Function Name1> 4](#_heading=h.3rdcrjn)

[a. Class Diagram 4](#_heading=h.26in1rg)

[b. Class Specifications 4](#_heading=h.lnxbz9)

[c. Sequence Diagram(s) 4](#_heading=h.35nkun2)

[d. Database queries 5](#_heading=h.1ksv4uv)

[2. <Feature/Function Name2> 5](#_heading=h.is565v)

[III. Database Tables 5](#_heading=h.1o97atn)

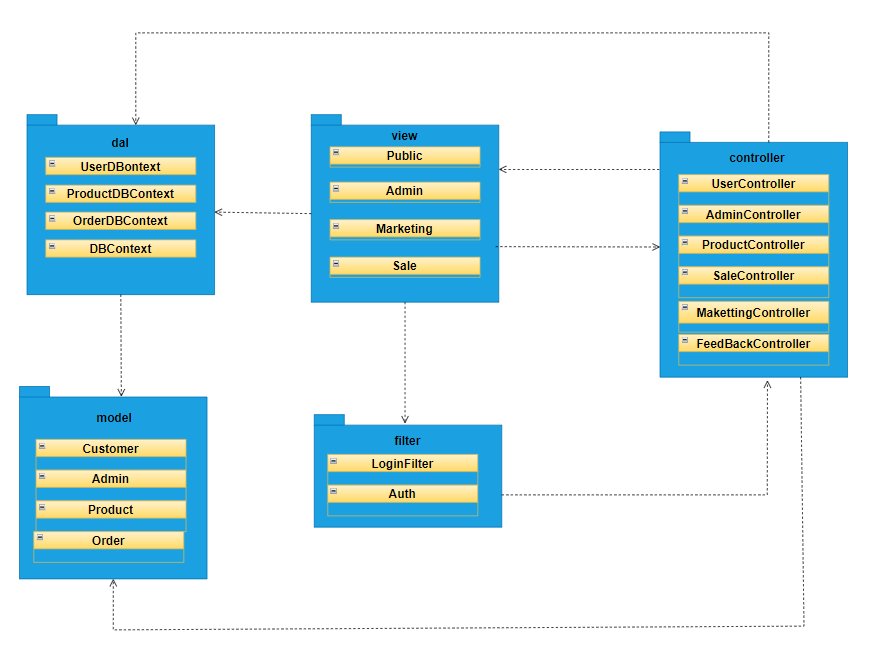
1. <Table name 1> 5

[2. <Table name 2…> 5](#_heading=h.32rsoto)

# I. Overview

## 1. Code Packages/Namespaces

## 

******

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

## 

| No | Package | Description |
| --- | --- | --- |
| 1 | controller | Acts as a go-between for the service layer and the Front-end, passing on different input requests |
| 2 | dal | This component is used to get data directly FROM the database and context component helps connect database |
| 3 | model | This component includes interfaces used to receive data retrieved FROM the database and classes used to receive data |
| 4 | view | This component displays the front-end. |
| 5 | filter | This component includes filter request classes. |

## 

## 2. Coding Conventions

* The class name should be a noun, in mixed cases the first letter of each word should be capitalized.
* Methods must be verbs, in the mixed case the first letter is lowercase and the first letter of each successive word is capitalized.
* Variable names must be meaningful, the first letter of the first word is lowercase, the first letter of the next word is uppercase.
* The name of the package is a noun, lowercase, with the meaning corresponding to the inner classes.
* Classes must be placed in a package.

## 3. Database Schema

## 

## 

# II. Code Designs

## 1. <Feature/Function Name1>

*[Provide the detailed design for the function <Feature/Function Name1>. It include Class Diagram, Class Specifications, and Sequence Diagram(s)]*

### a. Class Diagram

*[This part presents the class diagram for the relevant feature]*



### b. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### XYZ Class

*[Provide the detailed description for the class methods]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *<method name>* | *<Description of the method, including the inputs, outputs & internal method processing>* |
|  |  |  |

#### ABC Class

***Class Methods***

*[Provide the detailed description for the class methods]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *<method name>* | *<Description of the method, including the inputs, outputs & internal method processing>* |
|  |  |  |

### c. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*



### d. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 

## 

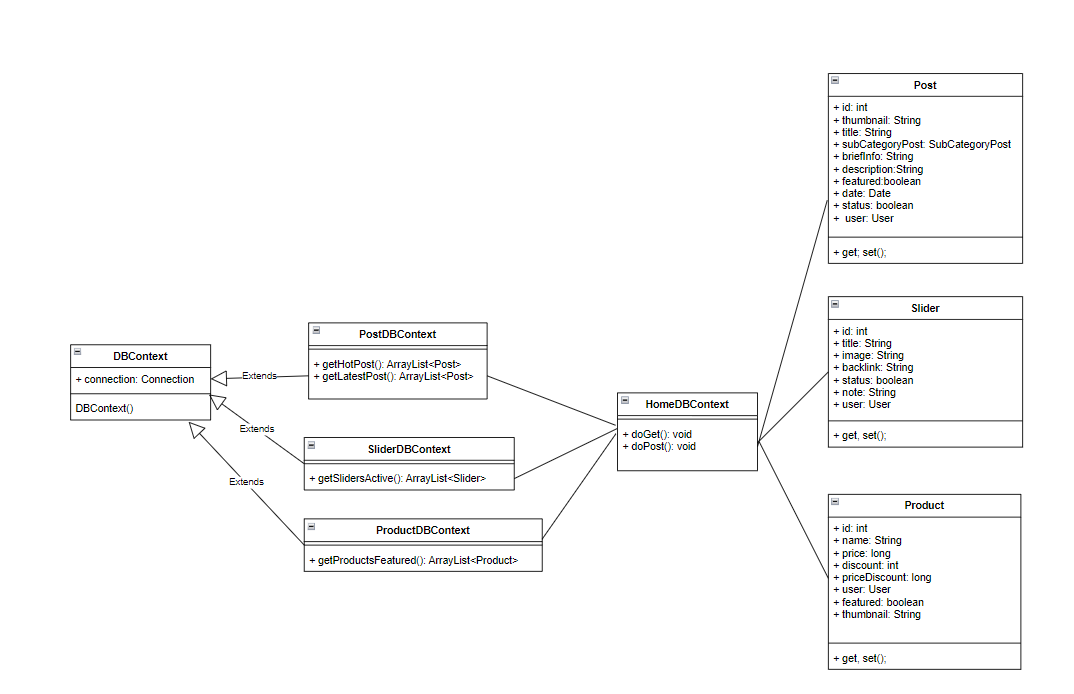
## 

## 

## 

## 1. Home page

## a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getHotPost();* | *-Get Top 3 hottest posts*  *-Return: Array Post’s information* |
| *02* | *getLatestPost();* | *- Get Top 3 Latest posts*  *- Return: Array Post’s information* |

#### 

#### SliderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSlidersActive();* | *-Get All slider have status active*  *-Return: Array Slider’s information* |

#### ProductDBContext

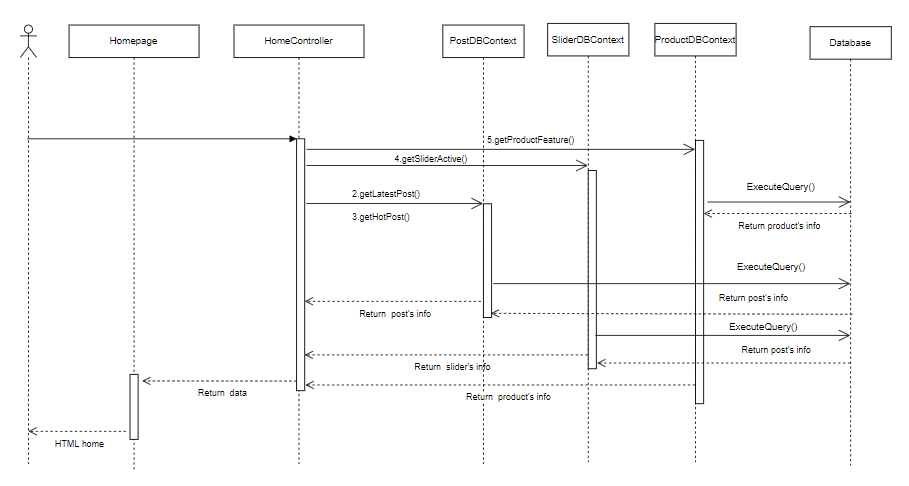
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getProductsFeature();* | *-Get Top 3 most featured products*  *-Return: Array Product’s information* |

#### HomeController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### c. Sequence Diagram(s)



### d. Database queries

* PostDBContext:
* select top 3 \* from Post

where feature = 1 and status = 1

order by dateUpdated desc

* select top 3 \* from Post

where status = 1

order by dateUpdated desc

* SliderDBContext:
* Select \* from Slider

Where status = 1

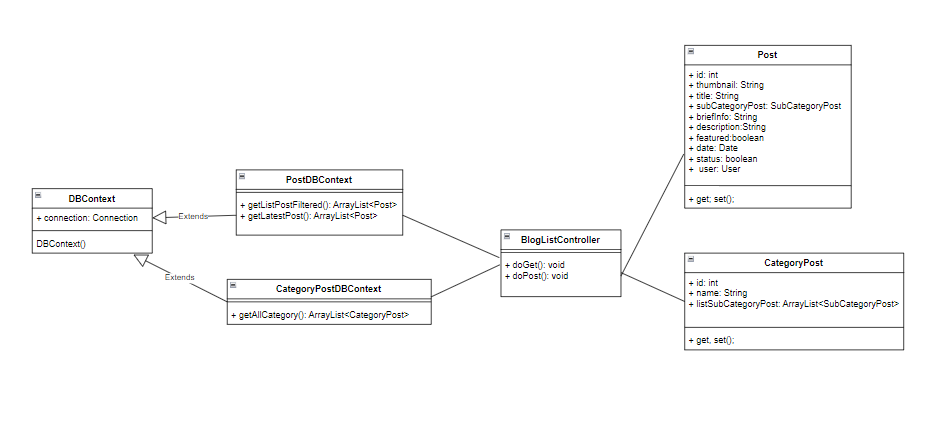
* ProductDBContext:
* Select top 3 \* from Product

where featured = 1

order by date desc

## 2. Public/Blog list

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getListPostFiltered(String searchContent, int idSubCategory, int pageIndex, int pageSize);* | *-Get Posts list have status active*  *- Input: searchContent, idSubcategory, pageIndex, pageSize*  *-Return: Array Post’s information* |
| *02* | *getLatestPost();* | *- Get Top 3 Latest posts*  *- Return: Array Post’s information* |

#### 

#### CategoryPostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategory();* | *-Get All category posts*  *-Return: Array category of posts* |

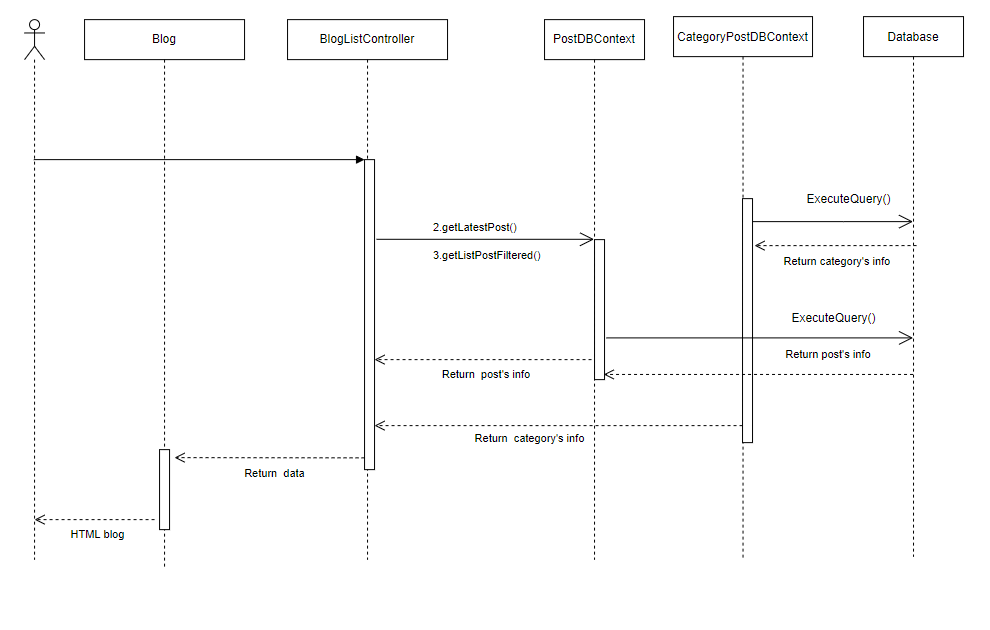
#### HomeController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)



### d. Database queries

* CategoryPostDBContext:
* Select id, name from CategoryPost
* PostDBContext:
* Select top 3 \* from Post

Where status = 1 Order by dateUpdated desc

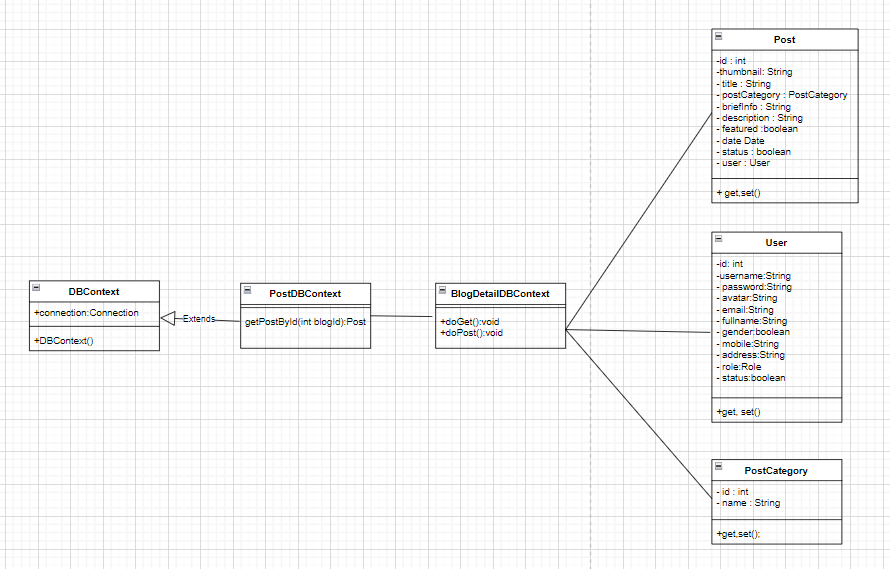
* Select \* from

(Select p.\*, c.name, c.idcategory , ROW\_NUMBER() over(order by dateUpdated DESC) as row\_index from Post p join SubCategoryPost c on p.categoryId = c.id and p.status = 1 and p.categoryId = ? and (p.title like ? or p.briefInfo like ?) PostPaging

Where row\_index >= (?-1)\*?+1 and row\_index <= ? \* ?

## 3. Public/Blog Detail

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getPostById(int blogId)* | *-Get a post’s info by id*  *-Input: blog Id*  *-Return: Post’s information* |

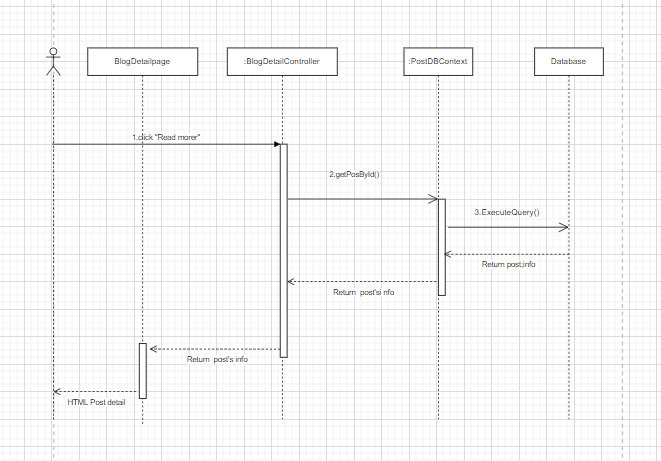
#### BlogDetailDBContext:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

SELECT Post.id ,[thumbnail] ,[title] ,[briefInfo] ,[description],[feature],[dateUpdated] ,Post.[status] ,[UserId] ,[User].fullname,PostCategory.id,PostCategory.name

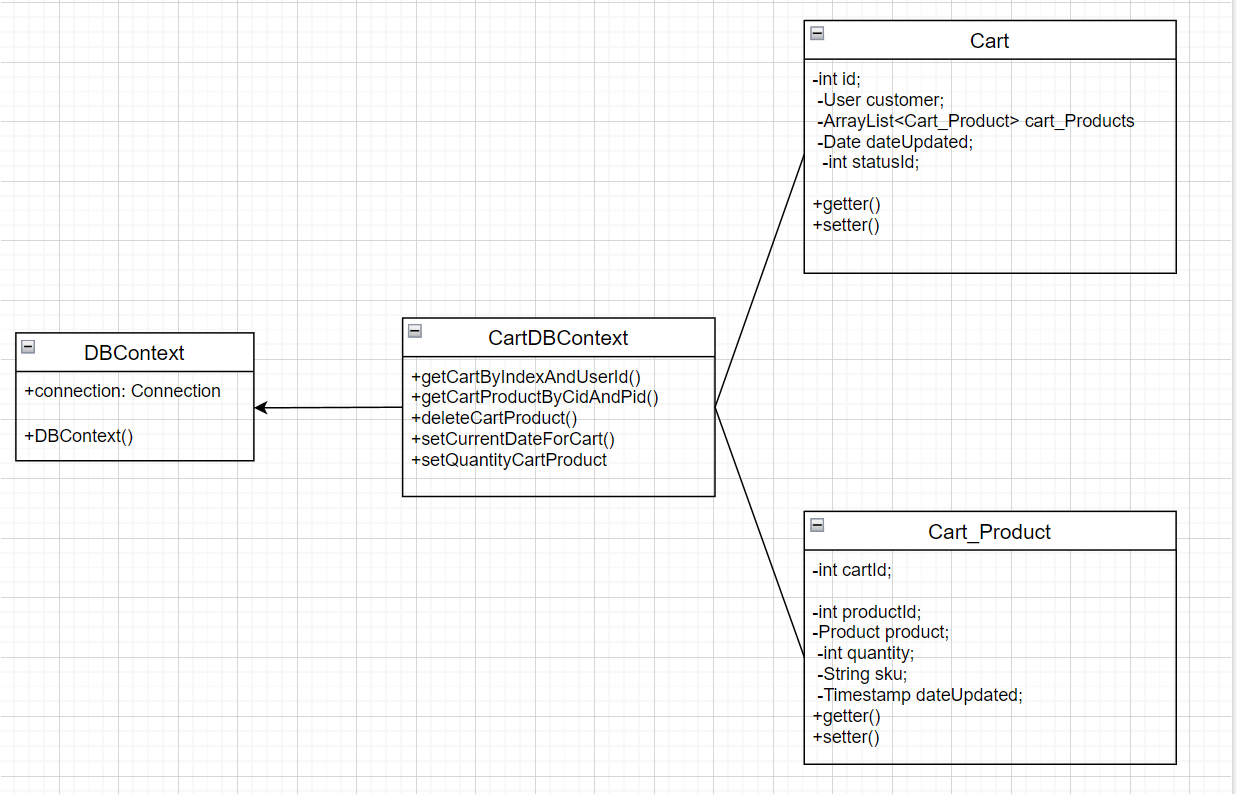
FROM [dbo].[Post] join PostCategory on Post.categoryId = PostCategory.id

join [User] on Post.UserId = [User].id

WHERE Post.id = ?

