

**SaleGas website**

**PROJECT**

**Software Design Specification**

– SPRING 2023 –

**Record of changeS**

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

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Introduction 4](#_gjdgxs)

[1. Purpose 4](#_30j0zll)

[2. System Overview 5](#_1fob9te)

3[. Design Map 5](#_1fob9te)

[II. Overview 4](#_gjdgxs)

[1. Code Packages 4](#_30j0zll)

[2. Database Design 5](#_1fob9te)

[a. Database Schema 5](#_3znysh7)

[b. Table Description 5](#_2et92p0)

c[. Development Tools 5](#_2et92p0)

[III. Code Designs](#_tyjcwt) 9

[1. Public Account](#_3dy6vkm) 9

[a. Class Diagram](#_1t3h5sf) 9

[b. Class Specifications](#_4d34og8) 10

[c. Sequence Diagram(s)](#_2s8eyo1) 12

[2. Public Order](#_17dp8vu) 20

[a. Class Diagram](#_3rdcrjn) 20

[b. Class Specifications](#_26in1rg) 20

[c. Sequence Diagram(s)](#_lnxbz9) 21

[3.Public Product](#_qsh70q) 23

[a. Class Diagram 2](#_3as4poj)3

[b. Class Specifications 2](#_1pxezwc)4

[c. Sequence Diagram(s) 2](#_49x2ik5)5

[4. Public News 2](#_35nkun2)7

[a. Class Diagram 2](#_1ksv4uv)7

[b. Class Specifications 2](#_44sinio)8

[c. Sequence Diagram(s) 2](#_2jxsxqh)8

[5. Public Category](#_z337ya) 30

[a. Class Diagram](#_3j2qqm3) 30

[b. Class Specifications](#_1y810tw) 30

[c. Sequence Diagram(s)](#_4i7ojhp) 33

[6. Public Warranty](#_2xcytpi) 33

[a. Class Diagram](#_1ci93xb) 33

[b. Class Specifications](#_3whwml4) 33

[c. Sequence Diagram(s)](#_2bn6wsx) 34

# 

# I. Introduction

## 1. Purpose

The purpose of the SaleGas website is to provide a platform for the sale and management of stoves, gas, gas accessories and news updates. This system allows the manager to upload products, manage sales, manage customers. Users can buy goods online at the website, read the news posted by SaleGas.

## 2. System Overview

Our website describes the purpose and goals of the SaleGas, identifies stakeholders such as customers, administrators, suppliers, payment gateway providers, and delivery partners. The SRS highlights the high-level functionality of the system, including features like product categories, search and filtering options, user registration and authentication, shopping cart, secure payments, order management, shipping and delivery integration, and customer support. Assumptions and constraints are also mentioned, such as web-based deployment, integration with third-party systems, compliance with regulations, and adherence to budget and timeline. Additionally, it acknowledges the potential need for interfacing with other existing systems.

## 3. Design Map

User Interface (UI):

● Home page: Displays banners, categories, newly updated, best-selling products and products by category.

● product shop: Displays products with images, descriptions, prices, and filtering/sorting options. Users can optionally display products in grid view and list view

● Product details: Provides detailed information about a specific product type.

● Cart: Allows users to add/remove items, adjust quantity and view total cost.

● Payment: Collect shipping and payment details to complete the order.

● User Accounts: Allows registration, login, and user profile management.

Database:

● User data: Stores user information such as name, email, phone number, address.

● Product Data: Contains information about available products, including specifications and quantity, product name price, product image.

● Order data: Track orders placed, including item, quantity and status.

● Authentication and authorization:

● User Registration: Allows new users to create accounts.

● Sign In/Log Out: Authenticate the user to access personalized features.

● Access control: Ensures access is allowed to certain actions or pages (e.g. only logged in users can place orders).

Core Functionality:

● Product management: Allows the administrator to add, edit, and delete kitchen products from the system.

● Sales management allows the administrator to view the sales.

● Cart management: Handle adding/removing items from cart and updating quantity.

● Warranty management: Handle customer's warranty claim.

● Order processing: Manage the workflow from order to fulfillment, including updating order status.

● Payment Gateway Integration: Integrate with a secure payment gateway to process online payments.

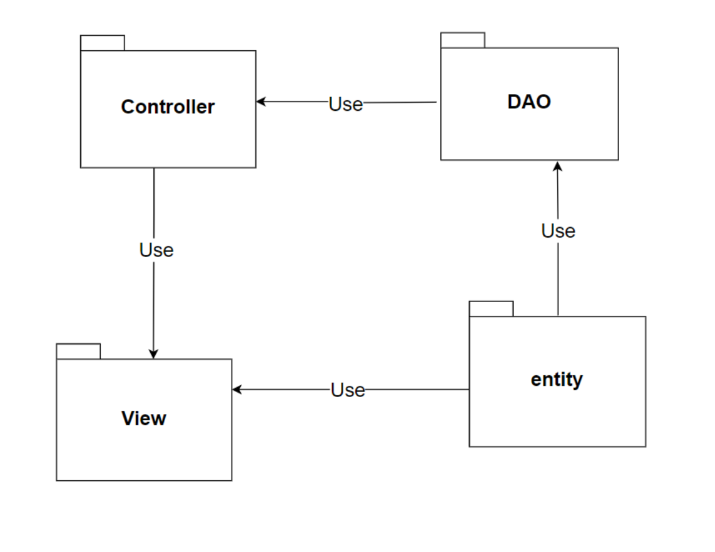
Search and Filtering:

● Allows users to search for kitchens based on keywords, categories, price ranges and other relevant filters.

● Provide search results pages with relevant product listings.

# II. Overview

## 1. Code Packages



***Package descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Package** | **Description** |
| 1 | Controller | Controllers have classes related to the management of user requests, including navigation agents, control functions, routing methods, and error handling. |
| 2 | DAO | DAO has the processes in data processing, including methods to access the database and modify data in the database. |
| 3 | Entity | Entity has objects that represent tables in the database. |
| 4 | View | View is what the website displays to the user and interacts with the user. |

## 2. Database Design

### a. Database Schema



### b. Table Description

Supplier

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | supplierID | int |  |  |  | x | PK |
| 2 | supplierName | nvarchar | 255 |  | x |  |  |
| 3 | companyName | nvarchar | 255 |  |  |  |  |
| 4 | contactInfor | varchar | 100 | x |  |  |  |
| 5 | phoneContact | varchar | 15 |  |  |  |  |
| 6 | [address] | nchar | 255 |  |  |  |  |
| 7 | [country] | nvarchar | 255 |  |  |  |  |
| 8 | [status] | tinyint |  |  |  |  |  |

Category:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | CategoryId | int |  | x |  | x | PK |
| 2 | CategoryName | nvarchar | 255 |  | x |  |  |
| 3 | [image] | nvarchar | max |  |  |  |  |

Product:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | ProductId | int |  |  |  | x | PK |
| 2 | supplierID | int |  |  | x |  | FK |
| 3 | productName | nvarchar | 255 |  | x | x |  |
| 4 | categoryID | int |  |  | x |  | FK |
| 5 | price | money |  |  |  |  |  |
| 6 | WarrantyPolicyID | int |  |  |  |  |  |
| 7 | quantity | int |  |  | x |  |  |
| 8 | productDetail | ntext |  |  |  |  |  |
| 9 | describe | ntext |  |  |  |  |  |
| 10 | discount | float |  |  |  |  |  |
| 11 | image | nvarchar | max |  |  |  |  |
| 12 | status | tinyint |  |  |  |  |  |

WarrantyPolicy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | WarrantyPolicyID | int |  | x |  | x | PK |
| 2 | warrantyPeriod | tinyint |  |  |  |  |  |
| 3 | WarrantyCategory | NVARCHAR | 255 |  |  |  |  |

User:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | userID | int |  | x |  | x | PK |
| 2 | username | nvarchar | 255 | x |  |  |  |
| 3 | password | nvarchar | 255 |  | x |  |  |
| 4 | firstName | nvarchar | 255 |  | x |  |  |
| 5 | lastName | nvarchar | 255 |  | x |  |  |
| 6 | email | vachar | 255 | x |  |  |  |
| 7 | phoneNumber | vachar | 15 | x |  |  |  |
| 8 | Address | nvachar | 255 |  |  |  |  |
| 9 | [status] | tinyint |  |  |  |  |  |
| 10 | discount | float |  |  |  |  |  |

Employee:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | EmployeeID | int |  |  |  | x | PK |
| 2 | Employeename | nvarchar | 255 | x | x |  |  |
| 3 | firstName | nvarchar | 255 |  | x |  |  |
| 4 | lastName | nvarchar | 255 |  | x |  |  |
| 5 | email | varchar | 255 |  |  |  |  |
| 6 | phoneNumber | varchar | 15 | x | x |  |  |
| 7 | address | nvarchar | 255 |  |  |  |  |
| 9 | role | int |  |  |  |  |  |

Order:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | OrderId | int |  |  |  | x | PK |
| 2 | userID | int |  |  | x | x | FK |
| 3 | OrderDate | date |  |  |  |  |  |
| 4 | deliveryAddress | nvarchar | 255 |  |  |  |  |
| 5 | status | int |  |  |  |  |  |
| 6 | totalMoney | money |  |  |  |  |  |

WarrantyInvoice:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | WarrantyInvoiceID | int |  |  |  | x | PK |
| 2 | productID | int |  |  | x |  |  |
| 3 | orderID | int |  |  | x |  |  |
| 4 | EmployeeID | int |  |  | x |  |  |
| 5 | maintenanceDate | date |  |  |  |  |  |
| 6 | appointmentDate | date |  |  |  |  |  |
| 7 | [status] | int |  |  |  |  |  |
| 8 | describe | ntext |  |  |  |  |  |
| 9 | comment | ntext |  |  |  |  |  |

OrderDetails:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 2 | OrderDetailID | int |  |  |  | x | PK |
| 3 | OrderID | int |  |  | X | X | FK |
| 4 | productID | int |  |  | x | x | FK |
| 5 | quantityOrder | int |  |  | x |  |  |
| 6 | price | money |  |  | x |  |  |
|  | contactForWarranty | tinyint |  |  |  |  |  |

News:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field name** | **Type** | **Size** | **Unique** | **Not null** | **PK/ FK** | **Notes** |
| 1 | NewsId | int |  |  |  | x | PK |
| 2 | EmployeeID | int | max |  | x | x | FK |
| 3 | newsTitle | ntext |  |  |  |  |  |
| 4 | tagname | ntext |  |  |  |  |  |
| 5 | newContent | ntext |  |  |  |  |  |
| 6 | [image] | varchar | max |  |  |  |  |
| 7 | DateUpload | date |  |  |  |  |  |
| 8 | isVisible | bit |  |  |  |  |  |
|  | parentNewsID | int |  |  |  |  |  |

### c. Development Tools

#### 3.1.1 HTML

HTML stands for HyperText Markup Language. It is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes.

