

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

**Software Design Document**

– HoChiMinh, September 2022 –

**Table of Contents**

[I. Overview](#_heading=h.1fob9te) **6**

[1. Code Packages/Namespaces](#_heading=h.lnxbz9) 6

[2. Database Schema](#_heading=h.qfczxfs6okw8) 7

[**II. Code Designs**](#_heading=h.661e4m1iw2hj) **10**

[1. <LOGIN >](#_heading=h.1ksv4uv) 10

[b. Class Specifications](#_heading=h.e6jdzu7icvjn) 11

[UserController Class](#_heading=h.kgvq6aky2mda) 11

[UserDAO Class](#_heading=h.hp4g4emx9hrb) 11

[DButil Class](#_heading=h.puip72766cei) 11

[c. Sequence Diagram(s)](#_heading=h.al9u58944w90) 11

[d. Database queries](#_heading=h.3o9qajbh4sdu) 12

[2. <SORT>](#_heading=h.5jei5j7ydty4) 12

[a. Class Diagram](#_heading=h.23jbt6mg2kt0) 12

[b. Class Specifications](#_heading=h.7mb2x2mybrho) 13

[Homecontroller Class](#_heading=h.upos0cy1i1ob) 13

[ProductDAO Class](#_heading=h.uzvb01buqvml) 13

[ProductImageDAO Class](#_heading=h.kdt0td9uu6pb) 13

[DButil Class](#_heading=h.sa7wpkvu6q98) 13

[c. Sequence Diagram(s)](#_heading=h.ig5sf8k512vg) 14

[d. Database queries](#_heading=h.eig7yn3t0nux) 14

[3. <LOAD CATEGORY>](#_heading=h.qn102eh111i9) 15

[a. Class Diagram](#_heading=h.45sj7i1ukbsj) 15

[b. Class Specifications](#_heading=h.la5tdtmvcuh2) 15

[Homecontroller Class](#_heading=h.5fu5ltiwxsi1) 15

[CategoryDAO Class](#_heading=h.qzccfifraqu4) 15

[DButil Class](#_heading=h.2yh5pcpp87j) 15

[c. Sequence Diagram(s)](#_heading=h.12c7c8d4v7qx) 16

[d. Database queries](#_heading=h.9ohfixsomfjh) 16

[4. <SEARCH>](#_heading=h.ythbzmijcf08) 17

[a. Class Diagram](#_heading=h.att4245p9z6) 17

[b. Class Specifications](#_heading=h.71wbv0kk9630) 18

[Homecontroller Class](#_heading=h.rrzz2bmgufjl) 18

[GetProductAjax Class](#_heading=h.d01p7x5veyia) 18

[ProductDAO Class](#_heading=h.unzn2gji7d4w) 18

[DButil Class](#_heading=h.bi285624qua1) 18

[c. Sequence Diagram(s)](#_heading=h.l7isk4pgw0sn) 18

[d. Database queries](#_heading=h.ugiuqg9skb03) 19

[5. <PRODUCT DETAIL>](#_heading=h.y9utrbna2f8f) 20

[a. Class Diagram](#_heading=h.y7im5tlmnnfz) 20

[b. Class Specifications](#_heading=h.ov34mqwyt9pb) 21

[HomeController Class](#_heading=h.s5guyx3w75eg) 21

[ReviewDAO Class](#_heading=h.kh4xbdshl7lb) 21

[ReviewImageDAO Class](#_heading=h.ocyqsdcu4ubd) 21

[UserDAO Class](#_heading=h.7txwknw0ki9v) 21

[ProductDAO Class](#_heading=h.5crwgk4fcjcn) 21

[ProductImageDAO Class](#_heading=h.gdq5r8r9jkd6) 22

[DButil Class](#_heading=h.2damo85qt9oi) 22

[c. Sequence Diagram(s)](#_heading=h.r0pmr4vhjbrr) 22

[d. Database queries](#_heading=h.xnpxjy26z0rr) 22

[6. <BUY PRODUCT>](#_heading=h.nhu4r4x0pcnt) 23

[a. Class Diagram](#_heading=h.79esnzyau6e1) 24

[b. Class Specifications](#_heading=h.dvknd6m884wz) 25

[CartController Class](#_heading=h.ltntxx7mp3g9) 25

[OrderDAO Class](#_heading=h.lmtwk86l6z7k) 25

[OrderByShopDAO Class](#_heading=h.v8v339vossyd) 25

[OrderDetailDAO Class](#_heading=h.izm6g8gocqsc) 25

[ProductDAO Class](#_heading=h.zfmzpsatual) 25

[DButil Class](#_heading=h.15meeg1kpq4w) 26

[c. Sequence Diagram(s)](#_heading=h.18to5jj5937s) 26

[d. Database queries](#_heading=h.640hdz48ymua) 26

[7. <BANNER>](#_heading=h.5wwp37dyzu3s) 27

[a. Class Diagram](#_heading=h.6z1uhl17mg73) 28

[b. Class Specifications](#_heading=h.51d3i3puyjef) 28

[HomeController Class](#_heading=h.a095k3och175) 28

[BannerDAO Class](#_heading=h.xf6qnx5yq7if) 28

[DButil Class](#_heading=h.ppsssqvkdwbe) 28

[c. Sequence Diagram(s)](#_heading=h.xloiomvsojc6) 28

[d. Database queries](#_heading=h.b2schu960zzz) 29

[Name Class](#_heading=h.70jafg2i7dmg) 30

[Name Class](#_heading=h.zdvuqi2eft11) 32

[Name Class](#_heading=h.q1akuoynlkzd) 33

[Name Class](#_heading=h.1wbsa6cj8x9z) 34

[Name Class](#_heading=h.bh4cd7bz3ssw) 36

[Name Class](#_heading=h.y4aa7oxddych) 38

[Name Class](#_heading=h.incprfr7ty6s) 40

[Name Class](#_heading=h.3k25cf4i0j7o) 42

[**III. Database Tables**](#_heading=h.makz6f4bndqt) **43**

[1. < city >](#_heading=h.3whwml4) 43

[2. < district >](#_heading=h.2bn6wsx) 43

[3. < Ward >](#_heading=h.qsh70q) 43

[4. < Role >](#_heading=h.3as4poj) 43

[5. < user >](#_heading=h.9anizzhfgka2) 43

[6. < category >](#_heading=h.49x2ik5) 44

[7. < delivery\_method >](#_heading=h.2p2csry) 44

[8. < payment\_method >](#_heading=h.147n2zr) 44

[9. < product >](#_heading=h.3o7alnk) 44

[10. < image\_product >](#_heading=h.j1ek9cvlo5k7) 45

[11. < order >](#_heading=h.ihv636) 45

[12. < order\_detail>](#_heading=h.a2quyqlasyav) 45

[13. < order\_by\_shop>](#_heading=h.huhbghm41to9) 47

[14. < review >](#_heading=h.qmdv8rxkem3m) 47

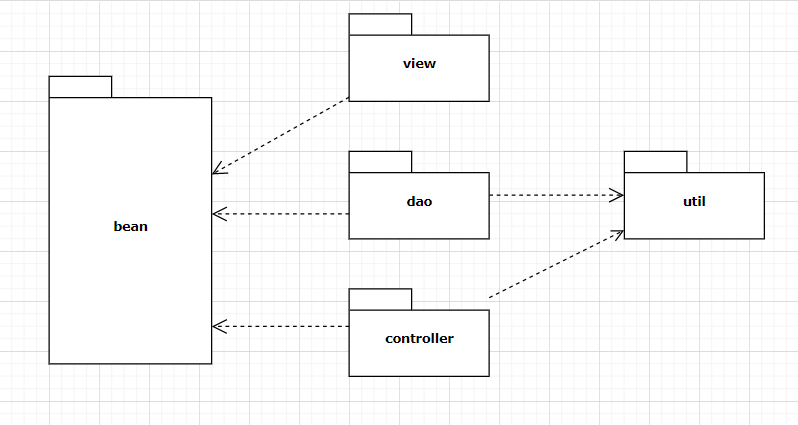
[15. < image\_review >](#_heading=h.2grqrue) 47

