

**LAPTOP SHOP**

**Software Design Specification**

– Hanoi, August 2022 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
| 15/05 | A | longnd | Create new RDS |
| 17/05 | A | longnd | Class diagram and specification, sequence diagram and database queries for user login function, home function |
| 17/05 | A | thangbq | Class diagram and specification, sequence diagram and database queries for blog list, blog details, product details function |
| 08/06 | A | thangbq | Class diagram and specification, sequence diagram and database queries for sales order list function |
| 08/06 | A | longnd | Class diagram and specification, sequence diagram and database queries for user authorization |
| 15/06 | A | thangbq | Class diagram and specification, sequence diagram and database queries for sales orderdetails function |
| 15/06 | A | longnd | Class diagram and specification, sequence diagram and database queries for cart details, cart information |
| 02/07 | A | longnd | Class diagram and specification, sequence diagram and database queries for product list |
| 02/07 | A | thangbq | Class diagram and specification, sequence diagram and database queries for order completion |
| 15/05 | A | longnd | Create new RDS |
| 17/05 | A | longnd | Class diagram and specification, sequence diagram and database queries for user login function, home function |
| 17/05 | A | thangbq | Class diagram and specification, sequence diagram and database queries for blog list, blog details, product details function |
| 08/06 | A | thangbq | Class diagram and specification, sequence diagram and database queries for sales order list function |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[**I. Overview**](#_heading=h.gjdgxs) **5**

[1. Code Packages](#_heading=h.30j0zll) 5

[2. Database Design](#_heading=h.1fob9te) 6

[a. Database Schema](#_heading=h.3znysh7) 6

[b. Table Descriptions](#_heading=h.lnxbz9) 6

[**II. Code Designs**](#_heading=h.tyjcwt) **8**

[1,User Login](#_heading=h.du5dztlkmfth) 8

[1. Class diagram](#_heading=h.h5t2mhi8biwy) 8

[2. Class specification](#_heading=h.m8q6ifnwezmr) 8

[3. Sequence diagram](#_heading=h.hmyb585fpn9t) 11

[4. SQL Script:](#_heading=h.4gdbq919celx) 11

[2,Home Pages](#_heading=h.f6g5zk5fwjgg) 11

[Class diagram](#_heading=h.8d3w4eeeyxpk) 11

[Class specification](#_heading=h.4kg7y8ozu0i7) 12

[Sequence Diagram](#_heading=h.xx80glmho8nr) 16

[SQL Script:](#_heading=h.f01lvrsew6rl) 16

[3, ProductsDetails](#_heading=h.dm7tvgh6xf0z) 16

[1, Class diagram](#_heading=h.o6e2ow7g8kqm) 16

[2. Class specification](#_heading=h.5mwm6al7tqlq) 16

[3. Sequence Diagram](#_heading=h.isj7344vm09k) 18

[SQL Query:](#_heading=h.ulc5b8cvo3ed) 18

[4, Login](#_heading=h.a0eizgexyh8w) 18

[1, Class diagram](#_heading=h.cxlb6ozhm974) 18

[2, Class Specification](#_heading=h.snz0gfbhwg5s) 18

[3. Sequence diagram](#_heading=h.pumhcbhn7l1z) 20

[4. SQL Script:](#_heading=h.rtqostwo6m83) 20

[5, BlogDetails](#_heading=h.4z2vibmpj7jr) 21

[1, Class diagram](#_heading=h.sygxj6lukwhr) 21

[2. Class specification](#_heading=h.tpk8rnwkcc2d) 21

[3. Sequence Diagram](#_heading=h.962jm3b17ju5) 23

[4, SQL Script](#_heading=h.tvoln3ovqz3t) 23

[6, BlogLists](#_heading=h.f2pws5az7rg) 24

[1, Class diagram](#_heading=h.hfq04c7m5bmn) 24

[2. Class specification](#_heading=h.fmdsyzeipaw8) 24

[3. Sequence Diagram](#_heading=h.pijpa6leh0h) 26

[4, SQL Script](#_heading=h.9l67ws47lezu) 26

[7, User Register](#_heading=h.pzcp8hhz6ruq) 27

[1, Class diagram](#_heading=h.v882etj5px4q) 27

[2. Class specification](#_heading=h.4sqi8lmuh4uh) 27

[3. Sequence Diagram](#_heading=h.po1r3dwk8ab1) 31

[4. SQL Script](#_heading=h.wna662tr2svw) 31

[8. Order List:](#_heading=h.1vva6uqlvqyk) 32

[1. Class Diagram](#_heading=h.37bgh42ys9ht) 32

[2. Class Specification](#_heading=h.gutua2y2rhum) 32

[3. Sequences Diagram](#_heading=h.55ww0lm7iurr) 35

[SQL Query](#_heading=h.h9zln6r73liw) 35

[9. User Authorization](#_heading=h.6zva9mprgzx3) 36

[1. Class diagram](#_heading=h.e8efvy3a9kd) 36

[2. Class specification](#_heading=h.smcme2uhhl4m) 36

[3. Sequence Diagram](#_heading=h.i8r9e144qiok) 40

[SQL Script](#_heading=h.d8q49920hv3q) 42

[10. Sales Dashboard](#_heading=h.3nja9pd7gngo) 43

[1, Class diagram](#_heading=h.enbk51vlqt2m) 43

[2. Class specification](#_heading=h.lbcwn6q3uuam) 43

[3. Sequences Diagram](#_heading=h.5xf5nis8ajb2) 46

[4.SQL Script](#_heading=h.4eutdmf0s3v5) 46

[11. Order Details](#_heading=h.o87cw1r1g5ai) 47

[1, Class Diagram](#_heading=h.43ru4bf0qra0) 47

[2, Class Specification](#_heading=h.7sq5ni8u2tz7) 47

[3. Sequences Diagram](#_heading=h.umejiilm3isc) 50

[4. SQL Script](#_heading=h.vexlomx3t0s2) 50

[12. Cart Completion](#_heading=h.i7tikonifzjb) 51

[1. Class Diagram](#_heading=h.nno7vdtvg0yc) 51

[2. Class Specification](#_heading=h.7c7pplo3fiu0) 51

[3. Sequences Diagram](#_heading=h.4u3qagzjxku) 56

[4. SQL Query](#_heading=h.c1npe7at0vm0) 56

[13. Cart Information](#_heading=h.ygz8f8v4dy3b) 57

[1. Class diagram](#_heading=h.sdpgkj6e4pm3) 57

[2. Class Specification](#_heading=h.as8gomoielx9) 57

[3. Sequences Diagram](#_heading=h.b9i3wi9e2may) 61

[4. Sql Query](#_heading=h.eizez3ffyaer) 61

[14, Cart Details](#_heading=h.ibksig9ljboy) 62

[1. Class Diagram](#_heading=h.77ypyk9ghqb2) 62

[2. Class Specification](#_heading=h.63eywx7ebc1r) 62

[3. Sequences Diagram](#_heading=h.ypauh924aaed) 65

[4. SQL Script](#_heading=h.8s07pt4kgfhl) 65

[15. MyOrder](#_heading=h.z458y28d2jc) 66

[1. Class Diagram](#_heading=h.rxlxa38zyoth) 66

[2. Class Specification](#_heading=h.ntuusalge1gu) 66

[3. Sequences Diagram](#_heading=h.mxal8a9wxsgu) 69

[4. SQLQuery](#_heading=h.3mpul6xud4po) 69

[16. Order Information](#_heading=h.copdr87672v) 70

[1. Class Diagram](#_heading=h.9a0203tba4ef) 70

[2. Class Specification](#_heading=h.9bbtyvq3km21) 70

[3. Sequences Diagram](#_heading=h.kzx7nq7lzh3d) 75

[4. SQL Query](#_heading=h.r3eztotspwx3) 76

[17. Feedback](#_heading=h.qnjix618t6md) 76

[1. Class Diagram](#_heading=h.f9vy6z4ge143) 76

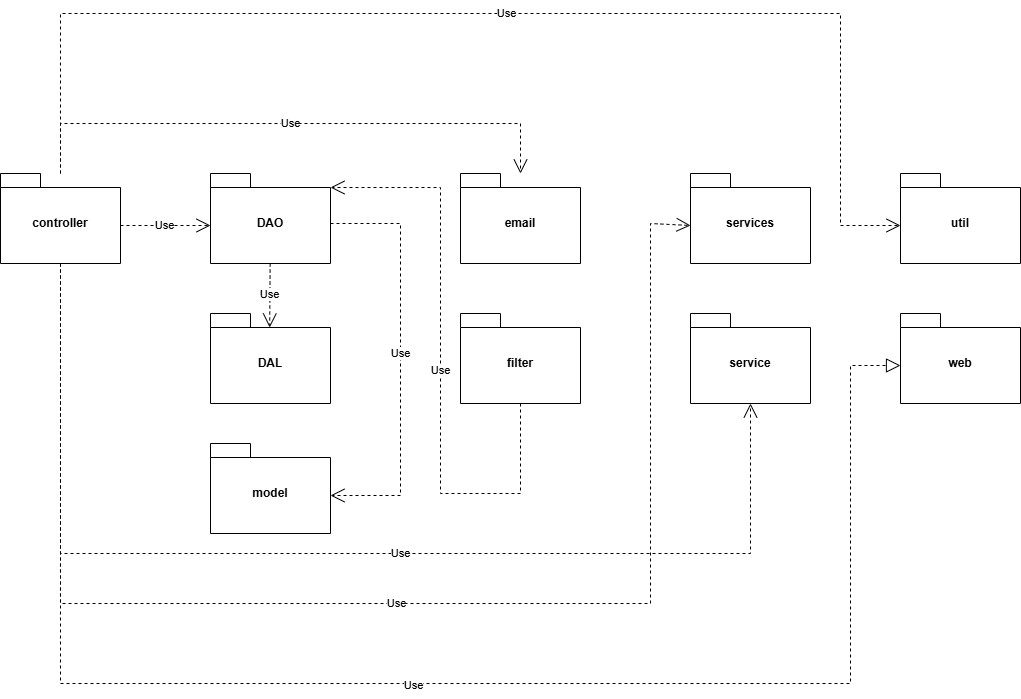
[2. Class Specification](#_heading=h.p9sonkudkif5) 76

[3. Sequences Diagram](#_heading=h.ohd4egnk6gu) 83

[4. SQL Query](#_heading=h.mav4ibfefp3t) 83

# I. Overview

## 1. Code Packages

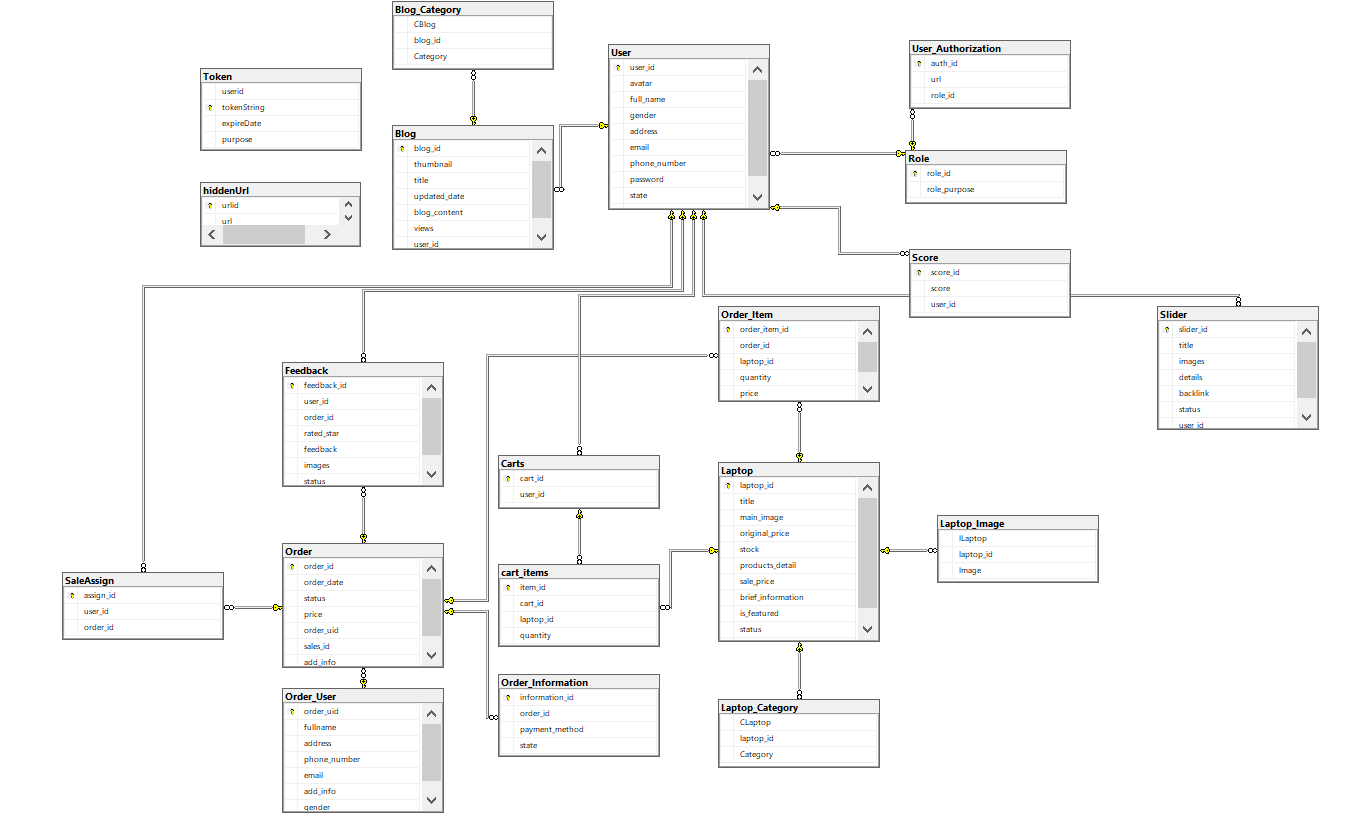


***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | controller | Contained all the servlet for the project |
| 02 | DAO | Contained Database Access Object based on the database table |
| 03 | DAL | Contains the DAL(Database Access Layer), which help the DAO to access database |
| 04 | filter | Contains filter, which run every time user visit certain url |
| 05 | model | Contains model class which used to get the data from the database or add then in |
| 06 | email | Contains class to send mail |
| 07 | service | Contain classes that help with direct problems |
| 08 | services | Contain classes that help with direct problems |
| 09 | util | Contain classes configurations |
| 10 | web | the front-end for the webpage |

## 2. Database Design

### a. Database Schema



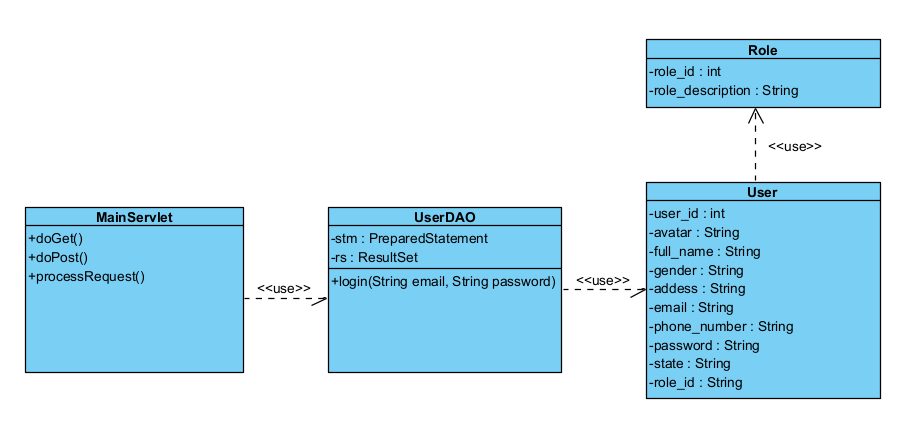
#### b. Table Descriptions

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | User | This table manage the user’s status  - Primary keys: **user\_id**  - Foreign keys: **role\_id** from **Role** |
| 02 | Role | This table manage the role from the project  - Primary keys: **role\_id** |
| 03 | Carts | This table manage the cart of a logged-in user, instead of using cookie  - Primary keys: **cart\_id** |
| 04 | cart\_items | This table manage the items of each logged-in cart  - Primary keys: **items\_id**  - Foreign keys: **cart\_id** from **Cart** |
| 05 | Order | This table manage the order(s) from the project  - Primary keys: **order\_id** |
| 06 | OrderItems | This table manage the items from each order  - Primary keys: **order\_item\_id**  - Foreign keys: **order\_id** from **Order** |
| 07 | OrderUser | This table manage the user who ordered said product  - Primary keys: **order\_uid**  - Foreign keys: **order\_id** from **Order** |
| 08 | Order Information | This table manage information from each product, including payment method  - Primary keys: **information\_id**  - Foreign keys: **order\_id** from **Order** |
| 09 | SaleAssign | This table saves the state of who assign who to which order  - Primary keys: **assign\_id**  - Foreign keys: + **order\_id** from **Order**  **+ user\_id** from **User** |
| 10 | Feedback | This table save the feedback for a product/for all service  - Primary keys: **feedback\_id**  - Foreign keys: + **order\_id** from **Order**  **+ user\_id** from **User** |
| 11 | Laptop | This table save the laptop item  - Primary keys: **laptop\_id** |
| 12 | Laptop\_Image | This table save the image for each laptop  - Primary keys: **ILaptop**  - Foreign keys: + **laptop\_id** from **Laptop** |
| 13 | Laptop\_Category | This table save the category for each laptop  - Primary keys: **CLaptop**  - Foreign keys: + **laptop\_id** from **Laptop** |
| 14 | Blog | This table save the list of blog  - Primary keys: **blog\_id** |
| 15 | Blog\_Category | This table save the category for each blog  - Primary keys: **CBlog**  - Foreign keys: + **blog\_id** from **Blog** |
| 16 | Slider | This table saves the slider  - Primary keys: **slider\_id**  - Foreign keys: + **user\_id** from **User** |
| 17 | Score | This table save each user’s score  - Primary keys: **score\_id**  - Foreign keys: + **user\_id** from **User** |
| 18 | Token | This table save token incase user forgot password  - Primary keys: **tokenString** |
| 19 | hiddenUrl | This table manage the url that should be hidden when showing the list of accessible url  - Primary keys: **urlid** |
| 20 | User\_Authorization | This table manage which user can access which url  - Primary keys: **authId**  - Foreign keys: **+ role\_id** from **Role** |