#### 3.1.2 CSS

[CSS](https://developer.mozilla.org/en-US/docs/Glossary/CSS) (Cascading Style Sheets) allows you to create great-looking web pages, but how does it work under the hood? This article explains what CSS is with a simple syntax example and covers some key terms about the language.

#### 3.1.3 JavaScript

JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.[10] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

**3.1.4 JSP/Servlet**

JSP stands for JavaServer Pages which is a technology for developing dynamic web pages. JSP helps developers to insert java code into HTML pages using special JSP tags. JSP is a type of Java servlet designed to create the user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, JSP actions and commands.

The Java Servlet API allows software developers to add dynamic content to Web servers using the Java platform. It generates content that is usually HTML, sometimes other languages such as XML.

#### 

#### 3.1.5 SQL

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product whose primary function is to store and retrieve data required by other software applications.

**3.1.6 Java 8**

Java 8 is a revolutionary release of the world’s #1 development platform. It includes a huge upgrade to the Java programming model and a coordinated evolution of the JVM, Java language, and libraries. Java 8 includes features for productivity, ease of use, improved polyglot programming, security and improved performance. Welcome to the latest iteration of the largest, open, standards-based, community-driven platform

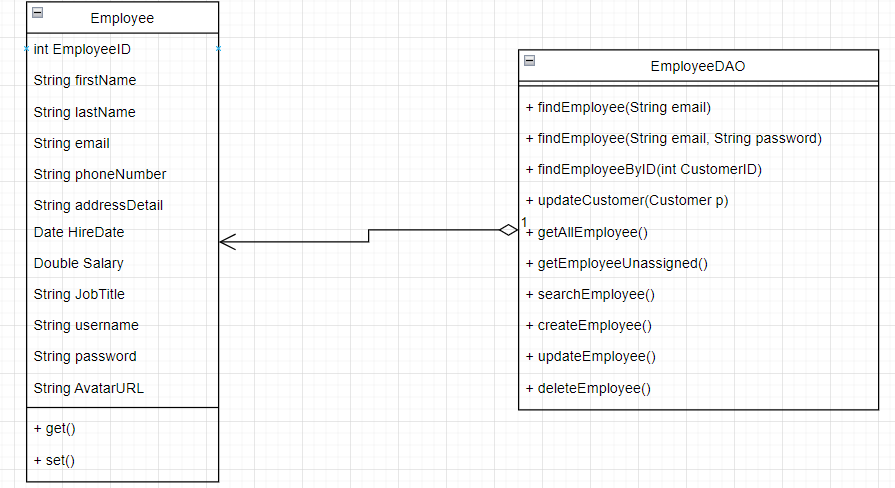
#### 3.1.7 Tomcat 10

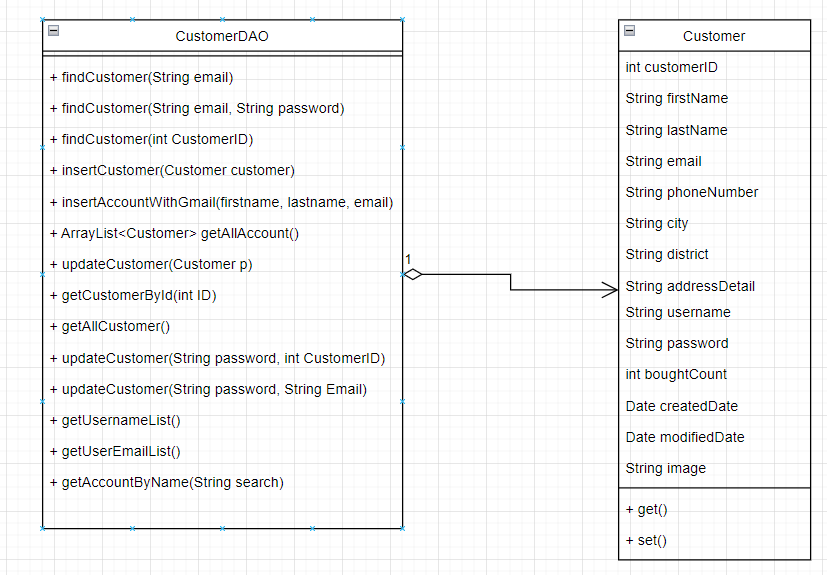
This is the top-level entry point of the documentation bundle for the Apache Tomcat Servlet/JSP container. Apache Tomcat version 10.0 implements the Servlet 5.0 and JavaServer Pages 3.0 specifications from Jakarta EE, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

# III. Code Designs

## 1. Public Account

### a. Class Diagram





### b. Class Specifications

#### CustomerDAO.java

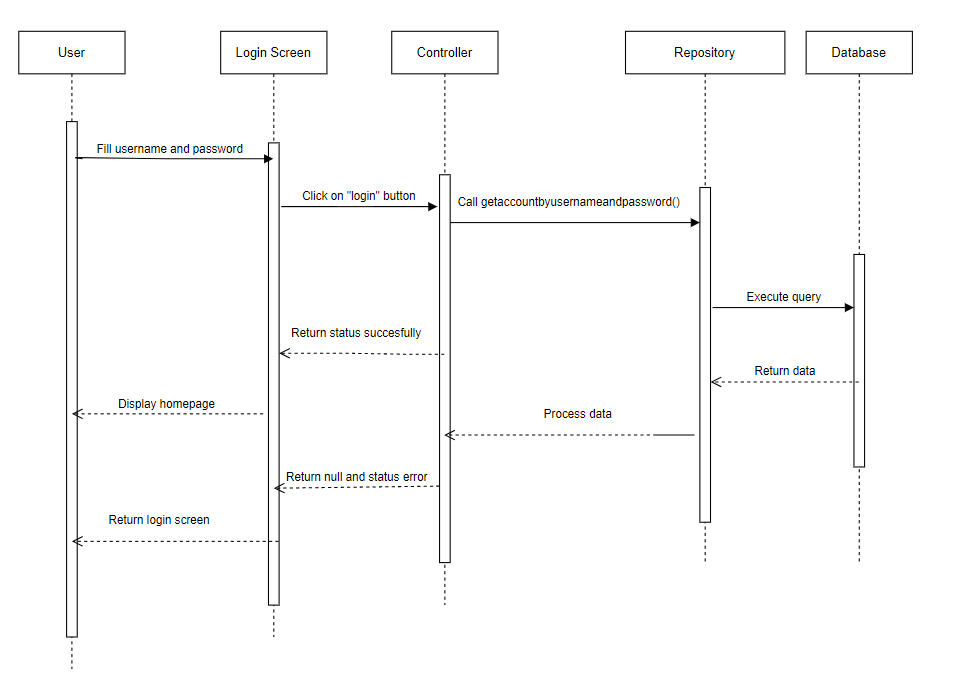
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public boolean checkUserToLogin(String xUsername, String xPassword) | Check info user to login |
| 02 | public String getAnEmail(String xEmail) | Get an email to check in database email exists to send forgot pass |
| 03 | public void insertUser(User u) | Register user |
| 04 | public List<User> getStatisticUser(int status) | get user filter by status |
| 05 | public int getQuantityNewUser() | Get quantity new user to statistic |
| 06 | public int getQuantityUsingUser() | Get quantity using user to statistic |
| 07 | public int getQuantityUsedUser() | Get quantity used user to statistic |
| 08 | public User getAnUser(String xUsername, String xEmail, String xPhoneNumber) | Get an user |
| 09 | public User getUserToAddSession(String xUsername) | Get some information of user to add session |
|  | public void updatePass(String xEmail, String xPass) | Update password |
|  | public void updateUser(User u) | Update information user |

#### EmployeeDAO.java

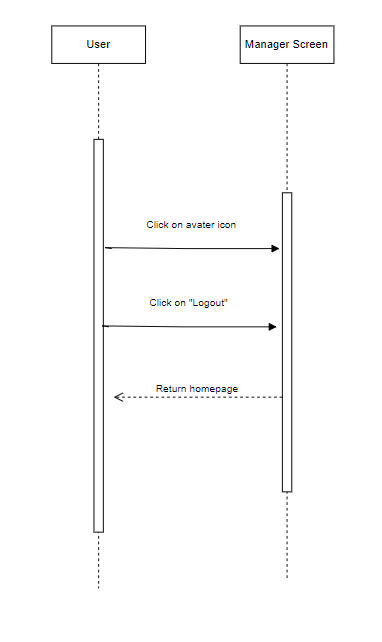
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public Employee findEmployee(String email, String password) | Handle request get Employee by email and password |
| 02 | public Employee findEmployeeByID(int EmployeeID) | Handle request get Employee by EmployeeID |
| 03 | public ArrayList<Employee> getAllEmployee() | Handle request get all Employee |
| 04 | public ArrayList<Employee> getEmployeeUnassigned() | Handle request get all unassigned Employee |
| 05 | public ArrayList<Employee> searchEmployee(String search) | Handle request search Employee |
| 06 | public List<Customer> getAllCustomer(); | Handle request get list customer |
| 07 | public int createEmployee(String FirstName, String LastName, String Email,String PhoneNumber, String Address, Date HireDate, Double Salary, String JobTitle, String username, String password, String AvatarURL) | Handle request add an Employee |
| 08 | public int updateEmployee(String JobTitle, String Email, String username, String password, Double Salary,String Address, String PhoneNumber, String AvatarURL, int EmployeeID) | Handle request update Employee information |
| 09 | public void deleteEmployee(int id) | Handle request delete Employee |

