

**<<Online Shop>>**

**Software Design Specification**

– Hanoi, August 2022 –

Record of changeS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | A\* M, D | In charge | Change Description |
| 2/3 | A | I.1 | Add Code Packages |
| 2/3 | A | I.2.a | Add Db Schema |
| 3/3 | M | I.1 | Update Code Packages |
| 3/3 | A | I.2.b | Add Db Descriptions |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview 4](#_Toc96516286)

[1. Code Packages 4](#_Toc96516287)

[2. Database Design 5](#_Toc96516288)

[a. Database Schema 5](#_Toc96516289)

[b. Table Description 5](#_Toc96516290)

[II. Code Designs 10](#_Toc96516291)

[1. <Feature/Function Name1> 10](#_Toc96516292)

[a. Class Diagram 10](#_Toc96516293)

[b. Class Specifications 10](#_Toc96516294)

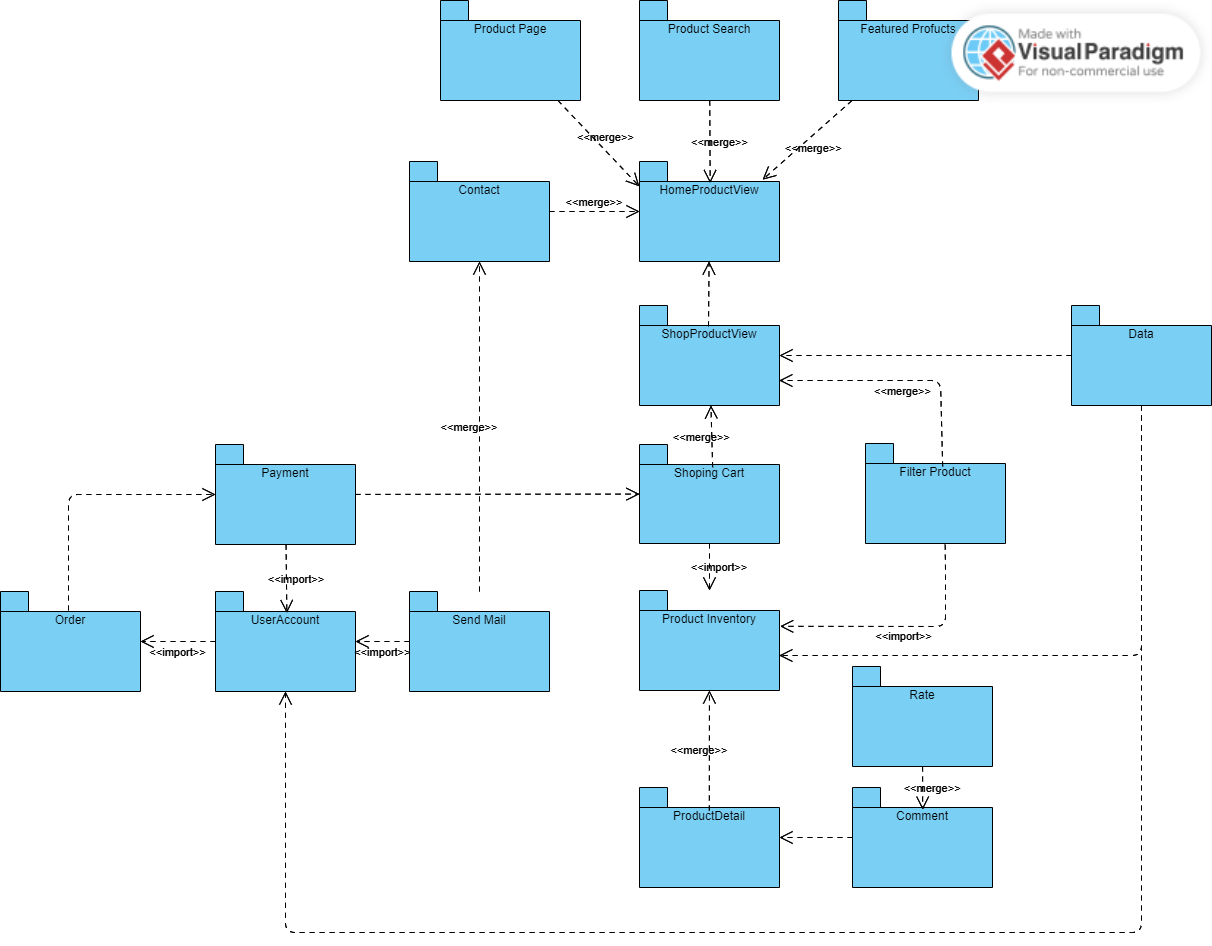
[c. Sequence Diagram(s) 10](#_Toc96516295)

[d. Database queries 11](#_Toc96516296)

[2. <Feature/Function Name2> 11](#_Toc96516297)

