

**TECH SHOP ONLINE SYSTEM**

**Software Design Document**

**Group 3**

– Can Tho, January 2023 –

**Table of Contents**

Table of Contents

[I. Overview 6](#_Toc130309027)

[1. Code Packages/Namespaces 6](#_Toc130309028)

[2. Coding conventions 7](#_Toc130309029)

[2. ERD 8](#_Toc130309030)

[3. Database Scheme 11](#_Toc130309031)

[II. Code Designs 14](#_Toc130309032)

[1. Home Page 14](#_Toc130309033)

[a. Class Diagram 14](#_Toc130309034)

[b. Class Specifications 15](#_Toc130309035)

[c. Sequence Diagram(s) 17](#_Toc130309036)

[d. Database queries 17](#_Toc130309037)

[2. User Login 18](#_Toc130309038)

[a. Class Diagram 18](#_Toc130309039)

[b. Class Specifications 18](#_Toc130309041)

[c. Sequence Diagram(s) 20](#_Toc130309042)

[d. Database queries 20](#_Toc130309043)

[3. User Profile 21](#_Toc130309044)

[a. Class diagram 21](#_Toc130309045)

[b. Class Specifications 21](#_Toc130309046)

[c. Sequence Diagram 22](#_Toc130309047)

[d. Database query 22](#_Toc130309048)

[4. Change Password 23](#_Toc130309049)

[a. Class Diagram 23](#_Toc130309050)

[b. Class Specifications 23](#_Toc130309051)

[c. Sequence Diagram(s) 25](#_Toc130309052)

[d. Database queries 25](#_Toc130309053)

[5. Reset password 26](#_Toc130309054)

[a. Class Diagram 26](#_Toc130309055)

[b. Class Specifications 26](#_Toc130309056)

[c. Sequence Diagram(s) 27](#_Toc130309057)

[d. Database queries 28](#_Toc130309058)

[6. User register 28](#_Toc130309059)

[a. Class Diagram 28](#_Toc130309060)

[b. Class Specifications 29](#_Toc130309061)

[c. Sequence Diagram(s) 30](#_Toc130309062)

[d. Database queries 31](#_Toc130309063)

[7. Product List 31](#_Toc130309064)

[a. Class Diagram 31](#_Toc130309065)

[b. Class Specifications 31](#_Toc130309066)

[c. Sequence Diagram(s) 33](#_Toc130309067)

[d. Database queries 33](#_Toc130309068)

[8.Product Details 34](#_Toc130309069)

[a. Class Diagram 34](#_Toc130309070)

[b. Class Specifications 34](#_Toc130309071)

[c. Sequence Diagram(s) 35](#_Toc130309072)

[d. Database queries 36](#_Toc130309073)

[9. Cart Details 36](#_Toc130309074)

[a. Class diagram 36](#_Toc130309075)

[b. Class Specifications 37](#_Toc130309076)

[c. Sequence Diagram 38](#_Toc130309077)

[d. Database query 38](#_Toc130309078)

[10. Cart Contact 39](#_Toc130309079)

[a. Class diagram 39](#_Toc130309080)

[b. Class Specifications 39](#_Toc130309081)

[c. Sequence Diagram 41](#_Toc130309082)

[d. Database query 41](#_Toc130309083)

[11. Cart Completion 42](#_Toc130309084)

[a. Class diagram 42](#_Toc130309085)

[b. Class Specifications 42](#_Toc130309086)

[c. Sequence Diagram 44](#_Toc130309087)

[d. Database query 44](#_Toc130309088)

[12.Blogs List 45](#_Toc130309089)

[a.Class diagram 45](#_Toc130309090)

[b.Class specification 45](#_Toc130309091)

[c.Sequence diagram 46](#_Toc130309092)

[d.Database query 47](#_Toc130309093)

[13.Blog Details 47](#_Toc130309094)

[a. Class Diagram 47](#_Toc130309095)

[b. Class Specifications 47](#_Toc130309096)

[c. Sequence Diagram(s) 49](#_Toc130309097)

[d. Database queries 49](#_Toc130309099)

[14. Order List 50](#_Toc130309100)

[a. Class diagram 50](#_Toc130309101)

[b. Class Specifications 50](#_Toc130309102)

[c. Sequence Diagram 52](#_Toc130309103)

[d. Database query 53](#_Toc130309104)

[15. Order Information 54](#_Toc130309105)

[a. Class diagram 54](#_Toc130309106)

[b. Class Specifications 54](#_Toc130309107)

[c. Sequence Diagram 58](#_Toc130309108)

[d. Database query 59](#_Toc130309109)

[16. Feedback 60](#_Toc130309110)

[a. Class diagram 60](#_Toc130309111)

[b. Class Specifications 60](#_Toc130309112)

[c. Sequence Diagram 63](#_Toc130309113)

[d. Database query 64](#_Toc130309114)

[17. DashBoard 64](#_Toc130309115)

[a. Class diagram 64](#_Toc130309116)

[b. Class Specifications 65](#_Toc130309117)

[c. Sequence Diagram 66](#_Toc130309118)

[d. Database query 66](#_Toc130309119)

[18. Slider List 67](#_Toc130309120)

[a. Class diagram 67](#_Toc130309121)

[b. Class Specifications 67](#_Toc130309122)

[c. Sequence Diagram 69](#_Toc130309123)

[69](#_Toc130309124)

[d. Database query 69](#_Toc130309125)

[19. Product List 70](#_Toc130309126)

[a. Class diagram 70](#_Toc130309127)

[b. Class Specifications 70](#_Toc130309128)

[c. Sequence Diagram 72](#_Toc130309129)

[d. Database query 72](#_Toc130309131)

[20. Post List 73](#_Toc130309132)

[a. Class Diagram 73](#_Toc130309133)

[b. Class Specifications 74](#_Toc130309134)

[c. Sequence Diagram(s) 75](#_Toc130309135)

[d. Database queries 75](#_Toc130309136)

[21. Dashboard 76](#_Toc130309137)

[a. Class diagram 76](#_Toc130309138)

[b. Class Specifications 76](#_Toc130309139)

[c. Sequence Diagram 78](#_Toc130309140)

[d. Database query 79](#_Toc130309141)

[22. Slider List 80](#_Toc130309142)

[a. Class Diagram 80](#_Toc130309143)

[b. Class Specifications 80](#_Toc130309144)

[c. Sequence Diagram(s) 81](#_Toc130309145)

[d. Database queries 81](#_Toc130309146)

[23. Slider Detail 82](#_Toc130309147)

[a. Class Diagram 82](#_Toc130309148)

[b. Class Specifications 82](#_Toc130309149)

[c. Sequence Diagram(s) 83](#_Toc130309150)

[d. Database queries 83](#_Toc130309151)

[24. Post detail 84](#_Toc130309152)

[a. Class Diagram 84](#_Toc130309153)

[b. Class Specifications 84](#_Toc130309154)

[c. Database queries 84](#_Toc130309155)

[III. Database Tables 85](#_Toc130309156)

[1. user 85](#_Toc130309157)

[2. order 85](#_Toc130309158)

[3. order\_detail 86](#_Toc130309159)

[4. cart 86](#_Toc130309160)

[5. settings. 86](#_Toc130309161)

[6. Role 87](#_Toc130309162)

[7.slide 87](#_Toc130309163)

[8. product 87](#_Toc130309164)

[9. cart\_detail 87](#_Toc130309165)

[10. blog 88](#_Toc130309166)

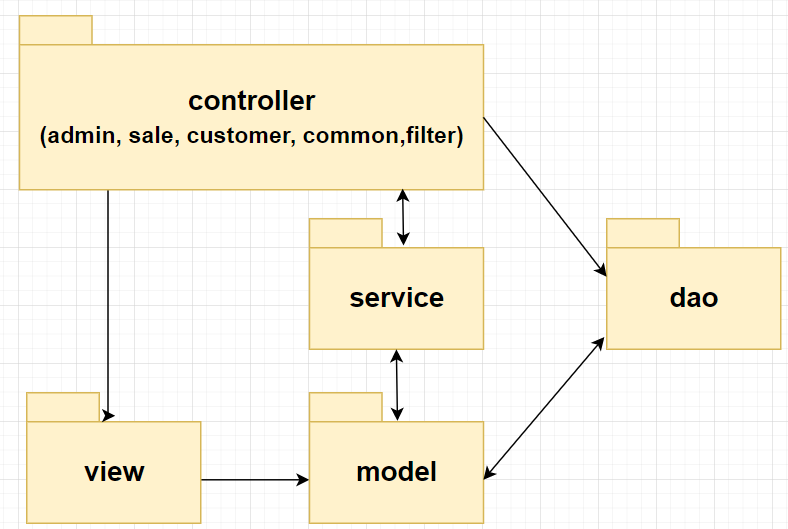
[11. feedback 88](#_Toc130309167)

[12. RoleAuthor 88](#_Toc130309168)

[13. Feature 88](#_Toc130309169)

# I. Overview

## 1. Code Packages/Namespaces



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

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 01 | dao | Contains, dbcontext use for connect to sql server and all classes to query the database. |
| 02 | view | Contains js, css, javascript to display user interfaces. |
| 03 | controller | In the controller package it contains 6 sub-folder is admin , marketing, sale, user, common, filter for admin controller, marketing controller, sale controller, user controller, common controller respectively. |
| 04 | service | Contains class or API use for support function in controller. |
| 05 | model | Contains all class that represent properties of data objects. |

## 2. Coding conventions

**Package**

* The package name needs to be set with the meaning corresponding to the classes.
* Not capitalized and is a noun.

Example:package student management , package countletter,....

**Classes**

* Name with the first letter of the word must be capitalized
* The name must have a certain meaning.

Example: Class Student, Class Doctor,....

**Variable**

* The variable name must properly represent the function and meaning of the variable.
* The first letter is a word that is not capitalized, the second word begins to be capitalized. For example: int inputNumber, int outputNumber,...
* The property variables in the Class are required to be private to increase information security. For example: private int inputNumber: private int outputNumber,...

**Method**

* The first letter of the method name must be lowercase and a verb, the letters from the second letter will be uppercase, the method name must have meaning. For example: getNumber();....
* The methods must have the right type of access for each purpose. For example, if only used in the class, it must be private, if it is used outside the package, it should be public.
* Comment the complete function with its function, input parameter, output parameter along with its data type.

**Format**

* Format code after done

## 2. ERD

## 3. Database Scheme

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

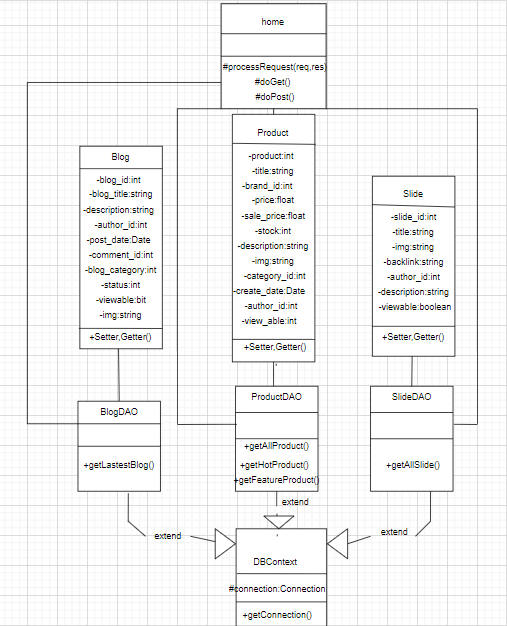
|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| *01* | *user* | *[user\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[fullname] [nvarchar](50) NULL,*  *[email] [nchar](50) NOT NULL,*  *[pasword] [nchar](20) NOT NULL,*  *[gender] [bit] NULL,*  *[phone] [nchar](11) NULL,*  *[address] [nvarchar](100) NULL,*  *[role] [int] NULL,*  *[active] [bit] NULL,*  *[[key]] [nvarchar](50) NULL,*  *[avatar] [ntext] NULL,* |
| *02* | *product* | *[product\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[title] [nvarchar](50) NOT NULL,*  *[brand\_id] [int] NULL,*  *[price] [float] NOT NULL,*  *[sale\_price] [float] NULL,*  *[stock] [int] NULL,*  *[description] [ntext] NULL,*  *[img] [ntext] NULL,*  *[category\_id] [int] NULL,*  *[create\_date] [date] NULL,*  *[author\_id] [int] NOT NULL,*  *[active] [int] NULL,* |
| *03* | *feedback* | *[feedback\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[product\_id] [int] NOT NULL,*  *[user\_id] [int] NOT NULL,*  *[rated] [int] NULL,*  *[fb\_content] [ntext] NULL,*  *[img] [ntext] NULL,*  *[feedback\_date] [date] NULL,*  *[active] [bit] NULL,*  *[feedback] – [FK\_feedback\_product] FOREIGN KEY([product\_id])* |
| *04* | *cart* | *[cart\_id] [int] IDENTITY(1,1) NOT NULL,*  *[user\_id] [int] NOT NULL,*  *[total\_price] [float] NULL,*  *[cart] – [FK\_cart\_user] FOREIGN KEY([user\_id])* |
| *05* | *cart\_detail* | *[cart\_detail\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[car\_id] [int] NOT NULL,*  *[product\_id] [int] NOT NULL,*  *[quantity] [int] NULL,* |
| *06* | *order* | *[order\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[user\_id] [int] NOT NULL,*  *[fullname] [nvarchar](50) NOT NULL,*  *[address] [nvarchar](100) NOT NULL,*  *[phone] [nchar](11) NOT NULL,*  *[order\_date] [date] NULL,*  *[total] [float] NULL,*  *[payment\_id] [int] NULL,*  *[status] [int] NULL,*  *[active] [bit] NULL,*  *[notes] [ntext] NULL,*  *[dbo].[order] – [FK\_order\_user] FOREIGN KEY([user\_id])* |
| *07* | *order\_detail* | *[order\_detail\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[order\_id] [int] NULL,*  *[quantity] [int] NULL,*  *[product\_id] [int] NULL,*  *[author\_id] [int] NOT NULL,*  *[title] [nvarchar](50) NOT NULL,*  *[brand\_id] [int] NULL,*  *[price] [float] NOT NULL,*  *[sale\_price] [float] NULL,*  *[stock] [int] NULL,*  *[description] [ntext] NULL,*  *[img] [ntext] NULL,*  *[category\_id] [int] NULL,*  *[order\_detail] – [FK\_order\_detail\_order] FOREIGN KEY([order\_id])* |
| *08* | *blog* | *[blog\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[blog\_title] [nvarchar](50) NOT NULL,*  *[description] [ntext] NOT NULL,*  *[img] [ntext] NULL,*  *[author\_id] [int] NOT NULL,*  *[post\_date] [date] NULL,*  *[blog\_category] [int] NULL,*  *[status] [int] NULL,*  *[viewable] [bit] NULL,*  *[blog] – [FK\_blog\_user] FOREIGN KEY([author\_id])* |
| *10* | *setting* | *[setting\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[type] [nvarchar](50) NULL,*  *[value] [nvarchar](50) NULL,*  *[orders] [int] NULL,*  *[active] [bit] NULL,* |
| *11* | *slide* | *[slide\_id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[title] [nvarchar](50) NULL,*  *[img] [ntext] NULL,*  *[backlink] [ntext] NULL,*  *[author\_id] [int] NOT NULL,*  *[description] [ntext] NULL,*  *[active] [bit] NULL,*  *[slide] – [FK\_slide\_user] FOREIGN KEY([author\_id])* |
| *12* | *Role* | *[id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[name] [nvarchar](100) NULL,* |

|  |  |  |
| --- | --- | --- |
| *13* | *RoleAuthor* | *[roleID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,*  *[active] [int] NOT NULL,*  *[feature] [int] NOT NULL,* |