### c. Sequence Diagram(s)

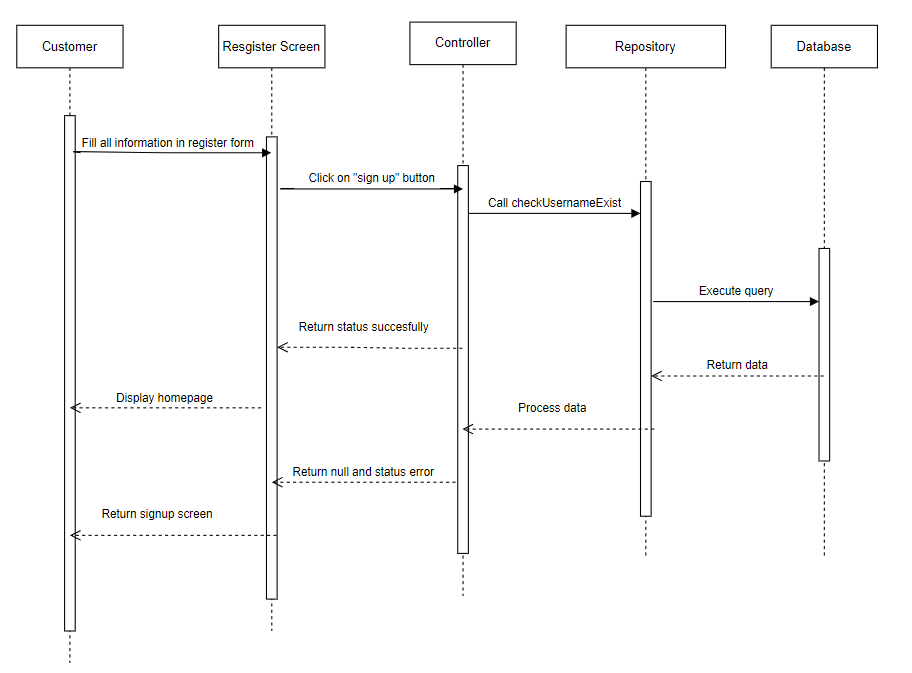
#### Login



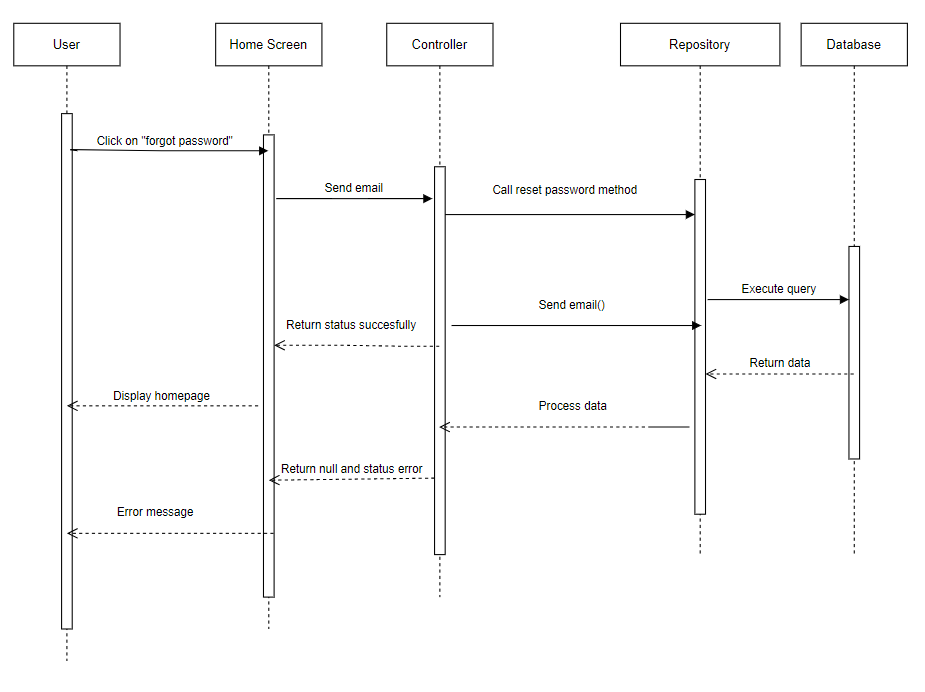
#### Logout



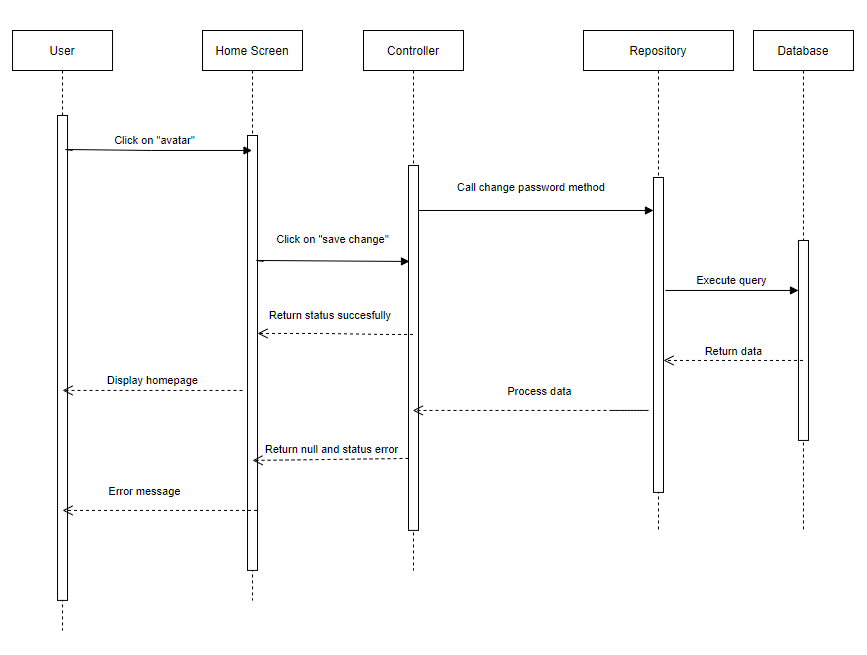
#### Signup



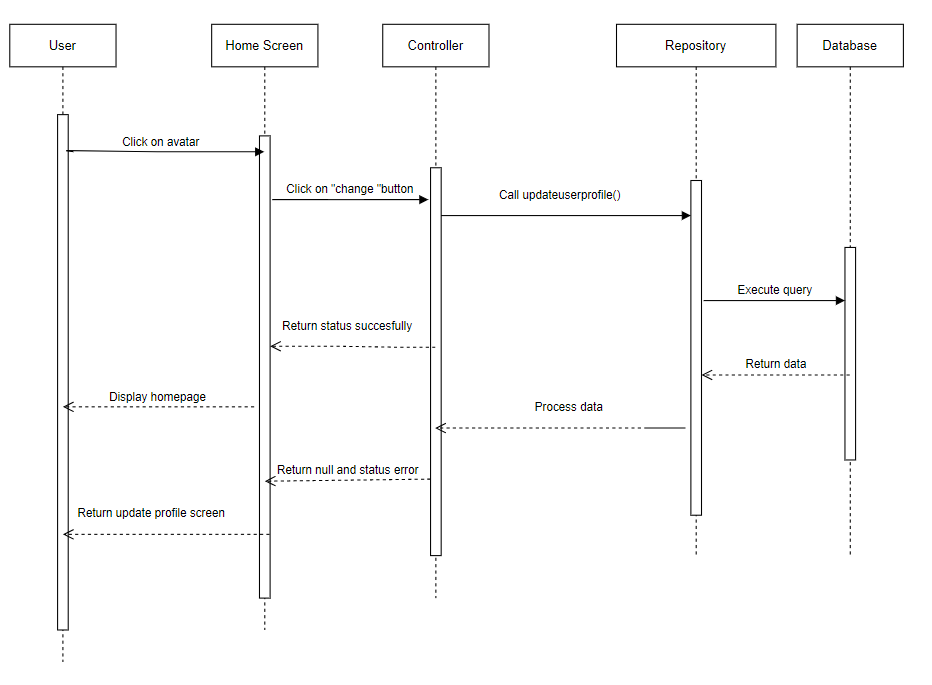
#### Forgot Password



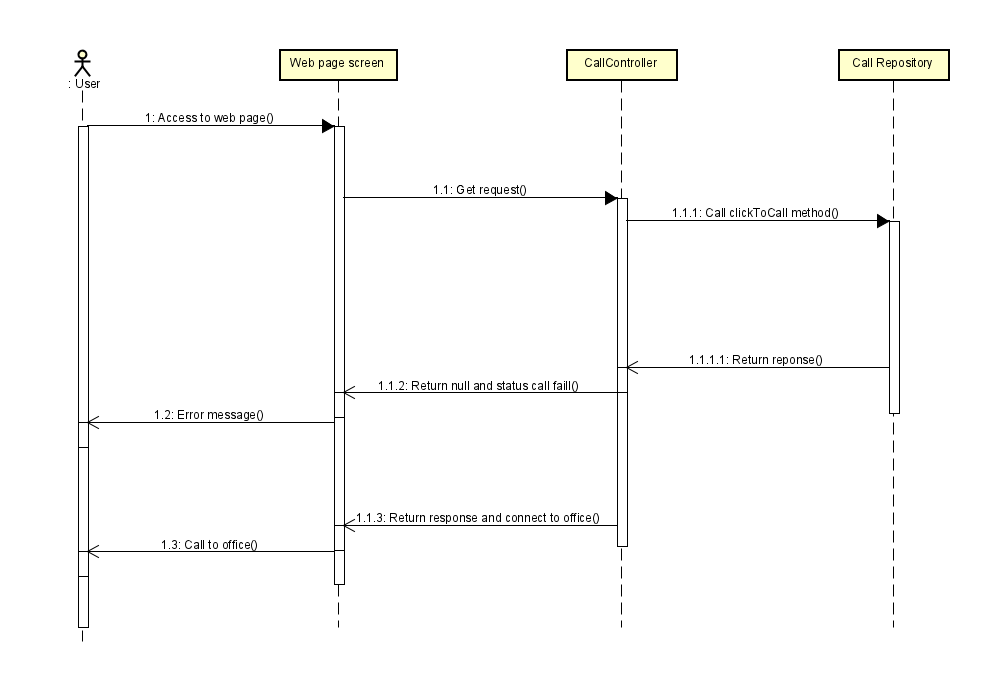
#### Reset Password



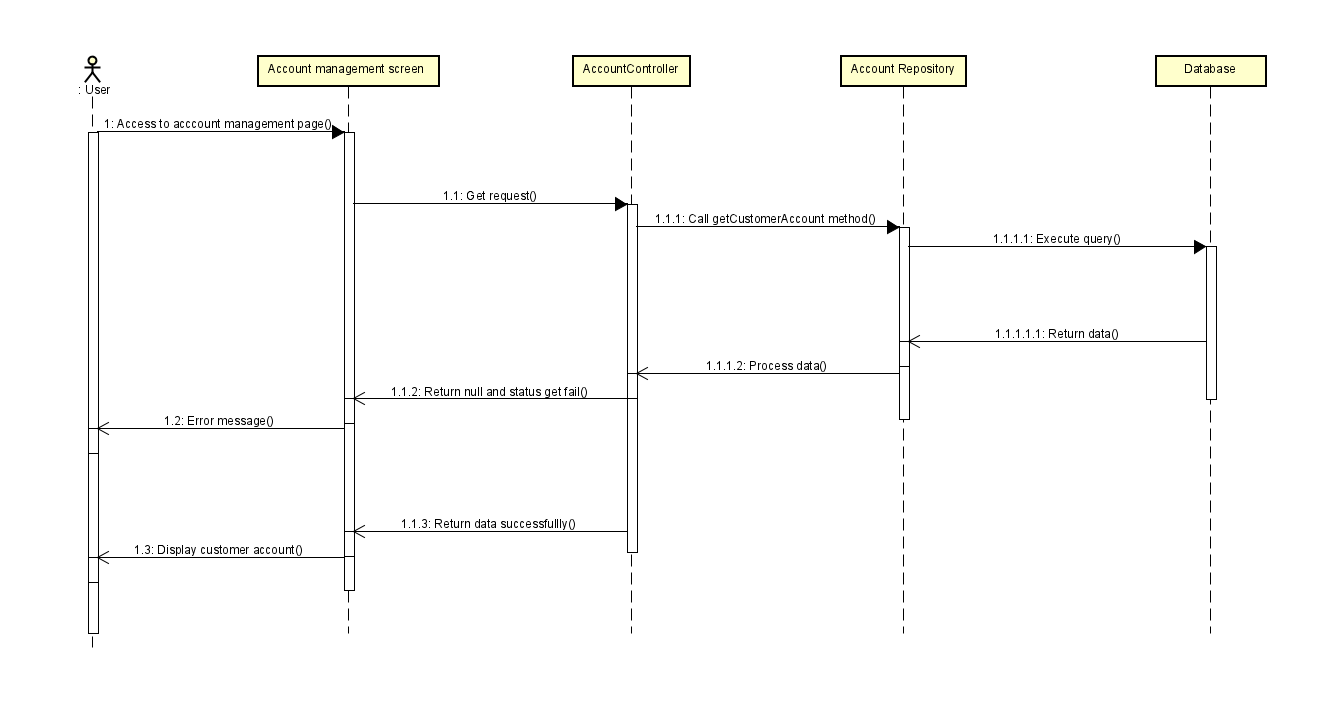
#### Update Profile



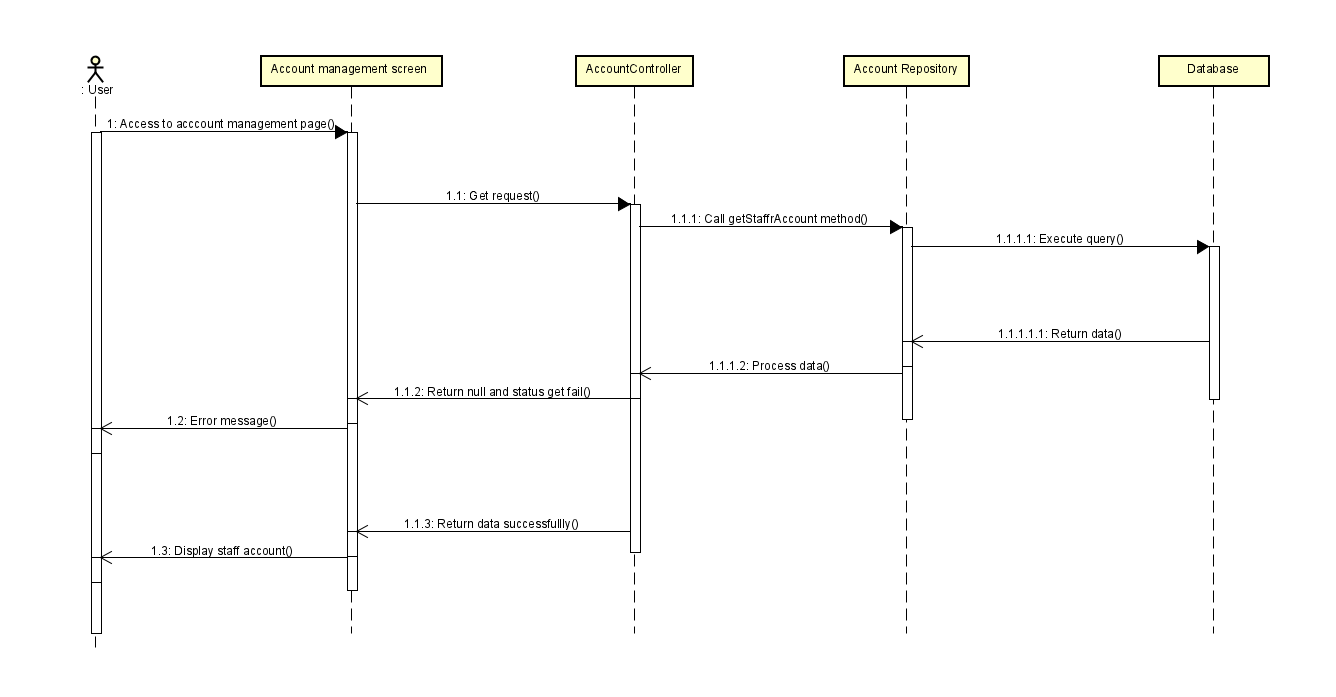
#### Click to call

S

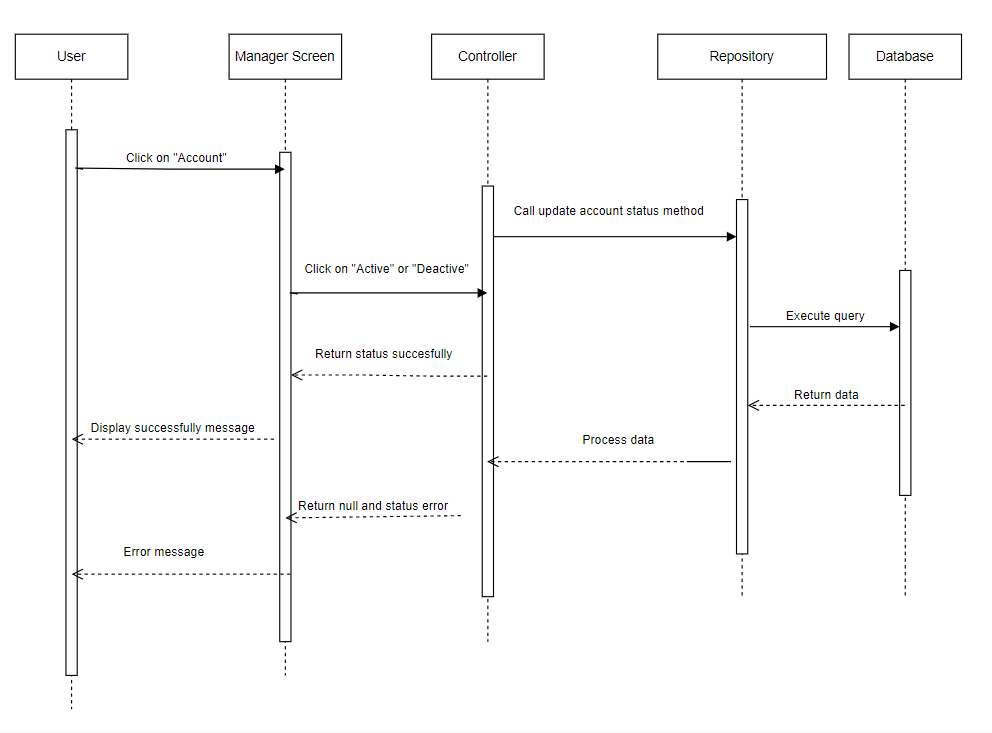
