

**ONLINE SHOPPING PLATFORM FOR STUDENTS, TEACHERS AT FPTU**

**Software Design Document**

– HoChiMinh, September 2022 –

**Table of Contents**

[I. Overview](#_heading=h.1fob9te) **9**

[1. Code Packages/Namespaces](#_heading=h.lnxbz9) 9

[2. Database Schema](#_heading=h.qfczxfs6okw8) 10

[**II. Code Designs**](#_heading=h.661e4m1iw2hj) **13**

[1. <LOGIN >](#_heading=h.1ksv4uv) 13

[b. Class Specifications](#_heading=h.e6jdzu7icvjn) 14

[UserController Class](#_heading=h.kgvq6aky2mda) 14

[UserDAO Class](#_heading=h.hp4g4emx9hrb) 14

[DButil Class](#_heading=h.puip72766cei) 14

[c. Sequence Diagram(s)](#_heading=h.al9u58944w90) 14

[d. Database queries](#_heading=h.3o9qajbh4sdu) 15

[2. <SORT>](#_heading=h.5jei5j7ydty4) 15

[a. Class Diagram](#_heading=h.23jbt6mg2kt0) 15

[b. Class Specifications](#_heading=h.7mb2x2mybrho) 16

[Homecontroller Class](#_heading=h.upos0cy1i1ob) 16

[ProductDAO Class](#_heading=h.uzvb01buqvml) 16

[ProductImageDAO Class](#_heading=h.kdt0td9uu6pb) 16

[DButil Class](#_heading=h.sa7wpkvu6q98) 16

[c. Sequence Diagram(s)](#_heading=h.ig5sf8k512vg) 17

[d. Database queries](#_heading=h.eig7yn3t0nux) 17

[3. <LOAD CATEGORY>](#_heading=h.qn102eh111i9) 18

[a. Class Diagram](#_heading=h.45sj7i1ukbsj) 18

[b. Class Specifications](#_heading=h.la5tdtmvcuh2) 18

[Homecontroller Class](#_heading=h.5fu5ltiwxsi1) 18

[CategoryDAO Class](#_heading=h.qzccfifraqu4) 18

[DButil Class](#_heading=h.2yh5pcpp87j) 18

[c. Sequence Diagram(s)](#_heading=h.12c7c8d4v7qx) 19

[d. Database queries](#_heading=h.9ohfixsomfjh) 19

[4. <SEARCH>](#_heading=h.ythbzmijcf08) 19

[a. Class Diagram](#_heading=h.att4245p9z6) 20

[b. Class Specifications](#_heading=h.71wbv0kk9630) 21

[Homecontroller Class](#_heading=h.rrzz2bmgufjl) 21

[GetProductAjax Class](#_heading=h.d01p7x5veyia) 21

[ProductDAO Class](#_heading=h.unzn2gji7d4w) 21

[DButil Class](#_heading=h.bi285624qua1) 21

[c. Sequence Diagram(s)](#_heading=h.l7isk4pgw0sn) 21

[d. Database queries](#_heading=h.ugiuqg9skb03) 22

[5. <PRODUCT DETAIL>](#_heading=h.y9utrbna2f8f) 23

[a. Class Diagram](#_heading=h.y7im5tlmnnfz) 23

[b. Class Specifications](#_heading=h.ov34mqwyt9pb) 24

[HomeController Class](#_heading=h.s5guyx3w75eg) 24

[ReviewDAO Class](#_heading=h.kh4xbdshl7lb) 24

[ReviewImageDAO Class](#_heading=h.ocyqsdcu4ubd) 24

[UserDAO Class](#_heading=h.7txwknw0ki9v) 24

[ProductDAO Class](#_heading=h.5crwgk4fcjcn) 24

[ProductImageDAO Class](#_heading=h.gdq5r8r9jkd6) 25

[DButil Class](#_heading=h.2damo85qt9oi) 25

[c. Sequence Diagram(s)](#_heading=h.r0pmr4vhjbrr) 25

[d. Database queries](#_heading=h.xnpxjy26z0rr) 26

[6. <BUY PRODUCT>](#_heading=h.nhu4r4x0pcnt) 27

[a. Class Diagram](#_heading=h.79esnzyau6e1) 27

[b. Class Specifications](#_heading=h.dvknd6m884wz) 28

[CartController Class](#_heading=h.ltntxx7mp3g9) 28

[OrderDAO Class](#_heading=h.lmtwk86l6z7k) 28

[OrderByShopDAO Class](#_heading=h.v8v339vossyd) 28

[OrderDetailDAO Class](#_heading=h.izm6g8gocqsc) 28

[ProductDAO Class](#_heading=h.zfmzpsatual) 28

[DButil Class](#_heading=h.jgynvby5dbep) 30

[c. Sequence Diagram(s)](#_heading=h.18to5jj5937s) 30

[d. Database queries](#_heading=h.640hdz48ymua) 31

[7. <BANNER>](#_heading=h.dc0b1p32r70r) 31

[a. Class Diagram](#_heading=h.yj2hbojze0n5) 31

[b. Class Specifications](#_heading=h.1zdtv2nzyhum) 32

[HomeController Class](#_heading=h.a095k3och175) 32

[BannerDAO Class](#_heading=h.xf6qnx5yq7if) 32

[DButil Class](#_heading=h.ppsssqvkdwbe) 32

[c. Sequence Diagram(s)](#_heading=h.w4hpozq7ic4r) 33

[d. Database queries](#_heading=h.2tliwz746hqw) 33

[8. <PRODUCT AUTHENTICATION>](#_heading=h.5517n8d9ur4g) 34

[a. Class Diagram](#_heading=h.qbcwi3v6767f) 34

[b. Class Specifications](#_heading=h.i7nbpvant9qr) 34

[Admin Controller Class](#_heading=h.70jafg2i7dmg) 34

[ProductDAO Class](#_heading=h.ge6wwyzayfkn) 35

[ProductImageDAO Class](#_heading=h.ck66sghi0s8j) 35

[c. Sequence Diagram(s)](#_heading=h.6smdut4kyctq) 35

[d. Database queries](#_heading=h.vrw5vcugjbjv) 35

[9. <DELETE PRODUCT ON WEBSITE>](#_heading=h.1uy3matsc8k0) 37

[a. Class Diagram](#_heading=h.y9y0kgsk6xsr) 37

[b. Class Specifications](#_heading=h.nmeggswzpfy5) 37

[Admin Controller Class](#_heading=h.1ylbt1343xmg) 37

[ProductDAO Class](#_heading=h.h0g8nrf1tfkw) 37

[ProductImgaeDAO Class](#_heading=h.ff7voq9nk0ub) 37

[DBUtil Class](#_heading=h.khik3m2erpl6) 38

[c. Sequence Diagram(s)](#_heading=h.ubcmf5xrzcf6) 38

[d. Database queries](#_heading=h.64c30p7vi3kk) 38

[10. <DASHBOARD>](#_heading=h.l1ugxojjvou8) 39

[a. Class Diagram](#_heading=h.iihq6qi9xidw) 39

[b. Class Specifications](#_heading=h.sdo5fq7qxxnm) 39

[DBUtil Class](#_heading=h.2i6kvsxbzk7h) 39

[AdminController Class](#_heading=h.2ol2xxaqm4h3) 39

[OrderDAO Class](#_heading=h.dnd1a7s81lkw) 40

[UserDAO Class](#_heading=h.5313osxo9nab) 40

[ReviewDAO Class](#_heading=h.b323dgcmgvd8) 40

[ProductDAO Class](#_heading=h.q1akuoynlkzd) 40

[c. Sequence Diagram(s)](#_heading=h.1q67p8qlfg3m) 40

[d. Database queries](#_heading=h.bzl9yhwd5ae0) 41

[11. <REVIEW AUTHENTICATION>](#_heading=h.rpblsasosac9) 42

[a. Class Diagram](#_heading=h.rftke87lzq6b) 42

[b. Class Specifications](#_heading=h.64yopft3it9x) 42

[AdminController Class](#_heading=h.jxy1ea2h9z74) 42

[ReviewDAO Class](#_heading=h.9aav7zb18vlj) 42

[DBUtil Class](#_heading=h.ial9txr1vce8) 42

[ReviewImageDAO Class](#_heading=h.aknxal86z1z6) 43

[UserDAO Class](#_heading=h.ovpjmm4rd6zn) 43

[c. Sequence Diagram(s)](#_heading=h.me7s86r2986s) 43

[d. Database queries](#_heading=h.z4ed3n50w9k8) 43

[12. <DELETE REVIEW ON WEBSITE>](#_heading=h.wfb2rej838p) 44

[a. Class Diagram](#_heading=h.17px9qva7fkd) 44

[b. Class Specifications](#_heading=h.5lv9fx9ptu5m) 44

[AdminController Class](#_heading=h.z0hdv6cn3azk) 44

[ReviewDAO Class](#_heading=h.89cs0tlsdy8) 45

[DBUtil Class](#_heading=h.iihhut1gqr8x) 45

[ReviewImageDAO Class](#_heading=h.oj207yqegzvv) 45

[UserDAO Class](#_heading=h.6cyxgc6tg2ft) 45

[c. Sequence Diagram(s)](#_heading=h.vbknxeom34xq) 45

[d. Database queries](#_heading=h.luo779vefmgf) 46

[13. <PROMOTE ROLE>](#_heading=h.a9n7u4i9eudd) 46

[a. Class Diagram](#_heading=h.q2aibi6hggrh) 46

[b. Class Specifications](#_heading=h.7cztcw2l3gjk) 47

[AdminController Class](#_heading=h.9avmkap9xyjn) 47

[UserDAO Class](#_heading=h.y8k0hw3ughfr) 47

[DBUtil Class](#_heading=h.ypsd39te2j14) 47

[c. Sequence Diagram(s)](#_heading=h.m2vf1h3zg7v6) 47

[d. Database queries](#_heading=h.2dnau1k4gvpz) 48

[14. <CREATE NEW PRODUCT>](#_heading=h.br6uwh25owl1) 48

[a. Class Diagram](#_heading=h.mvc33ng5rcii) 48

[b. Class Specifications](#_heading=h.fl0s8canqhzj) 48

[ProductImageDAO Class](#_heading=h.jlhmw090ma9j) 48

[DBUtil Class](#_heading=h.ddv1rc36locy) 48

[c. Sequence Diagram(s)](#_heading=h.ze6suxe2o55h) 49

[d. Database queries](#_heading=h.nu1bb1rx8b0k) 49

[15. <CART>](#_heading=h.pcn1wjbcpgza) 49

[a. Class Diagram](#_heading=h.mxku1aw3vaqb) 50

[b. Class Specifications](#_heading=h.sca4zmcq450y) 50

[CartController Class](#_heading=h.k3elw31fzol8) 50

[c. Sequence Diagram(s)](#_heading=h.w2ygwhh9vupg) 51

[d. Database queries](#_heading=h.7npmflvhlzfk) 51

[16. <USER DETAIL>](#_heading=h.afxteqjb0x4k) 53

[a. Class Diagram](#_heading=h.6agwwx4jbqb1) 53

[b. Class Specifications](#_heading=h.njqpd4r4680x) 53

[UserDAO Class](#_heading=h.mgtfeplcpyh3) 53

[DBuitlClass](#_heading=h.6pjm5lygq1a0) 53

[UserControllerClass](#_heading=h.fcaa69hmrmyl) 54

[c. Sequence Diagram(s)](#_heading=h.7orc1bshncqb) 54

[d. Database queries](#_heading=h.8iv3bfl3a1oi) 54

[17. <SHOP PAGE>](#_heading=h.4nju6wvwtc7y) 55

[a. Class Diagram](#_heading=h.4q02v6opt3te) 55

[b. Class Specifications](#_heading=h.n3ngpdzcod1q) 55

[UserDAO](#_heading=h.xm8yma61y06h) 55

[OrderByShopDAO](#_heading=h.nypaoi32ynix) 55

[ProductDAO Class](#_heading=h.mip4d3pusrpp) 56

[c. Sequence Diagram(s)](#_heading=h.8iy7vrjss7gx) 57

[d. Database queries](#_heading=h.3etwwcqe805o) 57

[18. <BILL DETAIL>](#_heading=h.dll8mkyhl4ae) 59

[a. Class Diagram](#_heading=h.g351fygc7yzg) 59

[b. Class Specifications](#_heading=h.qn8v4x78xwo3) 59

[OrderDAO](#_heading=h.n402nptdbhkk) 59

[OrderByShopDAO](#_heading=h.vlt71ed3dts5) 59

[OrderDetailDAO](#_heading=h.konenjboirot) 60

[ProductDAO Class](#_heading=h.qwiwv1vzmsgu) 60

[c. Sequence Diagram(s)](#_heading=h.gnzelb533osx) 61

[d. Database queries](#_heading=h.rwk1rmi8xx06) 61

[19. <SELLER PAGE>](#_heading=h.fh01pz7d0s2k) 63

[a. Class Diagram](#_heading=h.n2i4nhucmbfx) 63

[b. Class Specifications](#_heading=h.7ktxrhksp9ai) 63

[ProductDAO Class](#_heading=h.4ppq55bfcqxb) 63

[c. Sequence Diagram(s)](#_heading=h.dg7pnkjcjddt) 64

[d. Database queries](#_heading=h.3hzwf7ixr0gz) 64

[20. <PURCHASE HISTORY>](#_heading=h.3q9iqao0iz28) 66

[a. Class Diagram](#_heading=h.k93o732c4jm) 66

[b. Class Specifications](#_heading=h.s50pqbdgv6kp) 67

[OrderDAO](#_heading=h.bg7a2f4puj9k) 67

[OrderByShopDAO](#_heading=h.472vu55avntt) 67

[OrderDetailDAO](#_heading=h.fqde0reg2t9d) 67

[ProductDAO Class](#_heading=h.w166ad5kgvjc) 67

[c. Sequence Diagram(s)](#_heading=h.w0sr5jxdih7s) 68

[d. Database queries](#_heading=h.ug83oq7ltgjo) 68

[21. <PAYMENT>](#_heading=h.79972ys07n40) 69

[a. Class Diagram](#_heading=h.g8x1voo8y6vc) 70

[b. Class Specifications](#_heading=h.tnljl5p05rbr) 70

[GhnApi](#_heading=h.i0gxeg3ddd1s) 70

[OrderDAO](#_heading=h.ipmwczs9aa4d) 70

[OrderByShopDAO](#_heading=h.wbrbiktryiih) 70

[OrderDetailDAO](#_heading=h.grh27m1mafa3) 70

[ProductDAO Class](#_heading=h.4afoe3p8oh48) 70

[c. Sequence Diagram(s)](#_heading=h.xjmbuk32lq0v) 71

[d. Database queries](#_heading=h.ron77rq1vjn7) 71

[22. <REVIEW>](#_heading=h.j33hjg37e2ar) 72

[a. Class Diagram](#_heading=h.ytejur6jnxzb) 73

[b. Class Specifications](#_heading=h.3662sknhjwja) 73

[ReviewDAO](#_heading=h.xkjpmdih2uey) 73

[ReviewImageDAO](#_heading=h.bd0epxffftny) 73

[c. Sequence Diagram(s)](#_heading=h.q86xhuqaifhj) 74

[d. Database queries](#_heading=h.gvh3by6g9r73) 74

[**III. Database Tables**](#_heading=h.kuxd2sq26v6a) **75**

[1. < city >](#_heading=h.3whwml4) 75

[2. < district >](#_heading=h.2bn6wsx) 75

[3. < Ward >](#_heading=h.qsh70q) 75

[4. < Role >](#_heading=h.3as4poj) 75

[5. < user >](#_heading=h.9anizzhfgka2) 75

[6. < category >](#_heading=h.49x2ik5) 76

[7. < delivery\_method >](#_heading=h.2p2csry) 76

[8. < payment\_method >](#_heading=h.147n2zr) 76

[9. < product >](#_heading=h.3o7alnk) 76

[10. < image\_product >](#_heading=h.j1ek9cvlo5k7) 77

[11. < order >](#_heading=h.ihv636) 77

[12. < order\_detail>](#_heading=h.a2quyqlasyav) 77

[13. < order\_by\_shop>](#_heading=h.huhbghm41to9) 78

[14. < review >](#_heading=h.qmdv8rxkem3m) 78

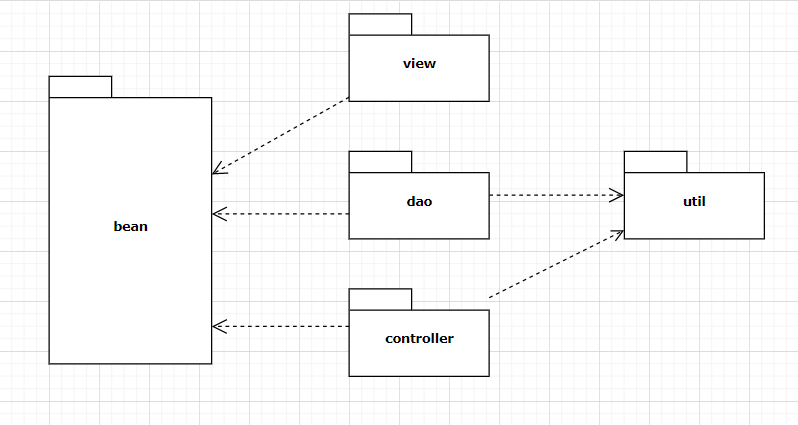
[15. < image\_review >](#_heading=h.2grqrue) 78