# II. Code Designs

## 1. Home Page

### a. Class Diagram

******

### b. Class Specifications

**Product Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Product()* | *Constructor without parameters* |
| *02* | *Product(int pid, String title,int brand\_id, float price,float sale\_price,int stock, String description, String image, int category\_id, String brief, Date create\_date, int author\_id,int viewable)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**Blog Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Blog() | Constructor without parameters |
| 02 | Blog(int blog\_id, String blog\_title, String description, int author\_id, int comment\_id, int blog\_category, int status,boolean viewable ,String image) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

**Slide Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Slide() | Constructor without parameters |
| 02 | Slide(int slide\_id, String title, String image, String backlink, int author\_id,String description,boolean viewable) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

**ProductDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getHotProduct() | Show hot products |
| 02 | getAllProduct() | Show products |
| 03 | getFeaturedGood() | Show featured products |

**SlideDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllSliderByViewable() | Show all slider image |

**BlogDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getLastestBlogs() | Show some latest blogs |

**DBContext Class**

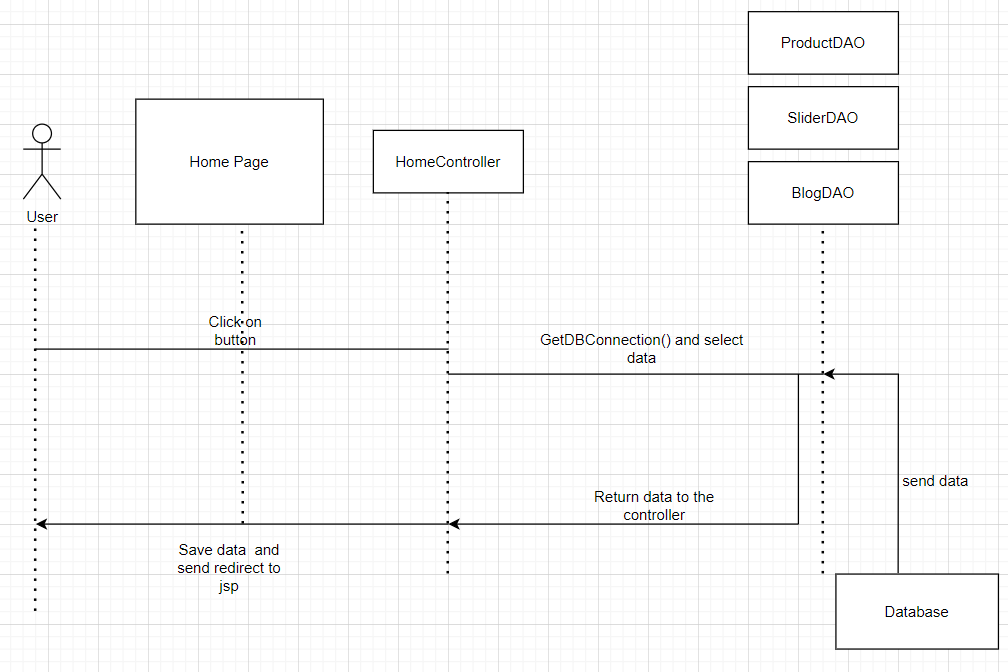
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connect to the database |

**HomeController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet | Method GET, redirect to view page |
| 02 | doPost | Default function |

### 

### c. Sequence Diagram(s)



### d. Database queries

|  |  |
| --- | --- |
| **Function/screen** | **Query** |
| getAllSlidersByViewable() | select s.img, s.title, s.description, s.backlink from slide s WHERE s.viewable = 1 |
| getAllProductByViewable() | select top(10) s.img, s.title, s.description, s.backlink from product s WHERE s.viewable = 1 |
| getLastestProduct() | select top(10) image,title,description,backlink from Product  where viewable = 1  order by create\_date desc |
| getLatestPost() | select top(10) image,title,description,backlink from Blog  where viewable = 1  order by create\_date desc |

## 

## 2. User Login

### a. Class Diagram

### 

### b. Class Specifications

**User**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | getUsername | Get username |
| 2 | setUsername | Set username |
| 3 | getPassword | Get password |
| 4 | setPassword | Set password |

**UserDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUser | Get username and password from database |
| 02 | checkUsername | Check if the username already exists in the database or not |

**DBContext**

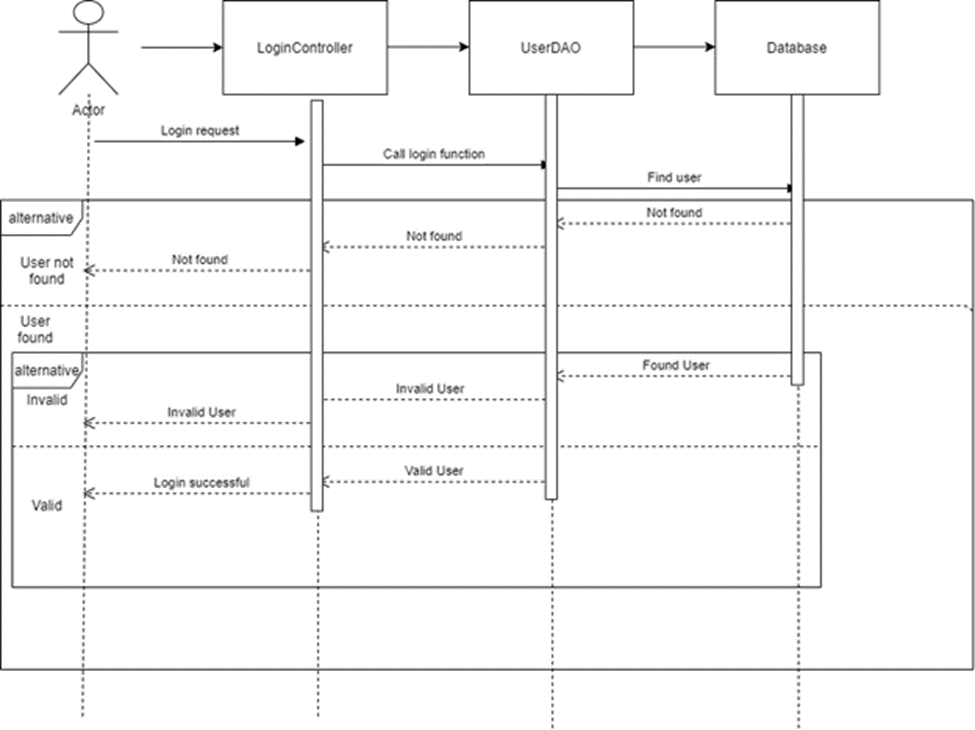
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connect to database |

**UserController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet | Method GET, redirect to login view page |
| 02 | doPost | check the login account, if it's correct, it will redirect to the next page and save cookies, if it's wrong, log back in |

### 

### c. Sequence Diagram(s)



### 

### 

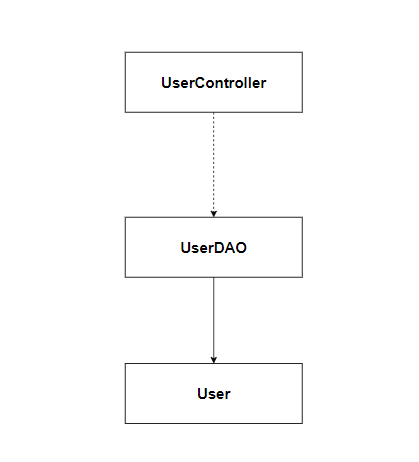
### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUsername() | Select username,password,role from User where username=? and password=? |
| 02 | checkUsername() | select username,password from User where username=? |

## 

## 3. User Profile

### a. Class diagram

****

### b. Class Specifications

**User Class**

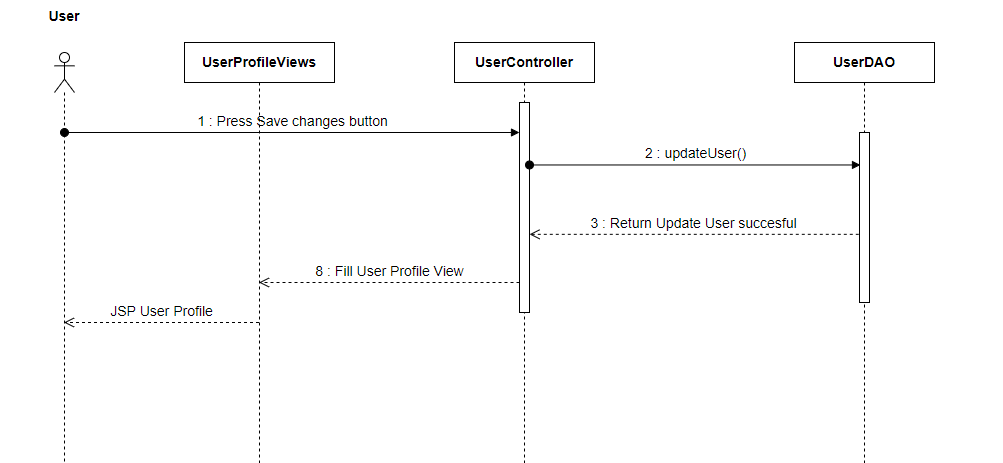
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateUserProfile() | update customer Information |
| 02 | getUserByUserId() | Get an User by an userID |

### c. Sequence Diagram

*Update User Profile*

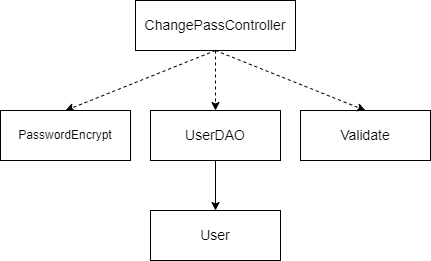
**

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getUserByUserId() | *select u.\*,s.settingStatus from user as u join setting as s on u.accountStatus = s.settingId where u.uid =?* |
| updateUserProfile() | *UPDATE user SET fullname = ?,title = ? ,phone = ?,address = ?,avatar = ? WHERE uid = ?;* |

## 4. Change Password

### a. Class Diagram



### 

### b. Class Specifications

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAccount() | Check the password is exist or not |
| 02 | ChangePassword() | Change the password |

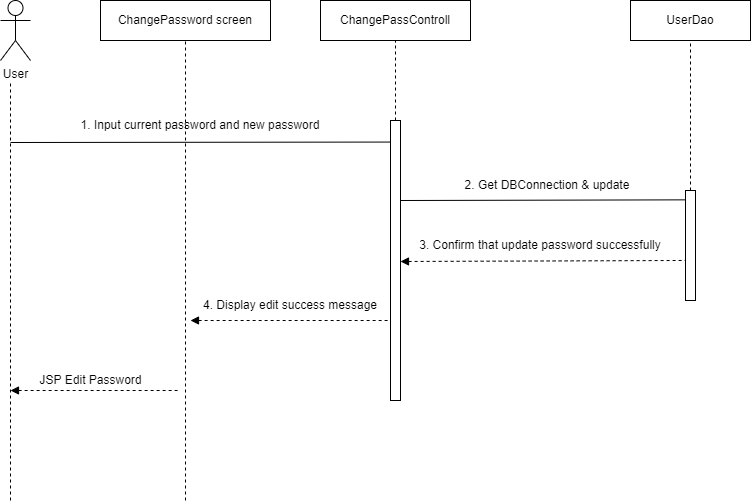
**Validate class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | CheckPassword() | Check the password format |

**PasswordEncrypt class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | generateEncryptedPassword() | Encrypted Password |

### c. Sequence Diagram(s)

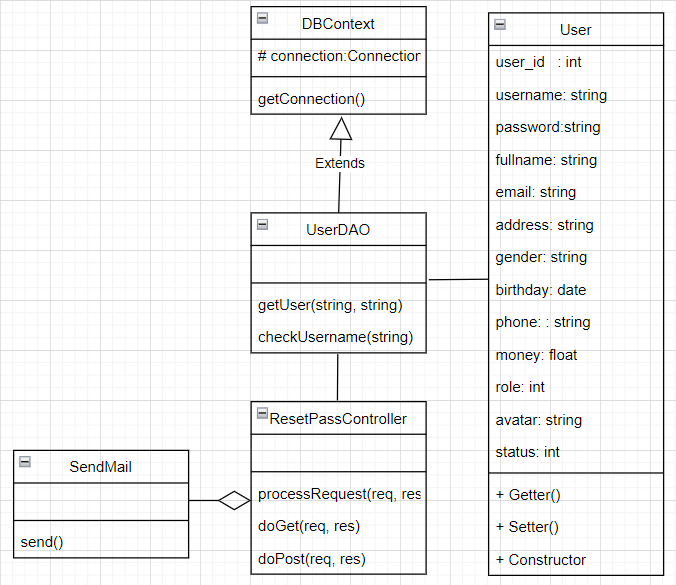
****

### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getAccount() | *SELECT \* FROM user WHERE email = ? and password = ?* |
| ChangePassword() | *UPDATE user SET password = ? WHERE email = ?;* |