# II. Code Designs

## 1,User Login

#### Class diagram



#### Class specification

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

***Role Class***

| *1* | getRole\_Id() | get a value for the role\_id attribute |
| --- | --- | --- |
| *2* | getRoleDescription() | get a value for the role description attribute |
| *3* | setRole\_Id() | assign a value for the role\_id attribute |
| *4* | setRoleDescription() | assign a value for the role description attribute |

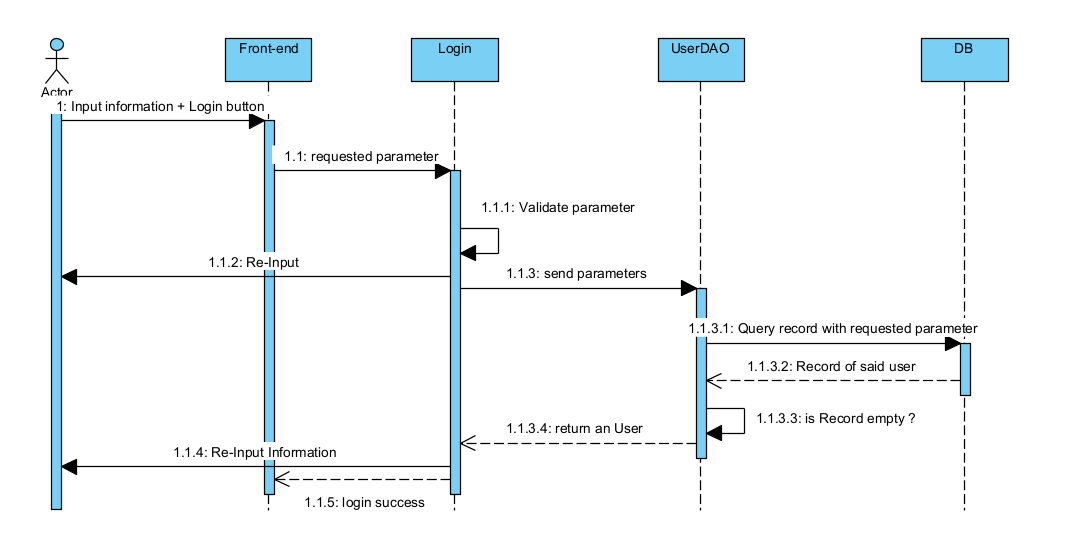
***LoginServlet Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet(HttpServletRequest request, HttpServletResponse response)* | *This method handles the HTTP ‘GET’ request. It forwards the request to the "login.jsp" page for the user to enter their credentials.*  ***Inputs****: ‘HttpServletRequest’, ‘HttpServletResponse’.*  ***Outputs****: The "login.jsp" view.*  ***Internal Processing****: Retrieves the "login.jsp" page and forwards the request and response objects to it.* |
| *02* | *doPost(HttpServletRequest request, HttpServletResponse response)* | *This method handles the HTTP `POST` request. It processes the login attempt using the provided credentials.*  ***Inputs****: `HttpServletRequest`, `HttpServletResponse`.*  ***Outputs****: Redirects to the referer page or the login page with an error message.*  ***Internal Processing****: Retrieves user credentials from the request, validates them using ‘UserDAO’, and either creates a new session for the user upon successful login or redirects back with an error message.* |

***UserDAO class :***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Login(String email, String password)* | *This method is used to authenticate a user based on their email and password. It checks if there's a user in the database with the provided email and password.*  ***Input****: The method takes two parameters:*  ***email****: A String representing the user's email.*  ***password****: A String representing the user's password.*  ***Output****: The method returns a ‘User’ object. If a user with the provided email and password is found in the database, a ‘User’ object representing that user is returned. If no such user is found, the method returns ‘null’.*  ***Internal Processing****:*  *This code performs a database operation to fetch a user based on provided email and password. It prepares and executes a SQL SELECT statement, checks if any user matches the provided credentials, and if a match is found, it creates a User object from the result. If no match is found or an exception occurs, it returns null.* |

#### Sequence diagram

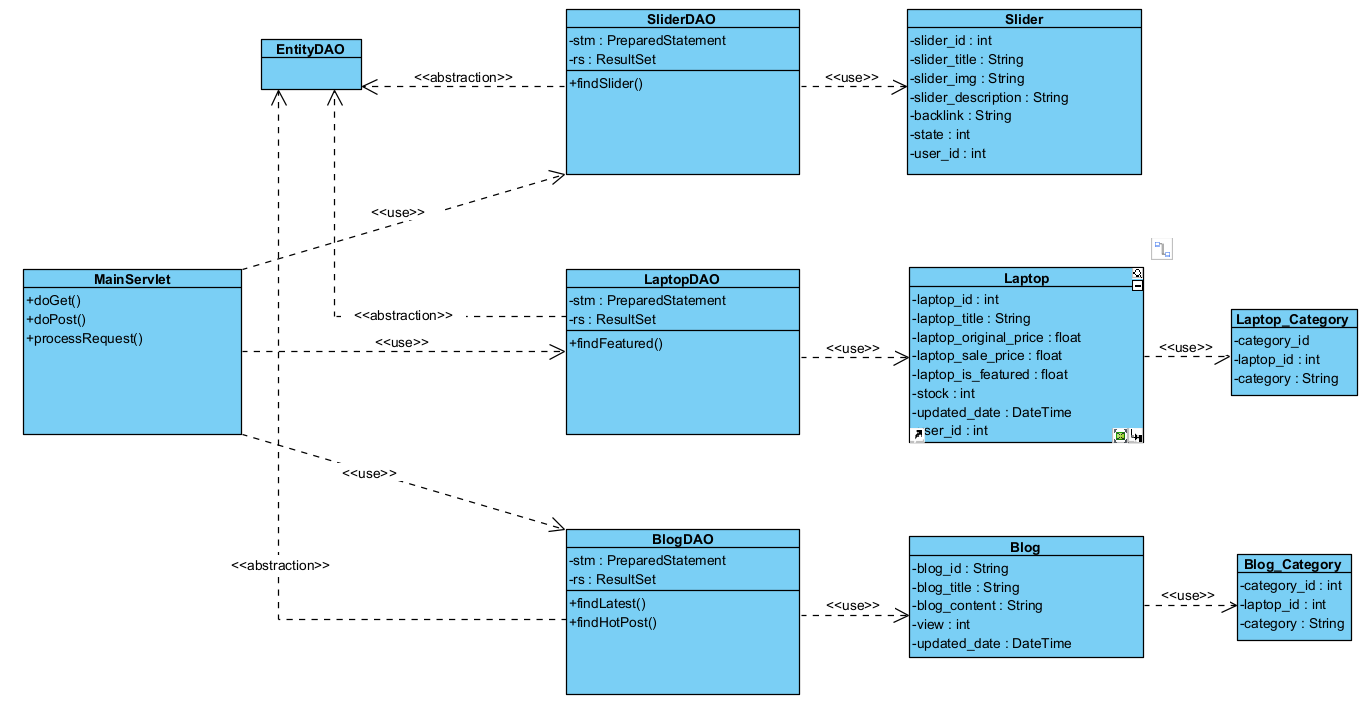


#### SQL Script:

* 1/ Verify Email & Password information
* SELECT \*
* FROM [user] WHERE email = ? AND password = ?
* 2/ Verify User role
* SELECT \*
* FROM [user] INNER JOIN [role] on user.role\_id = role.role\_id  WHERE user.id = ?

## 2,Home Pages

#### Class diagram



#### Class specification

***Blog Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getBlogID() | get a value for the blog\_id attribute |
| *02* | getBlogTitle() | get a value for the title attribute |
| *03* | getBlogContent() | get a value for the content attribute |
| *04* | getView() | get a value for the view attribute |
| *05* | getUpdatedDate() | get a value for the updated\_date attribute |
| *06* | setBlogID() | assign a value for the blog\_id attribute |
| *07* | setBlogTitle() | assign a value for the title attribute |
| *08* | setBlogContent() | assign a value for the content attribute |
| *09* | setView() | assign a value for the view attribute |
| *10* | setUpdatedDate() | assign a value for the updated\_date attribute |

***Laptop Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getLaptopID() | get a value for the blog\_id attribute |
| *02* | getLaptopTitle() | get a value for the title attribute |
| *03* | getOriginalPrice() | get a value for the original price attribute |
| *04* | getSalePrice() | get a value for the sale price attribute |
| *05* | getIsFeatured() | get a value for the isFeatured attribute |
| *06* | getStock() | get a value for the stock attribute |
| *07* | getUpdatedDate() | get a value for the updated\_date attribute |
| *08* | setLaptopID() | assign a value for the blog\_id attribute |
| *09* | setLaptopTitle() | assign a value for the title attribute |
| *10* | setOriginalPrice() | assign a value for the original price attribute |
| *11* | setSalePrice() | assign a value for the sale price attribute |
| *12* | setIsFeatured() | assign a value for the isFeatured attribute |
| *13* | setStock() | assign a value for the stock attribute |
| *14* | setUpdatedDate() | assign a value for the updated\_date attribute |

***Slider Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getSliderID() | get a value for the slider\_id attribute |
| *02* | getSliderTitle() | get a value for the title attribute |
| *03* | getSliderImg() | get a value for the title attribute |
| *04* | getSliderDescription() | get a value for the slider description attribute |
| *05* | getBackLink() | get a value for the back link attribute |
| *06* | getState() | get a value for the state attribute |
| *07* | setSliderID() | assign a value for the slider\_id attribute |
| *08* | setSliderTitle() | assign a value for the title attribute |
| *09* | setSliderImg() | assign a value for the title attribute |
| *10* | setSliderDescription() | assign a value for the slider description attribute |
| *11* | setBackLink() | assign a value for the back link attribute |

***Laptop\_Category Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCategory\_Id() | get a value for the category\_id attribute |
| *02* | getLaptop\_Id() | get a value for the laptop\_id attribute |
| *03* | getCategory() | get a value for the category attribute |
| *04* | setCategory\_Id() | assign a value for the category\_id attribute |
| *05* | setLaptop\_Id() | assign a value for the laptop\_id attribute |
| *06* | setCategory() | assign a value for the category attribute |

***Blog\_Category***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCategory\_Id() | get a value for the category\_id attribute |
| *02* | getBlog\_Id() | get a value for the laptop\_id attribute |
| *03* | getCategory() | get a value for the category attribute |
| *04* | setCategory\_Id() | assign a value for the category\_id attribute |
| *05* | setBlog\_Id() | assign a value for the laptop\_id attribute |
| *06* | setCategory() | assign a value for the category attribute |

***SliderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findSliders(int[] sArray)* | *This method is used to show list of sliders that corresponding in the database*  ***Input****: The method takes one parameters:*  ***int[] sArray: contains array of id for Sliders***  ***Output****: The method returns a List of Sliders that corresponding on the database*  ***Internal Processing****:*  *This code perform a query while attempt to convert the array of id to an IN(), so that it can access and return all the Slider with corresponding id* |

***LaptopDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findFeatured()* | *This method is used to show list of Laptop that has tag ‘is\_featured’ on.*  ***Input****: none*  ***Output****: A List of Laptop that has the tag ‘is\_Featured’ on*  ***Internal Processing****:*  *This code perform a query to search for 10 newest Laptop that has tag ‘is\_featured’* |

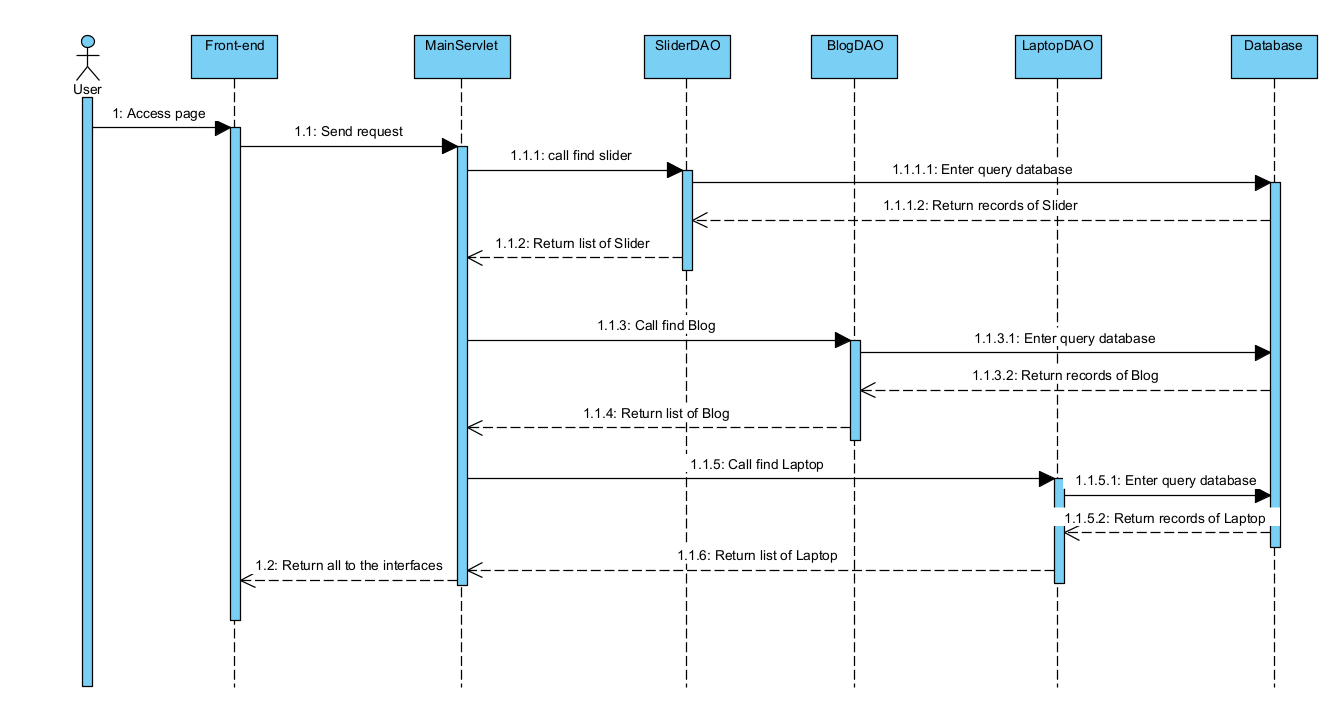
***BlogDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findLatest()* | *This method is used to show list of newest Blog*  ***Input****: none*  ***Output****: A List of newest Blog*  ***Internal Processing****:*  *This code perform a query to search for 3 newest Blog by executing ‘order by updated\_date DESC’* |
| *02* | *findHotPosts()* | *This method is used to show list of Blog that is have most view*  ***Input****: none*  ***Output****: A List of hot Blog*  ***Internal Processing****:*  *This code perform a query to search for 10 Hotest Blog by executing ‘ORDER BY view DESC’* |

***MainServlet Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest(HttpServletRequest request, HttpServletResponse response)* | *This method is used to show list of newest Blog*  ***Input****: HttpServletRequest request, HttpServletResponse response*  ***Output****: Data return to page ‘index.jsp’ to show to the user*  ***Internal Processing****:*  *The code called 4 method, findLatest(), findHotPosts(), findFeatured() to attempt to send the list back to ‘index.jsp’, The jsp then using JSTL to display data to user.* |