[16. < banner >](#_heading=h.vx1227) 78

# I. Overview

## 1. Code Packages/Namespaces

*[Provide the package diagram for each sub-system. The content of this section including the overall package diagram, the explanation, package and class naming conventions in each package. Please see the sample and description table format below – following Java project naming convention]*



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

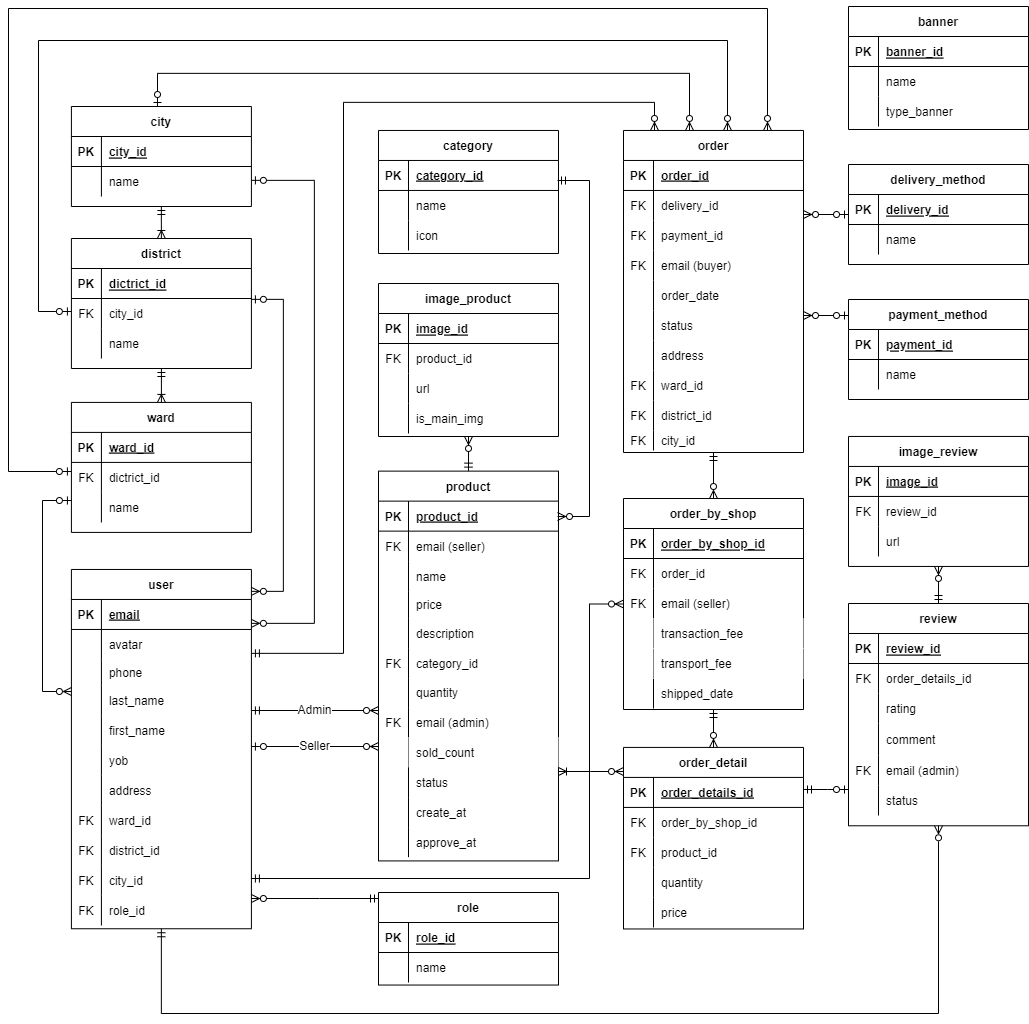
| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | bean | * Entity class * Classname + “DTO” (ex: ProductDTO.java) |
| 02 | view | * Is used for holds view for user and admin * lowerCase (ex:mainpage.jsp) |
| 03 | controller | * Is used for holds web controllers in MVC 2 models * Classname + “controller” (ex: HomeController.java) |
| 04 | dao | * Is used for holds class * Classname + “DAO” (ex: ProductDAO.java) |
| 05 | util | * The util methods should be moved to a helper or utility class or encapsulated within the class that uses them * Camelcase (ex: myToy.java) |

## 

## 

## 2. Database Schema

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



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

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 1 | user | - Primary keys: email  - Foreign keys: ward\_id, district\_id, city\_id, role\_id  - Contains information of all accounts logged into the shop |
| 2 | role | - Primary keys: role\_id  - Assign roles to accounts such as admin, sub-admin, user |
| 3 | product | - Primary keys: product\_id  - Foreign keys: email(seller), category\_id, email(admin)  - Contains information about all products posted by the user for sale |
| 4 | order | - Primary keys: order\_id  - Foreign keys: delivery\_id, payment\_id, email(buyer), ward\_id, district\_id, city\_id  - Contains information about the products ordered by the user (paid/unpaid) |
| 5 | order\_detail | - Primary keys: order\_detail\_id  - Foreign keys: order\_by\_shop\_id, product\_id  - Contains detailed information about the products the user has ordered (including the purchase price) |
| 6 | order\_by\_shop | - Primary keys: order\_by\_shop\_id  - Foreign keys: order\_id, email (seller)  - Order product list in cart by store and store payment information |
| 7 | category | - Primary keys: category\_id  - Contains kinds of products |
| 8 | image\_product | - Primary keys: image\_id  - Foreign keys: product\_id  - Contains the description picture of the product |
| 9 | delivery\_method | - Primary keys: delivery\_id  - Contains shipping methods such as: direct delivery, shipping by ahamove, ... |
| 10 | payment\_method | - Primary keys: payment\_id  - Contains payment types such as: direct payment, paypal payment,... |
| 11 | review | - Primary keys: review\_id  - Foreign keys: email (admin), order\_detail\_id  - Contains buyer feedback on their orders |
| 12 | image\_review | - Primary keys: image\_id  - Foreign keys: review\_id  - Image archive of the review |
| 13 | ward | - Primary keys: ward\_id  - Foreign keys: district\_id  - Contains information about the ward (location) |
| 14 | district | - Primary keys: district\_id  - Foreign keys: city\_id  - Contains information about the district (location) |
| 15 | city | - Primary keys: city\_id  - Contains information about the city (location) |
| 16 | banner | - Primary keys: banner\_id  - Contains slider: url, type (show in website) |

# 

# 

# 

# 

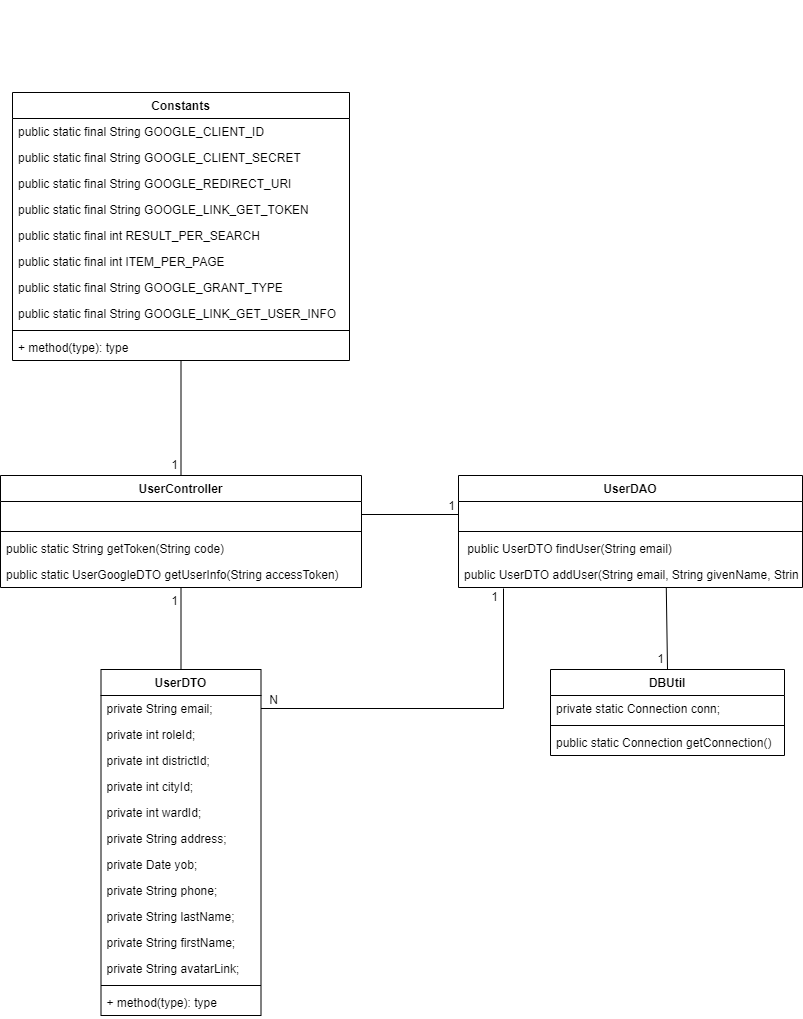
# 

# II. Code Designs

## 1. <LOGIN >

*[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]*

#### UserController Class

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

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static String getToken(String code)* | *return a access token from google server* |
| *02* | *public static UserGoogleDTO getUserInfo(String accessToken)* | *use that access token to get user info from google* |

#### UserDAO Class

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

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* | *return a user if exist in database* |
| *02* | *UserDTO addUser(String email, String givenName, String avatarLink)* | *create a user to database* |

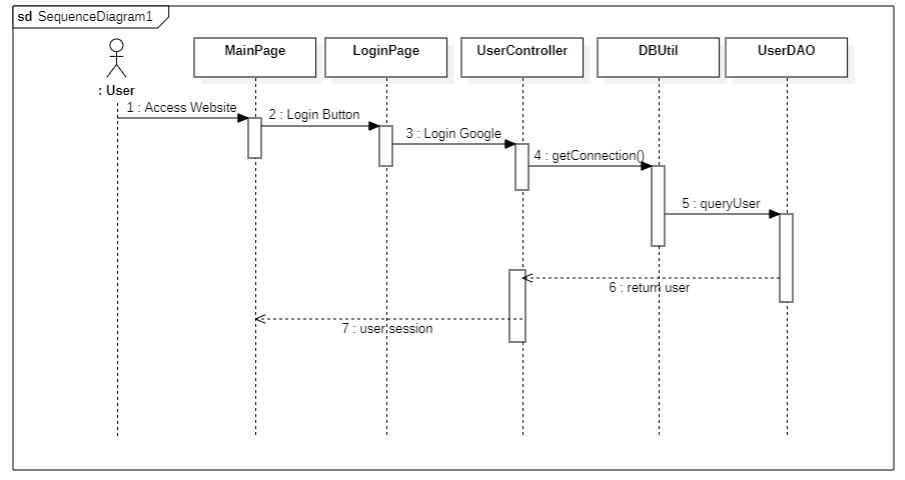
#### 

#### DButil Class

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

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *return a database connection* |

### c. Sequence Diagram(s)



### 

### d. Database queries

UserDAO:

SELECT [email], [avatar], [first\_name], [last\_name], [phone], [yob], [address], [ward\_id], [district\_id], [city\_id] , [role\_id]

FROM [user] WHERE email = ?

INSERT INTO [user] (email, avatar, last\_name, first\_name, role\_id) VALUES (?, ?, ?, ?, 0)

## 2. <SORT>

### a. Class Diagram

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### b. Class Specifications

#### Homecontroller Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *handle request from user* |

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *return total product where cateId = param* |
| *02* | *public String getFilter(int option, boolean trend)* | *return "order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* | *find all product images of the product by productID* |

#### 

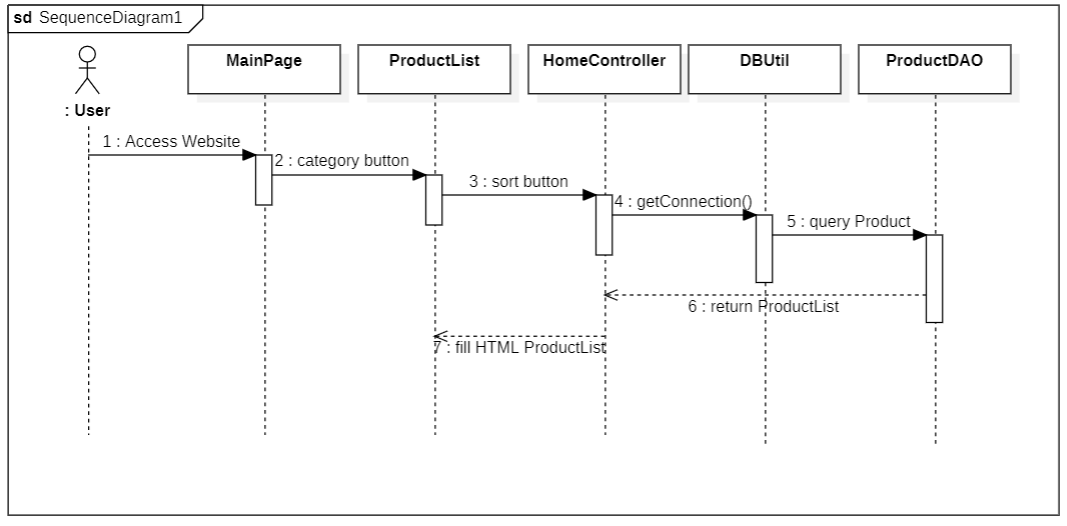
#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *return a database connection* |

### 

### c. Sequence Diagram(s)

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

**

### d. Database queries

ProductDAO:

SELECT COUNT (product\_id) FROMproduct WHERE category\_id = ?

SELECT email\_seller from product where product\_id = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity],

[email\_admin], [status], [create\_at], [approve\_at] , [sold\_count]

FROM product WHERE category\_ID = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count]

FROM product

SELECT product\_id, name, price, quantity, sold\_count

FROM product

WHERE email\_seller = ? AND status = 1 AND quantity > 0

SELECT product\_id, name FROM product

WHERE name LIKE ?

ORDER BY name OFFSET 0 rows FETCH first ? rows only

ProductImageDAO:

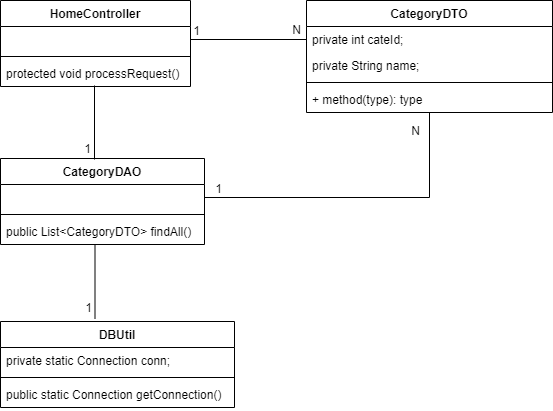
SELECT [image\_id], [url], [is\_main\_img] FROM image\_product WHERE product\_id = ?

