

**<<BAKERY SHOP>>**

**Software Design Document**

– Da Nang, Oct 2022 –

Table of Contents

[I. Software Design Document 2](#_Toc117576745)

[1. System Design 2](#_Toc117576746)

[1.1 System Architecture 2](#_Toc117576747)

[1.2 Package Diagram 3](#_Toc117576748)

[2. Database design 5](#_Toc117576749)

[3. Detail design 7](#_Toc117576750)

[3.1. Class Diagram 7](#_Toc117576751)

[3.2. Class Specification 7](#_Toc117576752)

[4 Sequence Diagram 12](#_Toc117576753)

[4.1 Register 12](#_Toc117576754)

[4.2 Login 12](#_Toc117576755)

[4.3 Search 13](#_Toc117576756)

[4.4 Update Product 13](#_Toc117576757)

[4.5 Update Post 14](#_Toc117576758)

[4.6 View Profile 15](#_Toc117576759)

[4.7 Cancel order 15](#_Toc117576760)

[4.8 Create an Order 15](#_Toc117576761)

[4.9 Update an Order 16](#_Toc117576762)

[4.10 Get a List of Product 17](#_Toc117576763)

[4.11 Get Product Detail 18](#_Toc117576764)

[4.12 Add Product 19](#_Toc117576765)

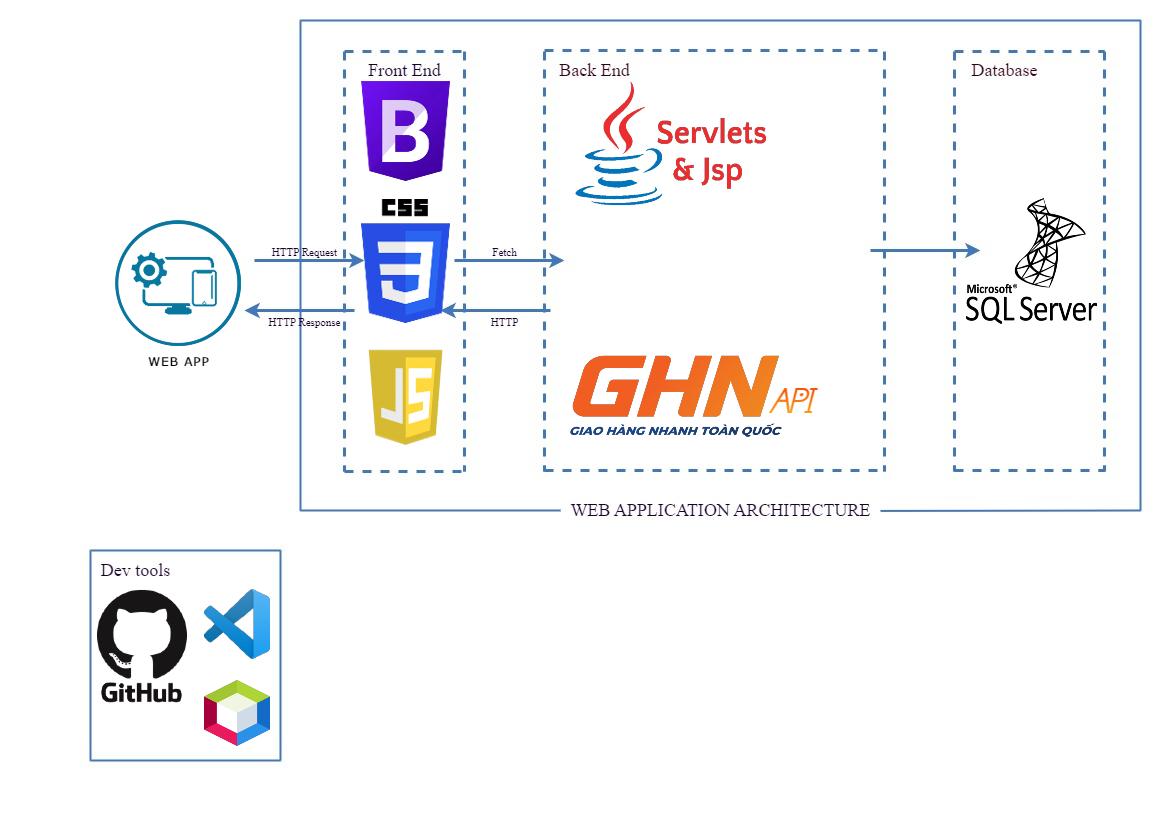
[4.13 Get Post List 20](#_Toc117576766)