## 4. Cart details

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### CartDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCartByIndexAndUserId(int index, int sizePage, String productName, int userId)* | *Get Cart by index, sizePage, productName, userId passed to function.*  *Return : only one cart.* |
| *02* | *getCartProductByCidAndPid(int cid, int pid)* | *Get Cart Product by cart Id and product Id passed to function.*  *Return : only one cart product.* |
| *03* | *deleteCartProduct(int productId, int cartId, boolean isAll)* | *Delete Cart Product by cart Id and product Id passed to function.*  *Return : delete one cart product.* |
| *04* | *setCurrentDateForCart(int cartId)* | *Update time after editing*  *return:Returns true if successful* |
| *05* | *setQuantityCartProduct(int productId, int cartId, int number)* | *Set quantity for cart product*  *return:Returns true if successful* |

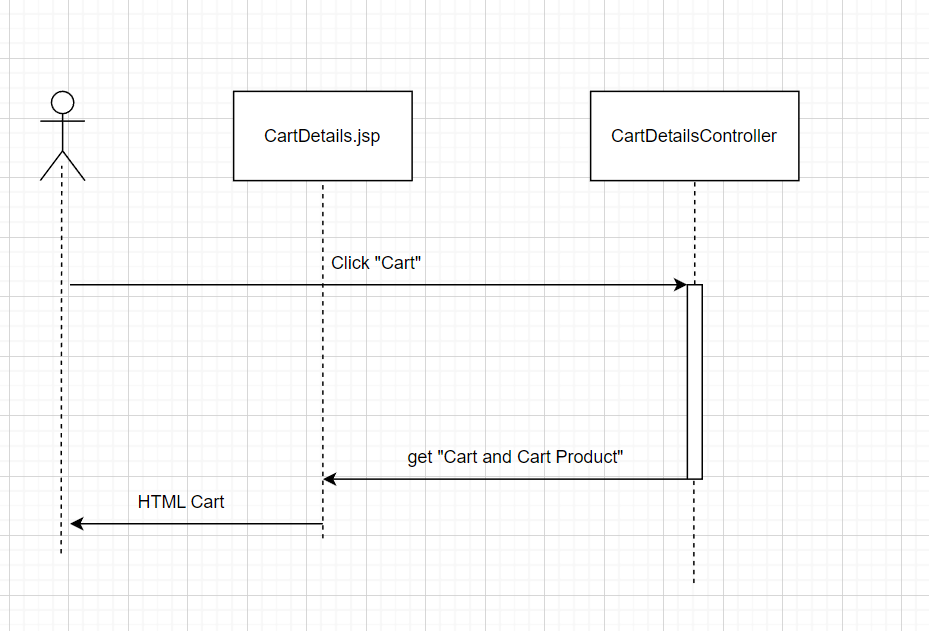
#### CartDetailsController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)



### d. Database queries

* select cp.\*,c.customerId, c.DateUpdated [CartUpdated], c.Status\_Id, [Product].quantity as product\_quantity"

from [Cart] as c join [Cart\_Product] cp on c.id = cp.cartId \n

join Product on Product.id = cp.ProductId\n

where customerId = ?

AND LOWER(productName) like '% + productName.toLowerCase() + %'

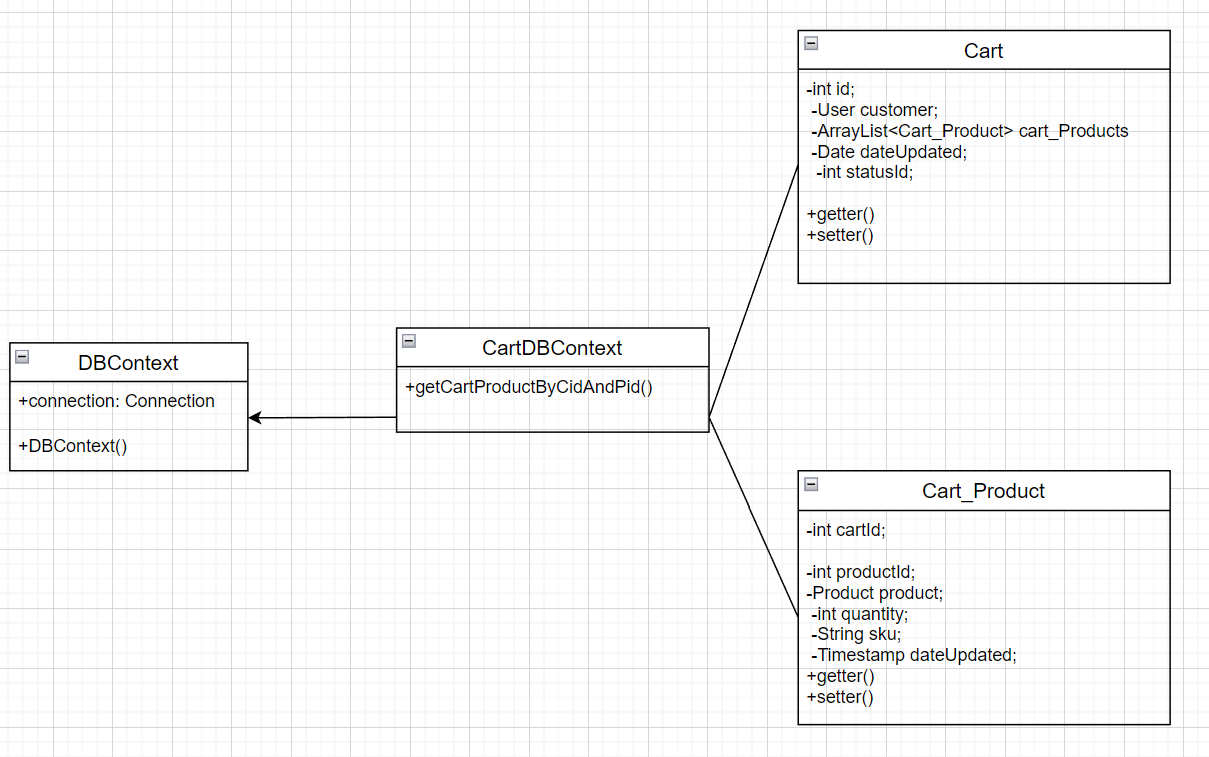
order by DateUpdated desc

offset ? rows fetch next ? rows only;

* select \* from Cart\_Product where CartId = ? And ProductId = ?
* DELETE FROM [dbo].[Cart\_Product]\n Where 1 = 1
* Update Cart set DateUpdated = ? Where Id = ?
* Update Cart\_Product set quantity = ? Where ProductId = ? And CartId = ?

## 5. Cart Contact

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCartProductByCidAndPid()* | *Get Cart Product by cart Id and product Id passed to function.*  *Return : only one cart product.* |

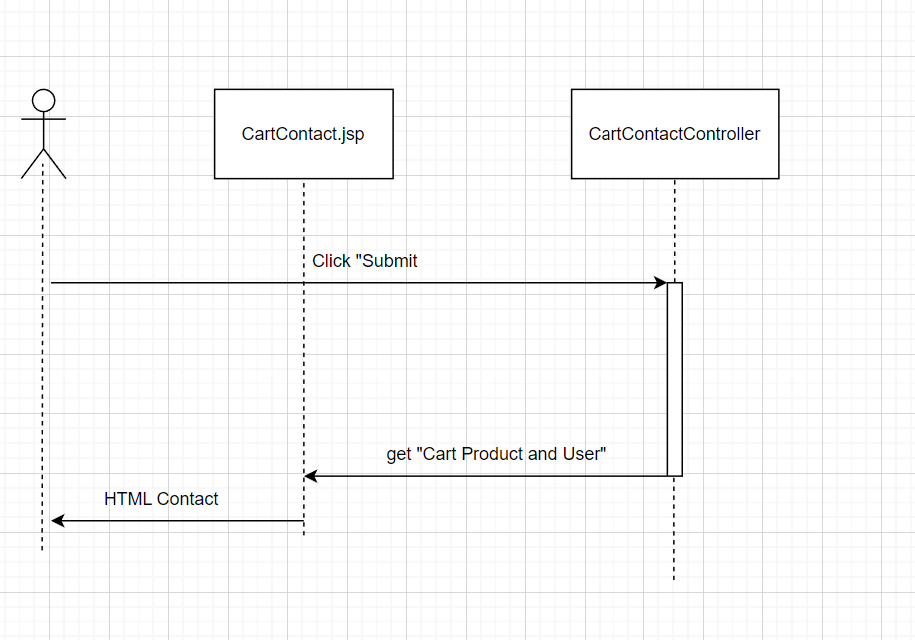
#### CartContactController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)

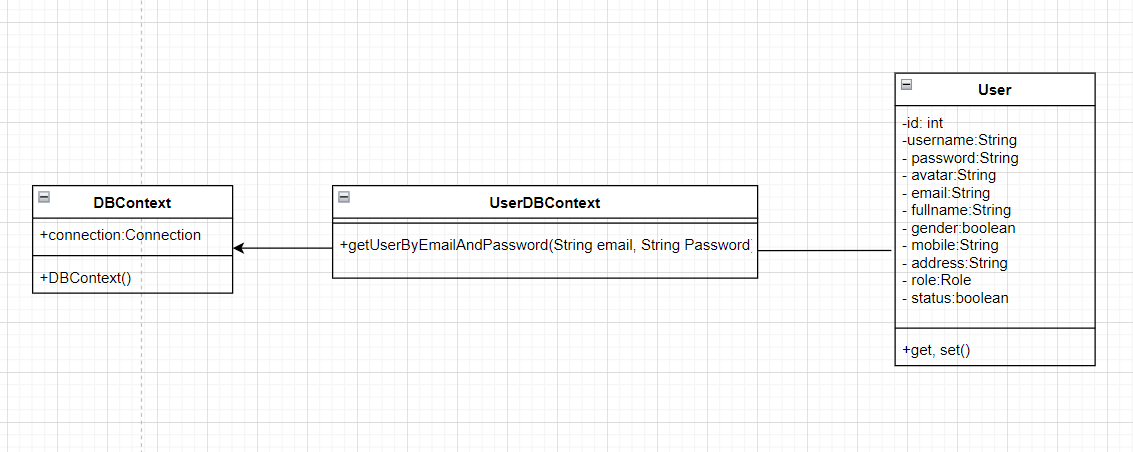


### d. Database queries

* select \* from Cart\_Product where CartId = ? And ProductId = ?

## 8. User Login

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserByEmailAndPassword(String email, String password)* | *Get User by email and password passed to function.*  *Return : only one user.* |

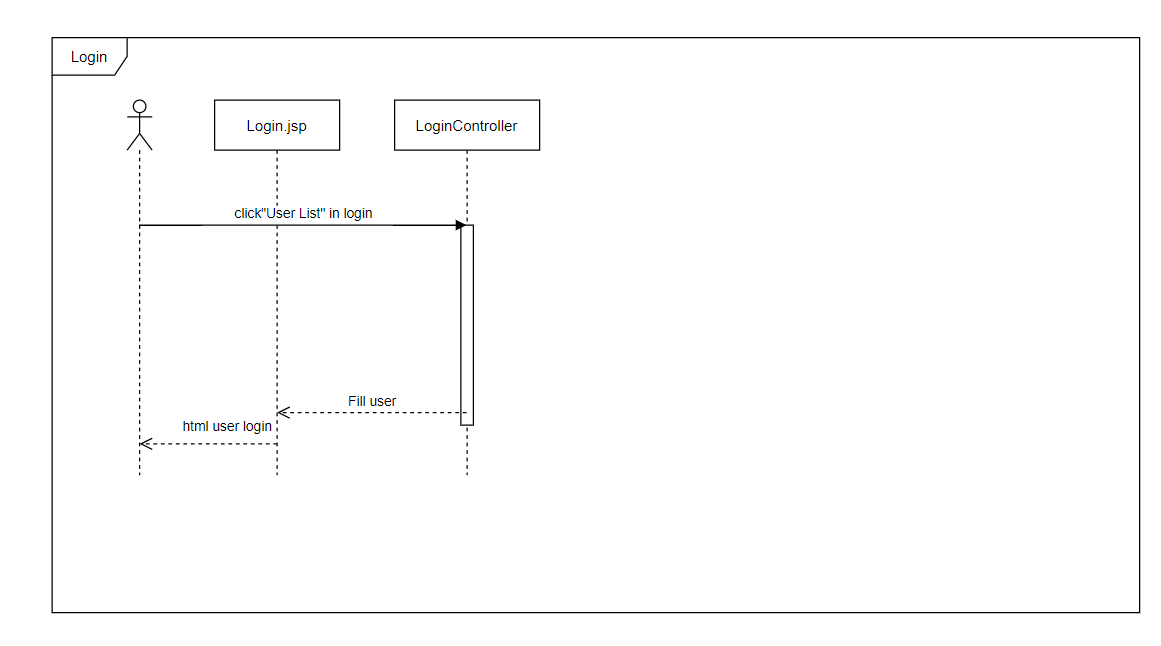
#### LoginController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)



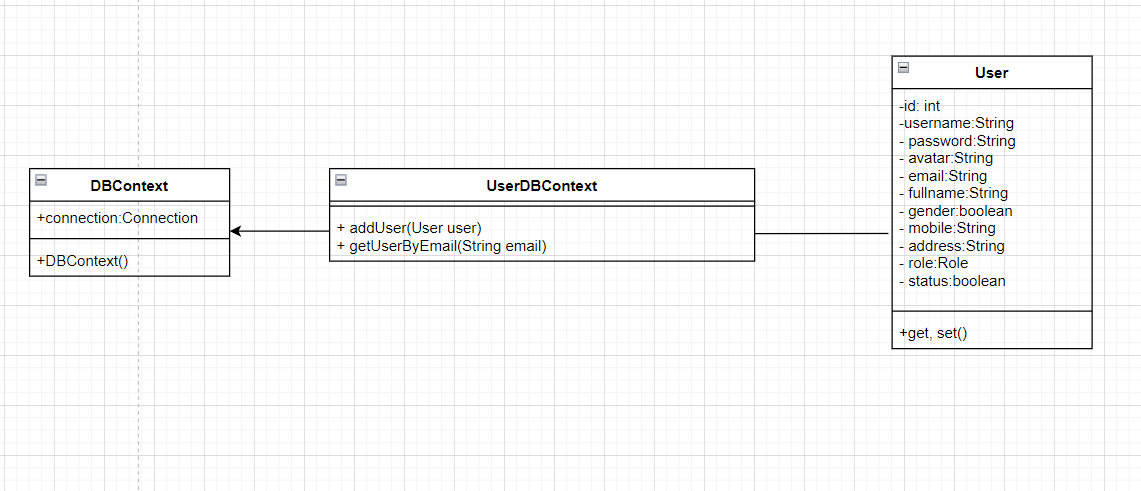
### d. Database queries

* Select u.\*, r.name rname from [User] u, [role] r

Where email = ?

## 9. User Register

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *addUser(User user)* | *Add a new user to the system.* |

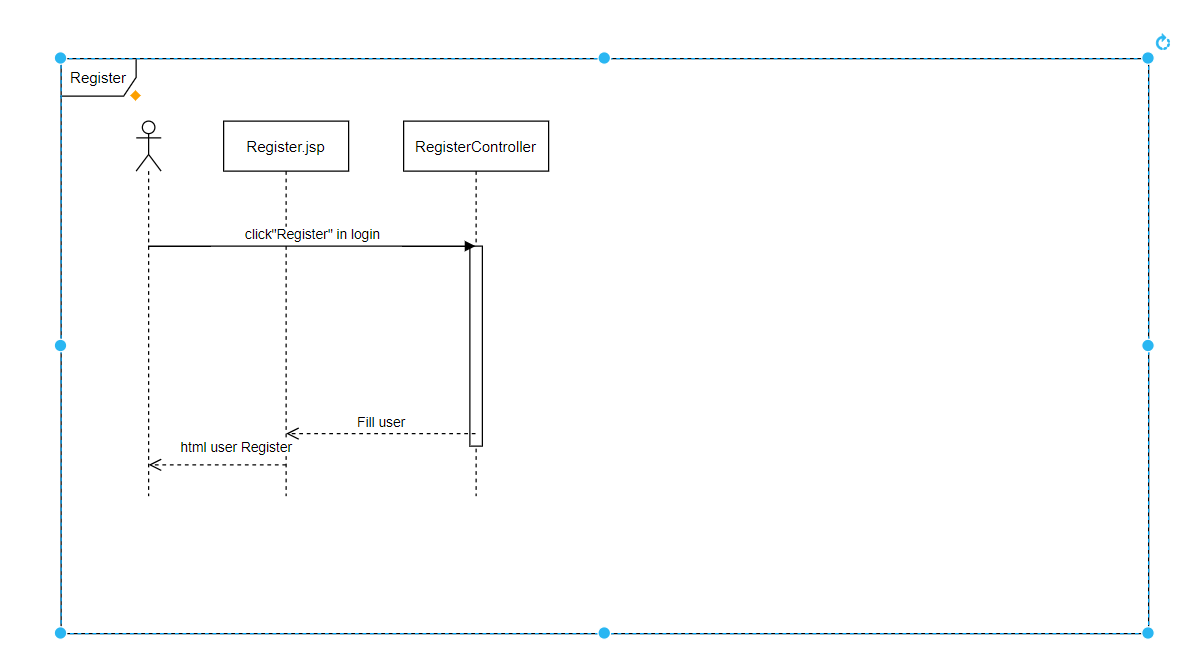
#### RegisterController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)



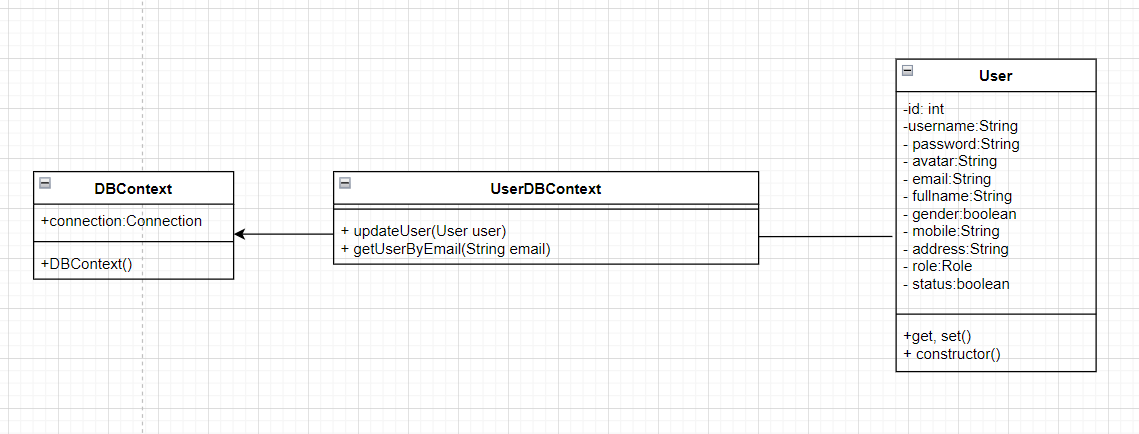
### d. Database queries

* Insert into [User]([password], email, fullname, gender, mobile, address, [Status], roleId, avatar)"

values (?,?,?,?,?,?,?,?,?)

