

**<<Fashion Shop>>**

**Software Design Specification**

– Hanoi, August 2022 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_heading=h.gjdgxs)

[1. Code Packages 4](#_heading=h.30j0zll)

[2. Database Design 4](#_heading=h.1fob9te)

[a. Database Schema 4](#_heading=h.3znysh7)

[b. Table Description 4](#_heading=h.2et92p0)

[II. Code Designs 5](#_heading=h.tyjcwt)

[1. <Feature/Function Name1> 5](#_heading=h.3dy6vkm)

[a. Class Diagram 5](#_heading=h.1t3h5sf)

[b. Class Specifications 5](#_heading=h.4d34og8)

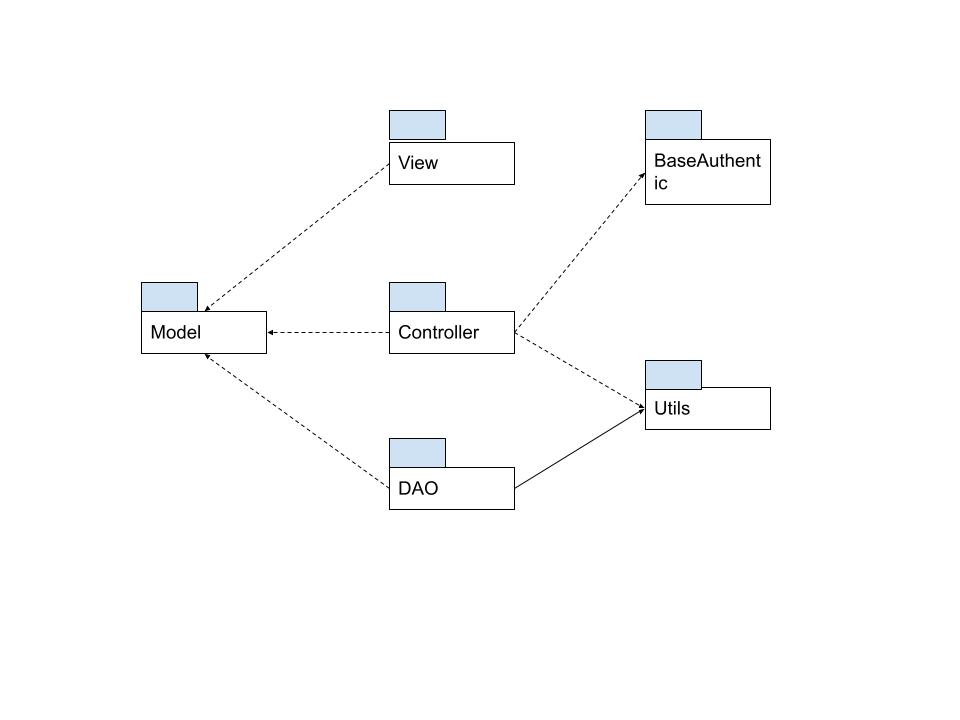
[c. Sequence Diagram(s) 5](#_heading=h.2s8eyo1)

[d. Database queries 6](#_heading=h.17dp8vu)

[2. <Feature/Function Name2> 6](#_heading=h.3rdcrjn)

# I. Overview

## 1. Code Packages



***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *01* | *Model* | *Contain* |
| *02* | *DAO* | *Contains classes that relevant to Database execution* |
| *03* | *Controller* | *implements business logic,* |
| *04* | *Utils* | *Contains sql queries, get DBConnection,* |
| *05* | *BaseAuthentic* | *Check if the account is authenticated to get access or not* |
| *06* | *View* | *Contain frond-end code, which displayed on screens* |
| *07* | *…* |  |

## 

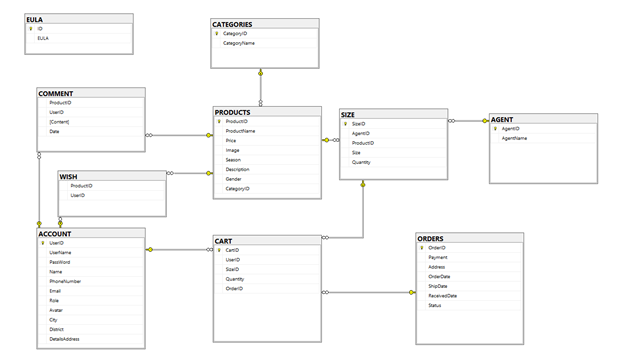
## 

## 

## 

## 2. Database Design

### a. Database Schema



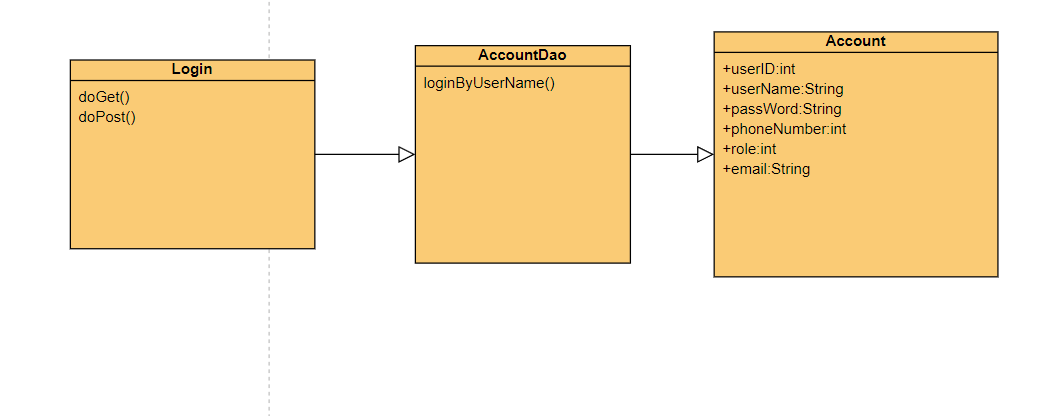
### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *ACCOUNT* | *Contains customer and employee account data* |
| *02* | *AGENT* | *Contains the agent id and name* |
| *03* | *CART* | *Contains customer cart information* |
| *04* | *CATEGORIES* | *Contains categories of items* |
| *05* | *COMMENTS* | *Customer reviews of each product by day* |
| *06* | *EULA* | *Contains information about the store's policy* |
| *07* | *ORDERS* | *Customer's ordering and payment information* |
| *08* | *PRODUCTS* | *Information of each product in store* |
| *09* | *SIZE* | *Information on the size and quantity of each product* |
| *10* | *WISH* | *Contains products that customers are interested in* |

# II. Code Designs

## 1. <Login>

### a. Class Diagram



### 

### b. Class Specifications

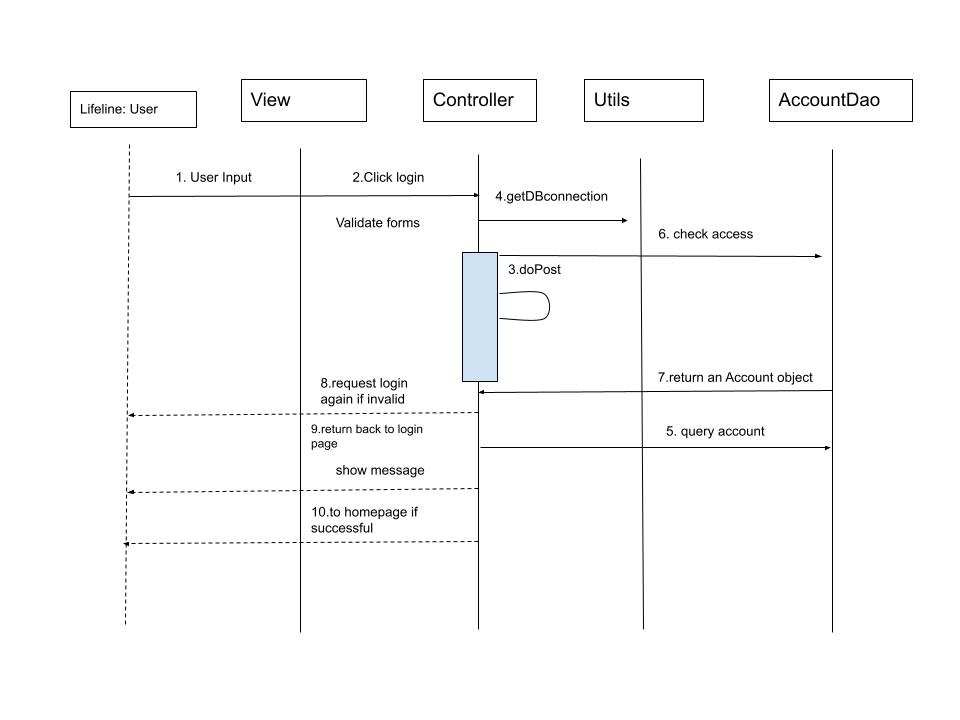
#### Account DAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *LoginbyUserName* | *Inputs: String username, String password*  *Output: Account object*  *internal processing: get the two inputs into login-query, that will read from the database the account that has username and password like two inputs. Otherwise, return null;* |
|  |  |  |