[16. < banner >](#_heading=h.vx1227) 47

# 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***

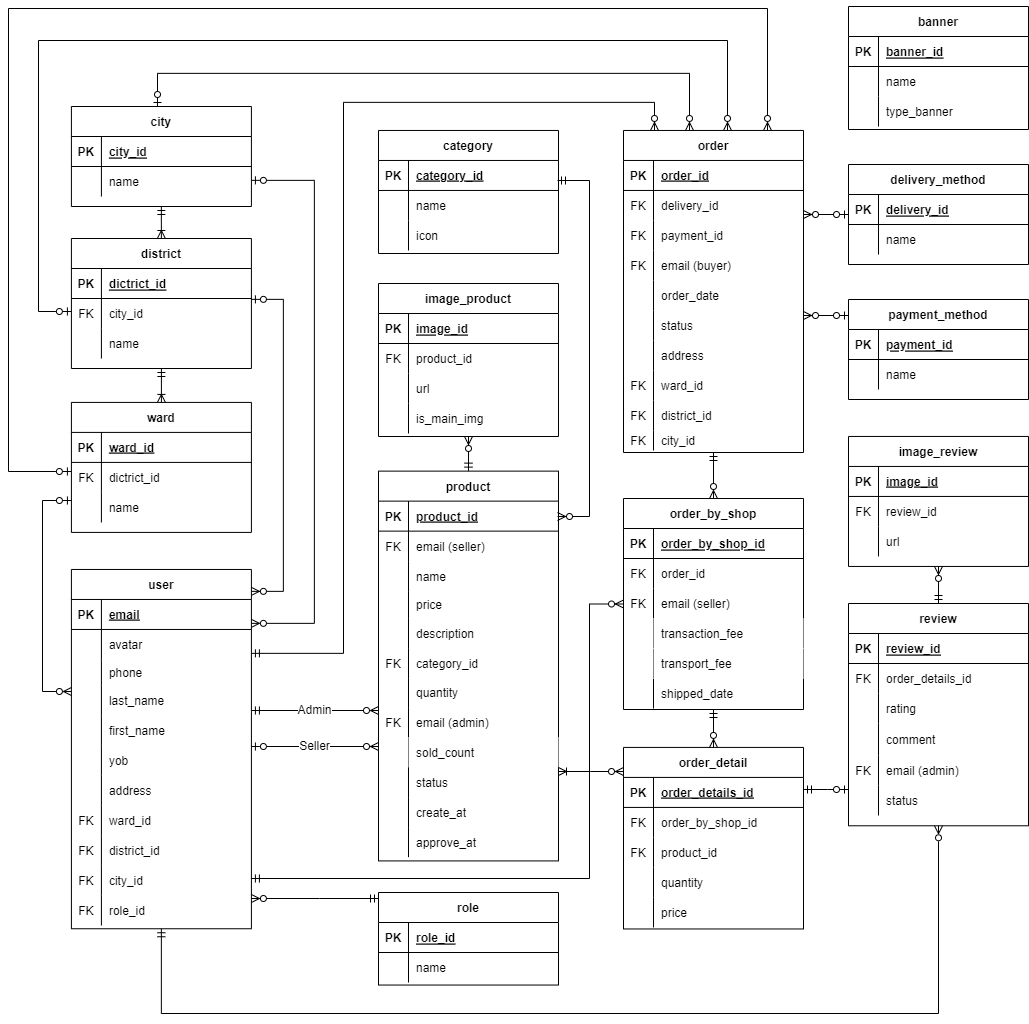
| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | bean | * Entity class * Classname + “DTO” (ex: ProductDTO.java) |
| 02 | view | * Is used for holds view for user and admin * lowerCase (ex:mainpage.jsp) |
| 03 | controller | * Is used for holds web controllers in MVC 2 models * Classname + “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) |