## 10. Reset Password

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserByEmail(String email)* | *Get User by email passed to function.*  *Return : only one user.* |
| *02* | *updateUser(User user)* | *Update user information in the system by the email transmitted.* |

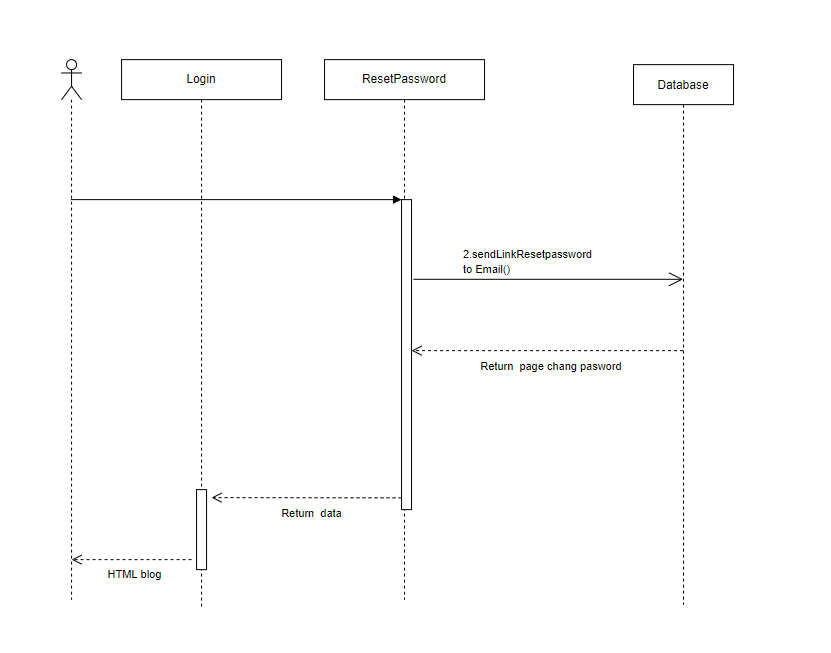
#### ChangePasswordController

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)

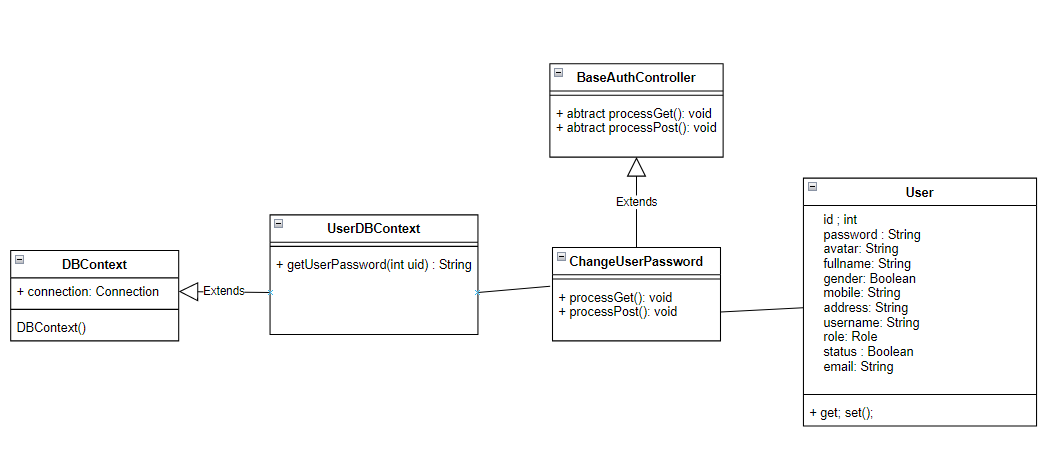


### d. Database queries

* Select u.\*, r.name rname from [User] u, [role] r where email = ?
* Update [User] set password=? where email = ?

## 11. Change User Password

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserPassword(int uid)* | *Get user password in string format*  *Return : Password of current user id in the session* |

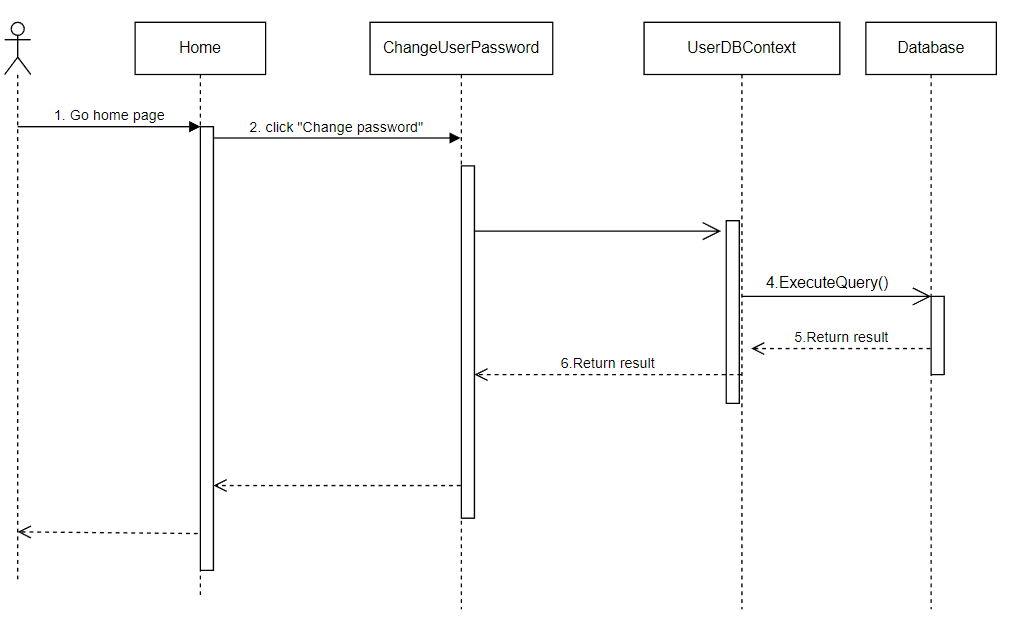
#### ChangeUserPassword

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)

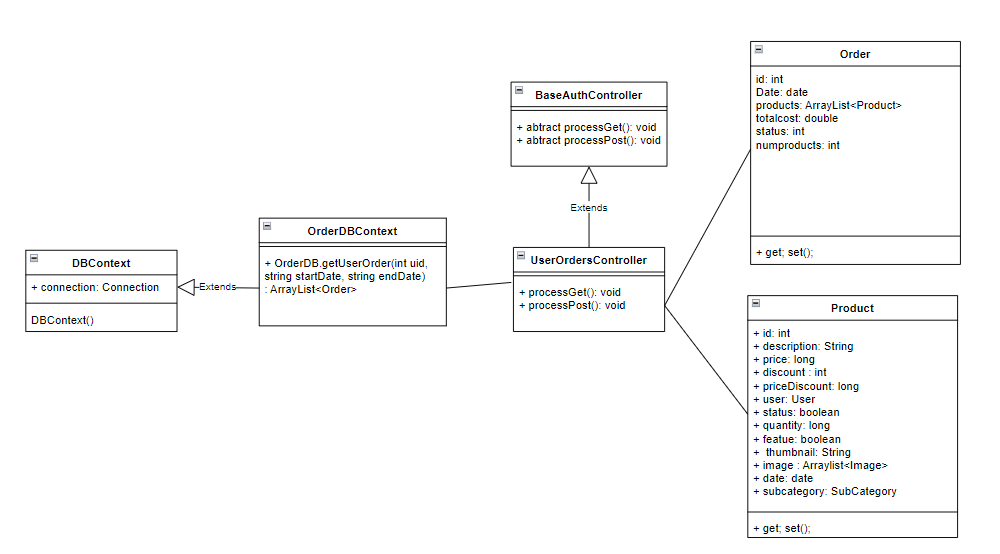


### d. Database queries

* SELECT [password] FROM [User] WHERE id = ?
* UPDATE [dbo].[User] SET [password] = ? WHERE id = ?

## 12. My orders

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserOrder(int uid, string startDate, string endDate)* | *Get all user orders in date range*  *Return : an ArrayList of orders* |

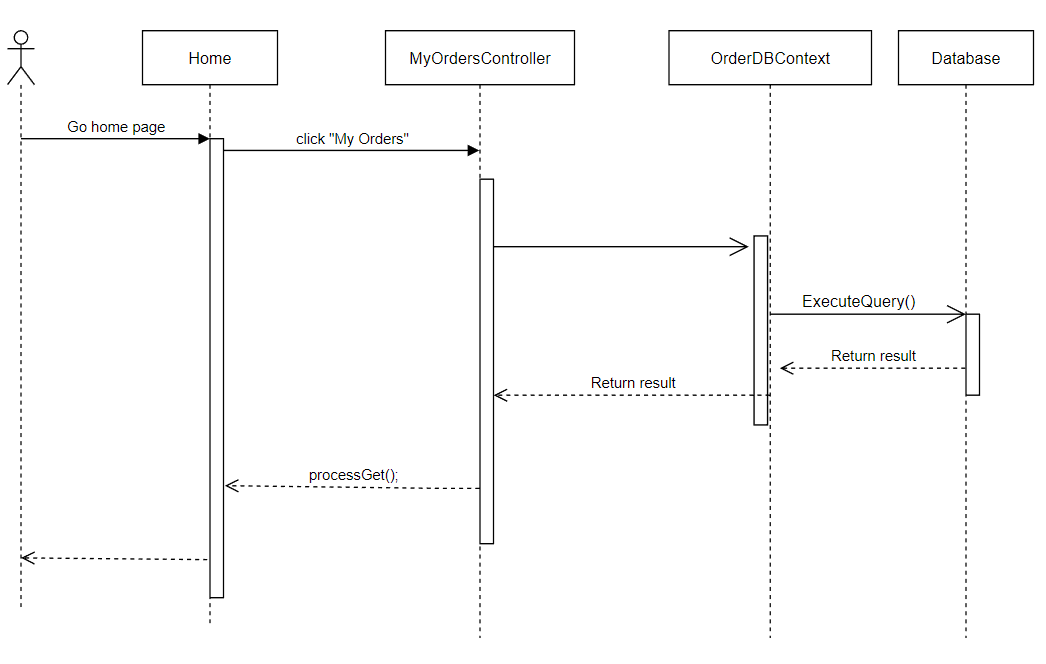
#### ChangeUserPassword

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *03* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### 

### c. Sequence Diagram(s)



### d. Database queries

* WITH \n t as

(SELECT [Order].id as OrderID, [Order].[date], [Order].totalPrice, [Product].[name] as ProductName, [Order].[status] as OrderStatus

FROM [Order] inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id

WHERE [Order].userId = ?)

b as

(SELECT [Order].id as OrderID, COUNT(Product.id) as NumberOfProducts

FROM [Order] inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id

WHERE [Order].userId = ? group by [Order].id),

c as

(SELECT a.\*

FROM ( SELECT DISTINCT t.OrderID FROM t) mo

CROSS APPLY (SELECT TOP 1 \* FROM t mi WHERE mi.OrderID = mo.OrderID) a

Select c.OrderID, c.[date], c.totalPrice, c.ProductName, c.OrderStatus, b.NumberOfProducts from c inner join b on c.OrderID = b.OrderID

WHERE c.[date] between ? and ?

## 13. User Authorization

### a. Class Diagram

### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### AddRoleController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *ArrayList<Feature> getFeatureByGroup(String groupName)* | *-Get all Feature*  *Return: all feature* |

#### EditRoleController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *addCustomer(User cusomter)* | *-Add customer to database*  *- Input : User information* |

#### c. Sequence Diagram(s)

* Customer List

- Add new user

- Edit new user

### d. Database queries

* SELECT [fullname] ,[password],[gender],[email],[mobile],[address] ,[roleId],Role.name,[User].[status]\,[User].id FROM [User] join Role on roleId = Role.id

- UPDATE [dbo].[User]

SET [status] = “”

WHERE id = “”

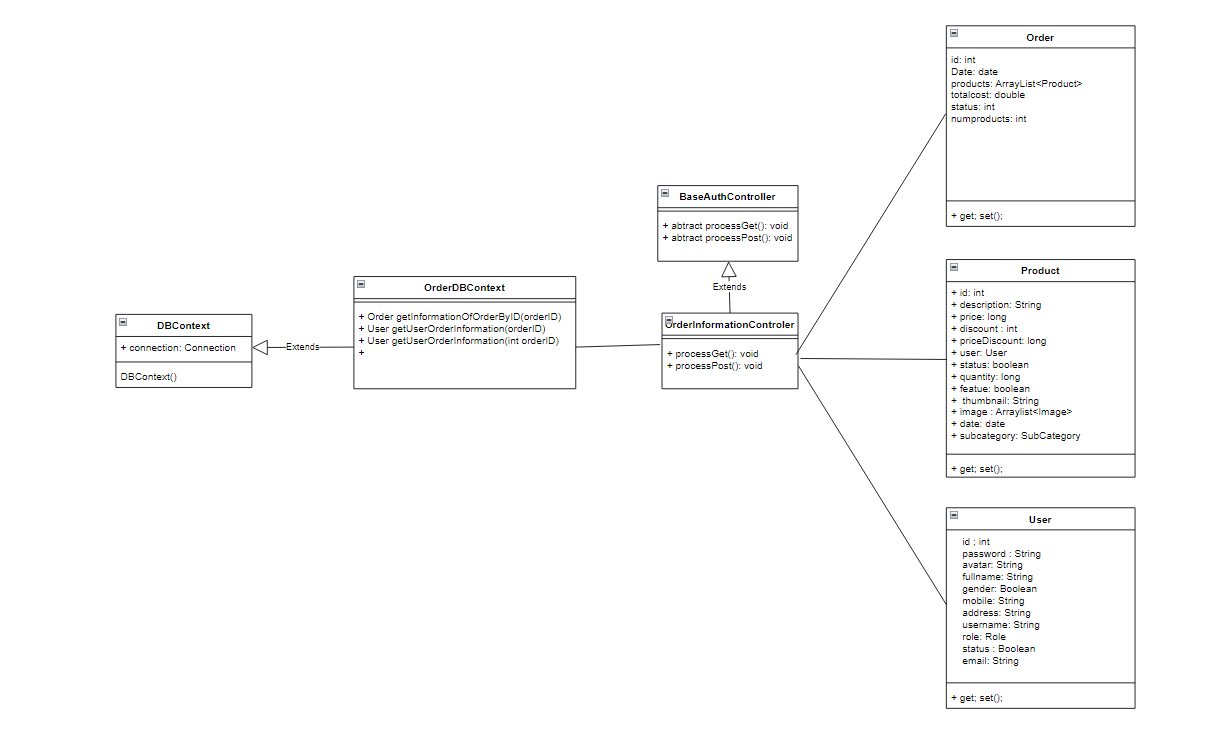
- INSERT INTO [dbo].[User]

([password] ,[avatar] ,[fullname] ,[gender] ,[mobile] ,[address] ,[roleId] ,[status] ,[email]), VALUES ()

* SELECT [fullname] ,[password] ,[gender] ,[email] ,[mobile] ,[address] ,[roleId] ,Role.name ,[User].[status] ,[User].id FROM [User] join Role on roleId = Role.id WHERE [User].id = ?

## 15. Order Information

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getUserOrderInformation(int orderID) : User* | *-Get information receive*  *-Input : orderID*  *-Return : User Information* |
| *02* | *+getInformationOfOrderByID(int orderID) : Order* | *-Get Information of this Order*  *- input : orderID*  *-Return : Order information* |
| *03* | *+ getListOrderProductOfUser(int orderID) : ArrayList<Product>* | *-Get list ordered of user*  *- input : orderID*  *-Return :list order product of user* |

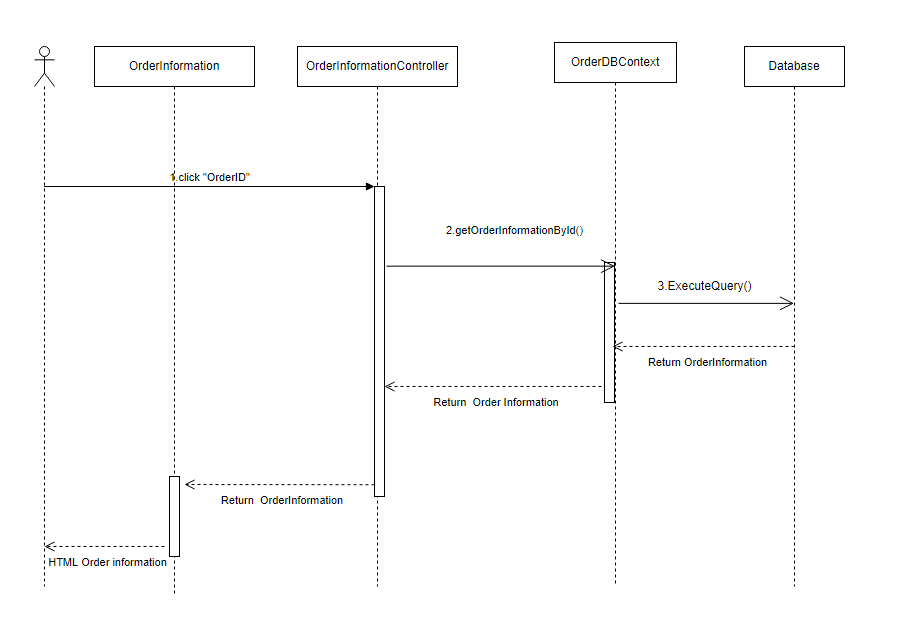
#### 

### 

### 

### 

### c. Sequence Diagram(s)



### d. Database queries

* select DISTINCT od.id, od.date, od.status, od.totalPrice

from

[Order] od WHERE od.id = ?

* select DISTINCT s.fullname, u.gender, s.email, s.phone

from

[Order] od

INNER JOIN ShipInfo s ON od.idShip = s.id

INNER JOIN [User] u on od.userId = u.id

WHERE od.id = ?

* select p.id as productID, p.thumbnail, p.name as pname, c.name as categoryName, o.quantity, o.discount, o.price, od.userId

from

[Order] od

INNER JOIN OrderDetail o ON o.orderId = od.id

INNER JOIN Product p ON o.productId = p.id

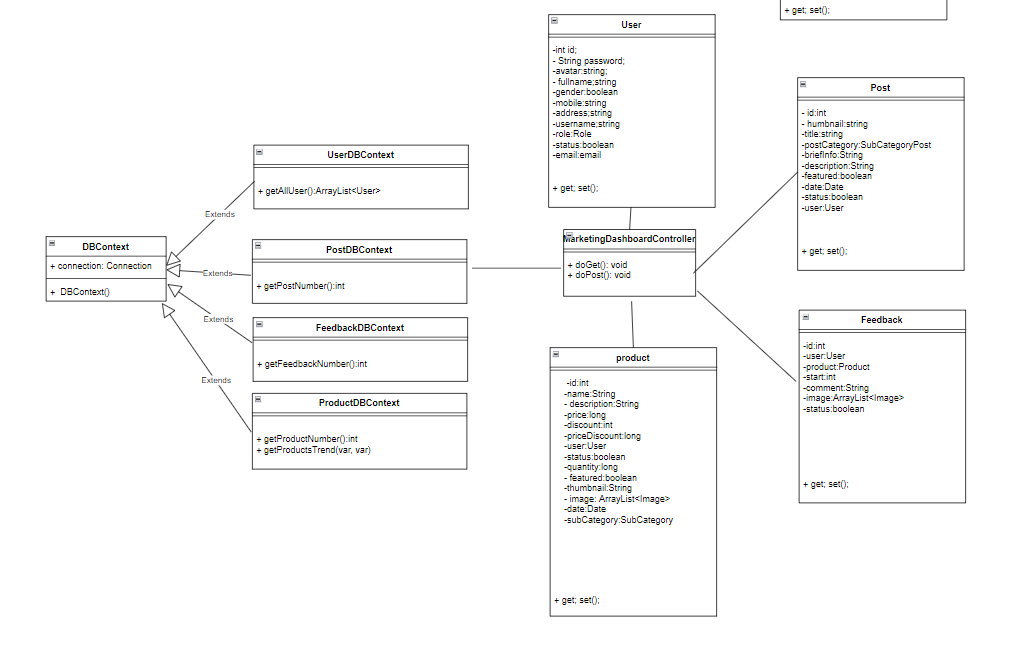
INNER JOIN SubCategory sub ON p.subCategoryId = sub.id

INNER JOIN Category c ON sub.categoryId = c.id

WHERE o.orderId = ?