# I. Overview

## 1. Code Packages

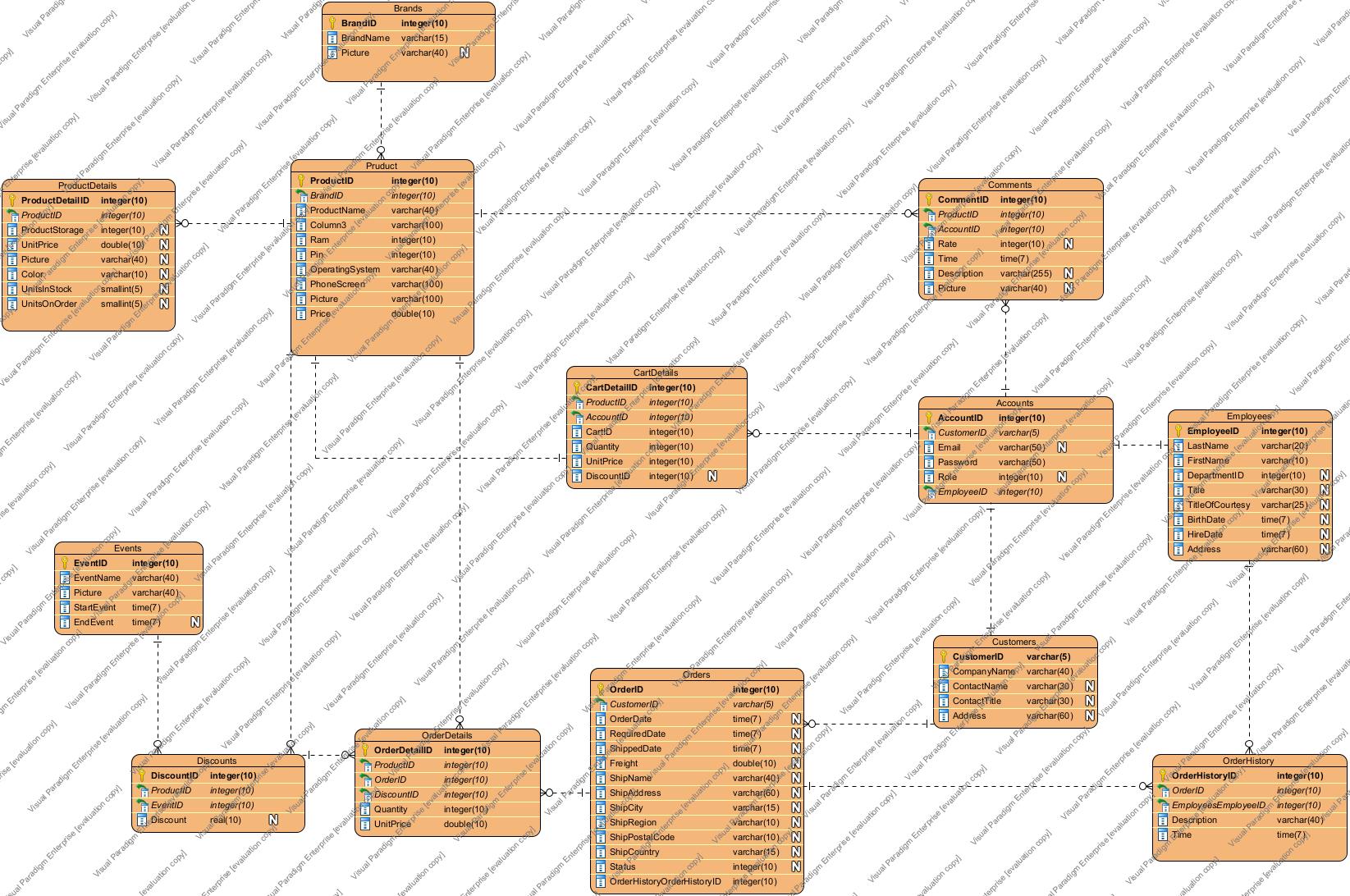


***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 1 | Home Product View | This refers to the landing page or homepage of a website or app that showcases various products or services that a business or organization offers. |
| 2 | Show Product View | This refers to a page or section on a website or app that displays the details and images of a specific product |
| 3 | Shopping Cart | This refers to the virtual cart or basket that users or customers use to store items they want to purchase while browsing a website or app. |
| 4 | Product Page | This refers to the individual page on a website or app that provides detailed information about a specific product. |
| 5 | Product Search | This refers to the feature on a website or app that allows users to search for specific products or services based on keywords, filters, or other criteria. This can help users find the products they are looking for more quickly and easily. |
| 6 | Featured Products | Featured and popular products are the best-selling items for your product, product range, or company. However, a featured product could be a new product line or seasonal products instead. |
| 7 | Product Inventory | This refers to a database or list of all the products that a company or organization has available for sale or distribution. |
| 8 | Product Detail | This refers to the specific information and attributes of a particular product, such as its features, specifications, size, weight, materials, and so on. |
| 9 | Comment | This refers to the ability for users or customers to leave feedback or reviews about a product or service. This can be done through a variety of channels, such as a comments section on a website, a social media platform, or an email. |
| 10 | Rate | This refers to the system or process for assigning a numerical or qualitative rating to a product or service based on its quality, value, or other factors. This can be done by customers or users, as well as by experts or reviewers. |
| 11 | Data | This refers to the information that is collected and stored about products, customers, orders, and other aspects of a business or organization. This data can be used for analysis, reporting, forecasting, and other purposes. |
| 12 | Contact | This refers to the methods by which customers or users can get in touch with a business or organization, such as phone, email, chat, or social media. |
| 13 | Send Mail | This refers to the feature that allows users or customers to send a message or request to a business or organization via email or other communication channels. |
| 14 | User Account | This refers to the personal profile or account that a user or customer creates on a website or app in order to access certain features or services. This can include details such as name, email address, password, payment information, and order history. |
| 15 | Payment | This refers to the process by which customers or users pay for products or services. This can include various payment methods such as credit cards, debit cards, PayPal, Apple Pay, and so on. |
| 16 | Order | This refers to the specific transaction or request made by a customer or user to purchase a product or service. |