#### Login Servlet

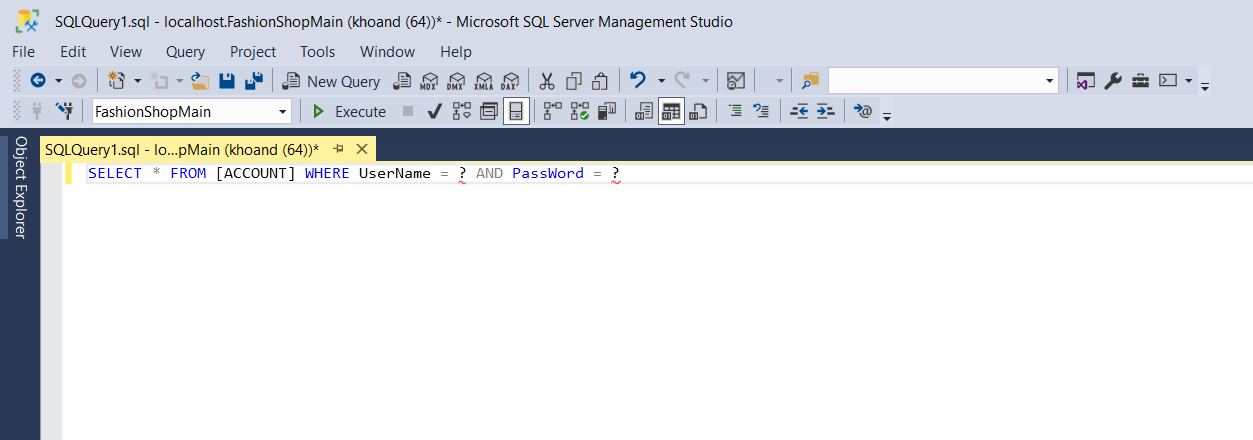
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *OutPut: void*  *Internal processing: send accessor to Login page* |
| *02* | *doPost* | *Inputs: HttpServlet Request, HttpServletResponse*  *OutPut: void*  *Internal Processing: get username and password that user input from forms. Then pass that two values into LoginByUserName function. If result return null: send back to Login page and show Loginfail message. Else, login successful and send to Homepage* |

### c. Sequence Diagram(s)



### d. Database Queries

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



## 

## 

## 

## 

## 2. <SignUp>

***a, Class Diagram***

***b***, ***Class specification***

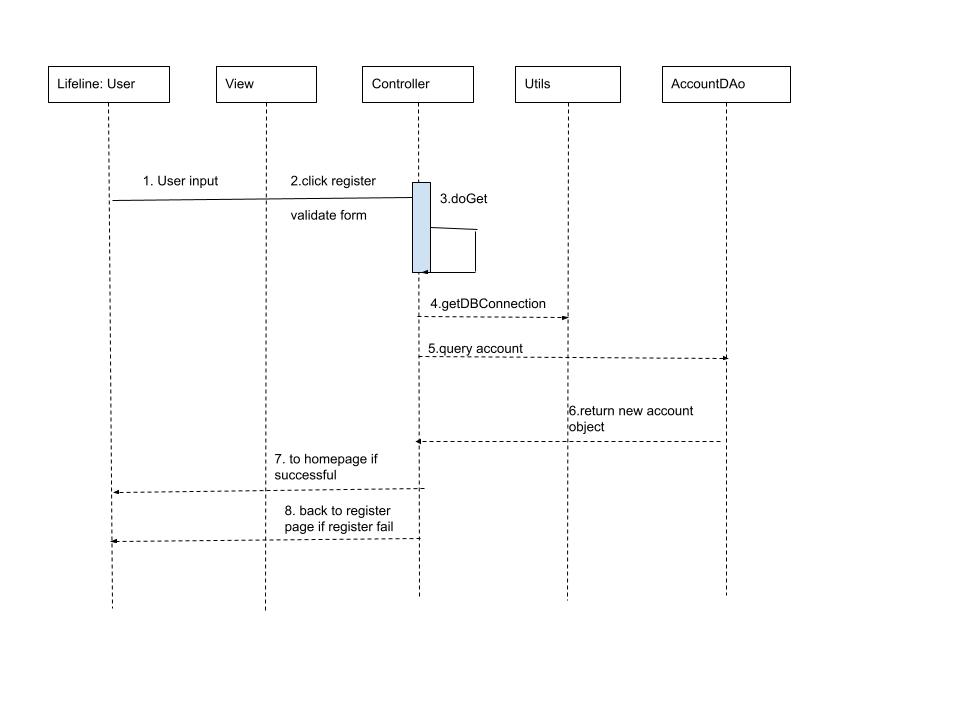
***AccountDao***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *SignUp* | *Inputs: Account object*  *Output: int*  *internal processing: get inputs from Account object, then set them to Signup query. Results return 1 if insert into database successful, else return 0* |
|  |  |  |

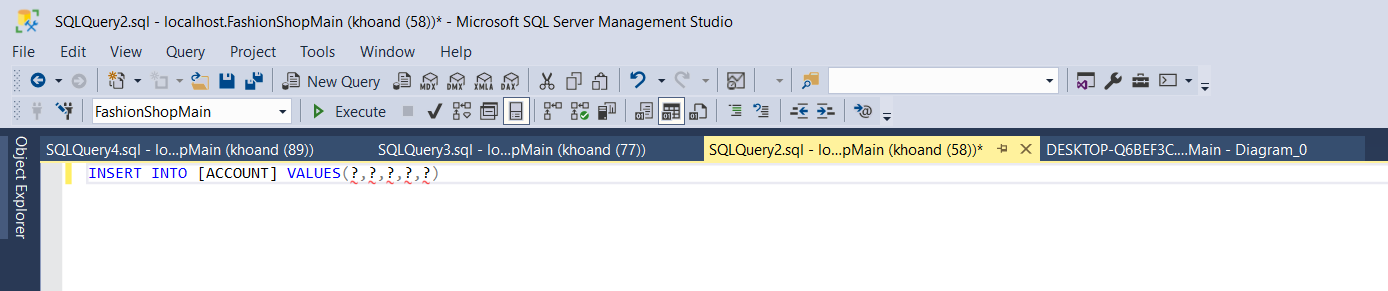
***SignUp servlet***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: get inputs from Register forms, then set them to new Account Object. After that, pass it to signup method. Finally, send to Login page to login by that account* |
|  |  |  |

***c, Sequence Diagrams***



***d, Database Query***



**3, Load Products and Category**

***a, Class Diagram***

***b, Class specification***

***setUpservlet***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: get inputs from Register forms, then set them to new Account Object. After that, pass it to signup method. Finally, send to Login page to login by that account* |
|  |  |  |

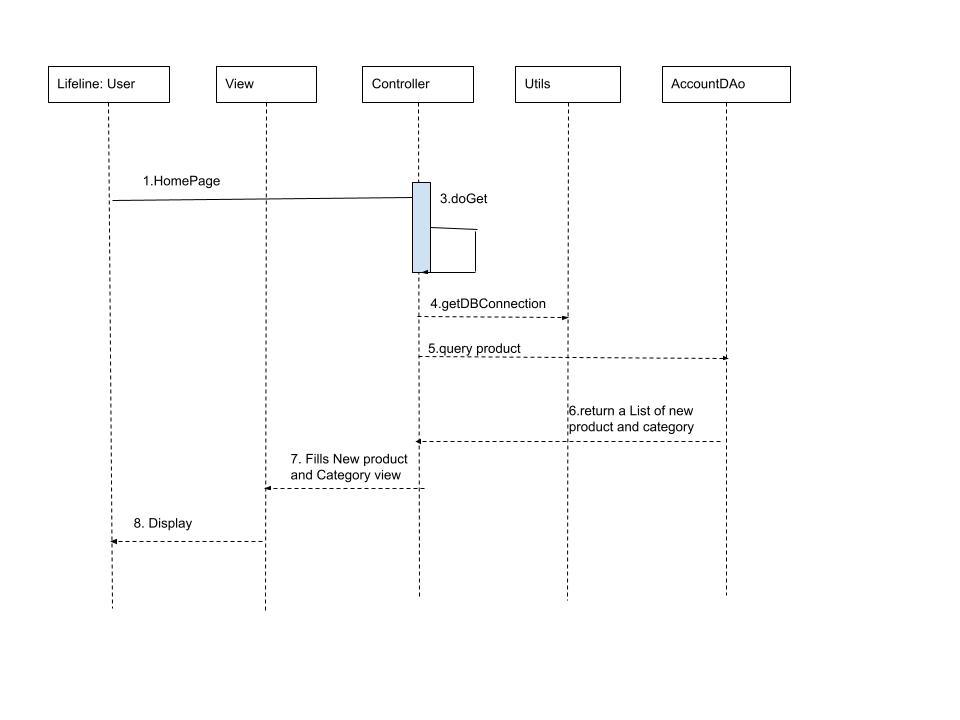
***ProductDao***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getNewProduct* | *Inputs: none*  *Output: ArrayList<Product>*  *internal processing: execute query to get records Product order by id, then read them from the database, add to the arrayList* |
|  |  |  |