## 

## 

## 3. <LOAD CATEGORY>

### a. Class Diagram



### b. Class Specifications

#### Homecontroller Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *this function is used for handle process from user* |

#### 

#### CategoryDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<CategoryDTO> findAll()* | *get all list of categories* |

#### 

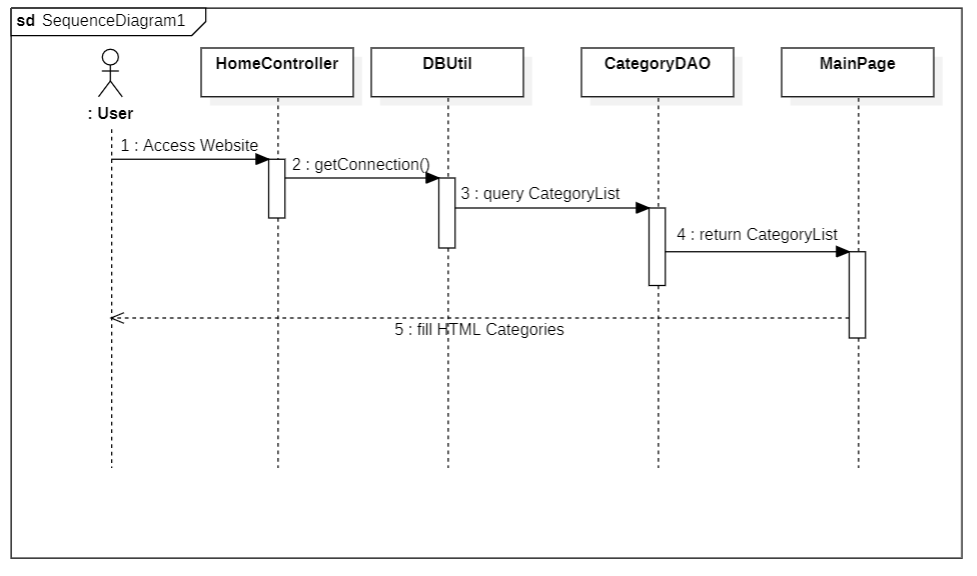
#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *return a database connection* |

### 

### c. Sequence Diagram(s)

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

**

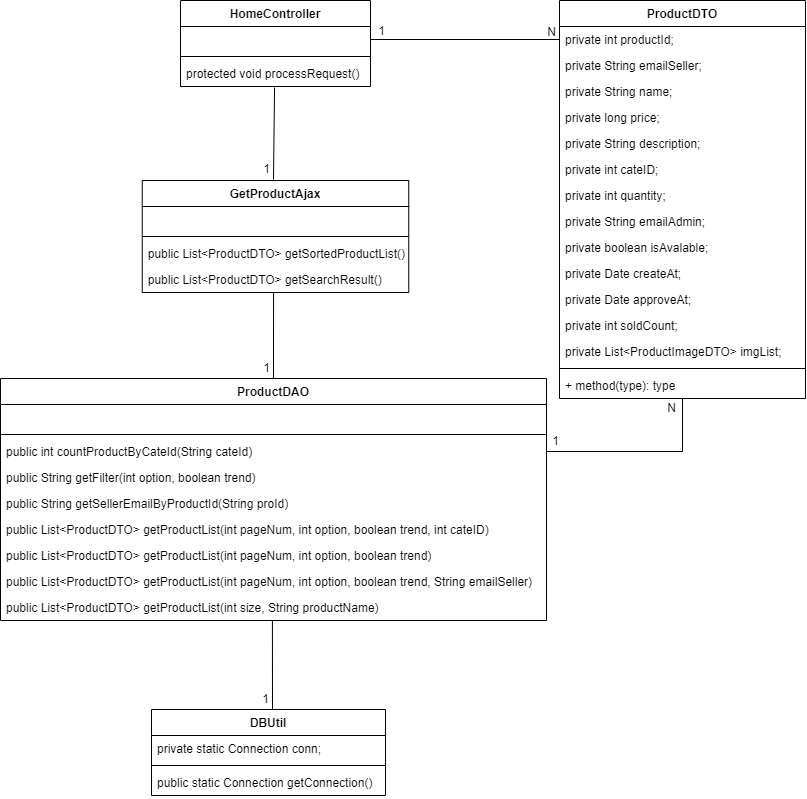
### d. Database queries

CategoryDAO:

SELECT category\_id, name, icon FROM category ORDER BY category\_id ASC

## 4. <SEARCH>

### a. Class Diagram



### 

### 

### 

### 

### b. Class Specifications

#### Homecontroller Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *this function is used for handle process from user* |

#### 

#### GetProductAjax Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductDTO>getSortedProductList()* | *get sorted product list (base on parameter of the request)* |
| *02* | *public List<ProductDTO>getSearchResult()* | *return a list of product(product name and product id) when user typing something to search* |

#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *get total product base on category id for paging* |
| *02* | *public String getFilter(int option, boolean trend)* | *return “order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

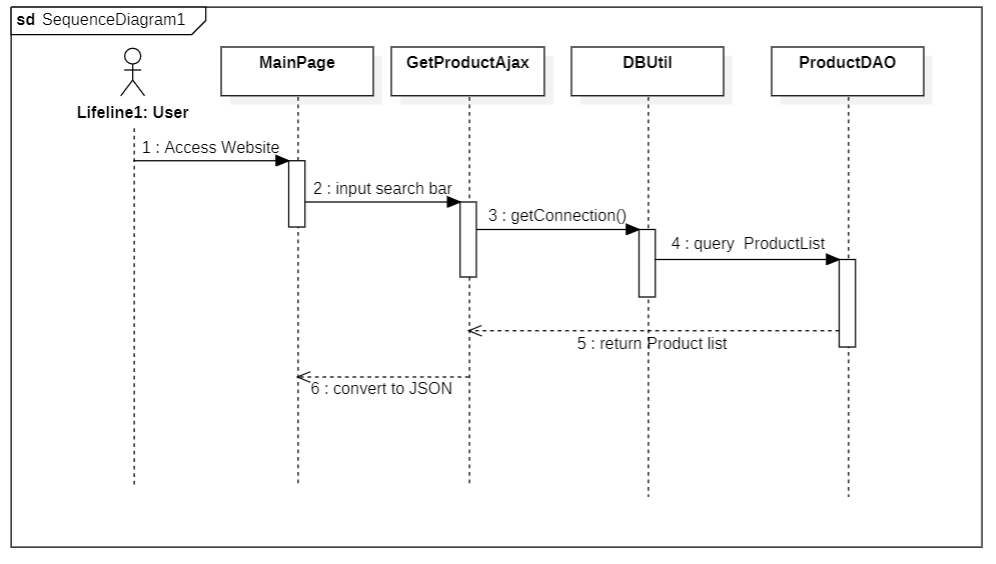
#### 

#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *return a sql Connection object* |

### c. Sequence Diagram(s)

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

**

### d. Database queries

ProductDAO

SELECT count(product\_id) FROM product WHERE category\_id = ?

SELECT email\_seller FROM product WHERE product\_id = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at] , [sold\_count]

FROM product WHERE category\_ID = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count]

FROM product

SELECT product\_id, name, price, quantity, sold\_count FROM product

WHERE email\_seller = ? AND status = 1 AND quantity > 0

SELECT product\_id, name

FROM product

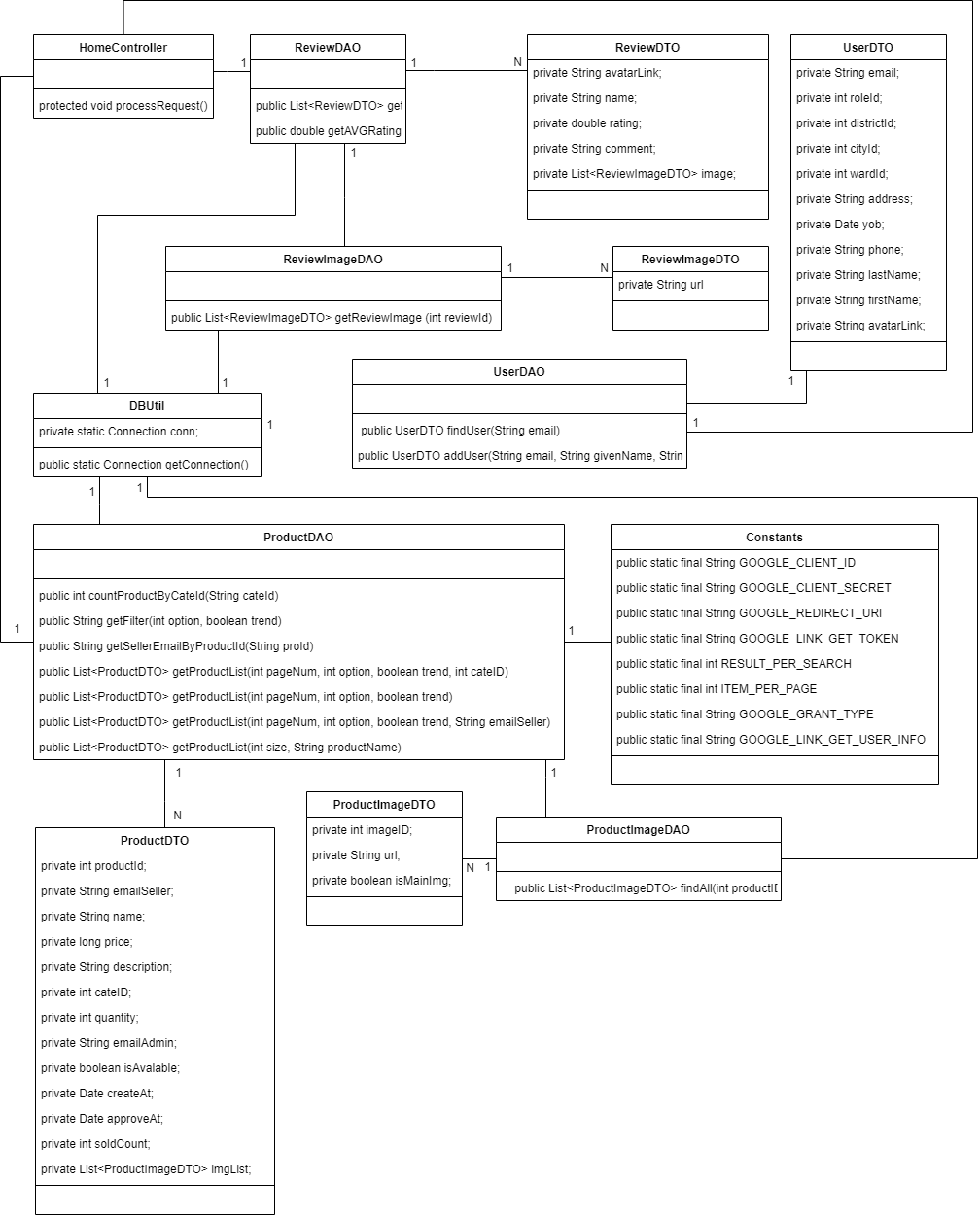
WHERE name LIKE ?

ORDER BY name OFFSET 0 rows FETCH first ? rows only

## 

## 5. <PRODUCT DETAIL>

### a. Class Diagram



### b. Class Specifications

#### HomeController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *take request and redirect to specified product* |

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewDTO> getReview(int productId)* | *show all review of product* |
| *02* | *public double getAVGRatingOfProduct(int productId)* | *show average rating of product* |

#### 

#### ReviewImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewImageDTO> getReviewImage (int reviewId)* | *get all images of a reviewer for product* |

#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* | *get information of seller* |
|  | *public UserDTO addUser(String email, String givenName, String avatarLink)* | *insert a data user into database* |

#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *get total product base on category id for paging* |
| *02* | *public String getFilter(int option, boolean trend)* | *return “order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return an email of seller that sell a product which have an id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* | *get all images of product* |

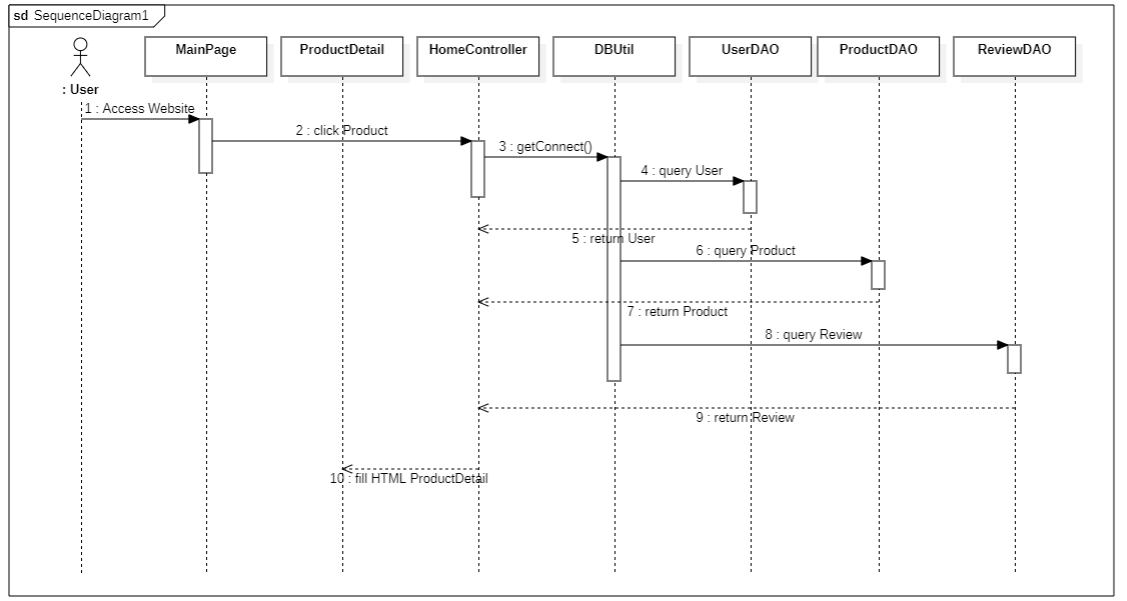
#### 

#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

### c. Sequence Diagram(s)

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

**

### 

### d. Database queries

ReviewImageDAO:

SELECT url FROM image\_review WHERE review\_id = ? ORDER BY image\_id

UserDAO:

SELECT [email], [avatar], [first\_name], [last\_name], [phone], [yob], [address], [ward\_id], [district\_id], [city\_id], [role\_id]

FROM [user] WHERE email = ?

INSERT INTO [user] (email, avatar, last\_name, first\_name, role\_id) VALUES (?, ?, ?, ?, 0)

ProductDAO:

SELECT COUNT(product\_id) FROM product WHERE category\_id = ?

SELECT email\_seller FROM product WHERE product\_id = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at] , [sold\_count]

FROM product WHERE category\_ID = ?

SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] FROM product

SELECT product\_id, name, price, quantity, sold\_count FROM product

WHERE email\_seller = ? AND status = 1 AND quantity > 0

SELECT product\_id, name FROM product

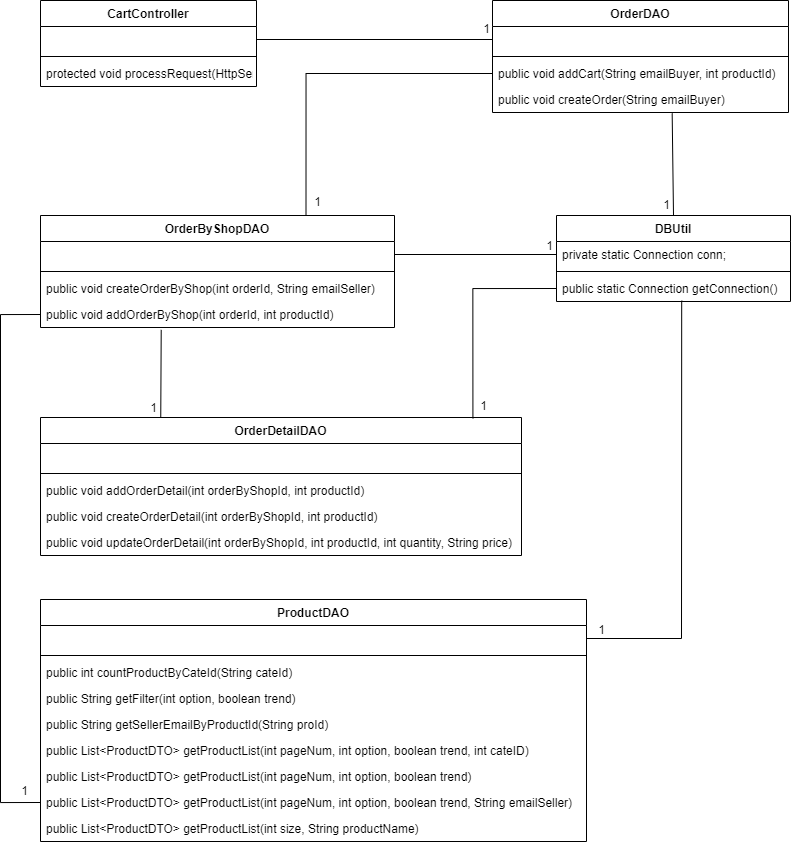
WHERE name LIKE ? ORDER BY name OFFSET 0 rows FETCH first ? rows only

ProductDAO:

SELECT url FROM image\_review WHERE review\_id = ? ORDER BY image\_id

## 6. <BUY PRODUCT>

### a. Class Diagram



### 

### b. Class Specifications

#### CartController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* | *Receive and process requests from users to add products to the order* |

### 

#### OrderDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void addCart(String emailBuyer, int productId)* | *check the existence or create a new order* |
| *02* | *public void createOrder(String emailBuyer)* | *create a new order* |

### 

#### OrderByShopDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void createOrderByShop(int orderId, String emailSeller)* | *check the existence or create a new order by shop* |
| *02* | *public void addOrderByShop(int orderId, int productId)* | *create a new order by shop* |

### 

#### OrderDetailDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void addOrderDetail(int orderByShopId, int productId)* | *check the existence or create a new order detail or update order detail* |
| *02* | *public void createOrderDetail(int orderByShopId, int productId)* | *create a new order detail* |
| *03* | *public void updateOrderDetail(int orderByShopId, int productId, int quantity, String price)* | *update to increase the number of products by 1 in the order* |

### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *get total product base on category id for paging* |
| *02* | *public String getFilter(int option, boolean trend)* | *return “order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return an email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

#### 

#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

### c. Sequence Diagram(s)

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

### 

### 

### d. Database queries

ProductDAO:

SELECT product\_idemail\_seller, name, price, description, category\_id, quantity, sold\_count FROM product WHERE product\_id = ? AND status = 1

SELECT product\_id, name, price, quantity, sold\_count

FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

ORDER BY sold\_count DECS OFFSET itemSkipped ROWS FETCH NEXT 8 ROWS ONLY

UserDAO:

SELECT email, avatar, first\_name, last\_name, phone, address, w.name, d.name, c.name

FROM [user] u LEFT JOIN ward w ON u.ward\_id = w.ward\_id

LEFT JOIN district d ON u.district\_id = d.district\_id

LEFT JOIN city c ON u.city\_id = c.city\_id

WHERE email = (SELECT email\_seller

FROM product

WHERE product\_id = ? )

ReviewDAO:

SELECT u.avatar, u.first\_name, u.last\_name, r.comment, r.rating, r.review\_id

FROM [user] u RIGHT JOIN [order] od ON u.email = od.email\_buyer

RIGHT JOIN order\_by\_shop os ON od.order\_id = os.order\_id

RIGHT JOIN (SELECT order\_detail\_id, order\_by\_shop\_id

FROM [order\_detail]

WHERE product\_id = ?) o ON os.order\_by\_shop\_id = o.order\_by\_shop\_id

LEFT JOIN review r ON r.order\_detail\_id = o.order\_detail\_id

ORDER BY r.review\_id

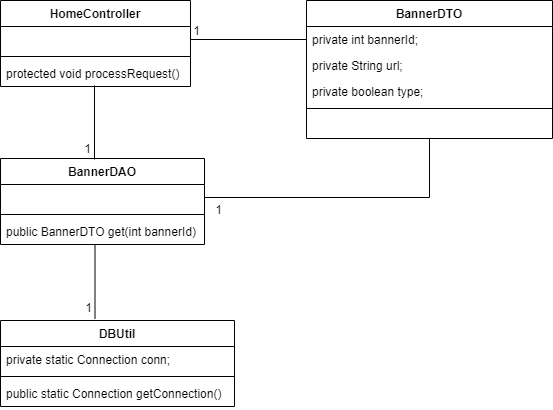
SELECT AVG(r.rating)

FROM [order\_detail] o LEFT JOIN review r ON r.order\_detail\_id = o.order\_detail\_id

WHERE product\_id = ?

## 7. <BANNER>

### a. Class Diagram



### b. Class Specifications

#### HomeController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *receive request and redirect to specific page* |

#### 

#### BannerDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public BannerDTO get(int bannerId)* | *get all data of a banner* |

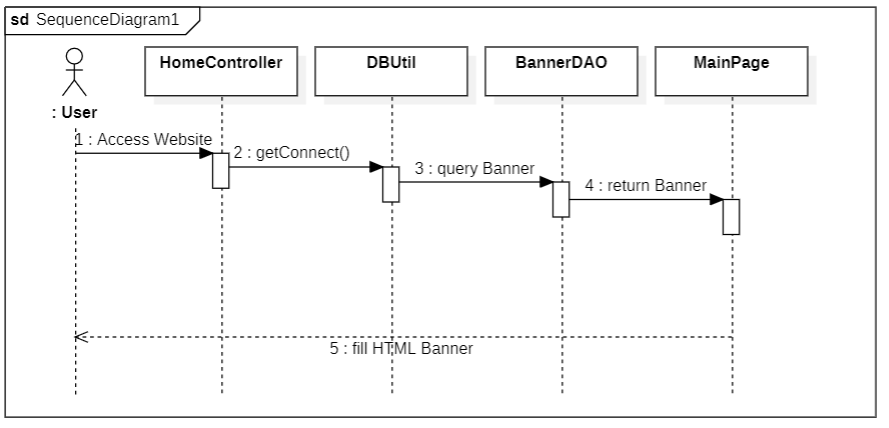
### 

#### DButil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

### 

### c. Sequence Diagram(s)

**

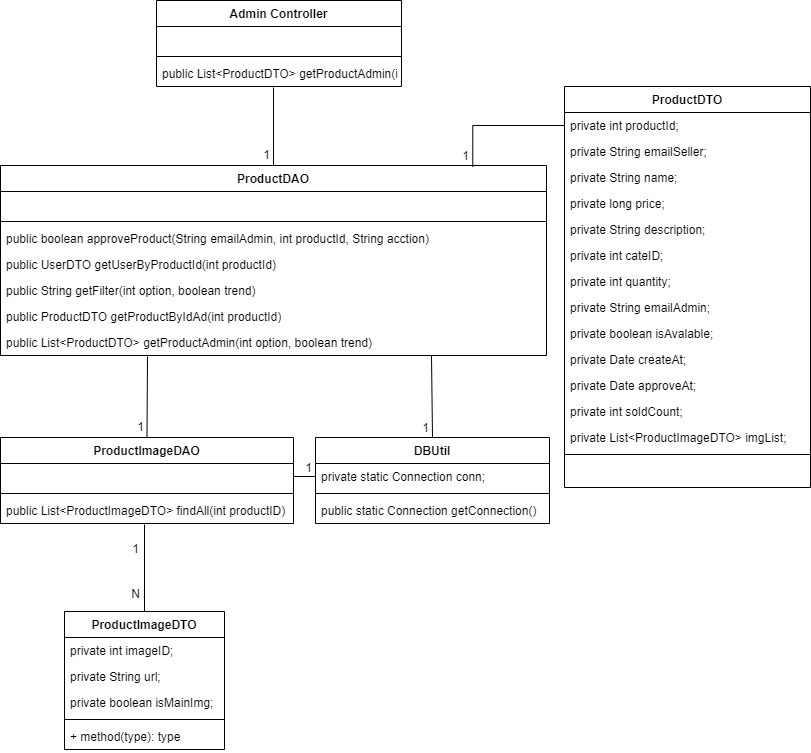
### d. Database queries

BannerDAO:

SELECT banner\_id, url, type\_banner FROM banner WHERE banner\_id = ?

## 8. <PRODUCT AUTHENTICATION>

### a. Class Diagram

****

### b. Class Specifications

#### Admin Controller Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List <ProductDTO> getProductAdmin()* | *Receive and process requests from admin and sub admin to product censorship* |

#### 

#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean approveProduct(String emailAdmin, int productId, String action)* | *approve action, update product data* |
| *02* | *public String getFilter(int option, boolean trend)* | *get filter sort by day (approve\_at, create\_at)* |
| *03* | *public ProductDTO getProductByIdAd(int productId)* | *get product detail by id for admin preview* |
| *04* | *public List<ProductDTO> getProductAdmin(int option, boolean trend)* | *get new product list for admin to authentication* |

#### 

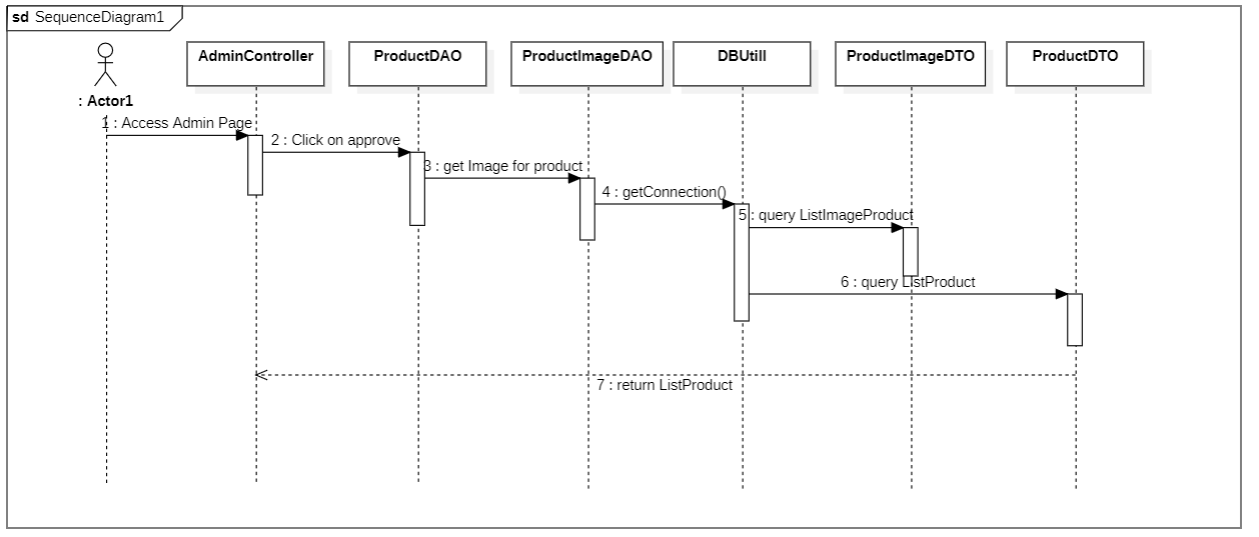
#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* | *get all images of a product* |

#### 

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

ProductDAO:

UPDATE [dbo].[product] SET email\_admin = emailAdmin, approve\_at java.sql.Date.valueOf(LocalDate.now()).toString(), status = WHERE product\_id = " + productId

SELECT email, avatar, first\_name, last\_name, phone, address, w.name, d.name, c.name FROM [user] u LEFT JOIN ward w ON u.ward\_id = w.ward\_id

LEFT JOIN district d ON u.district\_id = d.district\_id

LEFT JOIN city c ON u.city\_id = c.city\_id WHERE email = (SELECT email\_seller FROM product WHERE product\_id = ? )

SELECT [product\_id], [email\_seller], [name],[price], [description], [category\_id], [quantity], [sold\_count] FROM product WHERE product\_id = ? AND email\_admin is not null

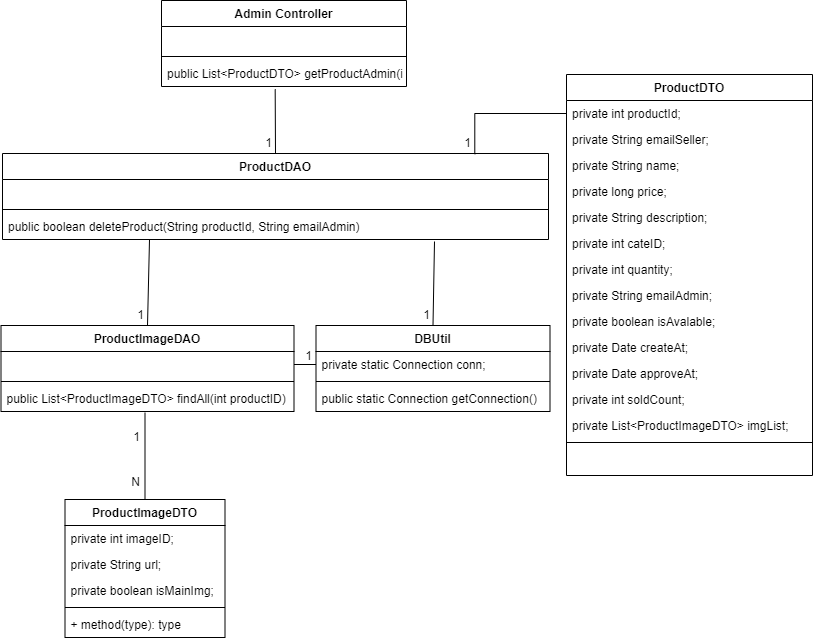
SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [create\_at] FROM product WHERE email\_admin is null AND status is null

ProdutImageDAO:

select [image\_id], [url], [is\_main\_img] from image\_product where product\_id = ?

## 9. <DELETE PRODUCT ON WEBSITE>

### a. Class Diagram

****

### b. Class Specifications

#### Admin Controller Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List <ProductDTO> getProductAdmin()* | *Receive and process requests from admin and sub admin to product censorship* |

#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean deleteProduct(String productId, String emailAdmin)* | *delete product* |

#### 

#### ProductImgaeDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* | *find all images of product* |

#### 

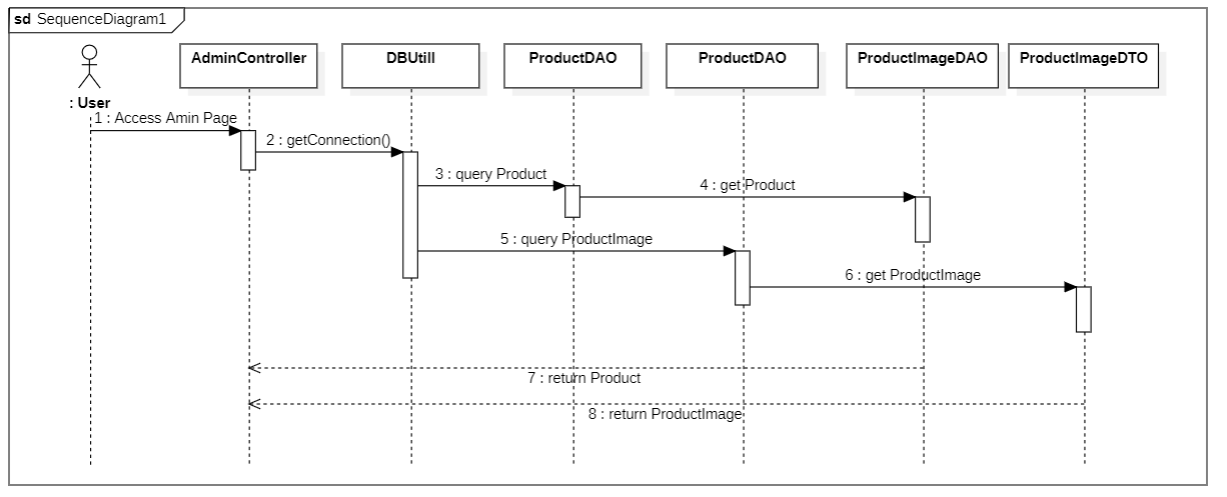
#### 

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

ProductDAO:

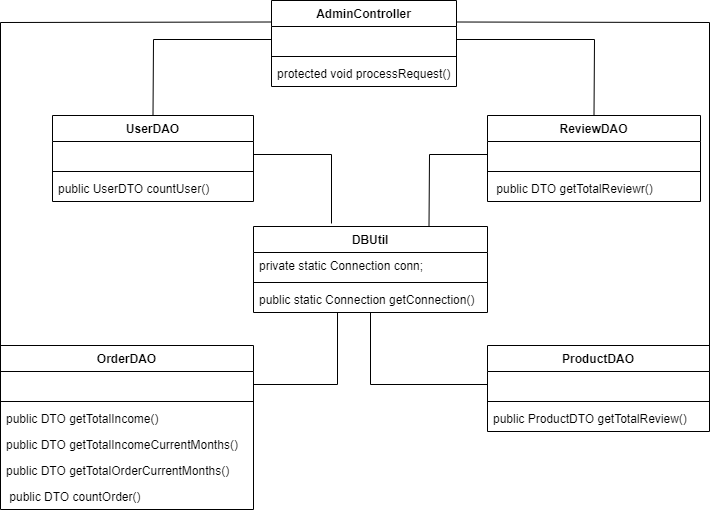
UPDATE [dbo].[product] set status = 0, email\_admin = ? WHERE product\_id = ?