## 2. Database Design

### a. Database Schema



### 

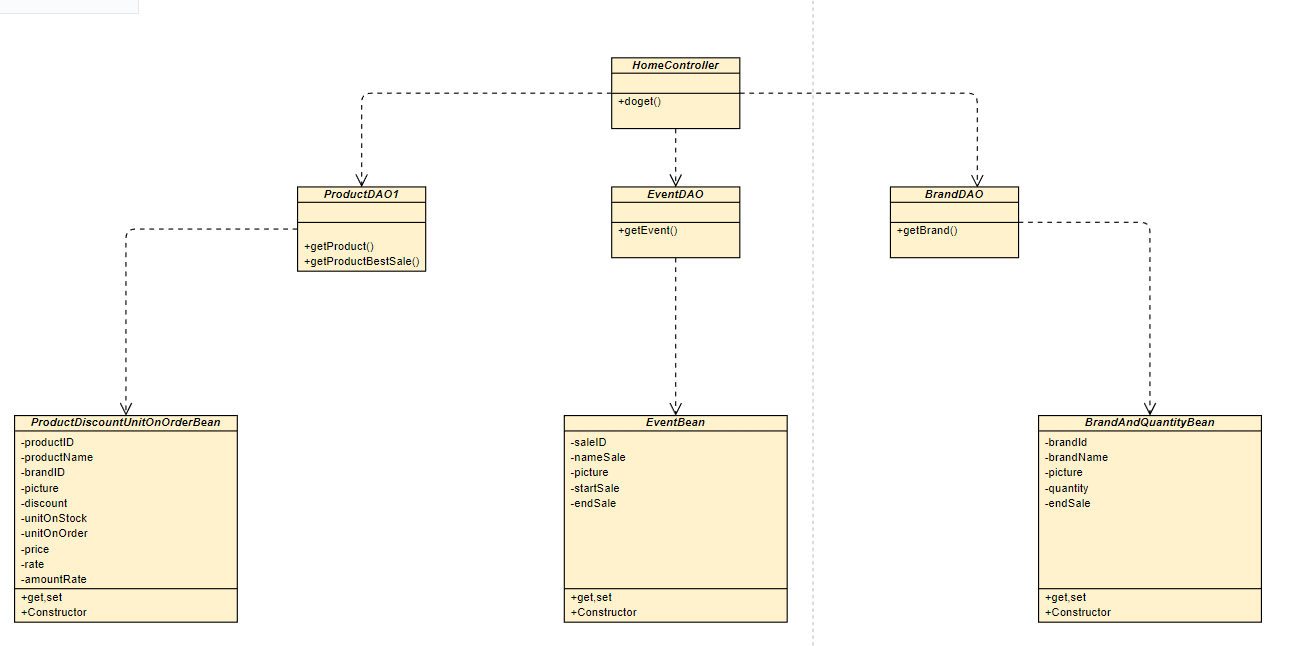
### b. Table Description

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 1 | Accounts | Accounts (Entity): The account of someone when accessing an online shopping website.  AccountID: Used to identify the account record.  Email: The account of the person who owns the account.  Password: The password of the person who owns the account.  CustomerID: ID of customer in Customers table.  EmployeeID: ID of employee in Employees table.  Role: Used to determine the user's function in the online shopping website (0 Admin, 1 Employee, 2 Customer).  Status: The status of the account (1: Active,2: Banned). |
| 2 | Customers | Customers (Entity): Customers who own an account can sign in to the store's website.  CustomerID: Used to identify the  customer record in the Customers table.  CompanyName: The company name of the customer. ContactName: The name of the customer.  ContactTitle: The position of the customer. Address: The address of the customer. |
| 3 | Employees |  |
| 4 | Comments | Comments (Entity): Customers' comments on the products available in the store.  CommentID: Used to identify the comment record in the Comments table.  ProductID: ID of the product that the customer comments on in the Products table.  AccountID: ID of the account that comments in the Accounts table.  Status: The status of the comment (1: Active, 2: Delete).  Rate: The customer's rating for this product, each account can rate a product only once.  Time: The time the user posted this comment.  Description: The user's comment on the product.  Picture: The picture that the user posts. |
| 5 | Products | Products (Entity): Products sold on the online shopping website  ProductID: Used to identify the product record in the Products table  ProductName: The name of the product  BrandID: The ID of the product's brand  Chip: The name of the chip in the product  Pin: The type of battery in the product  OperatingSystem: The operating system of the product  PhoneScreen: The specifications of the product's screen  Picture: The picture describing the product  Price: The product's price (Take the lowest product price in the ProductDetails table). |
| 6 | ProductDetails | ProductDetails (Entity): Detailed information about the product  ProductDetailsID: Used to identify the product detail record in the ProductDetails table  ProductID: The ID of the product that has this ProductDetail  ProductStorage: The capacity of the product  UnitPrice: The price of the product  Picture: The illustration image of the product with the properties in this table  Color: The color of the product  UnitsInStock: The number of items in stock  UnitsOnOrder: The number of items on order. |
| 7 | HistoryProducts | HistoryProducts (Entity): The history of updating, deleting, and creating a product  HistoryID: Used to identify the HistoryProduct record in the HistoryProducts table  AccountID: The ID of the account that performed this history  ProductID: The ID of the product that was updated, created, or deleted  ProductName: The name of the product  BrandID: The ID of the product's brand  Chip: The name of the chip in the product  Pin: The type of battery in the product  OperatingSystem: The operating system of the product  PhoneScreen: The specifications of the product's screen  Picture: The picture describing the product  Price: The product's price (Take the lowest product price in the ProductDetails table). |
| 8 | HistoryProductDetails | HistoryProductDetails (Entity): History of updates, deletions, and creations of ProductDetail  HistoryProductDetailID: Used to identify the HistoryProductDetail record in the HistoryProductDetailIDs table  AccountID: ID of the account that performed this history  ProductID: ID of the product that has this ProductDetail  ProductStorage: Storage capacity of the product  UnitPrice: Price of the product  Picture: Illustration picture of the product with its attributes  Color: Color of the product  UnitsInStock: Quantity in stock  UnitsOnOrder: Quantity on order. |
| 9 | Events | Events (Entity): The events within a certain time period of the store  EventID: Used to identify the Event record in the Events table  EventName: The name of the event  Picture: The image of the event  StartEvent: The start time of the event  EndEvent: The end time of the event. |
| 10 | Discounts | Discounts (Entity): The discount of a product in the store  DiscountID: Used to identify the Discount record in the Discounts table  ProductID: The ID of the product that has this discount  EventID: The ID of the event of the discount  Discount: The percentage of discount. |
| 11 | Orders | Orders (Entity): The customer's order when purchasing a product  OrderID: Used to identify the Order record in the Orders table  CustomerID: The ID of the customer who placed this order  OrderDate: The time the order was approved  RequiredDate: The time the product is requested to be shipped to the customer  ShippedDate: The actual time the product was shipped to the customer  Freight: The weight of the order  ShipName: The name of the customer who receives the shipment  ShipAddress: The address of the shipment  ShipCity: The shipping company  ShipRegion: The province of the shipment  ShipPostalCode: The shipment code  ShipCountry: The country of the shipment  Status: The status of the order (0: Order Deleted, 1: Approved, 2: Pending Approval) |
| 12 | OrderDetails | OrderDetails (Entity): The details of the order  OrderDetailID: Used to identify the OrderDetail record in the OrderDetails table  OrderID: The ID of the order that owns this OrderDetail  DiscountID: The ID of the Discount of this OrderDetail  ProductDetailID: The ID of the product in this OrderDetail  Quantity: The quantity of the product. |
| 13 | OrderHistories | OrderHistories: The history of changes in the status of an Order  OrderHistoryID: Used to identify the OrderHistory record in the OrderHistories table  EmployeeID: The ID of the employee who changed the status  Description: Approve or disapprove the order  Time: The time when the Order status was changed  OrderID: The ID of the order that was changed. |
| 14 | CartDetails | CartDetails (Entity):  CartDetailID:  DiscountID:  ProductDetailID:  Quantity: |
| 15 | Brands | Brands (Entity): The brand of the products in the shop  BrandID: Used to identify the Brand record in the Brands table  BrandName: The name of the brand  Picture: The logo of the brand. |
|  |  |  |