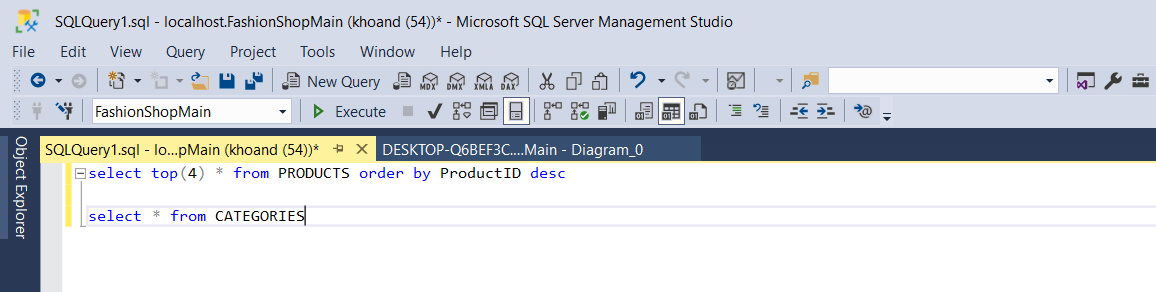
***CategoryDao***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getAllCategoryName* | *Inputs: none*  *Output: ArrayList<Product>*  *internal processing: execute query to get records Category, then read them from the database, add to the arrayList* |
|  |  |  |

***c, Sequence Diagrams***



***d, Database Query***



***4, Product Details***

***a, Class Diagram***

***b, Class specification***

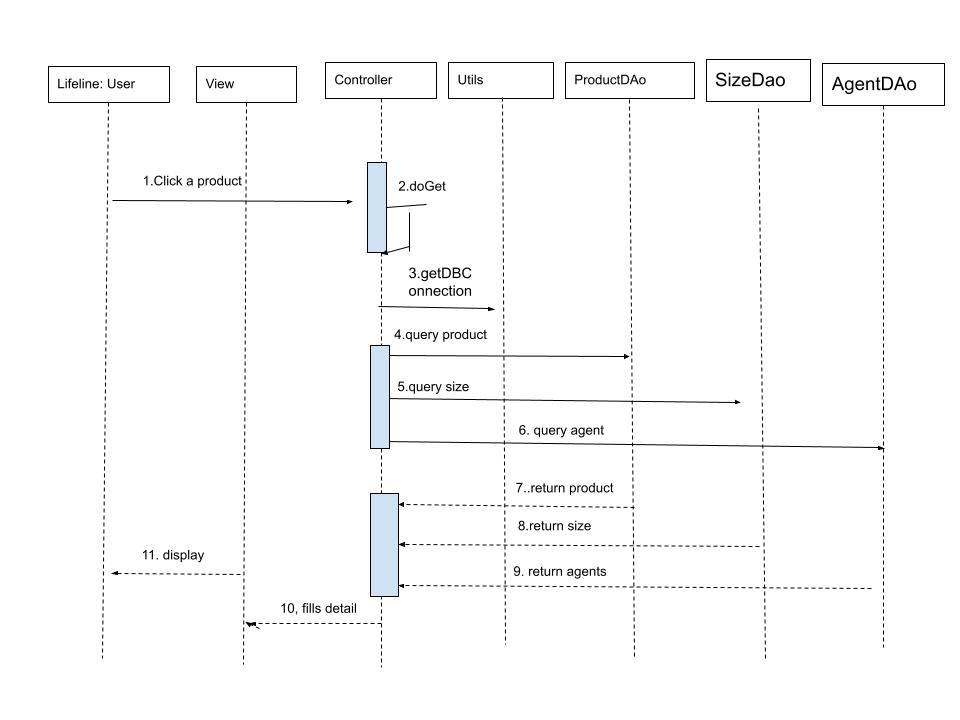
***ProductDao***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getNewProduct* | *Inputs: none*  *Output: ArrayList<Product>*  *internal processing: execute query to get records Product order by id, then read them from the database, add to the arrayList* |
| *02* | *getProductByProductId* | *Inputs: productId*  *Output: Product Object*  *Intenal processing: pass the productId to the query, then read the record from the database to create a Product object* |
| *03* | getSizebyProductId | *Inputs: productId*  *OutPut: an ArrayList<Size>*  *Internal processing: pass the productId to the query, then reaf the records from the database, create new Size object and add it to the list* |
| *04* | getAllAgents | *Inputs: none*  *Output: ArrayList<Product>*  *internal processing: execute query to get records Agent, then read them from the database, add to the arrayList* |

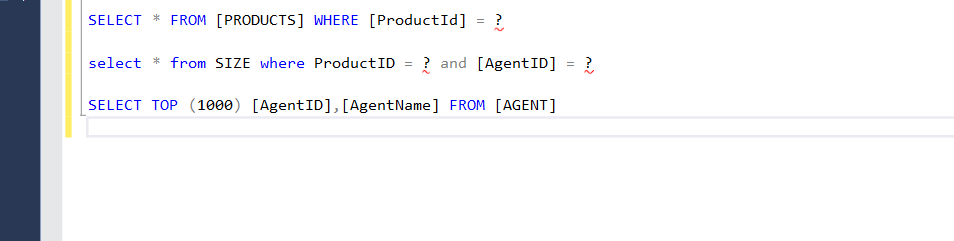
***productDetailServlet***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: getProductID from Url to access detail of that product, then get user inputs from the forms to pass it to above function. Finally, send it to detailProduct page* |
|  |  |  |

***c, Sequence Diagram***



d, Database Query



## 

## 5. Add to cart

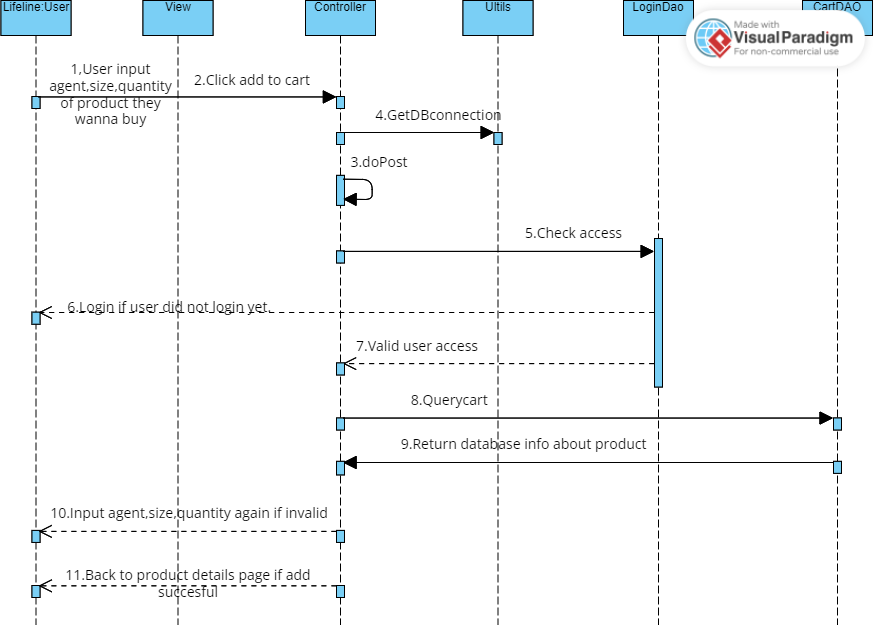
a, Class diagram

b, Class specification

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: getProductID from Url to access detail of that product, then get user inputs from the forms to pass it to above function. Finally, send it to detailProduct page* |
|  |  |  |

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getNewProduct* | *Inputs: none*  *Output: ArrayList<Product>*  *internal processing: execute query to get records Product order by id, then read them from the database, add to the arrayList* |
|  |  |  |