#### View customer account



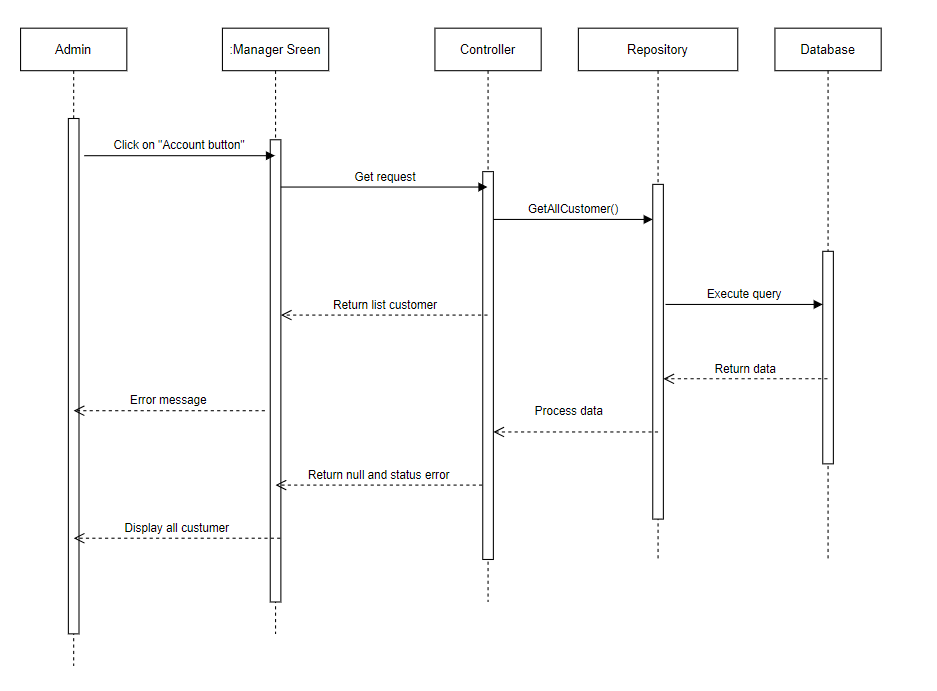
#### View emlpoyeeS account



#### Update account status

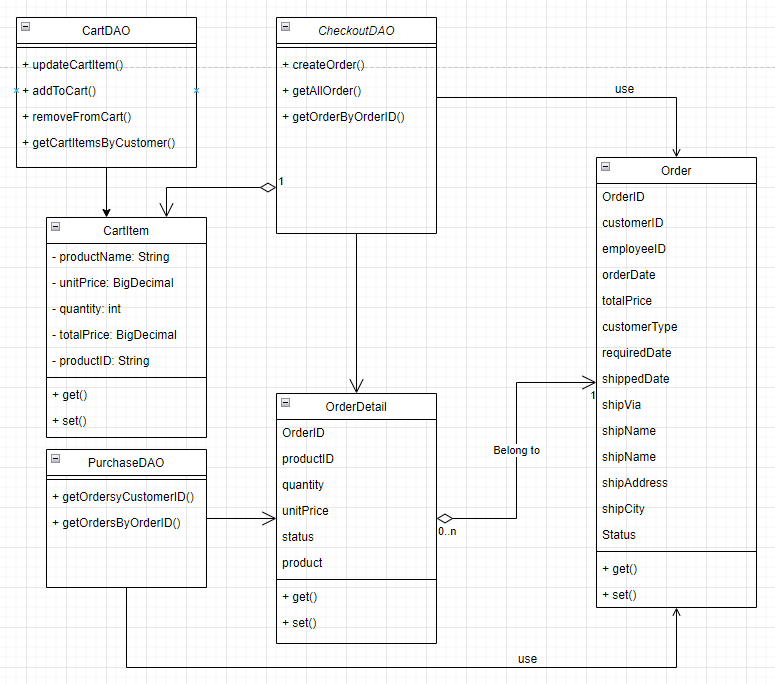


#### View l customer



## 2. Public Order

### a. Class Diagram



### b. Class Specifications

#### OrdersDAO

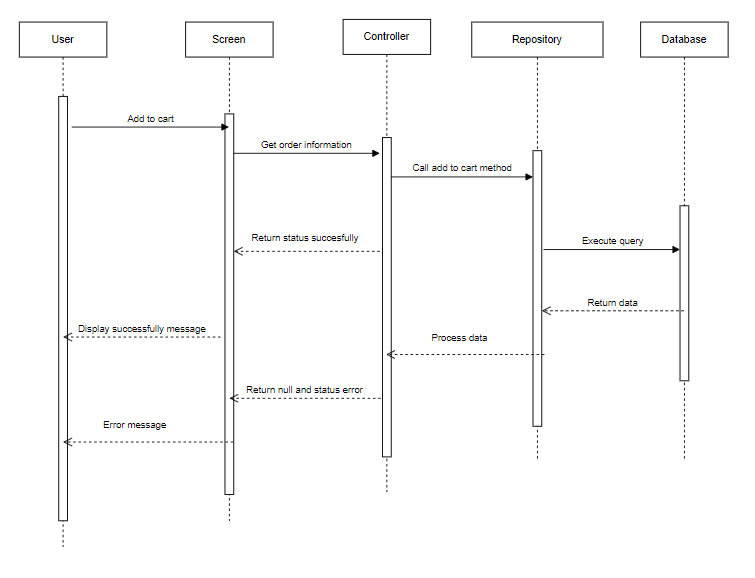
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public ArrayList<Order> getOrderByCustomerID(int customerID) { | Handle request get order by customer ID |