## 5. Reset password

### a. Class Diagram



### b. Class Specifications

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

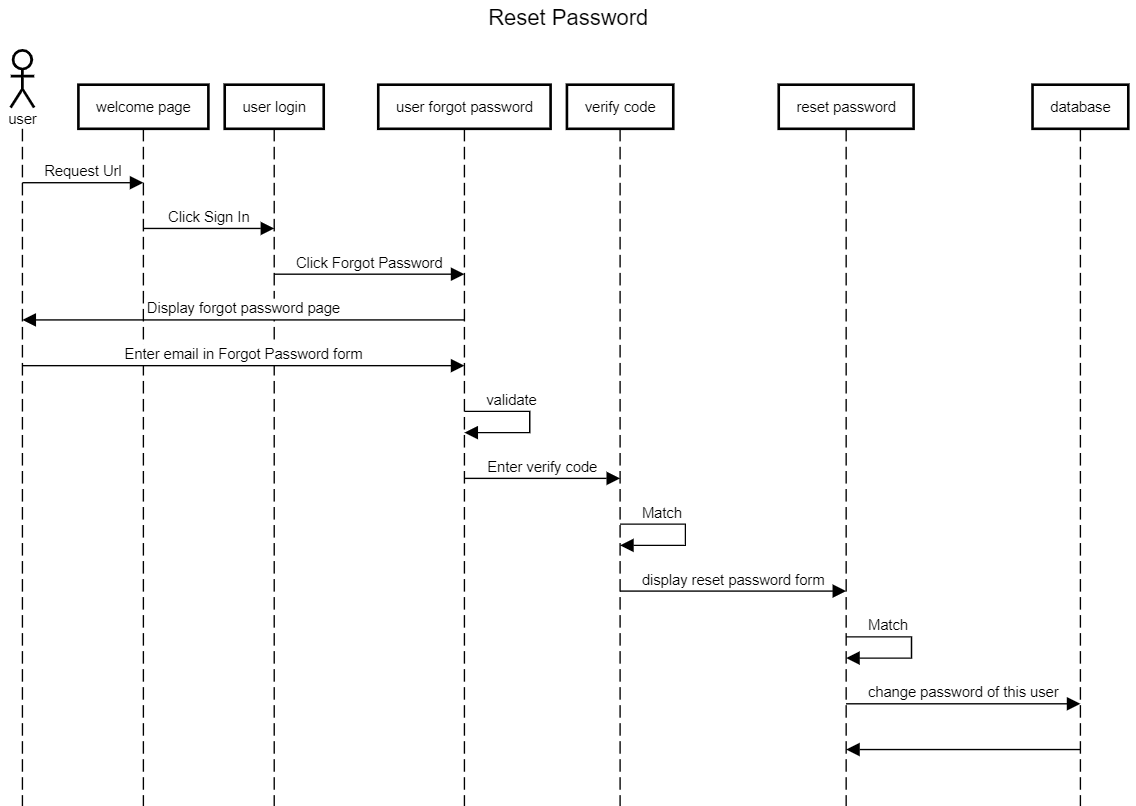
**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | ChangePassword() | Change the password |
| 02 | checkUserExist() | Check the correct Email |

**SendEmail class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | send() | send mail |

### c. Sequence Diagram(s)

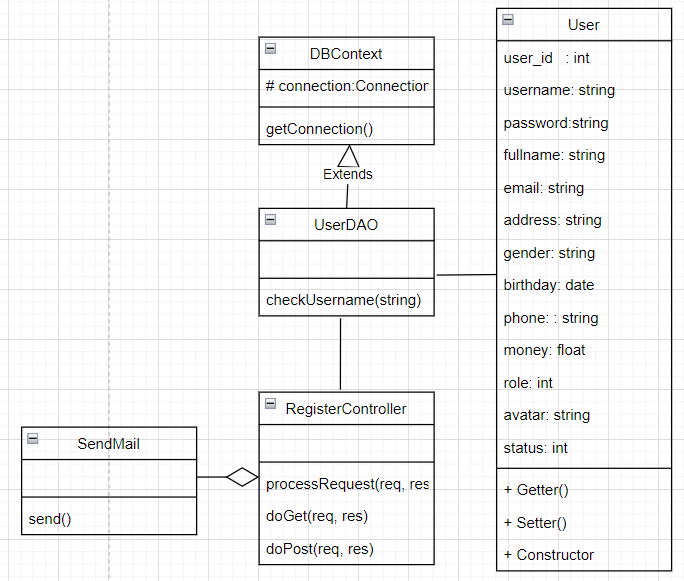


### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| checkUserExist() | *select \* from user where email = ?* |
| ChangePassword() | *UPDATE user SET password = ? WHERE email = ?;* |

## 6. User register

### a. Class Diagram



### b. Class Specifications

**User**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 1 | get, set Username | Get,set username |
| 2 | get, set Password | Get, set password |
| 3 | Get, set Dob | get, set date of birth |
| 4 | Get, set Image | Get, set link image of user |
| 5 | get,set Gender | Get, set gender of user |
| 6 | Get, set Fullname | Get, set fullname of user |
| 7 | Get, set Email | Get, set email address of user |
| 8 | Get, set Phone | Get, set number phone of user |

**UserDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | insertAccount | Insert new user to database |
| 02 | checkUsername | Check if the username already exists in the database or not |

**DBContext**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connect to database |

**SendEmail class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | send() | send mail |

**RegisterController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doGet | Method GET, redirect to sign up view page |
| 02 | doPost | Register a new user, the request is not to have the same username as the previous one |

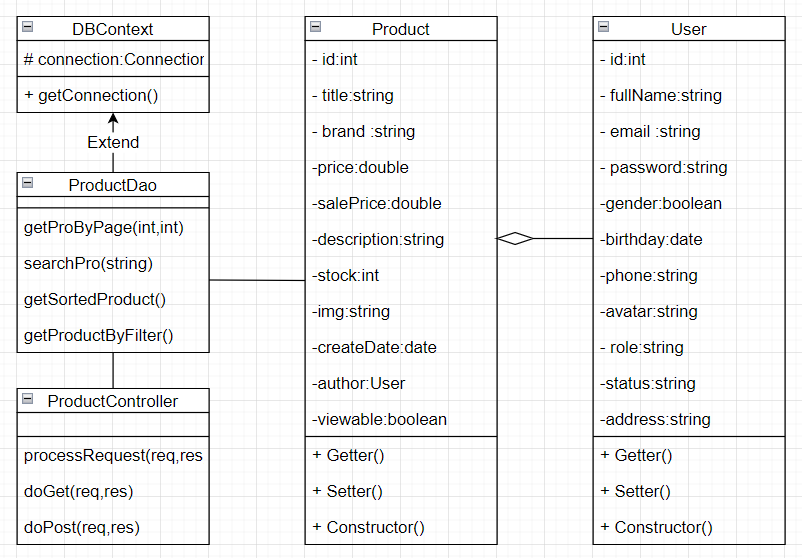
### c. Sequence Diagram(s)

### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Queries** |
| 01 | insertAccount() | insert into User(email,username ,password,image,fullname,gender,dob,phone) values(?,?,?,?,?,?,?,?) |
| 02 | checkUsername() | select username ,password from User where username =? |

## 7. Product List

### a. Class Diagram



### 

### b. Class Specifications

**Product Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Product()* | *Constructor without parameters* |
| *02* | *Product(int pid,String title,,String image, double price, double discount, int productStatus, Category category, Brand brand, String description, int quantity, Date updatedDate, int author)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User (int pid,String fullName,String email, String password, gender boolean, birthday date, phone string, avatar string, role string, status string,address string)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

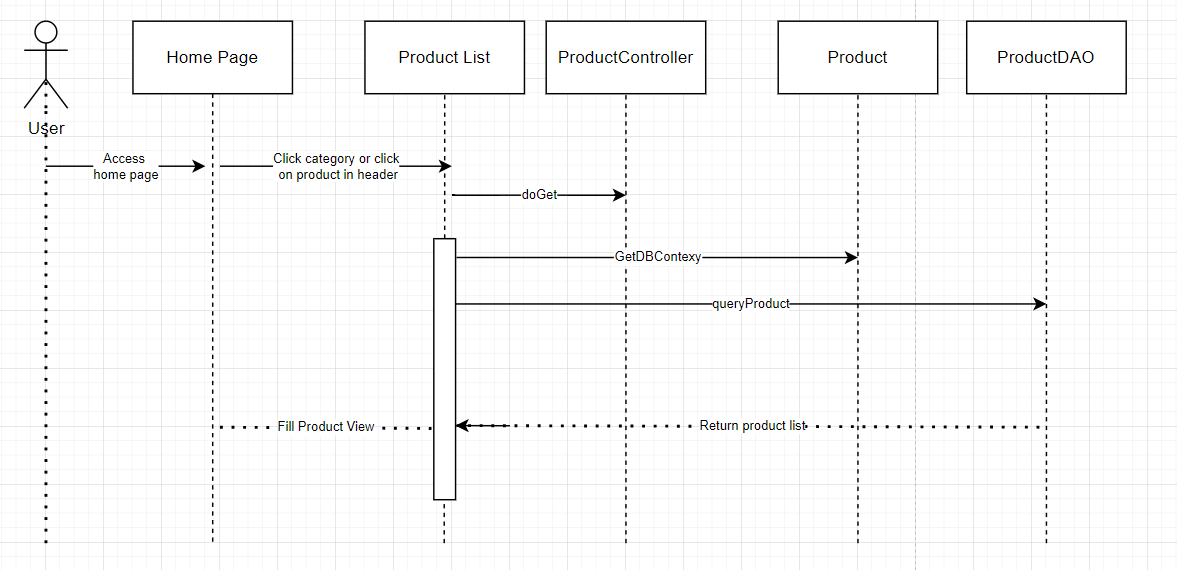
**ProductDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getProByPage(int,int) | get all Products in a specific page |
| 02 | searchPro(string) | Search Product by input of user |
| 03 | getSortedProduct() | get all Products and sort by selected option |
| 04 | getProductByFilter() | get all Products and sort by filter option |

**ProductController**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest(req,res)* | *User for process request of server* |
| *02* | *doGet(req,res)* | *Process GET method* |
| *03* | *doPost(req,res)* | *Process POST method* |

### c. Sequence Diagram(s)



### 

### 

### 

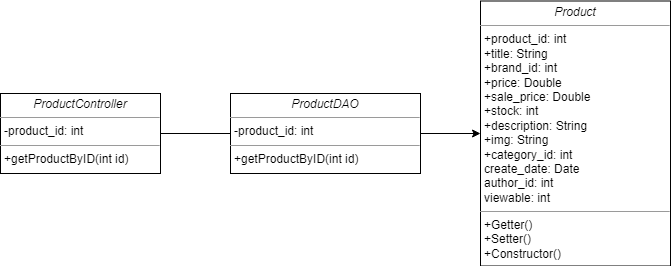
### 

### d. Database queries

|  |  |  |
| --- | --- | --- |
| **No** | **Function** | **SQL Query** |
| 01 | getAllProductByFilter(String brand, String category, String orderBy) | "select a.\*, substring(a.description,0,50) as brief, b.category from (select p.\* , s.[value] as brand from product p , setting as s \n"  + " where s.setting\_id = p.brand\_id and p.brand\_id like ? ) as a,\n"  + " (select p.product\_id, s.[value] as category from product p , setting as s\n"  + " where s.setting\_id = p.category\_id and p.category\_id like ?) as b\n"  + " where a.product\_id = b.product\_id \n" |
| 02 | searchPro(string) | select \* from product where product.title =? |

## 8.Product Details

### a. Class Diagram



### b. Class Specifications

**Product Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Product() | *Constructor without parameters* |
| 02 | *Product(int product\_id, String title,int brand\_id,*  *Double price, Double sale\_price, int stock,String description,String img,int category\_id,Date create\_date,int author\_id,int view able)* | *Constructor with parameters* |
| 03 | *Getters & Setters* | *Update and read value of a variable* |

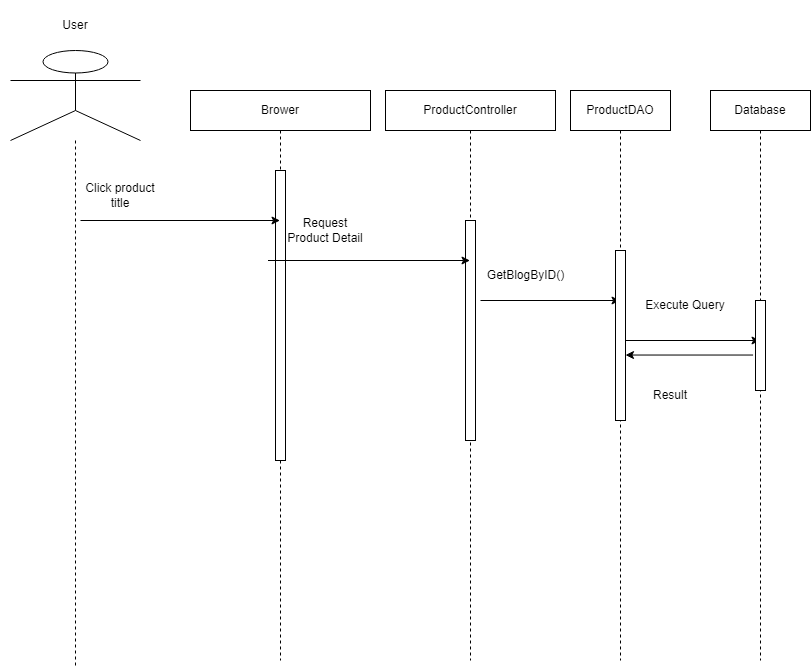
**ProductDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getProductById(String id) | *Enter Product details when users click on products* |

**ProductController Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest(req,res)* | *User for process request of server* |
| *02* | *doGet(req,res)* | *Process GET method* |
| *03* | *doPost(req,res)* | *Process POST method* |

### c. Sequence Diagram(s)

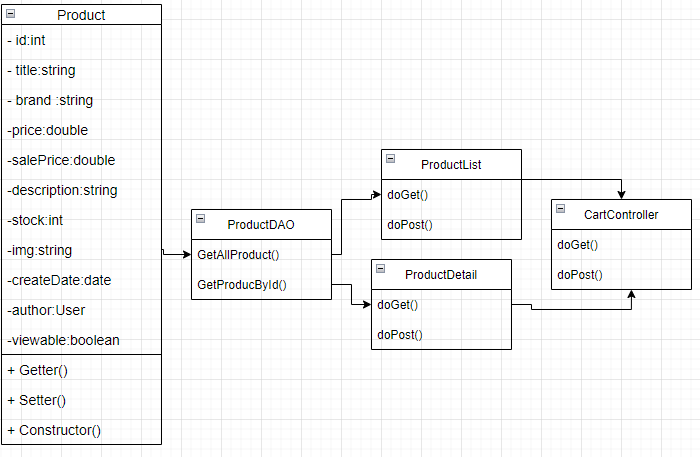


### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getProductById(String id) | *select \* from product where product\_id= ?* |