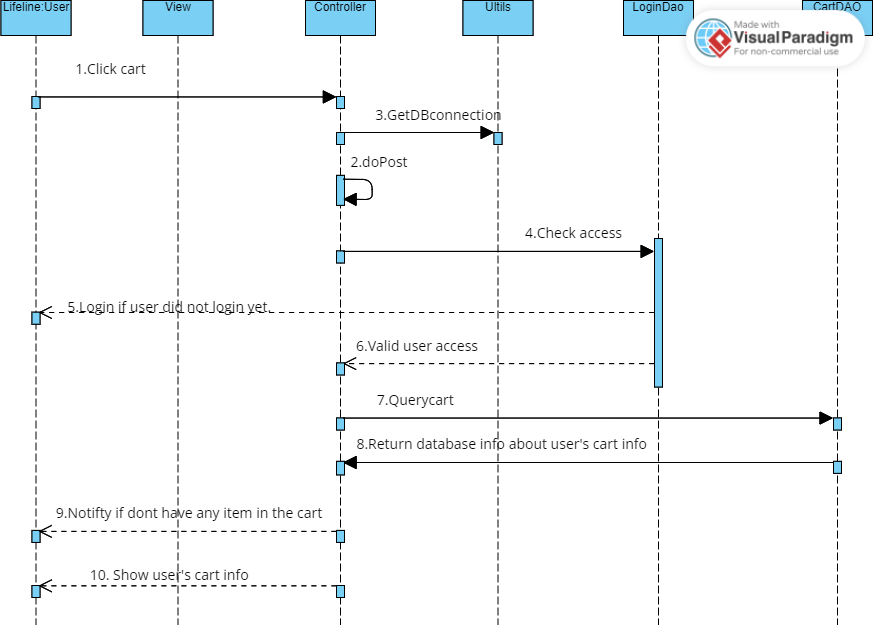
c, Sequence Diagram

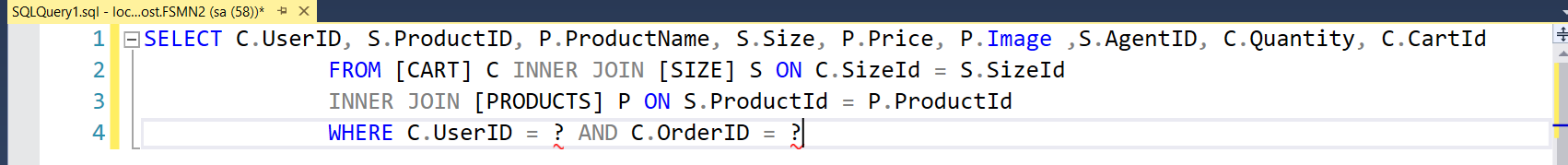


# d, Database query

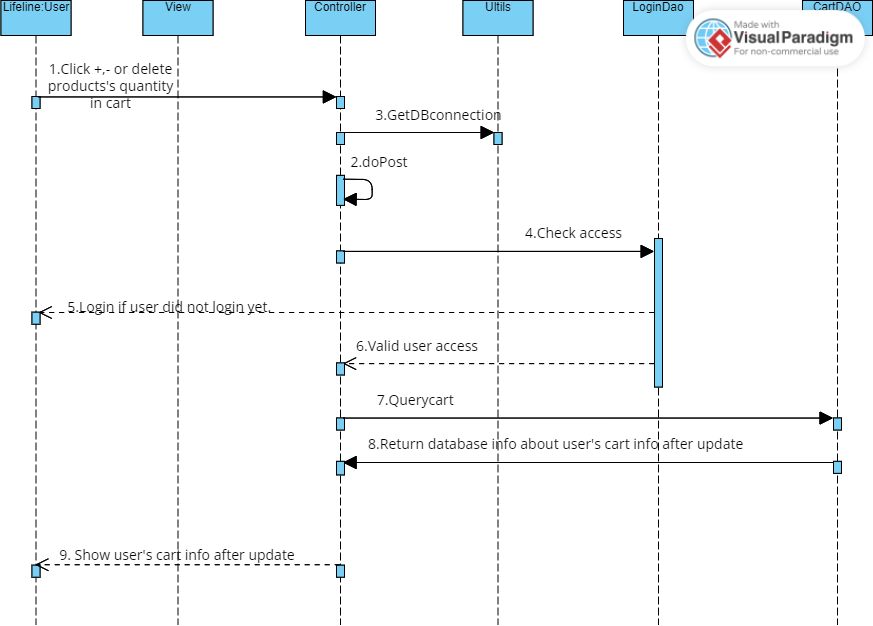


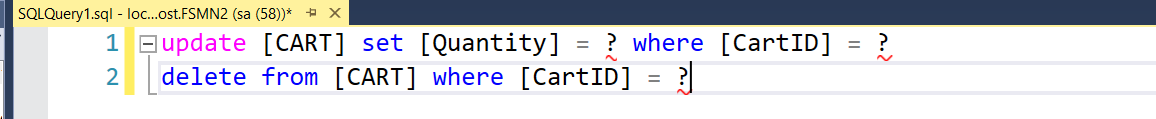
6, ViewCart





7,CRUD Cart





8, Load Product according to Agents

a, Class Diagram

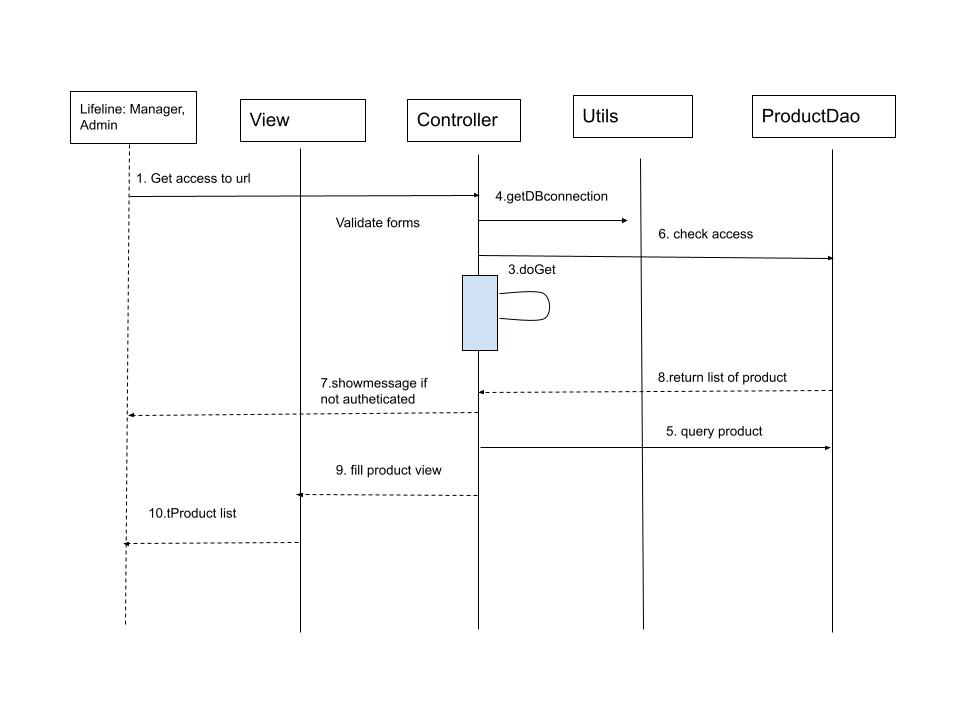
b, Class specification

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: getAgentID from Url to pass it to getProductsByAid function, return a list of product then fill views with those product* |
|  |  |  |

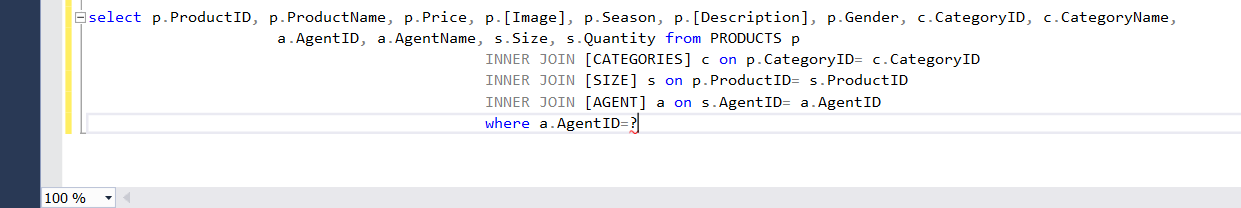
ProductDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getProductBiAid* | *Inputs: int*  *Output: ArrayList*  *internal processing: getAgentID from Url to pass it to getProductsByAid function, return a list of product then fill views with those product* |
|  |  |  |