# 

# 

# 

# 

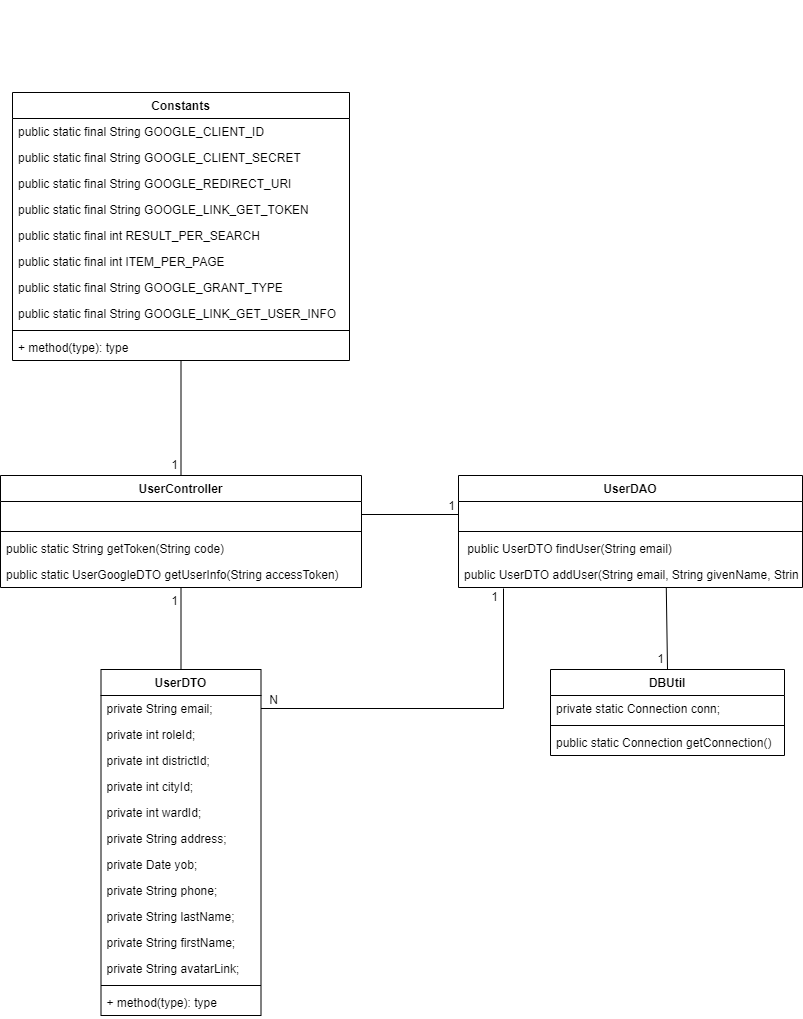
# 

# 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* |

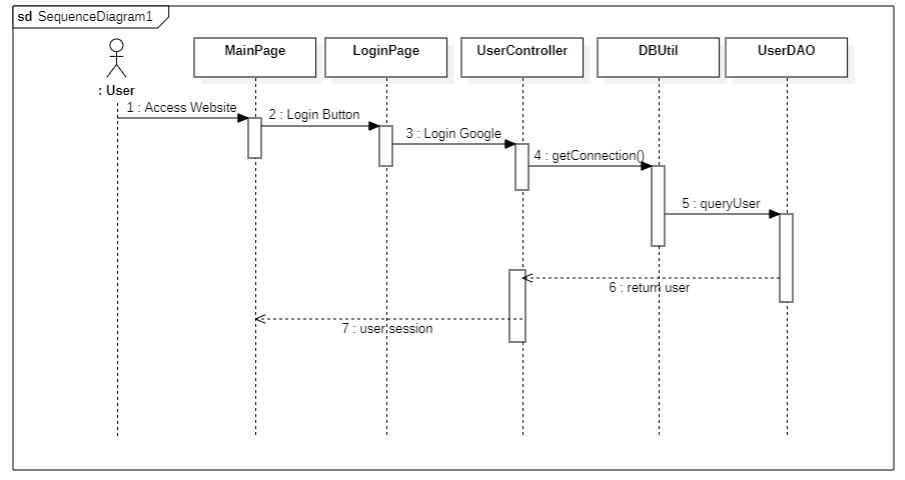
#### 

#### 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* |

#### 

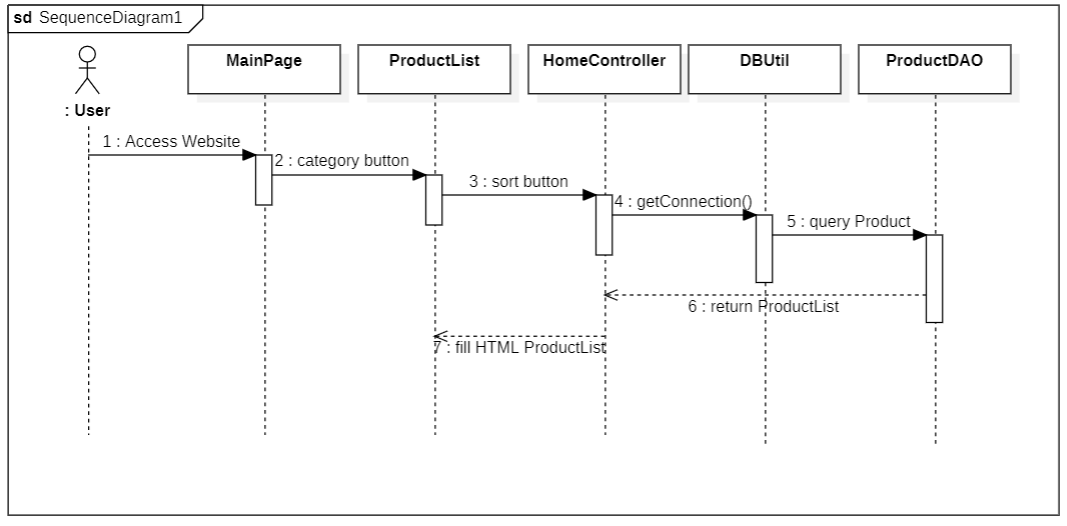
#### DButil Class

| **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:

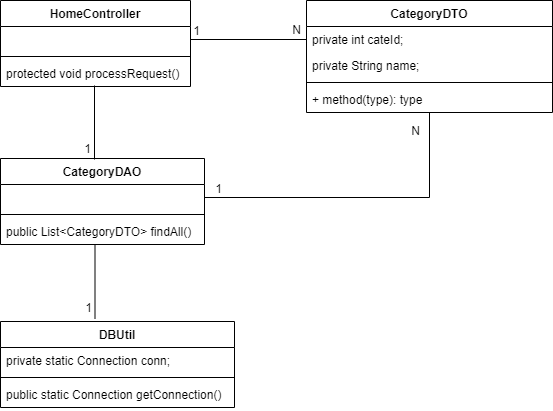
SELECT [image\_id], [url], [is\_main\_img] FROM image\_product WHERE product\_id = ?