#### Sequence Diagram

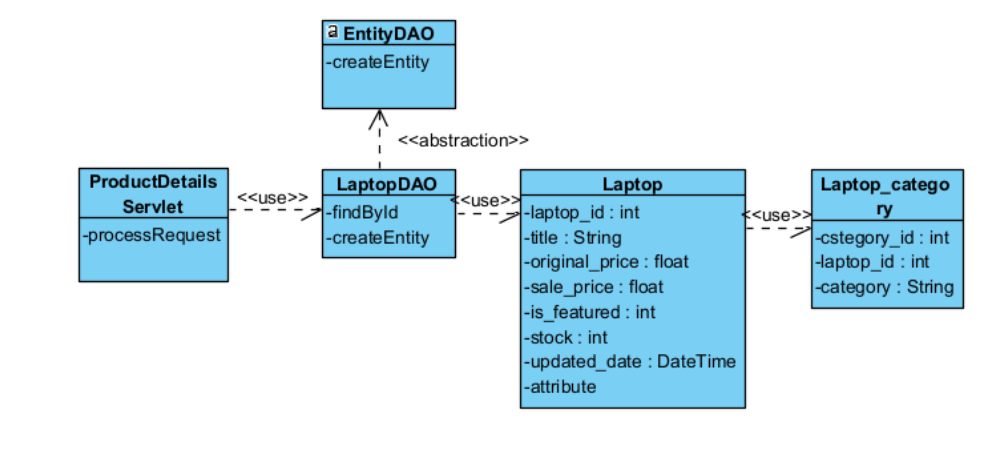
******

#### SQL Script:

* SELECT TOP 4 \* FROM Blog ORDER BY [views] DESC
* SELECT TOP 2 laptop\_id, title, main\_image, original\_price, stock, products\_detail, brief\_information, sale\_price, status, updated\_date FROM Laptop WHERE is\_featured = 1
* SELECT \* FROM Slider
* SELECT TOP 3 \* FROM Blog

## 3, ProductsDetails

### 1, Class diagram

****

### 2. Class specification

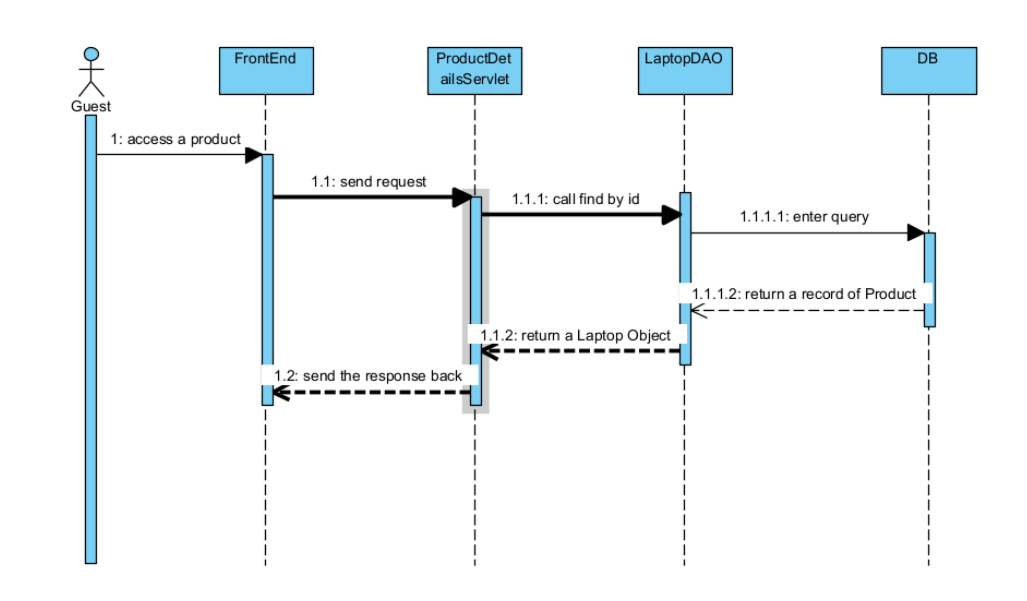
***Laptop Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getLaptopID() | get a value for the blog\_id attribute |
| *02* | getLaptopTitle() | get a value for the title attribute |
| *03* | getOriginalPrice() | get a value for the original price attribute |
| *04* | getSalePrice() | get a value for the sale price attribute |
| *05* | getIsFeatured() | get a value for the isFeatured attribute |
| *06* | getStock() | get a value for the stock attribute |
| *07* | getUpdatedDate() | get a value for the updated\_date attribute |
| *08* | setLaptopID() | assign a value for the blog\_id attribute |
| *09* | setLaptopTitle() | assign a value for the title attribute |
| *10* | setOriginalPrice() | assign a value for the original price attribute |
| *11* | setSalePrice() | assign a value for the sale price attribute |
| *12* | setIsFeatured() | assign a value for the isFeatured attribute |
| *13* | setStock() | assign a value for the stock attribute |
| *14* | setUpdatedDate() | assign a value for the updated\_date attribute |

***Laptop\_Category Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCategory\_Id() | get a value for the category\_id attribute |
| *02* | getLaptop\_Id() | get a value for the laptop\_id attribute |
| *03* | getCategory() | get a value for the category attribute |
| *04* | setCategory\_Id() | assign a value for the category\_id attribute |
| *05* | setLaptop\_Id() | assign a value for the laptop\_id attribute |
| *06* | setCategory() | assign a value for the category attribute |

### 3. Sequence Diagram

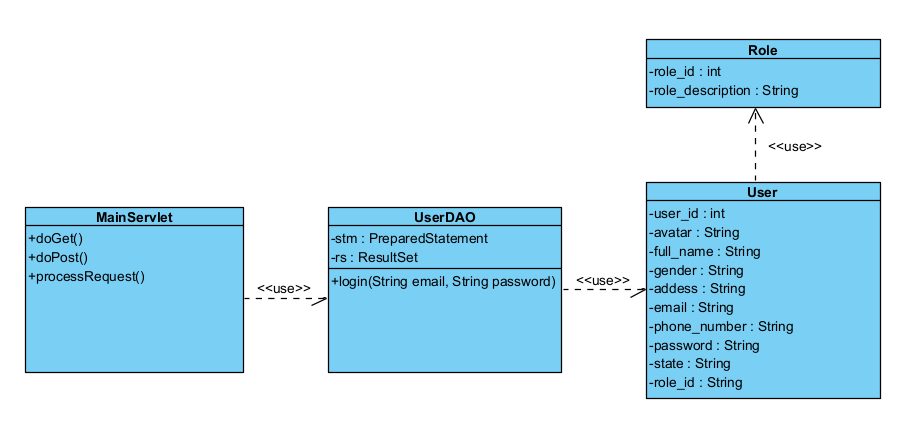
****

### SQL Query:

Select \* from Laptop where Laptop.laptop\_id = ?

## 4, Login

### 1, Class diagram



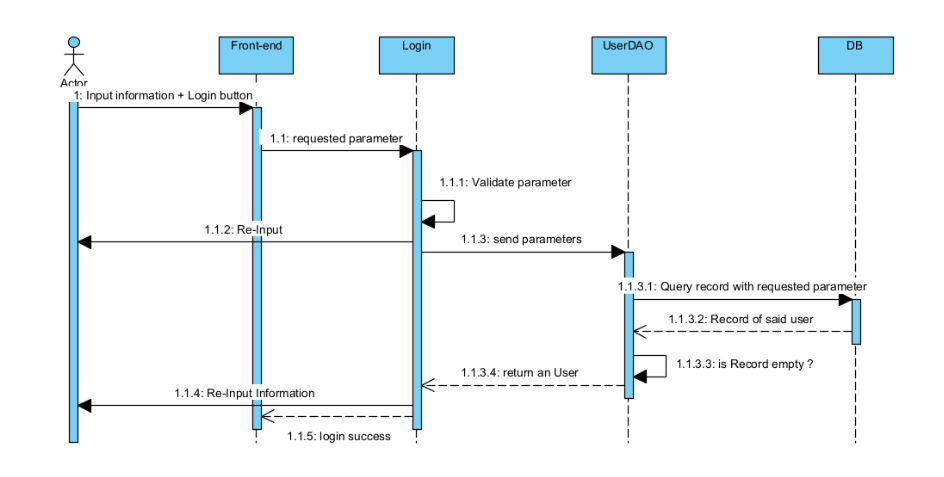
### 2, Class Specification

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *Login(String email, String password)* | *This method is used to authenticate a user based on their email and password. It checks if there's a user in the database with the provided email and password.*  ***Input****: The method takes two parameters:*  ***email****: A String representing the user's email.*  ***password****: A String representing the user's password.*  ***Output****: The method returns a ‘User’ object. If a user with the provided email and password is found in the database, a ‘User’ object representing that user is returned. If no such user is found, the method returns ‘null’.*  ***Internal Processing****:*  *This code performs a database operation to fetch a user based on provided email and password. It prepares and executes a SQL SELECT statement, checks if any user matches the provided credentials, and if a match is found, it creates a User object from the result. If no match is found or an exception occurs, it returns null.* |

### 3. Sequence diagram

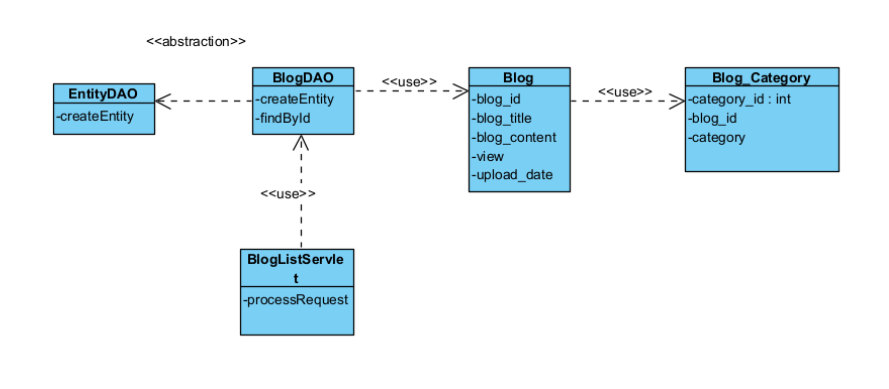


### 4. SQL Script:

* 1/ Verify Email & Password information
* SELECT \*
* FROM [user] WHERE email = ? AND password = ?
* 2/ Verify User role
* SELECT \*
* FROM [user] INNER JOIN [role] on user.role\_id = role.role\_id  WHERE user.id = ?

## 5, BlogDetails

### 1, Class diagram

******

### 2. Class specification

***Blog Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getBlogID() | get a value for the blog\_id attribute |
| *02* | getBlogTitle() | get a value for the title attribute |
| *03* | getBlogContent() | get a value for the content attribute |
| *04* | getView() | get a value for the view attribute |
| *05* | getUpdatedDate() | get a value for the updated\_date attribute |
| *06* | setBlogID() | assign a value for the blog\_id attribute |
| *07* | setBlogTitle() | assign a value for the title attribute |
| *08* | setBlogContent() | assign a value for the content attribute |
| *09* | setView() | assign a value for the view attribute |
| *10* | setUpdatedDate() | assign a value for the updated\_date attribute |

***Blog\_Category Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCategory\_Id() | get a value for the category\_id attribute |
| *02* | getBlog\_Id() | get a value for the laptop\_id attribute |
| *03* | getCategory() | get a value for the category attribute |
| *04* | setCategory\_Id() | assign a value for the category\_id attribute |
| *05* | setBlog\_Id() | assign a value for the laptop\_id attribute |
| *06* | setCategory() | assign a value for the category attribute |

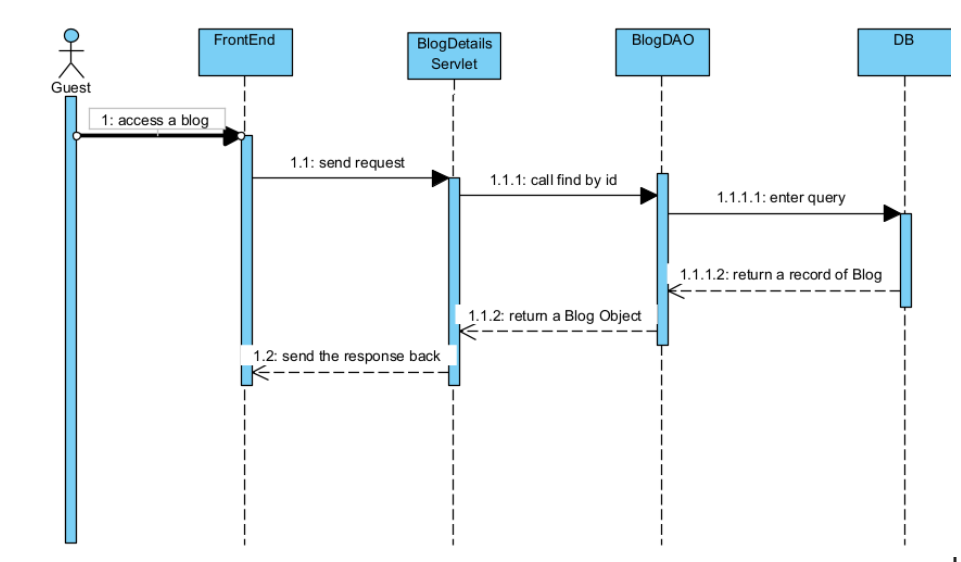
***BlogDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findById()* | *This method is used to return a Blog with corresponding id*  ***Input****: int id,*  ***Output****: A Blog that has coresponding id*  ***Internal Processing****:*  *The method execute a query in database then use rs to replace ? with actual value, then return a record of Blog, or null if no record found* |
| *02* | *createEntity()* | *This method extends from EntityDAO, returns an Object from database*  *Input: Result Set*  *Output: An Object of correct type, or null if expectation happens* |

***BlogDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest()* | *This method is take the value user took via request, call method getById() to return a blog with corresponding id for user*  ***Input****: HTTPServlet request, HTTPServlet response*  ***Output****: A Blog with same id as what user wanted to see, show to “single-post.jsp”*  ***Internal Processing****:*  *The method takes the request attribute, call BlogDAO to process it. The system then took the data to the database, return result to jsp which show to user* |

### 3. Sequence Diagram

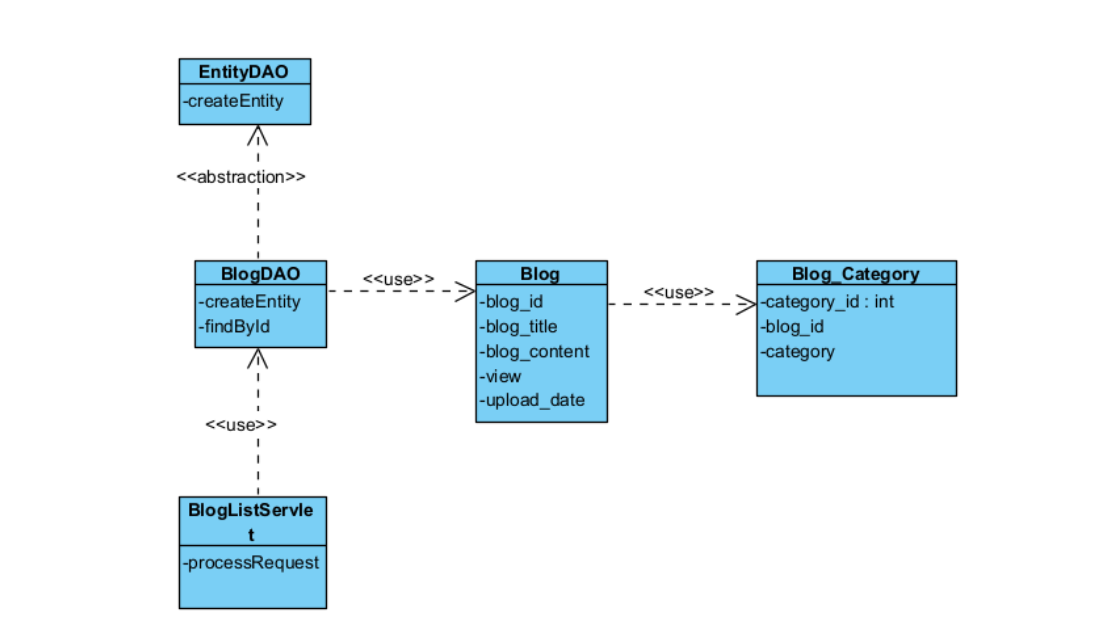
******

### 4, SQL Script

Select \* from Blog where Blog.blog\_id = ?