c, Sequence Diagram

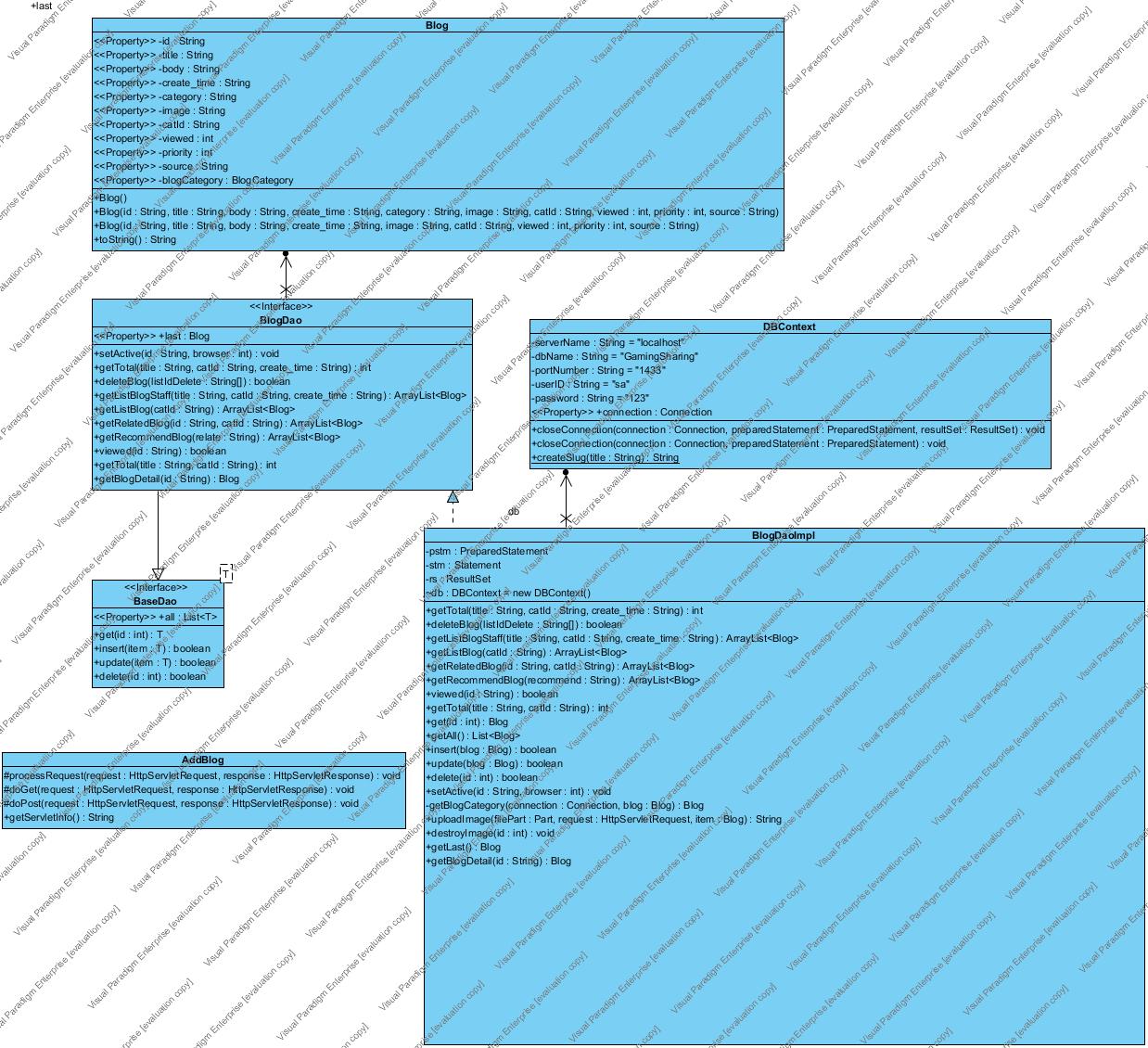


d, Database Diagram



9, AddProduct

a, Class diagram



b, Class specification

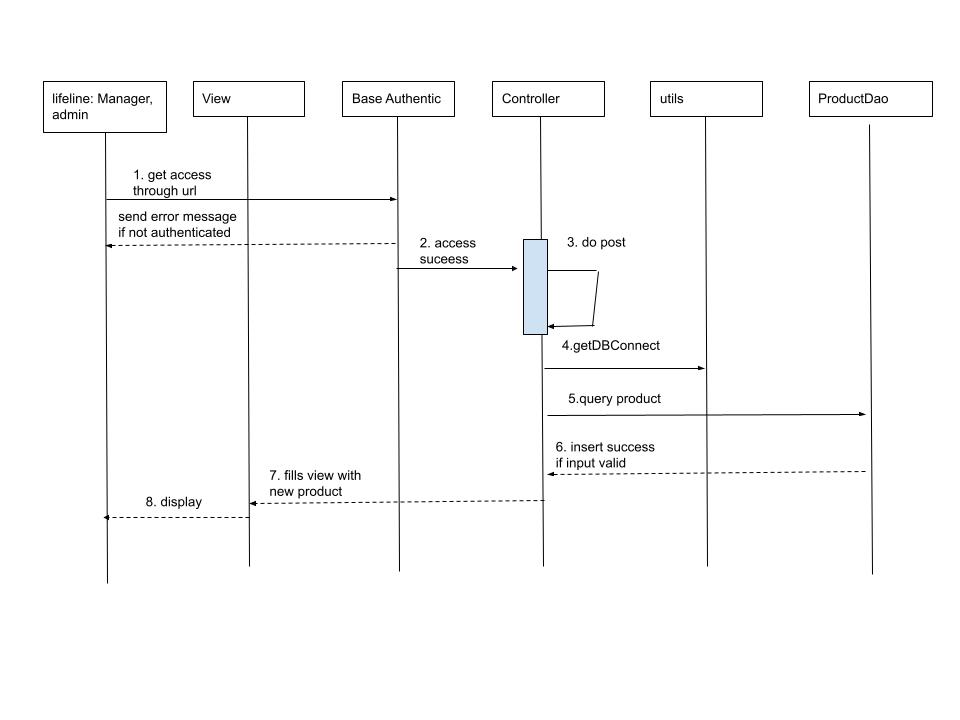
*InsertServlet*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *doGet* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: getAgentID from Url to pass it to* |
|  |  |  |

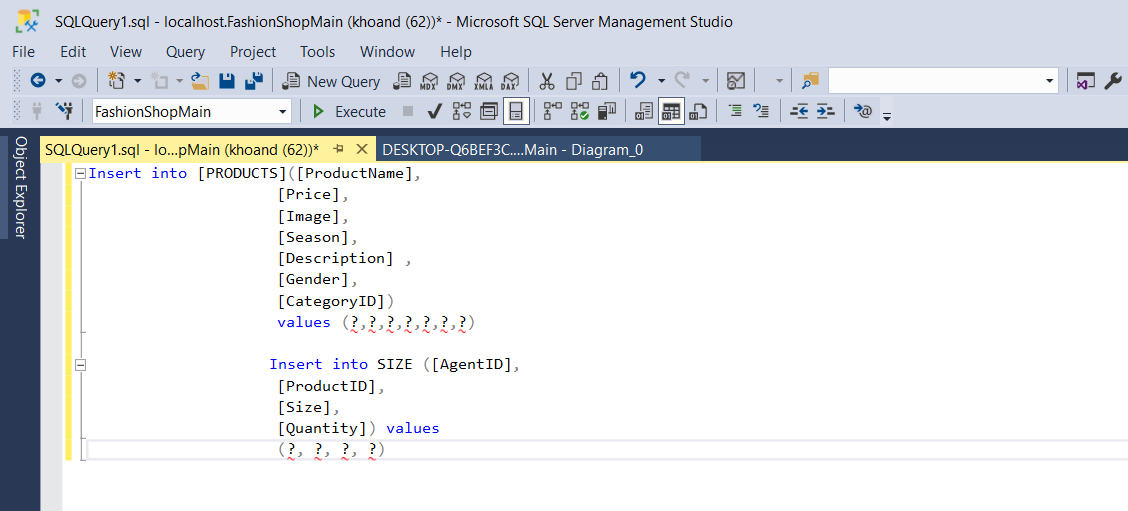
ProductDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getMaxIDProduct* | *Inputs: HttpServlet Request, HttpServletResponse*  *Output: void*  *internal processing: getAgentID from Url to pass it to* |
| *02* | *Insert* |  |