## 16. MKT Dashboard

### a. Class Diagram



### b. Class Specifications

#### DBContext

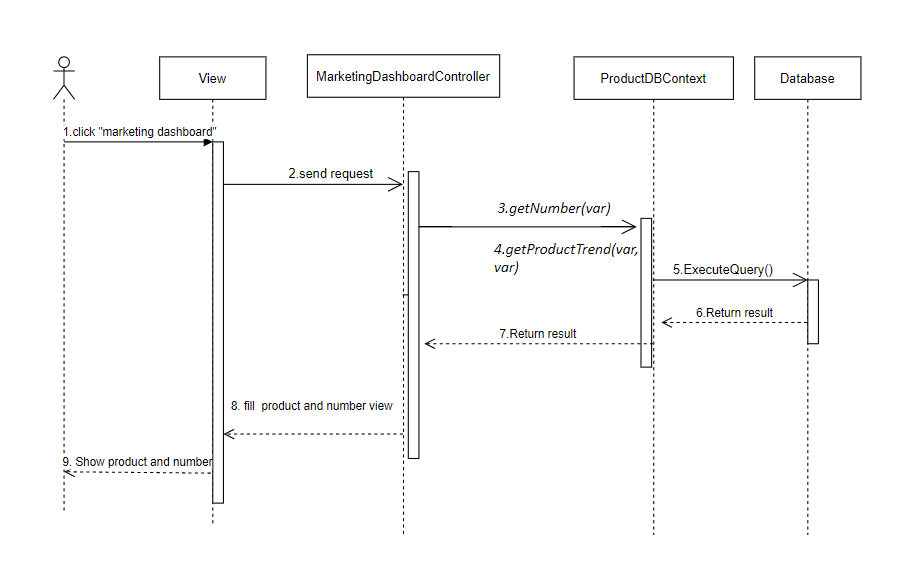
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### ProductDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+ getAllUser():ArrayList<User>* | *- Get list user*  *-Return : :List user* |
| *02* | *+ getPostNumber():int* | *-get post number*  *-Return : post number* |
| *03* | *+ getFeedbackNumber():int* | *-get feedback number*  *-Return : feedback number* |
| *04* | *+ getProductNumber():int* | *-get product number*  *-Return : product number* |
| *05* | *+ getProductsTrend(var, var)* | *-get product by trend*  *input: info date*  *return:Product rating and quantity* |

#### 

### b. Sequence diagram



### d. Database queries

select p.\*, p1.number from product as p,

(select top (10) od.productId, sum(od.quantity) as number from OrderDetail as [od] , [Order] as [o] where od.orderId = o.id"

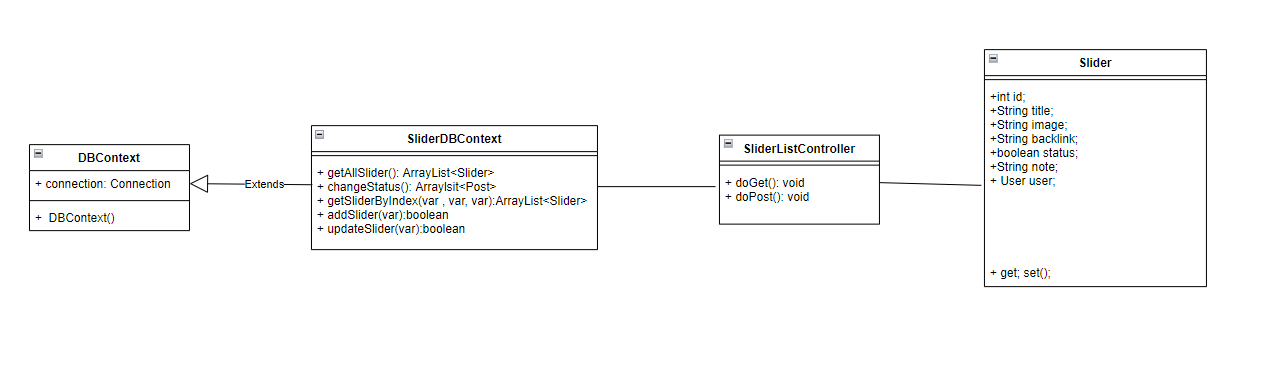
and o.userid in (select id from [User] where MONTH(dateCreated) = MONTH(GETDATE()) and YEAR(dateCreated) = YEAR(GETDATE()))";

group by od.productId

order by number desc) as p1 where p.id = p1.productId

## 20. Sliders List

### a. Class Diagram



### b. Class Specifications

#### DBContext

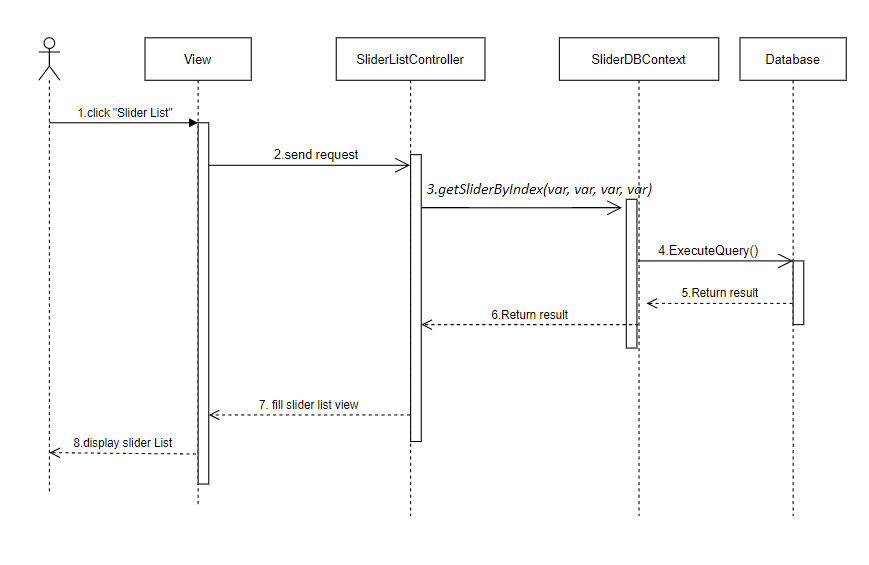
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### SliderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getAllSlider()* | *- Get list slider*  *-Return : List slider* |
| *02* | *+changeStatus(int id, boolean status)* | *-Change status slider*  *-Input: product id, status;* |
| *03* | *+getSliderByIndex(int index,int sizePage, int status, String search)* | *- Get list slider*  *-Input: index, sizePage, status, search;*  *-Return : satisfying slider list.* |
| *04* | *+getSliderById(int id)* | *-Get a slider*  *-Input: id of product* |
| *05* | *+addSlider(Slider s)* | *- Add a slider*  *-Input: slider;*  *-Return :true or false;.* |
| *06* | *+updateSlider(Slider s)* | *- Update a slider*  *-Input: slider;*  *-Return :true or false;.* |

#### 

### b. Sequence diagram



### d. Database queries

-UPDATE [Slider] set status = ? where id = ?

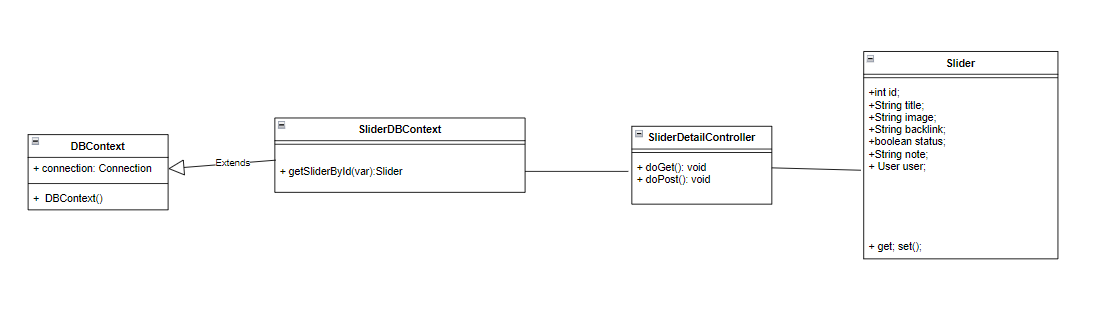
-select s.\*, u.[password], u.avatar, u.fullname, u.status [statusUser], u.mobile, u.address, u.roleId, u.email, u.gender from Slider s, [user] u where s.userid = u.id and 1=1 AND s.status = ?

order by Id offset ? rows fetch next ? rows only

-select s.\*, u.[password], u.avatar, u.fullname, u.status [statusUser], u.mobile, u.address, u.roleId, u.email, u.gender from Slider s, [user] u where s.userid = u.id and s.id = ?

## 21. Slider Details

### a. Class Diagram



### b. Class Specifications

#### DBContext

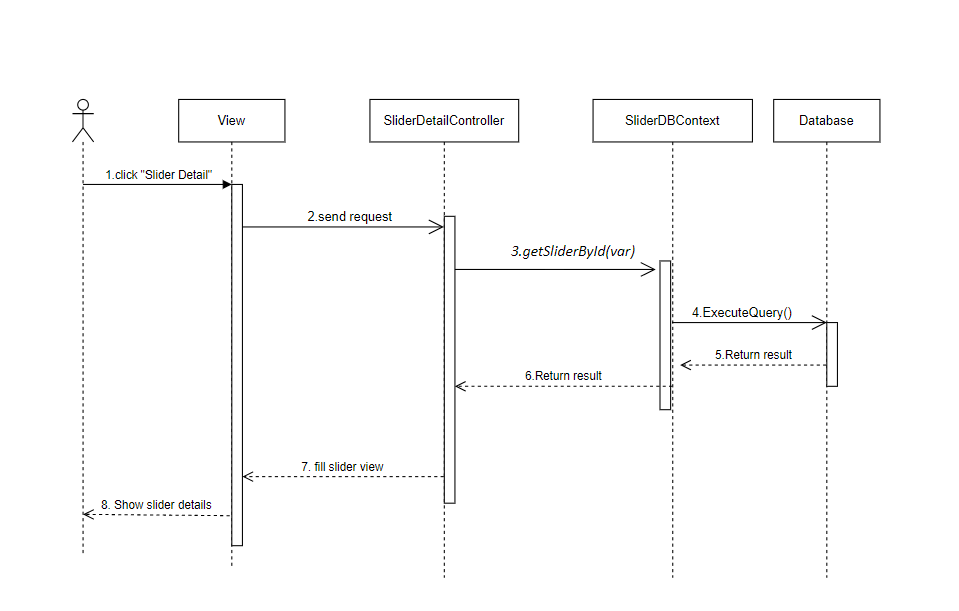
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### SliderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getSliderById(int id)* | *-Get a slider*  *-Input: id of product* |

#### 

### b. Sequence diagram



### d. Database queries

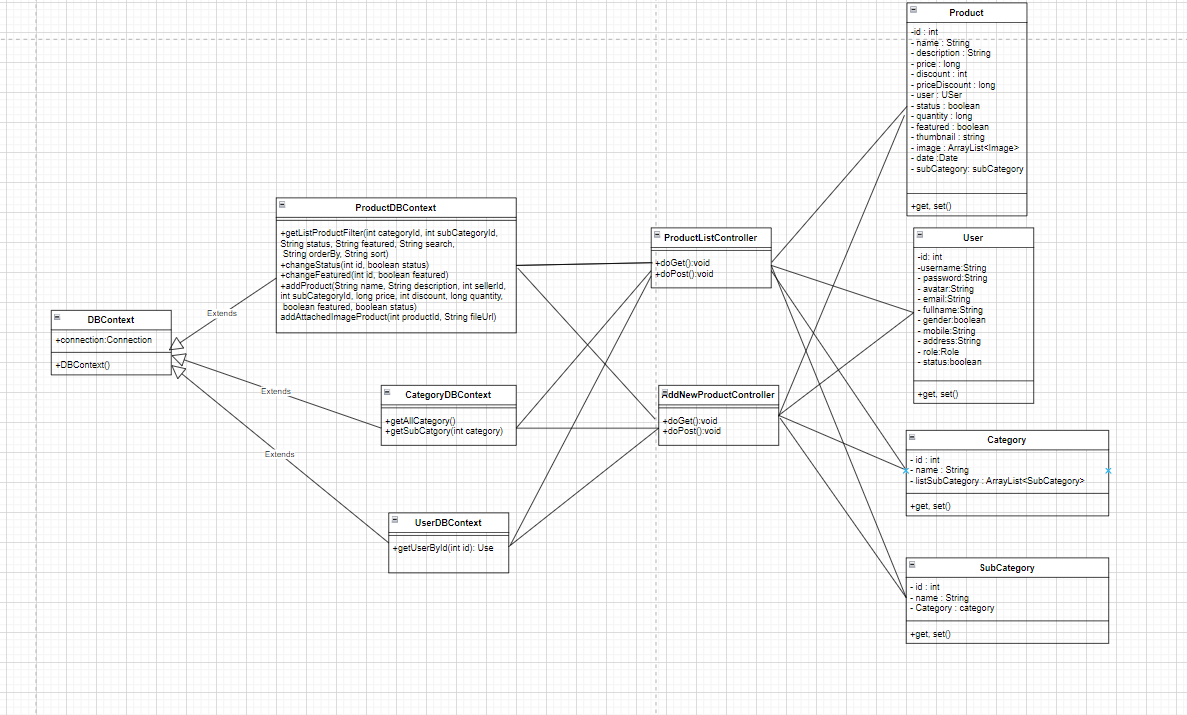
-Select s.\*, u.[password], u.avatar, u.fullname, u.status

[statusUser], u.mobile, u.address, u.roleId, u.email, u.gender

from Slider s, [user] u where s.userid = u.id and s.id = ?

## 22. Marketing/Products List

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### ProductDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getListProductFilter(int categoryId, int subCategoryId,*  *String status, String featured, String search,*  *String orderBy, String sort)* | *- Get list product filter with condition*  *-Input : categoryId, subCategoryId, status,featured, search, orderBy,sort*  *-Return : :List product satisfied* |
| *02* | *changeStatus(int id, boolean status)* | *-Change status product*  *-Input: product id, status;* |
| *03* | *+changeFeatured(int id, boolean featured)* | *-Change featured product*  *-Input: product id, status;* |
| *04* | *+addProduct(String name, String description, int sellerId,*  *int subCategoryId, long price, int discount, long quantity,*  *boolean featured, boolean status)* | *-Add new product*  *-Input: information of product* |

#### CategoryDBContextContext

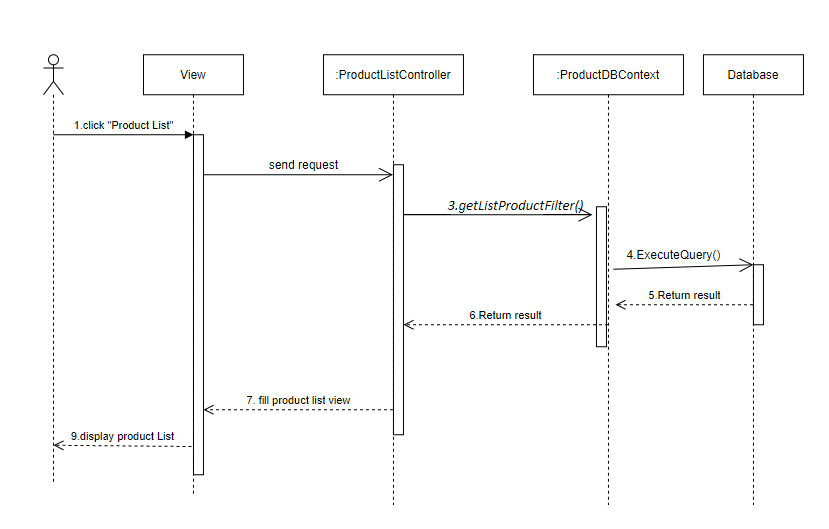
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getAllCategory()* | *-Get all category existed*  *-Input :*  *-Result : list all product’s category* |
| *02* | *+getSubCatgory(int category)* | *-Get subcategory*  *- input : Category Id*  *-Return: list subCategory* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserById(int id)* | *- Get a user*  *Input:User Id*  *Return: A User where user id contain parameter* |

#### 

### b. Sequence diagram



### d. Database queries

-DECLARE @Col\_Name VARCHAR(128) = ?

SELECT [Product].[id] ,[Product].[name],[description],[price] ,[discount]\ ,[sellerId]

,[featured],[thumbnail],[date],[subCategoryId] ,[SubCategory].[name] as [subCatogryName] ,[Category].id as categoryId ,[Category].[name] as [catogryName]

,[quantity] ,[status] ,[price] \* [discount] as [salePrice]

FROM [dbo].[Product] join SubCategory on subCategoryId = SubCategory.id

join Category on SubCategory.categoryId = Category.id

WHERE (1=1)

and categoryId = ?

and subCategoryId = ?

and [status] = ?

and [featured] = ?

and ([Product].[name] like ? or [Product].[id] = ?

ORDER BY CASE

WHEN @Col\_Name = 'id' THEN CAST([Product].[id] AS SQL\_VARIANT)

WHEN @Col\_Name = 'category' THEN CAST([Category].[name] AS SQL\_VARIANT)

WHEN @Col\_Name = 'subCategory' THEN CAST([SubCategory].[name] AS SQL\_VARIANT)

WHEN @Col\_Name = 'originalPrice' THEN CAST([price] AS SQL\_VARIANT)

WHEN @Col\_Name = 'featured' THEN CAST([featured] AS SQL\_VARIANT)

WHEN @Col\_Name = 'status' THEN CAST([status] AS SQL\_VARIANT)

END ?

-UPDATE [dbo].[Product]

SET [status] = ?

WHERE id = ?

-UPDATE [dbo].[Product]

SET [featured] = ?

WHERE id = ?

-INSERT INTO [dbo].[Product]

([name] ,[description] ,[price] ,[discount] ,[sellerId] ,[featured] ,[date],[subCategoryId]

,[quantity] ,[status])

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

-SELECT [id],[name]

FROM [dbo].[Category]

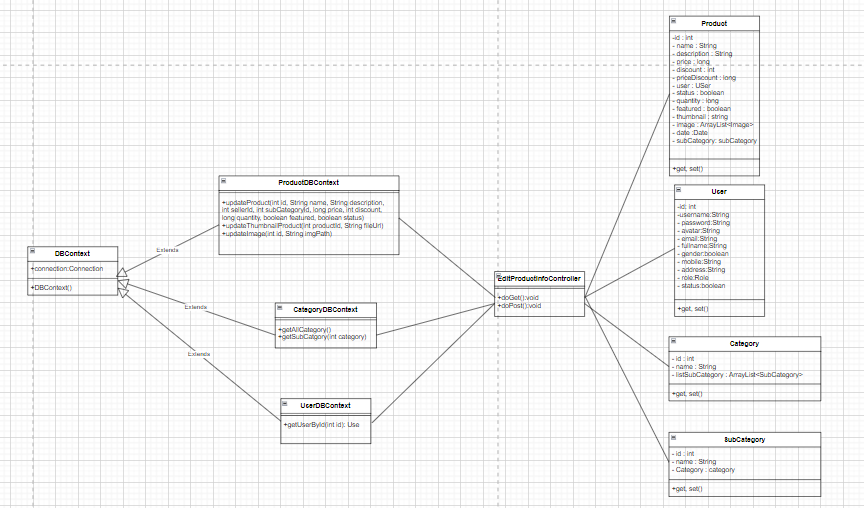
-SELECT [SubCategory].[id],[SubCategory].[name],[categoryId],[Category].[name]

FROM [dbo].[SubCategory] join Category on Category.id = SubCategory.categoryId

WHERE Category.id = ?

