TRƯỜNG ĐẠI HỌC FPT

# DIETARY SUPPLIMENTALS SHOP

## Software Design Document

– Can Tho, February 2023 –

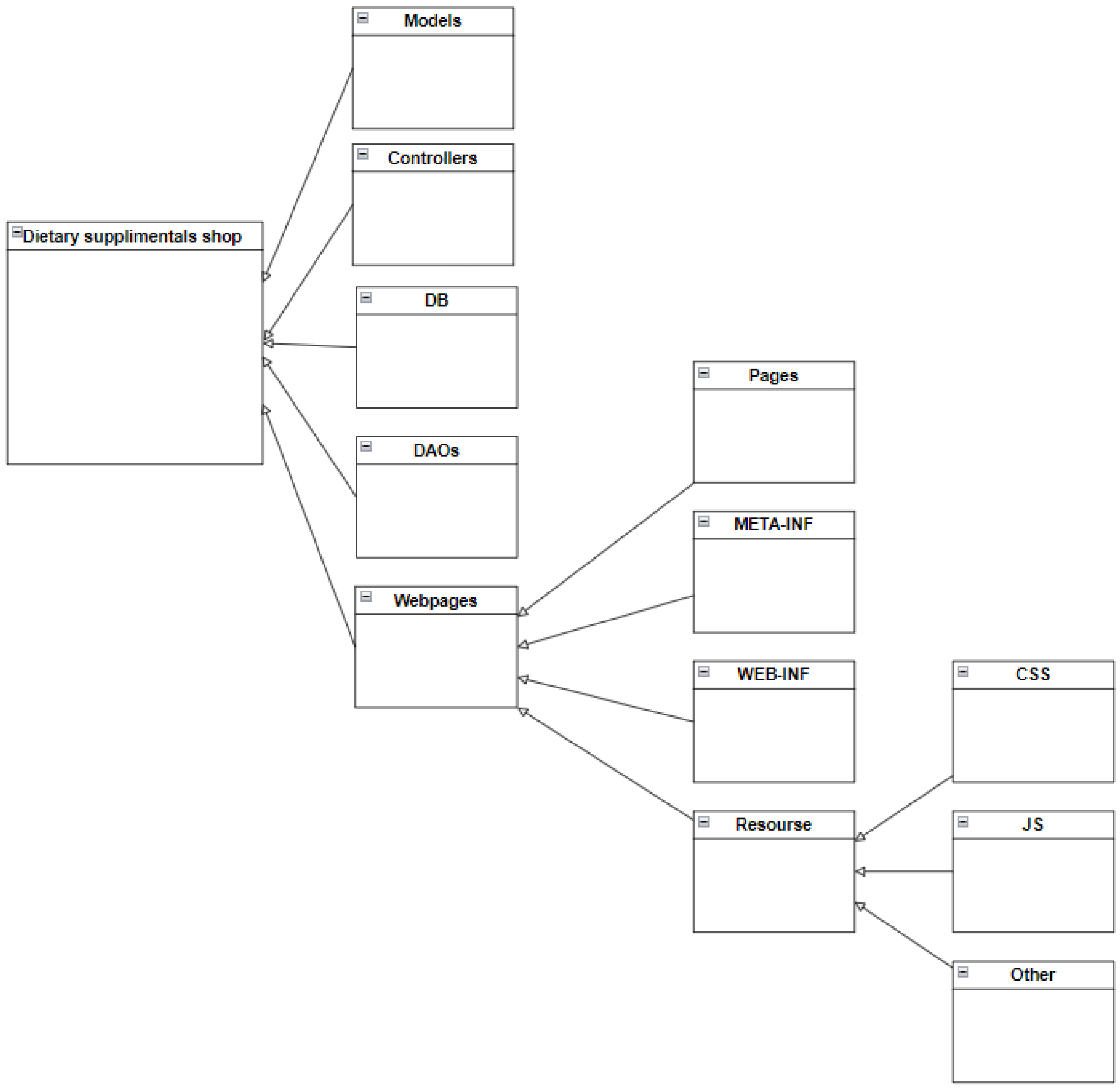
## VERSION HISTORY

| Version | Release date | Outline |
|---|---|---|
| 0.1 | 31-01-2023 | Start working |
| 1.0 | 03-02-2023 | The first SDD version |
| 1.5 | 07-02-2023 | Update some function |

## Table of contents

## I. Overview

### 1. Code Packages/Namespaces

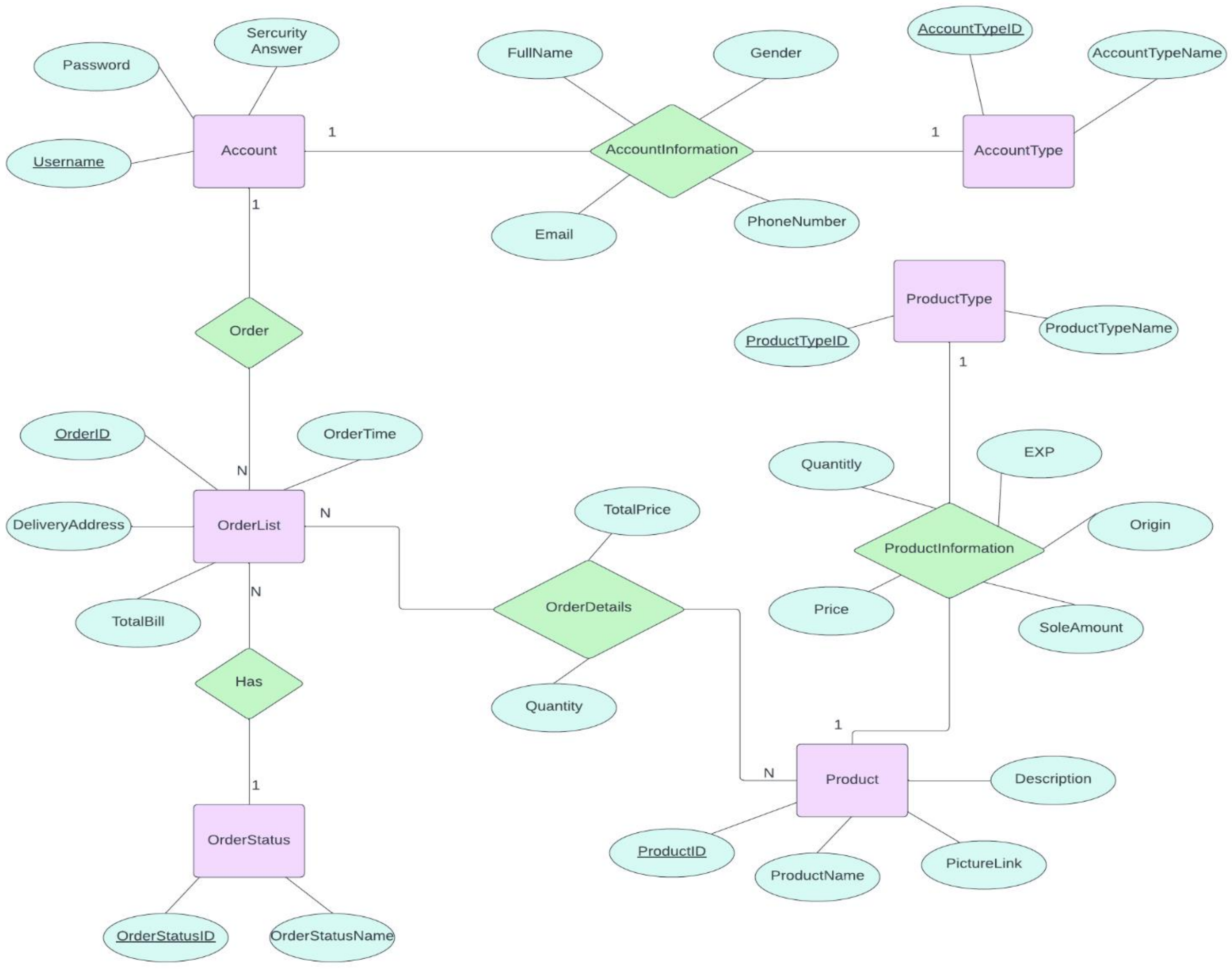


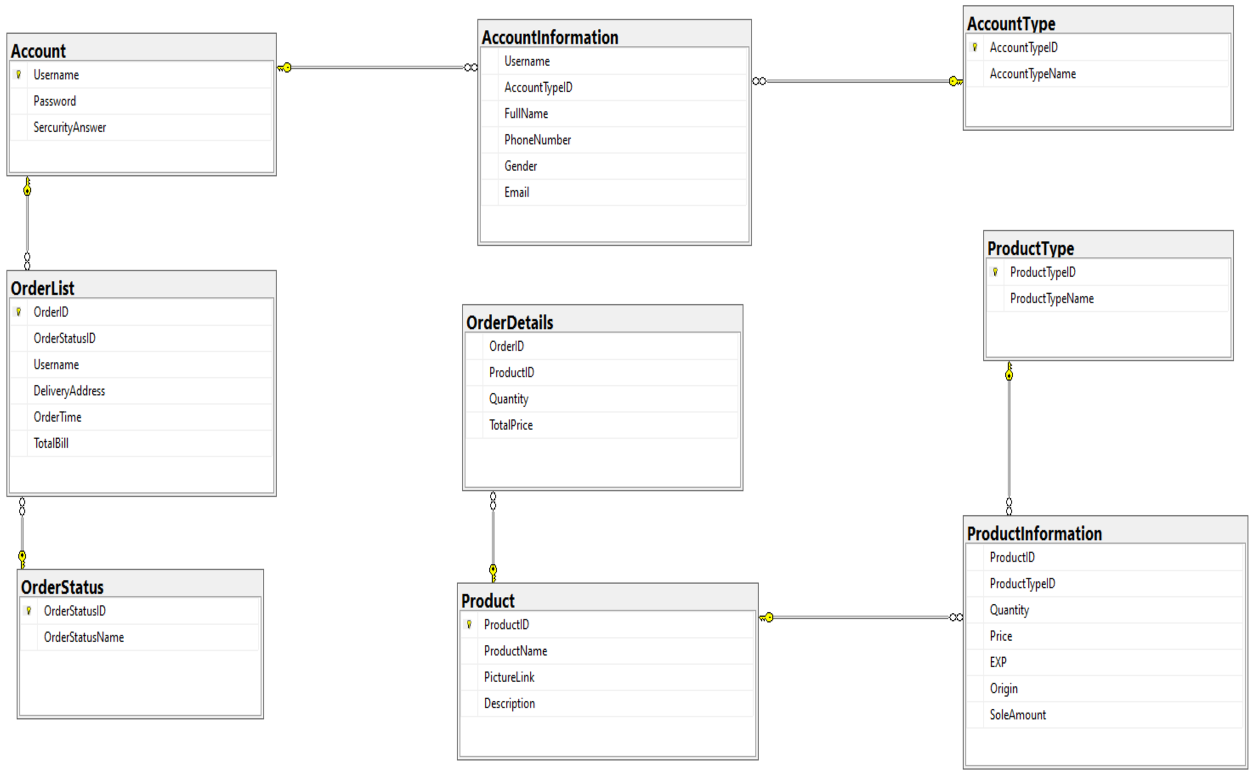
| No | Package | Description |
|---|---|---|
| 01 | Controllers | - The class in this package is responsible for handling requests from users, processing data, and sending responses to users through the web interface. |

| | | |
|---|---|---|
| | | - Naming the class according to the CamelCase usage specification with the class name describing the class's features<br>Example: AccountController, PaymentController,... |
| 02 | DAOs | - The class in this package is responsible for handling operations related to retrieving data from or saving data to the database. The class in this package provides methods to perform CRUD (Create, Read, Update, Delete) operations on the database.<br>- Naming the class according to the CamelCase usage rules with the name of the class that describes the feature of that class and prefixed with "DAO" to indicate that this class is related to the data query<br>Example: AccountDAO, OrderDAO,... |
| 03 | DB | - The class in this package is responsible for managing the connection to the database, including creating the connection, closing the connection, executing the query, and getting the results.<br>- Naming the class according to the CamelCase usage rule with the name of the class that describes the feature of that class and prefixed with "DB" (Database) to indicate the class is related to the database.<br>Example: DBConnection, DBManager,... |
| 04 | Model | - The class in this package describes the properties and methods of objects in the system, such as users, articles, products, etc.<br>- Naming the class according to the CamelCase usage rules with the name of the class that describes the class's features and prefixed with "Model" to |

| | | |
|---|---|---|
| | | indicate that the class is related to the data model.<br>Example: AccountModel, ProductModel,... |
| 05 | Webpages | Webpages include dynamic web pages, static HTML pages, CSS files, JavaScript and images. It is used to manage the user interface documents and provide the functionality for the website to interact with the user. |

## 2. ERD


Password
Sercurity Answer
FullName
Gender
AccountTypeID
AccountTypeName
Username
Account
1
AccountInformation
1
AccountType
1
Email
PhoneNumber
ProductType
ProductTypeID
ProductTypeName
Order
1
OrderID
OrderTime
Quantitly
EXP
N
TotalPrice
DeliveryAddress
OrderList
N
ProductInformation
Origin
N
OrderDetails
TotalBill
Price
SoleAmount
Has
Quantity
1
1
N
Product
Description
OrderStatus
ProductID
ProductName
PictureLink
OrderStatusID
OrderStatusName