c, Sequence diagram

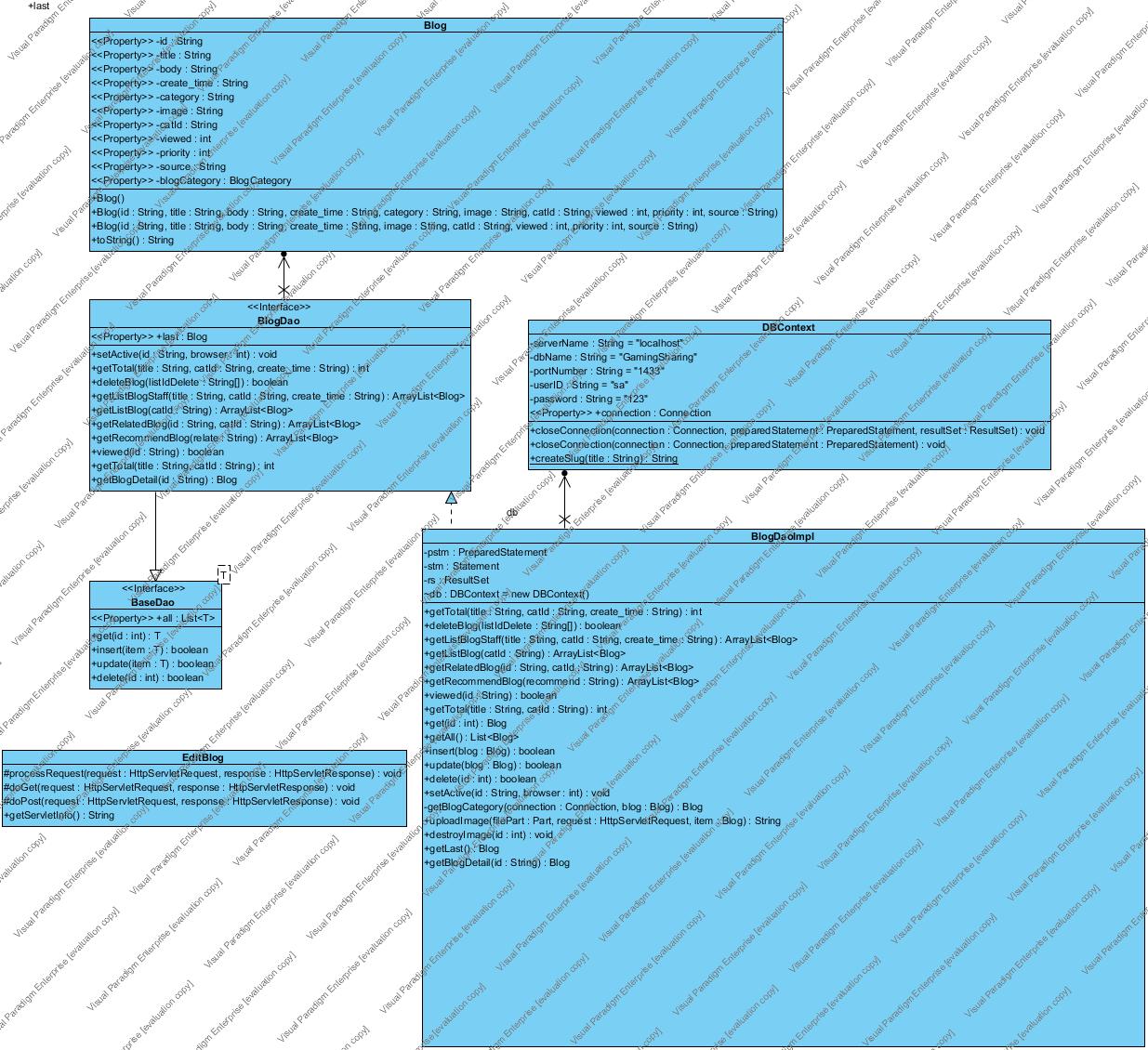


d, Database query



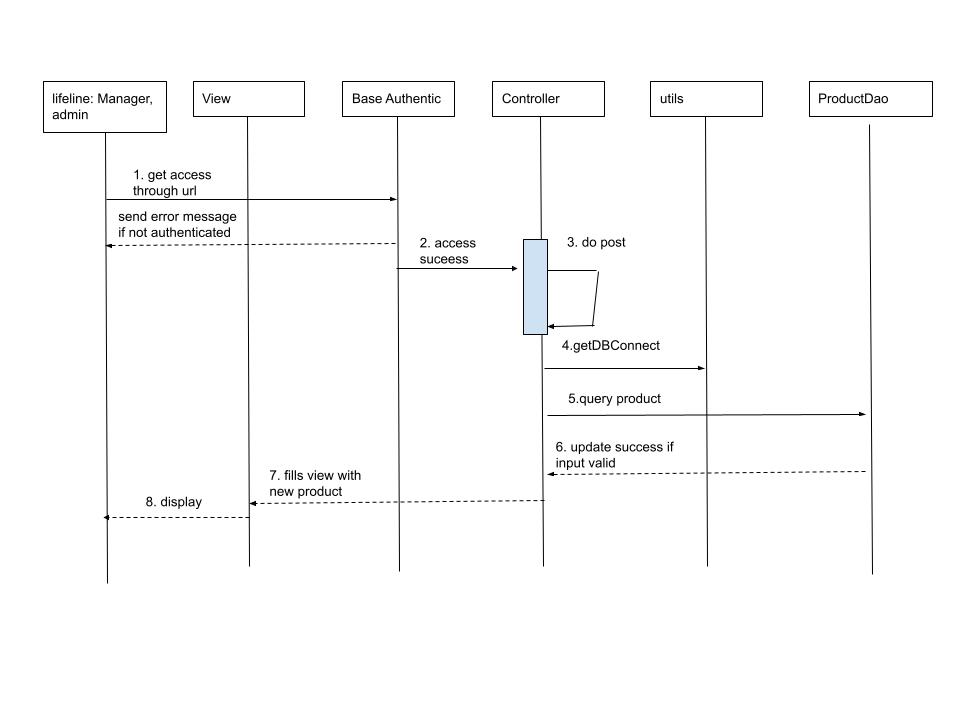
10. Update product

a, Class diagram

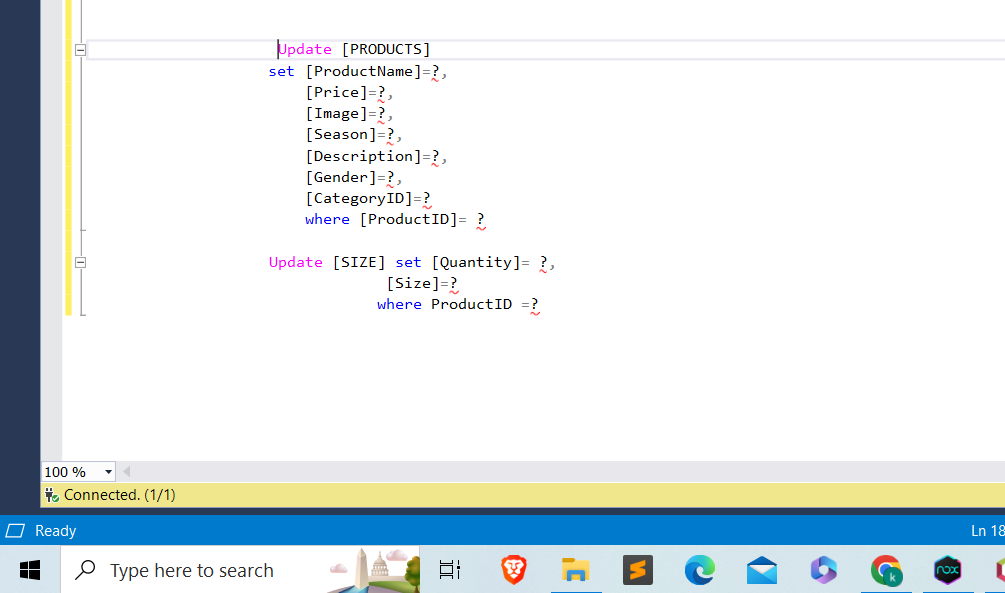


b, Class specification

c, Sequence diagram



d, Database query

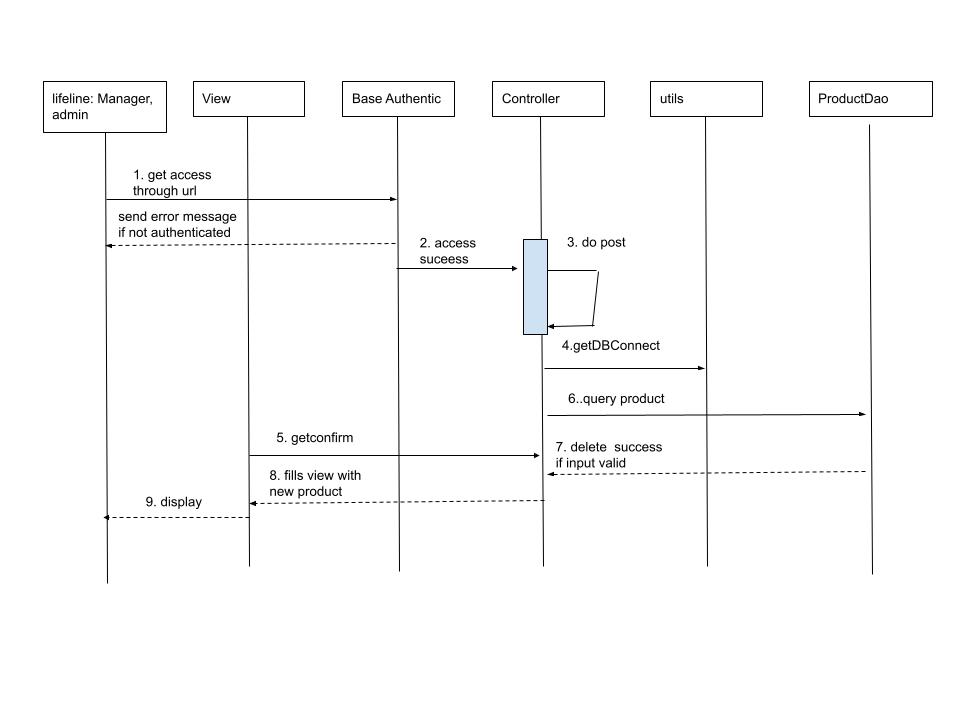


11, Delete product

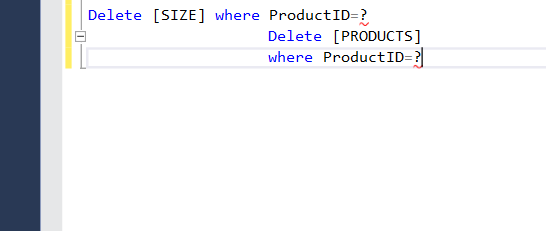
a, Class diagram