## 9. Cart Details

### a. Class diagram

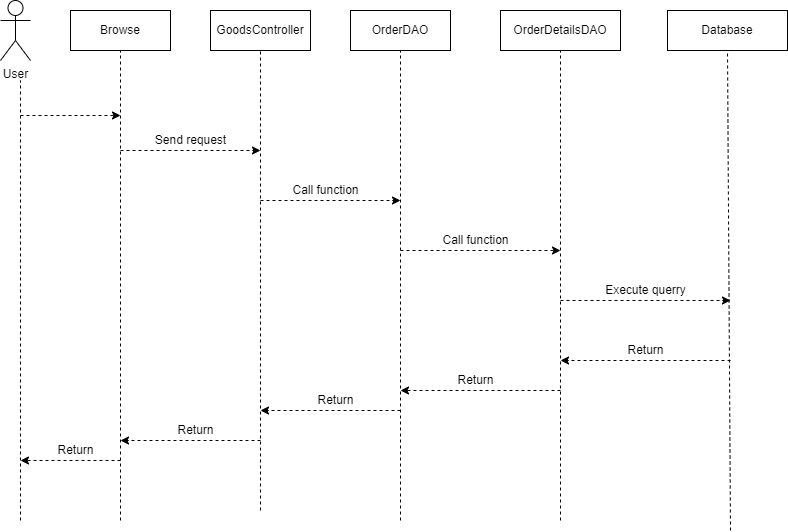


### b. Class Specifications

**Cart Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *cart()* | *Constructor without parameters* |
| *02* | *Cart(List<Items> items)* | *Constructor without parameters* |
| *03* | *Cart(int cartId, List<Items> items, int customerId)* | *Constructor with parameter* |
| *04* | *Getters & Setters* | *Update and read value of a variable* |
| *05* | *getItemsById(int id)* | *get product by id* |
| *06* | *getQuantityById(int id)* | *Get quantity of each product* |
| *07* | *addItem(Items t)* | *add item to cart* |
| *08* | *removeItem(int id)* | *remove an item* |
| *09* | *getTotalMoney()* | *Get total cost of the order* |

### c. Sequence Diagram



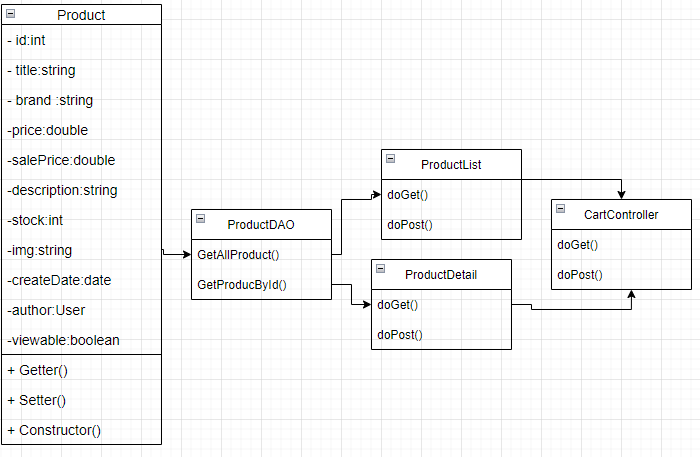
### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getOrderByUser(String status, String from, String to, int id, int index) | *select o.\*, s.orderstatus, od.numpro from (select [order].\*, s.[value] as orderstatus from setting s, [order] where [order].status = s.setting\_id) s, [order] o, (select count(order\_detail\_id) as numpro, order\_id from order\_detail group by order\_id ) as od where o.order\_id = s.order\_id and od.order\_id = o.order\_id and o.status like ? and o.order\_date between ? and ? and o.user\_id = ? order by order\_date desc OFFSET ? ROWS FETCH NEXT 4 ROWS ONLY* |
| deleteOrder(int oid) | *delete from order\_detail where order\_id =? delete from [order] where order\_id= ?* |
| cancelOrder(int orderID) | *UPDATE [dbo].[order] SET [status] = 16 WHERE order\_id = ?* |

## 

## 10. Cart Contact

### a. Class diagram



### b. Class Specifications

**Cart Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *cart()* | *Constructor without parameters* |
| *02* | *Cart(List<Items> items)* | *Constructor without parameters* |
| *03* | *Cart(int cartId, List<Items> items, int customerId)* | *Constructor with parameter* |
| *04* | *Getters & Setters* | *Update and read value of a variable* |
| *05* | *getItemsById(int id)* | *get product by id* |
| *06* | *getQuantityById(int id)* | *Get quantity of each product* |
| *07* | *addItem(Items t)* | *add item to cart* |
| *08* | *removeItem(int id)* | *remove an item* |
| *09* | *getTotalMoney()* | *Get total cost of the order* |

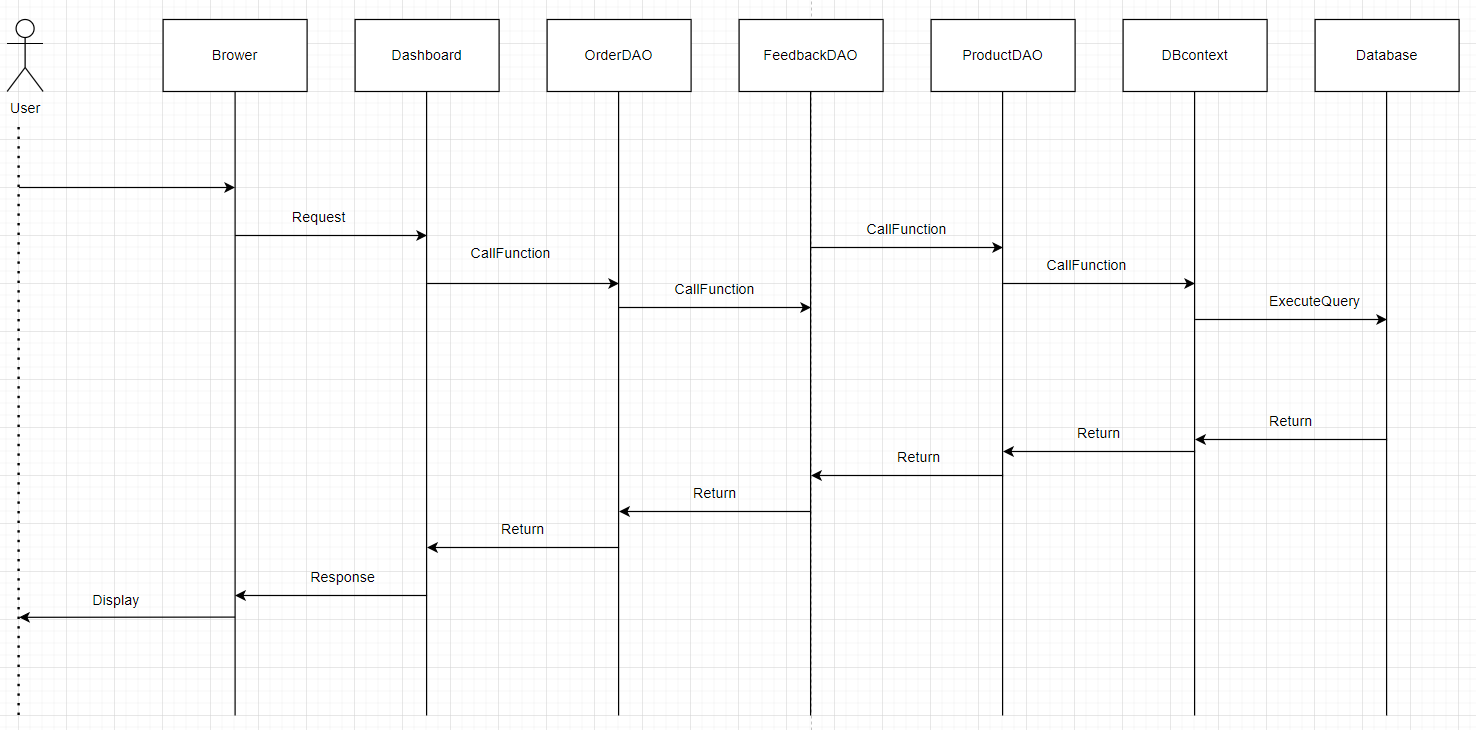
**OrderDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | addOrder() | *add information to order* |
| *02* | getLastOrder() | *get information of the last order* |

**OrderDetailsDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | addNewOrderDetail | *add information to order details* |

### c. Sequence Diagram



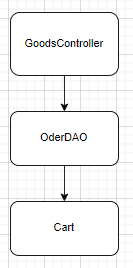
### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getOrderByUser(String status, String from, String to, int id, int index) | *INSERT INTO `onlineshop1`.`orders` (`customerId`, `totalCost`, `salesId`, `orderStatus`, `orderDate`, `lastUpdated`)*  *VALUES (?,?,?,25,CURRENT\_DATE(),CURRENT\_DATE())* |
| getLastOrder() | *select \* from orders ORDER BY orderId DESC LIMIT 1;* |
| addNewOrderDetail | *INSERT INTO `onlineshop1`.`orderdetails` (`orderId`,`productId`,`quantity`,`subCost`,`lastUpdated`)*  *VALUES(?,?,?,?,CURRENT\_DATE());* |

## 

## 11. Cart Completion

### a. Class diagram



### b. Class Specifications

**Cart Class**

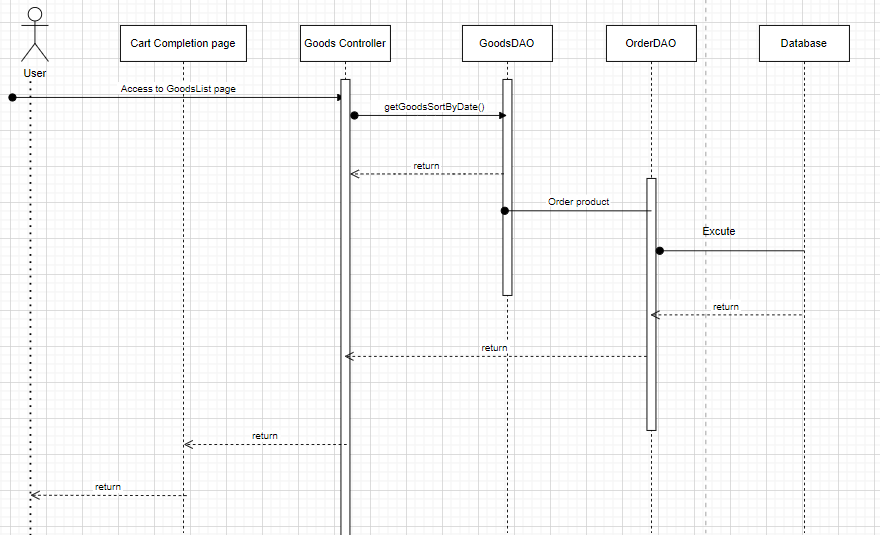
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Cart()* | *Constructor without parameters* |
| *02* | *Cart(List<Items> items)* | *Constructor without parameters* |
| *03* | *Cart(int cartId, List<Items> items, int customerId)* | *Constructor with parameter* |
| *04* | *Getters & Setters* | *Update and read value of a variable* |
| *05* | *getItemsById(int id)* | *get product by id* |
| *06* | *getQuantityById(int id)* | *Get quantity of each product* |
| *07* | *addItem(Items t)* | *add item to cart* |
| *08* | *removeItem(int id)* | *remove an item* |
| *09* | *getTotalMoney()* | *Get total cost of the order* |

**OrderDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getLastOrder()* | *Get info of last order* |

### 

### c. Sequence Diagram



### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| totalRevenue() | *SELECT SUM(totalCost) as 'revenue' FROM onlineshop1.orders*  *WHERE orderStatus = ?* |

## 

## 12.Blogs List

### a.Class diagram

### b.Class specification

**DBContext Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connect to the database |

**Blog Class**

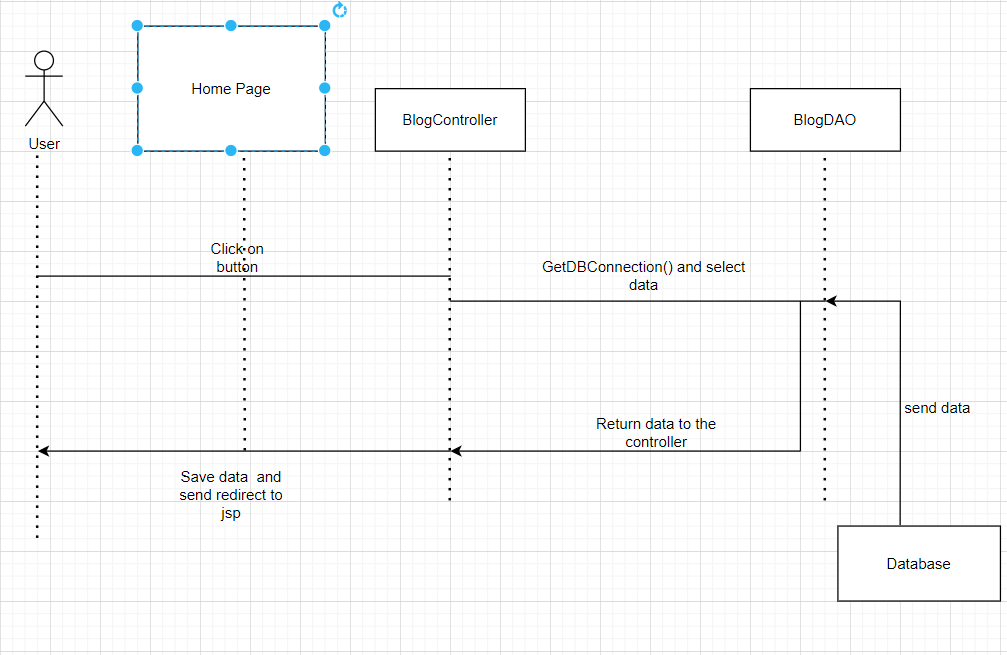
*Class Methods*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Blog() | Constructor without parameters |
| 02 | Blog(int blog\_id, String blog\_title, String description, int author\_id, int comment\_id, int blog\_category, int status,boolean viewable ,String image) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

**BlogDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getPostByViewPaging() | Show some post in a page define with order |
| 02 | getCategory() | Show some category of post on the side |
| 03 | getLatestBlog() | Show some latest blog. |

### c.Sequence diagram

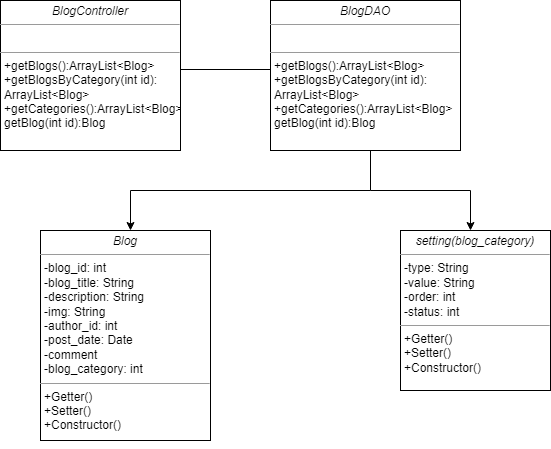
****

### d.Database query

|  |  |
| --- | --- |
| **Function/screen** | **Query** |
| getPostByViewPaging | select image,title,description from Blog where viewable = 1 order by post\_date offset ? rows fetch next ? rows only |
| getLatestBlog() | select top(10) image,title,description from Blog where viewable = 1 order by post\_date desc |
| getCategory() | select blog\_category from Blog |

## 13.Blog Details

### a. Class Diagram

****

### b. Class Specifications

**Blog Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Blog() | *Constructor without parameters* |
| 02 | *Blog(int blog\_id, String blog\_title,String description,*  *String img, int author\_id, Date post\_date,int comment\_id,int blog\_category,int status,int viewable)* | *Constructor with parameters* |
| 03 | *Getters & Setters* | *Update and read value of a variable* |