# II. Code Designs

## 1. <Home>

### a. Class Diagram



### b. Class Specifications

*ProductDAO1*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getProduct() | Get list of products sorted by input status.  Input int 1: Sorted by Discount.  Input int 2: Sorted by UnitOnOrder.  Input int 3: Sorted by UnitOnStock |
|  |  |  |

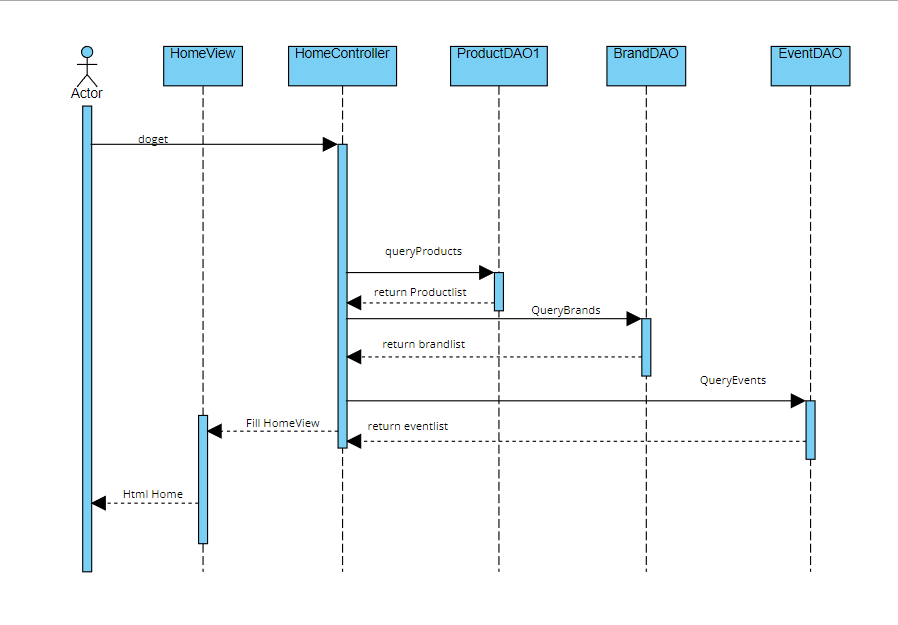
***EventDAO***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getEvent(); | Get list of event have exist this time. |
|  |  |  |

***BrandDAO***

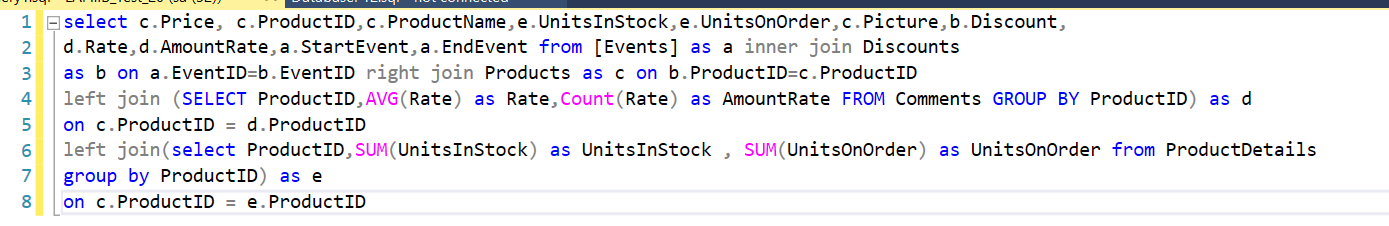
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getBrand(); | Get list of brand and quantity product of this brand. |
|  |  |  |

### c. Sequence Diagram(s)



### d. Database Queries

**SQL method getProduct():**



inner join with two table Event and Discount on ProductID

Right join with Product to take all Product.

Left join with query(Comment table Group by ProductID to take AVG(Rate), Count(Rate))

Left join with query(ProductDetailts Group by ProductID to take Sum(UnitInStock) and Sum(UnitsOnOrder))

**SQL method getEvent():**

select \* from [Events]:

Take all event form table event.