**Entity name:** Account

**Properties:** Username, Password, SecurityAnswer

**Entity name:** AccountType

**Properties:** AccountTypeID, AccountTypeName

**Entity name:** AccountInformation

**Properties:** Username, AccountTypeID, FullName, Gender, Email, PhoneNumber

**Entity name:** Product

**Properties:** ProductID, ProductName, PictureLink, Description

**Entity name:** ProductType

**Properties:** ProductTypeID, ProductTypeName

**Entity name:** ProductInformation

**Properties:** ProductID, ProductTypeID, Quantity,SoldAmount, Price, EXP, Origin

**Entity name:** Order

**Properties:** OrderID, Username, OrderStatusID ,DeliveryAddress, OrderTime, OrderStatus, TotalBill

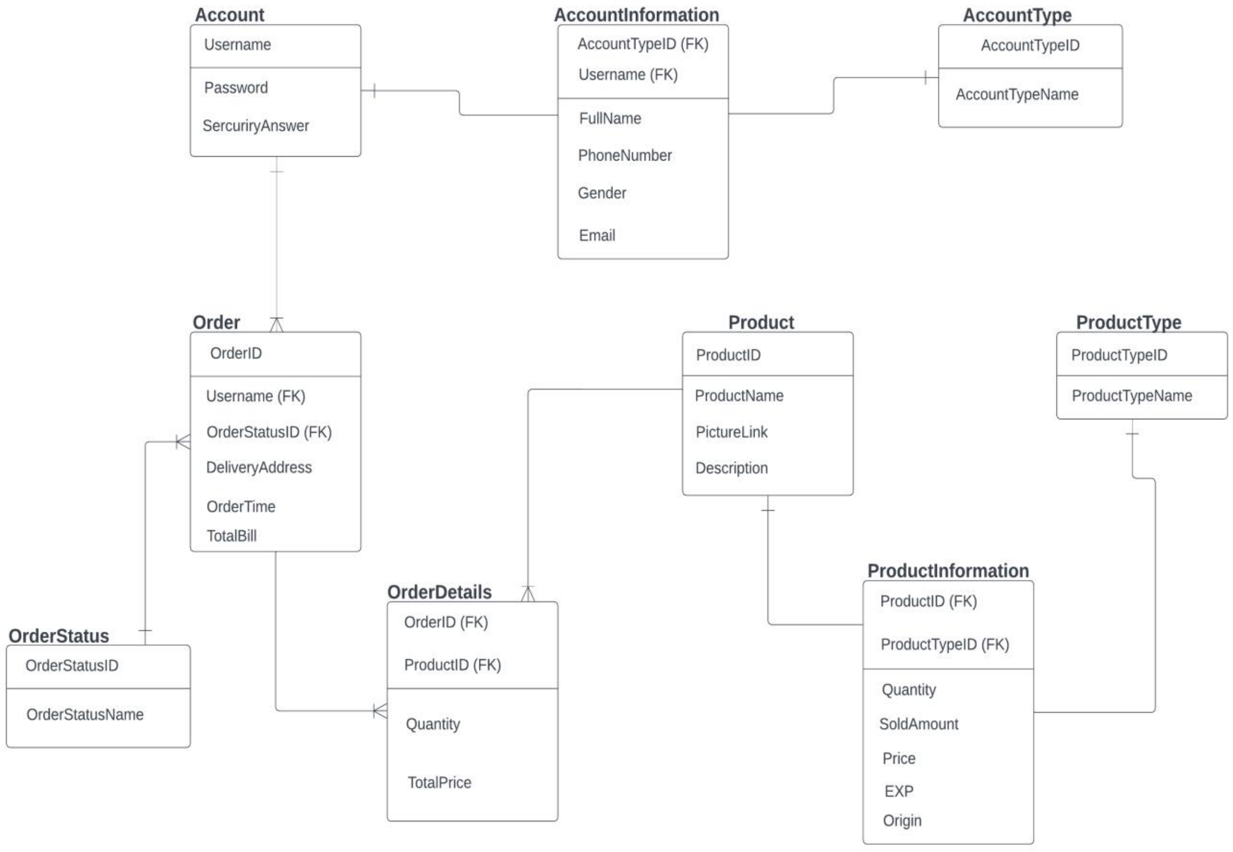
**Entity name:** OrderStatus

**Properties:** OrderStatusID, OrderStatusName

**Entity name:** OrderDetails

**Properties:** OrderID, ProductID, Quantity, TotalPrice

## 3. Database Schema



| **No** | **Table** | **Description** |
|---|---|---|
| ***01*** | Account | Table is used to store account information including username and password and answers to security questions.<br><br>- Primary keys: Username<br><br>- Foreign keys: No have |
| ***02*** | AccountType | *Table is used to store account type information*<br><br>*- Primary keys: AccountTypeID*<br><br>*- Foreign keys: No have* |

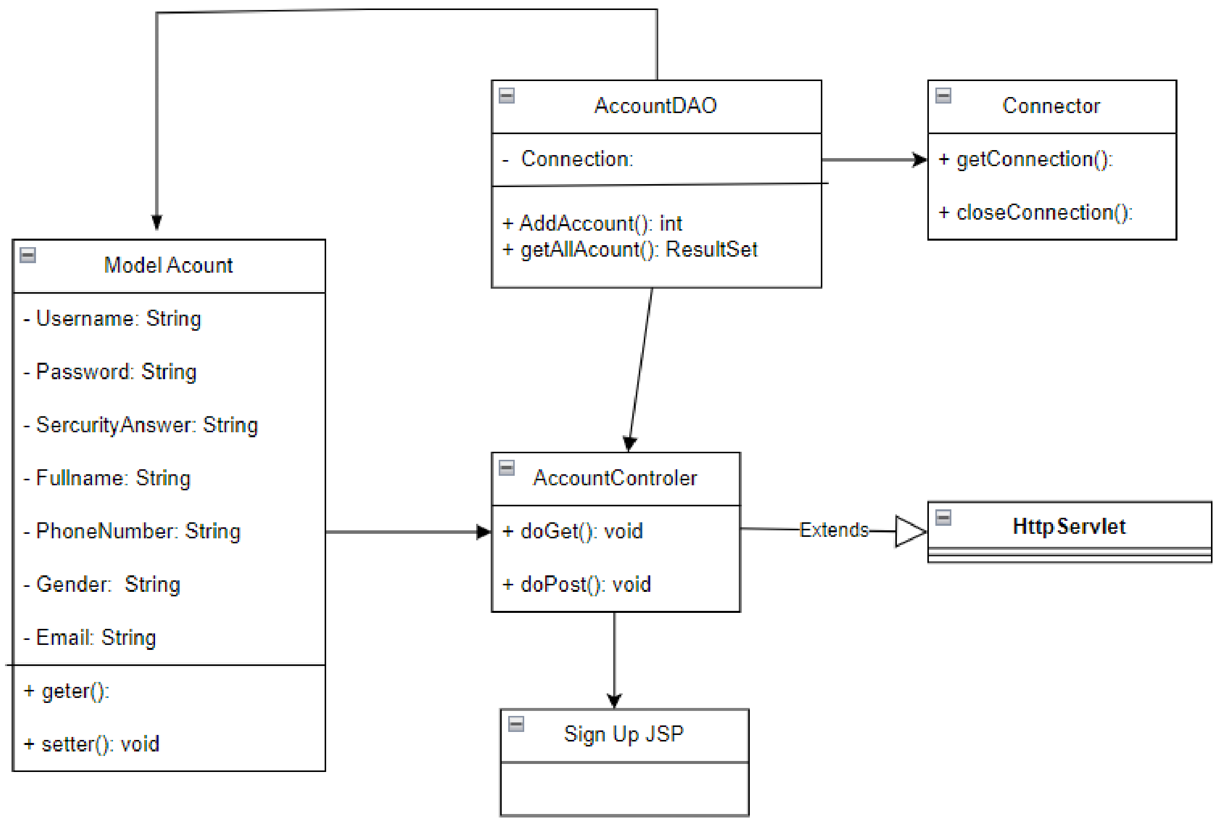
| | | |
|---|---|---|
| ***03*** | AccounntInformation | This is the relationship table between the Account table and the AccountType table used to store the account holder's personal information<br>- Primary keys: No have<br>- Foreign keys: AccountTypeID, Username |
| ***04*** | Product | Table used to store general information about products<br>- Primary keys: ProductID<br>- Foreign keys: No have |
| ***05*** | ProductType | Table used to store the product type and the name of the product type<br>- Primary keys: ProductTypeID<br>- Foreign keys: No have |
| ***06*** | ProductInformation | This is a relational table between the Product and ProductType tables used to store detailed product information including quantity and price<br>- Primary keys: No have<br>- Foreign keys: ProductID, ProductTypeID |
| ***07*** | Order | Table used to store general information about ordering<br>- Primary keys: OrderID<br>- Foreign keys: Username, OrderStatusID |
| ***08*** | OrderStatus | Table used to store the status of the order<br>- Primary keys: OrderStatusID<br>- Foreign keys: No have |
| ***09*** | OrderDetails | This is the relation table of Order table and Product table used to store order details including products and their quantity as well as their total price. |

| | | |
|---|---|---|
| | | - Primary keys: No have<br><br>- Foreign keys: OrderID, ProductID |

## II. Code Designs

### 1. Sign Up

#### a. Class Diagram



#### b. Class Specifications

*Account Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class. This class is non-parameter. The output depends on the attribute's class.* |
| ***02*** | ***setter()*** | *The setter method to set data in this class. The parameter is a value that needs to be set in this class. This class is a void return.* |

*AcountDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***AddAcount()*** | *Let the program create a new account in the database system. By getting an account object created from the controller.* |

|  |  |  |
|---|---|---|
| ***02*** | ***GetAllAccount()*** | *Get all account in database for check the valid of signup account. Let web can check the username, email a duplicate or not.* |

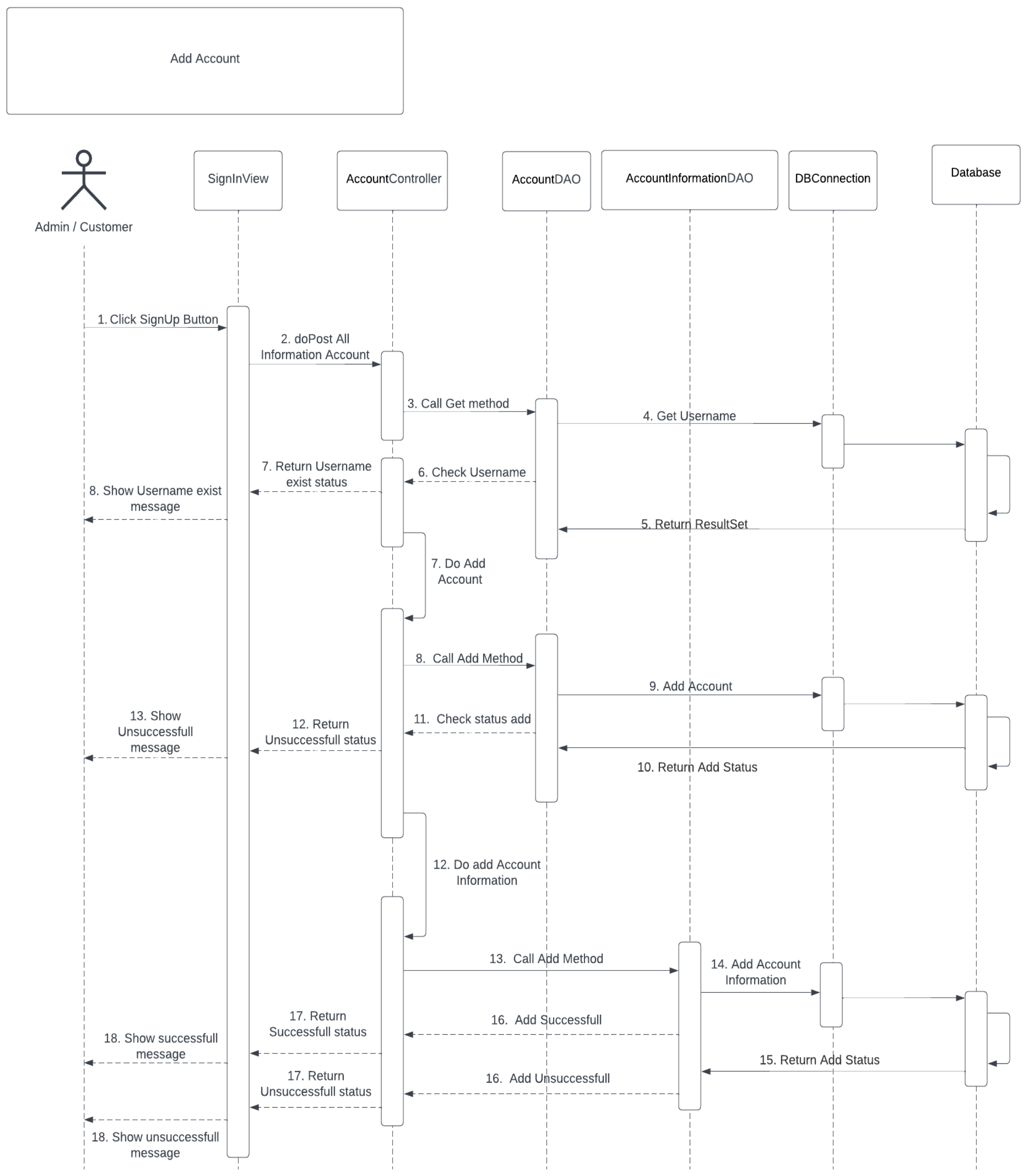
*Account Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Handle logic for accessing the new account registration page.*<br><br>*Check if the user is logged in, if logged in will redirect the user to the user's home page or account page, otherwise return the new account registration page for the user.* |
| ***02*** | ***doPost()*** | *Handle logic for new account registration. Get the data from the registration page and check if the information is valid or not.*<br><br>*If the credentials are valid, save the information to the database and redirect the user to the login page, otherwise display an error message.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)