**Setting Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Setting()* | *Constructor without parameters* |
| *02* | *Setting(String type,String value,int order,int status)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**BlogDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getBlog* | *Get blog by blog\_id* |

**BlogController Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest(req,res)* | *User for process request of server* |
| *02* | *doGet(req,res)* | *Process GET method* |
| *03* | *doPost(req,res)* | *Process POST method* |

### c. Sequence Diagram(s)

### 

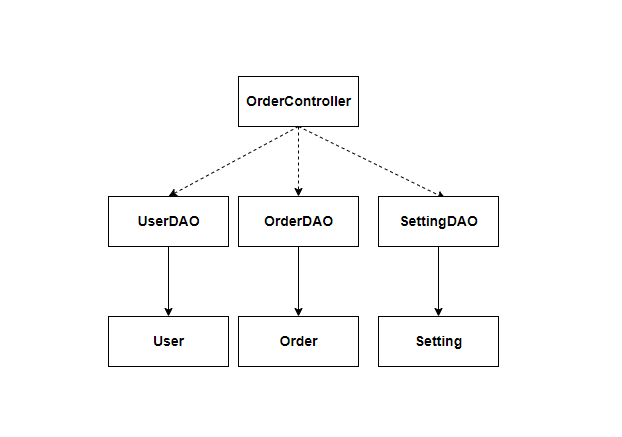
### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getBlogById(String id) | *select \* from blog where blog\_id = ?* |

## 

## 14. Order List

### a. Class diagram



### b. Class Specifications

**Setting Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Setting()* | *Constructor without parameters* |
| *02* | *Setting(int settingId, int settingType, String settingValue, String settingOrder, boolean settingStatus)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**SettingDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getSettingOrderValue() | get all Order Statuses Value from database |

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllUserByRole() | get all User with a specific role in the database |

**Order Class**

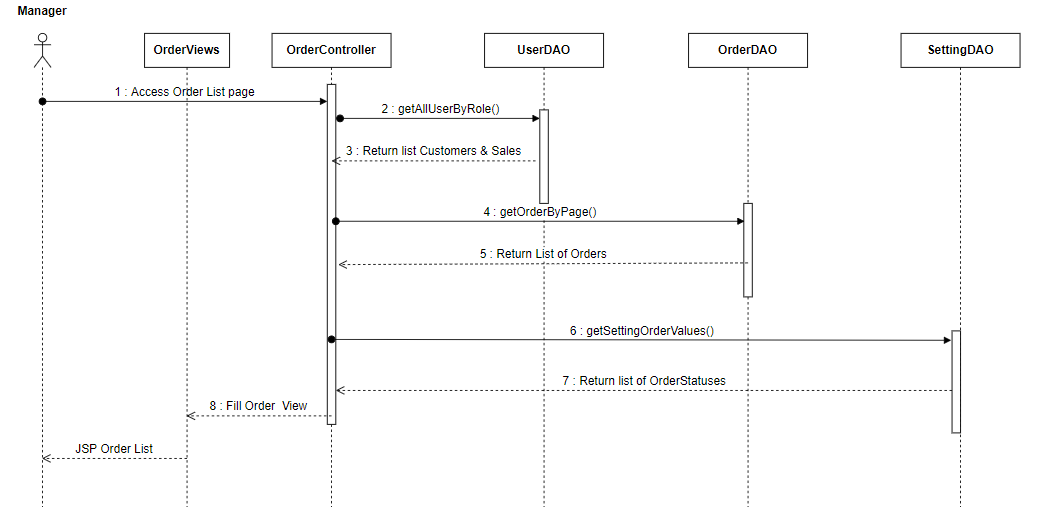
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Order()* | *Constructor without parameters* |
| *02* | *Order(int orderId, int customerId, float totalCost, int salesId, int orderStatus, Date orderDate, String salesNote)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**OrderDAO Class**

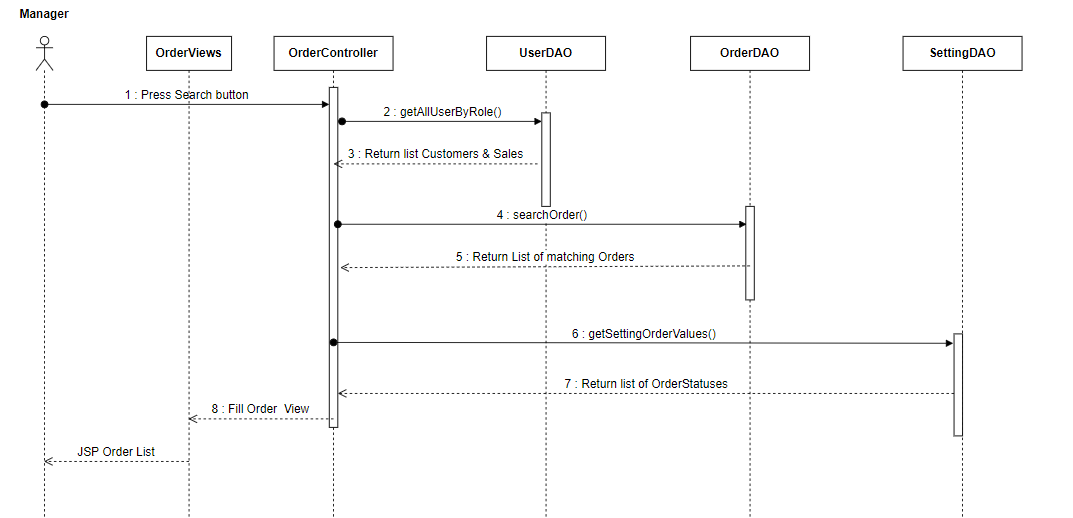
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getOrderByPage() | get all Order by a chosen page in the database |
| 02 | searchOrder() | Search an order by from the database |

### c. Sequence Diagram

*View all Orders*

**

*Search Order*

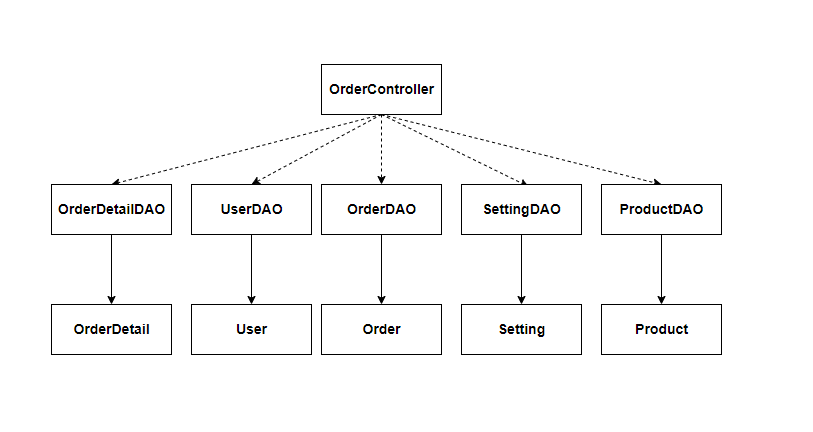
**

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getSettingOrderValue() | *select distinct settingValue from setting where settingType =?* |
| getAllUserByRole() | *select \* from User where role = ?* |
| getOrderByPage() | *select \* from orders LIMIT ?,?* |
| searchOrder() | select o.orderId ,o.orderDate ,o.customerId ,o.totalCost ,o.salesId ,o.orderStatus ,u.fullname from Orders o join User u on u.uid = o.customerId where (o.orderId = ? or u.fullname like ‘%?%’) and o.salesId = ? and o.orderStatus = ? and (o.orderDate between from ‘?’ and to’?’) |

## 15. Order Information

### a. Class diagram

****

### b. Class Specifications

**Setting Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Setting()* | *Constructor without parameters* |
| *02* | *Setting(int settingId, int settingType, String settingValue, String settingOrder, boolean settingStatus)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**SettingDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getSettingOrderValue() | get all Order Statuses Value from database |

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 02 | getUserByUserId() | Get an User by an userID |

**Order Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Order()* | *Constructor without parameters* |
| *02* | *Order(int orderId, int customerId, float totalCost, int salesId, int orderStatus, Date orderDate, String salesNote)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**OrderDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getOrderByOrderId() | get an Order by OrderID |
| 02 | updateOrderSaleInfor() | Update sale information of an order |
| 03 | updateTotalCost() | Update Total Cost value of an order |

**OrderDetail Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *OrderDetail()* | *Constructor without parameters* |
| *02* | *OrderDetail(int orderDetailId, int orderId, int productId, int quantity, float subCost, Date updatedDate)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**OrderDetailDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateProductOrderQuantity() | Update product quantity of an order details |
| 02 | getOrderDetailsByOrderDetailId() | Get an orderDetail by orderDetailId |
| 03 | deleteOrderDetail() | Delete an orderDetail from the database |
| 04 | addNewOrderDetail() | Add new Order Detail |

**Product Class**

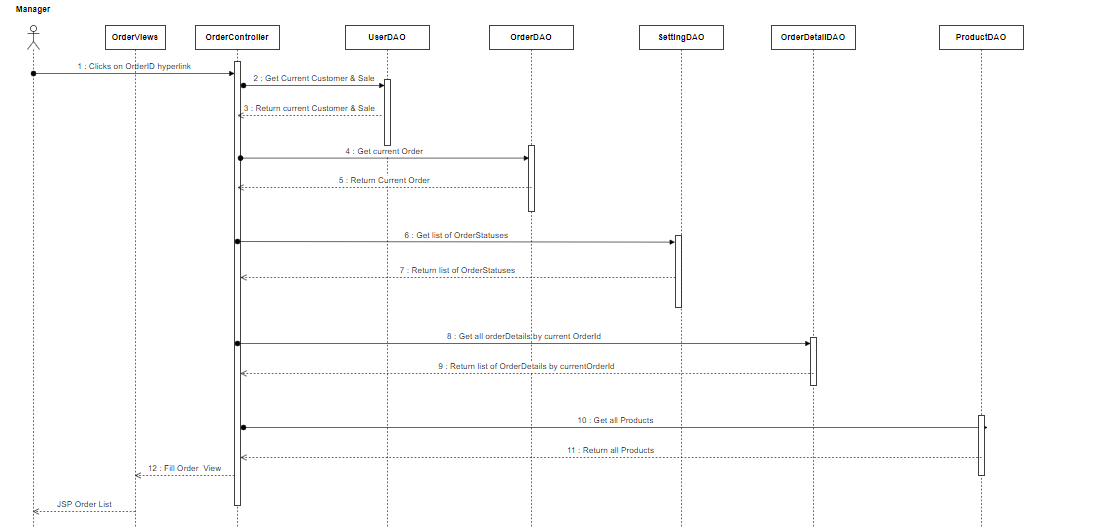
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Product()* | *Constructor without parameters* |
| *02* | *Product(int pid, String thumbnail, String title, double lprice, double sprice, String featured, int productStatus, int categoryID, String breif, int quantity, Date updatedDate, int salesId)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**ProductDAO Class**

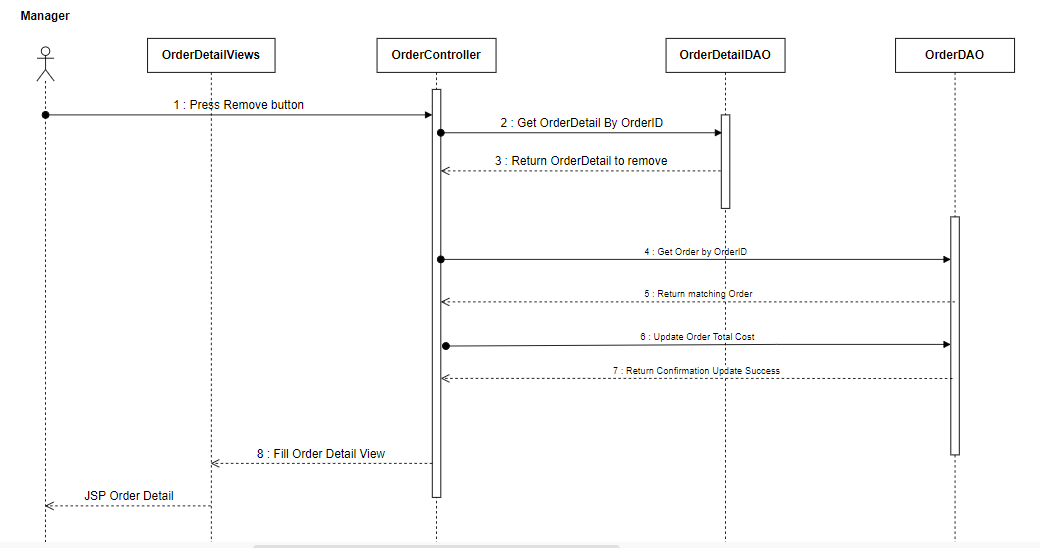
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllProducts() | get all Product from the database |
| 02 | getProductById() | Get a specific product by productId |
| 03 | updateProductQuantity() | Update product quantity in the Product table |

### c. Sequence Diagram

*View an Order Information*

**

*Cancel a Product from an Order*

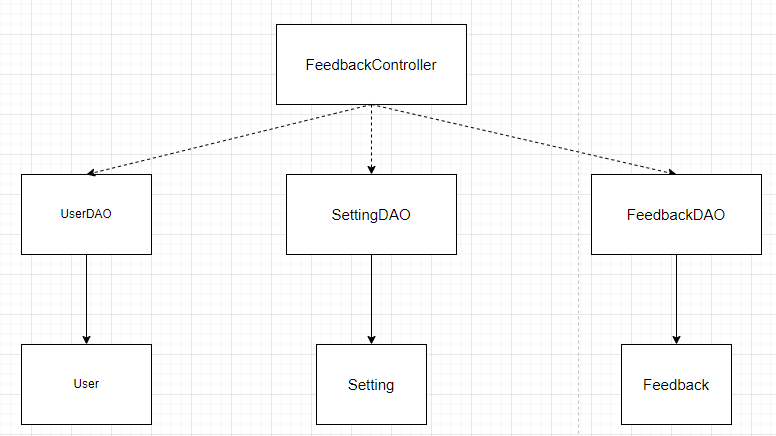
**

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getSettingOrderValue() | *select distinct settingValue from setting where settingType =?* |
| getUserByUserId() | *select u.\*,s.settingStatus from user as u join setting as s on u.accountStatus = s.settingId where u.uid =?* |
| getOrderByOrderId() | *select \* from orders where orderId = ?* |
| updateTotalCost() | *update orders SET totalCost = ? WHERE orderId = ? ;* |
| updateProductOrderQuantity() | *UPDATE orderdetails SET quantity = ? ,subCost = ?, lastUpdated = CURRENT\_DATE() WHERE orderDetailId = ?;* |
| getOrderDetailsByOrderDetailId() | *select \* from orderdetails where orderDetailId =* |
| deleteOrderDetail() | *DELETE FROM orderdetails WHERE orderDetailId = ? ;* |
| getAllProducts() | *select \* from product* |
| getProductById() | *Select \* from Product where productId = ?* |
| updateProductQuantity() | *UPDATE product SET quantity = ? WHERE productId = ?;* |