ProdutImageDAO:

select [image\_id], [url], [is\_main\_img] from image\_product where product\_id = ?

## 10. <[DASHBOARD](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.9ix3ptkixfcr)>

### a. Class Diagram

****

### b. Class Specifications

#### 

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *Receive and process requests from admins and sub-admins to get to the dashboard page* |

#### 

#### 

#### OrderDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int getTotalIncome()* | *get total income of the system at all time* |
| *02* | *public ArrayList<Integer> getTotalOrderCurrentMonths(int monthNumber)* | *get array of number of orders in number of month parameter passed in* |
| *03* | *public ArrayList<Integer> getTotalIncomeCurrentMonths(int monthNumber)* | *get array of number of income of the system in recent [month number] parameter passed in* |
| *04* | *public int countOrder()* | *count order* |

#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countUser()* | *return total user in the system* |

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int getTotalReview(int month)* | *return total review in the system* |

#### 

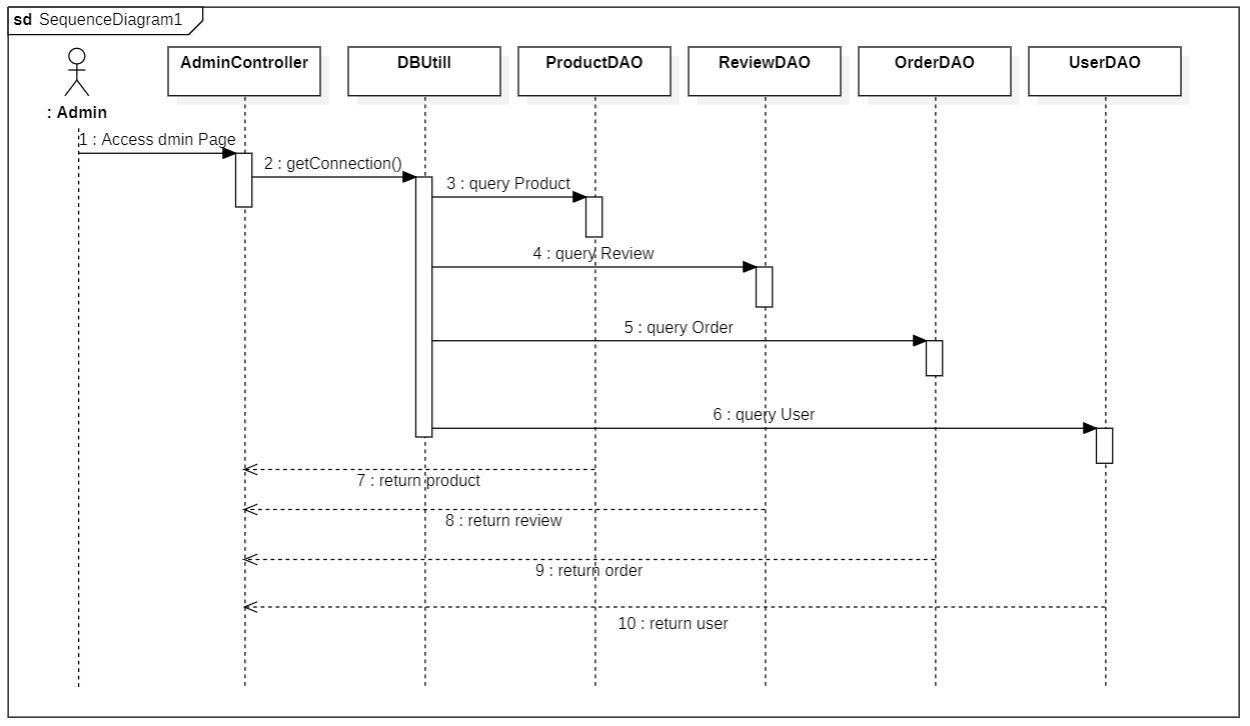
#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public ProductDTO getTotalReview()* |  |

#### 

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

UserDAO:

select count(email) from [dbo].[user]

ReviewDAO:

select count(review.review\_id) from review where month([date])= ?

OrderDAO:

SELECT SUM(od.price) (FROM SELECT order\_id FROM [order] WHERE MONTH([order\_date]) = ?) o LEFT JOIN order\_by\_shop os ON o.order\_id = os.order\_id LEFT JOIN order\_detail od ON od.order\_by\_shop\_id = os.order\_by\_shop\_id

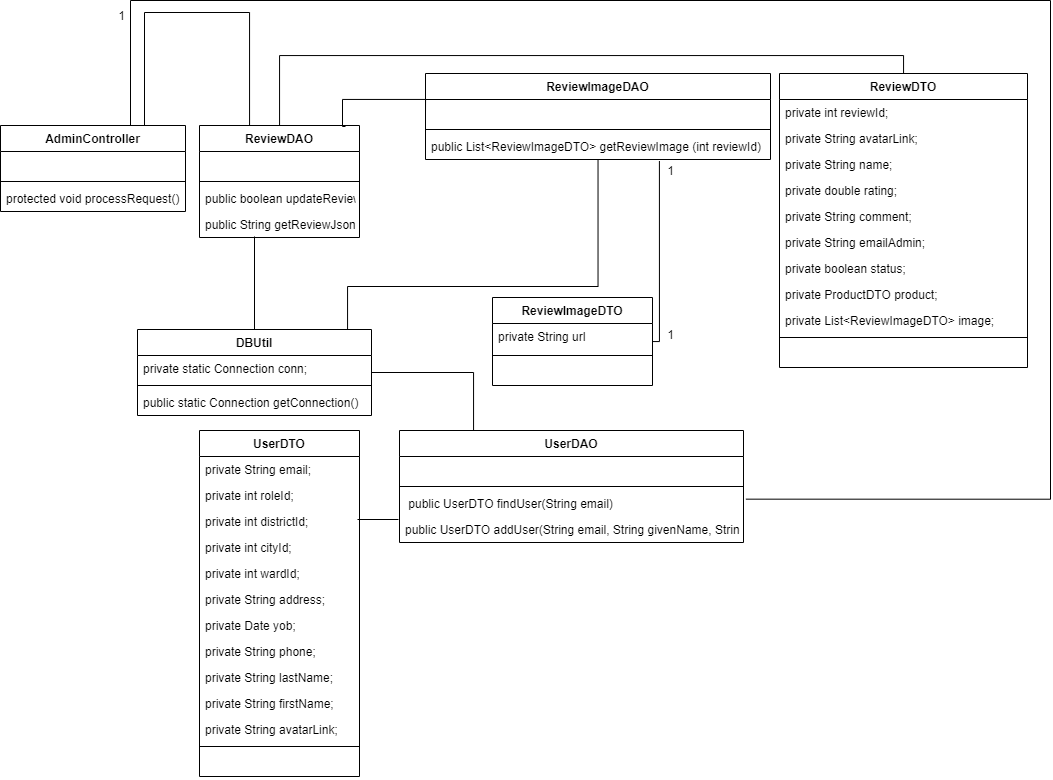
SELECT SUM(od.price) FROM (SELECT order\_id FROM [order]) o LEFT JOIN order\_by\_shop os ON o.order\_id = os.order\_id LEFT JOIN order\_detail od ON od.order\_by\_shop\_id = os.order\_by\_shop\_id

SELECT max([order\_id]) FROM [FEP\_DB].[dbo].[order] where status = 1

Select count(order\_id) from dbo.[order] where month(order\_date) = ?

## 11. <[REVIEW AUTHENTICATION](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.muud4yv7wd2u)>

### a. Class Diagram

****

### b. Class Specifications

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *Receive and process requests from users to censor review list* |

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateReview()* | *update status of a review* |
| *02* | *public String getReviewJson()* | *convert data to output to datatable* |

#### 

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

#### ReviewImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewImageDTO> getReviewImage (int reviewId)* | *get all image of a review* |

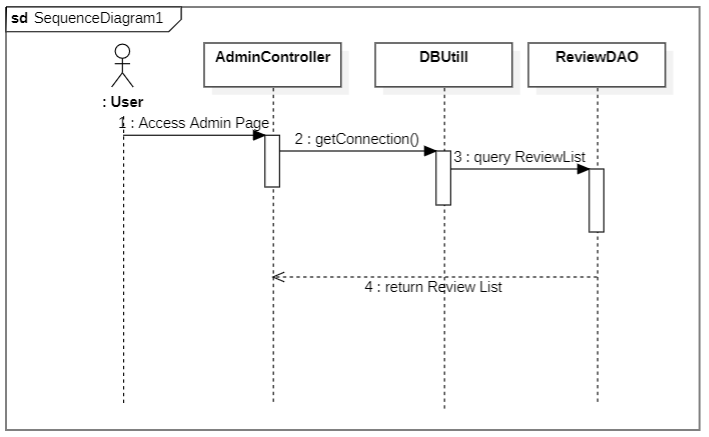
#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* | *retrieve users who have written reviews* |

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

ReviewDAO:

UPDATE review SET email\_admin = ?, status = ? WHERE review\_id = ?

UserDAO:

select [email], [avatar], [first\_name], [last\_name], [phone], [yob], [address], [ward\_id], [district\_id], [city\_id], [role\_id] from [user] where email = ?

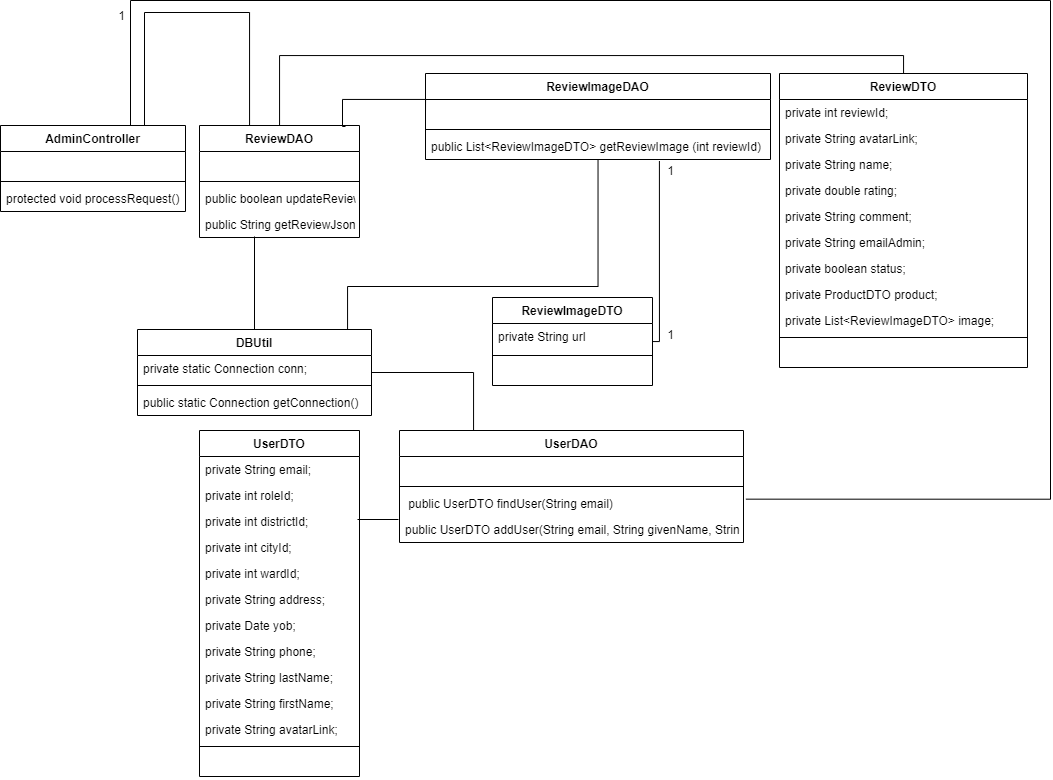
insert into [user] (email, avatar, last\_name, first\_name, role\_id) values (?, ?, ?, ?, 0)

ProdutImageDAO:

select [image\_id], [url], [is\_main\_img] from image\_product where product\_id = ?

## 12. <[DELETE REVIEW ON WEBSITE](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.bfznjt2c410v)>

### a. Class Diagram

****

### b. Class Specifications

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* | *Receive and process requests from users to delete review on website* |

#### 

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateReview()* | *Review moderation status update* |
| *02* | *public String getReviewJson()* | *convert data to output to datatable* |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

#### ReviewImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewImageDTO> getReviewImage (int reviewId)* | *get all photos of each review* |

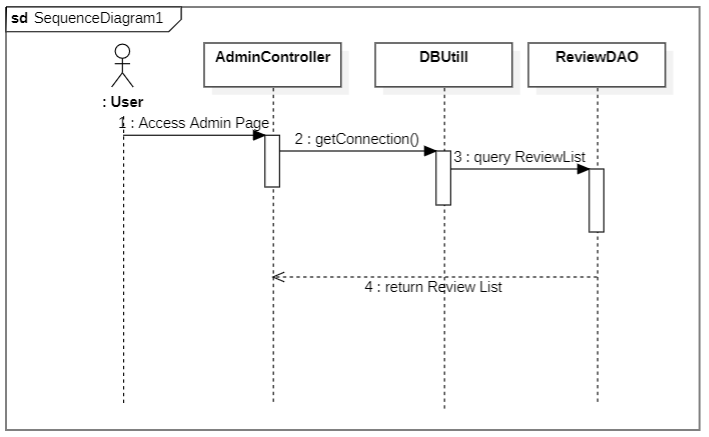
#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* | *retrieve users who have written reviews* |

#### 

### c. Sequence Diagram(s)

****

### 

### d. Database queries

ReviewDAO:

UPDATE review SET email\_admin = ?, status = ? WHERE review\_id = ?