b, Class specification

c, Sequence diagram



d, Database query

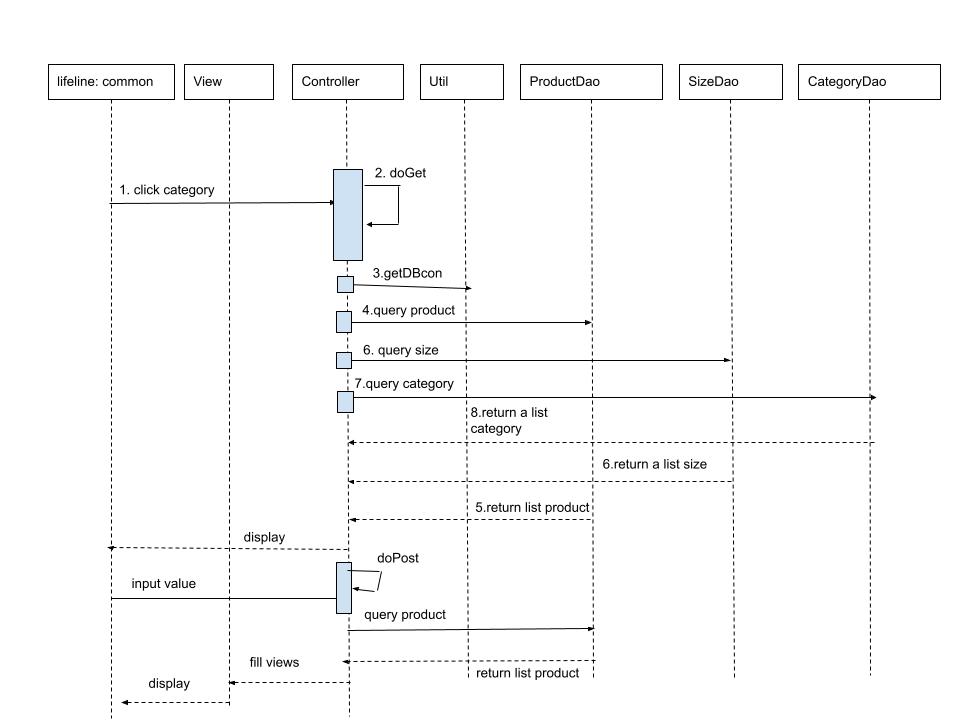


12, Filter product

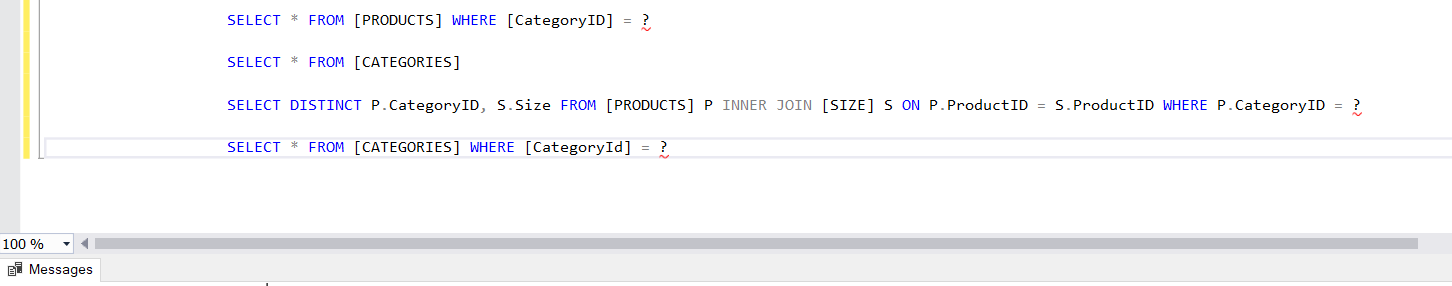
a, Class diagram

b, Class specification

c, Sequence diagram



d, Database query

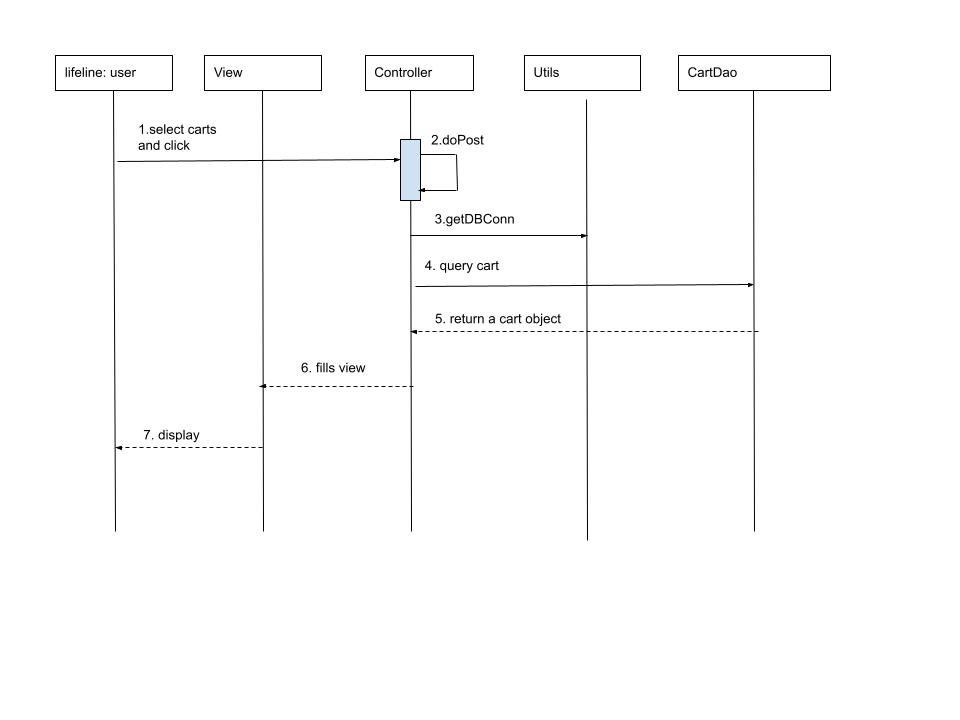


13, Submit cart to order

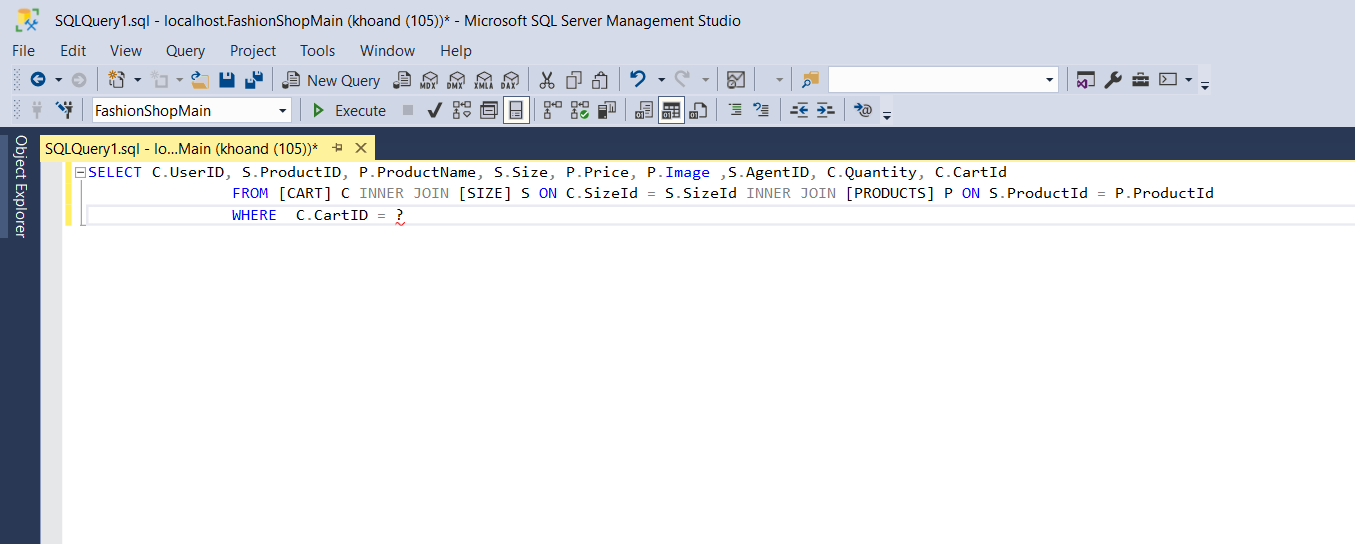
a, Class diagram