## 

## 

## 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* |

#### 

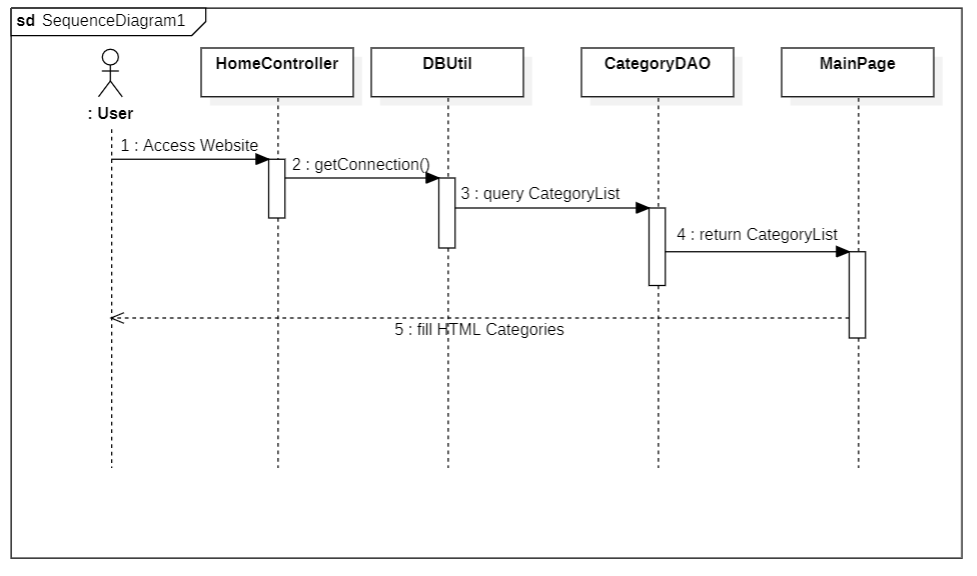
#### DButil Class

| **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*

**

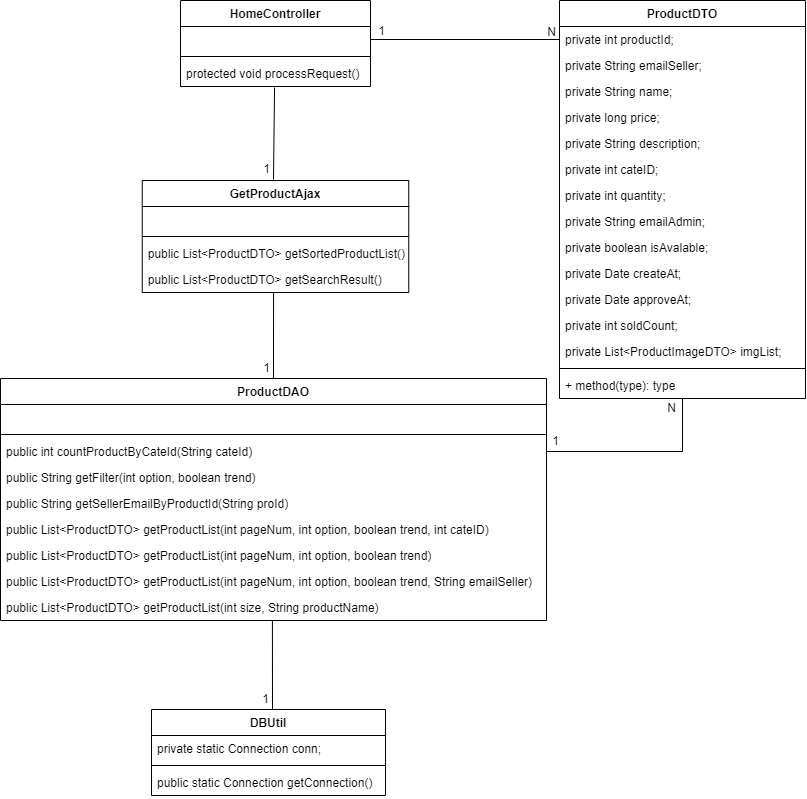
### d. Database queries

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* |

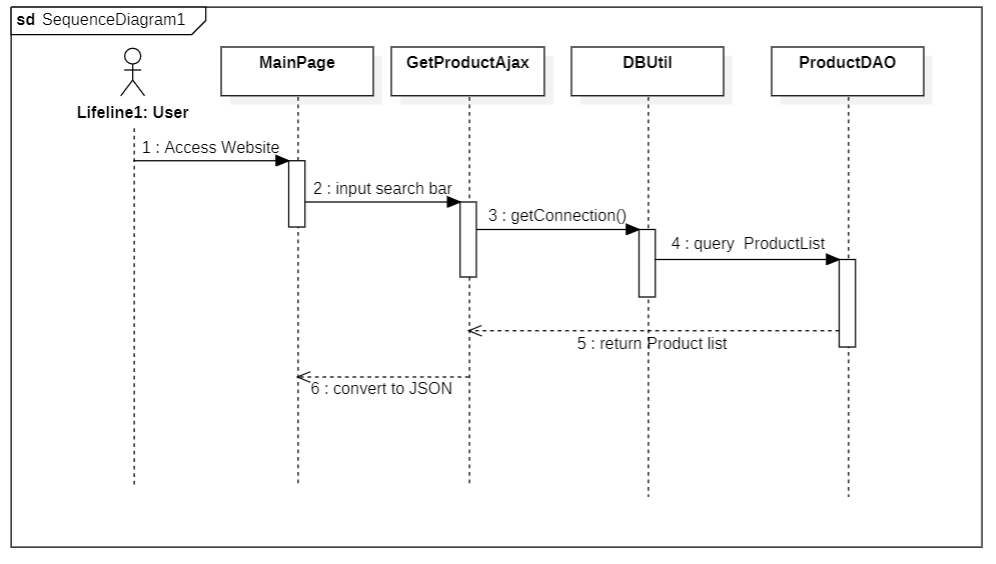
#### 

#### 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

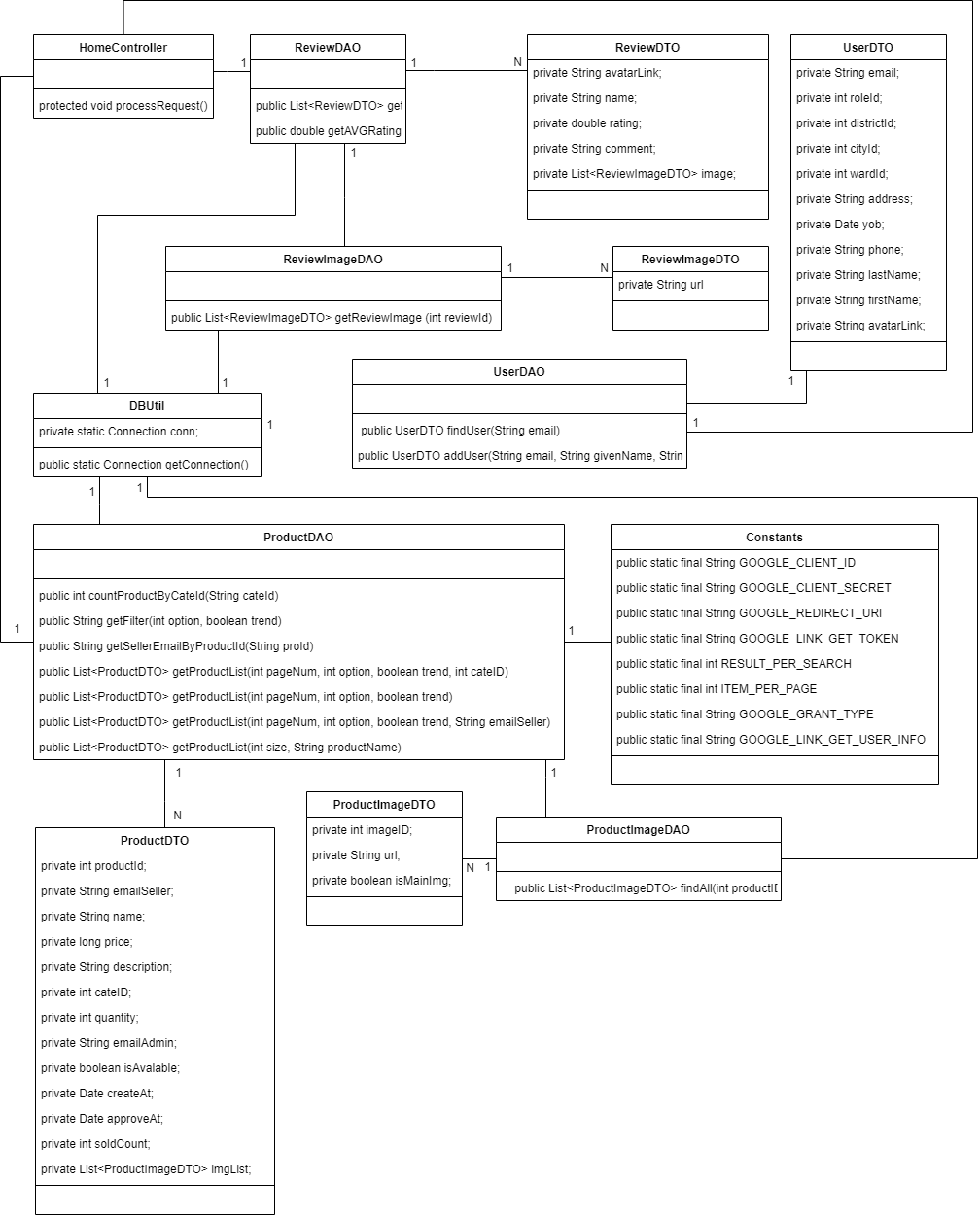
WHERE name LIKE ?