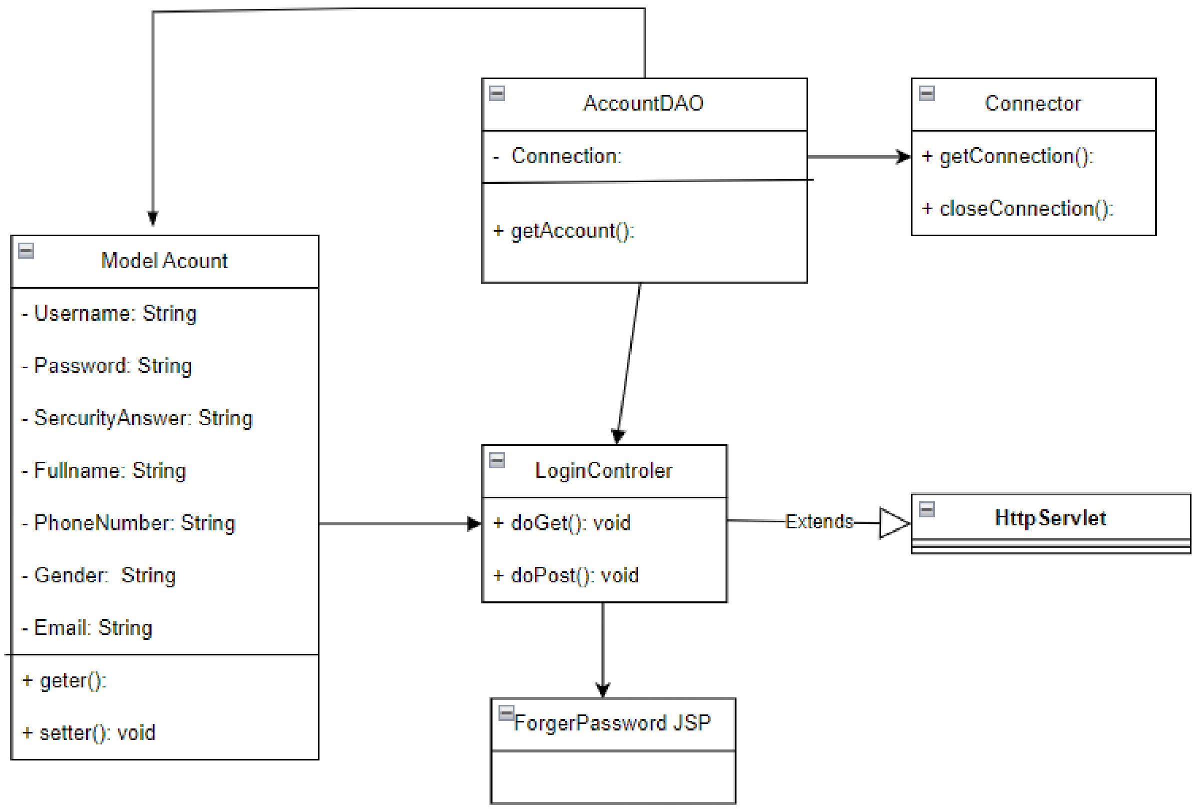
## d. Database queries

Insert into Account values(‘Username’, ‘Password’, ‘SecurityAnswer’)

Insert into AccountInformation values('<u>Username’, </u>‘AccountTypeID’, ‘FullName’, ‘Gender’, ‘Email’, ‘PhoneNumber’)

## 2. Login

### a. Class Diagram



### b. Class Specifications

*Account Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Account Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Returns HTML page for user to enter login information, check if user is logged in or not, if logged in then redirect user to main system page.* |

|  |  |  |
|---|---|---|
| ***02*** | ***doPost()*** | *Get the credentials from the HTML form and check if the credentials are valid.*<br><br>*If valid, save the login credentials to the user's session and redirect the user to the main system page, otherwise error messages to the user.* |

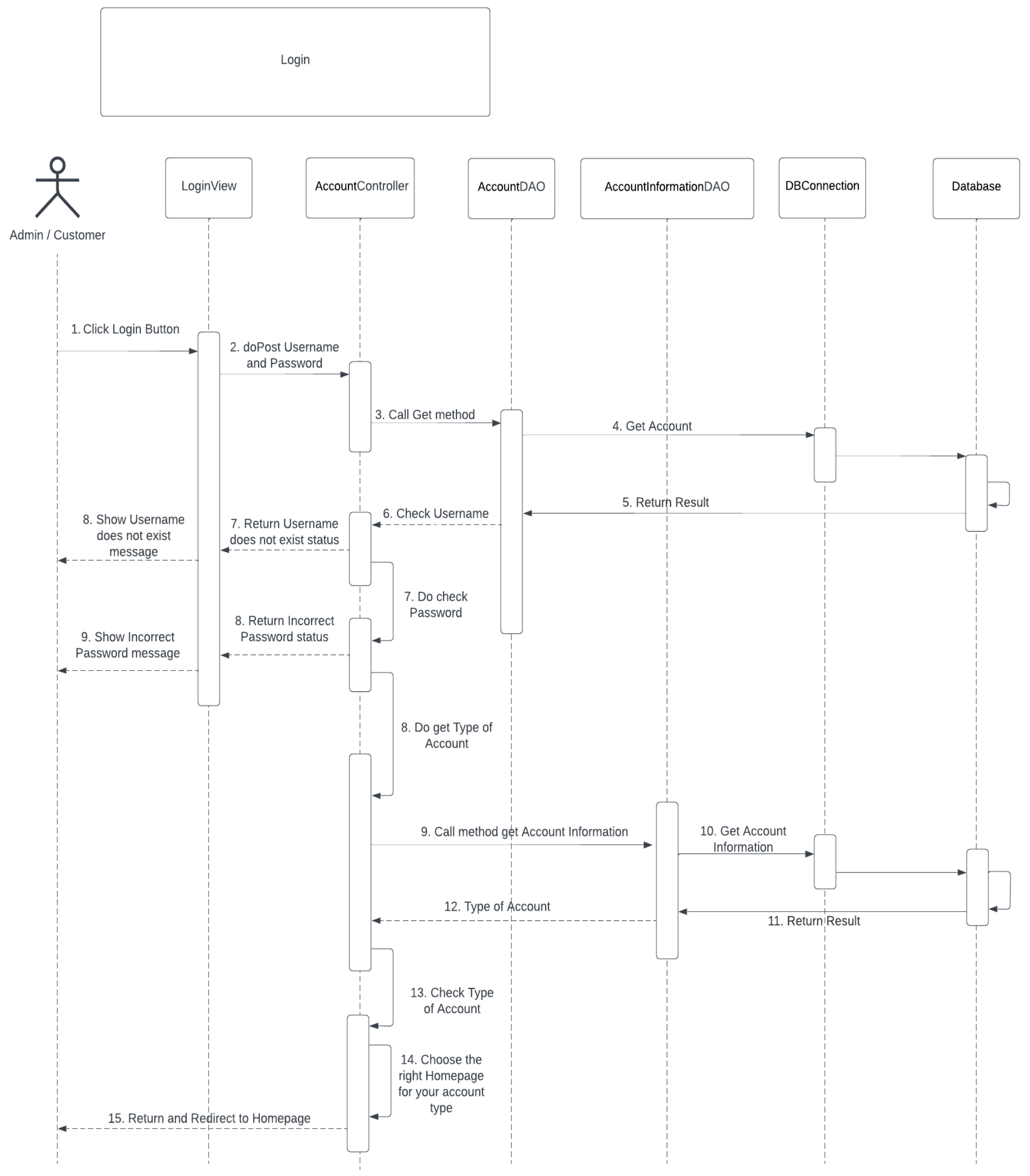
*AcountDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***GetAcount()*** | Searches for an account in the database based on the provided account information (either username or email address) and password. If the corresponding account is found, the function will return the account information, otherwise it will return null. |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)

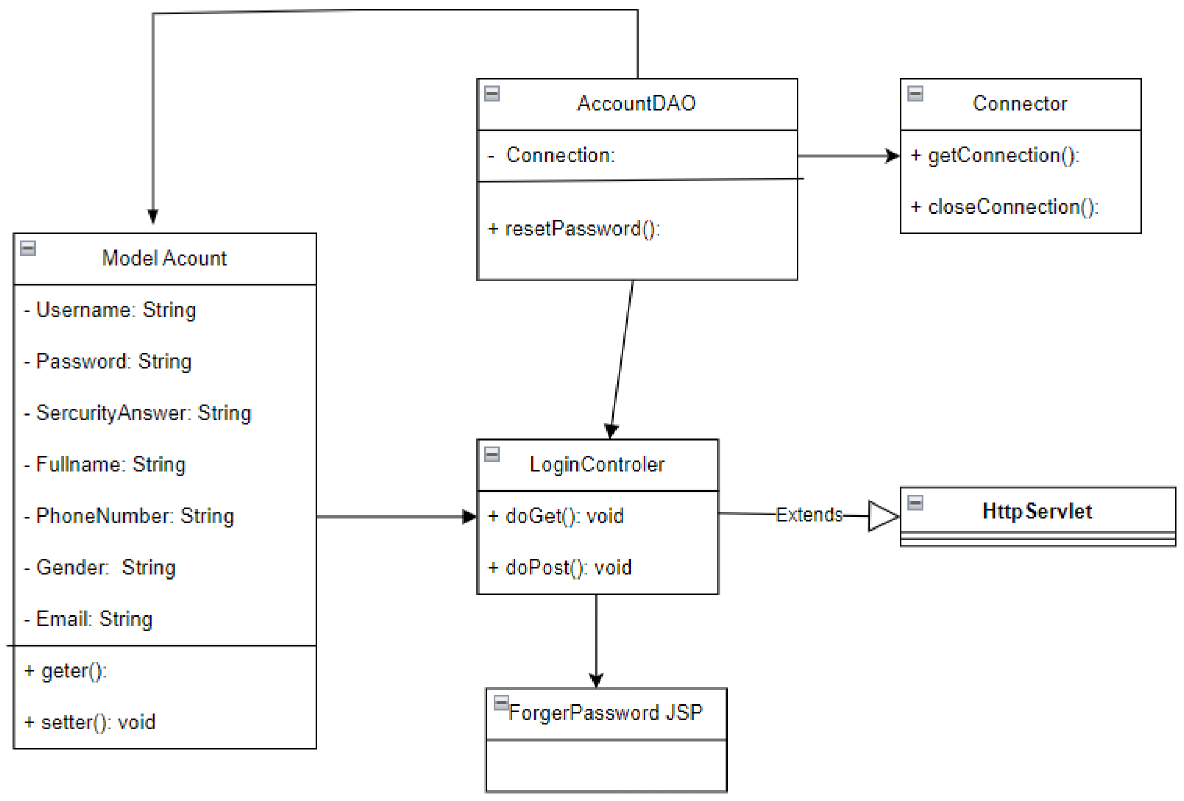

Login
Admin / Customer
LoginView
AccountController
AccountDAO
AccountInformationDAO
DBConnection
Database
1. Click Login Button
2. doPost Username and Password
3. Call Get method
4. Get Account
5. Return Result
6. Check Username
7. Return Username does not exist status
8. Show Username does not exist message
7. Do check Password
8. Return Incorrect Password status
9. Show Incorrect Password message
8. Do get Type of Account
9. Call method get Account Information
10. Get Account Information
11. Return Result
12. Type of Account
13. Check Type of Account
14. Choose the right Homepage for your account type
15. Return and Redirect to Homepage

## d. Database queries

Select Password from Account where Username = "Username"

## 3. Forget Password

### a. Class Diagram



### b. Class Specifications

*Account Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *To send user to the HTML forget password* |
| ***02*** | ***doPost()*** | *Get the credentials from the HTML form and check if the credentials are valid.* |

| | | |
|---|---|---|
| | | *If correct, the password will be reset to default string.* |