## 23. Marketing/Product Details

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### ProductDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateProduct(int id, String name, String description,*  *int sellerId, int subCategoryId, long price, int discount,*  *long quantity, boolean featured, boolean status)* | *-Update a product’s info*  *-Input: product’s id existed , new information of product* |
| *02* | *updateThumbnailProduct(int productId, String fileUrl)* | *-Update thumbnail product*  *-Input: product id, new path file* |
| *03* | *+updateImage(int id, String imgPath)* | *-Update attached image product*  *-Input: image id, new path file* |

#### CategoryDBContextContext

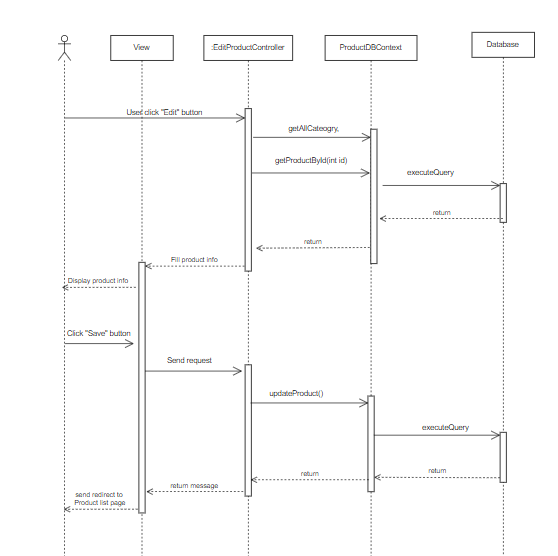
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getAllCategory()* | *-Get all category existed*  *-Input :*  *-Result : list all product’s category* |
| *02* | *+getSubCatgory(int category)* | *-Get subcategory*  *- input : Category Id*  *-Return: list subCategory* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserById(int id)* | *- Get a user*  *Input:User Id*  *Return: A User where user id contain parameter* |

#### 

### b. Sequence diagram



### d. Database queries

-

-SELECT [id],[name]

FROM [dbo].[Category]

-SELECT [SubCategory].[id],[SubCategory].[name],[categoryId],[Category].[name]

FROM [dbo].[SubCategory] join Category on Category.id = SubCategory.categoryId

WHERE Category.id = ?

-UPDATE [dbo].[Product]

SET [name] = ? ,[description] = ? ,[price] = ?,[discount] = ?,[sellerId] = ?,[featured] = ?,[date] = ?,[subCategoryId] = ?,[quantity] = ?,[status] = ?

WHERE id = ?

-UPDATE [dbo].[Product]

SET [thumbnail] = ?

WHERE id = ?

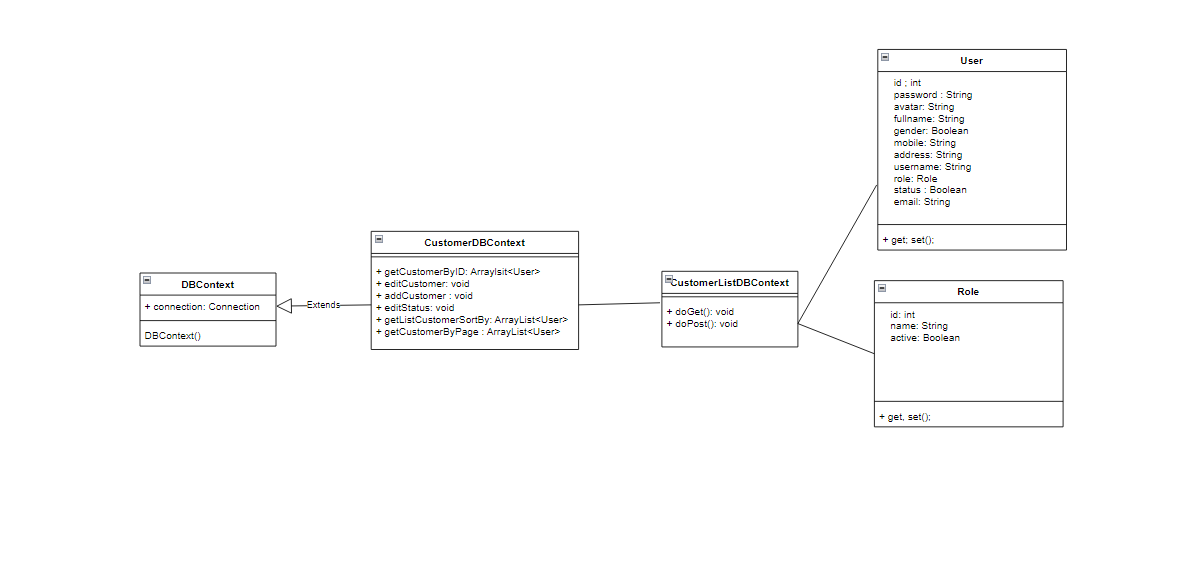
-UPDATE [dbo].[Image]

SET [image] = ?

WHERE id = ?

## 24. Customers List

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### CustomerListDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCusomerByPage(int userRole, String searchBy, String statusBy, int pageindex, int pagesize)* | *-Get all customer information in database*  *Return: All Customer in database* |

#### AddCustomerDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *addCustomer(User cusomter)* | *-Add customer to database* |

#### EditCustomerDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *editStatus(int id, boolean status)* | *-Edit information of customer* |

#### c. Sequence Diagram(s)

* Customer List

- Add new user

- Edit new user

### d. Database queries

* SELECT [fullname] ,[password],[gender],[email],[mobile],[address] ,[roleId],Role.name,[User].[status]\,[User].id FROM [User] join Role on roleId = Role.id

- UPDATE [dbo].[User]

SET [status] = ?

WHERE id = ?

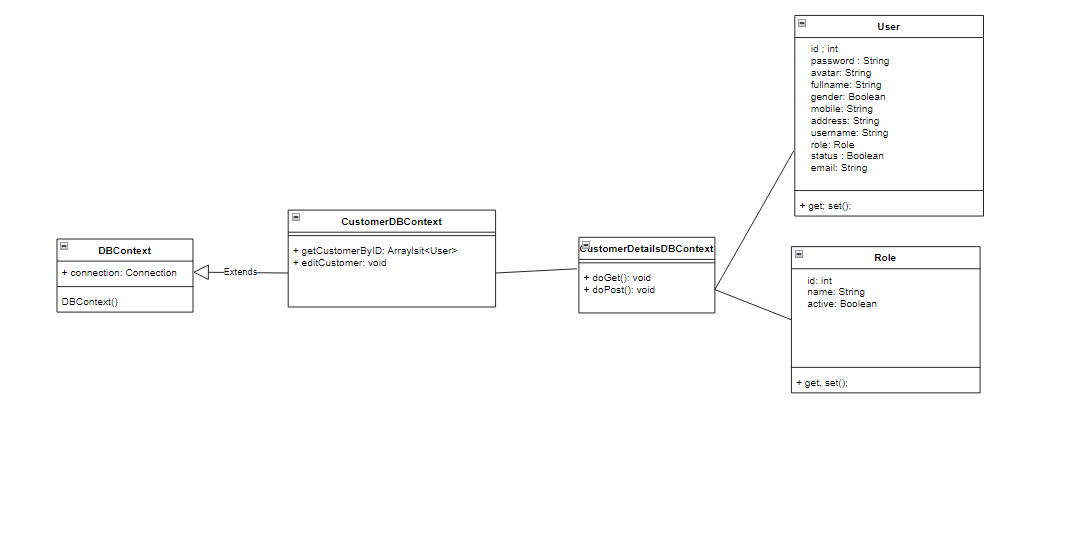
- INSERT INTO [dbo].[User]

([password] ,[avatar] ,[fullname] ,[gender] ,[mobile] ,[address] ,[roleId] ,[status] , [email]), VALUES ()

* SELECT [fullname] ,[password] ,[gender] ,[email] ,[mobile] ,[address] ,[roleId] ,Role.name ,[User].[status] ,[User].id FROM [User] join Role on roleId = Role.id WHERE [User].id = ?

## 25. Customers Details

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### CustomerDetailsDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getCustomerByID(int customerID)* | *-get information of customer by id*  *return: customer by that id* |
| *02* | *editCustomer(User customer)* | *-edit customer information* |

#### 

#### c. Sequence Diagram(s)

* Customer List

- Add new user

- Edit new user

### d. Database queries

- SELECT [fullname] ,[password] ,[gender] ,[email] ,[mobile] ,[address] ,[roleId] ,Role.name ,[User].[status] ,[User].id FROM [User] join Role on roleId = Role.id WHERE [User].id = ?

- UPDATE [dbo].[User] SET [fullname] = ?

,[gender] = ?

,[mobile] = ?

,[address] = ?

,[email] = ?

WHERE id = ?

26. Feedbacks List

27. Feedback Details

28. Sale Dashboard

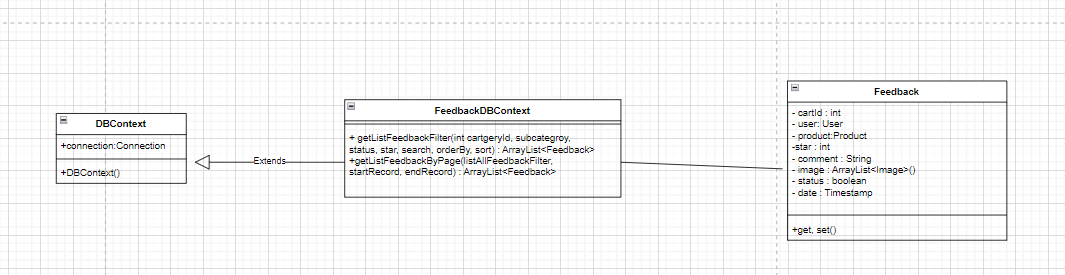
29. Orders List

30. Order Details

31. Admin Dashboard

## 32. Marketing/Feedback List

### a. Class Diagram



### b. Class Specifications

#### DBContext

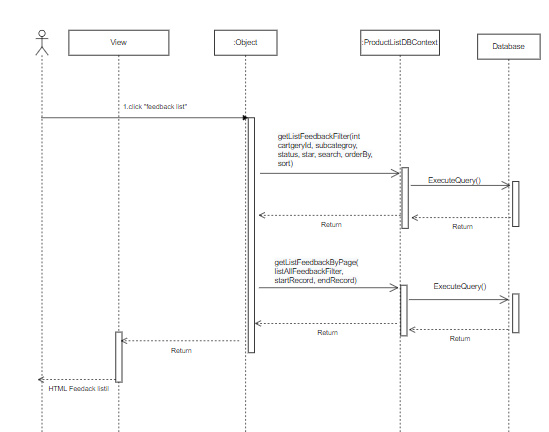
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### CartDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getListFeedbackFilter(int cartgeryId, subcategroy,*  *status, star, search, orderBy, sort) : ArrayList<Feedback>* | *-Get all Feedback satisfy condition*  *-Input : int cartgeryId, subcategroy, status, star, search, orderBy, sort*  *-Return : List feedback* |
| *02* | *+getListFeedbackByPage(listAllFeedbackFilter,*  *startRecord, endRecord) : ArrayList<Feedback>* | *-Get all product of cart*  *- input :listAllFeedbackFilter, startRecord, endRecord*  *-Return : List feedback* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

- getListFeedbackFilter(int cartgeryId, subcategroy, status, star, search, orderBy, sort) :

DECLARE @Col\_Name VARCHAR(128) = " '" + orderBy '" ;

sql += "SELECT [Feedback].id

,[User].[fullname]

,[Product].[name]

,[start]

,[comment]

,[image]

,[Feedback].[status]

FROM [dbo].[Feedback] join [User] on [User].id = Feedback.userId

join Product on Product.id = Feedback.productId

join SubCategory on Product.subCategoryId = SubCategory.id

join Category on Category.id =SubCategory.categoryId

WHERE (1=1)

" and categoryId = ?\n

" and subCategoryId = ?

" and [Feedback].[status] = ?

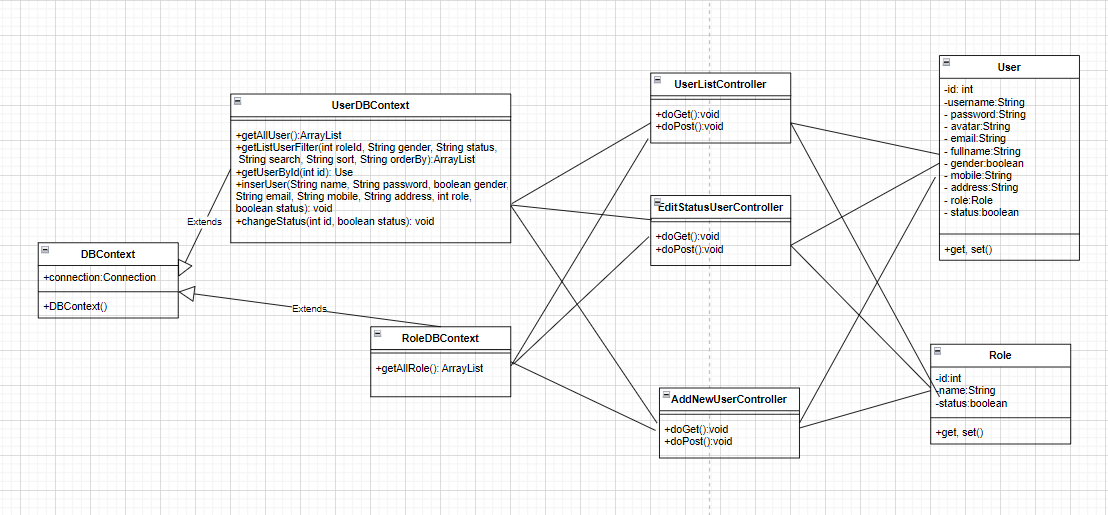
" and [start] = ?

"and ([User].[fullname] like ? or [comment] like ? or [Product].name like ?)

## 33. Admin/User List

### 

#### 



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllUser()* | *-Get all user information in database*  *Return: All User in database* |
| *02* | *getListUserFilter(int roleId, String gender, String status,*  *String search, String sort, String orderBy)* | *-Get list user with condition in database*  *Input: roleId, gender, status, search(part of name, email, …), sort(by id, name, gender…), orderBy(asc, desc) Return: All User satisfy with condition* |
| *03* | *getUserById(int id)* | *- Get a user*  *Input:User Id*  *Return: A User where user id contain parameter* |
| *04* | *changeStatus(int Id, boolean status)* | *-Change status of a user*  *Input: User Id, status* |
| *05* | *inserUser(String name, String password, boolean gender,*  *String email, String mobile, String address, int role,*  *boolean status)* | *-Add a user in to database*  *Input: name, password, gender, email, mobile, address,role, boolean status* |
| *06* | *checkAccountHaveEmailOrMobileExisted(String email, String phone)* | *-Check email and phone excited*  *-Input: email, phone number*  *-Return: true or false* |

#### RoleDBContext

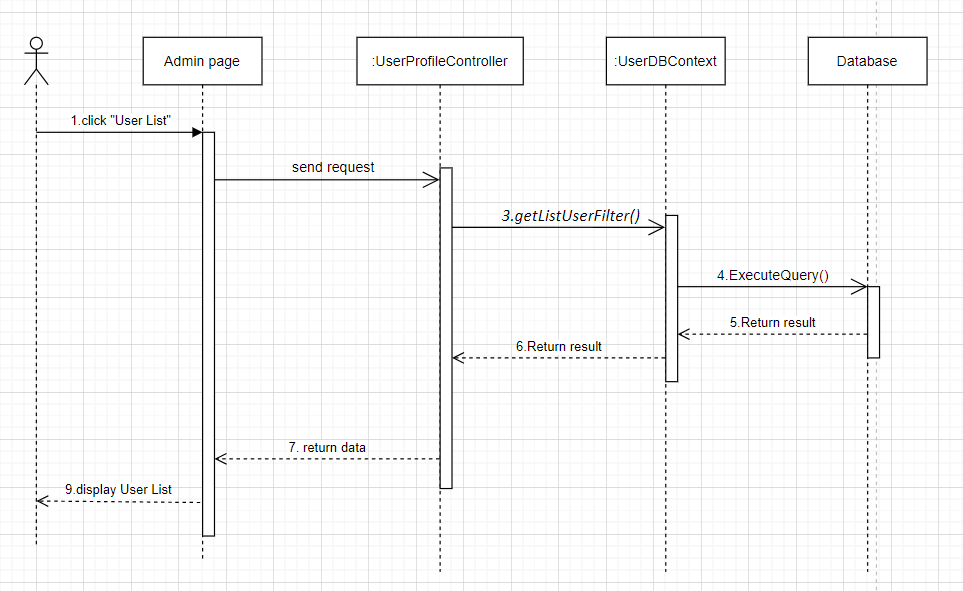
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllRole()* | *-Get all role’s information in database*  *Return: All role in database* |

#### UserListController

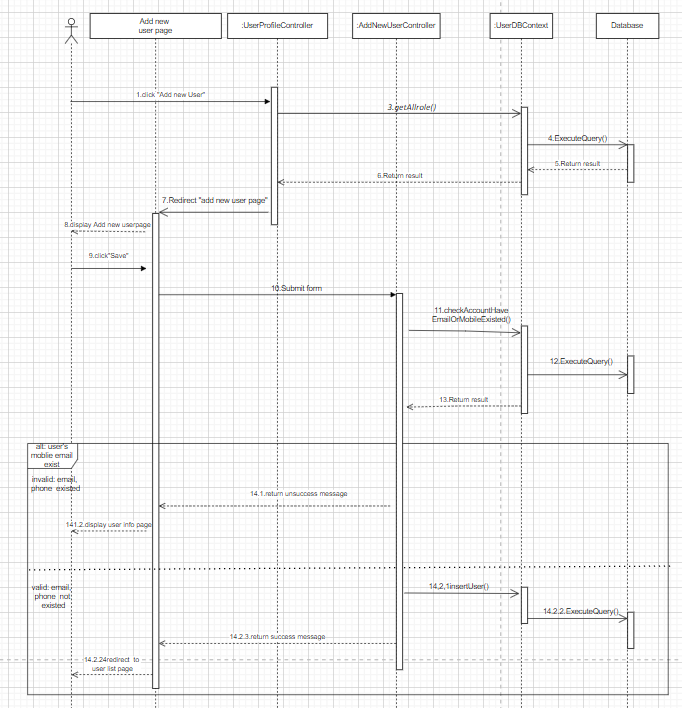
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### c. Sequence Diagram(s)

* User List



- Add new user



### d. Database queries

* SELECT [fullname] ,[password],[gender],[email],[mobile],[address] ,[roleId],Role.name,[User].[status]\,[User].id FROM [User] join Role on roleId = Role.id

- DECLARE @Col\_Name VARCHAR(128) = " '" + ?'" \n

SELECT [fullname] ,[password],[gender],[email],[mobile] ,[address],[roleId],Role.name,[User].[status] ,[User].id

FROM [User] join [Role] on roleId = [Role].id

WHERE (1=1)

and roleId = ? and gender = ? and [User].status = ? and (fullname like or email like ?or mobile like ?)