## 6, BlogLists

### 1, Class diagram

****

### 2. Class specification

***Blog Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getBlogID() | get a value for the blog\_id attribute |
| *02* | getBlogTitle() | get a value for the title attribute |
| *03* | getBlogContent() | get a value for the content attribute |
| *04* | getView() | get a value for the view attribute |
| *05* | getUpdatedDate() | get a value for the updated\_date attribute |
| *06* | setBlogID() | assign a value for the blog\_id attribute |
| *07* | setBlogTitle() | assign a value for the title attribute |
| *08* | setBlogContent() | assign a value for the content attribute |
| *09* | setView() | assign a value for the view attribute |
| *10* | setUpdatedDate() | assign a value for the updated\_date attribute |

***Blog\_Category Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCategory\_Id() | get a value for the category\_id attribute |
| *02* | getBlog\_Id() | get a value for the laptop\_id attribute |
| *03* | getCategory() | get a value for the category attribute |
| *04* | setCategory\_Id() | assign a value for the category\_id attribute |
| *05* | setBlog\_Id() | assign a value for the laptop\_id attribute |
| *06* | setCategory() | assign a value for the category attribute |

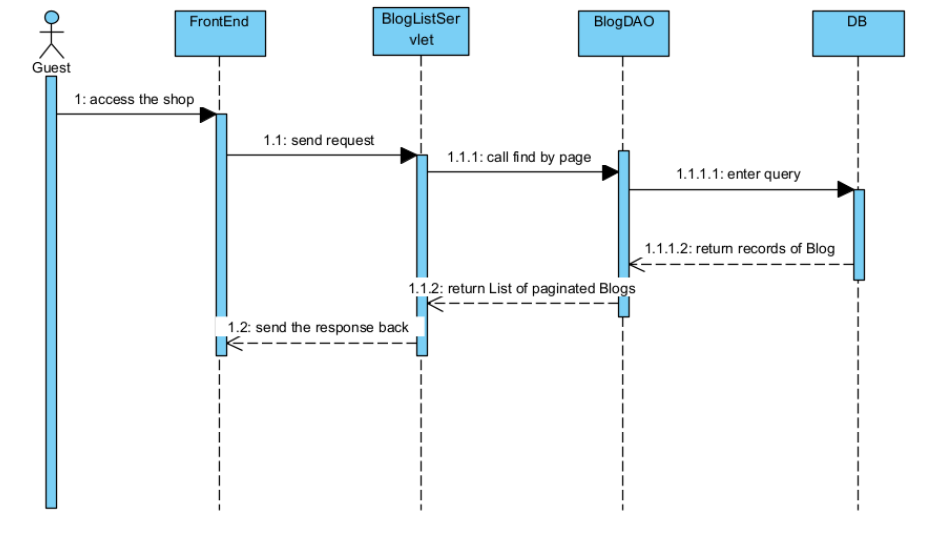
***BlogDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *findByPage()* | *This method is used to return a page of a search query*  ***Input****: int page, int totalPage,String[] category, String order\_by, String title*  ***Output****: A List of Blog that match the request and on corresponding page*  ***Internal Processing****:*  *The method convert the input parameter to be included in if they’re not null, then using ? so that stm.set can work. Lastly, the result being divided by page due to FETCH ? ROWS ONLY* |
| *02* | *createEntity()* | *This method extends from EntityDAO, returns an Object from database*  *Input: Result Set*  *Output: An Object of correct type, or null if expectation happens* |

***BlogDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest()* | *This method is take the value user took via request, call method getByPage() to return pages of search result for user*  ***Input****: HTTPServlet request, HTTPServlet response*  ***Output****: A List of Blog fit the criterias with enough data expected being fetch as pages, show to “blog.jsp”*  ***Internal Processing****:*  *The method takes the request attribute, call BlogDAO to process it. The system then took the data to the database, return result to jsp which show to user* |

### 3. Sequence Diagram

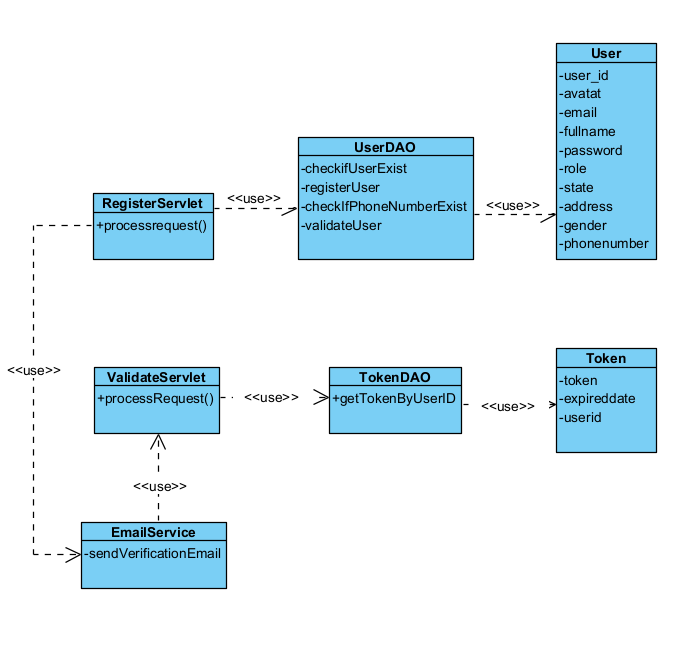
****

### 4, SQL Script

select \* from Blog INNER JOIN Blog\_Category ON Blog.blog\_id = Blog\_Category.blog\_id WHERE category IN (?. ?. ?) AND title LIKE ? OFFSET ? ROWS FETCH NEXT ? ROWS ONLY

## 7, User Register

### 1, Class diagram

****

### 2. Class specification

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

***UserDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkIfUserExist()* | *This method is used to check if an user is exist*  ***Input****: String email*  ***Output****: a true/false boolean to check if an user is already being created inside database using email*  ***Internal Processing****:*  *The method check if an user is already exist in database by validating the email. If it does, return true. if it doesn’t, return false* |
| *02* | *registerUser()* | *This method is used to create an user with inputted information*  *Input: an User Class*  *Output: an Object is being created in database* |
| *03* | *checkIfPhoneNumberExist()* | *This method is used to check if an user is exist by phone number*  ***Input****: String phone number*  ***Output****: a true/false boolean to check if an user is already being created inside database*  ***Internal Processing****:*  *The method check if an user’s phone number is already exist in database. If it does, return true. if it doesn’t, return false* |
| *04* | *ValidateUser* | *This method used to validate User*  ***Input****: int UserId*  ***Output****: User with said Id has state from “Unverified” to “Verified”*  ***Internal Processing****:*  *This method find one user with said id, set the state to “Verified”* |

**RegisterServlet Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest()* | *This method is used to handle request of register from Servlet*  ***Input****: HTTPServlet request, HTTPServlet response*  ***Output****: successfully created an user in the database and require the user to validate before process, or return the error to user into the register pop-up*  ***Internal Processing****:*  *The method called UserDAO to check phone number and email to make sure it’s validated, then create an User with customer roles and “unverified” state, then send and email to user and told them to validate* |

**ValidationServlet Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *processRequest()* | *This method is used to handle request of validation*  ***Input****: HTTPServlet request, HTTPServlet response*  ***Output****: successfully changes the state of user, or do not if user token expired*  ***Internal Processing****:*  *The method called TokenDAO to validate token, makes sure it’s not pass 30 minutes(expired), then validate User* |

***Token Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getToken() | get a value for the token attribute |
| *03* | getExpiredDate() | get a value for the expired date attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | setToken() | set a value for the token attribute |
| *06* | setExpiredDate() | set a value for the expired date attribute |

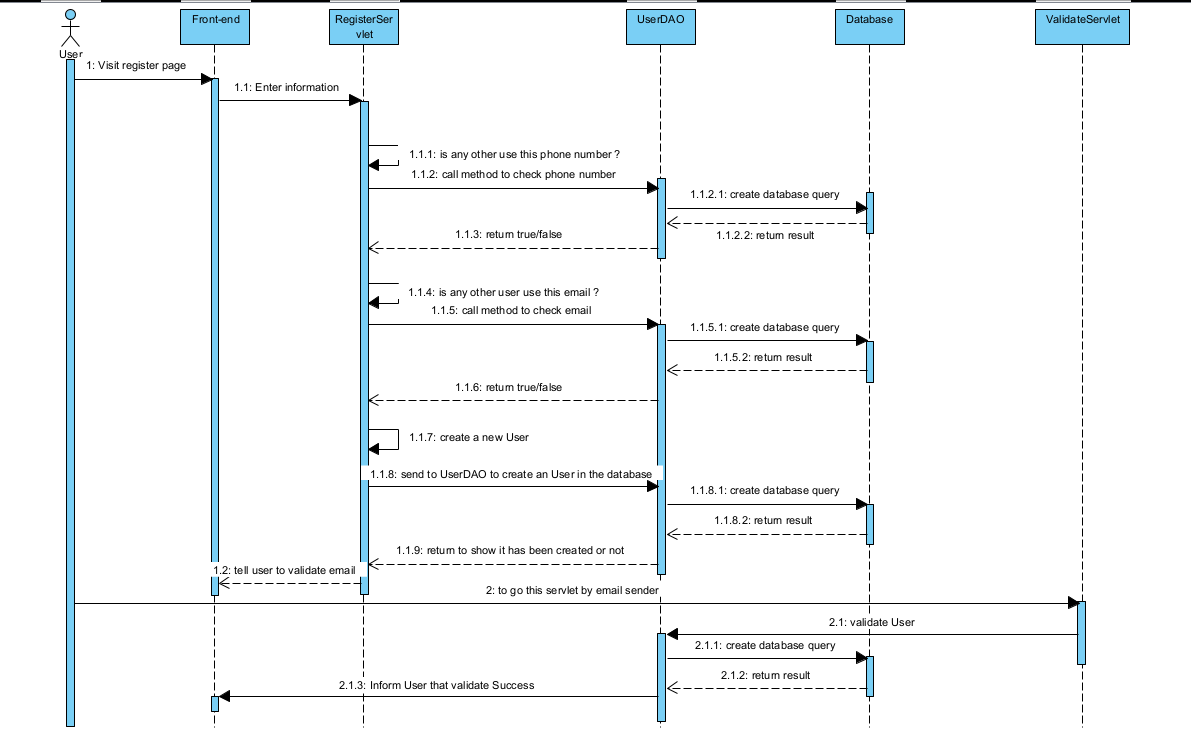
**TokenDAO Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getByUserId()* | *This method is used to return a token based on inputted user\_id*  ***Input****: int UserId*  ***Output****: a Token*  ***Internal Processing****:*  *The method use user\_id to attempt to search for Token, then return it or null if not found* |

**EmailService Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *sendVerificationEmail()* | *This method is send an email for user to validate their account*  ***Input****: none*  ***Output****: an email send to the user by gmail*  ***Internal Processing****:*  *The method using SimpleJavaMail to send an email to requested User* |

### 3. Sequence Diagram



### 4. SQL Script

**1/ Check if an email with an user exist**

Select 1 from [User] where email = ?

**2/ Check if a phone number with an user exist**

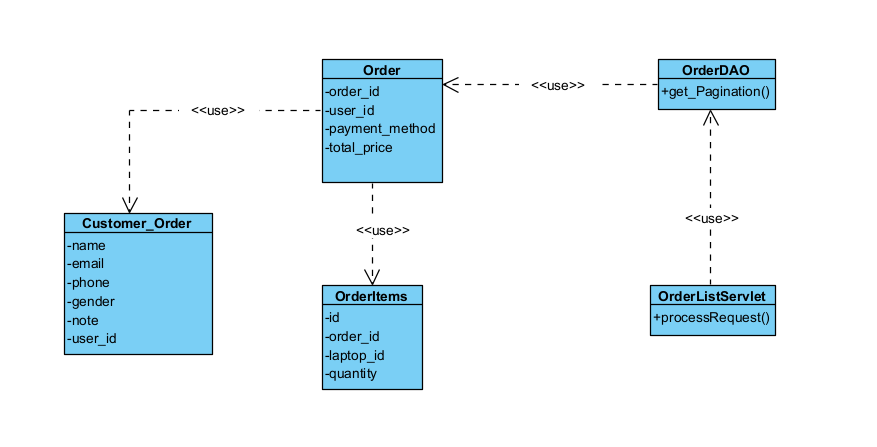
Select 1 from [User] where phone\_number = ?

**3/Create a new account**

Insert into [User] values(?,?,?,?,?,?,?,?,?)

## 8. Order List:

### Class Diagram

****

### Class Specification

***Order Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setOrderID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPayment() | return payment for an order |
| *06* | setPayment() | assign a value to the payment of an order |

***Order\_Item Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getID() | get a value from the cart\_id attribute |
| *02* | getOrderID() | get a value from the user\_id attribute |
| *03* | getProductId() | get value from the product\_id |
| *04* | getQuantity | get value from the quantity |
| *05* | setD() | set a value for the cart\_id attribute |
| *06* | setOrderID() | set a value for the email attribute |
| *07* | setProductId() | set a value for the email attribute |
| *08* | setQuantity() | set a value for the email attribute |

User\_Order Class:

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |

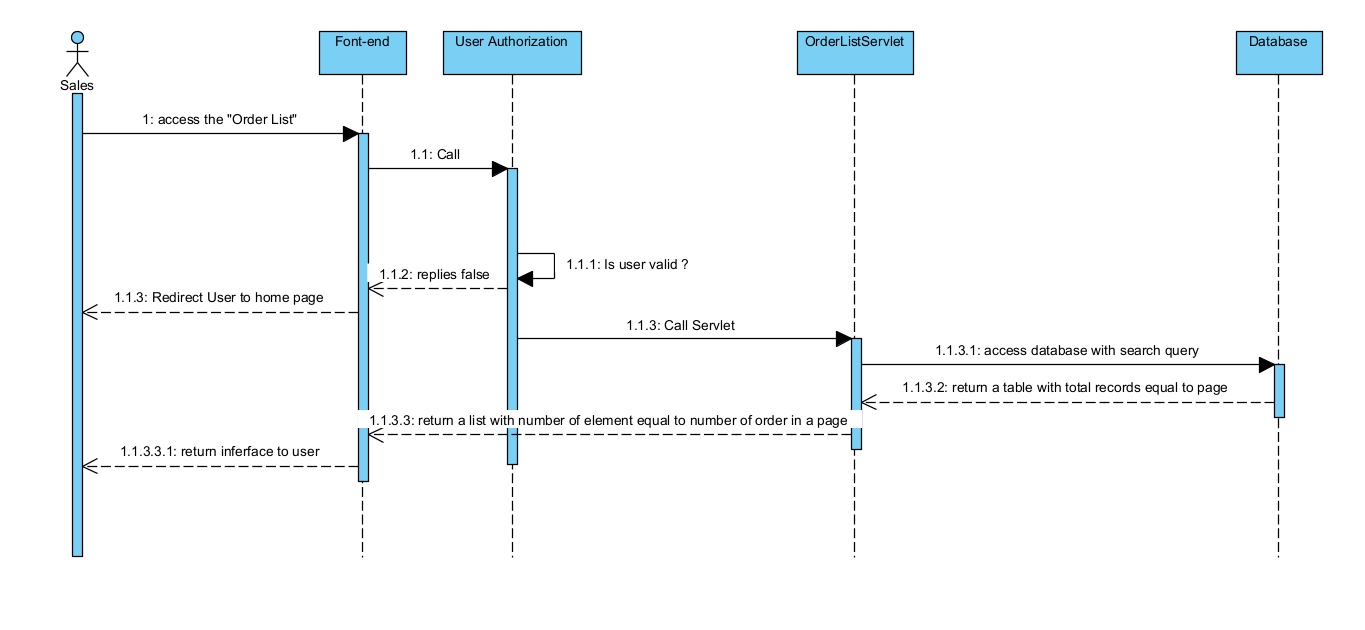
***OrderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByPage | This method, getOrderPage, is used to fetch a page of Order records from a database. It uses SQL queries to fetch the data and returns a list of Order objects. **Inputs:**  page: The page number of the results to fetch.  totalPerPage: The number of records to fetch per page.  order\_by: The column by which to order the results.  direction: The direction in which to order the results (ascending or descending).  search\_query: A query string to search the orders.  start: The start date for the orders.  end: The end date for the orders.  status: The status of the orders to fetch.  selectedLaptopId: An array of laptop IDs to filter the orders by.  **Output:** The method returns a list of Order objects that match the given criteria.  **Internal Processing:**  The method first constructs an SQL query string based on the provided parameters.  It then prepares a PreparedStatement with the constructed SQL query.  It sets the parameters of the PreparedStatement based on the provided method parameters.  It executes the query and fetches the result set.  It iterates over the result set, creates Order objects from the result set records, and adds them to a list.  If an SQLException is thrown during the process, it logs the exception and continues. |

***OrderlistServlet Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest | This method is used to fetch the list of order based on user previous input **Input**: HTTPServlet request, HTTPServetResponse  **Output**: Web redirect to orderlist.jsp with filter and a list of all order, paginatedly **Internal Process**: It fetch all the input and search by any of them, or a normal paginated of all return null |

### Sequences Diagram



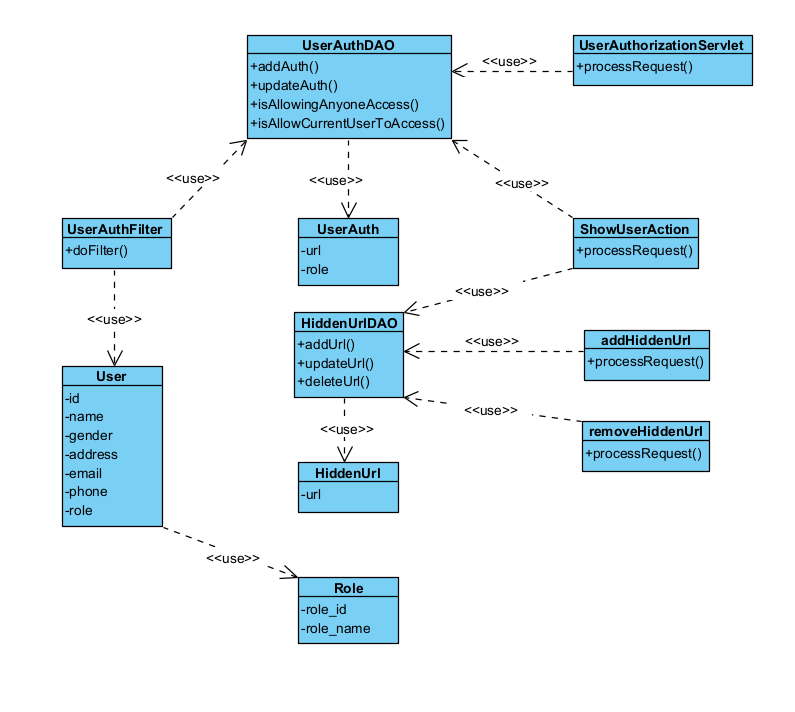
### SQL Query

**1/Show pagination**

SELECT \* FROM [Order] INNER JOIN Order\_User ON [Order].order\_uid = Order\_User.order\_uid INNER JOIN [Order\_Item] on [Order].order\_id = Order\_Item.order\_id AND order\_id = ? AND Order\_User.fullname LIKE ? AND order\_date >= ? AND order\_date < ? AND order\_date < ? ORDER BY ? ASC OFFSET ? ROWS FETCH NEXT ? ROWS ONLY