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* |

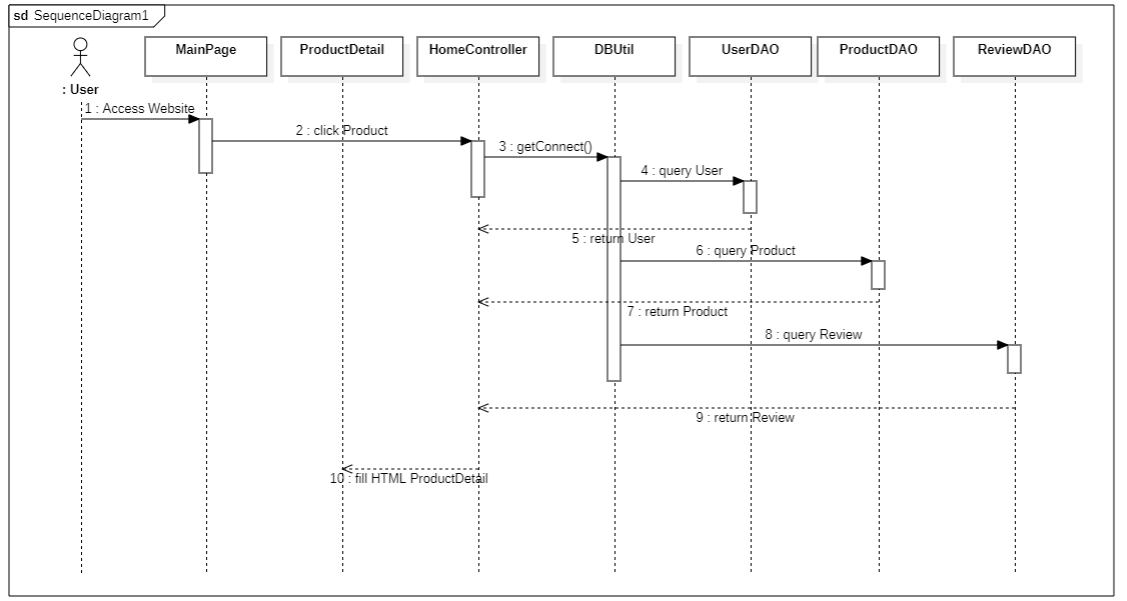
#### 

#### 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

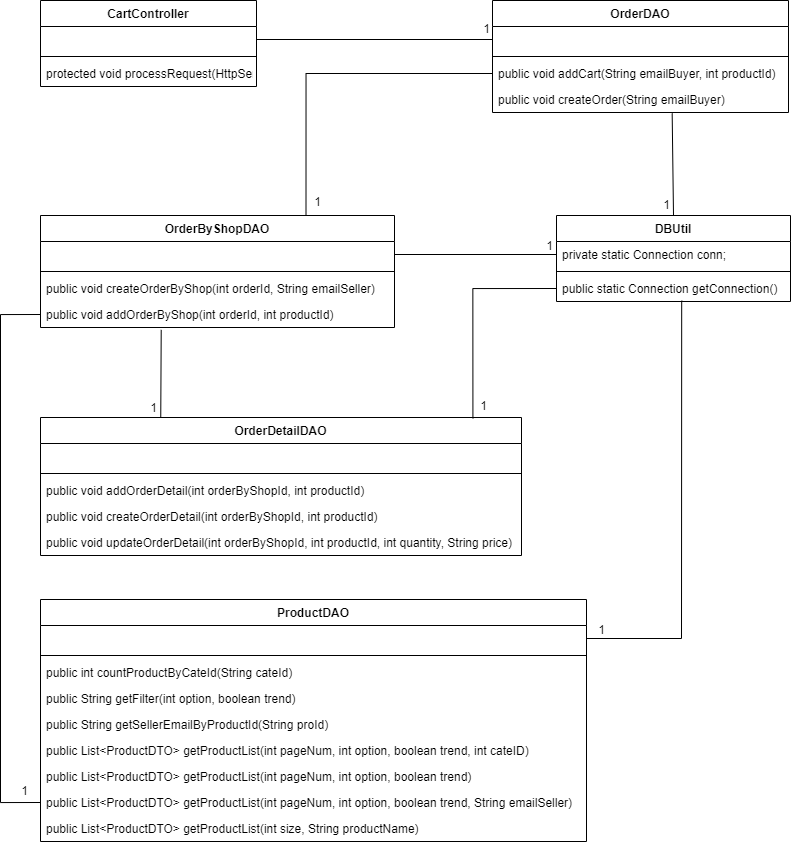
WHERE name LIKE ? ORDER BY name OFFSET 0 rows FETCH first ? rows only

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

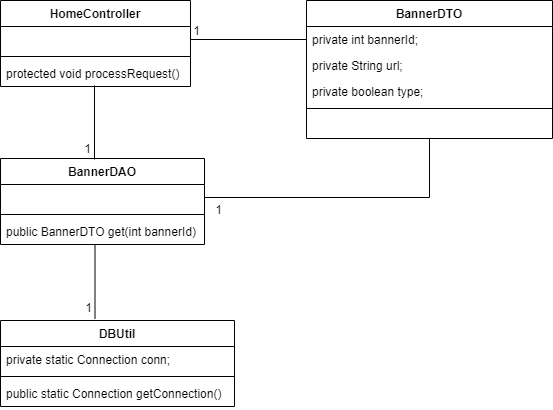
SELECT AVG(r.rating)

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* |

### 

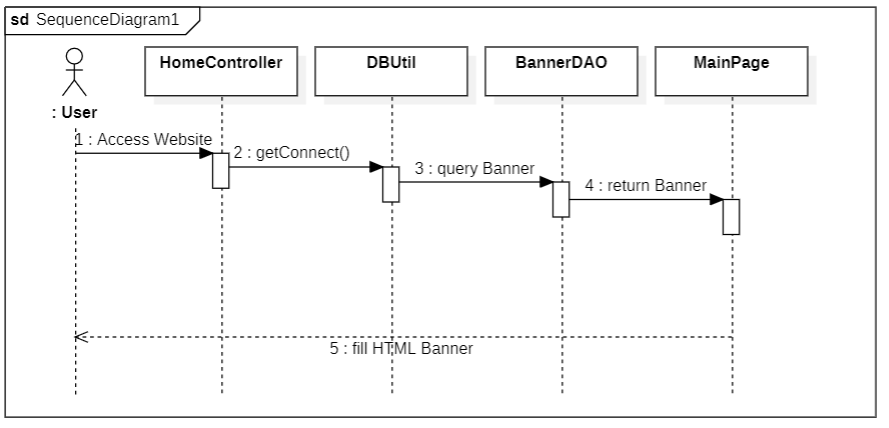
#### DButil Class

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

### 

### c. Sequence Diagram(s)

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

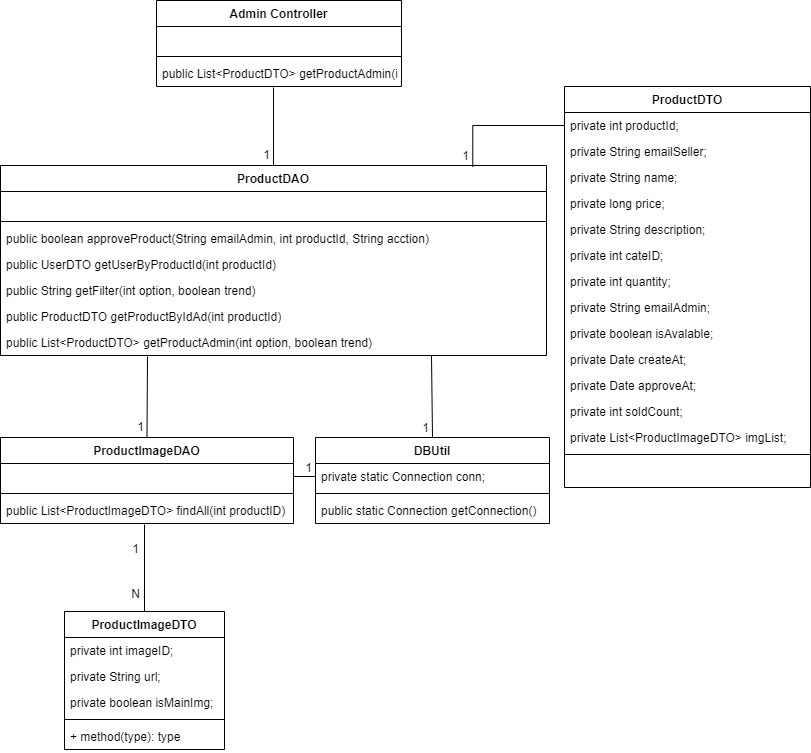
**

### 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()* |  |

#### 

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean approveProduct(String emailAdmin, int productId, String action)* |  |
| *02* | *public String getFilter(int option, boolean trend)* |  |
| *03* | *public ProductDTO getProductByIdAd(int productId)* |  |
| *04* | *public List<ProductDTO> getProductAdmin(int option, boolean trend)* |  |