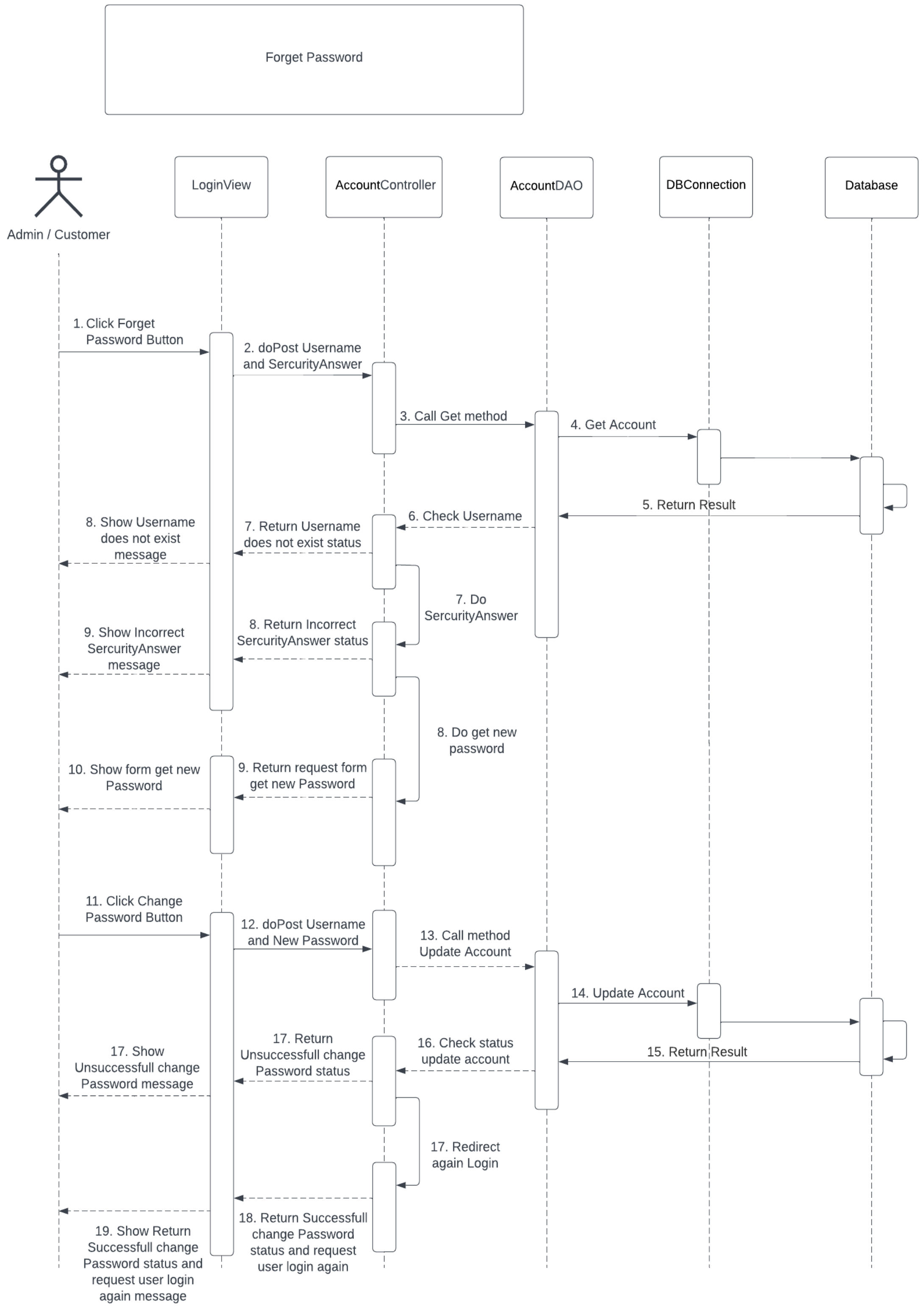
*AcountDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***ResetPassword()*** | *To reset password of user.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)

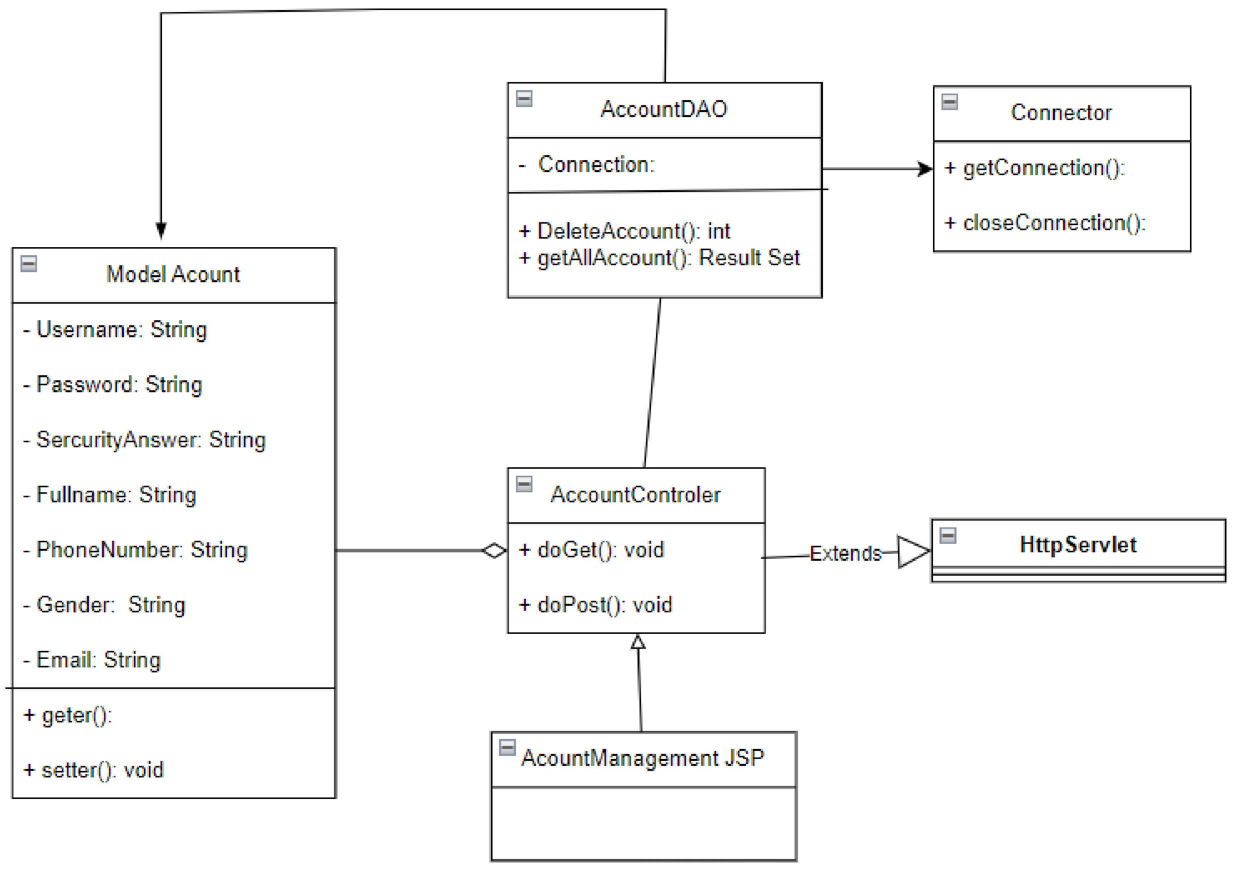

Forget Password
Admin / Customer
LoginView
AccountController
AccountDAO
DBConnection
Database
1. Click Forget Password Button
2. doPost Username and SercurityAnswer
3. Call Get method
4. Get Account
5. Return Result
6. Check Username
7. Return Username does not exist status
8. Show Username does not exist message
7. Do SercurityAnswer
8. Return Incorrect SercurityAnswer status
9. Show Incorrect SercurityAnswer message
8. Do get new password
9. Return request form get new Password
10. Show form get new Password
11. Click Change Password Button
12. doPost Username and New Password
13. Call method Update Account
14. Update Account
15. Return Result
16. Check status update account
17. Return Unsuccessfull change Password status
17. Show Unsuccessfull change Password message
17. Redirect again Login
18. Return Successfull change Password status and request user login again
19. Show Return Successfull change Password status and request user login again message

### d. Database queries

Select SecurityAnswer from Account where Username = 'Username'

## 4. Delete Account

### a. Class Diagram



### b. Class Specifications

*Account Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br><br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br><br>*Input: The data to be set in attribute's class* |

*AcountDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***DeleteAccount()*** | *Delete the user account in the database. Get the account information from the database and then use SQL statements or similar APIs to delete the account. Parameters such as account ID or account name may be required to identify the account to be deleted. Returns the success or failure result of the account deletion.* |
| ***02*** | ***GetAllAcount()*** | *Returns a list of all accounts currently stored in the system, used to display a list of accounts to administrators or to delete accounts based on the account's name or ID. The results are returned as a list or a table.* |

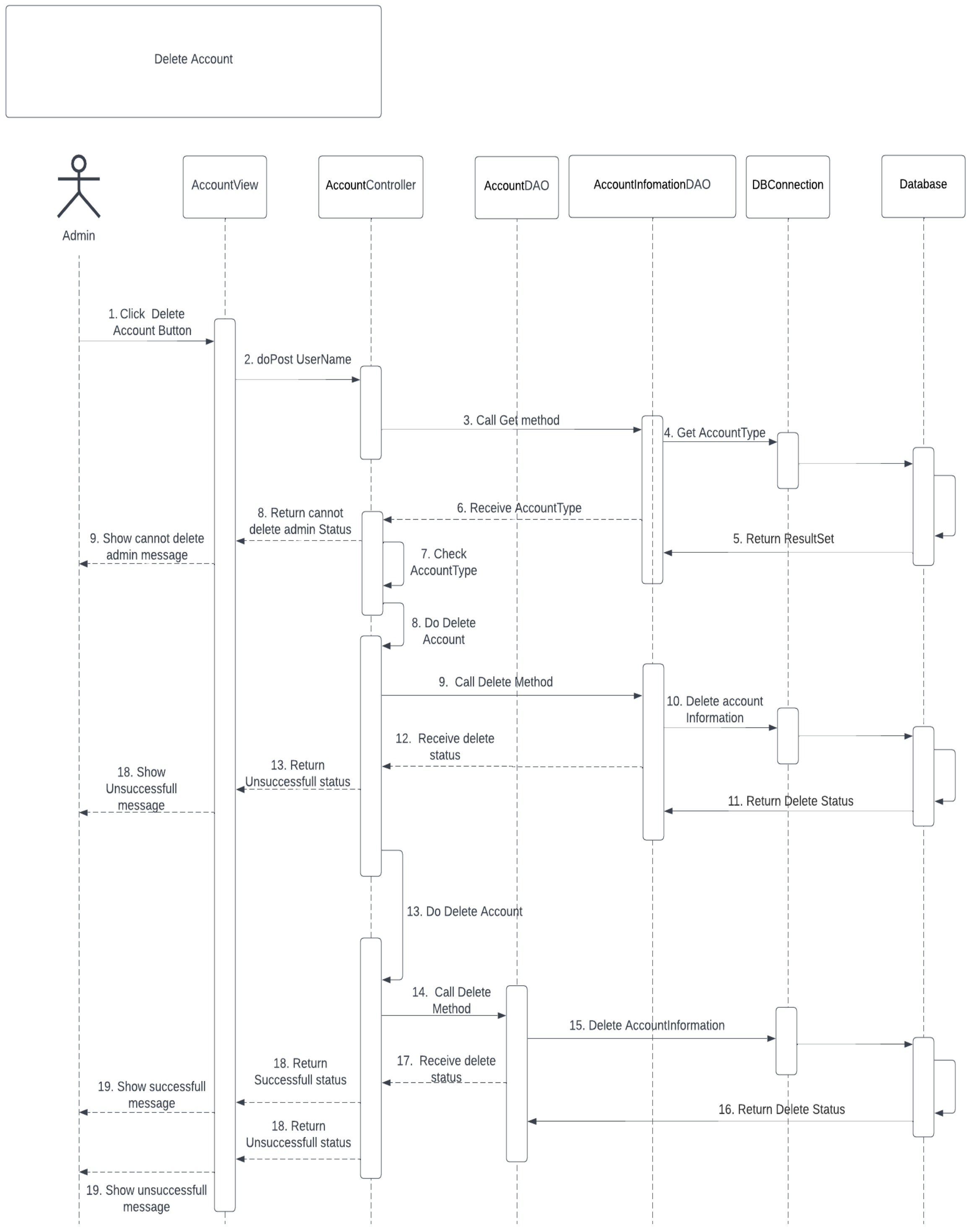
*Account Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet*** | *Process the display of the account deletion confirmation interface to the user, get information about the user's account from the database, and then display this information on the interface to confirm the deletion.* |
| ***02*** | ***DoPost()()*** | *Handles deletion of a user's account from the database. Get the account information from the input request, then use SQL statements to remove the corresponding account from the database.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)


Delete Account
Admin
AccountView
AccountController
AccountDAO
AccountInfomationDAO
DBConnection
Database
1. Click Delete Account Button
2. doPost UserName
3. Call Get method
4. Get AccountType
5. Return ResultSet
6. Receive AccountType
7. Check AccountType
8. Return cannot delete admin Status
9. Show cannot delete admin message
8. Do Delete Account
9. Call Delete Method
10. Delete account Information
11. Return Delete Status
12. Receive delete status
13. Return Unsuccessfull status
18. Show Unsuccessfull message
13. Do Delete Account
14. Call Delete Method
15. Delete AccountInformation
16. Return Delete Status
17. Receive delete status
18. Return Successfull status
19. Show successfull message
18. Return Unsuccessfull status
19. Show unsuccessfull message