ORDER BY CASE

WHEN @Col\_Name = 'id' THEN CAST([User].id AS SQL\_VARIANT)

WHEN @Col\_Name = 'fullname' THEN CAST(fullname AS SQL\_VARIANT)

WHEN @Col\_Name = 'gender' THEN CAST(gender AS SQL\_VARIANT)

WHEN @Col\_Name = 'email' THEN CAST(email AS SQL\_VARIANT)

WHEN @Col\_Name = 'mobile' THEN CAST(mobile AS SQL\_VARIANT)

WHEN @Col\_Name = 'role' THEN CAST(roleId AS SQL\_VARIANT)

WHEN @Col\_Name = 'status' THEN CAST([User].status AS SQL\_VARIANT)

END ?

- SELECT [fullname],[gender],[avatar],[email],[mobile],[address],[roleId],[Role].[name]

,[User].[status] FROM [dbo].[User] join [Role] on [User].roleId = [Role].id WHERE [User].id = ?

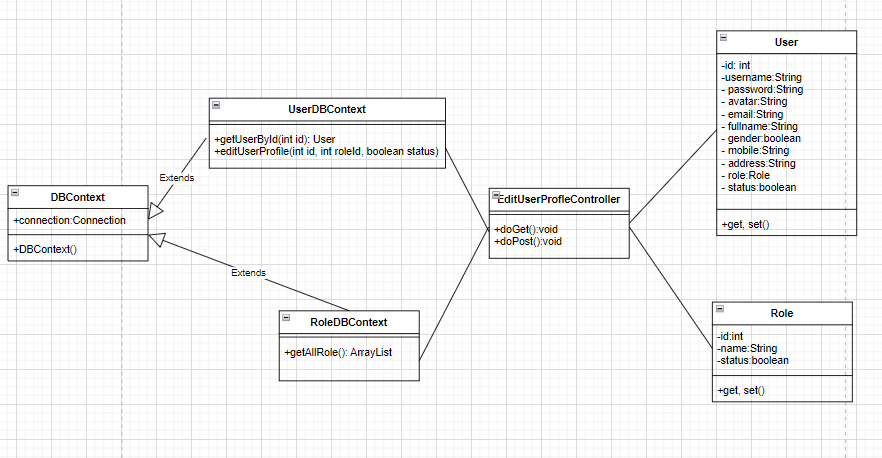
* INSERT INTO [dbo].[User] ([fullname],[password],[gender],[email],[mobile],[address],[roleId],

[status]) VALUES (N'?' ,?,?,?,?,N'?',?,?)

* UPDATE [dbo].[User] SET [status] = ? WHERE id = ?
* SELECT [id],[name] FROM [dbo].[Role]

## 34.Admin/ User detail

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserById(int id)* | *- Get a user by id*  *-Input:User Id*  *-Return: A User where user id contain parameter* |
| *02* | *editUserProfile(int id, int roleId, boolean status)* | *- Edit user’s role and user’s status by id*  *-Input: User id, roleId, status*  *Return: void* |

#### RoleDBContext

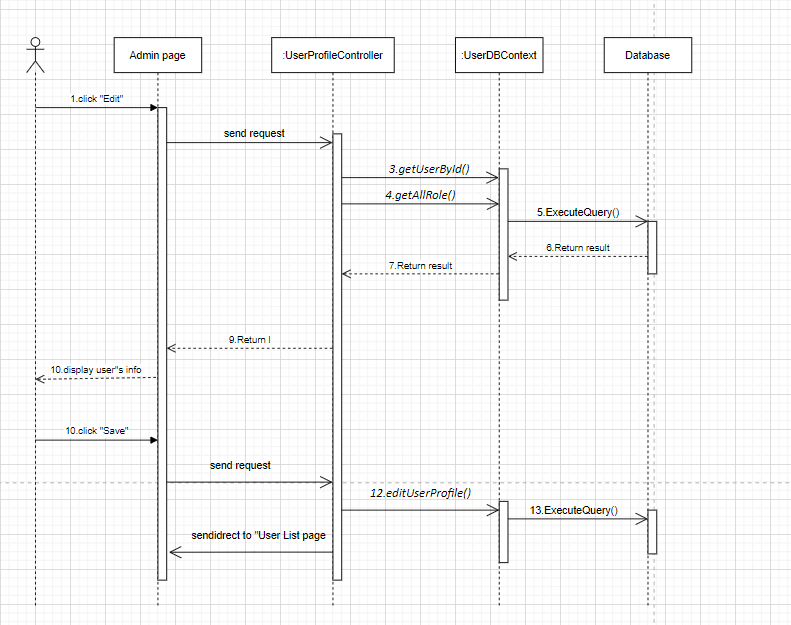
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllRole()* | *-Get all role’s information in database*  *Return: All role in database* |

#### EditUserProfile

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* SELECT [fullname],[gender],[avatar],[email],[mobile],[address],[roleId],[Role].[name]

,[User].[status] FROM [dbo].[User] join [Role] on [User].roleId = [Role].id WHERE [User].id = ?

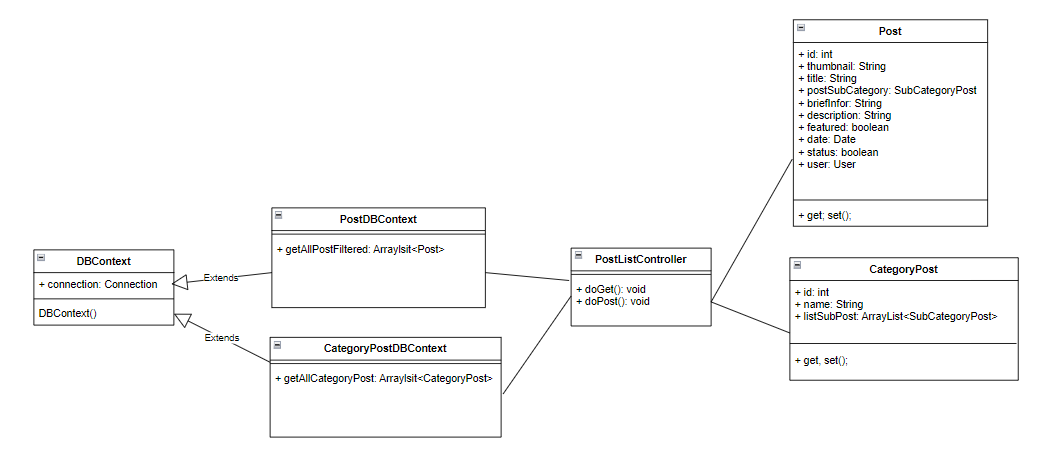
* UPDATE [User] SET [status] = ? ,roleId = ? WHERE id= ?
* SELECT [id],[name] FROM [dbo].[Role]

34. Settings List

35. Setting Details

## 35.Marketing/ Post List

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllPostFiltered(int id)* | *- Get list post filtered*  *-Input: idCategory, idStatus, searchBy, orderBy, sortBy*  *-Return: List Post* |

#### CategoryPostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategoryPost()* | *-Get all category Post*  *Return: List Category* |

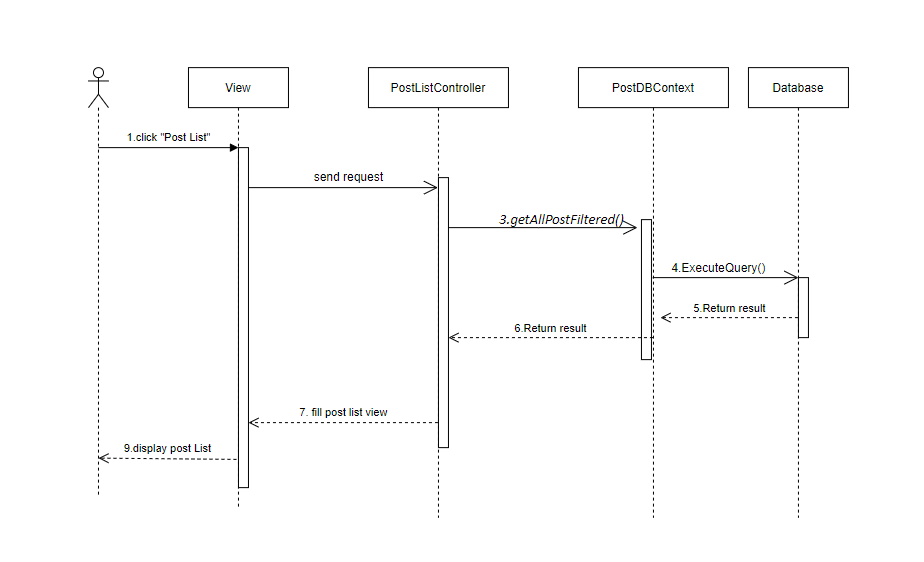
#### PostListController:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* CategoryPostDBContext
* Select \* from CategoryPost
* PostDBContext
* Select p.id, p.thumbnail, p.title, p.feature, p.[status], p.UserId, s.email, s.fullname, p.categoryId as idSubCategory, sp.[name] as nameSubCategory, cp.id as idCategoryPost, cp.[name] as nameCategoryPost

From post p join [user] s on p.UserId = s.id

join SubCategoryPost sp on p.categoryId = sp.id

join CategoryPost cp on cp.id = sp.idcategory and cp.id = ? and p.UserId = ? and p.[status] = ? and ( p.title like ? or s.fullname like ?)

ORDER BY CASE

WHEN @Col\_Name = 'title' THEN CAST(p.title AS SQL\_VARIANT)

WHEN @Col\_Name = 'category' THEN CAST(cp.id AS SQL\_VARIANT)

WHEN @Col\_Name = 'author' THEN CAST(p.UserId AS SQL\_VARIANT)

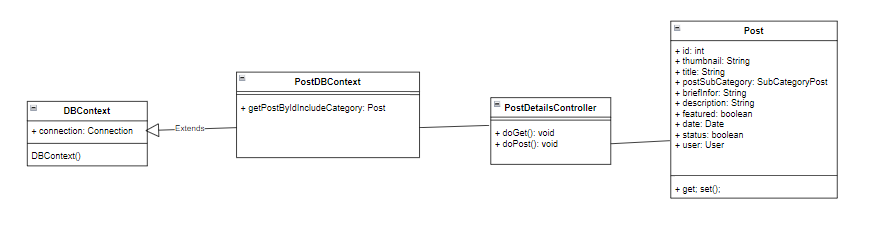
WHEN @Col\_Name = 'featured' THEN CAST(p.feature AS SQL\_VARIANT)

WHEN @Col\_Name = 'status' THEN CAST(p.status AS SQL\_VARIANT)

END

## 36.Marketing/ Post Details

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### 

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getPostByIdIncludeCategory(int id)* | *- Get post Details*  *-Input: idPost*  *-Return: A Post* |

#### 

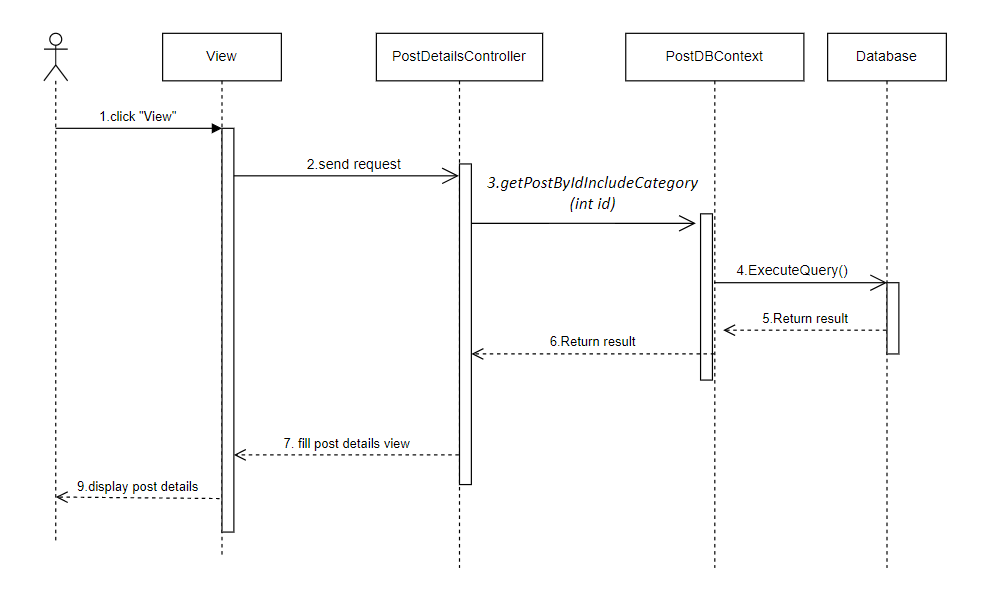
#### PostDetailsController:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* PostDBContext
* Select p.id, p.thumbnail, p.title,p.briefInfo, p.[description], p.feature,

p.[status], p.UserId, s.email, s.fullname, p.categoryId as idSubCategory,

sp.[name] as nameSubCategory, cp.id as idCategoryPost,

cp.[name] as nameCategoryPost, p.dateUpdated

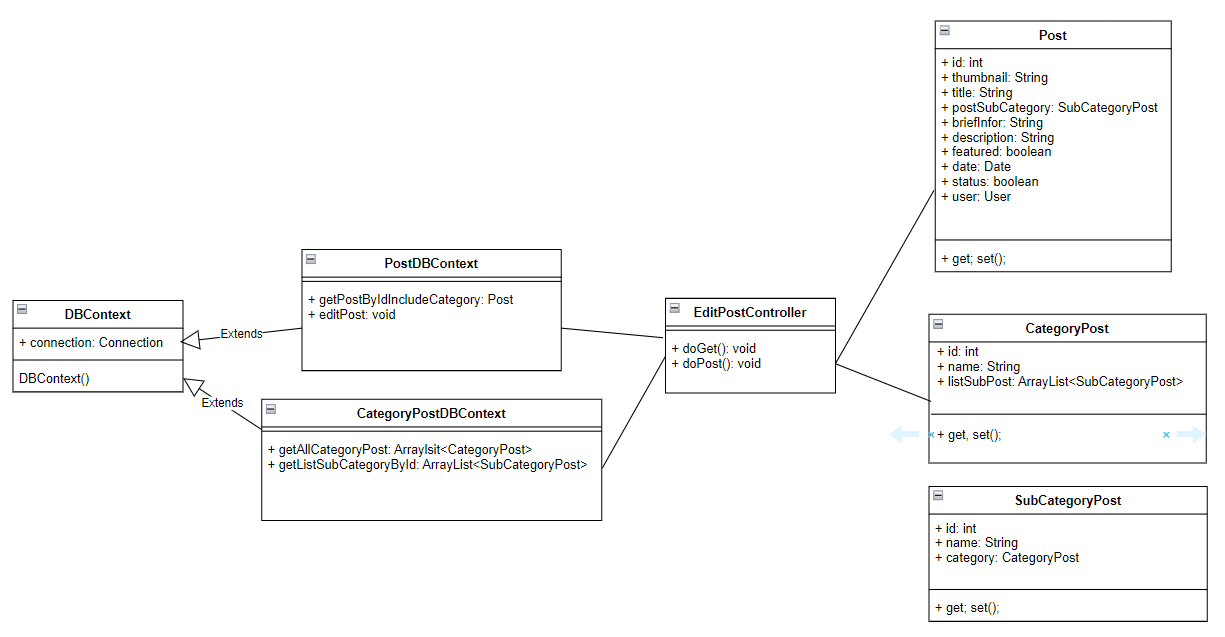
From post p join [user] s on p.UserId = s.id

join SubCategoryPost sp on p.categoryId = sp.id

join CategoryPost cp on cp.id = sp.idcategory and p.id = ?

## 37.Marketing/ Edit Post

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### CategoryPostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategoryPost()* | *- Get List Category Post*  *-Return: List Category Post* |
| *02* | *getListSubCategoryById()* | *- Get List SubCategory Post*  *-Input: idCategory*  *-Return: List SubCategory Post* |

#### PostDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getPostByIdIncludeCategory()* | *- Get post Details*  *-Input: idPost*  *-Return: A Post* |
| *02* | *editPost()* | *- Get post Details*  *-Input: idPost, thumbnail, title, briefInfor, description, date, idUser, featured, status, idCategory, idSubCategory*  *-Return: A Post* |

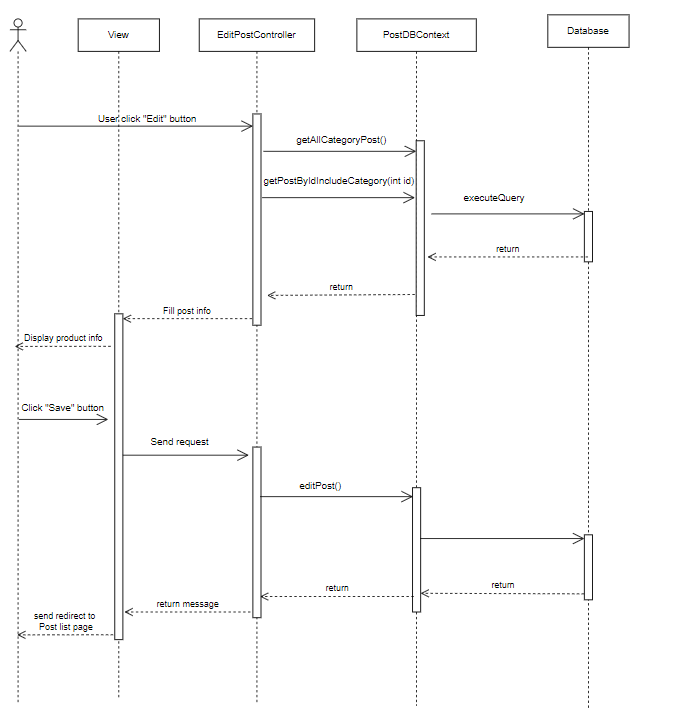
#### 

#### EditPostController:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

### c. Sequence Diagram(s)



### d. Database queries

* PostDBContext
* Select p.id, p.thumbnail, p.title,p.briefInfo, p.[description], p.feature, p.[status], p.UserId, s.email, s.fullname, p.categoryId as idSubCategory, sp.[name] as nameSubCategory, cp.id as idCategoryPost, cp.[name] as nameCategoryPost, p.dateUpdated

From post p join [user] s on p.UserId = s.id

join SubCategoryPost sp on p.categoryId = sp.id

join CategoryPost cp on cp.id = sp.idcategory and p.id = ?

* UPDATE [dbo].[Post]

SET

[title] = ? ,

[categoryId] = ?,

[briefInfo] = ?,

[description] = ?,

[feature] = ?,

[dateUpdated] = ?,

[status] = ?,

[UserId] = ?,

[thumbnail] = ?

WHERE [id] = ?

* CategoryPostDBContext
* Select \* from CategoryPost
* Select \* from SubCategoryPost Where idcategory = ?