## 9. User Authorization

### Class diagram



### Class specification

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

***Role Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | getRole\_Id() | get a value for the role\_id attribute |
| *2* | getRoleDescription() | get a value for the role description attribute |
| *3* | setRole\_Id() | assign a value for the role\_id attribute |
| *4* | setRoleDescription() | assign a value for the role description attribute |

***UserAuthorizationDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | addAuth() | **Input:** Class UserAuth  **Output:** Update the role authorization for a page  Internal Processing: This method using the value from UserAuth class to add the data into the database |
| *2* | updateAuth() | **Input:** Class UserAuth  **Output:** Update the role authorization for a page  Internal Processing: This method using the value from UserAuth class to update the data into the database |
| *3* | isAllowAnyOneAccess(String url) | **Input:** A URL.  **Output:** A boolean value indicating whether the URL can be accessed by anyone.  **Internal Processing:** It checks if the URL is exist anywhere in the UserAuthorization table. Return true if exist and otherwise |
| *4* | isAllowCurrentUserToAccess(User user, String url) | **Input:** A User object and a URL.  **Output:** A boolean value indicating whether the user can access the URL.  **Internal Processing:** It checks in the UserAuthorization table that is current user role with current url exist.Return true if such record exist, and otherwise |

***HiddenUrl Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | getUrl() | get a value for the url attribute |
| *2* | setUrl() | assign a value to the url attribute |

***HiddenUrlDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | addUrl() | **Input:** String url  **Output:** Add a new url to be hidden in the database  **Internal Processing**: It using PreparedStatement and add a new record onto the database |
| *2* | deleteUrl() | **Input:** String url  **Output:** Remove the url from the database  **Internal Processing**: It using PreparedStatement to delete one record where url is exactly matched this url |

***UserAuth Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | doFilter() | **Input:** No input.  **Output:** No direct output, but it prevents user from accessing pages that they do not hav authorizations over  **Internal Processing:** It fetch the current user role by fetching userAuth session(or guest role if they do not logged in)**,** then search from UserAuthorization via the UserAuthDAO. If it did, allow the user to logged in, otherwise, redirect user to the first page they can access to |

***UserAuthorization Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | processRequest() | **Input:** HttpServletRequest, HttpServletResponse  **Output:** inject data into userauthorization.jsp then show to the user  **Internal Processing:** It call servlet to fetch all url, call RoleDAO to fetch all current type of role, call hiddenUrl and UserAuthorization to get all type of currently configured role, inject and shows them out to the front-end |

***ShowUserAction Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | processRequest() | **Input:** HttpServletRequest, HttpServletResponse  **Output:** inject data into showAction.jsp then show to the user  **Internal Processing:** It call the UserAuthorization table and HiddenUrl to remove some that current user has role over but it should not shows out, then display |

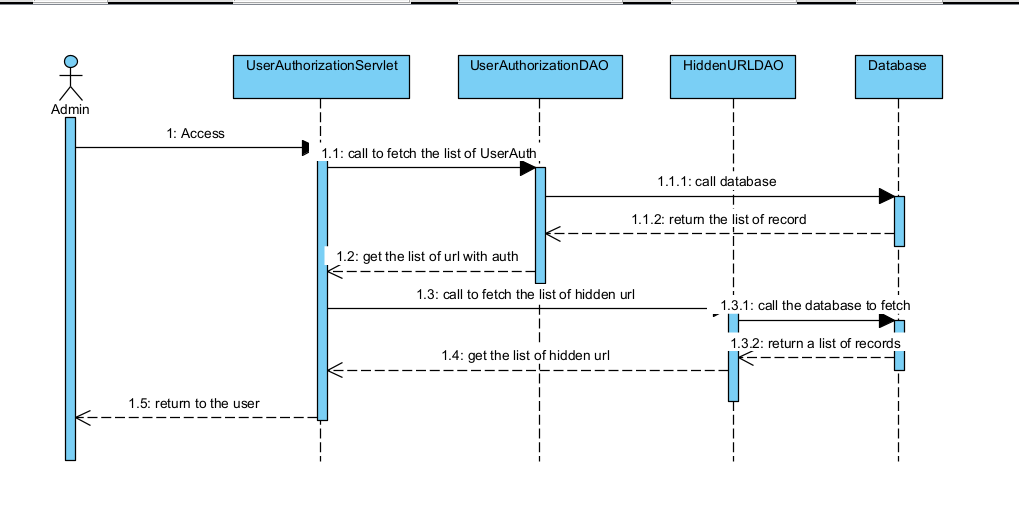
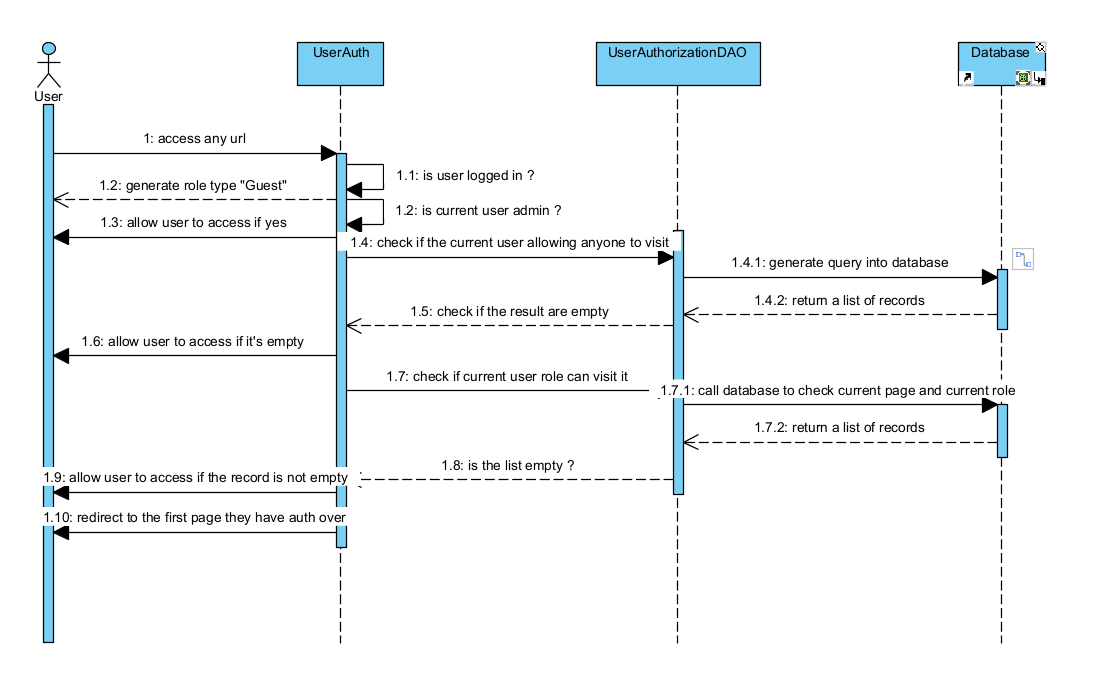
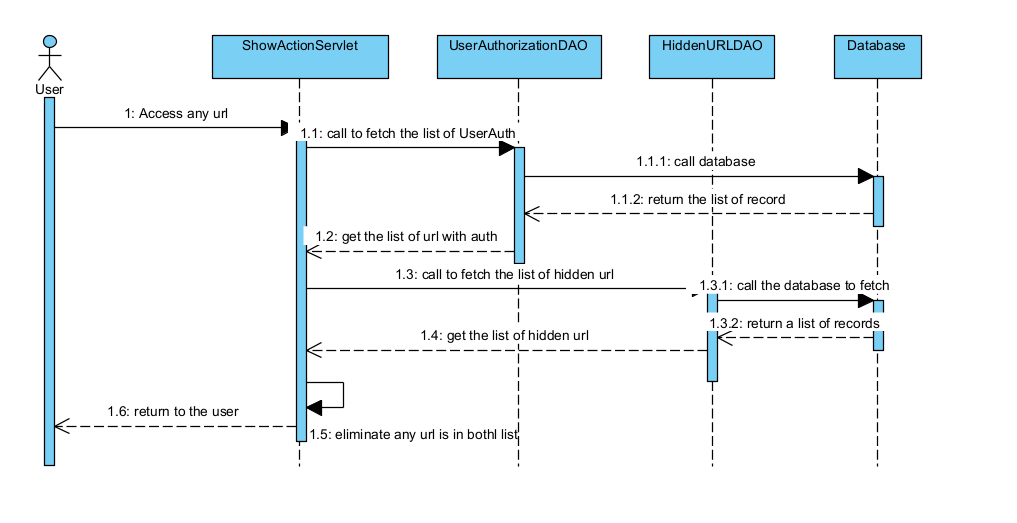
***addHiddenUrl Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | processRequest() | **Input:** HttpServletRequest, HttpServletResponse  **Output:** add the url to the database  **Internal Processing:** It validate the data to find the url string arrays. Return to the referrer if neither able to insert or no information was found |

***deleteHiddenUrl Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *1* | processRequest() | **Input:** HttpServletRequest, HttpServletResponse  **Output:** delete the record with matched url in the hiddenUrl page  **Internal Processing:** It fetch the url, validate then remove the url with said name in the database |

### Sequence Diagram



### SQL Script

***SQL Commands***

**1/ Find User with id**

SELECT \* FROM [User] where user\_id = ?

**2/ Find Role with id**

SELECT \* FROM [Role] where role\_id = ?

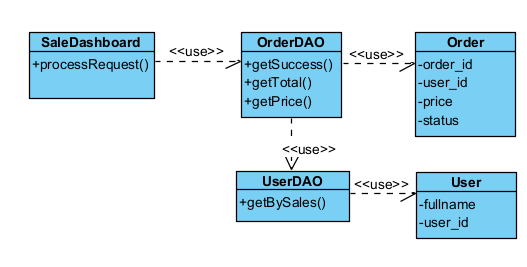
**3/Create authorization for URL page:**  
INSERT INTO User\_Authorization VALUES (?,?)

**4/HiddenUrl page**

INSERT INTO hidden\_url VALUES(?)

## 10. Sales Dashboard

### 1, Class diagram



### 2. Class specification

***Order Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***User Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |

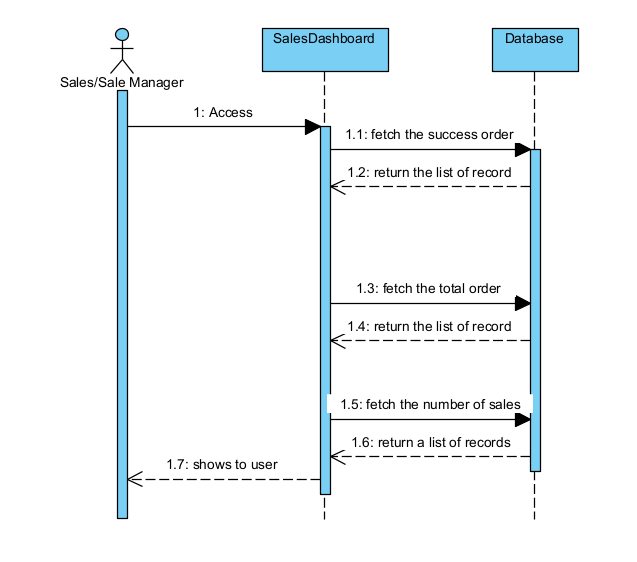
***UserOrderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getBySales() | using inputted user\_id, it return an User class with role of sales Input: int user\_id  Input: None  Output: List of Sales role User  Internal Processing: This method find in the database all user that has the role sales and then return it |

**OrderDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getSuccess() | This method return all the order that success that inside the date range and who approved it  Input: Date start, Date end, User u  Output: List of Order that fitted the criteria  Internal Processing: This method find the order by executing the query and the provided criteria |
| *01* | getTotal) | This method return all the order that inside the date range and who approved it  Input: Date start, Date end, User u  Output: List of Order that fitted the criteria  Internal Processing: This method find the order by executing the query and the provided criteria |
| *01* | getPrice() | This method return all the price that success that inside the date range and who approved it  Input: Date start, Date end, User u  Output: List of price that fitted the criteria  Internal Processing: This method find the order by executing the query and the provided criteria |

### 3. Sequences Diagram



### 4.SQL Script

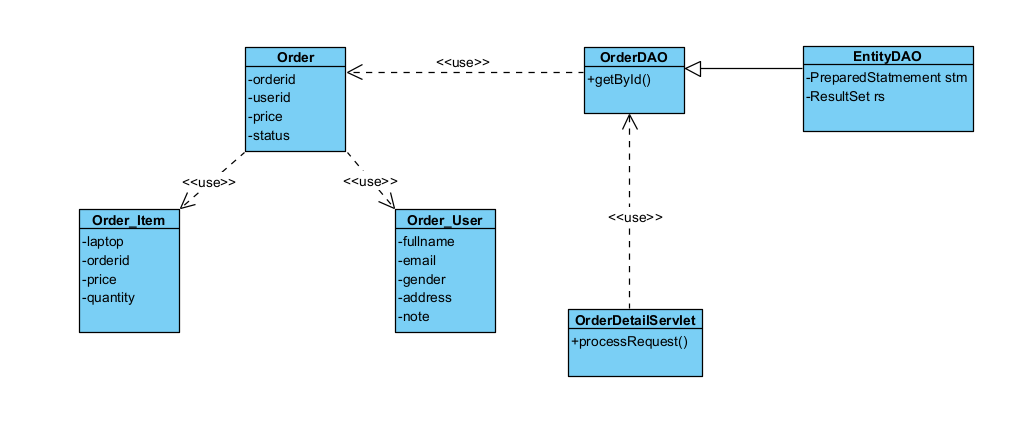
**/Total**

SELECT \* FROM [Order] where Order.updated\_date between ? and ?  
**/Success**

SELECT \* FROM [Order] where Order.updated\_date between ? and ? and state == 3

## 11. Order Details

### 1, Class Diagram



### 2, Class Specification

**Order Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***OrderUser Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |
| *09* | getGender() | return value of gender |
| *10* | setGender() | assign a value for gender |

***OrderItem Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | setOrderID() | get a value for the order\_id attribute |
| *03* | getLaptop() | get a value for the laptop attribute |
| *04* | setLaptop() | set a value for the laptop attribute |
| *05* | getPrice() | return price of the order |
| *06* | setPrice() | assign a value to the price of an order |
| *07* | getQuantity() | return quantity of laptop in an order |
| *08* | setQuantity() | assign a value for quantity of laptop |

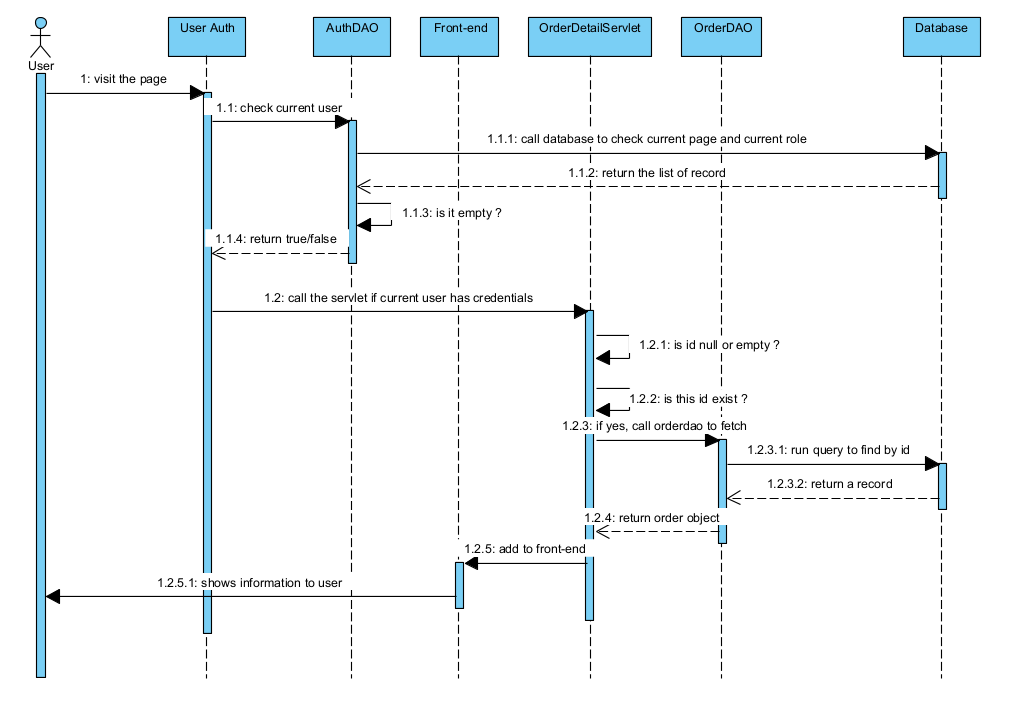
**OrderDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByID() | This method return an order that has the id corresponding to the id is requested  Input: String id  Output: an Order with the corresponding id  Intenal Processing: This method create a search query to find an order into the database. if the result has exist, fetch it via method createntity, otherwise, return null |
| *01* | getTotal) | this method used to create an entity of Order used ResultSet  Input: ResultSet rs  Output: an Order using that rs;  Internal Processing: This fetch from the RS, then create the Laptop and Order by calling both DAO, filling up this class and then return it |

**OrderDetailServlet Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest | This method shows the pag to the user  Input: HTTPServletRequest, HttpServletResponse  Output: a page with requested information, matched the requirement  Intenal Processing: This method first fetch the id, check if it’s null, check if it’s exist. After both has been fulfilled, create data and send to jsp. Otherwise, redirect and shows error |

### 3. Sequences Diagram



### 4. SQL Script

**Select order by ID**

SELECT \* FROM [Order] where order\_id = ?