ReviewImageDAO:

SELECT url FROM image\_review WHERE review\_id = ? ORDER BY image\_id

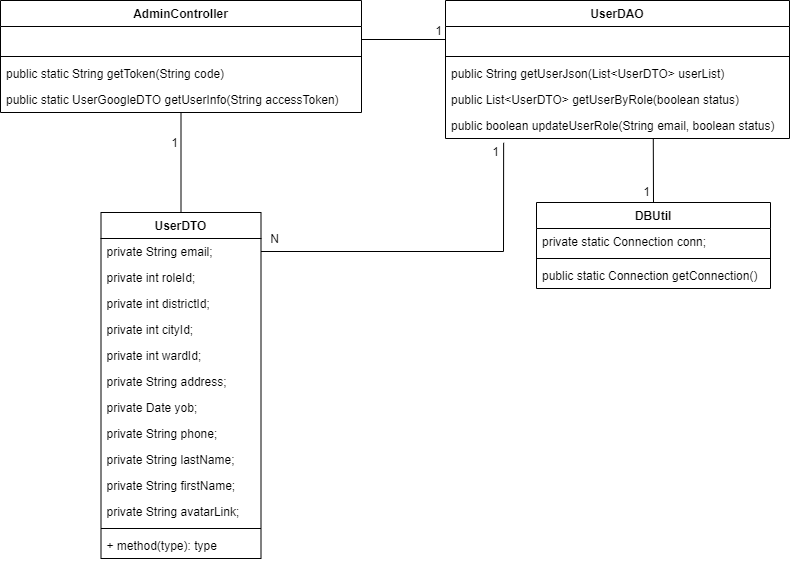
UserDAO:

select [email], [avatar], [first\_name], [last\_name], [phone], [yob], [address], [ward\_id], [district\_id], [city\_id], [role\_id] from [user] where email = ?

insert into [user] (email, avatar, last\_name, first\_name, role\_id) values (?, ?, ?, ?, 0)

## 13. <PROMOTE ROLE>

### a. Class Diagram

****

### 

### b. Class Specifications

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static UserGoogleDTO getUserInfo(String accessToken)* | *Receive and process requests from users to update user role* |

#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public String getUserJson(List<UserDTO> userList)* | *convert data to output to datatable* |
| *02* | *public List<UserDTO> getUserByRole(boolean status)* | *get list user to update role* |
| *03* | *public boolean updateUserRole(String email, boolean status)* | *update role* |

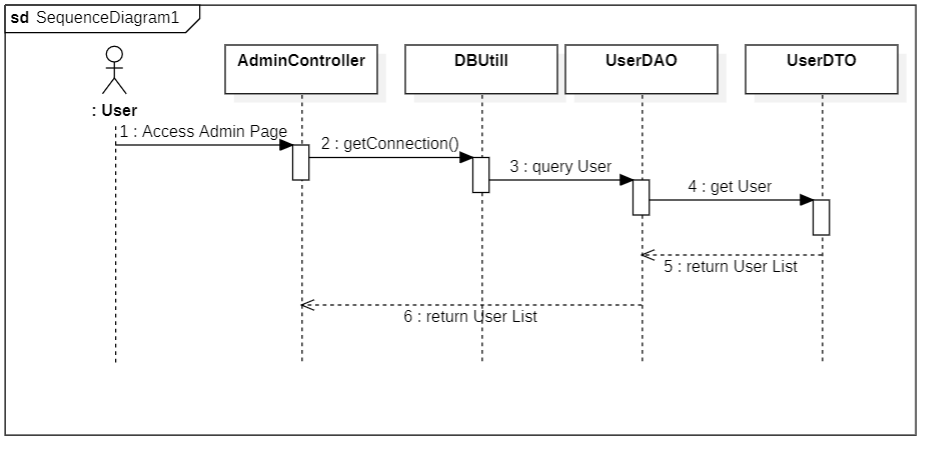
#### 

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

### c. Sequence Diagram(s)

****

### 

### d. Database queries

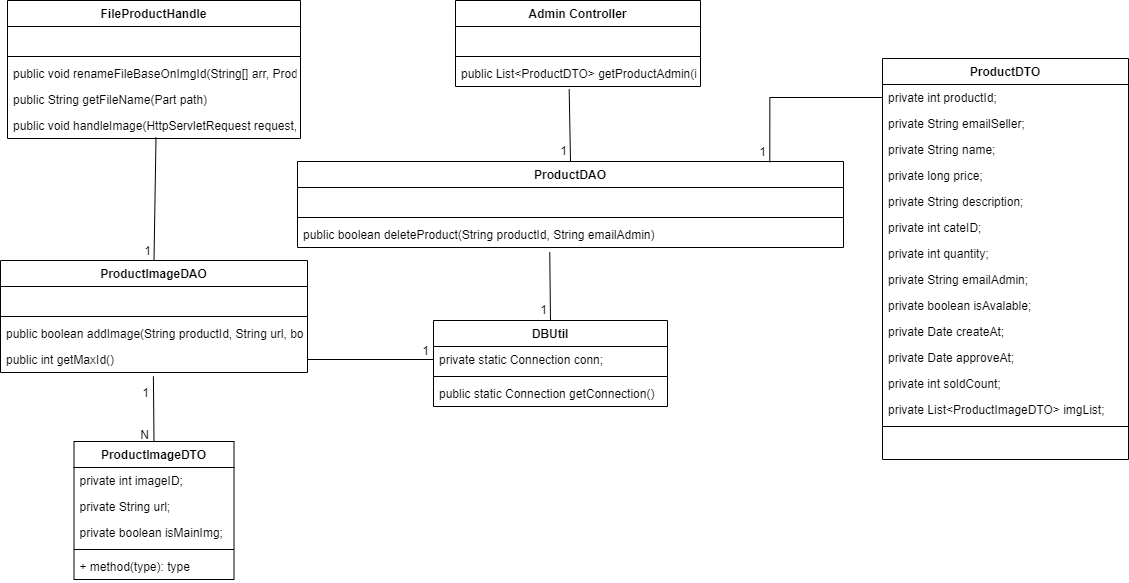
UserDAO:

SELECT [email],[avatar], [first\_name], [last\_name], [phone], [role\_id] FROM [user] WHERE role\_id = ?

UPDATE [user] SET role\_id = ? WHERE email = ?

## 14. <[CREATE NEW PRODUCT](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.wwllibadfayc)>

### a. Class Diagram

****

### b. Class Specifications

#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean addImage(String productId)* | *add image to the database* |
| *02* | *public int getMaxId()* | *get max id of the product table* |

#### 

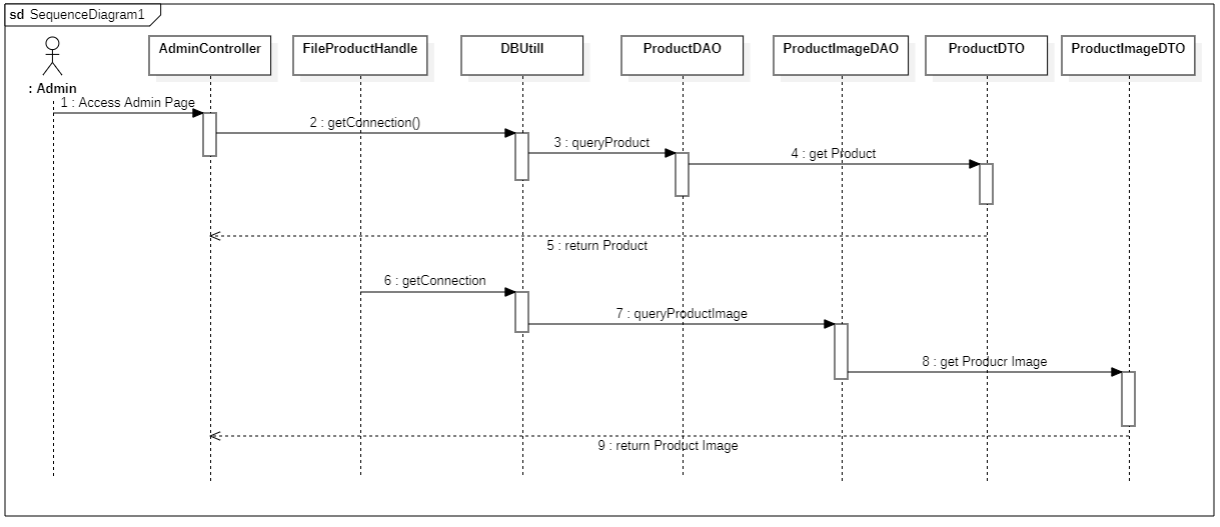
#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

ProductDAO:

UPDATE [dbo].[product] set status = 0, email\_admin = ? WHERE product\_id = ?

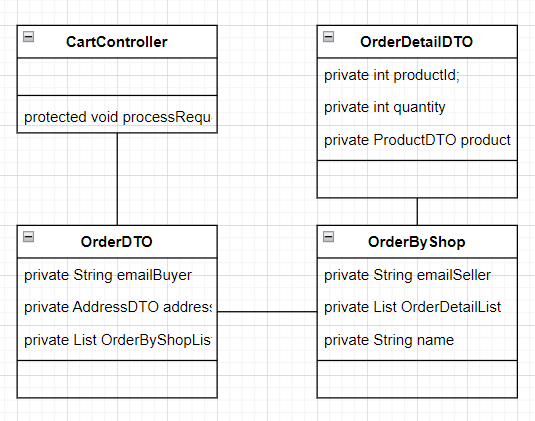
ProductImageDAO:

INSERT INTO [dbo].[image\_product] ([product\_id], [url], [is\_main\_img] ) VALUES (? , ? , ?)

select max(image\_id) from image\_product

## 15. <CART>

### a. Class Diagram



### b. Class Specifications

#### CartController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest(HttpServletRequest request, HttpServletResponse response)* | *update and store data on session* |

### c. Sequence Diagram(s)

### 

### 

### d. Database queries

OrderDAO:

SELECT order\_id FROM [order] WHERE email\_buyer = ? AND [status] = 0

INSERT INTO [order] (email\_buyer) VALUES (?)

OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

OrderDetailDAO:

SELECT quantity FROM [order\_detail] WHERE order\_by\_shop\_id = ? AND product\_id = ?

INSERT INTO [order\_detail] (order\_by\_shop\_id, product\_id) VALUES (?, ?)

UPDATE order\_detail SET quantity = ? , price = ? WHERE order\_by\_shop\_id = ? AND product\_id = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

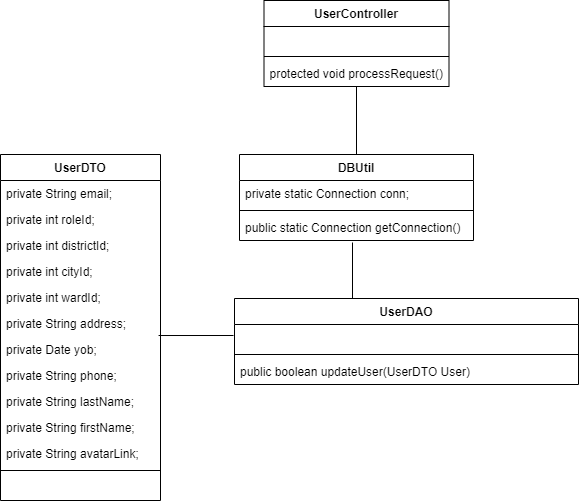
select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

## 16. <[USER DETAIL](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.msptmc6amvik)>

### a. Class Diagram

****

### b. Class Specifications

#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateUser(UserDTO user)* | *update user information to the database* |

#### 

#### 

#### DBuitlClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* | *create connection between project and database* |

#### 

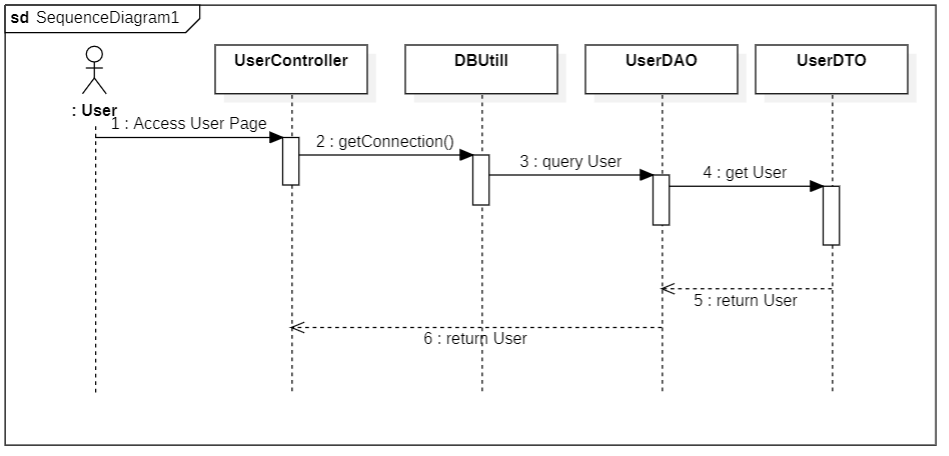
#### 

#### UserControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest(HttpServletRequest request, HttpServletResponse response)* | *handle requests* |

#### 

### c. Sequence Diagram(s)

****

### d. Database queries

## 

UserDAO:

update [user] set [first\_name] = ?,

[last\_name] = ? ,

[phone] = ? ,

[yob] = ? ,

[address] = ? ,

[ward\_id] = ? ,

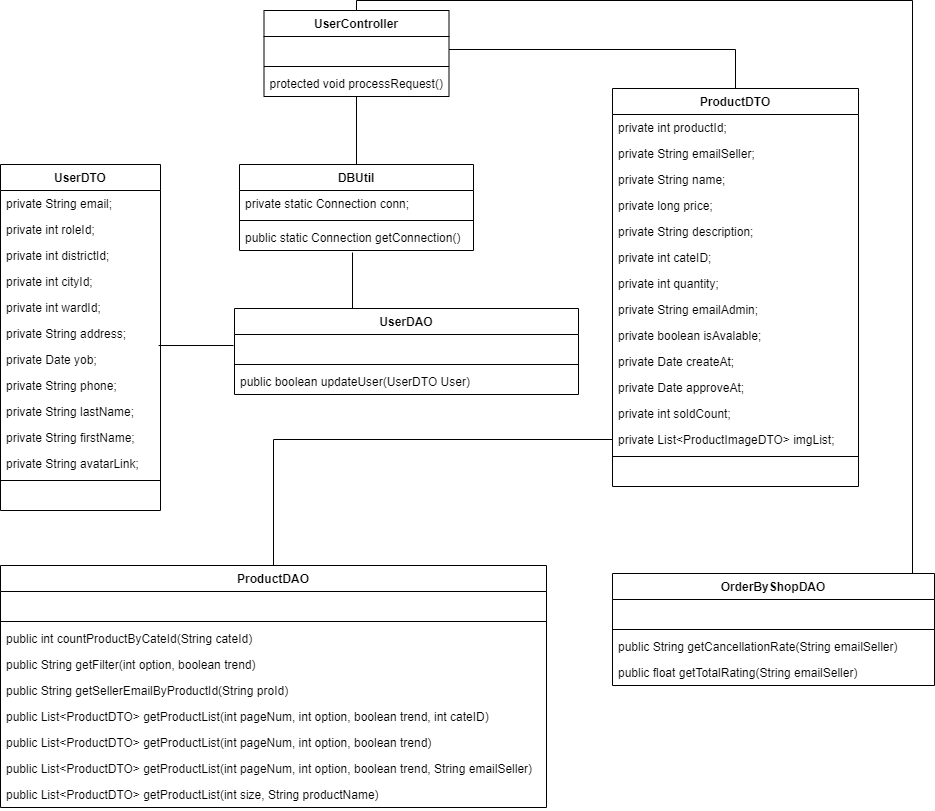
[district\_id] = ? ,

[city\_id] = ? ,

where [email] = ?