#### CartDAO

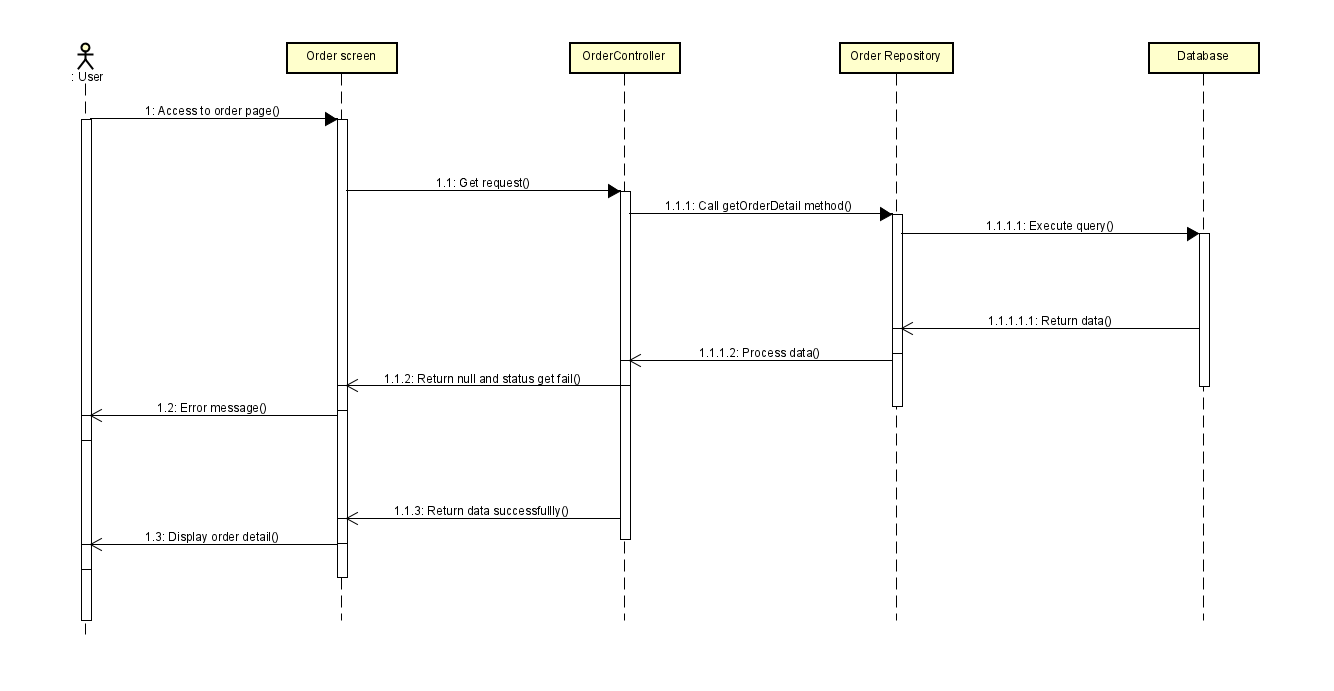
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public void updateCartItem(CartItem item, int customerID) | Handle request update cart item |
| 02 | public void addToCart(CartItem item, int customerID) | Handle request add to cart |
| 03 | public void removeFromCart(String productID, int customerID) | Handle request remove from cart |
| 04 | public List<CartItem> getCartItemsByCustomer(String customerID) | Handle request get cart item by customer |