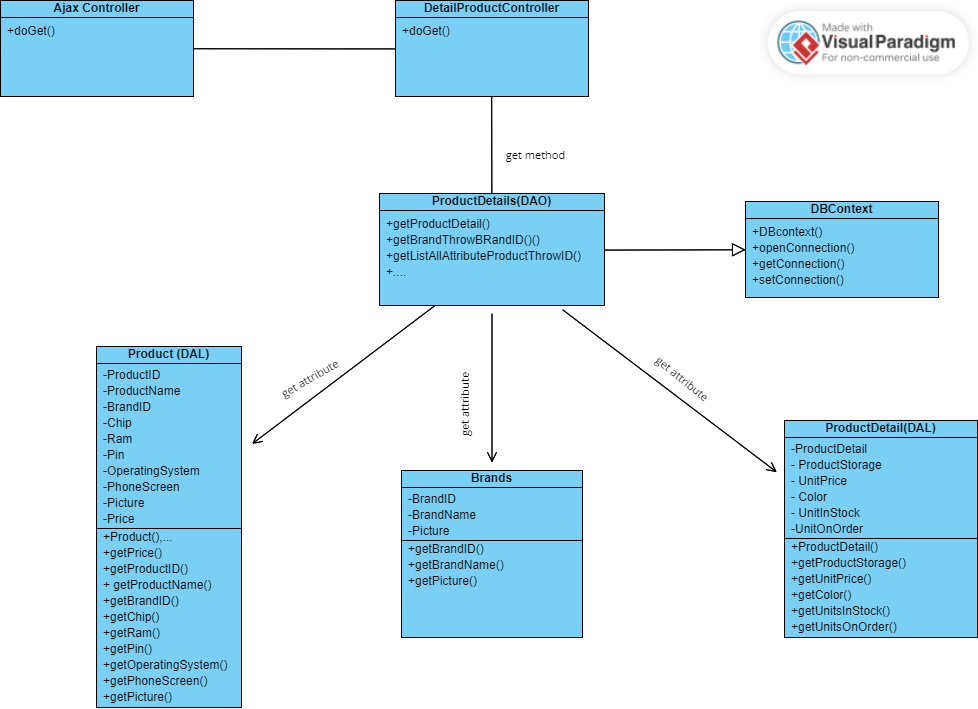
**SQL method getBrand():**

Select \* from Brands inner join (select BrandID,Count(ProductID) as Quantity from Products group by BrandID)as b ON Brands.BrandID=b.BrandID

Brands inner join with query( Products table group by BrandID to take Quantity of Product) on BrandID