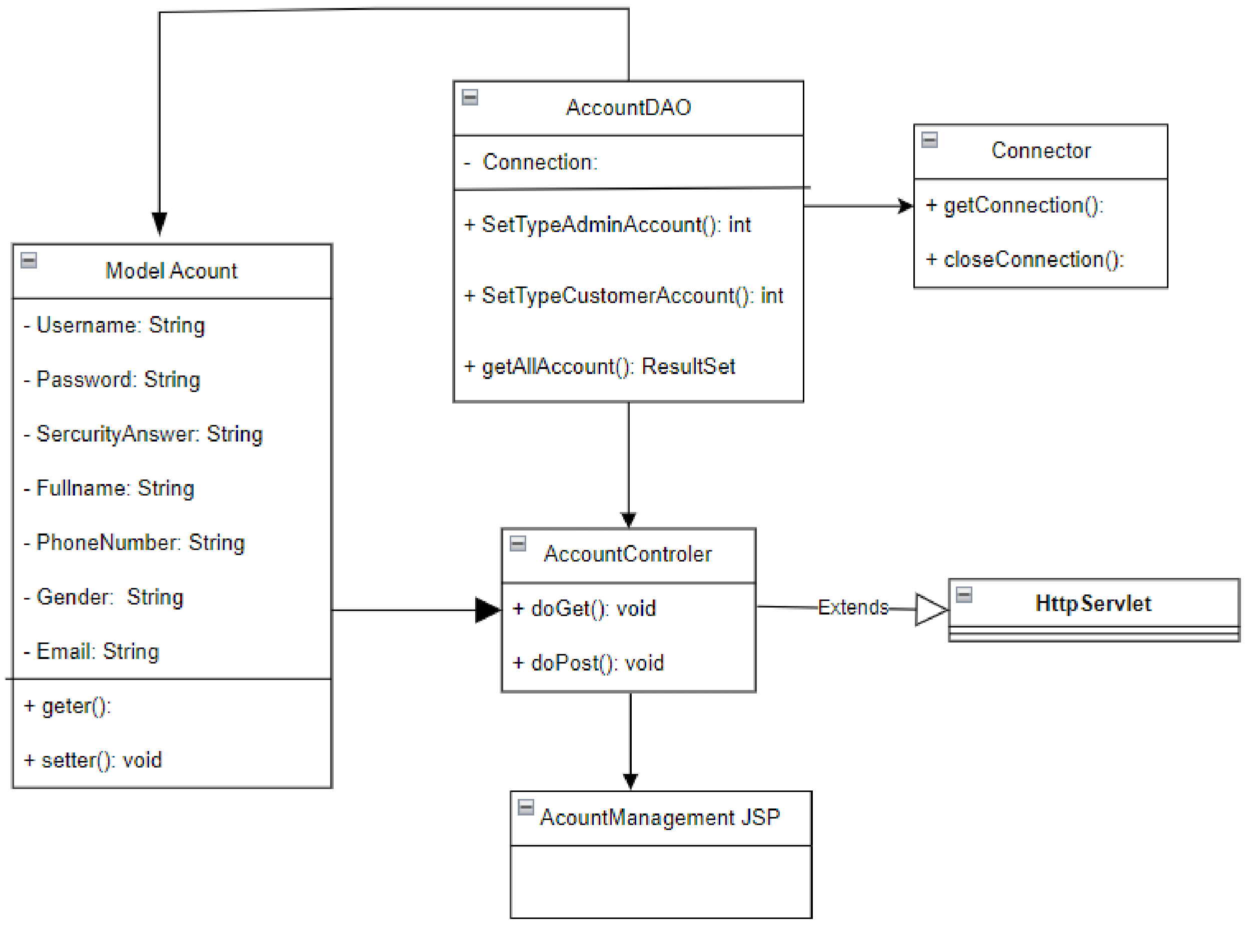
### d. Database queries

Delete from AccountInformation where username = 'username'

Delete from Account where username = 'username'

## 5. Change Permission Account

### a. Class Diagram



### b. Class Specifications

*Account Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br><br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.* |

| | | |
|---|---|---|
| | | *Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Handling logic when changing account permissions.*<br><br>*Get the ID of the account that the admin needs to change, proceed to call the function in AccountDAO to get the account type to which the account is assigned. When receiving the account type, proceed to reset them. If the account type is admin, change it back to customer and vice versa.* |
| ***02*** | ***doPost()*** | *For the doPost() function for changing account permissions, this function does not take any action* |

*AcountDAO class*

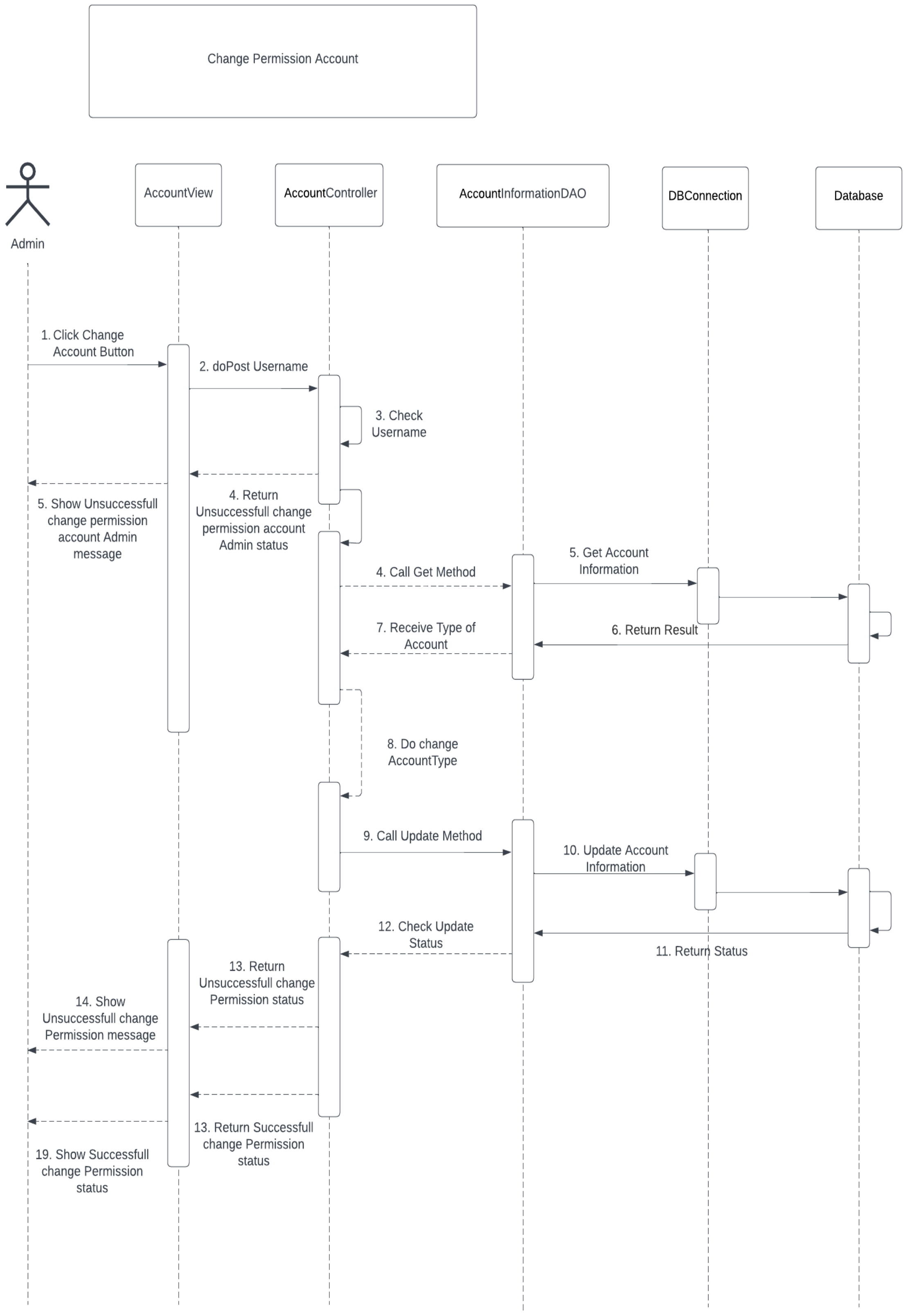
| No | Method | Description |
|---|---|---|
| ***01*** | ***SetTypeAdminAccount()*** | *The task changes the account's permissions from Customer to Admin. This function can take as a parameter the ID of the account that needs to change permissions and use SQL statements to update the authority information in the database. After successful update, the function will return true or false.* |
| ***02*** | ***SetTypeCustomerAccount()*** | *The task changes the account's permissions from Admin to Customer. This function can take as a parameter the ID of the account that needs to change permissions and use SQL statements to update the authority information in the database. After successful update, the function will return true or false.* |
| ***03*** | ***GetAllAcount()*** | *Query and return a list containing all accounts currently stored in the database. Get a list of accounts to display in the user interface or to perform account permissions switching.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using |

|  |  |  |
|---|---|---|
|  |  | the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)

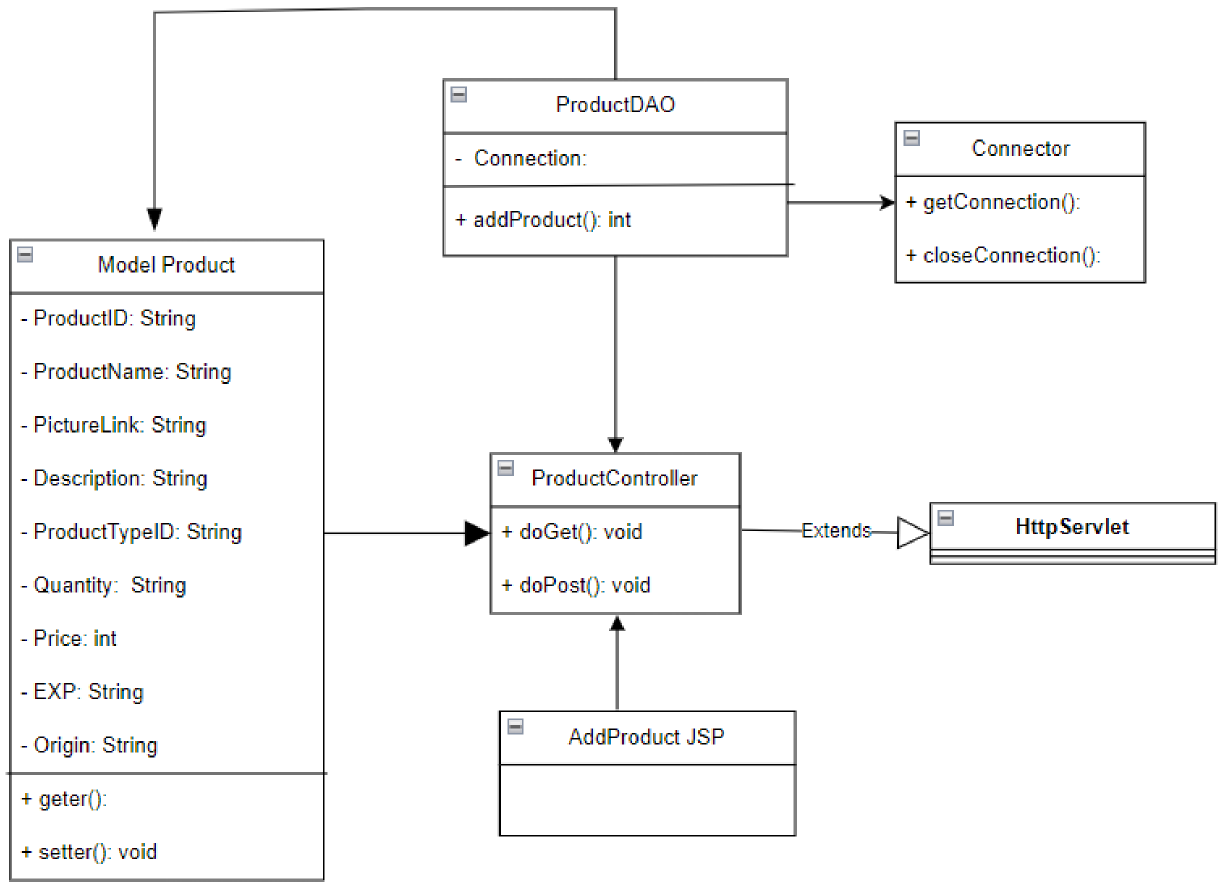

Change Permission Account
Admin
AccountView
AccountController
AccountInformationDAO
DBConnection
Database
1. Click Change Account Button
2. doPost Username
3. Check Username
4. Return Unsuccessfull change permission account Admin status
5. Show Unsuccessfull change permission account Admin message
4. Call Get Method
5. Get Account Information
6. Return Result
7. Receive Type of Account
8. Do change AccountType
9. Call Update Method
10. Update Account Information
11. Return Status
12. Check Update Status
13. Return Unsuccessfull change Permission status
14. Show Unsuccessfull change Permission message
13. Return Successfull change Permission status
19. Show Successfull change Permission status

### d. Database queries

Update AccountInformation set AccountTypeID = 'AccountTypeID', FullName = 'Fullname', PhoneNumber = 'PhoneNumber', Gender = 'Gender', Email = 'Email' where Username = 'Username'

## 6. Add Product

### a. Class Diagram



### b. Class Specifications

*Product Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Handle logic when adding new product. It proceeds to call the function in ProductDAO to add a new product.* |
| ***02*** | ***doPost()*** | *Checking if the new ID, continue getting parameter to add. If ID is existed, anounce adding fail & return product page.* |