### c. Sequence Diagram(s)

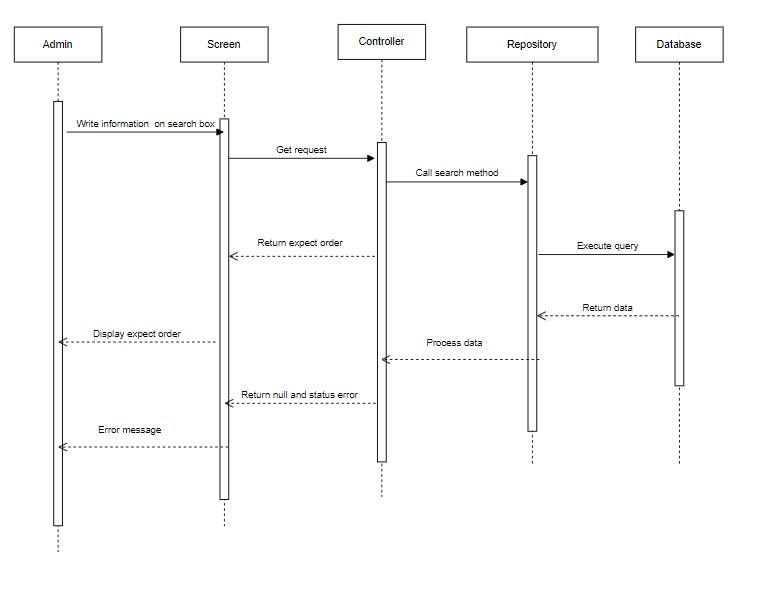
#### Add to cart



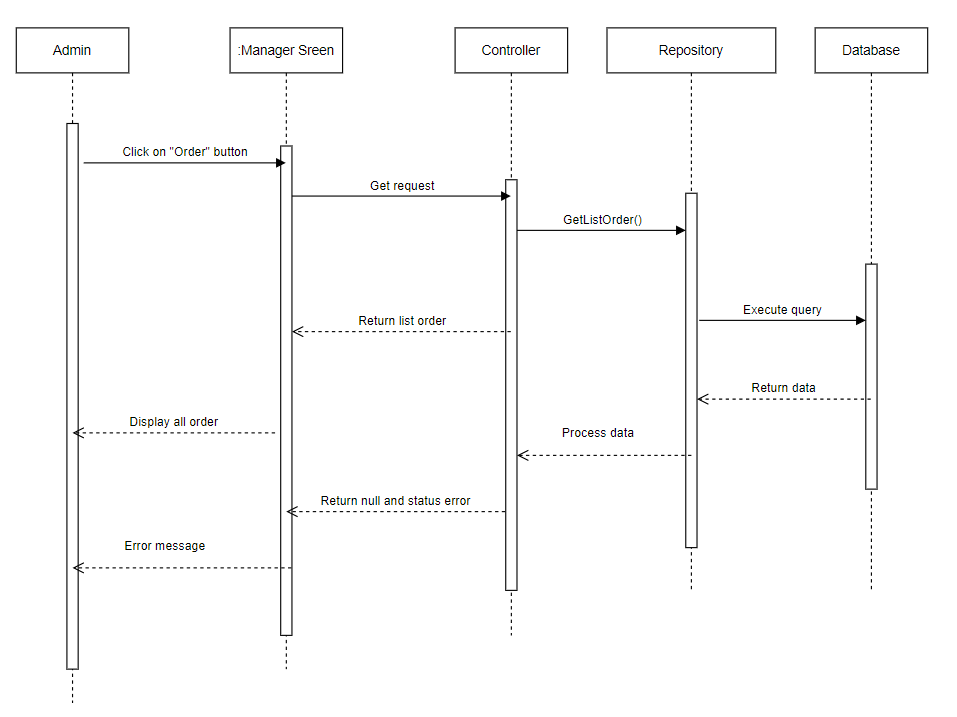
#### View order detail



#### Search Order

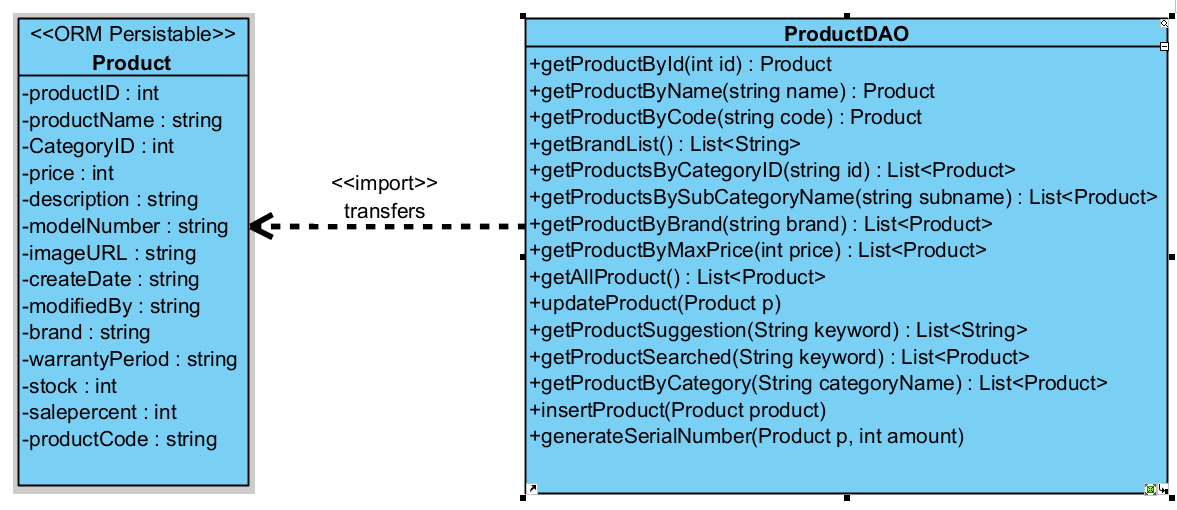


#### View all order



## 3. Public Product

### a. Class Diagram



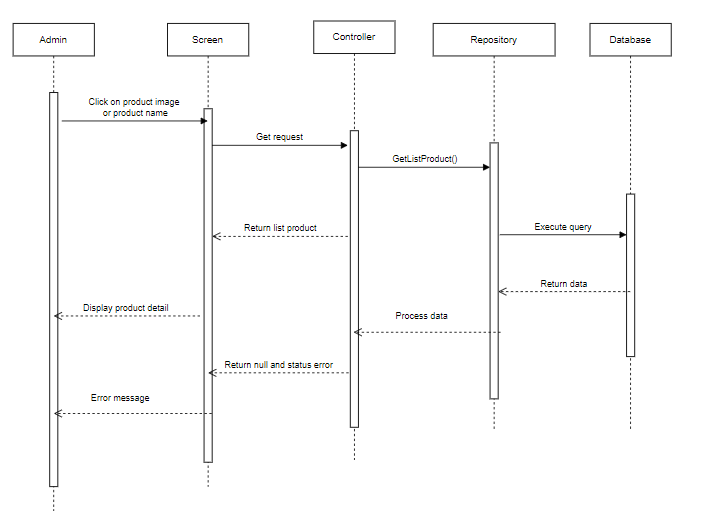
### b. Class Specifications

#### ProductDAO Class

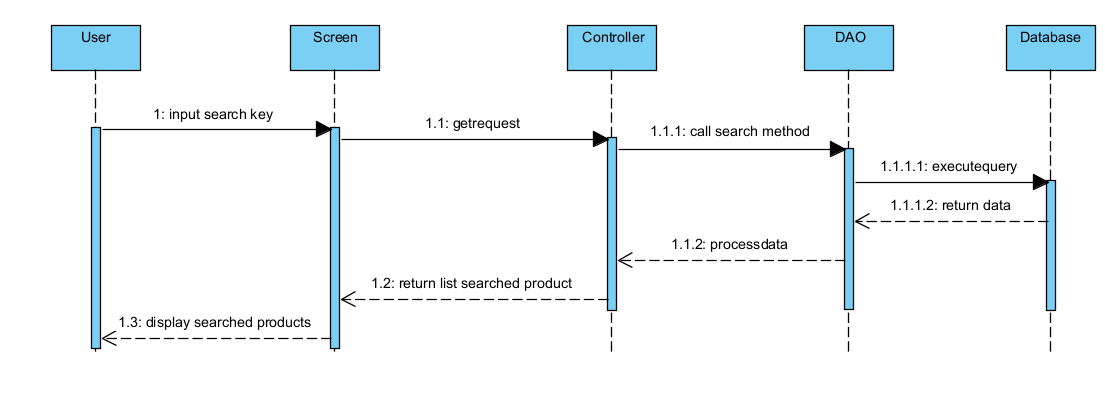
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public Product getProductById(int productId) | Handle request get product by productID |
| 02 | public Product getProductByName(String productName) | Handle request to get product by name |
| 03 | public Product getProductByCode(String productCode) | Handle request get product by its url code |
| 04 | public List<String> getBrandList() | Handle request get all brands |
| 05 | public List<Product> getProductsByCategoryID(int categoryID) | Handle request get products by CategoryId |
| 06 | public List<Product> getProductsByBrand(String brand) | Handle request get products by Brand |
| 07 | public List<Product> getProductsByPrice(double maxPrice) | Handle request get products by maxPrice |
| 08 | public List<Product> getAllProducts() | Handle request get All products |
| 09 | public void updateProduct(Product p) | Handle request update Product |
| 10 | public List<String> getProductSuggestions(String keyword) | Handle request get list of suggestion for product |
| 11 | public List<Product> getSearch(String keyword) | get searched value |
| 12 | public List<Product> getProductsByCategory(String categoryName) | Handle request get List of product by categoryName |
| 13 | public void insertProduct(Product product) | Insert product |
|  |  |  |