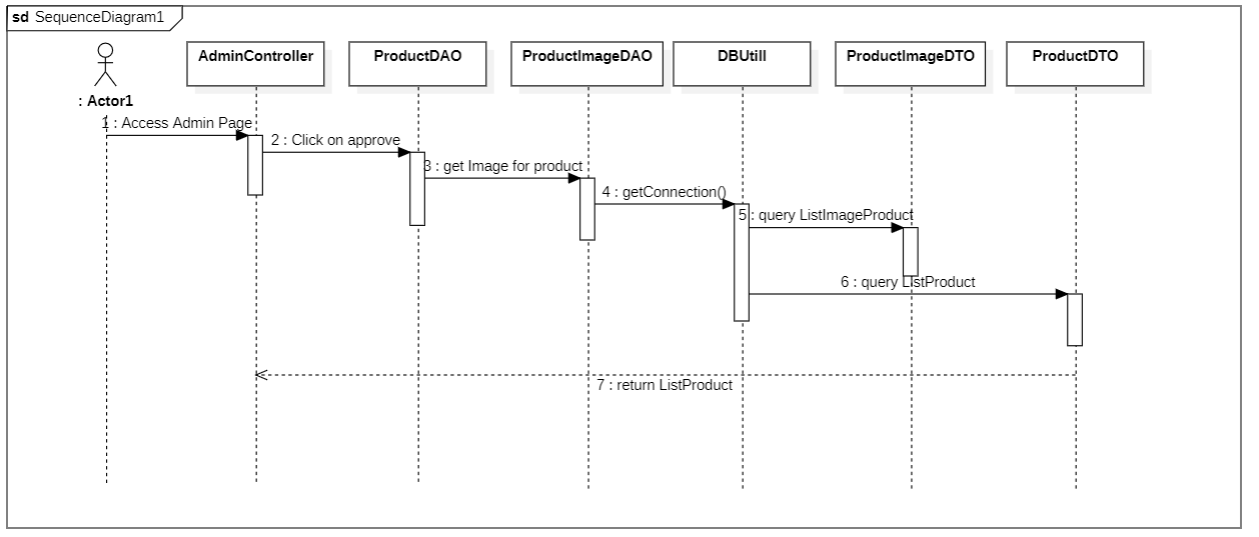
#### 

#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* |  |

#### 

**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

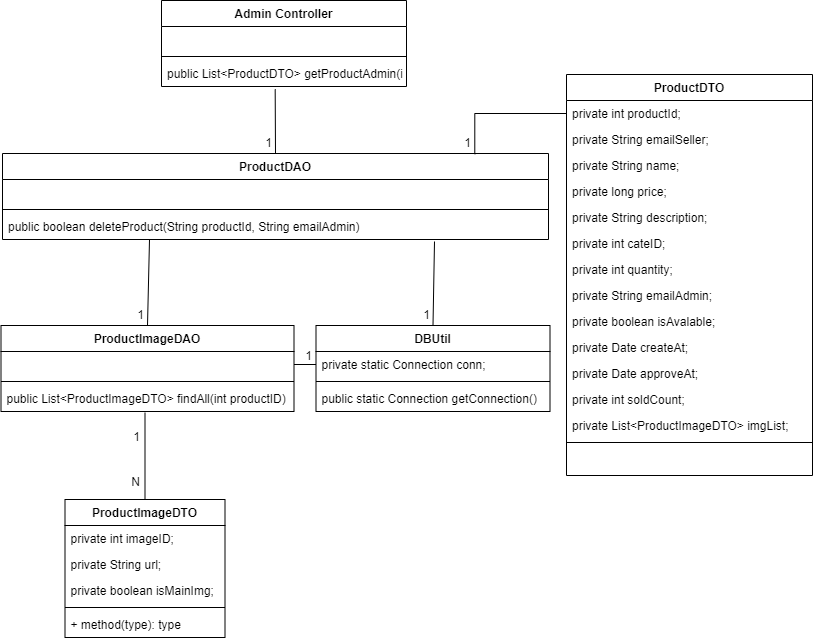
SELECT [product\_id], [email\_seller], [name], [price], [description], [category\_id], [quantity], [create\_at] FROM product WHERE email\_admin is null AND status is null

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()* |  |

#### ProductDAO Class

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

#### 

#### ProductImgaeDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductImageDTO> findAll(int productID)* |  |

#### 

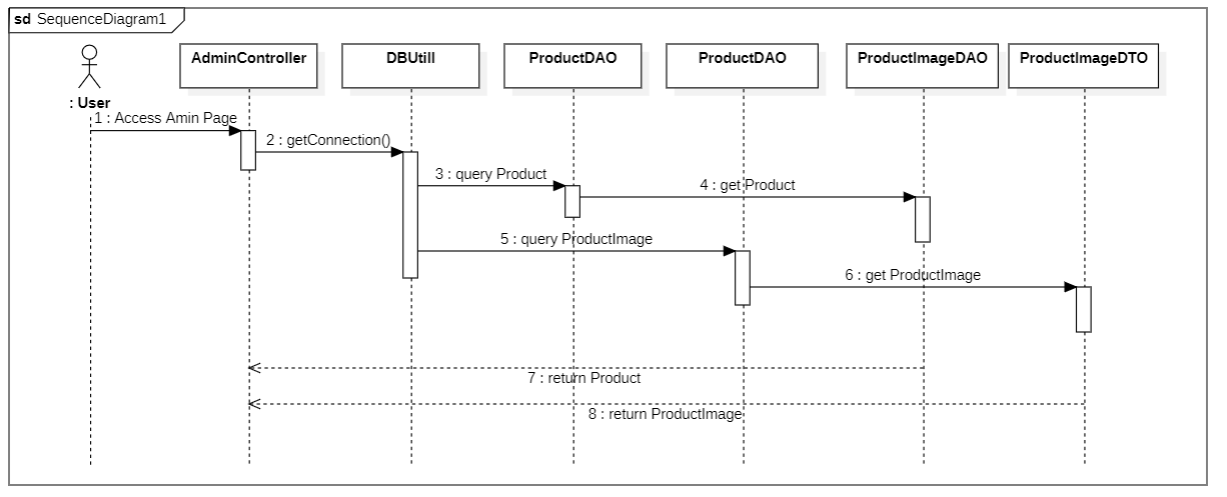
#### 

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### 

**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>**

**a. Class Diagram**

**b. Class Specifications**

#### Name Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* |  |  |

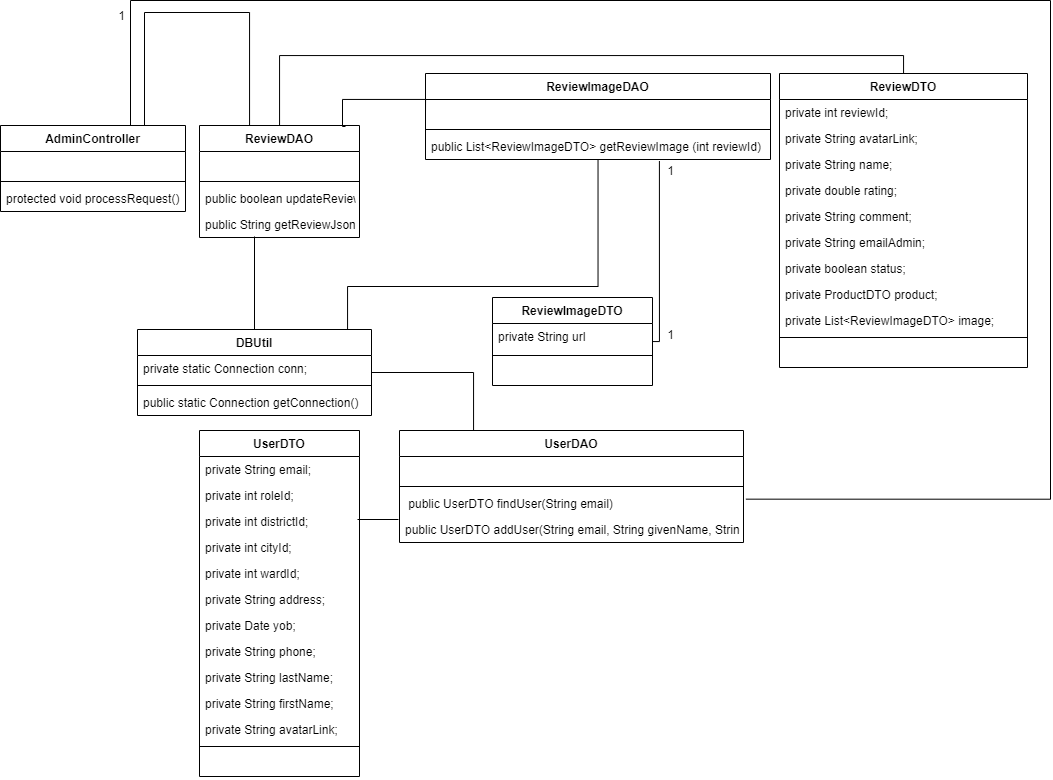
#### 

**c. Sequence Diagram(s)**

**d. Database queries**

**11. <REVIEW AUTHENTICATION>**

**a. Class Diagram**

****

**b. Class Specifications**

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* |  |

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateReview()* |  |
| *02* | *public String getReviewJson()* |  |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### 

#### ReviewImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewImageDTO> getReviewImage (int reviewId)* |  |

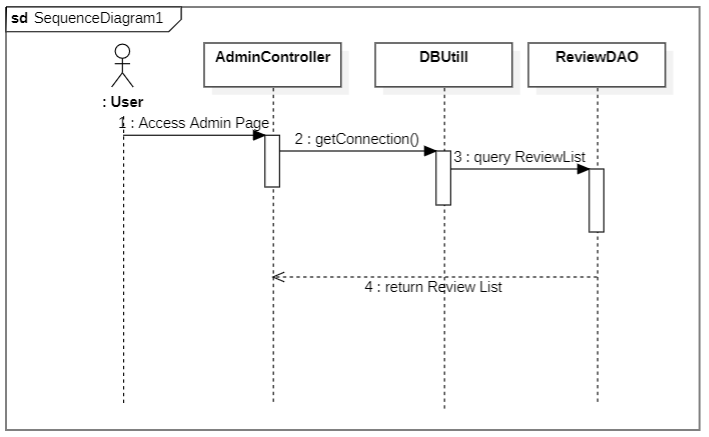
#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* |  |
| *02* | *public UserDTO addUser(String email, String givenName, String avatarLink)* |  |

#### 

**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 = ?

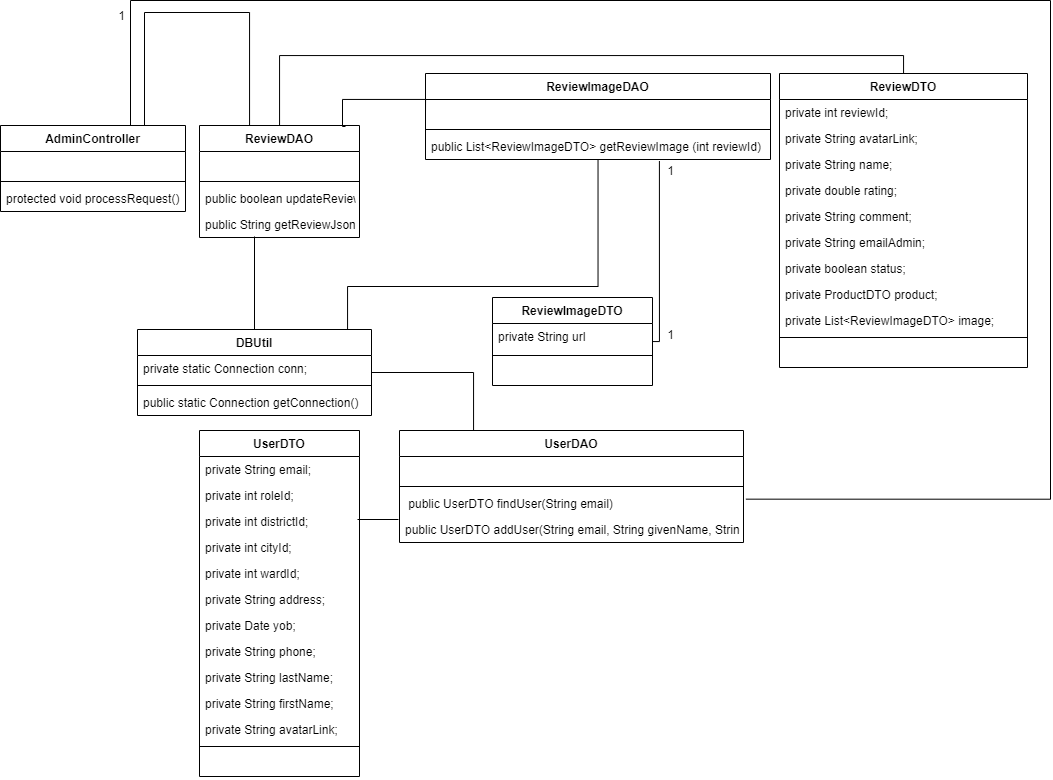
insert into [user] (email, avatar, last\_name, first\_name, role\_id) values (?, ?, ?, ?, 0)

ProdutImageDAO:

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

**12. <DELETE REVIEW ON WEBSITE>**

**a. Class Diagram**

****

**b. Class Specifications**

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *protected void processRequest()* |  |

#### 

#### ReviewDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean updateReview()* |  |
| *02* | *public String getReviewJson()* |  |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### 

#### ReviewImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ReviewImageDTO> getReviewImage (int reviewId)* |  |

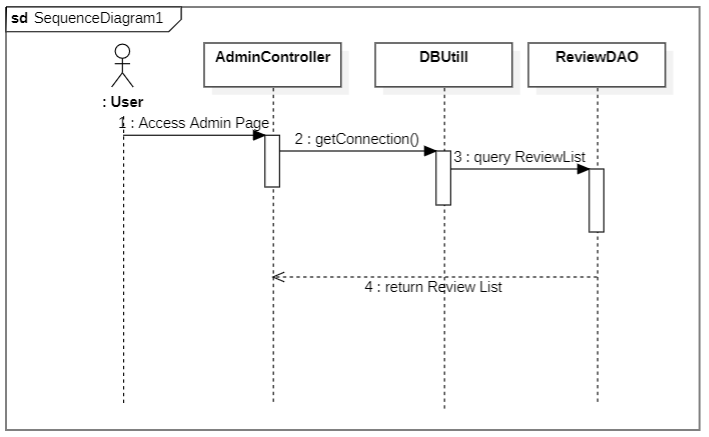
#### 

#### UserDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public UserDTO findUser(String email)* |  |
| *02* | *public UserDTO addUser(String email, String givenName, String avatarLink)* |  |

#### 

**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

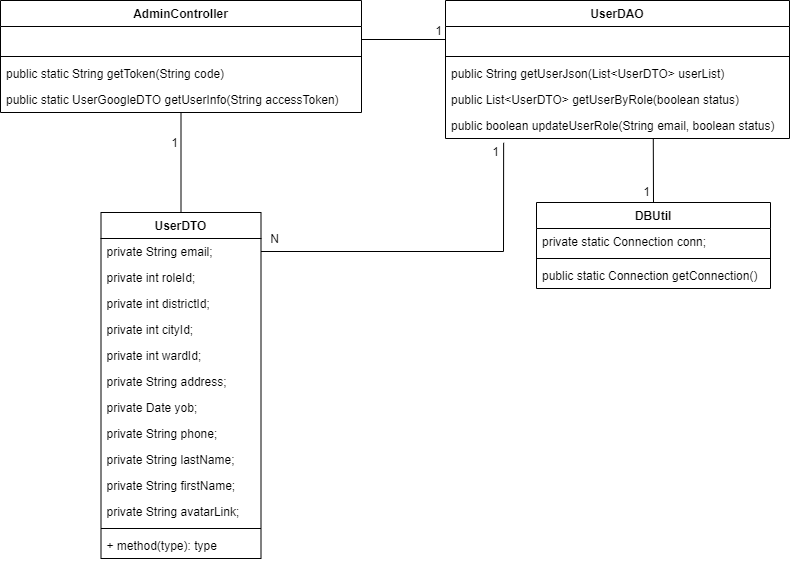
UserDAO:

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 String getToken(String code)* |  |
| *02* | *public static UserGoogleDTO getUserInfo(String accessToken)* |  |