## 2. <Detail Product >

### a. Class Diagram

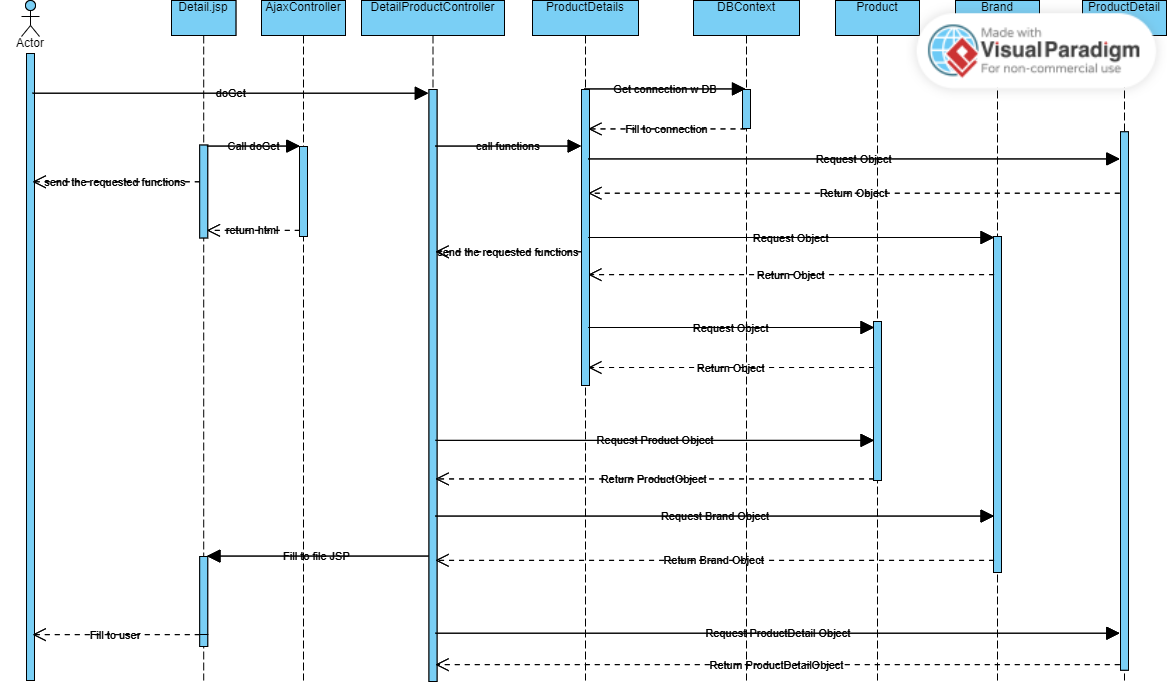


### b. Class Specifications

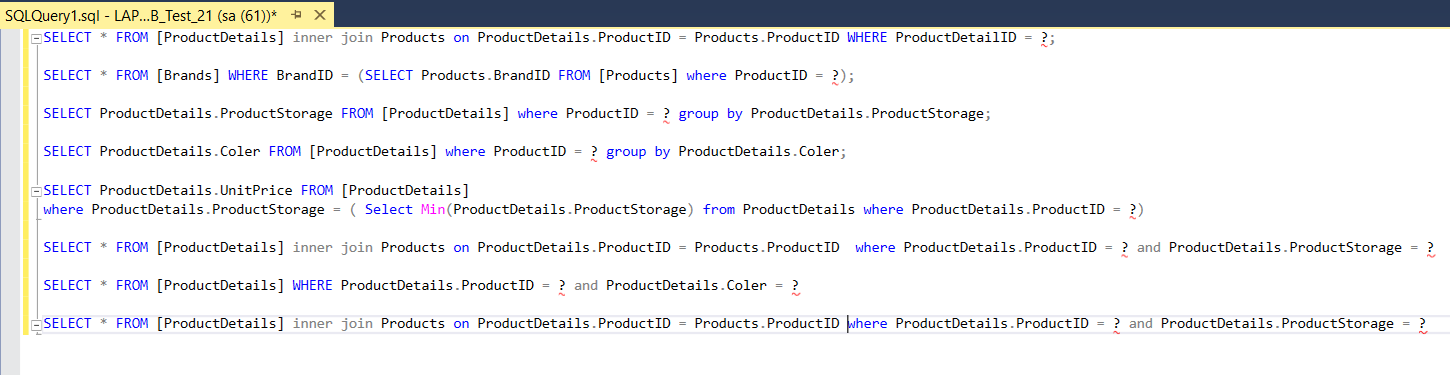
*ProductDetails (DAO)*

|  |  |  |
| --- | --- | --- |
| No | Method | Description |
| 01 | getProductDetail() | Get a ProductDetail object. |
| 02 | getBrandThrowBrandID() | Get all of attribute of Brand Object. |
| 03 | getListAllAttributeProductThrowID() | Get list ProductDetail Objects by ProductDetaiIID |
| 04 | getListStorageProductThrowID() | Get list storage in ProductDetail table by ProductID inner join Products table |
| 05 | getListColerProductThrowID() | Get list color in ProductDetail by ProductID inner join Products table |
| 06 | getSpecifiByIDAndStorage() | Get specification of a product |

### c. Sequence Diagram(s)



### d. Database Queries



**SQL method productDetail():**

1. SELECT \* FROM [ProductDetails] inner join Products on ProductDetails.ProductID = Products.ProductID WHERE ProductDetailID = ?;
2. SELECT ProductDetails.ProductStorage FROM [ProductDetails] where ProductID = ? group by ProductDetails.ProductStorage;
3. SELECT ProductDetails.Coler FROM [ProductDetails] where ProductID = ? group by ProductDetails.Coler;
4. SELECT ProductDetails.UnitPrice FROM [ProductDetails]

where ProductDetails.ProductStorage = ( Select Min(ProductDetails.ProductStorage) from ProductDetails where ProductDetails.ProductID = ?)

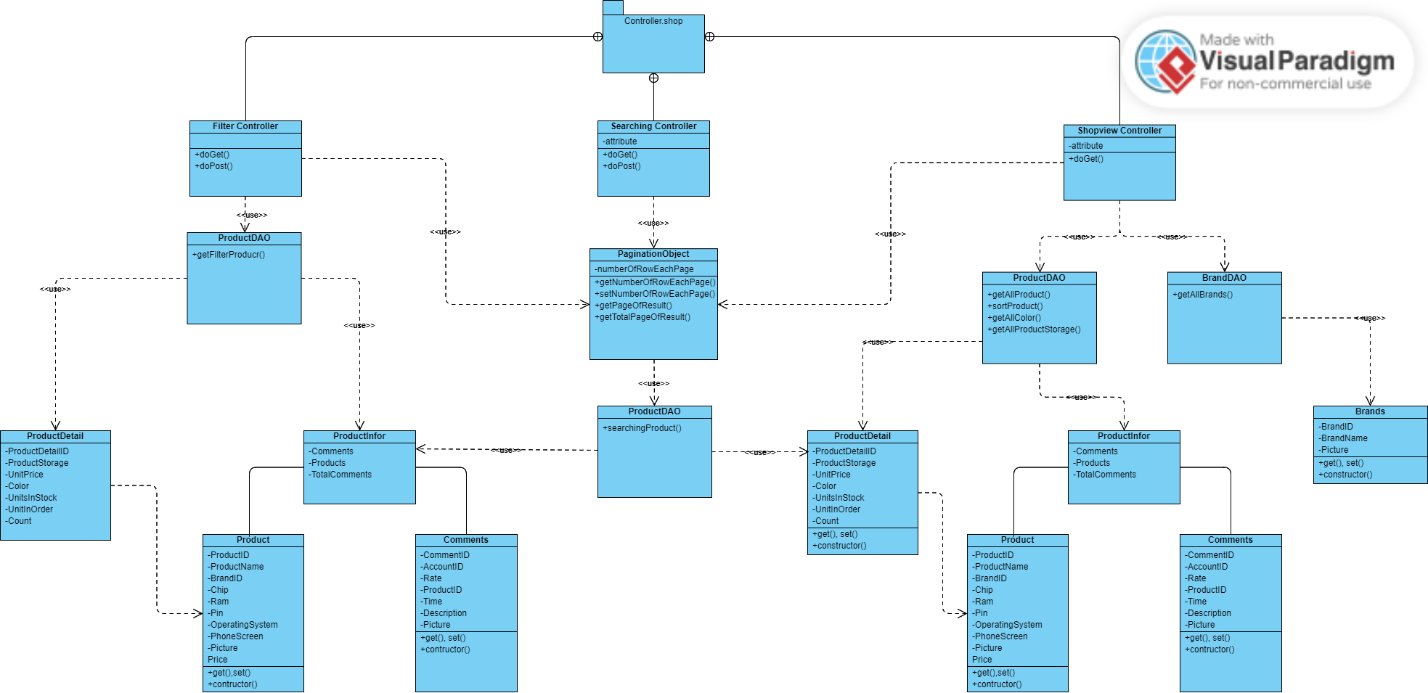
1. SELECT \* FROM [ProductDetails] inner join Products on ProductDetails.ProductID = Products.ProductID where ProductDetails.ProductID = ? and ProductDetails.ProductStorage = ?
2. SELECT \* FROM [ProductDetails] WHERE ProductDetails.ProductID = ? and ProductDetails.Coler = ?
3. SELECT \* FROM [ProductDetails] inner join Products on ProductDetails.ProductID = Products.ProductID where ProductDetails.ProductID = ? and ProductDetails.ProductStorage = ?