### c. Sequence Diagram(s)

#### View product list



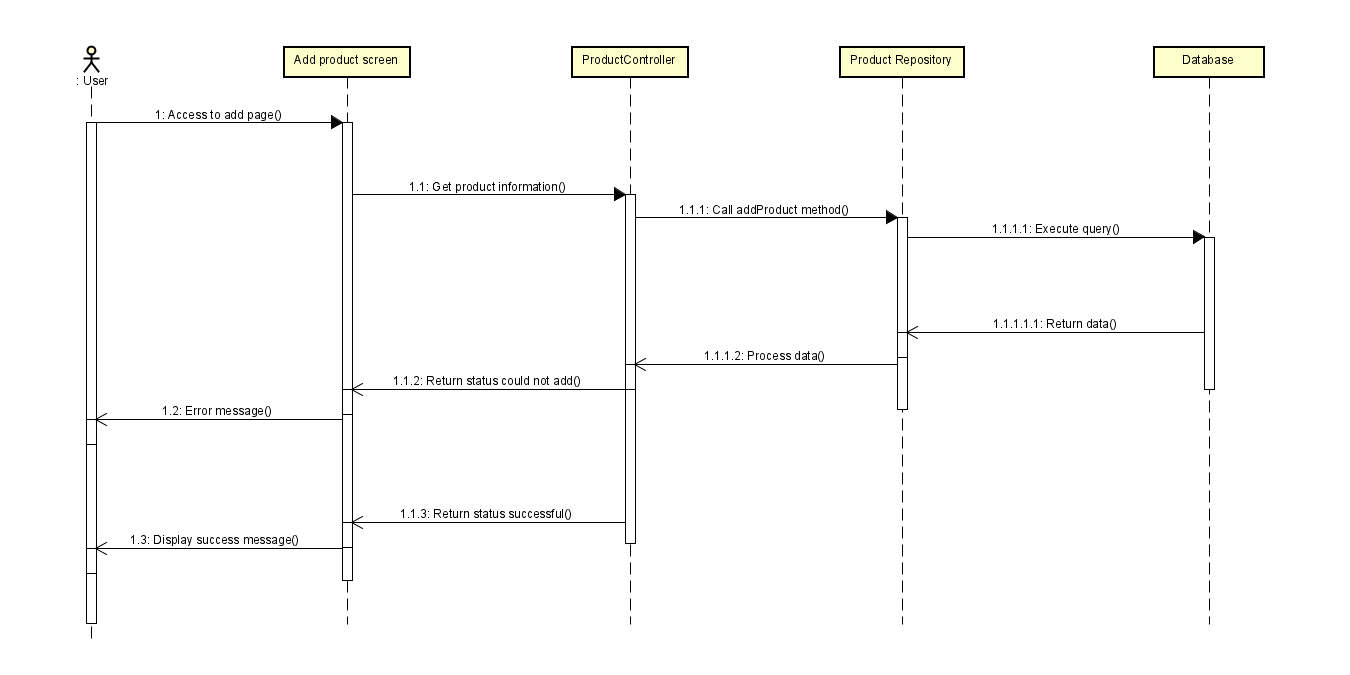
#### Search product



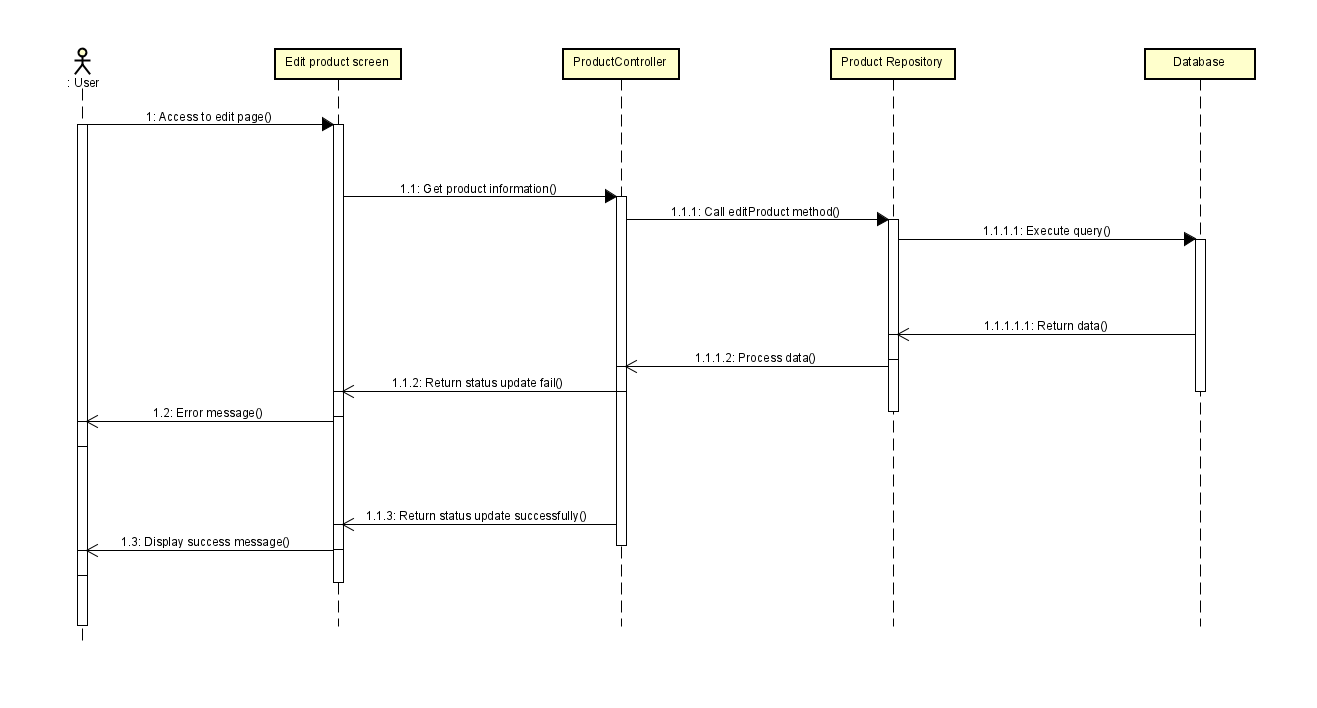
#### View product details

#### 

#### Add Product

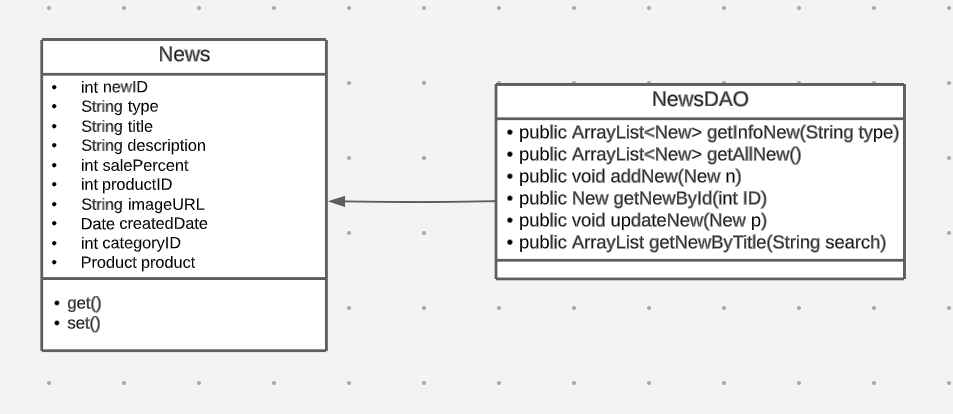


#### Edit product



## 4. Public News

### a. Class Diagram



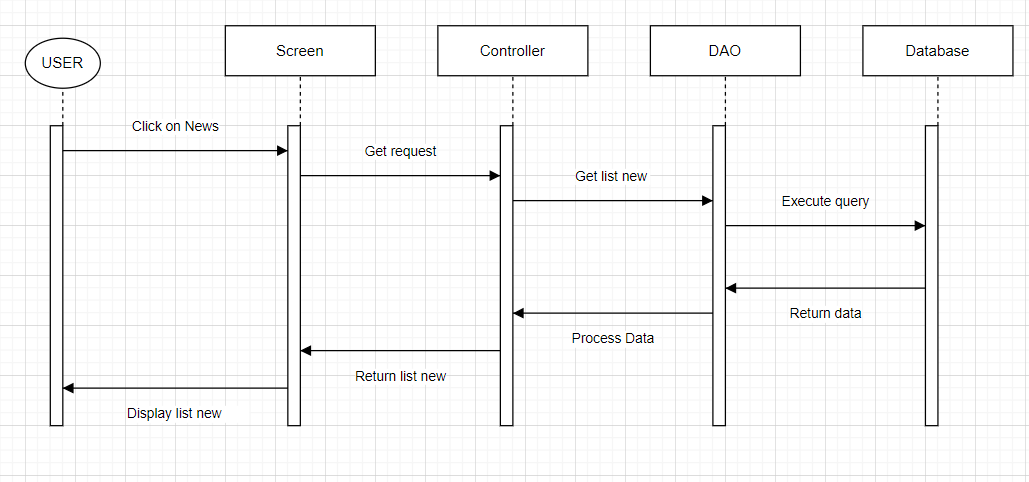
### b. Class Specifications

#### NewsDAO

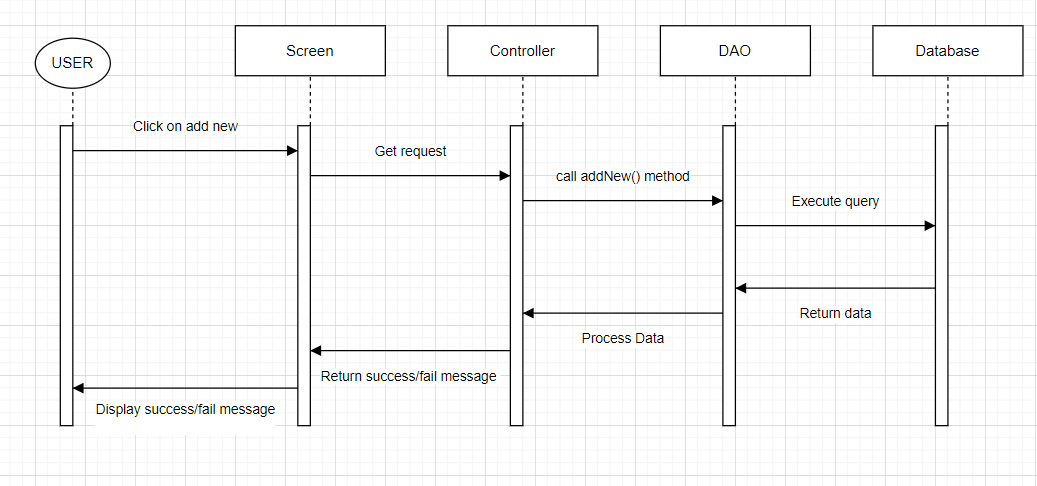
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public ArrayList<New> getInfoNew(String type) | Handle request get new by type |
| 02 | public ArrayList<New> getAllNew() | Handle request get all new |
| 03 | public void addNew(New n) | Handle request add new |
| 04 | public New getNewById(int ID) | Handle request get new by id |
| 05 | public void updateNew(New p) | Handle request update new |
| 06 | public void deleteNew(int idd) | Handle request delete new |
| 07 | public ArrayList getNewByTitle(String search) | Handle request get new by title |