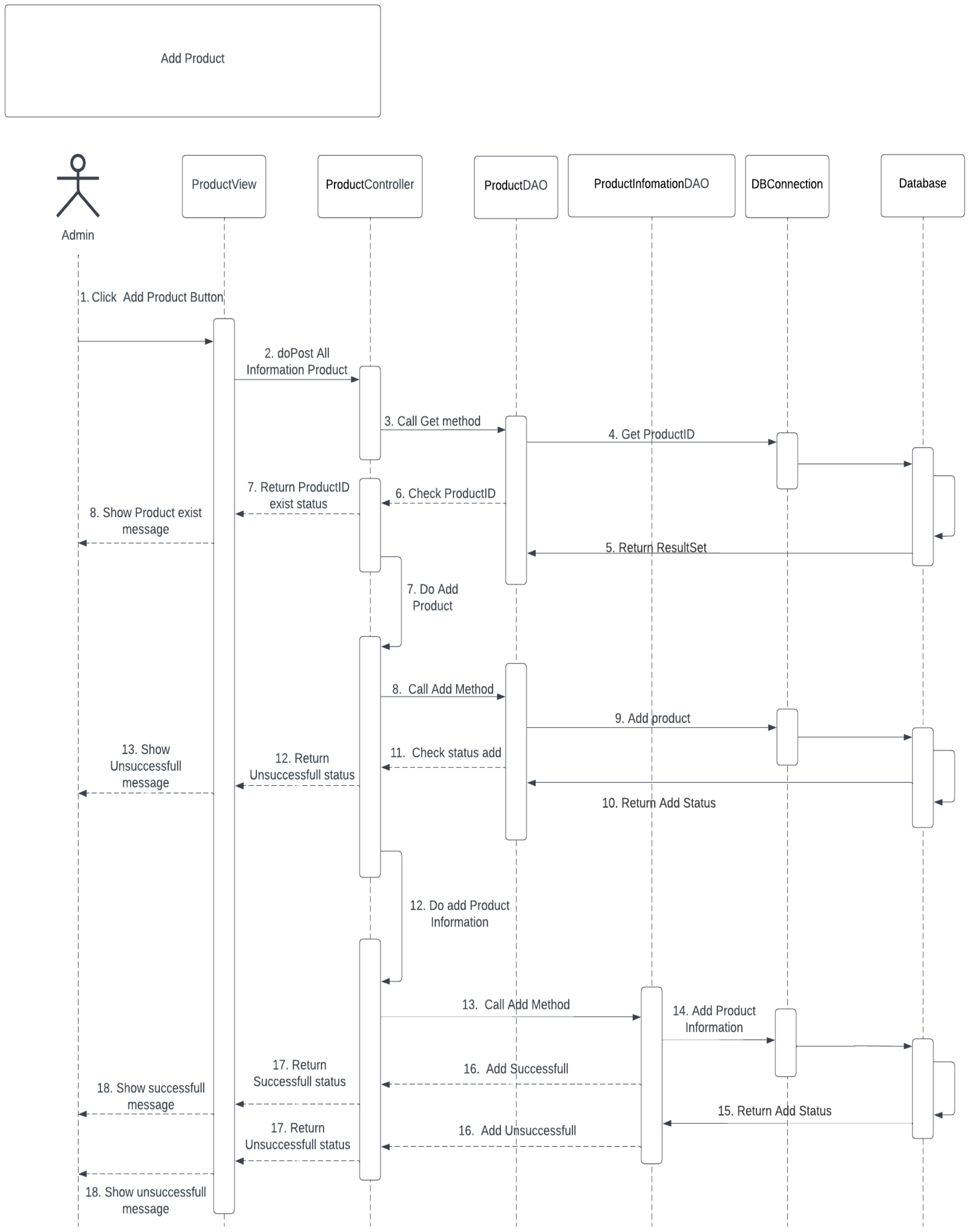
*ProductDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***AddProduct()*** | *Add a new product to the database. Get product information such as name, description, price, quantity, etc. Execute the corresponding SQL statements to add new products to the database.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)


Add Product
Admin
ProductView
ProductController
ProductDAO
ProductInfomationDAO
DBConnection
Database
1. Click Add Product Button
2. doPost All Information Product
3. Call Get method
4. Get ProductID
5. Return ResultSet
6. Check ProductID
7. Return ProductID exist status
8. Show Product exist message
7. Do Add Product
8. Call Add Method
9. Add product
10. Return Add Status
11. Check status add
12. Return Unsuccessfull status
13. Show Unsuccessfull message
12. Do add Product Information
13. Call Add Method
14. Add Product Information
15. Return Add Status
16. Add Successfull
16. Add Unsuccessfull
17. Return Successfull status
17. Return Unsuccessfull status
18. Show successfull message
18. Show unsuccessfull message

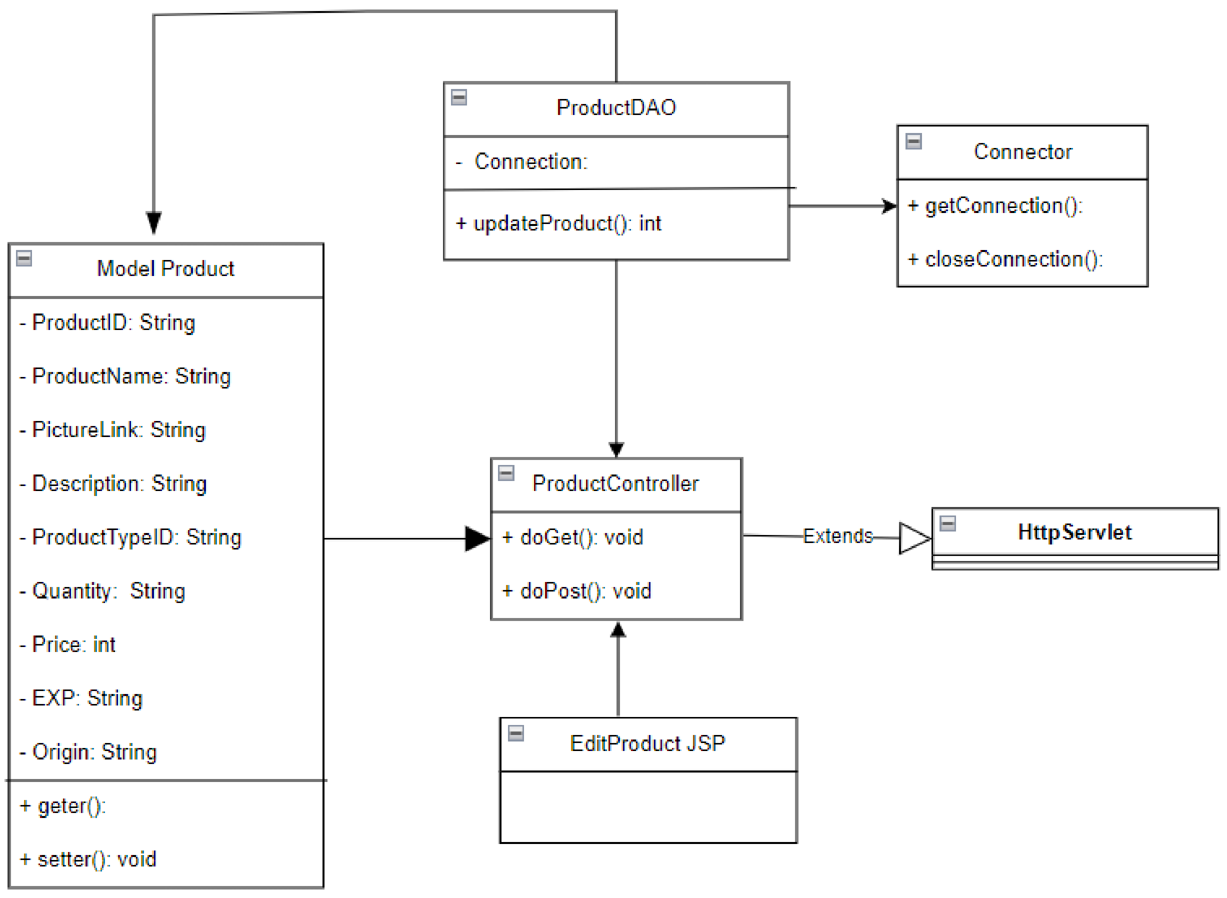
### d. Database queries

Insert into Product values ('ProductID','ProductName', 'PictureLink', 'Description')

Insert into ProductInformation values ('ProductID','ProductTypeID', 'Quantity', 'Price','EXP','Origin')

## 7. Edit Product

### a. Class Diagram



### b. Class Specifications

*Product Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Handle logic when edit product information. It proceeds to call the function in ProductDAO to edit product.* |
| ***02*** | ***doPost()*** | *Checking ID, if existed accept edit information except ID. If ID is not existed or information not change, it will display anouncement.* |

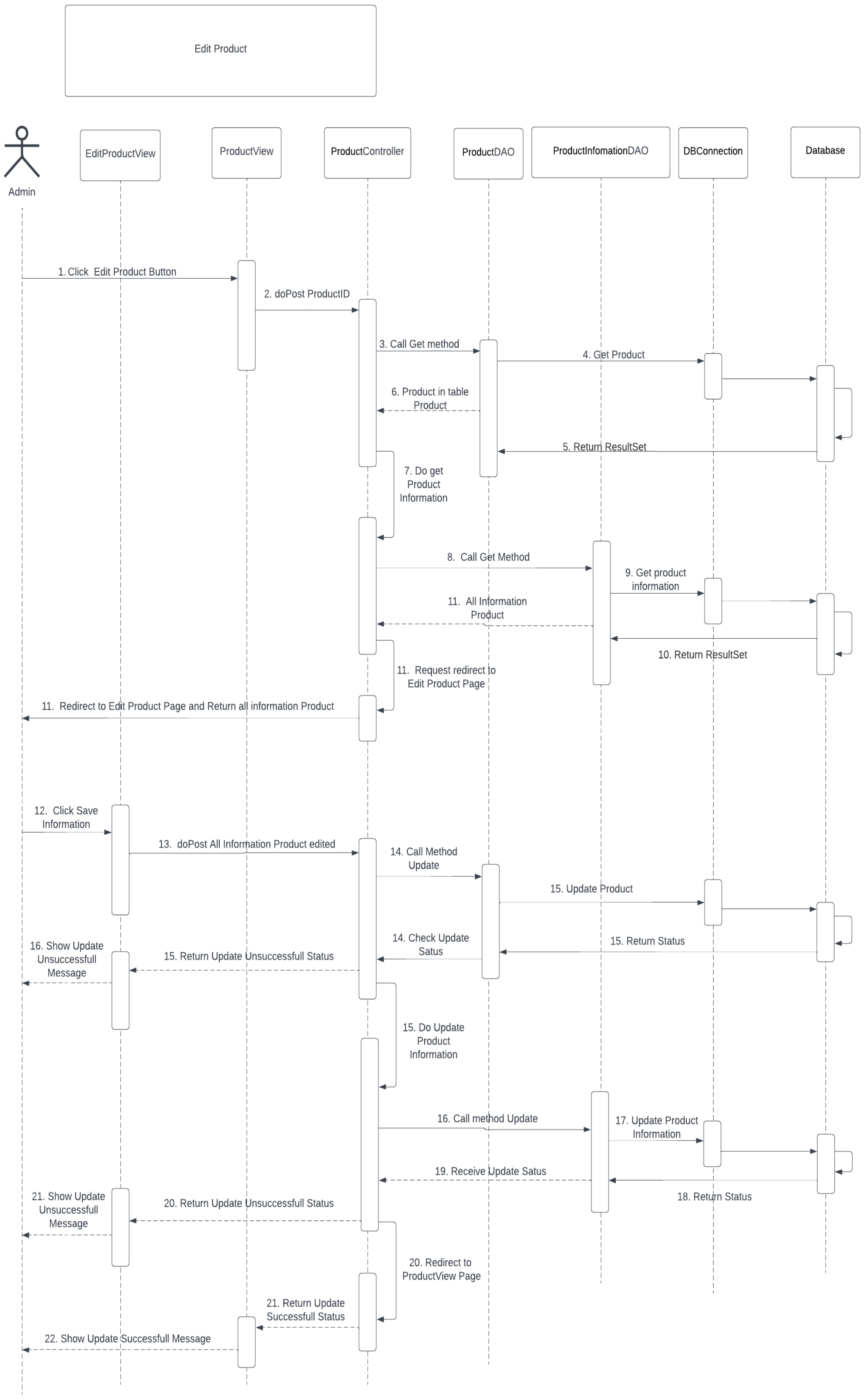
*ProductDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***updateProduct()*** | *Update/edit product information. Get a product object with the new properties and use SQL statements to update the information in the database corresponding to that product.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)



Edit Product
Admin
EditProductView
ProductView
ProductController
ProductDAO
ProductInfomationDAO
DBConnection
Database
1. Click Edit Product Button
2. doPost ProductID
3. Call Get method
4. Get Product
6. Product in table Product
5. Return ResultSet
7. Do get Product Information
8. Call Get Method
9. Get product information
11. All Information Product
10. Return ResultSet
11. Request redirect to Edit Product Page
11. Redirect to Edit Product Page and Return all information Product
12. Click Save Information
13. doPost All Information Product edited
14. Call Method Update
15. Update Product
16. Show Update Unsuccessfull Message
15. Return Update Unsuccessfull Status
14. Check Update Satus
15. Return Status
15. Do Update Product Information
16. Call method Update
17. Update Product Information
19. Receive Update Satus
18. Return Status
21. Show Update Unsuccessfull Message
20. Return Update Unsuccessfull Status
20. Redirect to ProductView Page
21. Return Update Successfull Status
22. Show Update Successfull Message