**SQL method Brand():**

SELECT \* FROM [Brands] WHERE BrandID = (SELECT Products.BrandID FROM [Products] where ProductID = ?);

## 3. <Shop>

### a. Class Diagram

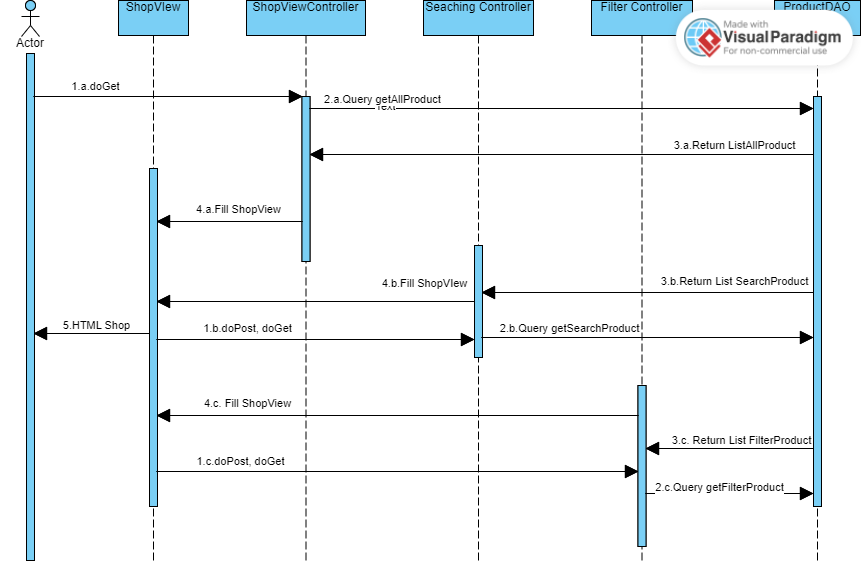


### b. Class Specifications

ProductDAO

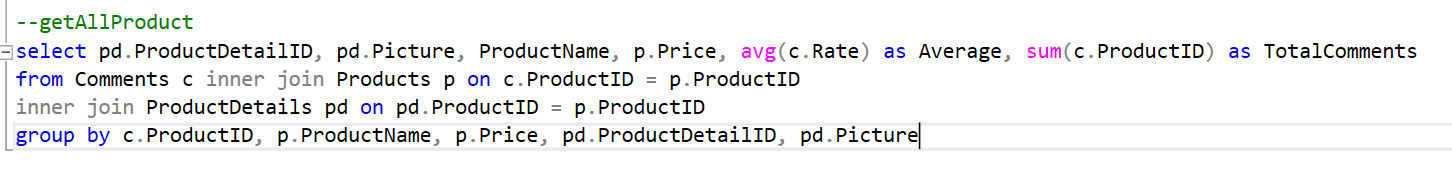
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getFilterProduct() | Get list product by Filter Condition in database |
| 02 | getAllProduct() | Get All product in database |
| 03 | getAllColor() | Get All color in database |
| 04 | getAllProductStorage() | Get All storage in database |
| 05 | sortProduct() | Get list product which sorted by sort’s condition |
| 06 | searchProducts | Get list product which searched by search’s condition |