### c. Sequence Diagram(s)

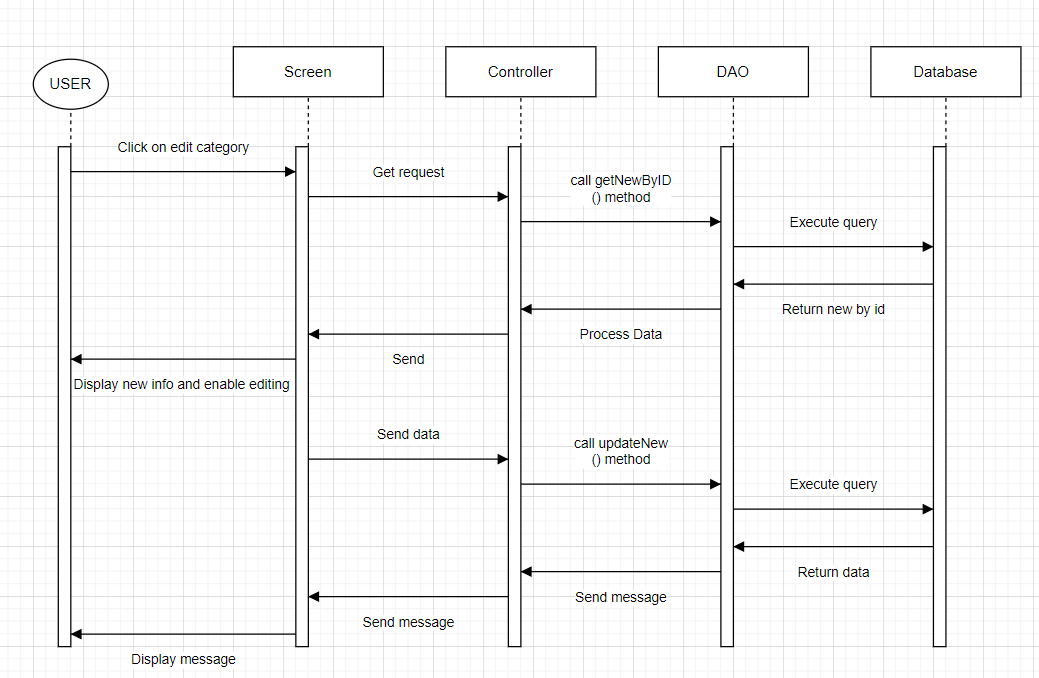
#### View list news



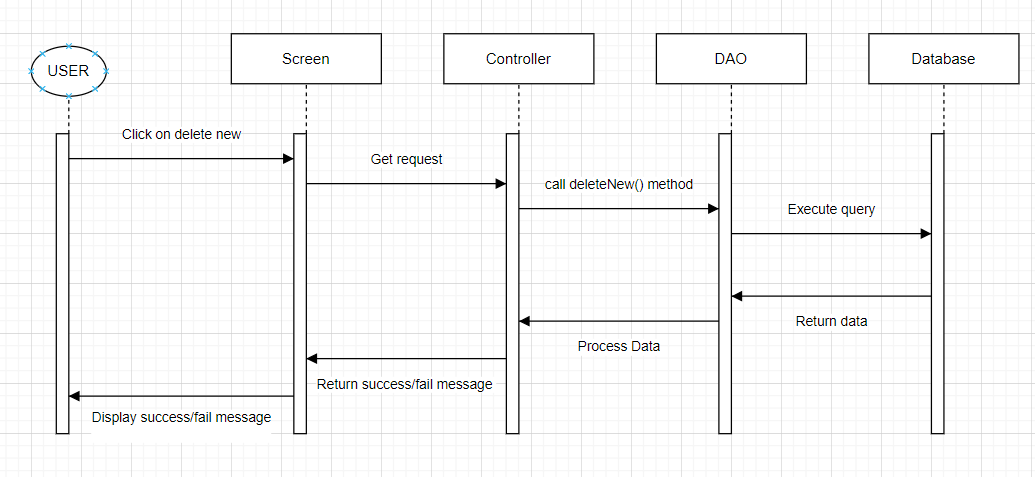
#### Add news



#### Edit news

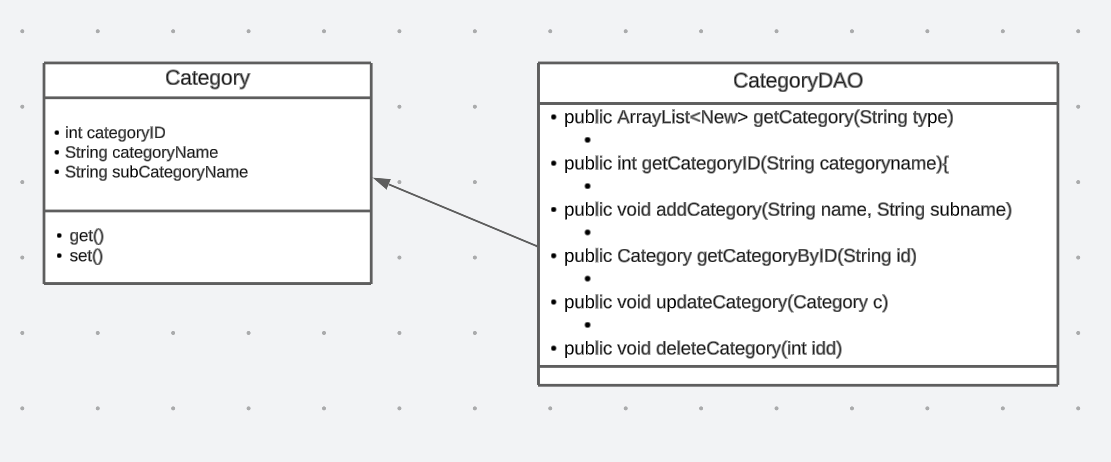


#### Delete news



## 5. Public Category

### a. Class Diagram



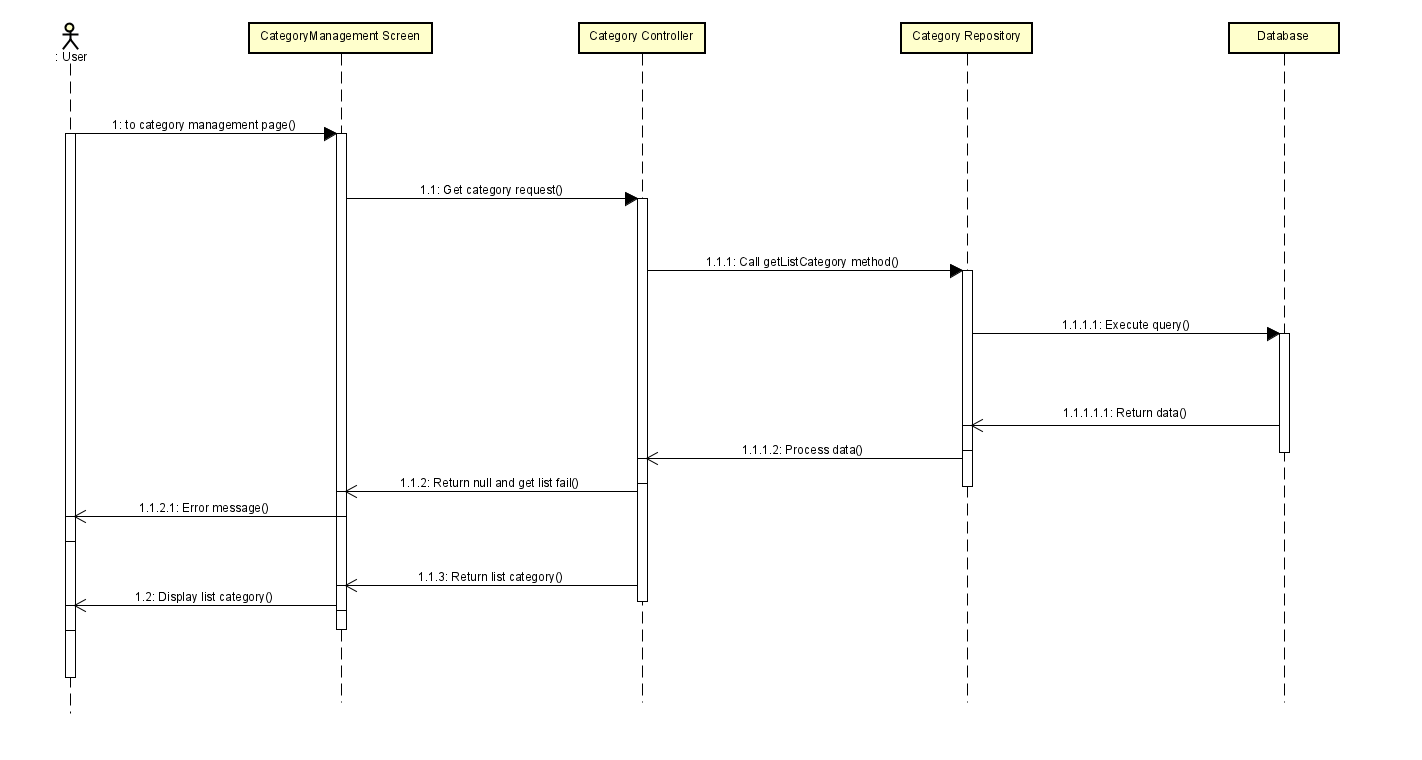
### b. Class Specifications

#### CategoryDAO

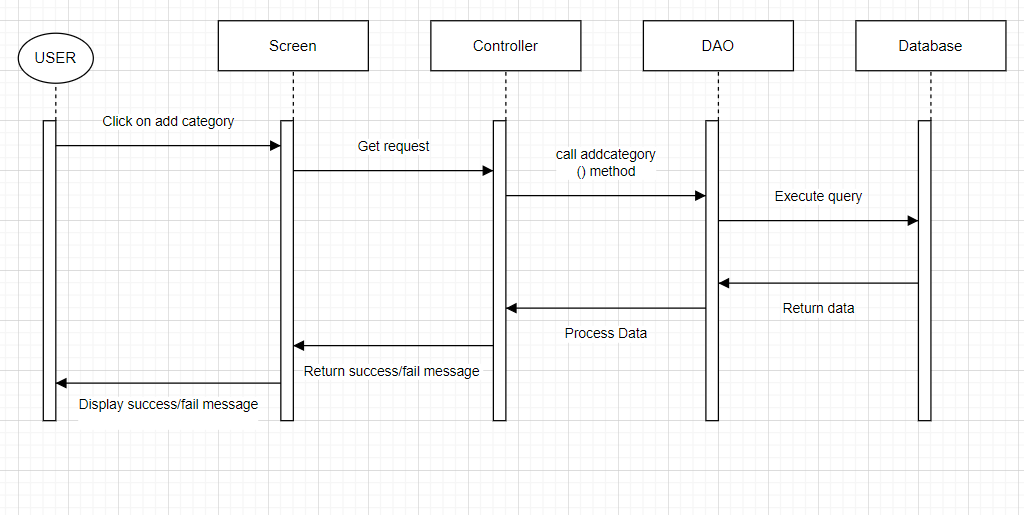
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public ArrayList<New> getCategory(String type) | Handle request get list category |
| 02 | public int getCategoryID(String categoryname){ | Handle request get cate id by subcate name |
| 03 | public void addCategory(String name, String subname) | Handle request add category |
| 04 | public Category getCategoryByID(String id) | Handle request get category by id |
| 05 | public void updateCategory(Category c) | Handle request update category |
| 06 | public void deleteCategory(int idd) | Handle request delete category |

### c. Sequence Diagram(s)

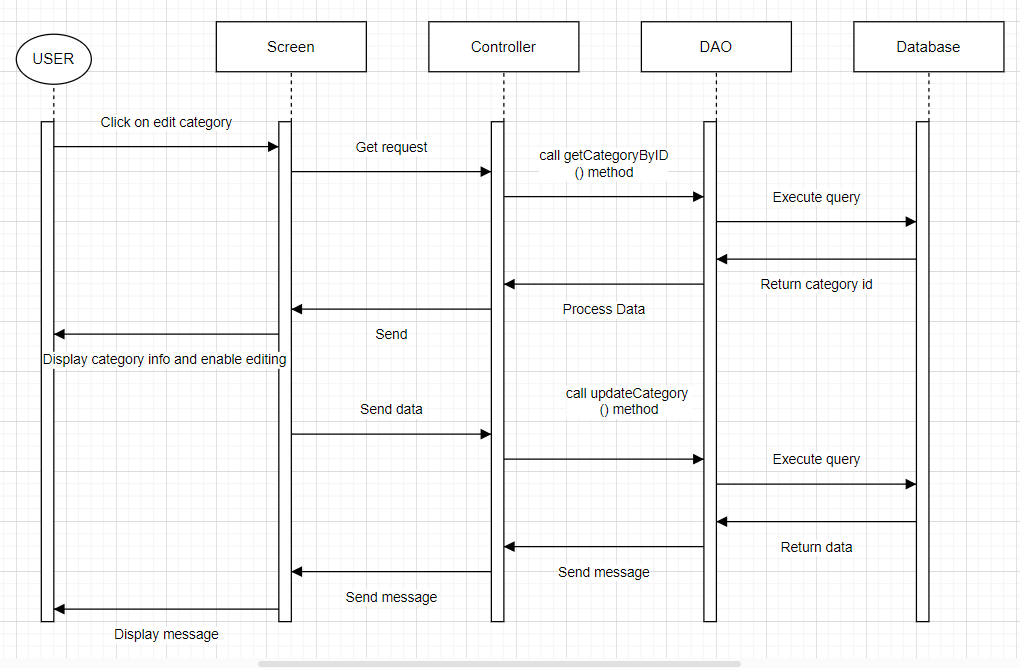
#### View list category

****

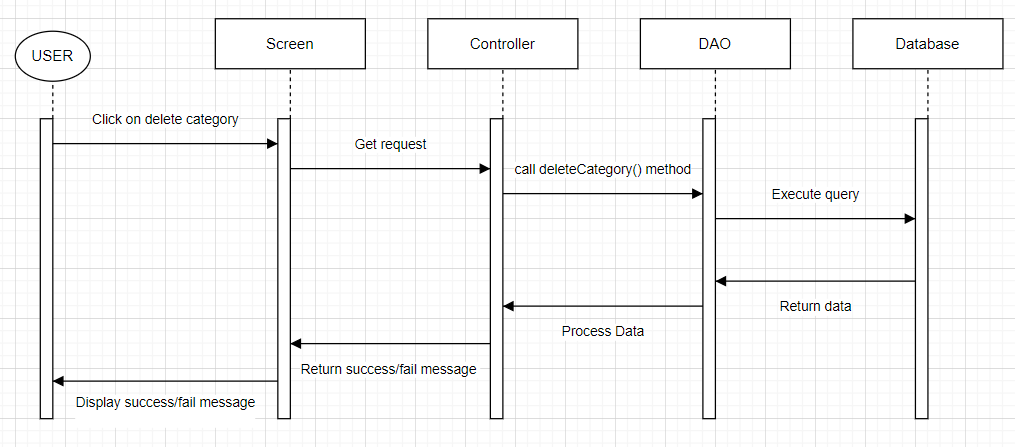
#### Add category

****

#### Edit category

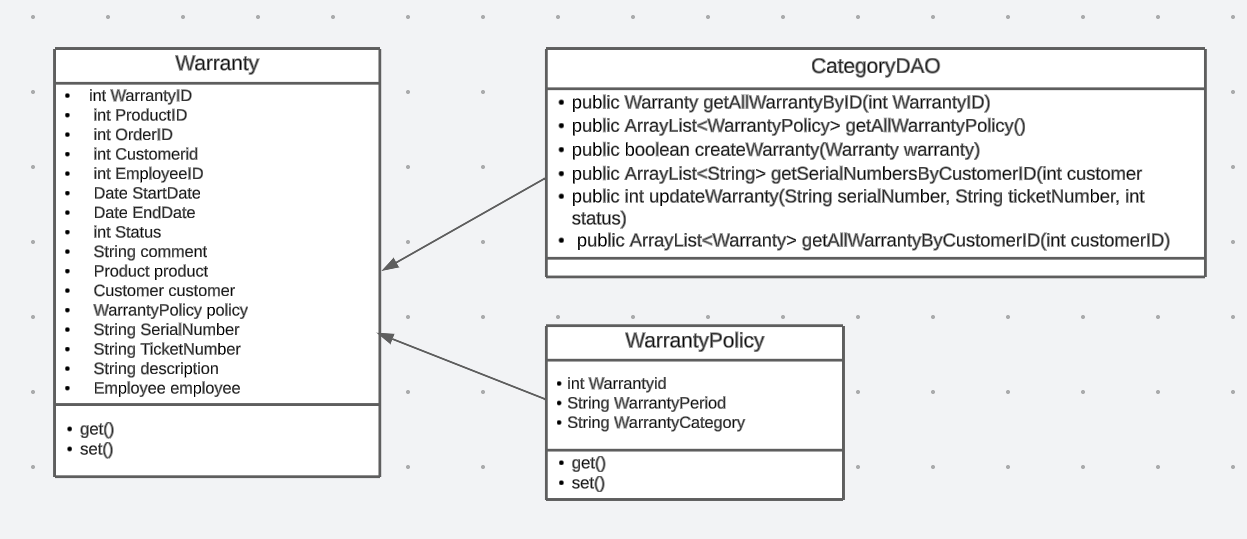
****

#### Delete category



## 6.Public Warranty

### a. Class Diagram



### b. Class Specifications

#### WarrantyDAO.java

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | public Warranty getAllWarrantyByID(int WarrantyID) | get all warranty by warrantyID |
| 02 | public ArrayList<WarrantyPolicy> getAllWarrantyPolicy() | get all warranty policies |
| 03 | public boolean createWarranty(Warranty warranty) | Create warranty request |
| 04 | public ArrayList<String> getSerialNumbersByCustomerID(int customerID) | get serialnumber by customerID |
| 05 | public int updateWarranty(String serialNumber, String ticketNumber, int status) | Handle request update warranty status based on serial number and ticket number |
| 06 | public ArrayList<Warranty> getAllWarrantyByCustomerID(int customerID) | Get all warranty by customerID |

### c. Sequence Diagram(s)

#### Request warranty



#### Tracking warranty request