## 16. Feedback

### a. Class diagram



### b. Class Specifications

**Setting Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Setting()* | *Constructor without parameters* |
| *02* | *Setting(int settingId, int settingType, String settingValue, String settingOrder, boolean settingStatus)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**SettingDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getSettingOrderValue() | get all Order Statuses Value from database |

***User Class***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**UserDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 02 | getUserByUserId() | Get an User by an userID |

**OrderDetail Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *OrderDetail()* | *Constructor without parameters* |
| *02* | *OrderDetail(int orderDetailId, int orderId, int productId, int quantity, float subCost, Date updatedDate)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**OrderDetailDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | updateProductOrderQuantity() | Update product quantity of an order details |
| 02 | getOrderDetailsByOrderDetailId() | Get an orderDetail by orderDetailId |

**Feedback Class**

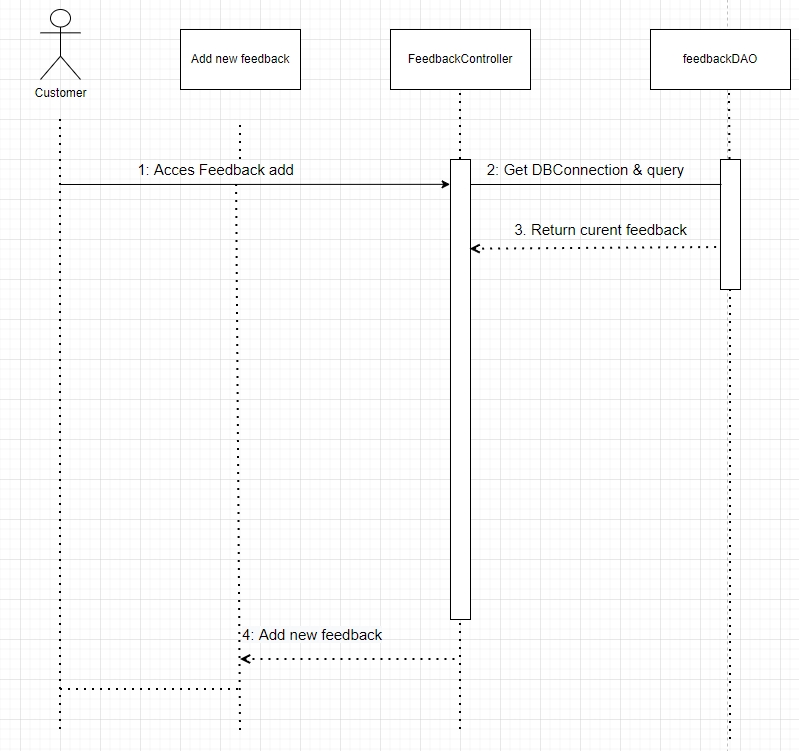
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Feedback()* | *Constructor without parameters* |
| *02* | *Feedback(int feedbackId, int customerId, int ratedStart, int productID, String image, int feedbackStatus, String feedbackContent, Date updatedDate, String title, String note)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**FeedbackDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | addNewFeedback() | add new feedback |

### c. Sequence Diagram

*Add new feedback*

**

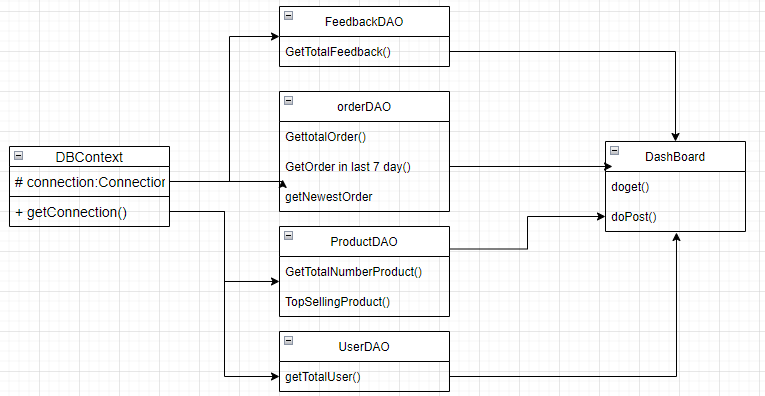
### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getSettingOrderValue() | *select distinct settingValue from setting where settingType =?* |
| getOrderByOrderId() | *select \* from orders where orderId = ?* |
| getOrderDetailsByOrderDetailId() | *select \* from orderdetails where orderDetailId =* |
| addNewFeedback() | *INSERT INTO feedback (customerId,ratedStar,productId,image,feedbackStatus,feedbackContent,updatedDate)\n"*  *+ "VALUES (?,?,?,?,?,?,'2022-03-15');* |

## 

## 17. DashBoard

### a. Class diagram



### b. Class Specifications

**OrderDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getTotalOrder() | *get total number order* |
| *02* | getLastOrderin7day() | *get number of the last 7 day order* |
| *03* | getNewestOrder() | *get information of 10 newest order* |

**ProductDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getTotalProduct | *get total number product* |
| *02* | topsellingproduct | *GET TOP 5 TOP SELLING* |

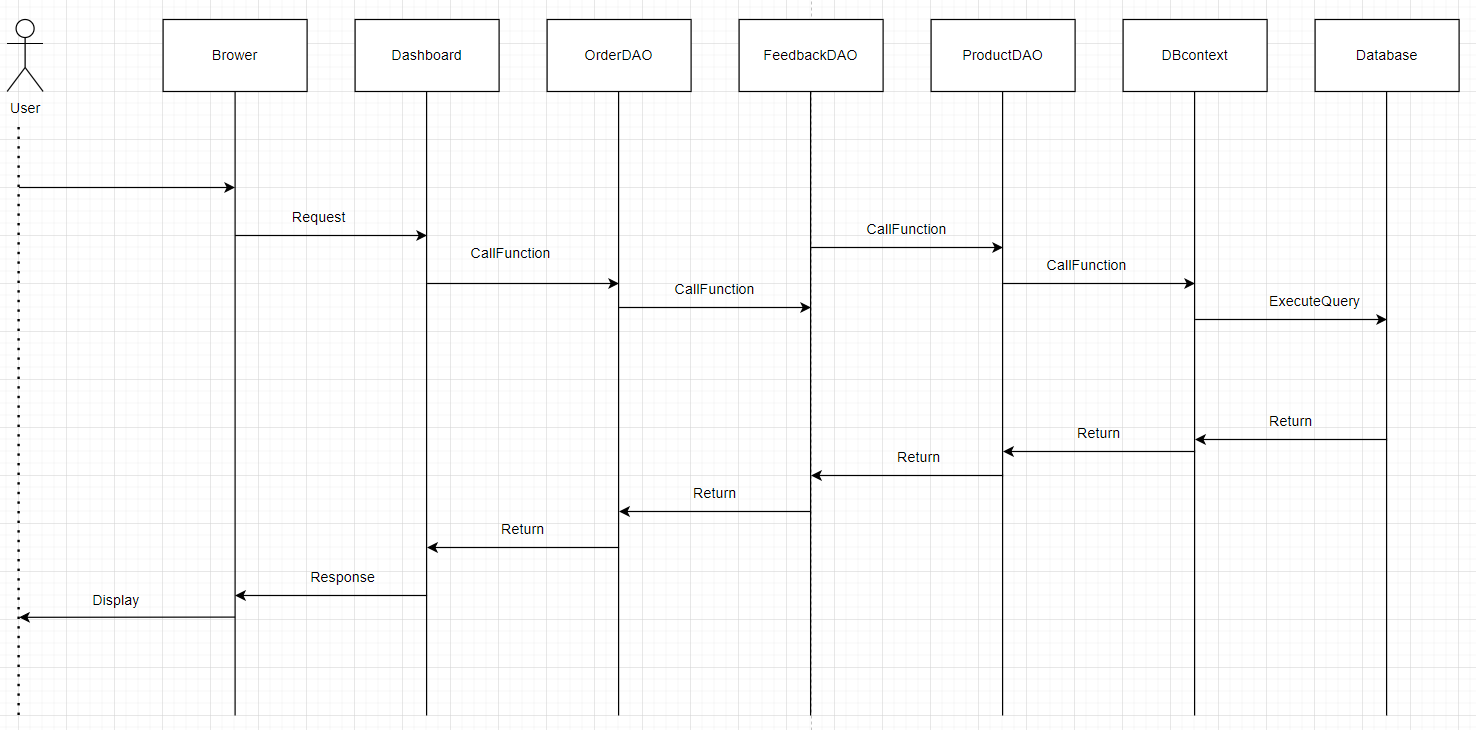
**FeedbackDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getTotalFeedback | *get total number Feedback* |

**UserDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getTotalUser | *get total number User* |

### c. Sequence Diagram

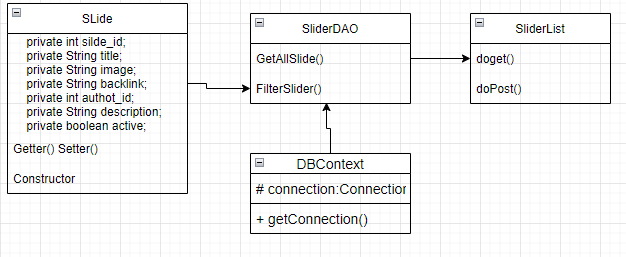


### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getTotalNumberPoduct() | *select count(\*) from product* |
| getTotalNumberUser() | *select count(\*) from [user]* |
| addNewOrderDetail | *INSERT INTO `onlineshop1`.`orderdetails` (`orderId`,`productId`,`quantity`,`subCost`,`lastUpdated`)*  *VALUES(?,?,?,?,CURRENT\_DATE());* |
| getTotalNumberFeedback | *select count(\*) from feedback* |
| getTotalNumberOrder | *select count(\*) from [order]* |
| getTotalProfit | *select sum(total) from [order]* |
| getTotalOrderLast7day() | *select COUNT(\*) as numberorder,order\_date from [order] where order\_date between getdate() - 8 and getdate()-1 group by order\_date order by order\_date asc* |
| getTopSelling() | *select top 5 sum([quantity]) as numberproduct, title from [order\_detail] group by title order by numberproduct desc* |
| getLastestOrder() | *select top 10 [order].\*, s.[value] as orderstatus from setting s, [order] where [order].status = s.setting\_id order by order\_date* |

## 18. Slider List

### a. Class diagram



### b. Class Specifications

**SliderDAO**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *GetAllSlide()* | *Get total number of slidee* |
| *02* | *switchStatus(int sid, int sstatus)* | *turn off or on status of slide* |
| *03* | *FilterSlider()* | *Get all slide by filter* |
| *04* | *updateSlide(String title, String img, String backlink, int authorId, String descrip, int active, int slideid)* | *update slide* |

**Slider**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *private int silde\_id;*  *private String title;*  *private String image;*  *private String backlink;*  *private int authot\_id;*  *private String description;*  *private boolean active;* | *Atrtribute* |
| *02* | *getter()* | *get value of attribute* |
| *03* | *setter()* | *set value of attribute* |
| *04* | *Slide()* | *Constructor* |

**DBCOntext**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | DBContext | *return connection* |

### c. Sequence Diagram

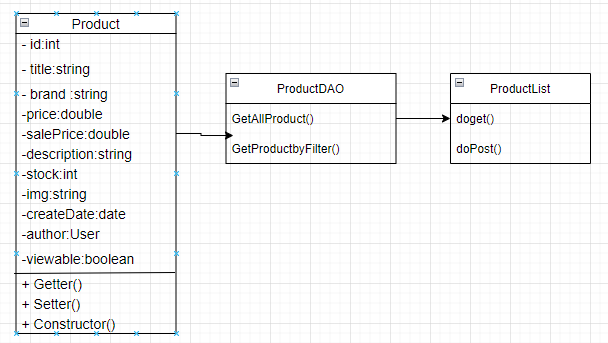
### 

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getNumberSlideWithCondition() | *select COUNT(\*) from slide where active like ? and title like ? and backlink like ?* |
| updateSlide(String title, String img, String backlink, int authorId, String descrip, int active, int slideid) | *update slide set [title]=?, [img] = ? ,[backlink] = ? ,[author\_id] = ? ,[description] = ? ,[active] = ? where [slide\_id] = ?* |
| switchStatus(int sid, int sstatus) | *update slide set [active] = ? where [slide\_id] = ?* |
| deleteSlide(int sid) | *delete from slide where slide\_id = ?* |

## 19. Product List

### a. Class diagram



### b. Class Specifications

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User (int pid,String fullName,String email, String password, gender boolean, birthday date, phone string, avatar string, role string, status string,address string)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**ProductDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllProductForMarketing(String brand, String category, String status, String title, String brief, int index) | Getproduct by fileter |
| 02 | updateProduct(String title, int brandId, double price, double salePrice, int stock, String description, String img, int cateId, int pid, String createDate, int active) | Edit product |
| 03 | addNewProduct() | Add new product |
| 04 | deleteProduct(int pid) | delete from product |

**ProductList**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *processRequest(req,res)* | *User for process request of server* |
| *02* | *doGet(req,res)* | *Process GET method* |
| *03* | *doPost(req,res)* | *Process POST method* |

### c. Sequence Diagram

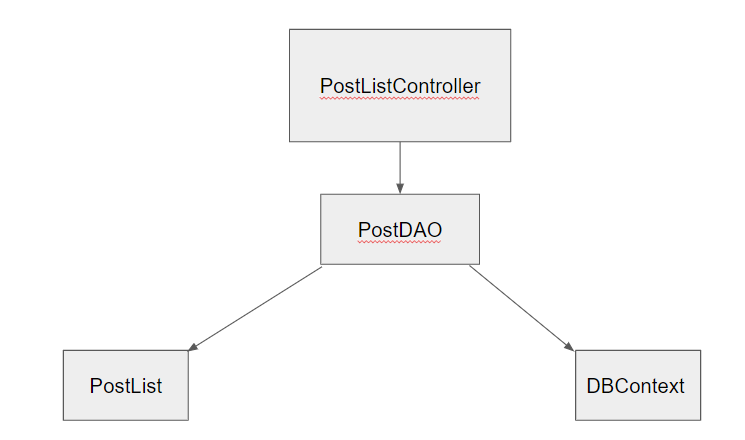
### 

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getAllProductForMarketing(String brand, String category, String status, String title, String brief, int index) | select a.\*, b.category from (select p.\* , s.[value] as brand, substring([description],0,60) as brief from product p , setting as s where s.setting\_id = p.brand\_id and p.brand\_id like ? ) as a, \n" (select p.product\_id, s.[value] as category from product p , setting as s where s.setting\_id = p.category\_id and p.category\_id like ?) as b where a.product\_id = b.product\_id and a.title like ? and a.brief like ? and a.active like ? order by a.product\_id asc OFFSET ? ROWS FETCH NEXT 15 ROWS ONLY |
| updateProduct(String title, int brandId, double price, double salePrice, int stock, String description, String img, int cateId, int pid, String createDate, int active) | update product set [title] = ? ,[brand\_id] = ? ,[price] = ? ,[sale\_price] = ? ,[stock] = ? ,[description] = ? ,[img]=? , [category\_id] = ? ,[create\_date] = ? ,[active] = ? where product\_id = ? |
| addNewProduct() | insert into product ([title],[brand\_id] ,[price] ,[sale\_price] ,[stock],[description],[img] ,[category\_id],[create\_date],[author\_id],[active]) values (?,?,?,?,?,?,?,?,GETDATE(),?,?) |
| deleteProduct(int pid) | delete from product where product\_id = ? |