## 17. <SHOP PAGE>

### a. Class Diagram

****

### b. Class Specifications

#### 

#### UserDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateUser(UserDTO User)* | *update user* |

#### 

#### OrderByShopDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public String getCancellationRate(String emailSeller)* | *get cancellation order rate of a seller* |
| *02* | *public float getTotalRating(String emailSeller)* | *get review rating of a seller* |

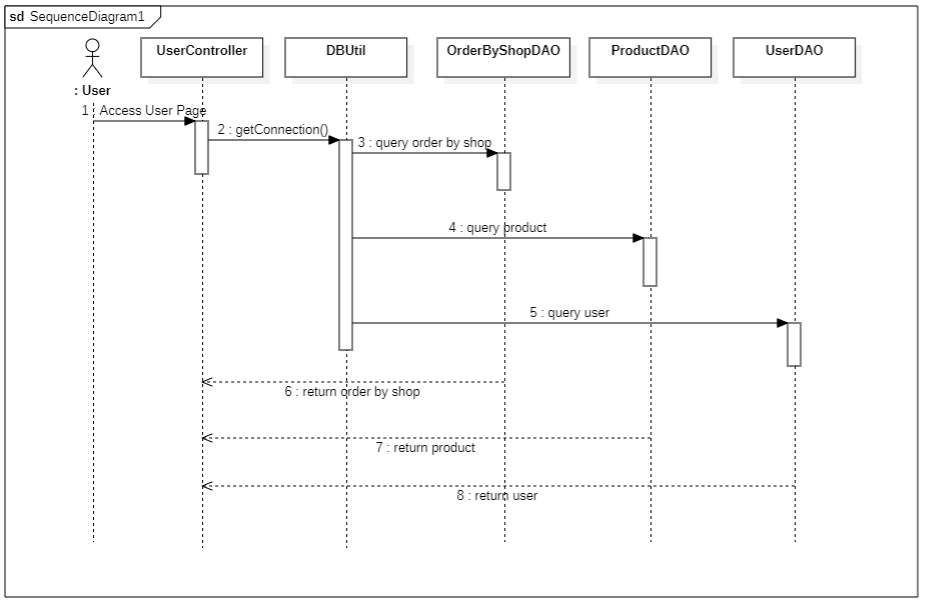
#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *return total product where cateId = param* |
| *02* | *public String getFilter(int option, boolean trend)* | *return "order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

### 

UserDAO:

update [user] set [first\_name] = ?,

[last\_name] = ? ,

[phone] = ? ,

[yob] = ? ,

[address] = ? ,

[ward\_id] = ? ,

[district\_id] = ? ,

[city\_id] = ? ,

where [email] = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

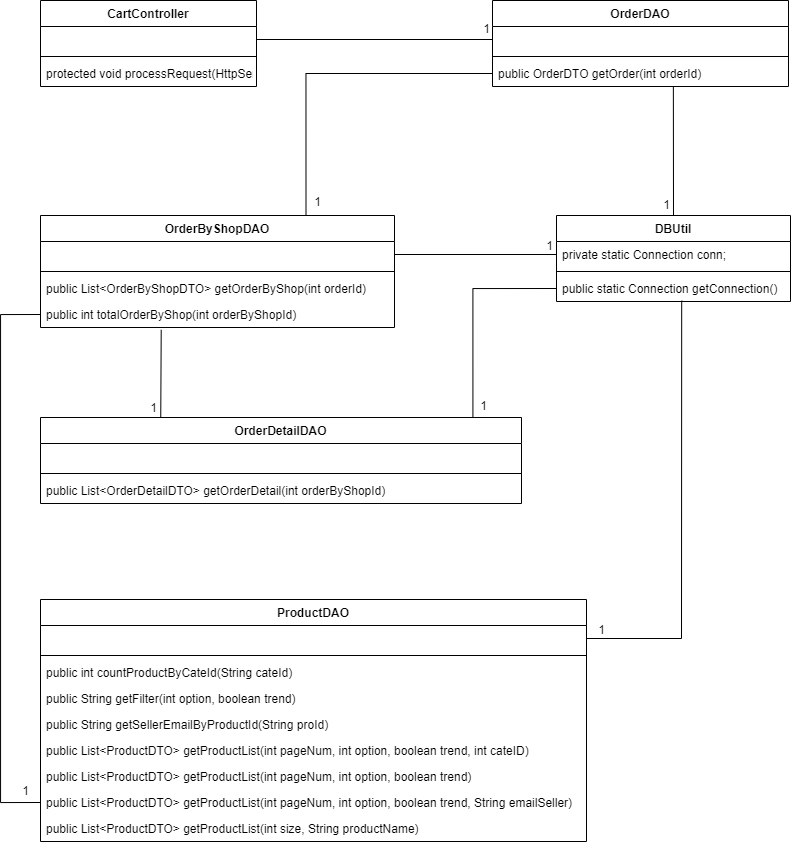
OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

## 18. <BILL DETAIL>

### a. Class Diagram

****

### b. Class Specifications

#### 

#### OrderDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public OrderDTO getOrder(int orderId)* | *get order information* |

#### 

#### OrderByShopDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<OrderByShopDTO> getOrderByShop(int orderId)* | *get details of order list by shop* |
| *02* | *public int totalOrderByShop(int orderByShopId)* | *calculate the price of the order by shop* |

#### 

#### OrderDetailDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<OrderDetailDTO> getOrderDetail(int orderByShopId)* | *get order list details* |

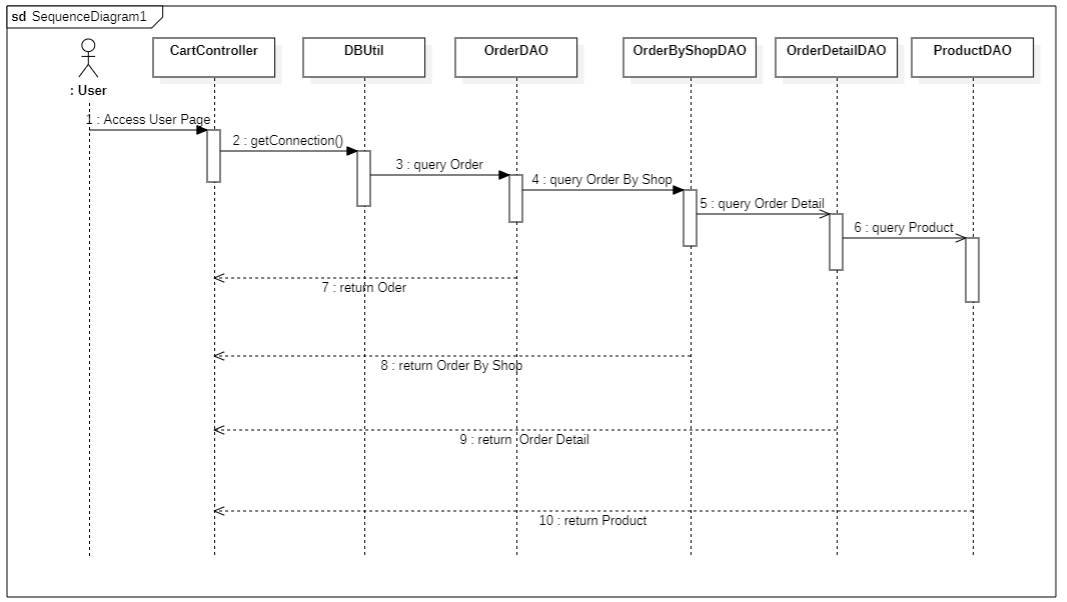
#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *return total product where cateId = param* |
| *02* | *public String getFilter(int option, boolean trend)* | *return "order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

UserDAO:

SELECT [order\_id], delivery\_id, payment\_id, email\_buyer, order\_date, status, address, ward\_id, district\_id, city\_id FROM [order] WHERE email\_buyer = ? AND [status] = ?

OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

OrderDetailDAO:

SELECT quantity FROM [order\_detail] WHERE order\_by\_shop\_id = ? AND product\_id = ?

INSERT INTO [order\_detail] (order\_by\_shop\_id, product\_id) VALUES (?, ?)

UPDATE order\_detail SET quantity = ? , price = ? WHERE order\_by\_shop\_id = ? AND product\_id = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

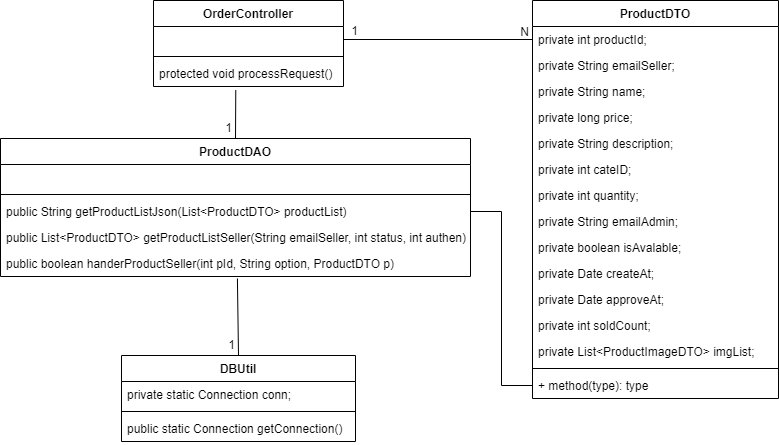
SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

## 

## 19. <SELLER PAGE>

### a. Class Diagram

****

### b. Class Specifications

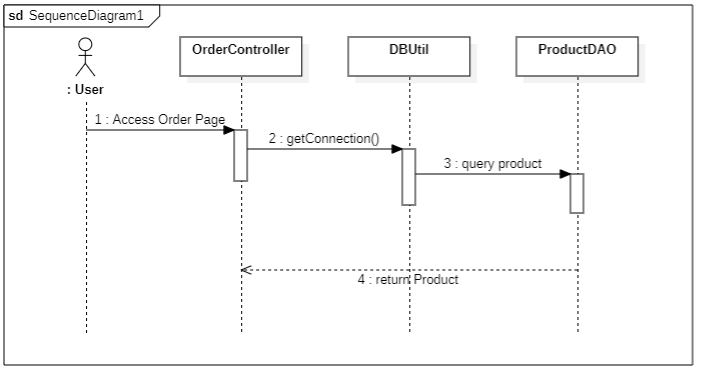
#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public String getProductListJson(List<ProductDTO> productList)* | *convert list to json for datatable* |
| *02* | *public List<ProductDTO> getProductListSeller(String emailSeller, int status, int authen)* | *get the product listings that the seller requested* |
| *03* | *public boolean handerProductSeller(int pId, String option, ProductDTO p)* | *handling seller requests* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

UserDAO:

SELECT [order\_id], delivery\_id, payment\_id, email\_buyer, order\_date, status, address, ward\_id, district\_id, city\_id FROM [order] WHERE email\_buyer = ? AND [status] = ?

OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

OrderDetailDAO:

SELECT quantity FROM [order\_detail] WHERE order\_by\_shop\_id = ? AND product\_id = ?

INSERT INTO [order\_detail] (order\_by\_shop\_id, product\_id) VALUES (?, ?)

UPDATE order\_detail SET quantity = ? , price = ? WHERE order\_by\_shop\_id = ? AND product\_id = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

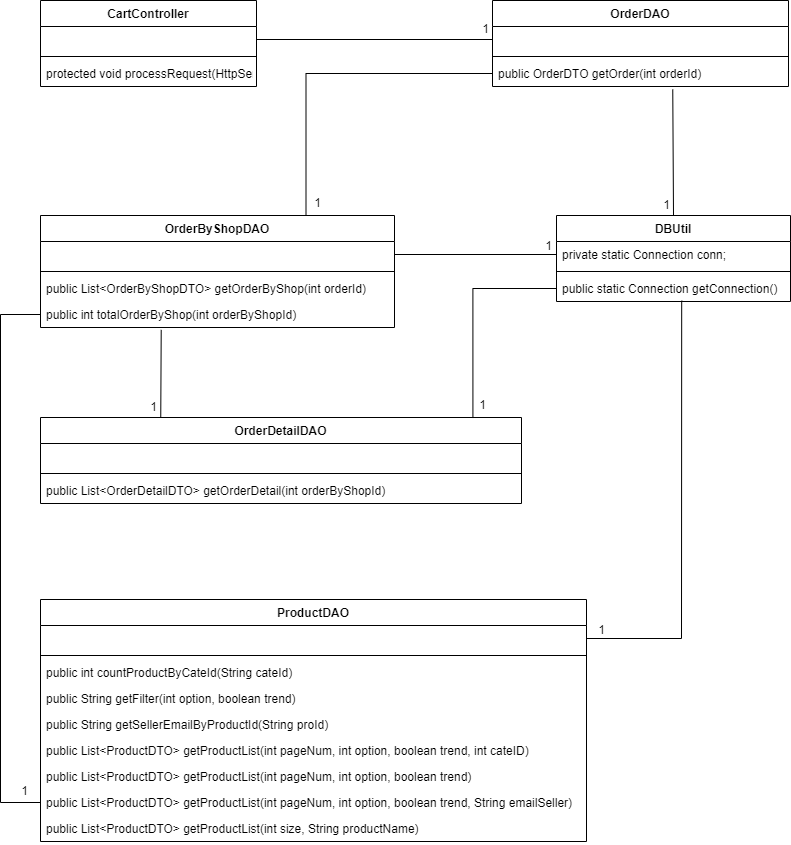
SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

## 

## 20. <PURCHASE HISTORY>

### a. Class Diagram

****

### 

### b. Class Specifications

#### 

#### OrderDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public OrderDTO getOrder(int orderId)* | *return a order base on order id* |

#### 

#### OrderByShopDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<OrderByShopDTO> getOrderByShop(int orderId)* | *return a list of order by shop base on order id* |
| *02* | *public int totalOrderByShop(int orderByShopId)* | *count order by shop base on order id* |

#### 

#### OrderDetailDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<OrderDetailDTO> getOrderDetail(int orderByShopId)* | *get order details* |

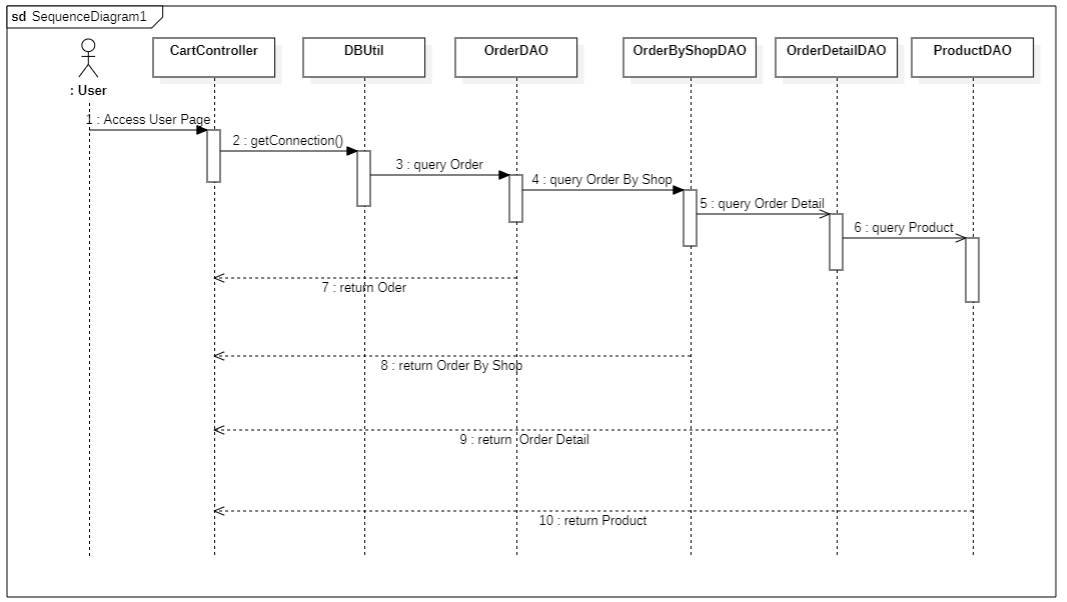
#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *return total product where cateId = param* |
| *02* | *public String getFilter(int option, boolean trend)* | *return "order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

UserDAO:

SELECT [order\_id], delivery\_id, payment\_id, email\_buyer, order\_date, status, address, ward\_id, district\_id, city\_id FROM [order] WHERE email\_buyer = ? AND [status] = ?

OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

OrderDetailDAO:

SELECT quantity FROM [order\_detail] WHERE order\_by\_shop\_id = ? AND product\_id = ?

INSERT INTO [order\_detail] (order\_by\_shop\_id, product\_id) VALUES (?, ?)