**/Select order\_user by id**

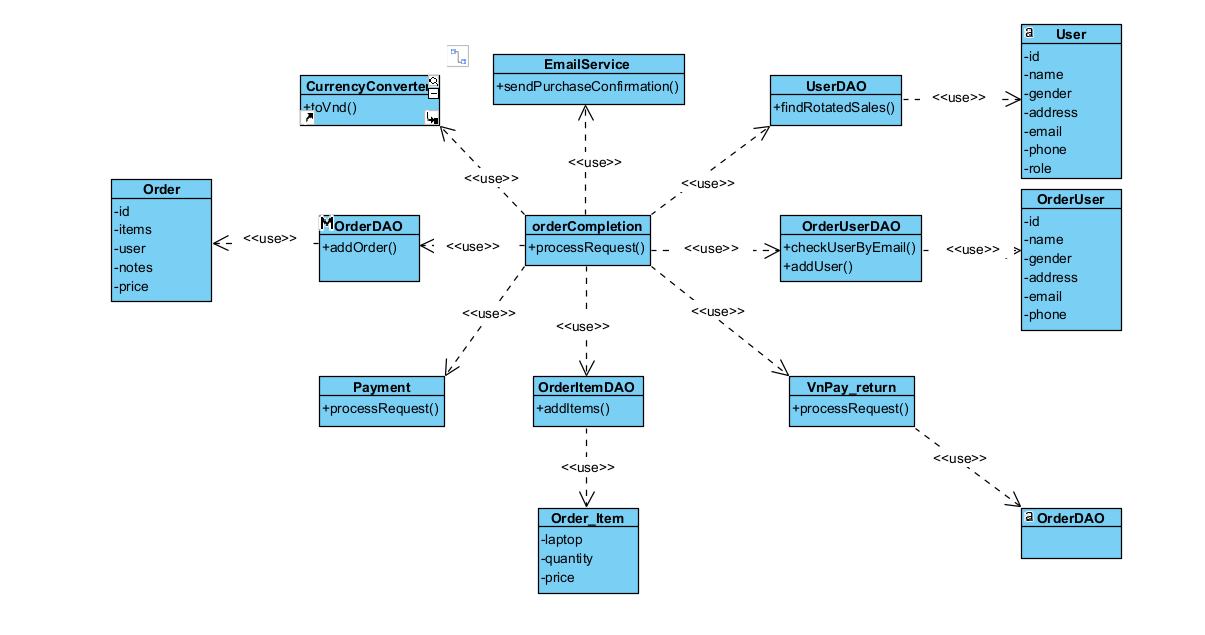
SELECT \* FROM [Order\_User] where order\_id = ?

**/Select ordered Items by id**

SELECT \* FROM [Order\_Item] where order\_id - ?

## 12. Cart Completion

### 1. Class Diagram



### 2. Class Specification

**Order Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***OrderUser Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |
| *09* | getGender() | return value of gender |
| *10* | setGender() | assign a value for gender |

***OrderItem Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | setOrderID() | get a value for the order\_id attribute |
| *03* | getLaptop() | get a value for the laptop attribute |
| *04* | setLaptop() | set a value for the laptop attribute |
| *05* | getPrice() | return price of the order |
| *06* | setPrice() | assign a value to the price of an order |
| *07* | getQuantity() | return quantity of laptop in an order |
| *08* | setQuantity() | assign a value for quantity of laptop |

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

**EmailService Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | sendpurchaseConfirmation() | This method send the mail of the order details to user via their email  Input: Order\_User u, HashMap<Laptop, Integer>, String receiveDate  Output: the mail being send to the user with their name, to their email and the items being displayed  Intenal Processing: This method using Simple Java Mail to send to the user |

**OrderUserDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | findByEmail() | This method check if 1 user with this mail already exist  Input: String email  Output: true if the user is found, false otherwise  Intenal Processing: This method using SQL to find the mail with 1 user in the database |
| *02* | addUser | This method insert the order into the database  Input: OrderUser u  Output: add the user into the database  Intenal Processing: This method using SQL to add the user to the database |

**UserDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | findRotatedSales() | This method rotatedly select the sales  Input: None  Output: id of the sales being rotated  Intenal Processing: This method using SQL to find the first min sales with lowest order management count |

**OrderDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addOrder() | This method add the order to the database  Input: Order class o  Output: the order being added in the database, set the id back to the order  Intenal Processing: This method using SQL to add the order into the database |

**OrderItemDAO Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addOrderItem() | This method rotatedly select the sales  Input: OrderItem class o  Output: the order item being added into the database  Intenal Processing: This method using SQL to add the orderitem into the database |

**payment Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | This method using configured data to redirect user to VnPay page  Input: HttpServletRequest request, HttpServletResponse response  Output: the payment page shows  Internal Processing: Using the hashkey, the code and provided payment information, redirect user to the VnPay payment page |

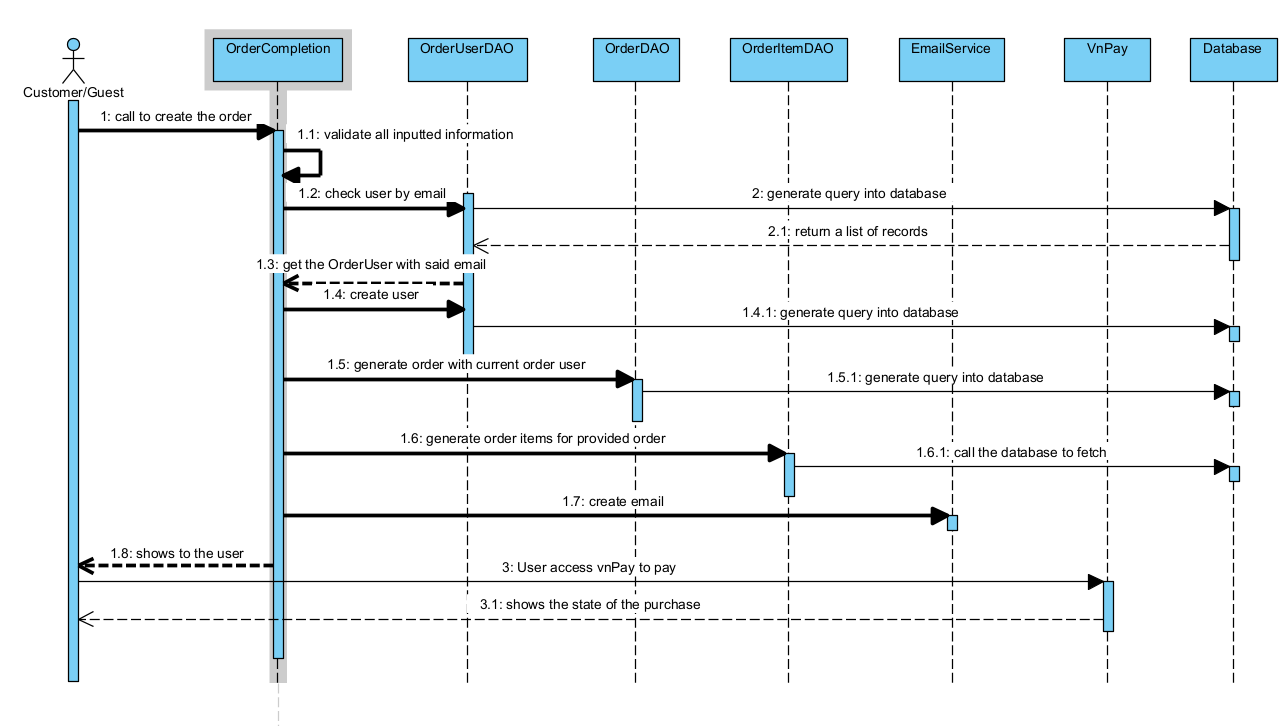
**vnPay\_return Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | This method is called after the payment has been done, using to detect payment status, therefore handle it  Input: HttpServletRequest request, HttpServletResponse response  Output: the page handler the vnpay parameter throw, and deal with it  Internal Processing: Using the hashkey, the code and provided payment information, redirect user to the VnPay payment page |

**orderCompletion Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | This method is handler all the overall payment step: update status of order, find active sales, create order user, sendPurchase Confirmation mail, predict the day order returns  Input: HttpServletRequest request, HttpServletResponse response  Output: the page handler the overall payment steps  Internal Processing: Using OrderDAO, OrderUserDAO, UserDAO, emailService mentioned above |

### 3. Sequences Diagram



### 4. SQL Query

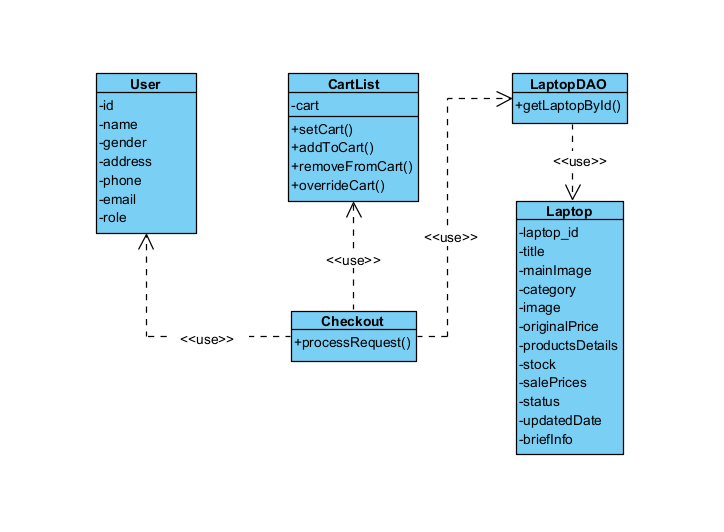
/INSERT INTO [Order] values (?,?,?,?,?,?)  
/INSERT INTO [Order\_User] values (?,?,?,?,?)

/INSERT INTO [Order\_Item] values (?,?,?,?)

/Update [Order] SET status = ? WHERE id = ?

## 13. Cart Information

### 1. Class diagram



### 2. Class Specification

***User Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the user\_id attribute |
| *02* | getAvatar() | get a value for the avatar attribute |
| *03* | getEmail() | get a value for the email attribute |
| *04* | getFullname() | get a value for the name attribute |
| *05* | getPassword() | get a value for the password attribute |
| *06* | getRole() | get a value for the role attribute |
| *07* | getState() | get a value for the state of user attribute |
| *08* | getAddress() | get a value for the address attribute |
| *09* | getGender() | get a value for the gender attribute |
| *10* | getPhoneNumber | get a value for the number of user attribute |
| *11* | setUserID () | assign a value for the user\_id attribute |
| *12* | setAvatar() | assign a value for the avatar attribute |
| *13* | setEmail() | assign a value for the email attribute |
| *14* | setFullname() | assign a value for the name attribute |
| *15* | setPassword() | assign a value for the password attribute |
| *16* | setRole() | assign a value for the role attribute |
| *17* | setState() | assign a value for the state of user attribute |
| *18* | setAddress() | assign a value for the address attribute |
| *19* | setGender() | assign a value for the gender attribute |
| *20* | setPhoneNumber() | assign a value for the number of user attribute |
| *21* | toString() | Return how object should act when being called to print out |

***CartList***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addToCart() | it add the product with it’s quantity to the HashMap session, or updating one if it’s already in the map  input: Laptop laptop, int quantity  output: the list of HashMap session with new item  Internal Processing: it check by search for the key, if it’s empty, add the new Laptop as key and quantity as it value, otherwise update the existing one |
| *02* | setCart() | it update one product that exist  Input: Laptop laptop, int quantity  Output: update the cart session  Internal Processing: It edit the quantity of a certain corresponding key |
| *03* | deleteFromCart() | It delete one entity  Input: Laptop laptop  Output: remove the cart from the map  Internal Processing: It find and remove the Laptop entity by the key |
| *04* | getCart() | It show the HashMap out  input: none  output: the HashMap with Laptop and it’s quantity  Internal Processing: it returns the HashMap |

***Laptop Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getLaptopID() | get a value for the blog\_id attribute |
| *02* | getLaptopTitle() | get a value for the title attribute |
| *03* | getOriginalPrice() | get a value for the original price attribute |
| *04* | getSalePrice() | get a value for the sale price attribute |
| *05* | getIsFeatured() | get a value for the isFeatured attribute |
| *06* | getStock() | get a value for the stock attribute |
| *07* | getUpdatedDate() | get a value for the updated\_date attribute |
| *08* | setLaptopID() | assign a value for the blog\_id attribute |
| *09* | setLaptopTitle() | assign a value for the title attribute |
| *10* | setOriginalPrice() | assign a value for the original price attribute |
| *11* | setSalePrice() | assign a value for the sale price attribute |
| *12* | setIsFeatured() | assign a value for the isFeatured attribute |
| *13* | setStock() | assign a value for the stock attribute |
| *14* | setUpdatedDate() | assign a value for the updated\_date attribute |

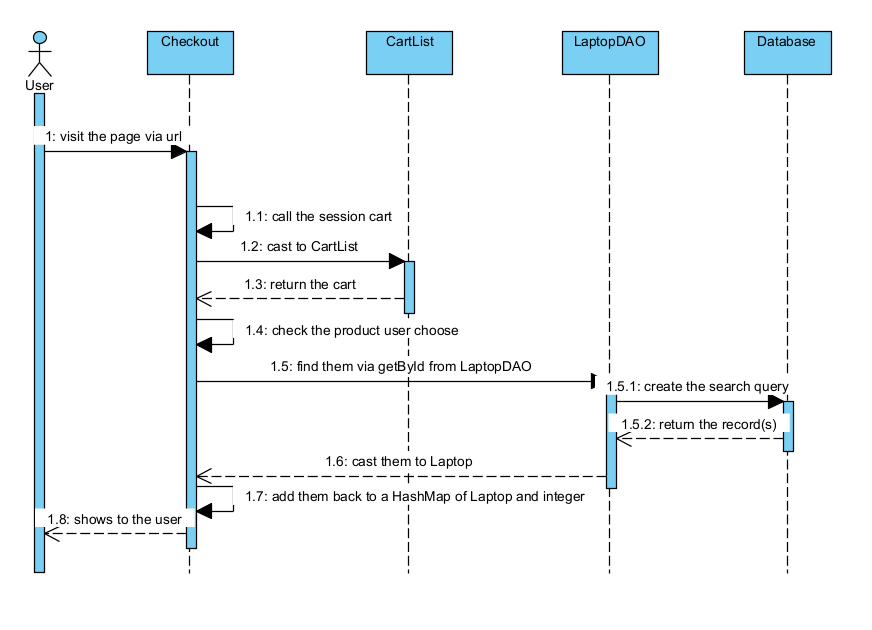
***LaptopDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getLaptopByID() | This method returns the Laptop that has the latop\_id equals to the id it contains  Input: int id  Output: Laptop l  Internal Processing: it will fetch from database the return a laptop that corresponding via jdbc and return the first laptop it meets(because id is unique for each laptop |

**Checkout Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | This method fetch the ids user choose and display them, allow user to view their chosen product, quantity, and the details for order  Input: HttpServletRequest req, HttpServletResponse res  Output: Showing the checkout page to user  Internal Processing: This method call the Cartlist to get the cart, pick what user choose by comparing it with id, and convert them to Laptop via getLaptopById, and displaying them with checkout.jsp |

### 3. Sequences Diagram



### 4. Sql Query

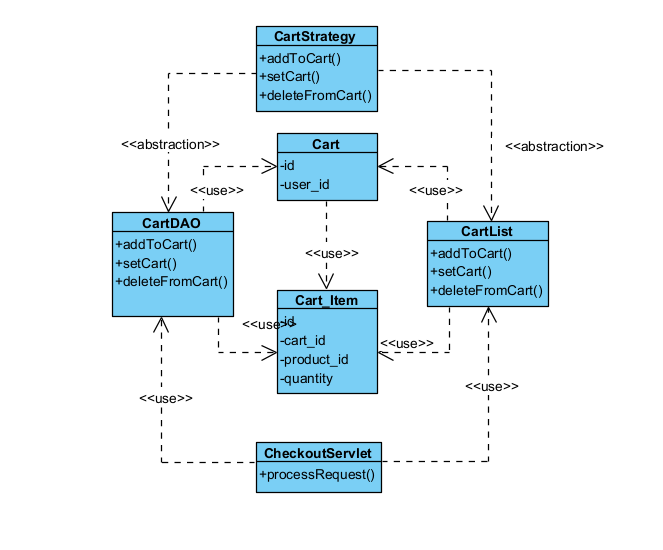
/INSERT INTO [Order] values (?,?,?,?,?,?)  
/INSERT INTO [Order\_User] values (?,?,?,?,?)