## 20. Post List

### a. Class Diagram



### b. Class Specifications

**PostList Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | PostList() | Constructor without parameters |
| 02 | PostList(int postId, String thumbnail, String postTitle, String breifInformation, String postContent, String postAuthor, String category, String featured, int satatusPL, Date postDate) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

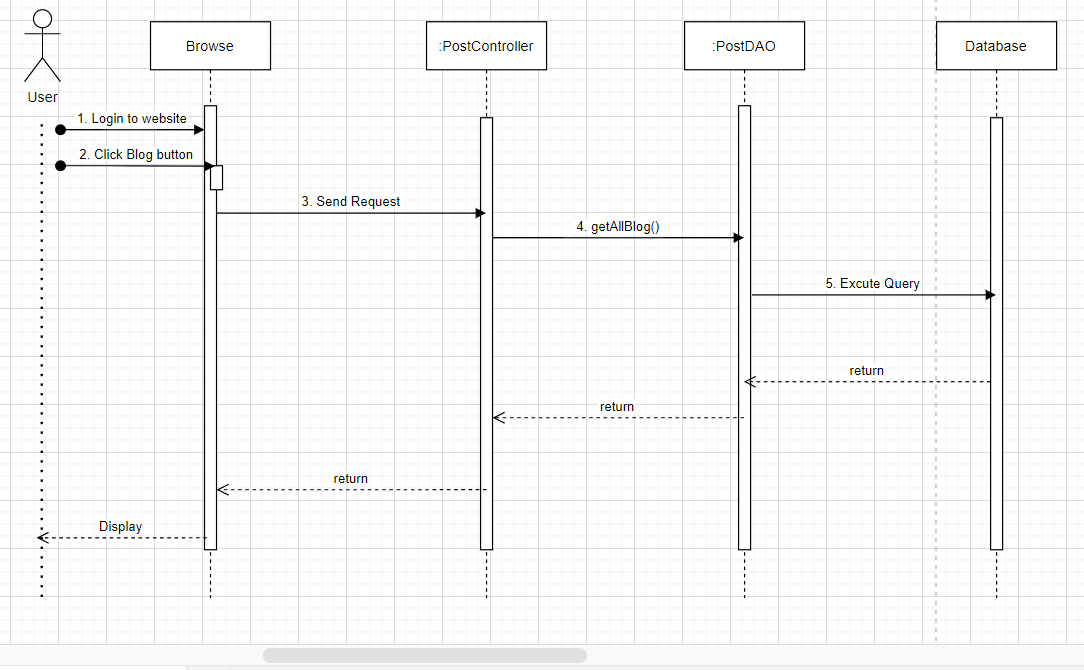
**DBContext Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connects to the database |

**PostDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Delete() | Delete Post |
| 02 | Post() | Add a Blog |
| 03 | Update() | Update a Blog |

### c. Sequence Diagram(s)

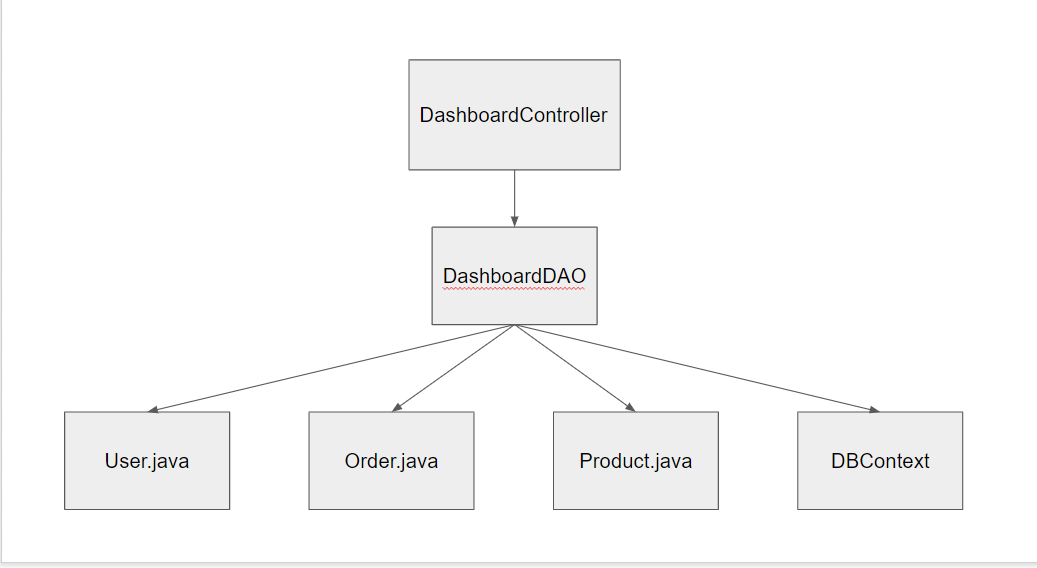


### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| insertPost() | *INSERT INTO `onlineshop3`.`post` (`thumbnail`, `postTitle`, `postContent`, `postAuthor`) \n" +*  *" VALUES (?, ?, ?, ?);* |
| updatePost() | *UPDATE `onlineshop3`.`post` SET `thumbnail` = ?, `postTitle` = ?, `postContent` = ? WHERE (`postId` = ?);* |
| getPost() | *SELECT p.thumbnail, p.postTitle, p. postContent, p.postAuthor FROM onlineshop3.post as p* |

## 21. Dashboard

### a. Class diagram



### b. Class Specifications

**Product Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Product()* | *Constructor without parameters* |
| *02* | *Product(int pid, String thumbnail, String title, double lprice, double sprice, String featured, int productStatus, int categoryID, String breif, int quantity, Date updatedDate, int salesId)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**User Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *User()* | *Constructor without parameters* |
| *02* | *User(int uid, String fullname, String title, boolean gender, String phone, String address, int aid, String avatar)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

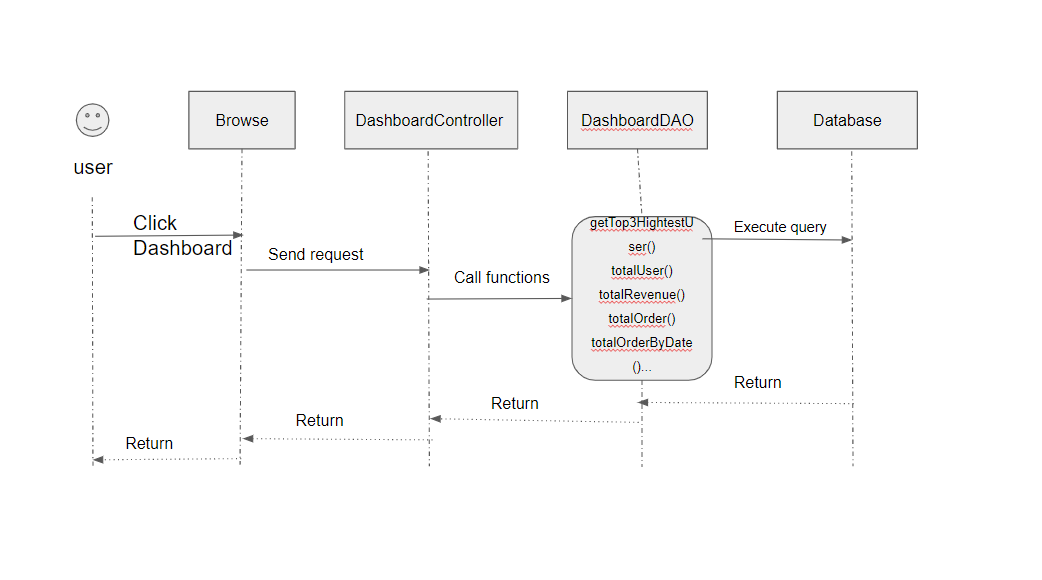
**Order Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *Order()* | *Constructor without parameters* |
| *02* | *Order(int orderId, int customerId, float totalCost, int salesId, int orderStatus, Date orderDate, String salesNote)* | *Constructor with parameter* |
| *03* | *Getters & Setters* | *Update and read value of a variable* |

**DashboardDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | getTop3HightestUser() | *GGet the top 3 customers who buy the most* |
| *02* | totalUser() | *Get total user in system* |
| *03* | totalRevenue() | *Get total Revenue* |
|  | totalOrder() | *Get total order* |
|  | totalOrderByDate() | *get total order by date* |
|  | totalOrderByStatus(status) | *get total order by status* |
|  | totalProductByCategory(categoryId) | *get total product by category* |

### c. Sequence Diagram

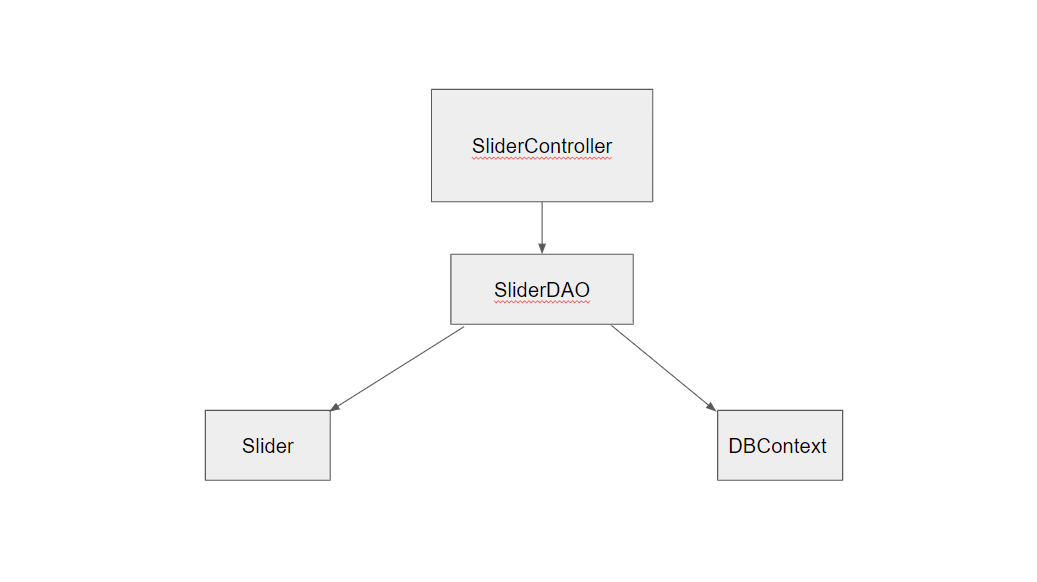
**

### d. Database query

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getTop3HightestUser() | *with x as (SELECT o.customerId as i, SUM(o.totalCost) as r FROM onlineshop1.orders as o*  *GROUP BY o.customerId)*  *SELECT u.fullname, i, r FROM x*  *INNER JOIN onlineshop1.user as u on u.uid = i*  *order by r desc limit 3* |
| totalUser() | *SELECT COUNT(uid) FROM onlineshop1.user* |
| totalRevenue() | *SELECT SUM(totalCost) as 'revenue' FROM onlineshop1.orders WHERE orderStatus = ?* |
| totalOrder() | *SELECT COUNT(orderId) FROM onlineshop1.orders* |
| totalOrderByDate() | *SELECT COUNT(orderId) FROM onlineshop1.orders*  *WHERE orderDate between '?' and '?’* |
| totalOrderByStatus(status) | *SELECT COUNT(orderId) FROM onlineshop1.orders*  *WHERE orderStatus = 20* |
| totalProductByCategory(categoryId) | *SELECT COUNT(productId) FROM onlineshop1.product WHERE categoryId = ?* |

## 22. Slider List

### a. Class Diagram



### b. Class Specifications

**Slider Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Slider() | Constructor without parameters |
| 02 | Slider(int s\_id, String s\_title, String s\_image, String back\_link, int s\_status, String s\_notes) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

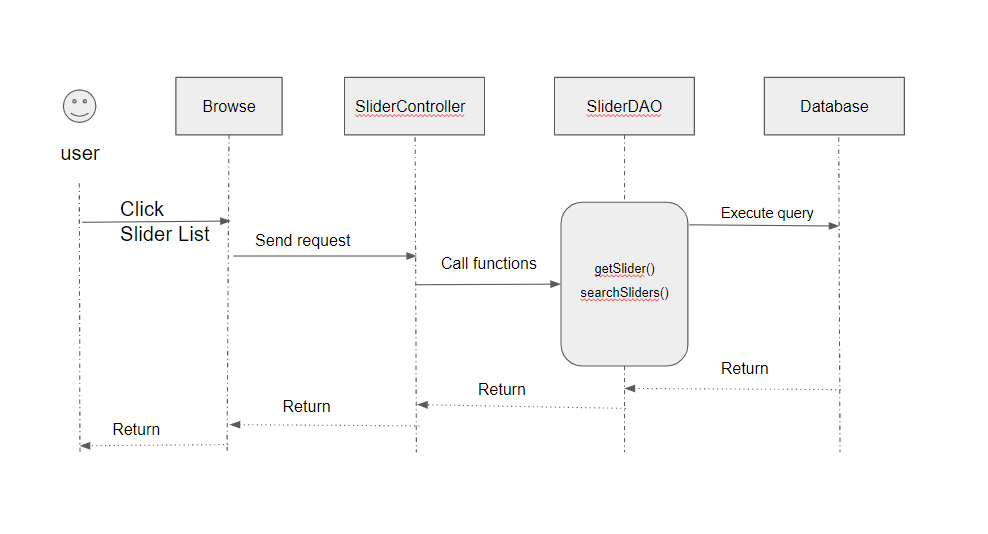
**DBContext Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connects to the database |

**SliderDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 03 | getSliders() | Get all of sliders |
| 04 | searchSliders() | Search sliders |