### d. Database queries

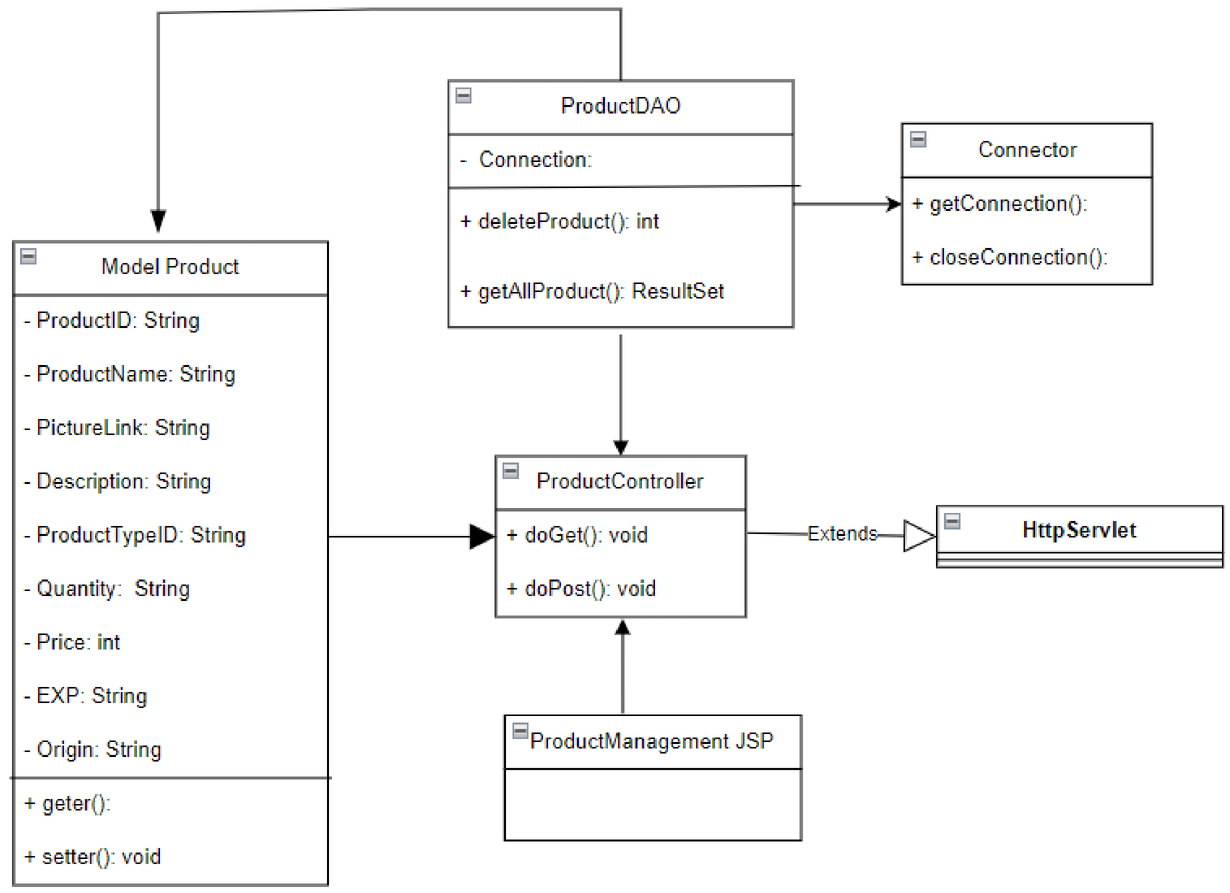
```
Select
Product.ProductID,Product.ProductName,Product.PictureLink,Product.[Description],
ProductInformation.Price, ProductInformation.Quantity, ProductInformation.Origin,
ProductInformation.[EXP], ProductType.ProductTypeName from
Product,ProductInformation,ProductType where Product.ProductID =
ProductInformation.ProductID and ProductInformation.ProductTypeID =
ProductType.ProductTypeID
```

```
Update ProductInformation set ProductTypeID = 'ProductTypeID',Quantity =
'Quantity',Price = 'Price',EXP = 'EXP',Origin = 'Origin'  where ProductID = 'ProductID'
```

```
Update Product set ProductName='ProductName',ProductName=
'PictureLink',ProductName= 'Description' where ProductID = 'ProductID'
```

## 8. Delete Product

### a. Class Diagram



### b. Class Specifications

*Product Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|

| | | |
|---|---|---|
| ***01*** | ***doGet()*** | *Get the ID of the Product to be deleted. Make a function call in ProductDAO and ProductInformation to delete the product's data in the child table first, then delete the data from the parent table.* |
| ***02*** | ***doPost()*** | *Since this function only needs to get the product's ID to perform the functions, the doPost() function is not used.* |

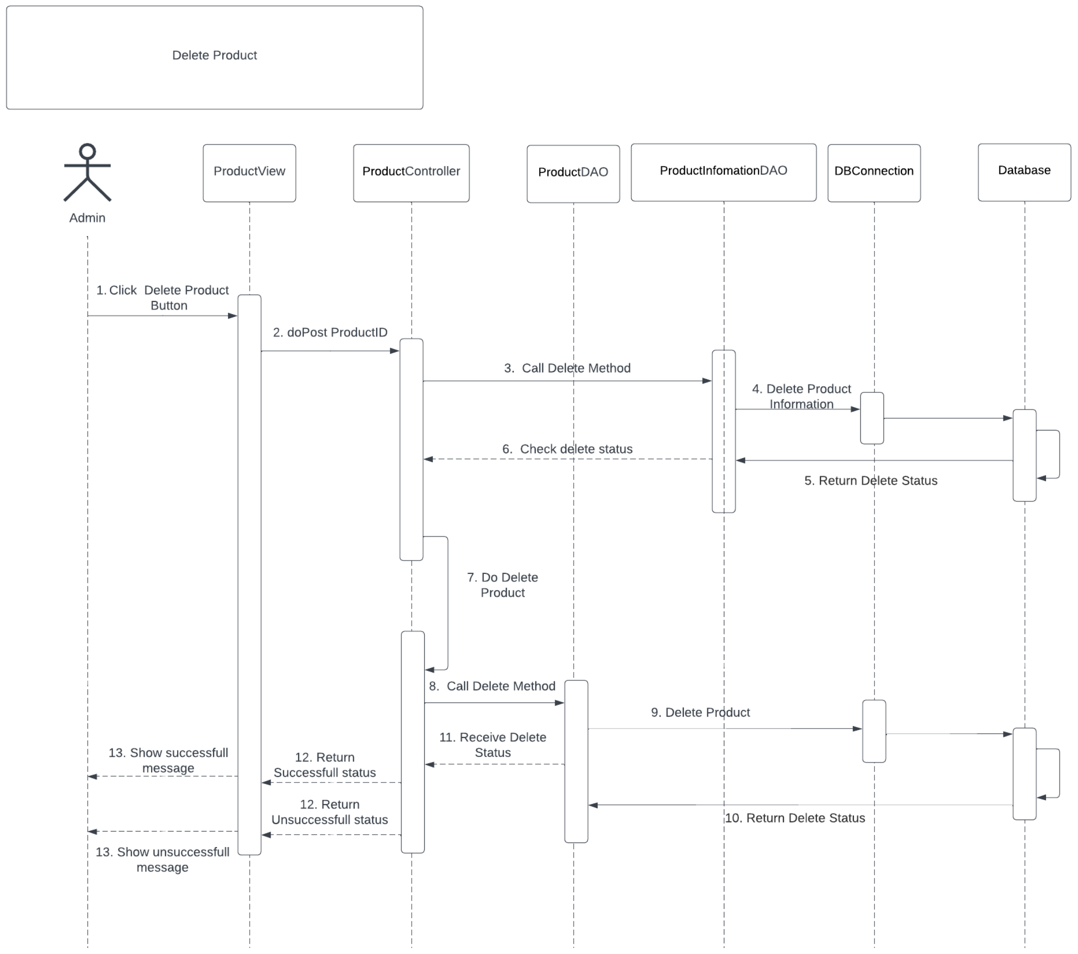
*ProductDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***deleteProduct()*** | *Remove information about that product from the database and update its status again. The function will take one parameter, named the ID of the product to be deleted, and use the SQL statement to perform the deletion. If the deletion is successful, the function will return true, otherwise, it will return false.* |
| ***02*** | ***GetAllProduct()*** | *Query and return all products currently in the database. The result of this function can be used to display to the user.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)



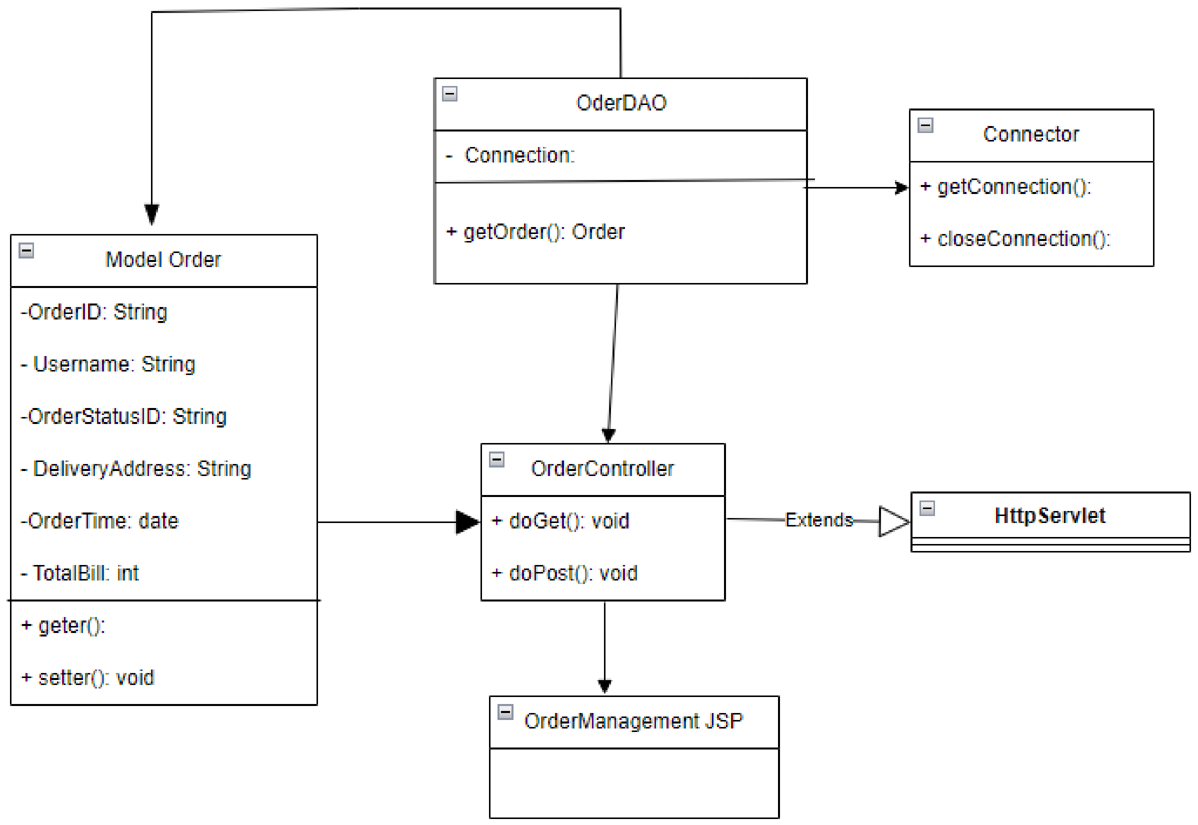
## d. Database queries

Delete from ProductInformation where ProductID = 'ProductID'

Delete from Product where ProductID = 'ProductID'

## 9. View Order Details

### a. Class Diagram



### b. Class Specifications

*Oder Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br><br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br><br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Handle function when viewing order details.*<br><br>*Get the ID of the order when clicking the view order details button. Here will call the function in OrderDAO and OrderDetailsDAO* |

|  |  |  |
|---|---|---|
|  |  | *containing the available query statements to get all the product list of the order from the OrderDetails table* |
| ***02*** | ***doPost()*** | *Since the order can only be viewed without editing anything, the doPost() method is not used* |

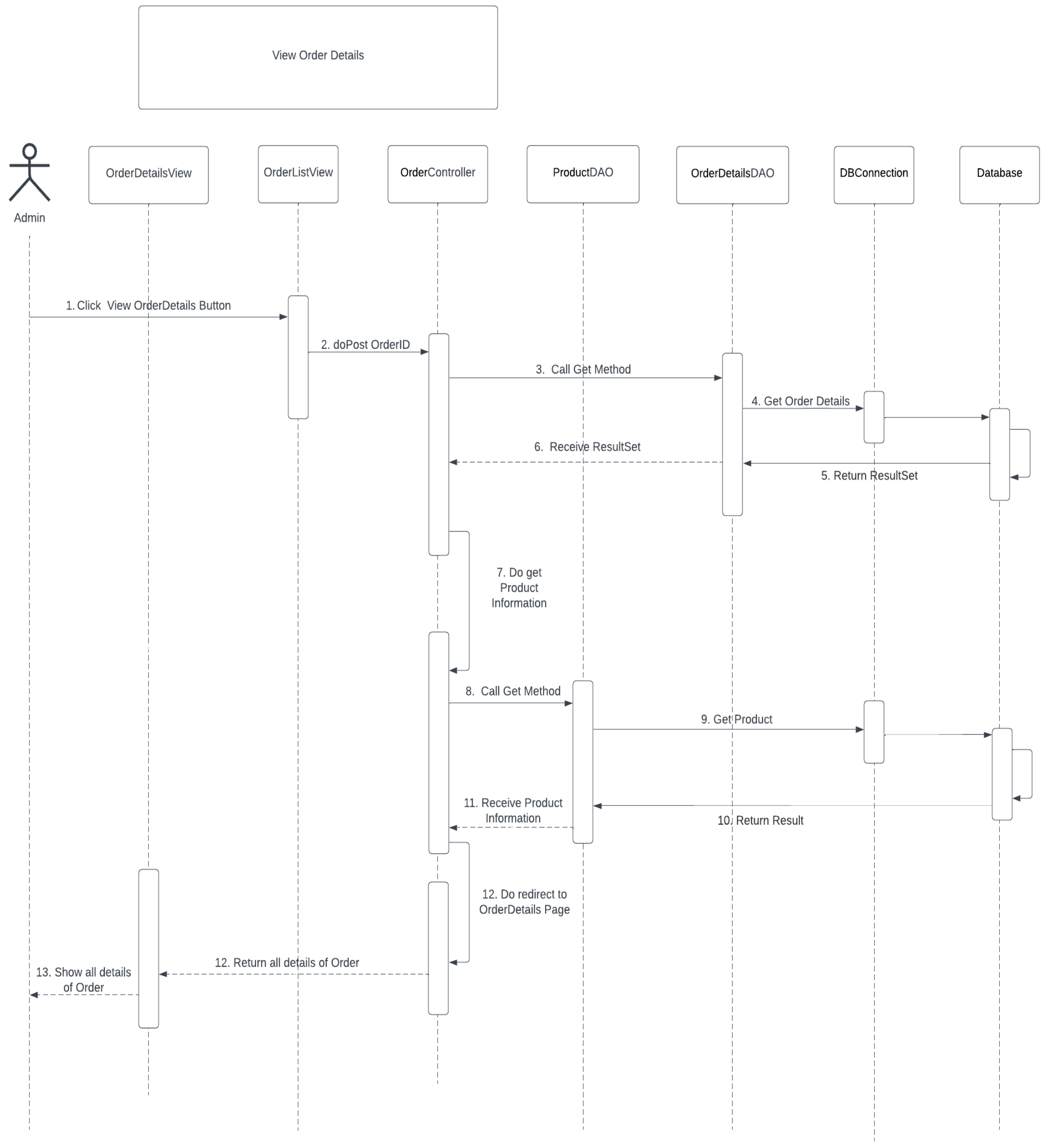
*OrderDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getOrder()*** | *Get order details from database. The function takes an order ID as an argument and returns the corresponding order information.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)