UPDATE order\_detail SET quantity = ? , price = ? WHERE order\_by\_shop\_id = ? AND product\_id = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

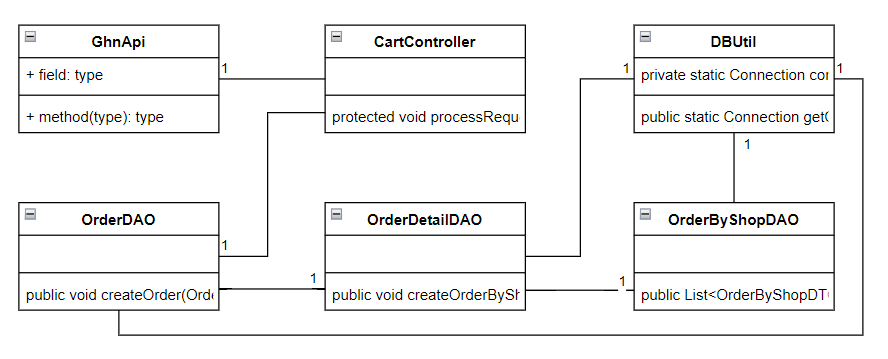
SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

## 

## 21. <PAYMENT>

### a. Class Diagram

****

### b. Class Specifications

#### GhnApi

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static String createOrder(UserDTO user, UserDTO seller, List<OrderDetailDTO> orderDetails, String cod)* | *create a giao hang nhanh order using REST api* |

#### OrderDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void createOrder(OrderDTO order)* | *insert data into database in table order* |

#### 

#### OrderByShopDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<OrderByShopDTO> getOrderByShop(int orderId)* | *insert data into database in table order\_by\_shop* |

#### 

#### OrderDetailDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void createOrderByShop(int orderId, OrderByShopDTO obs)* | *insert data into database in table order\_detail* |

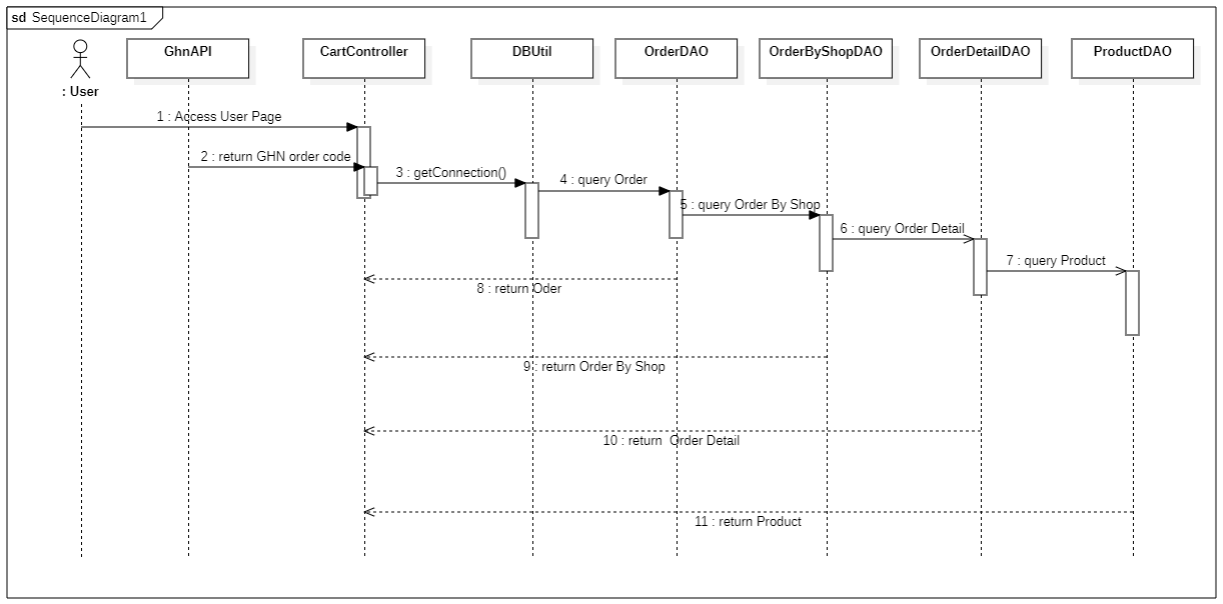
#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* | *return total product where cateId = param* |
| *02* | *public String getFilter(int option, boolean trend)* | *return "order by” sql statement* |
| *03* | *public String getSellerEmailByProductId(String proId)* | *return a email of seller that sell a product which have a id passed in* |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where category id = parameter passed in* |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc)* |
| *06* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, String emailSeller)* | *return a product list base on page number, sort by option (Use the static int final variable in this class),*  *data trend (desc, asc) where email of seller = parameter passed in* |
| *07* | *public List<ProductDTO> getProductList(int size, String productName)* | *get array of product base on product name and size of the array passed in* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

UserDAO:

SELECT [order\_id], delivery\_id, payment\_id, email\_buyer, order\_date, status, address, ward\_id, district\_id, city\_id FROM [order] WHERE email\_buyer = ? AND [status] = ?

OrderByShopDAO:

INSERT INTO [order\_by\_shop] (order\_id, email\_seller) VALUES (?, ?)

SELECT order\_by\_shop\_id FROM order\_by\_shop WHERE order\_id = ? AND email\_seller = ?

OrderDetailDAO:

SELECT quantity FROM [order\_detail] WHERE order\_by\_shop\_id = ? AND product\_id = ?

INSERT INTO [order\_detail] (order\_by\_shop\_id, product\_id) VALUES (?, ?)

UPDATE order\_detail SET quantity = ? , price = ? WHERE order\_by\_shop\_id = ? AND product\_id = ?

ProductDAO:

select count(product\_id) from product where category\_id = ?

select email\_seller from product where product\_id = ?

select product\_id, name from product where name like ? order by name offset 0 rows fetch first ? rows only

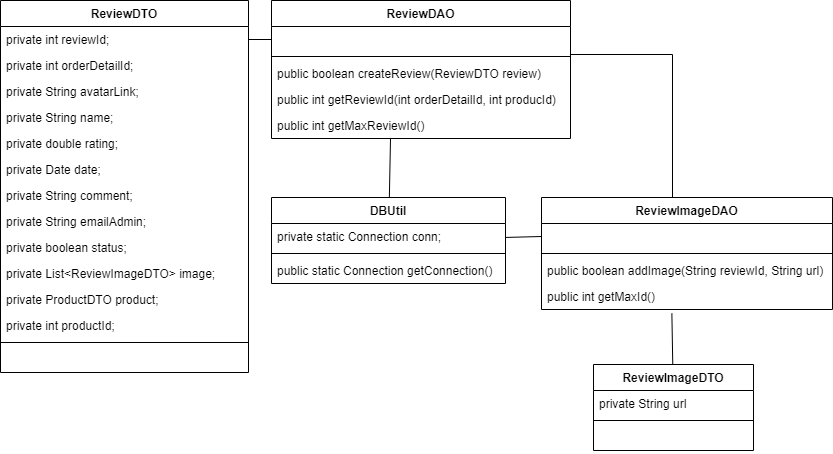
select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

SELECT product\_id, name, price, quantity, sold\_count FROM product WHERE email\_seller = ? AND status = 1 AND quantity > 0

select [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [email\_admin], [status], [create\_at], [approve\_at], [sold\_count] from product where category\_ID = ?

## 22. <[R](https://docs.google.com/document/d/1e0yA5Z7L8BlFa-ovUPd5vcr6bQ8bFkrO/edit#heading=h.hk9b0i40nsjz)EVIEW>

### a. Class Diagram

****

### b. Class Specifications

#### 

#### ReviewDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean createReview(ReviewDTO review)* | *create review to database* |
| *02* | *public int getReviewId(int orderDetailId, int producId)* | *get review id from order detail id and product id* |
| *03* | *public int getMaxReviewId()* | *get maximum of review id* |

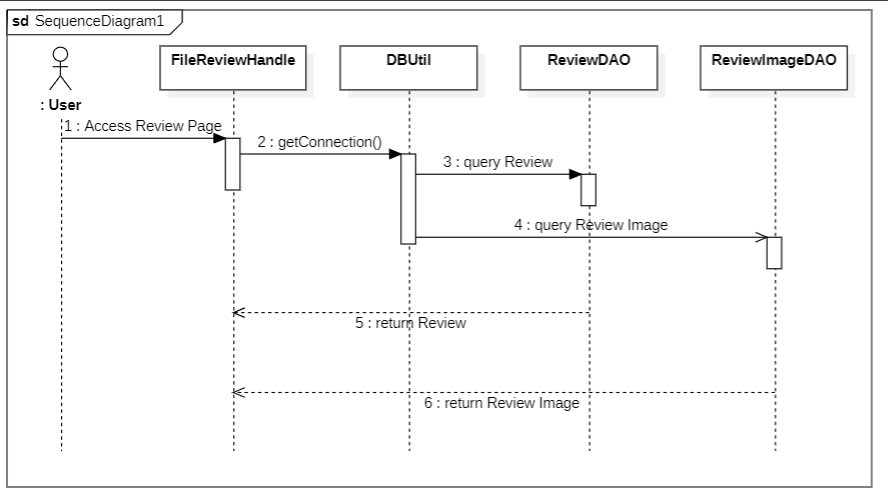
#### 

#### ReviewImageDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean addImage(String reviewId, String url)* | *add review image to database* |
| *02* | *public int getMaxId()* | *get maximum of review image id* |

#### 

### c. Sequence Diagram(s)

**

### d. Database queries

INSERT INTO [dbo].[image\_review] ([review\_id] , [url]), values (?, ?)

select max([image\_id]) from [image\_review]

select review\_id from review r

inner join order\_detail od on r.order\_detail\_id = od.order\_detail\_id

where r.order\_detail\_id = ? and od.product\_id = ?

select max(review\_id) from review

INSERT INTO [dbo].[review] ([order\_detail\_id] , [rating], [comment], [email\_admin], [status], [date]) values (?, ?, ?, ?, ?, ?)

# 

# III. Database Tables

## 1. < city >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | city\_id | NVARCHAR | 20 | x | x | PRIMARY KEY |  |
| 2 | name | NVARCHAR | 255 |  | x |  |  |

## 2. < district >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | district\_id | NVARCHAR | 20 | x | x | PRIMARY KEY |  |
| 2 | name | NVARCHAR | 255 |  | x |  |  |
| 3 | city\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_DISTRICT\_RE\_CITY |

## 3. < Ward >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | ward\_id | NVARCHAR | 20 | x | x | PRIMARY KEY |  |
| 2 | name | NVARCHAR | 255 |  | x |  |  |
| 3 | district\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_WARD\_RE\_DISTRICT |

## 4. < Role >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | role\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | name | NVARCHAR | 10 | x | x |  |  |

## 

## 5. < user >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | email | VARCHAR | 50 | x | x | PRIMARY KEY |  |
| 2 | avatar | TEXT |  |  | x |  |  |
| 3 | first\_name | NVARCHAR | 40 |  |  |  |  |
| 4 | last\_name | NVARCHAR | 40 |  |  |  |  |
| 5 | phone | VARCHAR | 10 | x |  |  |  |
| 6 | yob | DATE |  |  |  |  |  |
| 7 | address | NTEXT |  |  |  |  |  |
| 8 | ward\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_USER\_RE\_WARD |
| 19 | district\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_USER\_RE\_DISTRICT |
| 10 | city\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_USER\_RE\_CITY |
| 11 | role\_id | INT |  |  |  | FOREIGN KEY | FK\_USER\_RE\_ROLE |

## 6. < category >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | category\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | name | NVARCHAR | 40 | x | x |  |  |
| 3 | icon | VARCHAR | 100 |  |  |  |  |

## 7. < delivery\_method >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | delivery \_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | name | NVARCHAR | 40 | x | x |  |  |

## 8. < payment\_method >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | payment \_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | name | NVARCHAR | 40 | x | x |  |  |

## 9. < product >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | product\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | email (seller) | VARCHAR | 50 |  | x | FOREIGN KEY | FK\_PRODUCT\_RE\_SELLER |
| 3 | name | NVARCHAR | 100 |  | x |  |  |
| 4 | price | NUMERIC | (12.0) |  | x |  | CHECK price >= 0 |
| 5 | description | NTEXT |  |  |  |  |  |
| 6 | category\_id | INT |  |  | x | FOREIGN KEY | FK\_PRODUCT\_RE\_CATEGORY |
| 7 | quantity | INT |  |  | x |  | DEFAULT (0)  CHECK quantity >= 0 |
| 8 | email\_admin | VARCHAR | 50 |  |  | FOREIGN KEY | FK\_PRODUCT\_RE\_ADMIN |
| 9 | sold\_count | INT |  |  | x |  | DEFAULT (0) |
| 10 | status | BIT |  |  | x |  | DEFAULT (0) |
| 11 | create\_at | DATE |  |  | x |  |  |
| 12 | approve\_at | DATE |  |  |  |  |  |

## 10. < image\_product >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | image \_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | product\_id | INT |  |  | x | FOREIGN KEY | FK\_IMAGE\_PRODUCT\_RE\_PRODUCT |
| 3 | url | TEXT |  |  |  |  |  |
| 4 | Is\_main\_img | BIT |  |  | x |  | DEFAULT (0) |

## 11. < order >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | order\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | delivery\_id | INT |  |  |  | FOREIGN KEY | FK\_ORDER\_RE\_DELIVERY |
| 3 | payment\_id | INT |  |  |  | FOREIGN KEY | FK\_ORDER\_RE\_PAYMENT |
| 4 | email (buyer) | VARCHAR | 50 |  | x | FOREIGN KEY | FK\_ORDER\_RE\_BUYER |
| 5 | order\_date | DATE |  |  |  |  |  |
| 6 | shipper\_date | DATE |  |  |  |  |  |
| 7 | address | NTEXT |  |  |  |  |  |
| 8 | ward\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_ORDER\_RE\_WARD |
| 9 | district\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_ORDER\_RE\_DISTRICT |
| 10 | city\_id | NVARCHAR | 20 |  |  | FOREIGN KEY | FK\_ORDER\_RE\_CITY |

## 12. < order\_detail>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | order\_details\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | order\_by\_shop\_id | INT |  |  | x | FOREIGN KEY | FK\_ORDER\_DETAIL\_RE\_ORDER |
| 3 | product\_id | INT |  |  | x | FOREIGN KEY | FK\_ORDER\_DETAIL\_RE\_PRODUCT |
| 4 | quantity | INT |  |  | x |  | CHECK quantity >= 0 |
| 5 | price | NUMERIC | (12,0) |  |  |  | CHECK price >= 0 |

## 13. < order\_by\_shop>

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | order\_by\_shop\_id | INT |  | x | x | PRIMARY KEY | INDENTITY (0,1) |
| 2 | order \_id | INT |  |  | x | FOREIGN KEY | FK\_FEP\_ORDER\_BY\_SHOP\_RE\_FEP\_ORDER |
| 3 | email (seller) | VARCHAR | 50 |  | x | FOREIGN KEY | FK\_FEP\_ORDER\_BY\_SHOP\_RE\_USER |
| 4 | transaction\_fee | NUMERIC (12,0) |  |  | x |  |  |
| 5 | transport\_fee | NUMERIC (12,0) |  |  |  |  |  |
| 6 | shipped\_date | DATE |  |  |  |  |  |

## 14. < review >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | review \_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | order\_detail\_id | INT |  |  | x | FOREIGN KEY | FK\_REVIEW\_RE\_ODER\_DETAIL |
| 3 | rating | FLOAT |  |  | x |  |  |
| 4 | comment | NTEXT |  |  |  |  |  |
| 5 | email (admin) | VARCHAR | 50 |  |  | FOREIGN KEY | FK\_REVIEW\_RE\_ADMIN |
| 6 | status | BIT |  |  | x |  | DEFAULT (0) |

## 15. < image\_review >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | image \_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | review\_id | INT |  |  | x | FOREIGN KEY | FK\_IMAGE\_REVIEW\_RE\_REVIEW |
| 3 | url | TEXT |  |  | x |  |  |

## 16. < banner >

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | banner\_id | INT |  | x | x | PRIMARY KEY | IDENTITY (0,1) |
| 2 | name | VARCHAR | 100 |  | x |  |  |
| 3 | type\_banner | BIT |  |  | x |  | DEFAULT (0) |