### c. Sequence Diagram(s)

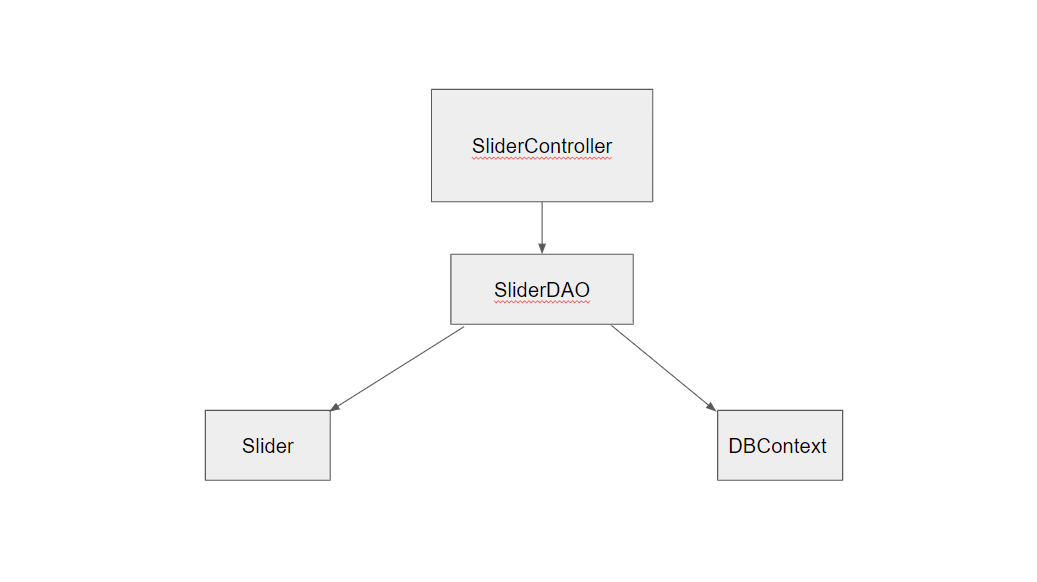


### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getSliders() | *select s.slider\_id, s.s\_title, s.s\_imgage, s.s\_backlink, s.s\_notes*  *from onlineshop1.slider as s* |
| searchSliders() | *select s.slider\_id, s.s\_title, s.s\_imgage, s.s\_backlink, s.s\_notes*  *from onlineshop1.slider as s Where s.s\_title = ?* |

## 23. Slider Detail

### a. Class Diagram



### b. Class Specifications

**Slider Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Slider() | Constructor without parameters |
| 02 | Slider(int s\_id, String s\_title, String s\_image, String back\_link, int s\_status, String s\_notes) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

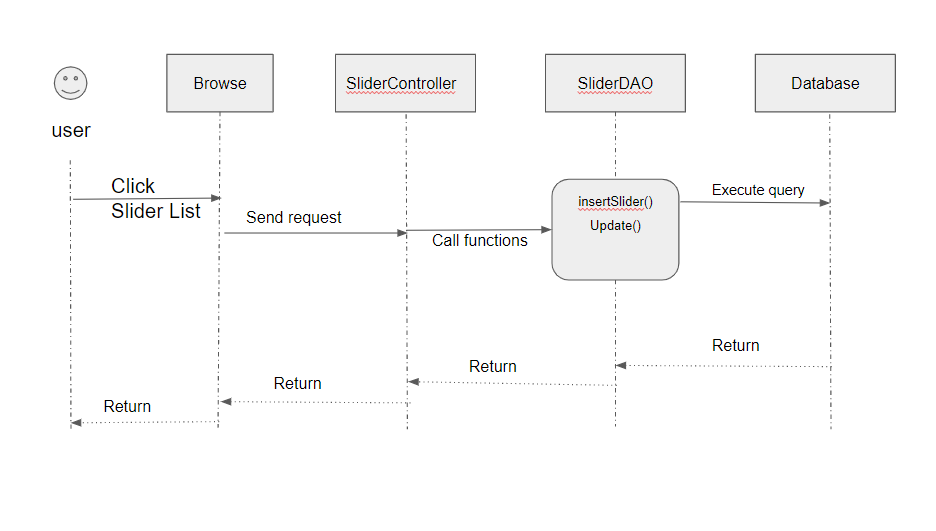
**DBContext Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connects to the database |

**SliderDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | insertSlider() | Add a Slider |
| 02 | Update() | Update a Slider |

### c. Sequence Diagram(s)

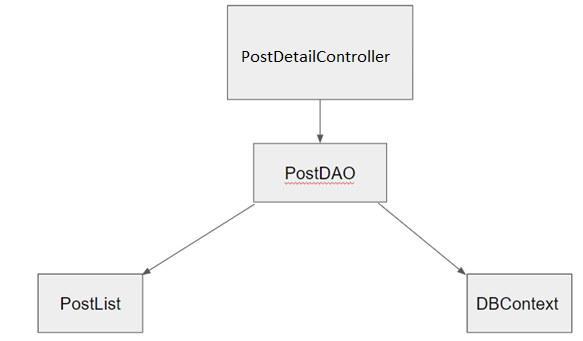


### d. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| insertSlider() | *INSERT INTO onlineshop1.slider (`s\_title`, `s\_imgage`, `s\_backlink`, `s\_notes`) VALUES (?, ?, ?, ?)* |
| update() | *UPDATE `onlineshop1`.`slider` SET `s\_title` = ?, `s\_imgage` = ?, `s\_backlink` = ?, `s\_notes` = ? WHERE (`slider\_id` = ?);* |
|  |  |

## 24. Post detail

### a. Class Diagram

****

### b. Class Specifications

**Post Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | Post() | Constructor without parameters |
| 02 | Post(int postId, String thumbnail, String postTitle, String breifInformation, String postContent, String postAuthor, String category, String featured, int satatusPL, Date postDate) | Constructor with parameters |
| 03 | Getters and setters | Update and read value of a variable |

**DBContext Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | DBContext | Connects to the database |

**PostDAO Class**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getAllMarketingPost | get Post |
| 02 | Post() | Add a Blog |

### c. Database queries

|  |  |
| --- | --- |
| **Function** | **SQL Query** |
| getAllMarketingPost() | *select b.blog\_id,b.blog\_title,b.description,b.img,b.active,s.value,u.fullname from blog b join setting s on s.setting\_id = b.blog\_category join [user] u on u.user\_id = b.author\_id where 1 = 1 and s.value like ? and b.author\_id between ? and ? and b.active between ? and ? and b.blog\_title like ? order by ?* |
| updatePost() | *update blog set description = ? , blog\_title = ?,active = ?,img = ?,blog\_category = ? where blog\_id = ?* |
| AddNewPost() | *INSERT INTO [dbo].[blog] VALUES (? ,? ,? ,? ,? ,? ,? ,? ,?)* |

# III. Database Tables

## 1. user

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | user\_id | int |  | x | x | PK | mã của người dùng, tự động tăng lên khi insert vào |
| 2 | fullname | nvarchar | 50 |  | x |  | Tên đầy đủ của người dùng |
| 2 | active | bit |  |  |  |  | Thể hiện trạng thái của account đang hoạt động hay bị ngưng hoạt động |
| 4 | avatar | ntext |  |  |  |  | ảnh đại diện của người dùng |
| 5 | email | nvarchar | 50 | x | x |  | email của người dùng khi nhập vào |
| 6 | password | nvarchar | 50 |  | x |  | Mật khẩu được mã hóa dưới dạng MD5 |
| 7 | gender | bit |  |  |  |  | giới tính của người dùng |
| 8 | phone | nchar | 11 | x |  |  | số điện mà người dùng nhập vào |
| 9 | address | nvarchar | 100 |  |  |  | địa chỉ mà người dùng nhập vào |
| 10 | role | int |  |  |  |  | quyền của người dùng trên trang web |
| 11 | [key] | nvarchar | 50 |  |  |  | mã xác nhận của người dùng được gửi qua mail khi đăng ký |

## 2. order

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | order\_id | int |  | x | x | PK | mã của sản phẩm, tự động tăng lên khi insert vào |
| 2 | user\_id | int |  |  | x |  | mã của người dùng, tự động tăng lên khi insert vào |
| 3 | fullname | nvarchar |  |  |  |  | Tên đầy đủ của người dùng |
| 4 | address | nvarchar |  |  |  |  | địa chỉ mà người dùng nhập vào |
| 5 | phone | nchar |  |  |  |  | số điện mà người dùng nhập vào |
| 6 | order\_date | date |  |  |  |  | ngày mà người dùng đặt hàng |
| 7 | total | float |  |  |  |  | tổng giá tiền sau khi đặt hàng |
| 8 | payment\_id | int |  |  |  | FK | khoá ngoại id sản phẩm thanh toán |
| 9 | status | int |  |  |  |  | trạng thái giao hàng |
| 10 | active | bit |  |  |  |  | Trạng thái hiển thị |
| 11 | notes | ntext |  |  |  |  | Ghi lại phản hồi của người dùng |

## 3. order\_detail

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | order\_detail\_id | int |  | x | x | PK | khóa chính của bảng order\_detail |
| 2 | order\_id | int |  |  | x |  | mã của sản phẩm, tự động tăng lên khi insert vào |
| 3 | quantity | int |  |  |  |  | số lượng sản phẩm |
| 4 | product\_id | int |  |  |  |  | mã của sản phẩm, tự động tăng lên khi insert vào |
| 5 | author\_id | int |  |  |  |  | mã khóa ngoại của người tạo |
| 6 | title | nvarchar |  |  |  |  | tên của sản phẩm |
| 7 | brand\_id | int |  |  |  |  | mã của brand, tự động tăng lên khi insert vào |
| 8 | price | float |  |  |  |  | giá tiền của mỗi loại sản phẩm |
| 9 | sale\_price | float |  |  |  |  | giá sản phẩm khi đã được giảm giá |
| 10 | stock | int |  |  |  |  | số sản phẩm còn lại trong kho |
| 11 | description | ntext |  |  |  |  | Thông tin mô tả |
| 12 | img | ntext |  |  |  |  | ảnh của sản phẩm |
| 13 | category\_id | int |  |  |  |  | mã khóa của loại sản phẩm |

## 4. cart

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | cart\_id | int |  |  |  | PK | Mã của đơn hàng,tự tăng |
| 2 | user\_id | int |  |  |  | FK | khóa ngoại tới bảng user |
| 3 | total price | float |  |  | x |  | Tổng số tiền của đơn hàng |

## 5. settings.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | setting\_id | int |  | x |  | PK | khóa chính của bảng setting |
| 2 | type | nvarchar | 50 |  | x |  | loại giá trị |
| 3 | value | nvarchar | 50 |  | x |  | giá trị của những hoạt động thương mại |
| 4 | orders | int |  |  | x |  | Các đơn đặt hàng |
| 5 | active | bit |  |  | x |  | Thể hiện trạng thái phân quyền |

## 6. Role

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  |  | x | PK | Id của role trong hệ thống |
| 2 | name | nvarchar | 100 |  |  |  | tên của vai trò trong hệ thống |

## 7.slide

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | slide\_id | int |  | x | x | PK | id của slide |
| 2 | title | nvarchar | 50 |  |  |  | tiêu đề |
| 3 | img | ntext |  |  | x |  | ảnh của slide |
| 4 | backlink | ntext |  |  | x |  | với mỗi slide riêng biệt sẽ đưa 1 trang riêng biệt |
| 5 | author\_id | int |  |  |  |  | id của tác giả |
| 6 | description | ntext |  |  | x |  | Thông tin mô tả |
| 7 | active | bit |  |  | x |  | Thể hiện trạng thái |

## 8. product

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | product\_id | int |  | x | x | PK | mã của sản phẩm, tự động tăng lên khi insert vào |
| 2 | title | nvarchar | 50 |  | x |  | tên của sản phẩm |
| 3 | brand\_id | int |  |  |  | FK | mã phân loại thương hiệu |
| 4 | price | float |  |  | x |  | giá tiền của mỗi loại sản phẩm |
| 5 | sale\_price | float |  |  |  | FK | giá sản phẩm khi đã được giảm giá |
| 6 | stock | int |  |  |  |  | số sản phẩm còn lại trong kho |
| 7 | description | ntext |  |  |  |  | Thông tin mô tả sản phẩm |
| 8 | img | ntext |  |  |  |  | ảnh của sản phẩm |
| 9 | category\_id | int |  |  |  | FK | mã khóa ngoại của loại sản phẩm |
| 10 | create\_date | date |  |  |  | FK | ngày đăng sản phẩm |
| 11 | author\_id | int |  |  | x | FK | mã khóa ngoại của người tạo |
| 12 | active | int |  |  |  |  | Thể hiện trạng thái của sản phẩm đang hoạt động hay bị ngưng hoạt động |

## 9. cart\_detail

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | cart\_detail\_id | int |  | x |  | PK | mã khóa chính của thông tin đơn hàng, tự tăng |
| 2 | cart\_id | int |  | x |  | FK | mã khóa ngoại đến card |
| 3 | product\_id | int |  | x |  | FK | mã khóa ngoại đến sản phẩm |
| 4 | quantity | int |  |  | x |  | số lượng sản phẩm nằm trong đơn hàng |

## 10. blog

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | blog\_id | int |  | x |  | PK | khóa chính của bảng blog |
| 2 | blog\_title | nvarchar | 50 |  |  |  | tiêu đề của blog |
| 3 | description | ntext |  |  |  |  | thông tin mô tả sản phẩm của blog |
| 4 | img | ntext |  |  | x |  | ảnh sản phẩm của blog |
| 5 | author\_id | int |  | x |  | FK | mã khóa ngoại của bảng blog |
| 6 | post\_date | date |  |  | x |  | ngày đăng bài |
| 7 | blog\_category | int |  |  | x |  | loại blog |
| 8 | status | int |  |  | x |  | trạng thái của blog(draft,public) |
| 9 | views | int |  |  |  |  | số lần đọc bài |
| 10 | active | bit |  |  | x |  | Thể hiện trạng thái của blog đang hoạt động hay bị ngưng hoạt động |

## 11. feedback

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | feedback\_id | int |  | x |  | PK | khóa chính |
| 2 | product\_id | int |  | x |  | FK | khóa ngoại đến bảng product |
| 3 | user\_id | int |  | x |  | FK | khóa ngoại đến bảng user |
| 4 | rated | int |  |  | x |  | số sao đánh giá |
| 5 | fb\_content | ntext |  |  | x |  | nội dung đánh giá |
| 6 | img | ntext |  |  | x |  | ảnh đánh giá |
| 7 | feedback\_date | date |  |  | x |  | ngày đăng đánh giá |
| 8 | active | bit |  |  | x |  | Thể hiện trạng thái của feedback đang hoạt động hay bị ngưng hoạt động |

## 12. RoleAuthor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | roleID | int |  | x |  | FK | khóa ngoại nối bảng role |
| 2 | active | int |  |  |  |  | Thể hiện trạng thái của quản lí phân quyền hoạt động hay bị ngưng hoạt động |
| 3 | feature | int |  | x |  | FK | khóa ngoại nối tới bảng feature |

## 

## 13. Feature

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| 1 | id | int |  | x |  | PK | khóa chính của bảng feature |
| 2 | url | ntext |  |  | x |  | url của những trang có trong hệ thống |
| 3 | isPublic | int |  |  | x |  | xác định người dùng có thể xem được khi chưa đăng nhập |