#### UserDAO Class

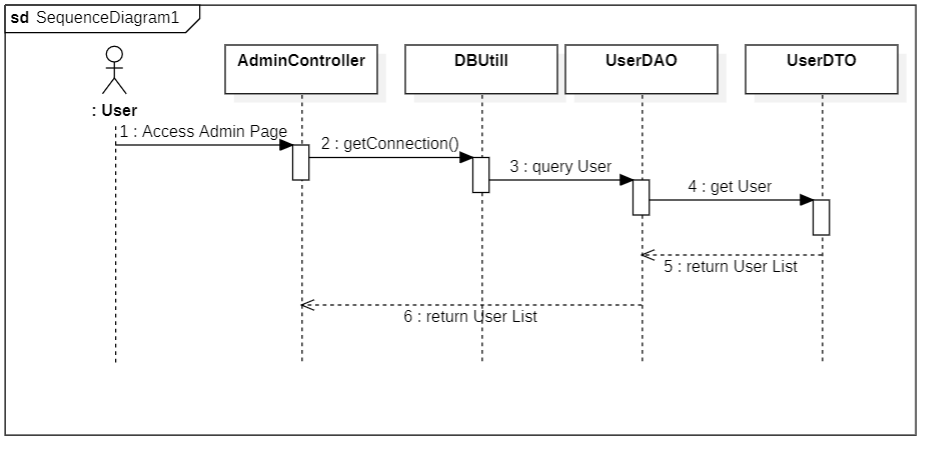
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public String getUserJson(List<UserDTO> userList)* |  |
| *02* | *public List<UserDTO> getUserByRole(boolean status)* |  |
| *03* | *public boolean updateUserRole(String email, boolean status)* |  |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### 

**c. Sequence Diagram(s)**

****

**d. Database queries**

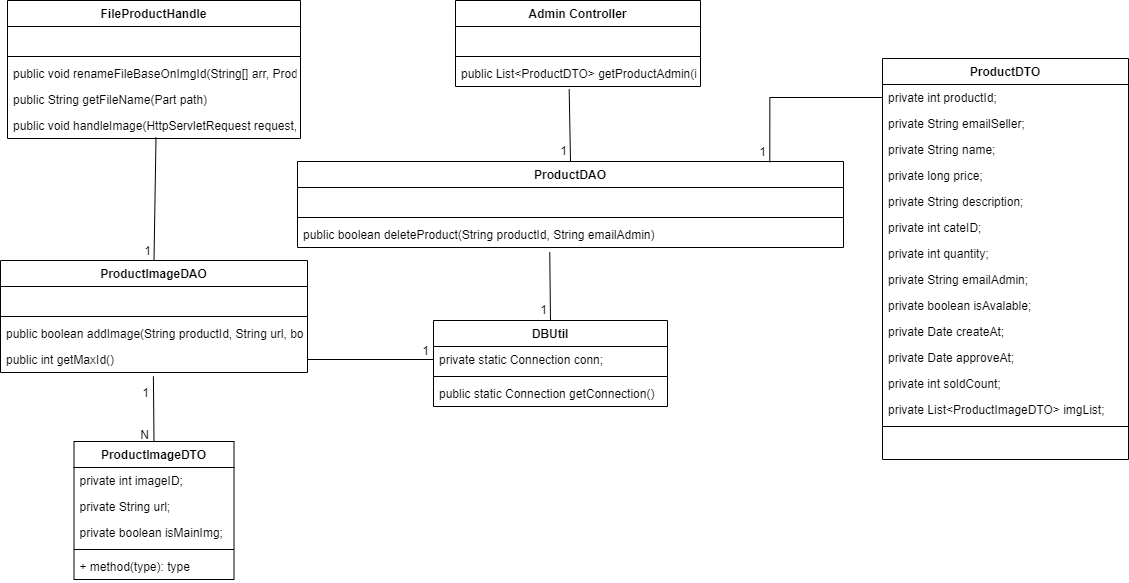
UserDAO:

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>**

**a. Class Diagram**

****

**b. Class Specifications**

#### AdminController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public List<ProductDTO> getProductAdmin()* |  |

#### ProductDAO Class

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

#### ProductImageDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public boolean addImage(String productId)* |  |
| *02* | *public int getMaxId()* |  |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### 

**c. Sequence Diagram(s)**

**d. Database queries**

ProductDAO:

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

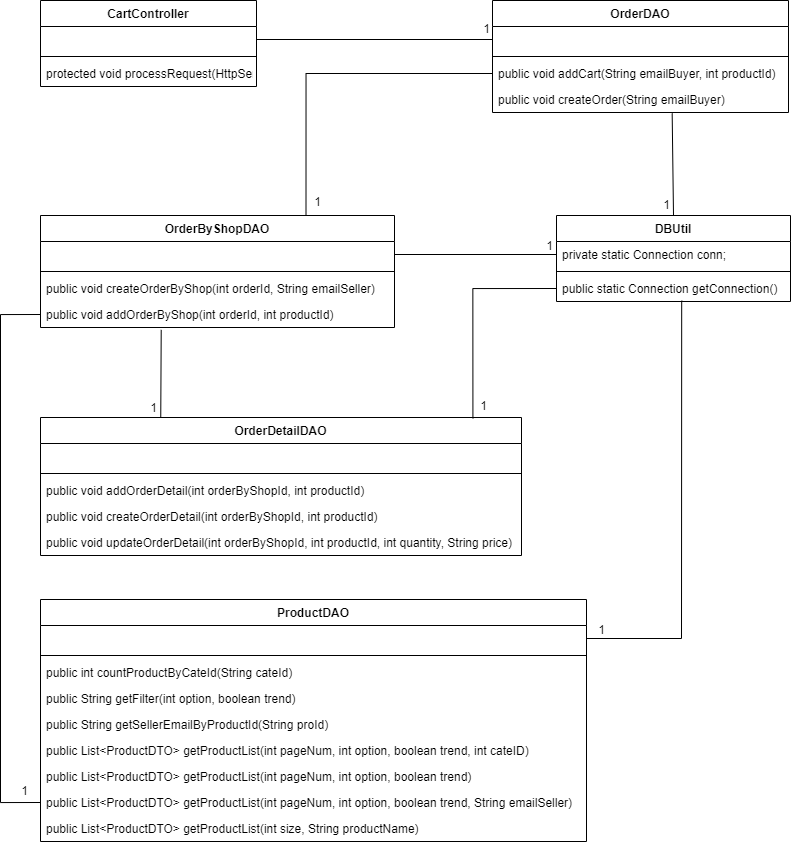
ProductImageDAO:

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)* |  |

#### 

#### OrderDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void addCart(String emailBuyer, int productId)* |  |
| *02* | *public void createOrder(String emailBuyer)* |  |

#### OrderByShopDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void createOrderByShop(int orderId, String emailSeller)* |  |
| *02* | *public void addOrderByShop(int orderId, int productId)* |  |

#### DBUtil Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public static Connection getConnection()* |  |

#### OrderDetailDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public void addOrderDetail(int orderByShopId, int productId)* |  |
| *02* | *public void createOrderDetail(int orderByShopId, int productId)* |  |
| *03* | *public void updateOrderDetail(int orderByShopId, int productId, int quantity, String price)* |  |

#### ProductDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *public int countProductByCateId(String cateId)* |  |
| *02* | *public String getFilter(int option, boolean trend)* |  |
| *03* | *public String getSellerEmailByProductId(int productId)* |  |
| *04* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend)* |  |
| *05* | *public List<ProductDTO> getProductList(int pageNum, int option, boolean trend, int cateID)* |  |
| *06* | *public List<ProductDTO> getProductList(int size, String productName)* |  |
| *07* | *public List<ProductDTO> getProductList(int pageNum, int item\_per\_page, int option,*  *boolean trend, String emailSeller)* |  |
| *08* | *public List<ProductDTO> getProductList(int pageNum, int item\_per\_page, int option, boolean trend, int cateID)* |  |

#### 

**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>**

**a. Class Diagram**

**b. Class Specifications**

#### Name Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* |  |  |

#### 

**c. Sequence Diagram(s)**

**d. Database queries**

# 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) |