## 

## 

## 

## 

## 

## 

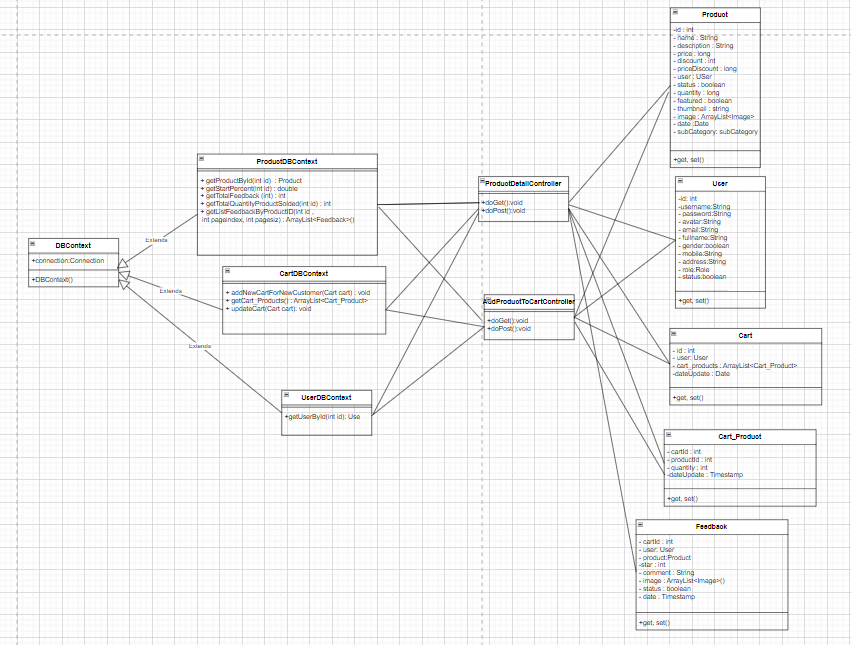
## 

## 

## 

## 38.Public feature/ Product details

## a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### ProductDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | *getProductById(int id)* | *-Get product infomation*  *-Input : Product Id*  *- Return : A product info* |
| *2* | *getStartPercent(int id)* | *-Get average star of feedbacks of a product*  *-Input : Product id*  *- Return: number percent of start* |
| *3* | *getTotalFeedback (int id)* | *-Get total feedback of a prouduct*  *-Input : Product Id*  *- Return: Number of feedback of product* |
| *4* | *getTotalQuantityProductSolded(int id)* | *-Get quantity solded of a product*  *-Input: Product Id*  *-Retrun: Number quantityt solded* |
| *5* | *getListFeedbackByProductID(int id ,*  *int pageindex, int pagesiz)* | *-Get list Feedback information*  *-Input : Product Id, page index, page size*  *-Return: List feedbak* |

#### CartDBContext

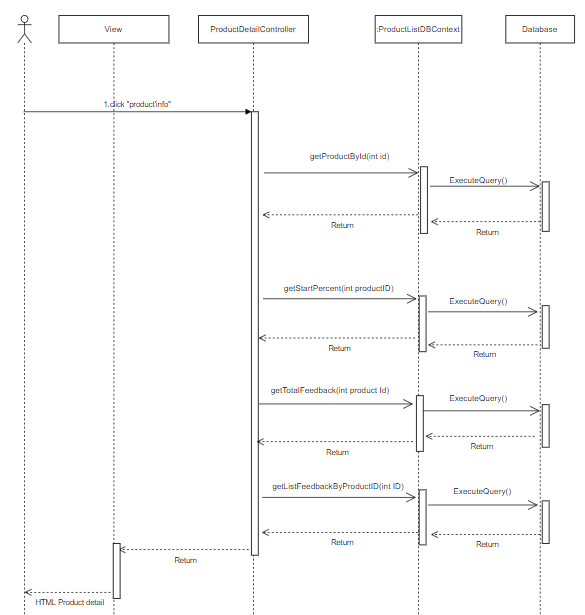
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+addNewCartForNewCustomer(Cart cart)* | *-Add a cart info to database*  *-Input : Cart information*  *-Return :* |
| *02* | *+getCart\_Products(int id) : ArrayList<Cart\_Product>* | *-Get all product of cart*  *- input : customer id*  *-Return: list Cart\_Product* |
| *03* | *updateCart(Cart cart)* | *-Update a cart info to database*  *-Input : Cart information*  *-Return :* |

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getUserById(int id)* | *- Get a user*  *Input:User Id*  *Return: A User where user id contain parameter* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* getProductById(int id):

SELECT [Product].[id]

,[Product].[name]

,[description]

,[price]

,[discount]

,[sellerId]

,[User].[fullname]

,[featured]

,[thumbnail]

,[date]

,[subCategoryId]

,[SubCategory].[name]

,[Category].[id]

,[Category].[name]

,[quantity]

,[Product].[status]

,[User].email

,[User].avatar

FROM [dbo].[Product] join SubCategory on subCategoryId = SubCategory.id

join Category on Category.id = SubCategory.categoryId

join [User] on [User].id = sellerId

Where [Product].[id] = ?

* getStartPercent(int id):

SELECT

[start]

FROM [Feedback] join Product on Feedback.productId = Product.id

Where [Feedback].[status] = 1

and Product.id = ?

* getTotalFeedback(int id):

SELECT

Count(\*)

FROM [Feedback] join Product on Feedback.productId = Product.id

Where [Feedback].[status] = 1

and Product.id = ?

* getTotalQuantityProductSolded(int id):

select Count([OrderDetail].quantity)

from

OrderDetail join [Order] on [Order].id = OrderDetail.orderId

where [Order].[status] = 1 and OrderDetail.productId = ?

* getListFeedbackByProductID(int id):

with x as (select ROW\_NUMBER() OVER (ORDER BY f.id desc) as r

, f.id, f.userId, f.productId, f.start,f.comment ,f.image, f.status, u.fullname, u.avatar, f.date from Feedback f INNER JOIN [User] u

ON f.userId = u.id

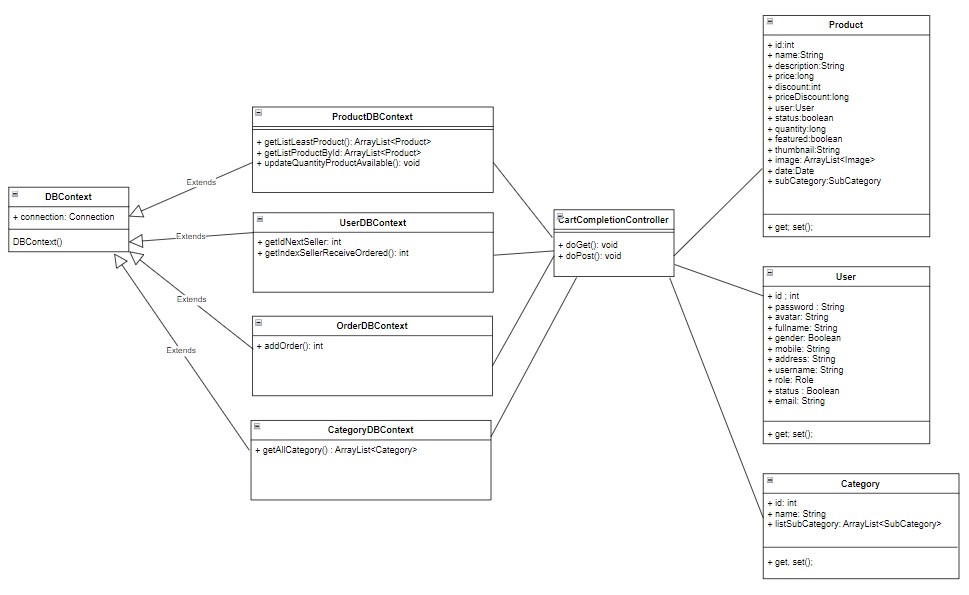
where f.status = 1

AND f.productId = ? )

SElECT\* FROM x where r between (? - 1)\* ? +1 and ? \* ?

## 39.Public feature/ Cart Completion

## a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### 

#### ProductDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getListLeastProduct()* | *-get list latest product*  *-Return : list product* |
| *02* | *+getListProductById()* | *-Get product order*  *- input : Product[] productOrders, int idUser*  *-Return : list product* |
| *03* | *+updateQuantityProductAvailable()* | *-Update quantity product available*  *-Input : Product[] productOrders* |

#### 

#### UserDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getIdNextSeller()* | *-Get Id of next seller*  *- input : int indexNextSellerReceiveOrder*  *-Return : id* |
| *02* | *+getIndexSellerReceiveOrdered()* | *-Get index of seller receiver ordered previous*  *-input: int lastSellerReceiveOrders*  *-Return : index* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+addOrder()* | *- Add order*  *- input : Product[] productOrders, int total, string fullname, string address, string phone, string note, int idPayment, int idSeller, string email, int idCustomer* |

#### CategoryDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+getAllCategory()* | *- Get list category product*  *- Return list Category product* |

#### 

### 

### c. Sequence Diagram(s)

### 

### d. Database queries

* ProductDBContext
* select top 3 \* from [Product] where status = 1 and quantity > 0 order by [date] desc, id desc
* Select cart.\*, Cart\_Product.ProductId, Cart\_Product.Quantity, product.thumbnail, Product.[name], product.sellerId, [User].fullname,

Product.price, product.discount

from cart join Cart\_Product

on cart.id = Cart\_Product.CartId

join Product on Product.id = Cart\_Product.ProductId

join [User] on [User].id = product.sellerId

where cart.customerId = ? and Cart\_Product.ProductId = ?

* select id, quantity from Product where id = ?

UPDATE [dbo].[Product] SET [quantity] = ? WHERE id = ?

* UserDBConext
* Select \* from

( Select id, ROW\_NUMBER() over(order by id asc) as row\_index from [User]

where roleId = 3 and status = 1) indexSeller

where id = ?

* Select \* from

( Select id, ROW\_NUMBER() over(order by id asc) as row\_index from [User]

where roleId = 3 and status = 1) indexSeller where row\_index = ?

* OrderDBContext
* INSERT INTO [dbo].[ShipInfo]

([fullname] ,[address],[phone] ,[email])

VALUES(?,?,?,?)

SELECT @@IDENTITY AS idShip

INSERT INTO [dbo].[Order]

([userId]

,[totalPrice]

,[note]

,[status]

,[date]

,[idShip]

,[payment]

,[sellerid])

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

SELECT @@IDENTITY AS id

INSERT INTO [dbo].[OrderDetail]

([orderId]

,[productId]

,[discount]

,[quantity]

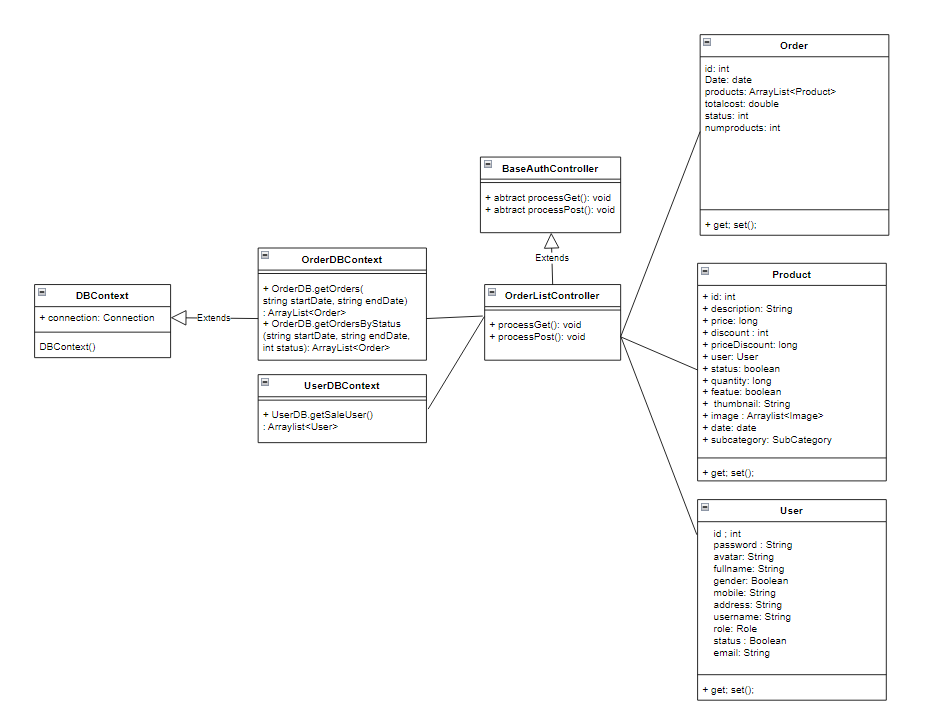
,[price])

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

* CategoryDBContext
* Select id, name from Category

## 40. Sale / OrderList

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getOrders(string startDate, string endDate)* | *Get all user ordered in the system by date range* |
| *02* | *getOrdersByStatus(string startDate, string endDate, int status)* | *Get all user ordered in the system by date range and status code.* |

#### 

* **UserDBContext**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSaleUser()* | *Get all user role is SALE.* |

#### 

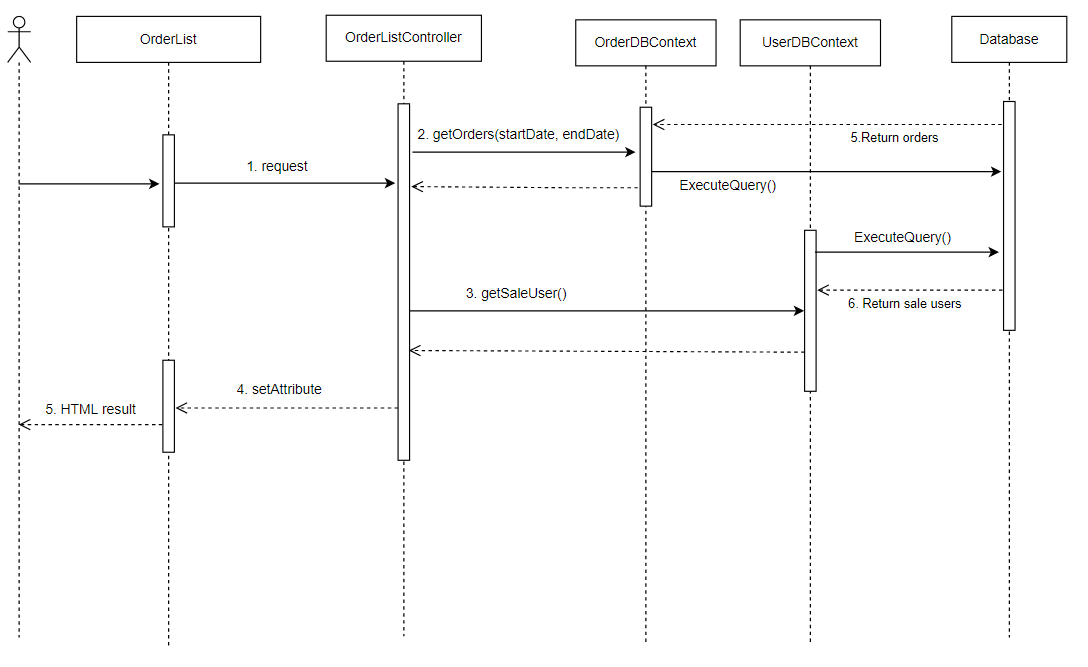
#### OrderListController:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* **OrderDBContext**
* WITH

t as

(SELECT [Order].id as OrderID,[User].fullname as CustomerName,[User].id as CustomerID, [Order].[date], [Order].totalPrice, [Product].[name] as ProductName, [Order].[status] as OrderStatus, [Order].sellerId

FROM

[User] inner join [Order] on [User].id = [Order].userId

inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id)

b as

(SELECT [Order].id as OrderID, COUNT(Product.id) as NumberOfProducts

FROM [Order] inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id

group by [Order].id),

c as

(

SELECT a.\*

FROM (

SELECT DISTINCT t.OrderID

FROM t

) mo

CROSS APPLY

(

SELECT TOP 1 \*

FROM t mi

WHERE mi.OrderID = mo.OrderID

) a

)

Select c.OrderID, c.[date], c.totalPrice, c.CustomerID, c.CustomerName, c.ProductName, c.OrderStatus, b.NumberOfProducts, c.sellerId, [User].fullname as SellerName

from c inner join b on c.OrderID = b.OrderID

left outer join [User] on c.sellerId = [User].id

WHERE c.[date] between ? and ?";

* WITH

t as

(SELECT [Order].id as OrderID,[User].fullname as CustomerName,[User].id as CustomerID, [Order].[date], [Order].totalPrice, [Product].[name] as ProductName, [Order].[status] as OrderStatus, [Order].sellerId

FROM

[User] inner join [Order] on [User].id = [Order].userId

inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id),

b as

(SELECT [Order].id as OrderID, COUNT(Product.id) as NumberOfProducts

FROM [Order] inner join OrderDetail ON [Order].id = OrderDetail.orderId

inner join Product ON OrderDetail.productId = Product.id

group by [Order].id),

c as

(

SELECT a.\*

"FROM (

SELECT DISTINCT t.OrderID

FROM t

) mo

CROSS APPLY

(

SELECT TOP 1 \*

FROM t mi

WHERE mi.OrderID = mo.OrderID ) a

)

Select c.OrderID, c.[date], c.totalPrice, c.CustomerID, c.CustomerName, c.ProductName, c.OrderStatus, b.NumberOfProducts, c.sellerId, [User].fullname as SellerName

from c inner join b on c.OrderID = b.OrderID

left outer join [User] on c.sellerId = [User].id

WHERE c.[date] between ? and ? and c.OrderStatus = ?