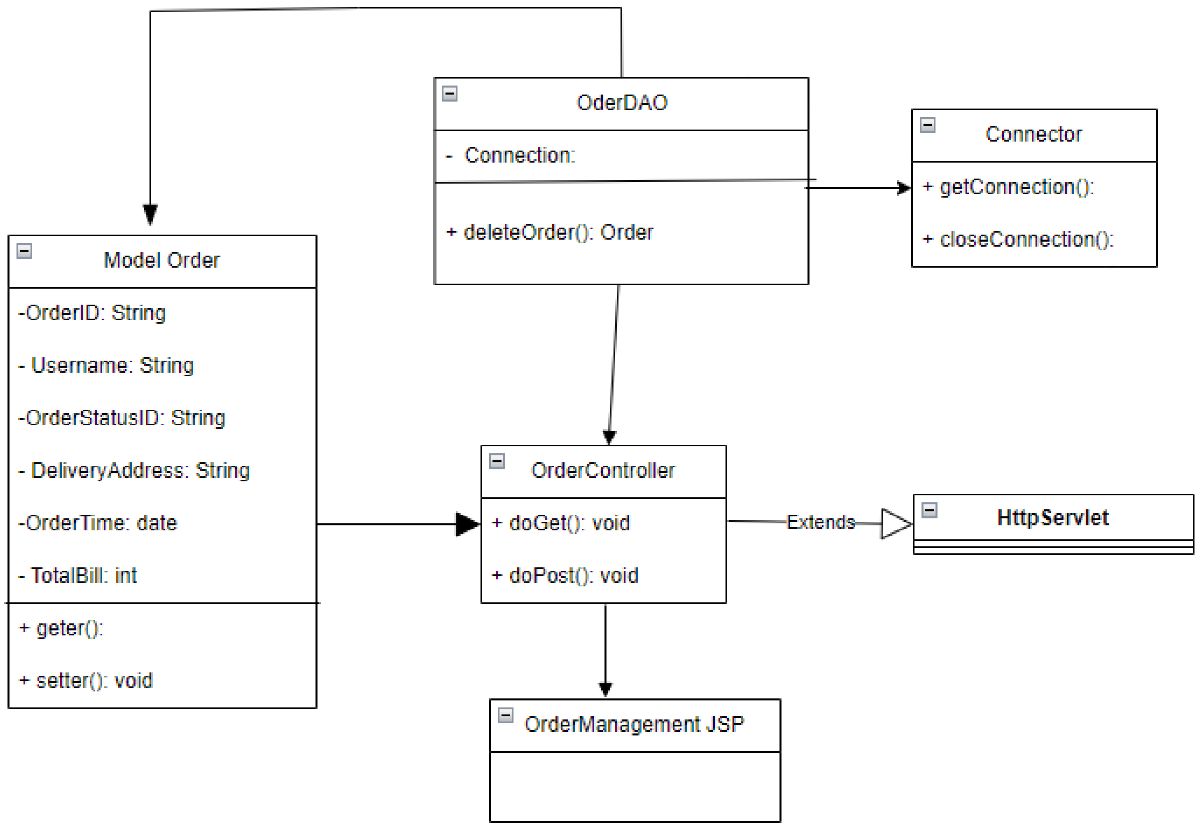


## d. Database queries

Select [OrderList].OrderID,[OrderDetails].ProductID, Product.ProductName, [OrderDetails].Quantity, [OrderDetails].TotalPrice from [Order],[OrderDetails],Product where [OrderList].OrderID = [OrderDetails].OrderID and [OrderDetails].ProductID = Product.ProductID

## 10. Delete Order

### a. Class Diagram



### b. Class Specifications

*OderList Class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getter()*** | *The geter method to get data in this class.*<br>*Output: one of all value in this class* |
| ***02*** | ***setter()*** | *The setter method to set data in this class.*<br>*Input: The data to be set in attribute's class* |

*Controller*

| No | Method | Description |
|---|---|---|
| ***01*** | ***doGet()*** | *Get the ID of the Order to be deleted. Call a function in OrderListDAO and OrderDetailsDAO to delete the order's data in the child table first, then delete the data from the parent table* |

| | | |
|---|---|---|
| ***02*** | ***doPost()*** | *Since this function only needs to get the order's ID to perform the functions, the doPost() function is not used* |

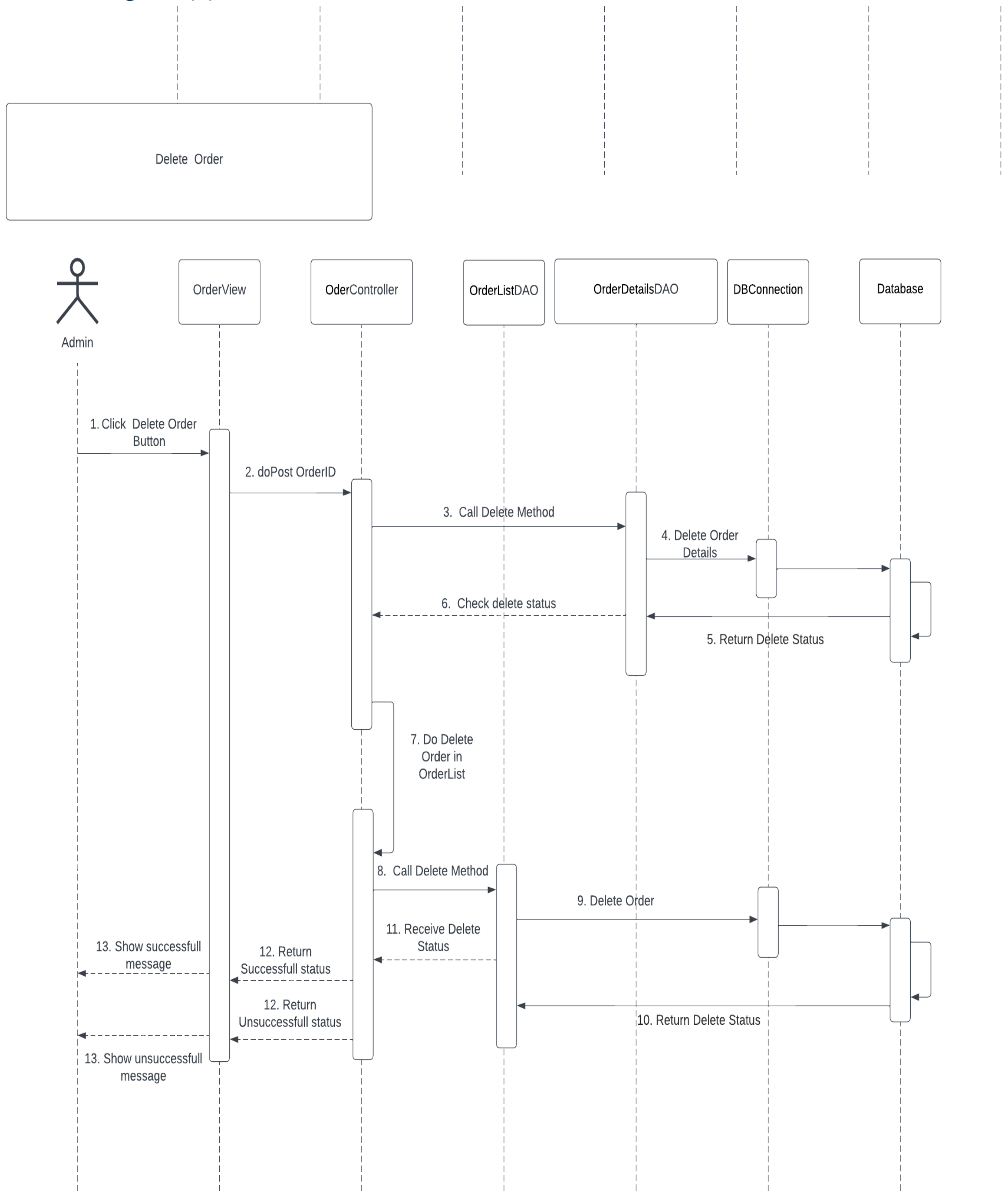
*OrderDAO class*

| No | Method | Description |
|---|---|---|
| ***01*** | ***deleteOrder()*** | *Delete order information from the database. Take the ID of the order to be deleted as a parameter and use SQL statements to delete the corresponding information from the orders table. The result of the function will return a value indicating whether the deletion was successful or failed.* |

*Connector*

| No | Method | Description |
|---|---|---|
| ***01*** | ***getConnector()*** | The **getConnection(String url)** method of Java DriverManager class attempts to establish a connection to the database by using the given database URL. The appropriate driver from the set of registered JDBC drivers is selected. |
| ***02*** | ***closeConnector()*** | By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. |

## c. Sequence Diagram(s)



## d. Database queries

delete from [OrderDetails] where OrderID = ' OrderID '

delete from [OrderList] where OrderID = ' OrderID '

## III. Database Tables

### 1. Account

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | Username | varchar | 20 | x | x | PK | Validation: Username length have to equal or more than 8 characters including letter and number. |
| 2 | Password | varchar | 20 | | x | | Validation: Password length must equal or more than 8 characters including letter (have to include lower and upper letter) and number. |
| 3 | SecurityAnswer | nvarchar | 50 | | x | | Validation: Includes answers to account security questions |

### 2. AccountType

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | AccountTypeID | varchar | 10 | x | x | PK | |
| 2 | AccountTypeName | varchar | 10 | | x | | Validation: Do not be the same to AccountTypeID |

### 3. AccountInformation

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | Username | varchar | 20 | | x | FK | Validation: Username length have to equal or more than 8 characters including letter and number. |
| 2 | AccountTypeID | varchar | 10 | | x | FK | |
| 3 | FullName | nvarchar | 50 | | x | | Validation: Names with numbers or special characters are not accepted |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 4 | PhoneNumber | int |  |  | x |  | Validation:<br>PhoneNumber just accept 10-number. |
| 5 | Gender | varchar | 6 |  | x |  | Validation:<br>Includes only 2 words Male or Female |
| 6 | Email | nvarchar | 50 |  | x |  | Validation:<br>Must have “@” |

## 4. Product

|  | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | ProductID | varchar | 10 | x | x | PK | Validation:<br>2 first characters are brief of product name, 2 next characters are type of product, 2 last characters are number. |
| 2 | ProductName | nvarchar | 50 |  | x |  | Validation:<br>Not same to ProductID |
| 3 | PictureLink | nvarchar | MAX |  | x |  |  |
| 4 | Description | nvarchar | MAX |  | x |  |  |

## 5. ProductType

|  | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | ProductTypeID | varchar | 20 | x | x | PK | Validation:<br>Accented fonts are not accepted |
| 2 | ProductTypeName | nvarchar | 20 |  | x |  | Validation:<br>Do not be the same to ProductTypeID |

## 6. ProductInformation

|  | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | ProductID | varchar | 20 |  | x | FK |  |
| 2 | ProductTypeID | varchar | 20 |  | x | FK |  |
| 3 | Quantity | int |  |  | x |  | Validation:<br>Only accept positive number |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | Price | int | | | x | | Validation: Only accept positive number |
| 5 | EXP | date | | | x | | Validation: Must be date not accept character |
| 6 | Origin | nvarchar | 20 | | x | | |

## 7. OrderStatus

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | OrderStatusID | varchar | 20 | x | x | FK | |
| 2 | OrderStatusName | nvarchar | 20 | | x | FK | Validation: Do not be the same to OrderStatusID |

## 8. OrderList

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | OderID | varchar | 10 | x | x | PK | Validation: 4 last characters are day and month that order is created. |
| 2 | UserName | varchar | 20 | | x | FK | |
| 3 | OrderStatusID | varchar | 10 | | x | FK | |
| 4 | DeliveryAddress | nvarchar | MAX | | x | | Validation: Includes all delivery information |
| 5 | OrderTime | datetime | | | x | | Validation: Include date and time customer start order |
| 6 | TotalBill | int | | | x | | Validation: Must greater than 0 and is not a negative number |

## 9. OrderDetails

| | Field name | Type | Size | Unique | Not Null | PK/FK | Notes |
|---|---|---|---|---|---|---|---|
| 1 | OrderID | varchar | 20 | | x | FK | Validation: |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  | 4 last characters are day and month that order is created. |
| 2 | ProductID | varchar | 20 |  | x | FK | Validation:<br>2 first characters are brief of product name, 2 next characters are type of product, 2 last characters are number. |
| 3 | Quantity | int |  |  | x |  | Validation:<br>Only accept positive number |
| 4 | TotalPrice | int |  |  | x |  | Validation:<br>Must greater than 0 and is not a negative number |