/INSERT INTO [Order\_Item] values (?,?,?,?)

/Update [Order] SET status = ? WHERE id = ?

## 14, Cart Details

### 1. Class Diagram



### 2. Class Specification

***Cart Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getCartID() | get a value for the cart\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | getCartID() | set a value for the cart\_id attribute |
| *04* | getUserID() | set a value for the email attribute |

***Cart\_Item Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getID() | get a value from the cart\_id attribute |
| *02* | getCartID() | get a value from the user\_id attribute |
| *03* | getProductId() | get value from the product\_id |
| *04* | getQuantity | get value from the quantity |
| *05* | setD() | set a value for the cart\_id attribute |
| *06* | setCartID() | set a value for the email attribute |
| *07* | setProductId() | set a value for the email attribute |
| *08* | seQuantity() | set a value for the email attribute |

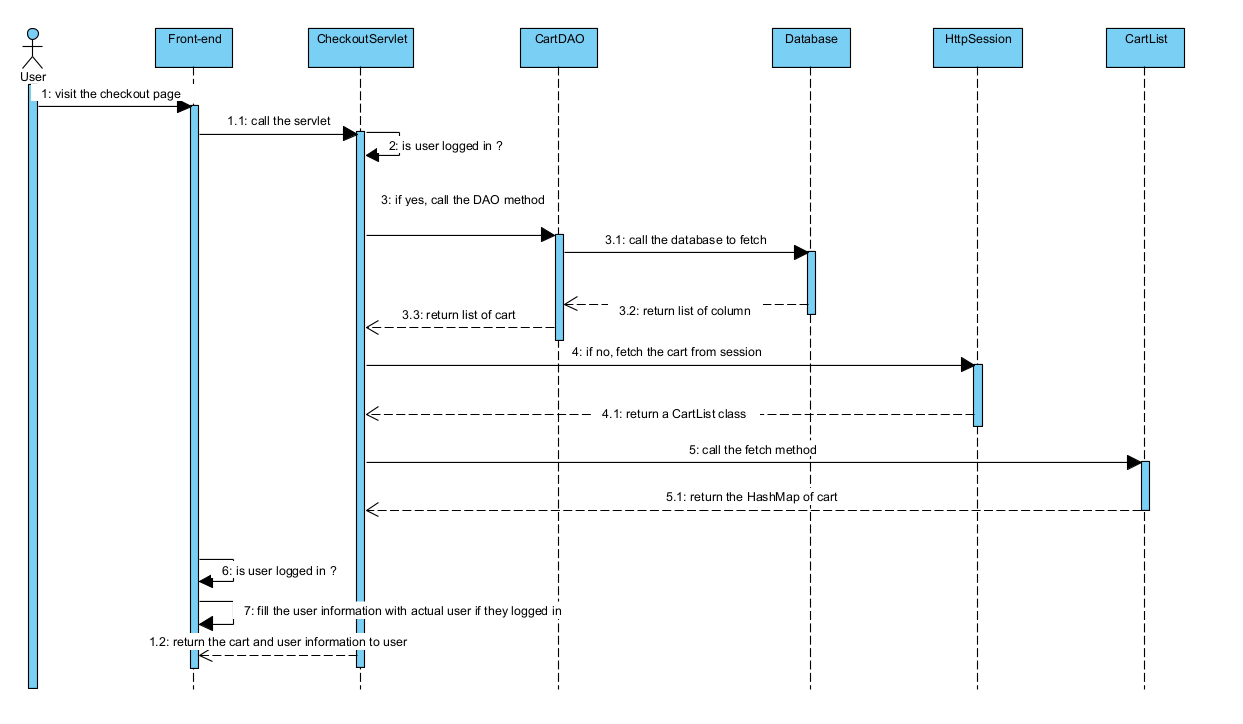
***CartList***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addToCart() | it add the product with it’s quantity to the HashMap session, or updating one if it’s already in the map  input: Laptop laptop, int quantity  output: the list of HashMap session with new item  Internal Processing: it check by search for the key, if it’s empty, add the new Laptop as key and quantity as it value, otherwise update the existing one |
| *02* | setCart() | it update one product that exist  Input: Laptop laptop, int quantity  Output: update the cart session  Internal Processing: It edit the quantity of a certain corresponding key |
| *03* | deleteFromCart() | It delete one entity  Input: Laptop laptop  Output: remove the cart from the map  Internal Processing: It find and remove the Laptop entity by the key |
| *04* | getCart() | It show the HashMap out  input: none  output: the HashMap with Laptop and it’s quantity  Internal Processing: it returns the HashMap |

***CartDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | addToCart() | it add the product with it’s quantity to the HashMap session, or updating one if it’s already in the map  input: Laptop laptop, int quantity  output: add the item and quantity to the Database  Internal Processing: it check by search for the key, if it’s empty, add the new Laptop as key and quantity as it value, otherwise update the existing one |
| *02* | setCart() | it update one product that exist  Input: Laptop laptop, int quantity  Output: update the cart session  Internal Processing: It edit the quantity of a certain corresponding key |
| *03* | deleteFromCart() | It delete one entity  Input: Laptop laptop  Output: remove the cart from the map  Internal Processing: It find and remove the Laptop entity by the key |
| *04* | getCart() | It show the HashMap out  input: none  output: the HashMap with Laptop and it’s quantity  Internal Processing: it returns the HashMap |

### 3. Sequences Diagram



### 4. SQL Script

SELECT cart\_id FROM Carts WHERE user\_id = ?

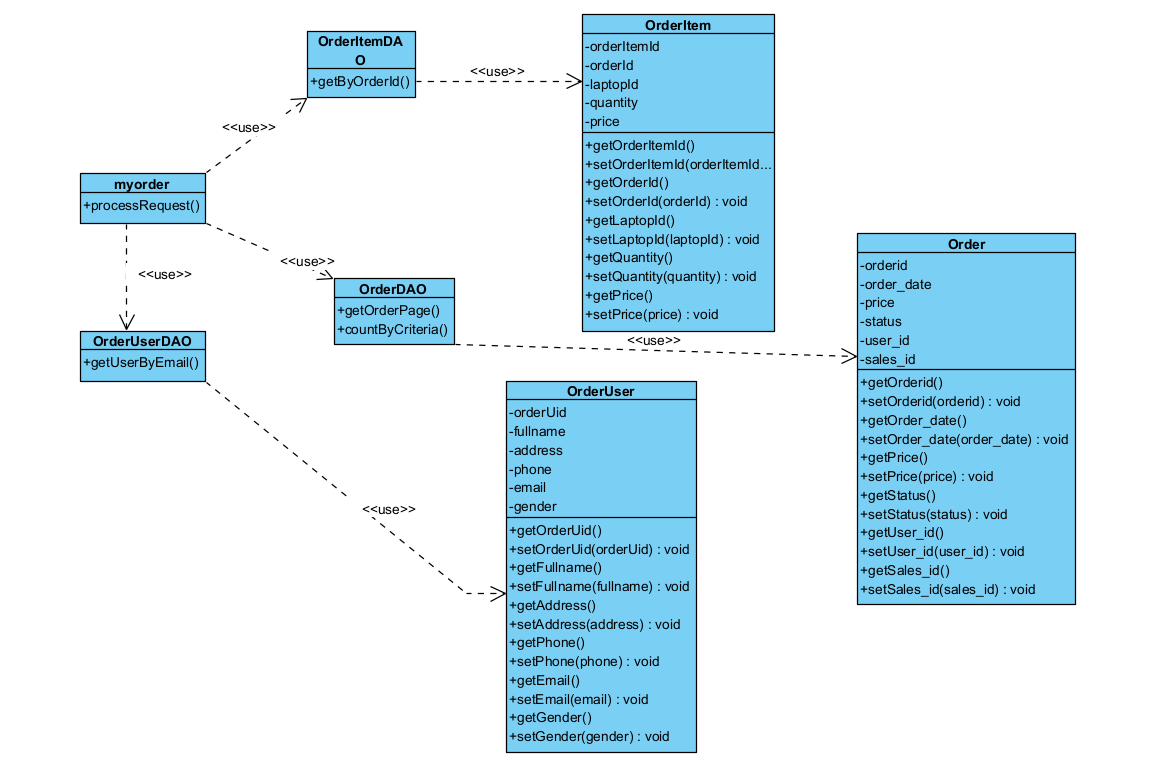
INSERT INTO Carts (user\_id) VALUES (?)  
SELECT quantity FROM cart\_items WHERE cart\_id = ? AND laptop\_id = ?

UPDATE cart\_items SET quantity = ? WHERE cart\_id = ? AND laptop\_id = ?

INSERT INTO cart\_items (cart\_id, laptop\_id, quantity) VALUES (?, ?, ?)  
UPDATE cart\_items SET quantity = ? WHERE cart\_id = ? AND laptop\_id = ?  
DELETE FROM cart\_items WHERE cart\_id = ? AND laptop\_id = ?  
SELECT \* FROM cart\_items WHERE cart\_id = ?

## 15. MyOrder

### 1. Class Diagram



### 2. Class Specification

**Order Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***OrderUser Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |
| *09* | getGender() | return value of gender |
| *10* | setGender() | assign a value for gender |

***OrderItem Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | setOrderID() | get a value for the order\_id attribute |
| *03* | getLaptop() | get a value for the laptop attribute |
| *04* | setLaptop() | set a value for the laptop attribute |
| *05* | getPrice() | return price of the order |
| *06* | setPrice() | assign a value to the price of an order |
| *07* | getQuantity() | return quantity of laptop in an order |
| *08* | setQuantity() | assign a value for quantity of laptop |

***OrderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByPage() | This method return a list of constant elements, called “page” for pagination  Input: int page, int totalPerPage, String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return a list of page orders  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, provide a stimulate pagination |
| *02* | countByCriteria() | This method count how many order(s) fit the criteria  Input: String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return number of order fit the criteria  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, then add to the database, retrieve number of rows exist, return them |

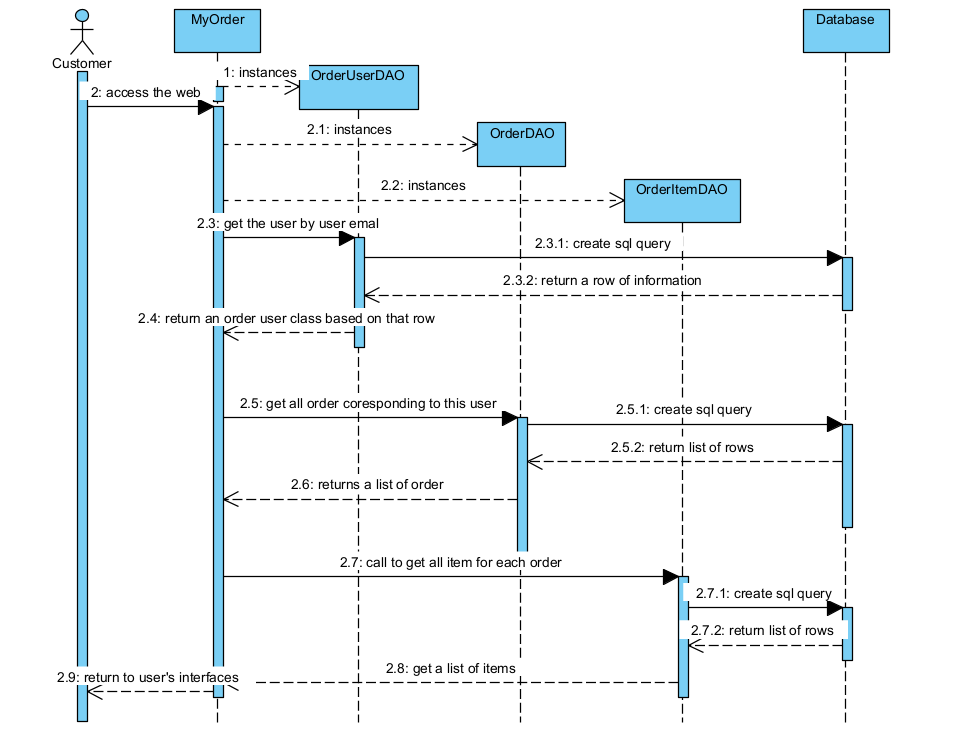
***OrderUserDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserByEmail | This method return an order customer based on the string email  Input: String email  Output: an order user  Internal Processing: This method using PreparedStatement and ResultSet to communicate with SQL, check if there’s an row after executing the query, fetch them as an order\_user if exist, or return null |

***OrderItemDAOClass:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByOrderID() | This method return a list of product + their quantity, price via the orderid  Input: int orderId  Output: a list of Order\_Item  Internal Processing: This method using PreparedStatement and ResultSet to communicate with SQL, check if there’s an row after executing the query, fetch them each as an List of OrderItem if there’s, otherwise return null. |

### 3. Sequences Diagram

****

### 4. SQLQuery

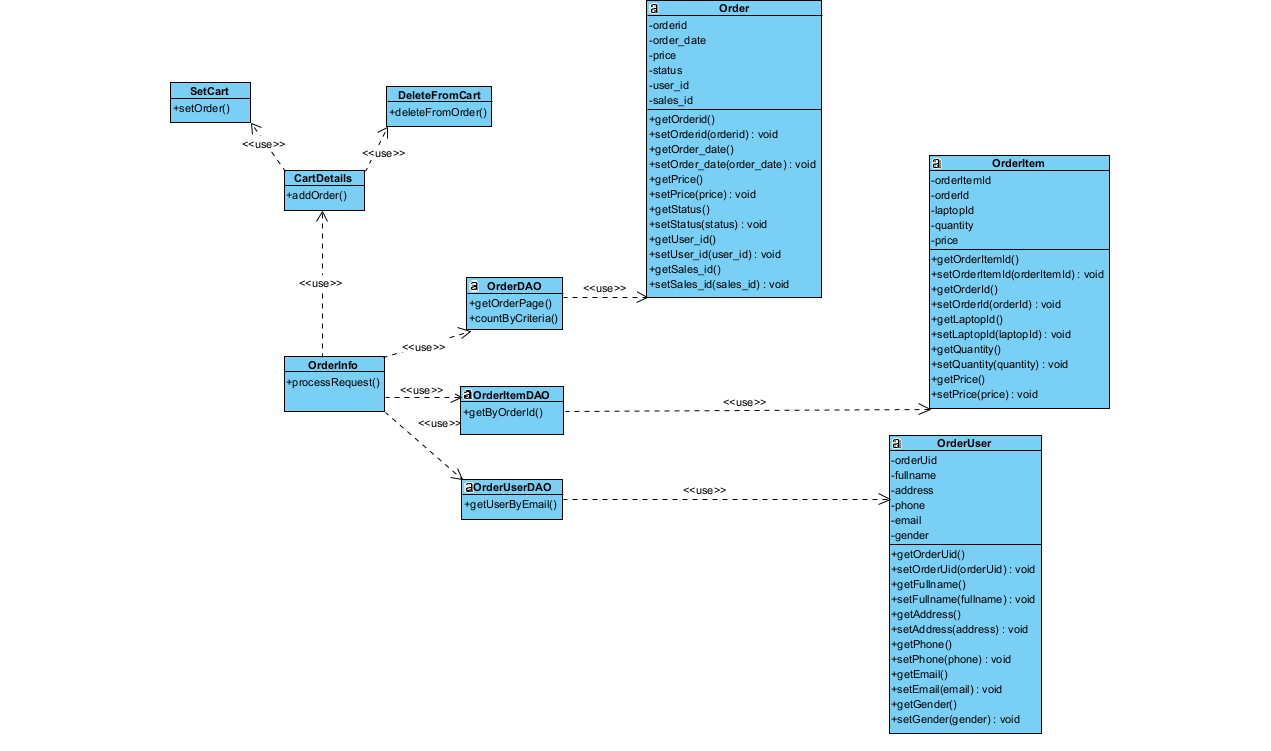
SELECT \* FROM [Order] where order\_uid = ?

SELECT \* FROM [Order\_Item] WHERE order\_id = ?

SELECT \* FROM Order\_User WHERE order\_uid = ?