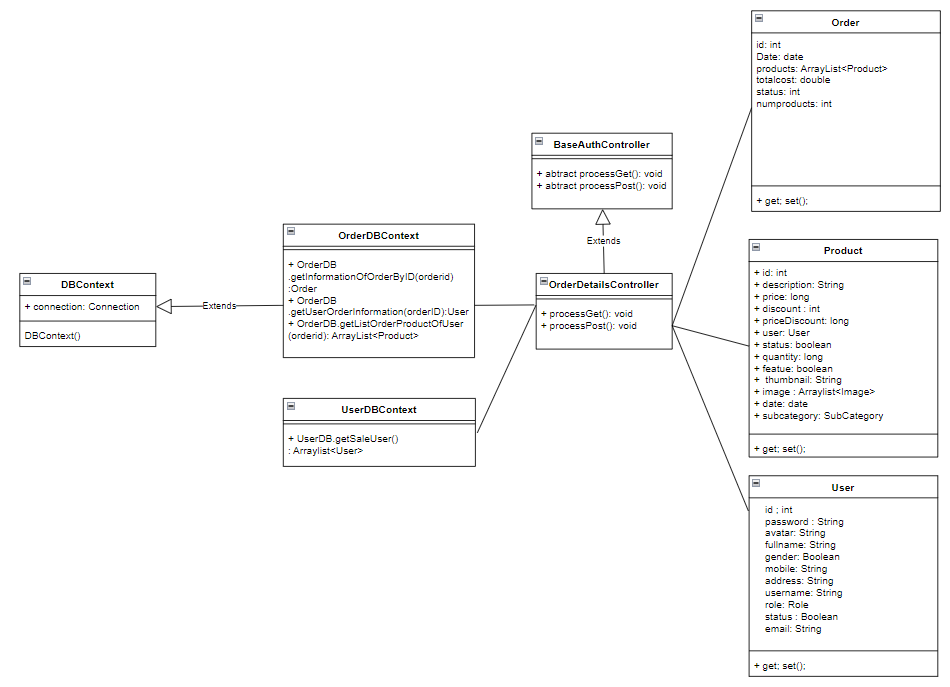
* **UserDBContext:**

Select \* from [User]

Where roleId = 2

## 41. Sale / OrderDetails

### a. Class Diagram



### b. Class Specifications

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getInformationOfOrderByID(orderID)* | *Get order info in the system by id.* |
| *02* | *getUserOrderInformation(orderID)* | *Get user info of the order.* |
| *02* | *getListOrderProductOfUser(orderID)* | *Get all products that purchased by the customer in an order.* |

#### 

* **UserDBContext**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getSaleUser()* | *Get all user role is SALE.* |

#### 

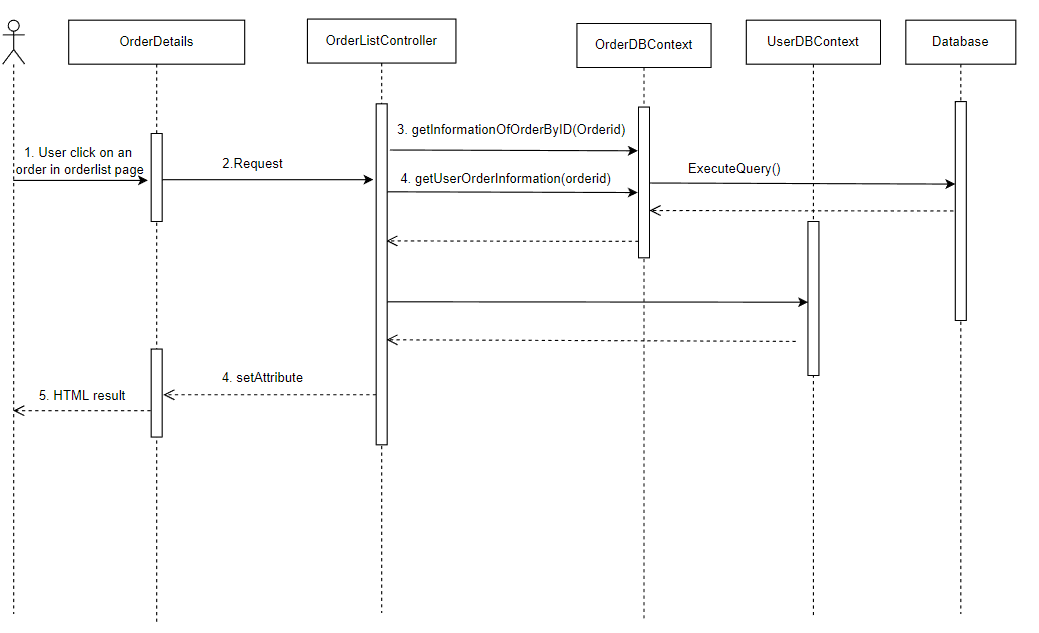
#### OrderListController:

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |
| *02* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *-Input: HttpServletRequest,ServletResponse response* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* **OrderDBContext:**
* select DISTINCT od.id, od.date, od.status, od.totalPrice, od.sellerid, [User].fullname as sellerName, od.note as CustomerNote, od.sellernote as SaleNote

, od.cancelledReason

from

[Order] od inner join [User] on od.sellerid = [User].id

WHERE od.id = ?

* select DISTINCT s.fullname, u.gender, s.email, s.phone

from

[Order] od

INNER JOIN ShipInfo s ON od.idShip = s.id

INNER JOIN [User] u on od.userId = u.id

WHERE od.id = ?

* select p.id as productID, p.thumbnail, p.name as pname, c.name as categoryName, o.quantity, o.discount, o.price, od.userId

from

[Order] od

INNER JOIN OrderDetail o ON o.orderId = od.id

INNER JOIN Product p ON o.productId = p.id

INNER JOIN SubCategory sub ON p.subCategoryId = sub.id

INNER JOIN Category c ON sub.categoryId = c.id

WHERE o.orderId = ?

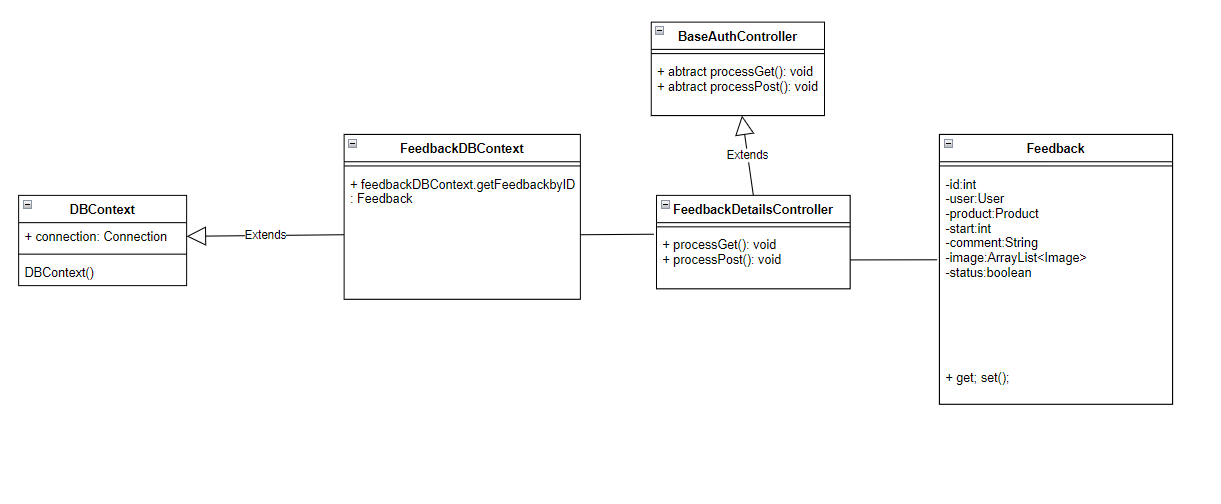
* **UserDBContext**

Select \* from [User]

Where roleName = ‘Sale’

## 42. Marketing / FeedbackDetails

### a. Class Diagram



### b. Class Specifications

#### DBContext

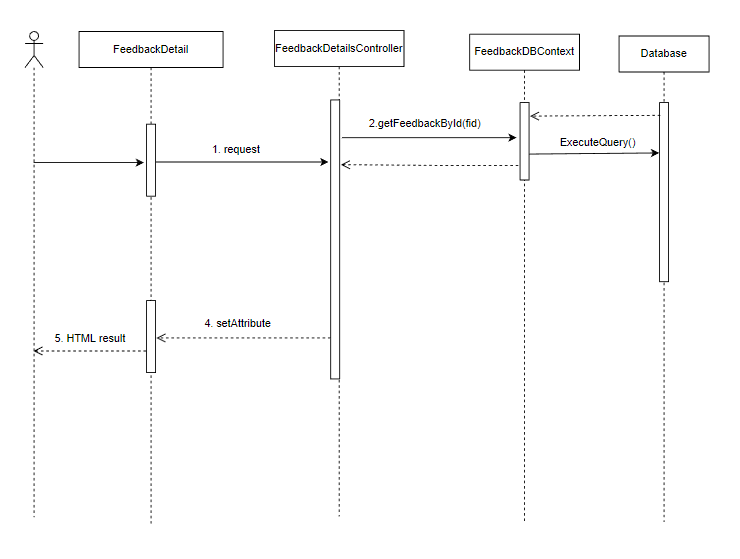
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

* **FeedbackDBContext**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getFeedbackById(fid)* | *Get feedback details by ID.* |

#### 

### c. Sequence Diagram(s)



### d. Database queries

* **FeedbackDBContext:**

SELECT [Feedback].id

,[User].[fullname] as CustomerName

,[User].id as [uid]

,[User].mobile as UserMobile

,[User].email as UserEmail

,[Product].[name] as ProductName

,[Product].id as ProductId

,[Product].thumbnail as ProductThumbnail

,[start]

,[comment]

,[Feedback].[date]

,[Feedback].[status]

FROM [dbo].[Feedback] join [User] on [User].id = Feedback.userId

join Product on Product.id = Feedback.productId

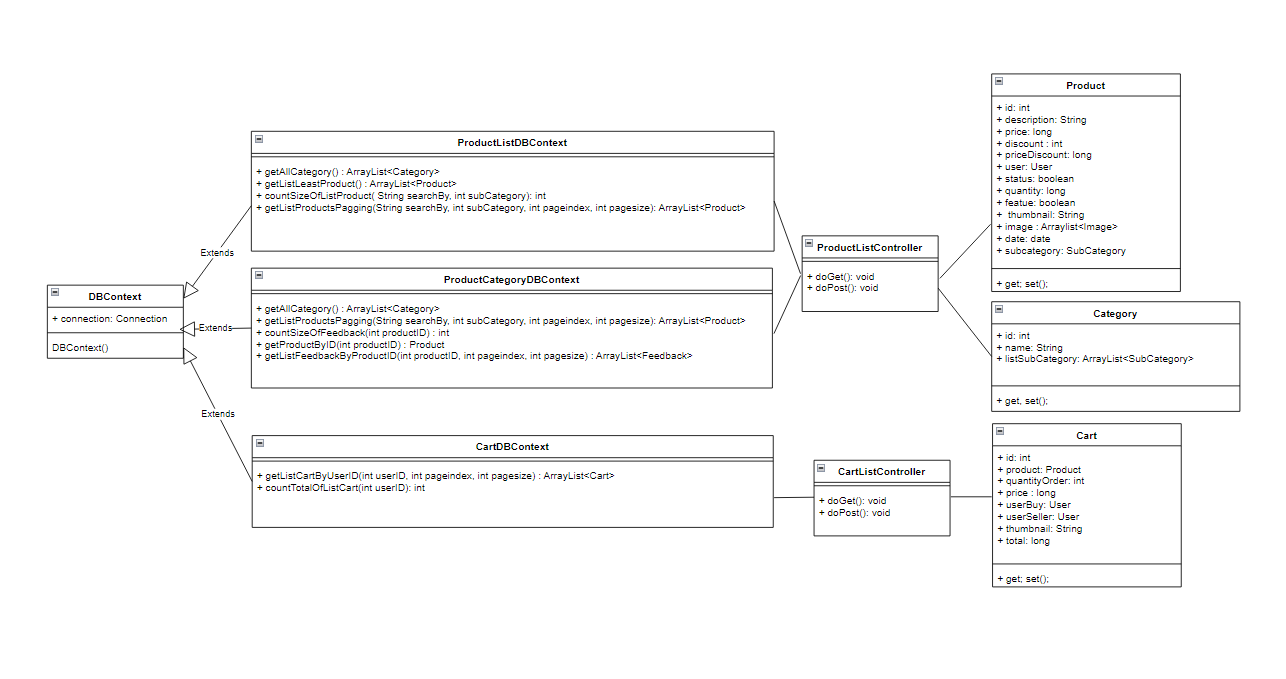
join SubCategory on Product.subCategoryId = SubCategory.id

join Category on Category.id =SubCategory.categoryId

WHERE [Feedback].id = ?

## 43. Public/Product List

### a. Class Diagram



### b. Class Specifications

#### DBContext

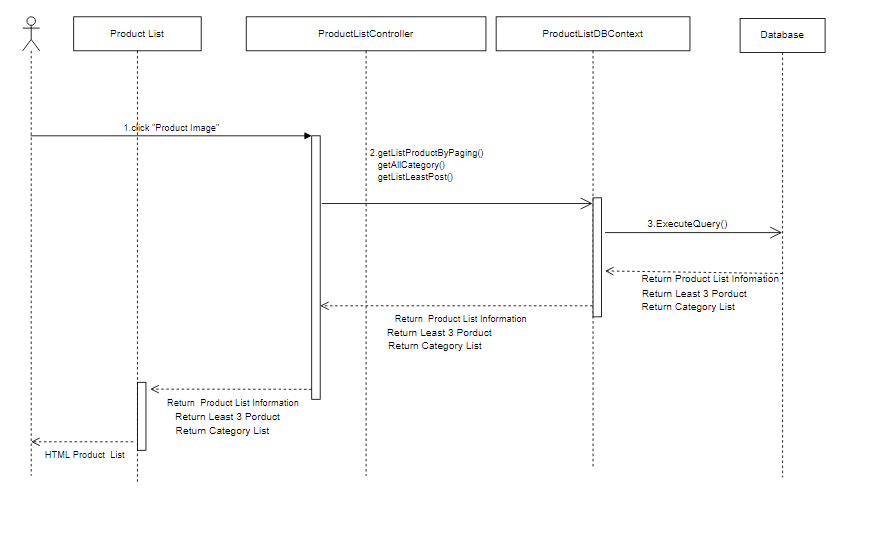
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+ getAllCategory() : ArrayList<Category>* | *-Get list of all category*  *-Return : List of all category* |
| *02* | *+getListLeastProduct(): ArrayList<Product>* | *-Get list of 3 newest product*  *-Return : List of 3 newest product* |
| *03* | *+ArrayList<Product> getListProductsPagging()* | *-Get list of all product*  *-Return : List of all product list* |

### 

### c. Sequence Diagram(s)



### d. Database queries

* Select id, name from Category
* select top 3 \* from [Product]

where status = 1 and quantity > 0

order by [date] desc, id desc

* with x as (select ROW\_NUMBER() OVER (ORDER BY date desc, id desc) as r

, \* from [Product] where status = 1 and subCategoryId = ?

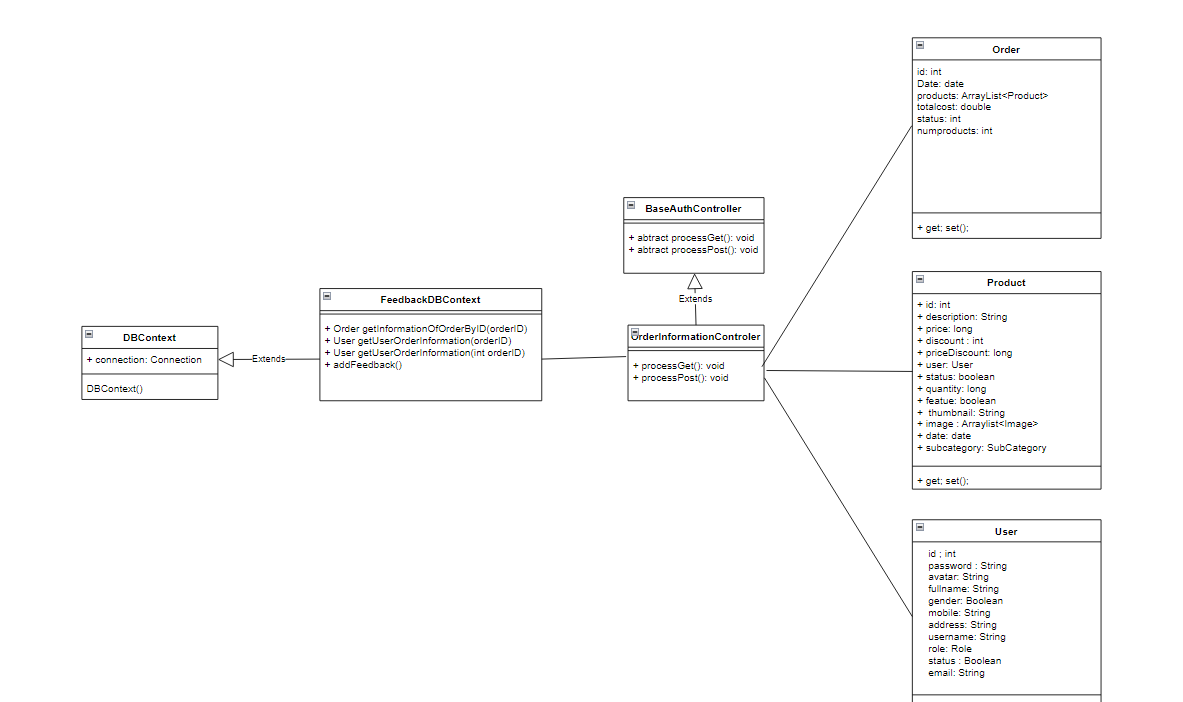
AND (name like ?

OR description like ? ))

SElECT\* FROM x where r between (? - 1)\* ? +1 and ? \* ?

## 44. FeedBack

### a. Class Diagram



### b. Class Specifications

#### DBContext

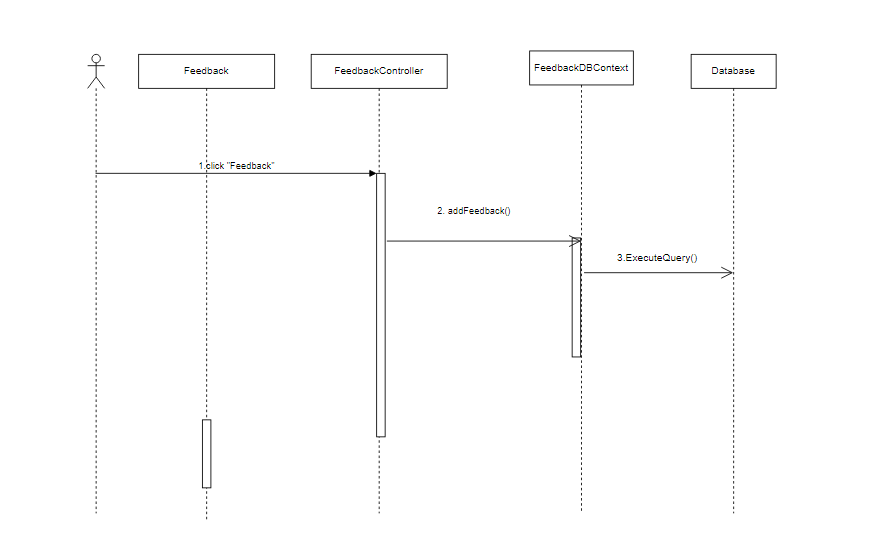
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext()* | *-Get connection with database* |

#### OrderDBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *+ getAllCategory() : ArrayList<Category>* | *-Get list of all category*  *-Return : List of all category* |
| *02* | *+getListLeastProduct(): ArrayList<Product>* | *-Get list of 3 newest product*  *-Return : List of 3 newest product* |
| *03* | *+ArrayList<Product> getListProductsPagging()* | *-Get list of all product*  *-Return : List of all product list* |
| *04* | *+addFeedback()* | *–Add feedback information for product* |

### 

### c. Sequence Diagram(s)



### d. Database queries

INSERT INTO [dbo].[Feedback]

([userId]

,[productId]

,[start]

,[comment]

,[image]

,[status]

,[date])

VALUES

(?

,?

,?

,?

,?

,?

,?)

# III. Database Design

## 1. Database Schema

*[Provide the tables relationship like example below – following MySQL database naming convention]*

## 

## 2. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | User | Store the information of all users  - Primary keys: userID  - Foreign keys: roleId |
| 02 | Role | Store the role for users.  -Primary key: id |
| 03 | Blog | Store the information of all blogs  - Primary keys: id  - Foreign keys: userID, categoryID |
| 04 | BlogCategory | Contain blog category  -Primary key : id |
| 05 | Feedback Replies | Store feedback replies  - Primary keys: id  - Foreign keys: userID |
| 06 | Feedback | Store feedback of customer  - Primary keys: id  - Foreign keys: productID, orderID |
| 07 | Order | Store information of orders  -Primary key: id  - Foreign keys: userID, statusID |
| 08 | OrderStatus | Store all order status  - Primary keys: id |
| 09 | OrderDetail | Store detail of each order  - Primary key: id  - Foreign key: orderID, productID |
| 10 | Product | Store the information of all products  - Primary keys: productID  - Foreign keys: subCategoryID, sellerID |
| 11 | SubCategory | Contain subcategory of product  - Primary keys: subCategoryID  - Foreign keys: categoryID |
| 12 | Category | Contain category of subcategory  - Primary keys:categoryID |