b, Class specification

c, Sequence diagram



d, Database query

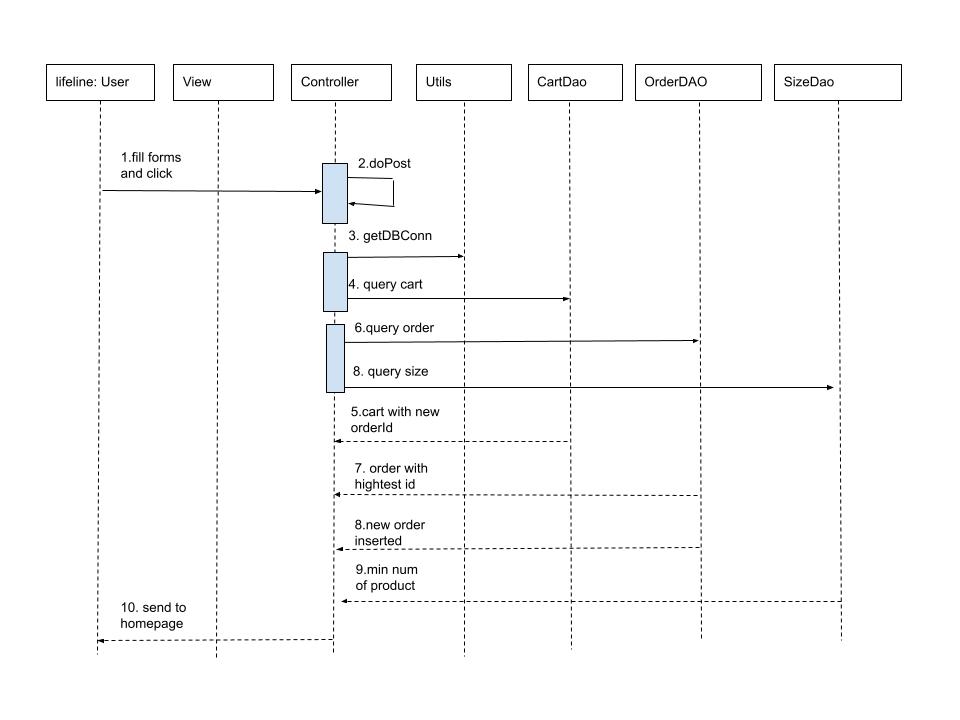


14. Submit Order

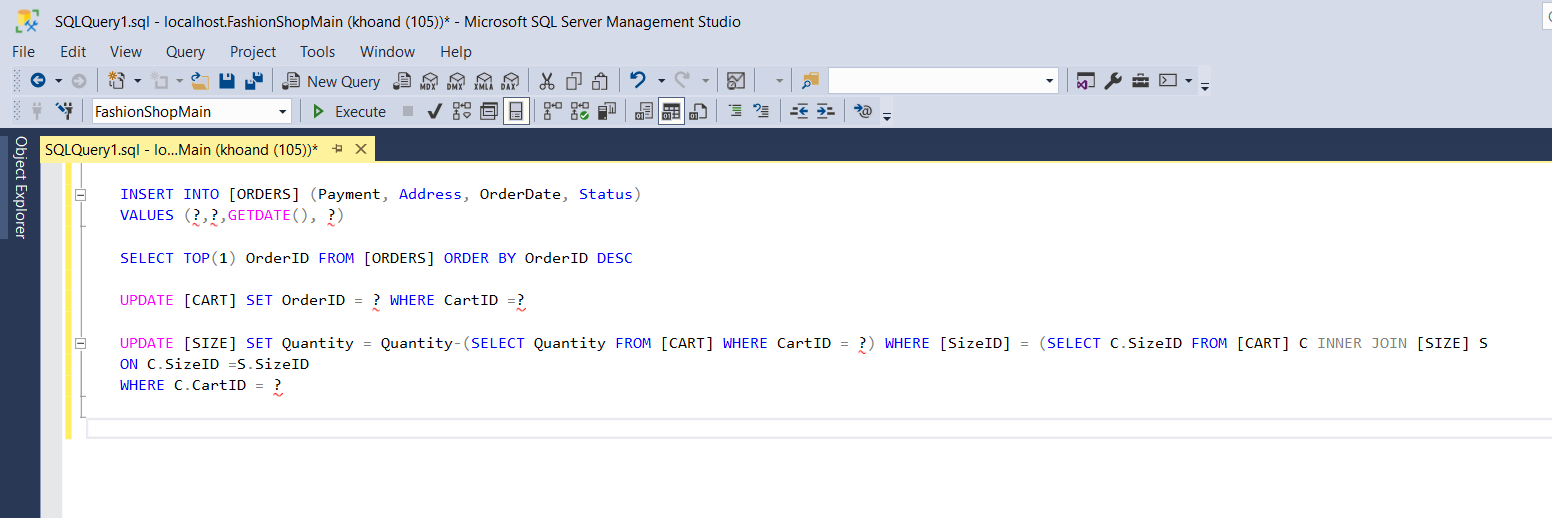
a, Class diagram

b, Class specification

c, Sequence diagram



d, Database query

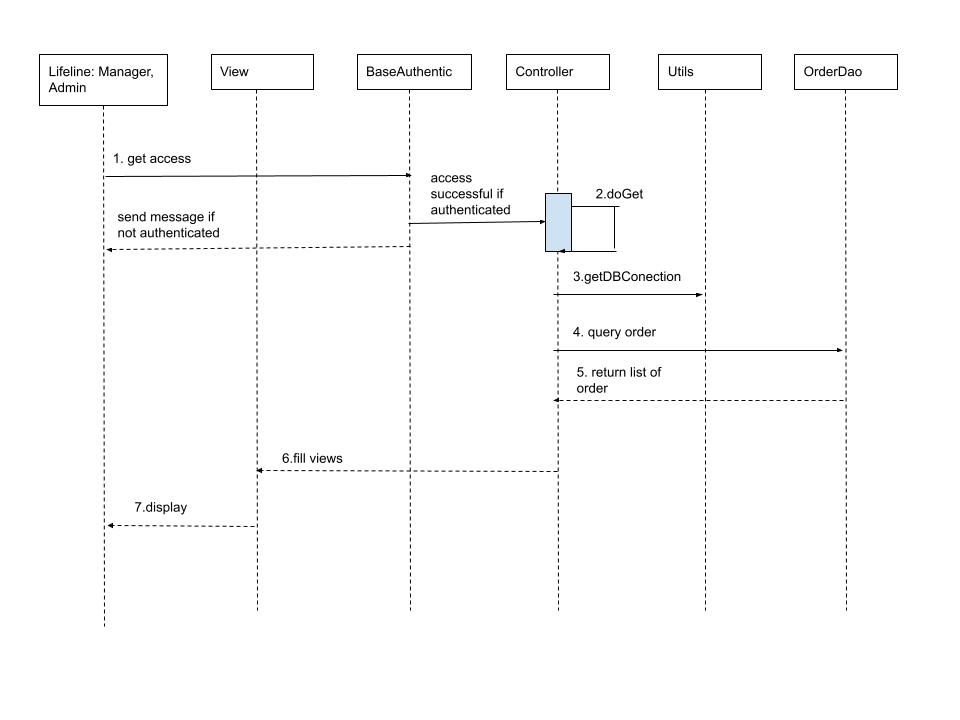


15, View Order

a, Class diagram

b, Class specification

c, Sequence diagram



d, Database query