### c. Sequence Diagram(s)

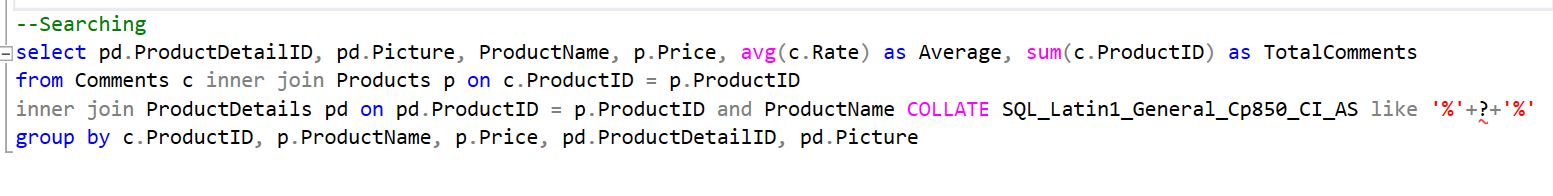


### d. Database Queries

**SQL method getAllProduct():**

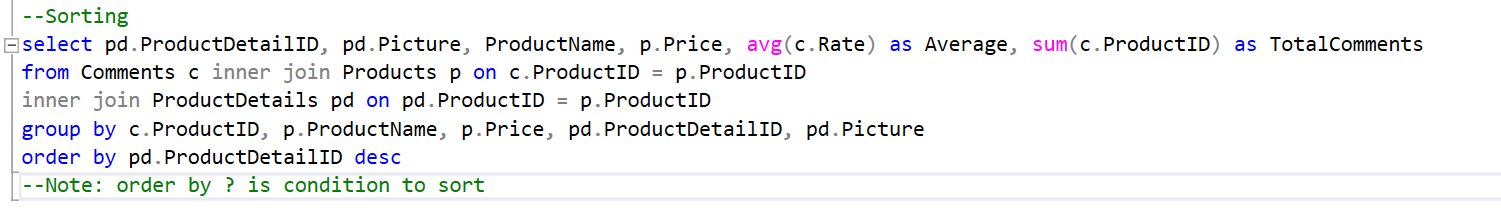


**SQL method searchProducts():**

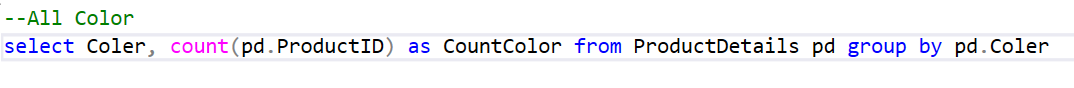
****

Note: “?” meaning condition to search

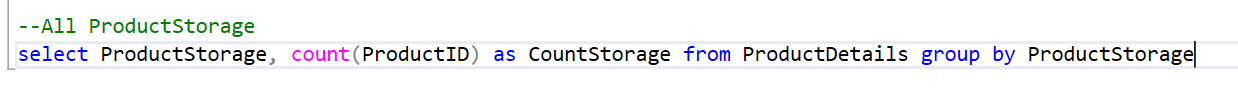
**SQL method sortProducts():**

****

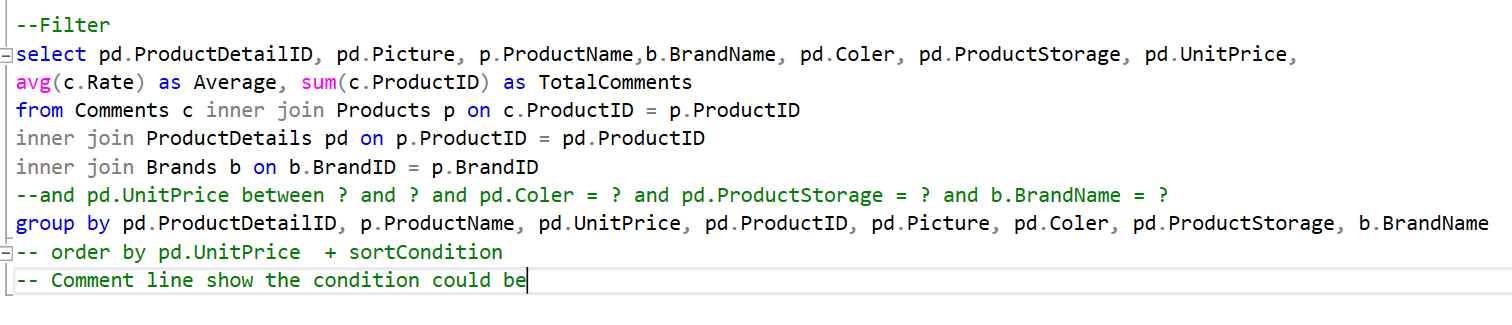
**SQL method getAllColor():**

****

**SQL method getAllProductStorage():**

****

**SQL method getFilterProduct ():**

****

## 4. <Contact>

### a. Class Diagram

Diagram, box and whisker chart

Description automatically generated

### b. Class Specifications

*ProductDAO1*

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | sendEmail() | Get information from users and send to admin email.  Input 1: Email.  Input 2: Name.  Input 3: Subject.  Input 4: Message. |

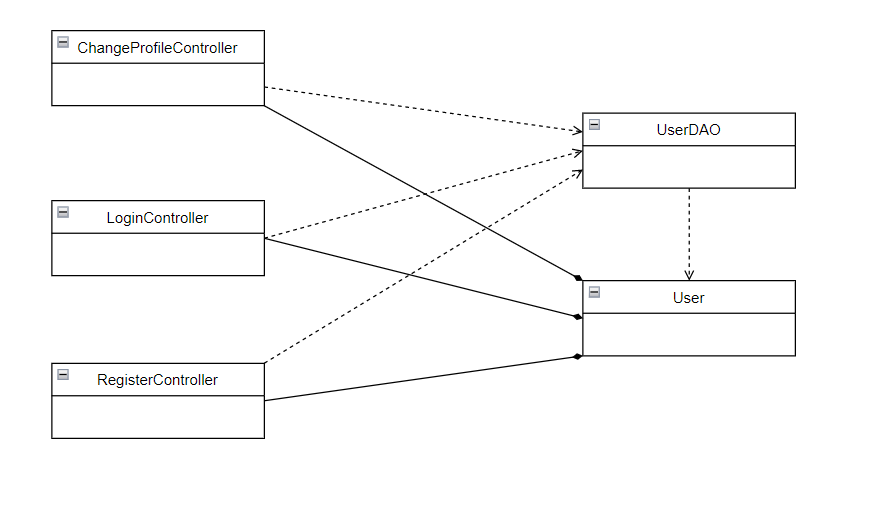
### c. Sequence Diagram(s)

Diagram

Description automatically generated

## 5. <Authentication>

### a. Class Diagram



### b. Class Specifications

***UserDao***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | getUsers() | get user by email, password |
| 02 | checkAccountExist() | check if email register has already exist in database |
| 03 | register() | insert new user with username, email, name into database |
| 04 | updateProfile() | update profile of an user in database |

**ChangeProfile**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost() | Load the form, get data and update data into database |
|  |  |  |

**Login**

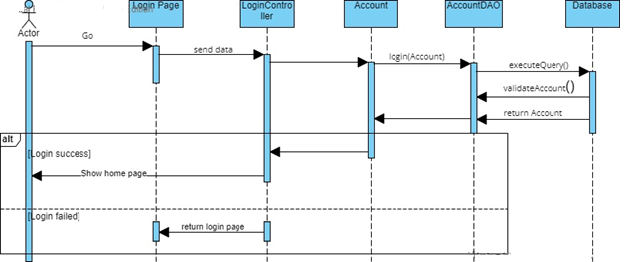
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | doPost() | get data and check if user exist |
|  |  |  |

**Register**

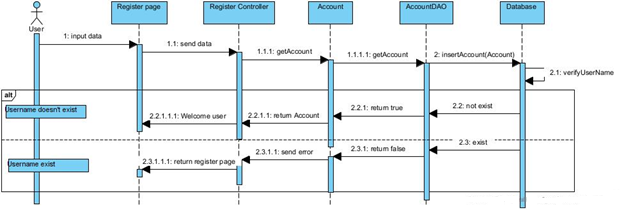
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | processRequest() | get data to insert new user into database |

### c. Sequence Diagram(s)

**Sign In**



**Sign Up**



### d. Database Queries

checkAccountExist(): select \* from [User] where email =?

register(): insert into [User](username,password,email,role\_id) Values(?,?,?,0)

getUsers(): select \* from [User] u where u.email = ? and password = ?

updateProfile(): update [User] set username = ?, about\_me = ?,  connect = ?  where user\_id =?