## 16. Order Information

### 1. Class Diagram



### 2. Class Specification

**Order Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***OrderUser Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |
| *09* | getGender() | return value of gender |
| *10* | setGender() | assign a value for gender |

***OrderItem Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | setOrderID() | get a value for the order\_id attribute |
| *03* | getLaptop() | get a value for the laptop attribute |
| *04* | setLaptop() | set a value for the laptop attribute |
| *05* | getPrice() | return price of the order |
| *06* | setPrice() | assign a value to the price of an order |
| *07* | getQuantity() | return quantity of laptop in an order |
| *08* | setQuantity() | assign a value for quantity of laptop |

***OrderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByPage() | This method return a list of constant elements, called “page” for pagination  Input: int page, int totalPerPage, String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return a list of page orders  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, provide a stimulate pagination |
| *02* | countByCriteria() | This method count how many order(s) fit the criteria  Input: String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return number of order fit the criteria  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, then add to the database, retrieve number of rows exist, return them |

***OrderUserDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserByEmail | This method return an order customer based on the string email  Input: String email  Output: an order user  Internal Processing: This method using PreparedStatement and ResultSet to communicate with SQL, check if there’s an row after executing the query, fetch them as an order\_user if exist, or return null |

***OrderItemDAOClass:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByOrderID() | This method return a list of product + their quantity, price via the orderid  Input: int orderId  Output: a list of Order\_Item  Internal Processing: This method using PreparedStatement and ResultSet to communicate with SQL, check if there’s an row after executing the query, fetch them each as an List of OrderItem if there’s, otherwise return null. |

**Order Class:**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | getUserID() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setUserID() | set a value for the user\_id attribute |
| *05* | getPrice() | return payment for an order |
| *06* | setPrice() | assign a value to the payment of an order |
| *07* | getStatus() | get a value for the status attribute |
| *08* | setStatus() | set a value for the status attribute |

***OrderUser Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserID() | get a value for the order\_id attribute |
| *02* | getName() | get a value for the user\_id attribute |
| *03* | setUserID() | set a value for the order\_id attribute |
| *04* | setName() | set a value for the user\_id attribute |
| *05* | getPhone() | return payment for an order |
| *06* | setPhone() | assign a value to the payment of an order |
| *07* | getAddress() | return value for address |
| *08* | setAddress() | assign value for address |
| *09* | getGender() | return value of gender |
| *10* | setGender() | assign a value for gender |

***OrderItem Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getOrderID() | get a value for the order\_id attribute |
| *02* | setOrderID() | get a value for the order\_id attribute |
| *03* | getLaptop() | get a value for the laptop attribute |
| *04* | setLaptop() | set a value for the laptop attribute |
| *05* | getPrice() | return price of the order |
| *06* | setPrice() | assign a value to the price of an order |
| *07* | getQuantity() | return quantity of laptop in an order |
| *08* | setQuantity() | assign a value for quantity of laptop |

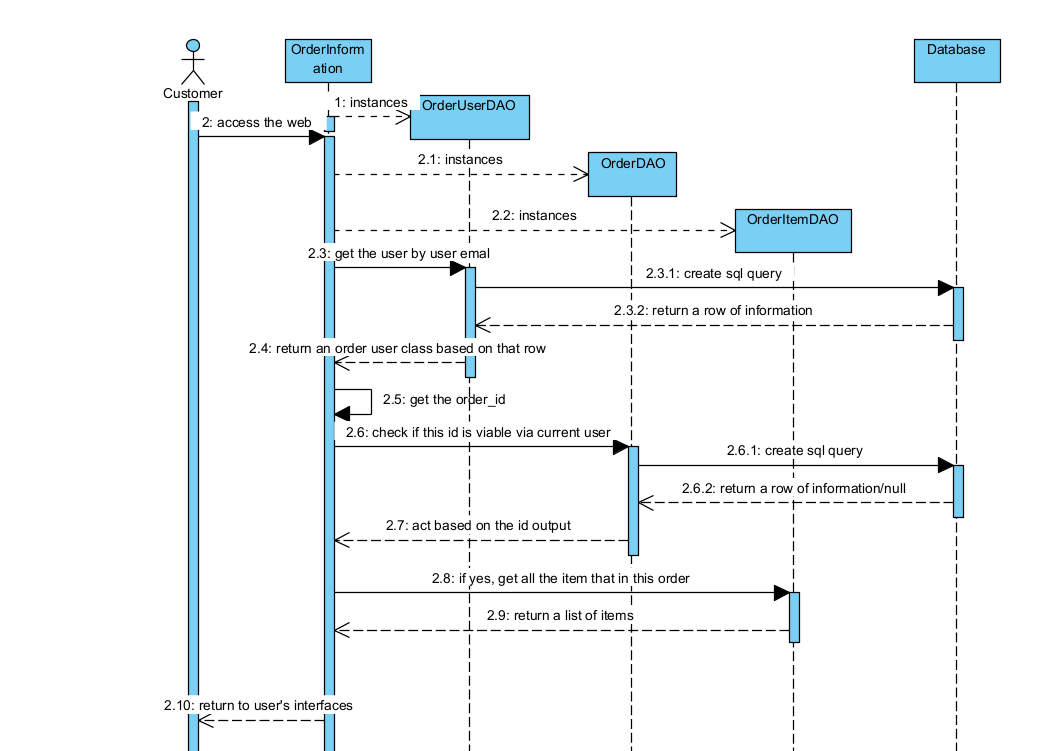
***OrderDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getByPage() | This method return a list of constant elements, called “page” for pagination  Input: int page, int totalPerPage, String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return a list of page orders  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, provide a stimulate pagination |
| *02* | countByCriteria() | This method count how many order(s) fit the criteria  Input: String order\_by, String direction, String search\_query, Date start, Date end, String status, String[] selectedLaptopId  Output: return number of order fit the criteria  Internal Processing: This method gather if any information is exist, and won’t add if they’re not presences, then add to the database, retrieve number of rows exist, return them |

***OrderUserDAO Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getUserByEmail | This method return an order customer based on the string email  Input: String email  Output: an order user  Internal Processing: This method using PreparedStatement and ResultSet to communicate with SQL, check if there’s an row after executing the query, fetch them as an order\_user if exist, or return null |

### 3. Sequences Diagram



### 4. SQL Query

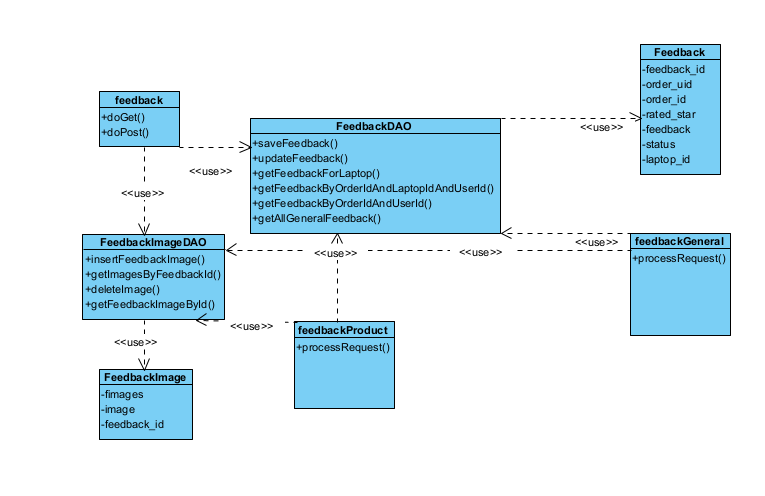
SELECT \* FROM [Order] where order\_uid = ?

SELECT \* FROM [Order\_Item] WHERE order\_id = ?

SELECT \* FROM Order\_User WHERE order\_uid = ?

## 17. Feedback

### 1. Class Diagram



### 2. Class Specification

***Feedback Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getFeedback\_id() | get a value for the Feedback\_id attribute |
| *02* | setFeedback\_id() | set a value for the Feedback\_id attribute |
| *03* | getOrder\_uid() | get a value for the Order\_uid attribute |
| *04* | setOrder\_uid() | set a value for the Order\_uid attribute |
| *05* | getOrder\_id() | get a value for the Order\_id attribute |
| *06* | setOrder\_id() | set a value for the Order\_id attribute |
| *07* | getRated\_star() | get a value for the Rated\_star attribute |
| *08* | setRated\_star() | set a value for the Rated\_star attribute |
| *09* | getFeedback() | get a value for the Feedback attribute |
| *10* | setFeedback() | set a value in the Feedback attribute |
| *11* | getStatus() | get a value for the Status attribute |
| *12* | setStatus() | set a value in the Status attribute |
| *13* | getLaptop\_id() | get a value for the Laptop\_id attribute |
| *14* | setLaptop\_id() | set a value in the Laptop\_id attribute |

***FeedbackImage Class:***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | getFimages() | get a value for the Fimages attribute |
| *02* | setFimages() | set a value for the Fimages attribute |
| *03* | getImage() | get a value for the Image attribute |
| *04* | setImage() | set a value for the Image attribute |
| *05* | getFeedback\_id() | get a value for the Feedback\_id attribute |
| *06* | setFeedback\_id() | set a value for the Feedback\_id attribute |

***FeedbackDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | saveFeedback() | Save a new feedback entry in the database.  **Inputs**: orderUid, orderId, productId, rating, feedbackText.  **Internal Processing**:   * Prepares an SQL INSERT statement. * Sets the parameters for order\_uid, order\_id, rated\_star, laptop\_id, feedback, and status. * Executes the update and retrieves generated keys.   **Output**: Returns the generated feedback ID if successful; otherwise, returns -1. |
| *02* | updateFeedback() | Update an existing feedback entry.  **Inputs:** feedbackId, rating, feedbackText.  **Internal Processing:**   * Prepares an SQL UPDATE statement. * Sets parameters for rated\_star and feedback. * Executes the update.   **Output:** None (performs an update operation). |
| *03* | getFeedbackForLaptop() | Retrieve feedback for a specific laptop.  **Inputs**: laptopId.  **Internal Processing**:   * Prepares an SQL SELECT statement to find feedback entries by laptop\_id. * Iterates over the result set, converting each row into a Feedback entity.   **Output**: A list of Feedback objects associated with the given laptop ID. |
| *04* | getFeedbackByOrderIdAndLaptopIdAndUserId() | Retrieve feedback for a specific order, laptop, and user.  **Inputs**: order\_id, laptop\_id, orderUid.  **Internal Processing**:   * Prepares an SQL SELECT statement using order\_id, laptop\_id, and order\_uid. * Executes the query and converts the first result row into a Feedback entity.   **Output**: A Feedback object if found; otherwise, null. |
| *05* | getFeedbackByOrderIdAndUserId() | Retrieve general feedback for a specific order and user.  **Inputs**: order\_id, orderUid.  **Internal Processing**:   * Prepares an SQL SELECT statement using order\_id and order\_uid with laptop\_id as null. * Executes the query and converts the first result row into a Feedback entity.   **Output**: A Feedback object if found; otherwise, null. |
| *06* | getAllGeneralFeedback() | Retrieve all general feedback without specific laptop associations.  **Inputs**: None.  **Internal Processing**:   * Prepares an SQL SELECT statement where laptop\_id is null. * Iterates over the result set, converting each row into a Feedback entity.   **Output**: A list of Feedback objects with laptop\_id as null. |

***FeedbackImageDAO Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | insertFeedbackImage() | Insert a new image path associated with a feedback entry.  **Inputs**: imagePath, feedbackId.  **Internal Processing**:   * Prepares an SQL INSERT statement for the Feedback\_Image table. * Sets parameters for image and feedback\_id. * Executes the update.   **Output**: None (inserts a new record). |
| *02* | getImagesByFeedbackId() | Retrieve images for a specific feedback entry.  **Inputs**: feedback\_id.  **Internal Processing**:   * Prepares an SQL SELECT statement to find images by feedback\_id. * Iterates over the result set, converting each row into a FeedbackImage entity.   **Output**: A list of FeedbackImage objects associated with the given feedback ID. |
| *03* | deleteImage() | Delete an image by its ID.  **Inputs**: imageId.  **Internal Processing**:   * Prepares an SQL DELETE statement for the Feedback\_Image table. * Sets the parameter for the image ID. * Executes the update and checks the number of rows affected.   **Output**: Returns true if an image was deleted; otherwise, false. |
| *04* | getFeedbackImageById() | Retrieve a specific image by its ID.  **Inputs**: imageId.  **Internal Processing**:   * Prepares an SQL SELECT statement using the image ID. * Executes the query and converts the first result row into a FeedbackImage entity.   **Output**: A FeedbackImage object if found; otherwise, null. |

***feedback Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | doGet() | Display the feedback form and any existing feedback for a specific order and product.  **Inputs**: email, orderId, productId from request parameters.  **Internal Processing**:   * Validates the email format. * Retrieves the user and their orders using OrderUserDAO and OrderDAO. * Collects items for each order using OrderItemDAO. * If orderId is provided, retrieves the selected order and its items. * If productId is present, checks for existing feedback using FeedbackDAO and associated images using FeedbackImageDAO. * Sets attributes for the JSP to render the form and existing feedback.   **Output**: Forwards to feedback.jsp. |
| *02* | doPost() | Handle submission of feedback and image management.  **Inputs**: email, orderId, productId, rating, feedbackText, feedbackId, action, and uploaded images.  **Internal Processing**:   * Retrieves the user using OrderUserDAO. * Checks if the request is for image deletion; deletes the image file and database entry if so. * Determines the upload directory and ensures it exists. * Updates existing feedback or creates new feedback using FeedbackDAO. * Processes uploaded images, saves them to disk, and records their paths in the database with FeedbackImageDAO.   **Output**: Redirects to the referrer with success or error messages in the session. |

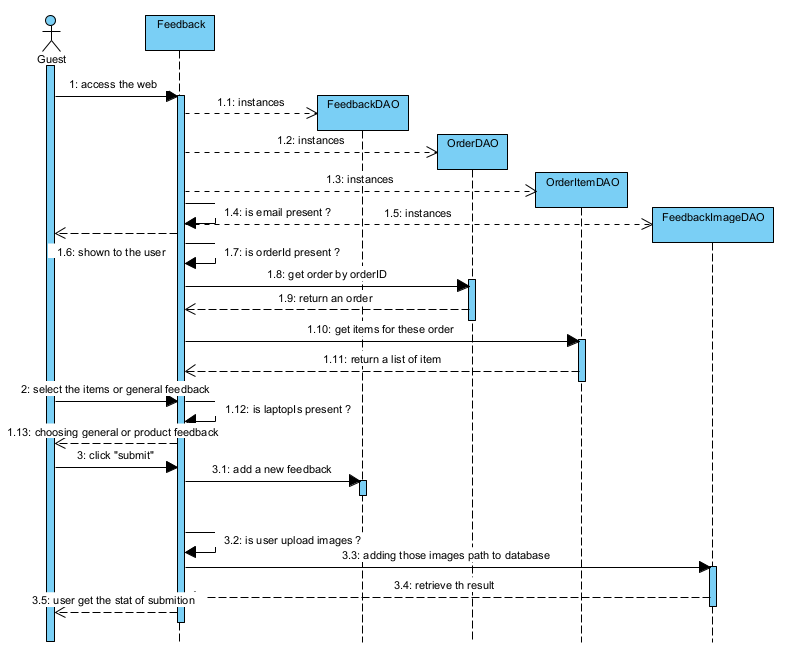
***feedbackGeneral Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | Handle both GET and POST requests to fetch, validate, and display feedback.  **Internal Processing**:   * **Feedback Retrieval**: Uses FeedbackDAO to get all general feedback. * **Validation**:   + Filters feedback to ensure ratings are between 1 and 5.   + Ensures feedback text is not empty. * **Organization**:   + Creates a map to associate feedback with images using FeedbackImageDAO.   + Organizes feedback by the user with OrderUserDAO. * **Sorting**: Sorts the feedback by rating in descending order.   **Output**: Sets the sorted feedback list as a request attribute and forwards the request to customer-review.jsp. |

***feedbackProduct Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | processRequest() | Handle both GET and POST requests to fetch, validate, and display feedback.  **Internal Processing**:   * **Feedback Retrieval**: Uses FeedbackDAO to get all feedback for 1 product * **Validation**:   + Filters feedback to ensure ratings are between 1 and 5.   + Ensures feedback text is not empty. * **Organization**:   + Creates a map to associate feedback with images using FeedbackImageDAO.   + Organizes feedback by the user with OrderUserDAO. * **Sorting**: Sorts the feedback by rating in descending order.   **Output**: Sets the sorted feedback list as a request attribute and forwards the request to customer-review.jsp. |

### 3. Sequences Diagram



### 4. SQL Query

**1/ Check if an email with an user exist**

INSERT INTO Feedback (order\_uid, order\_id, rated\_star, laptop\_id, feedback, status) VALUES (?, ?, ?, ?, ?, ?)

UPDATE Feedback SET rated\_star = ?, feedback = ? WHERE feedback\_id = ?

**2/Update Feedback**

SELECT \* FROM Feedback WHERE laptop\_id = ?

**3/General Feedabck**

SELECT \* FROM Feedback WHERE order\_id = ? and laptop\_id = ? and order\_uid = ?

**4/ Display General Feedback**

SELECT \* FROM Feedback WHERE laptop\_id is null

**5/ Add Images**

INSERT INTO Feedback\_Image (image, feedback\_id) VALUES (?, ?)

**6/Delete Images**

DELETE FROM Feedback\_Image WHERE fimages = ?

**7/Get Images**

SELECT \* FROM Feedback\_Image WHERE feedback\_id = ?