[4.14 Get Post Detail 21](#_Toc117576767)

# I. Software Design Document

## 1. System Design

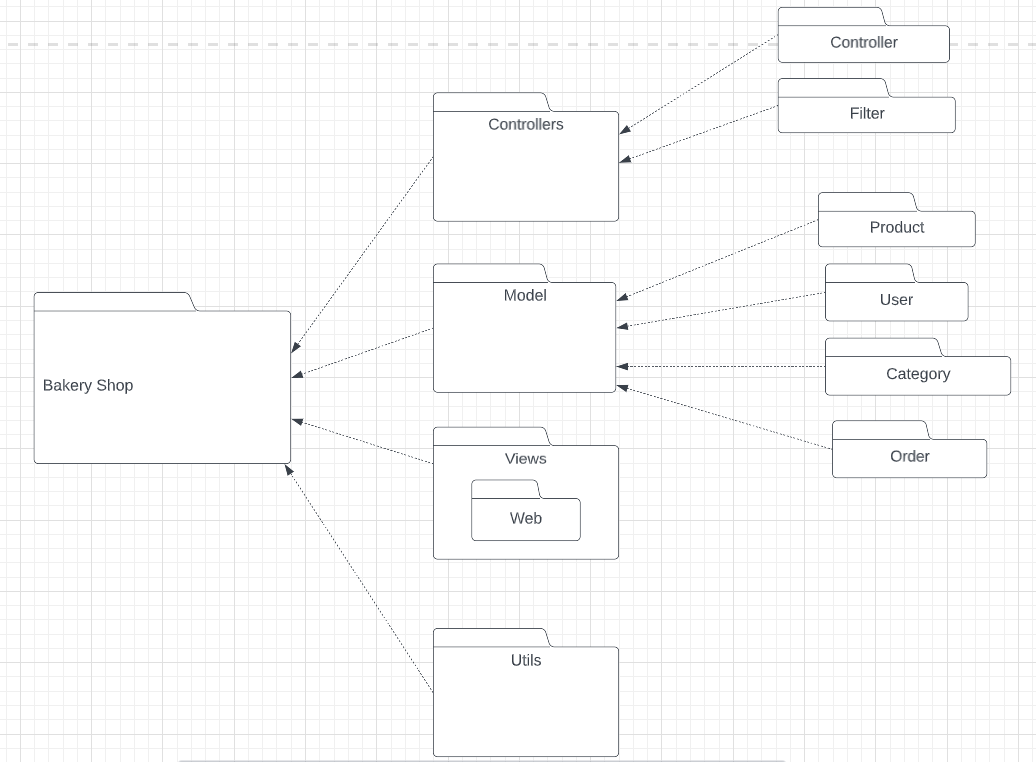
### 1.1 System Architecture



***System architecture component description***

|  |  |  |
| --- | --- | --- |
| No | Component | Description |
| 01 | Web app | This is BakeryShop web app |
| 02 | CSS | This is used to style and layout web pages |
| 03 | Bootstrap | This is CSS framework |
| 04 | Javascript | This is use for form validation and fetching to API |
| 05 | Servlet & JSP | This is used for extending the capabilities of the Web server |
| 06 | GHN API | This is for taking address of Da Nang by fetching |
| 07 | SQL Server | This is used for managing and storing information |
| 08 | Github | This is one of the deploy-enable app which sets a connection to app service deployment. |
| 09 | VSCode | This is for front-end developer |
| 10 | Netbean | This is for hosting tomcat server and compiler of the source |

### 1.2 Package Diagram

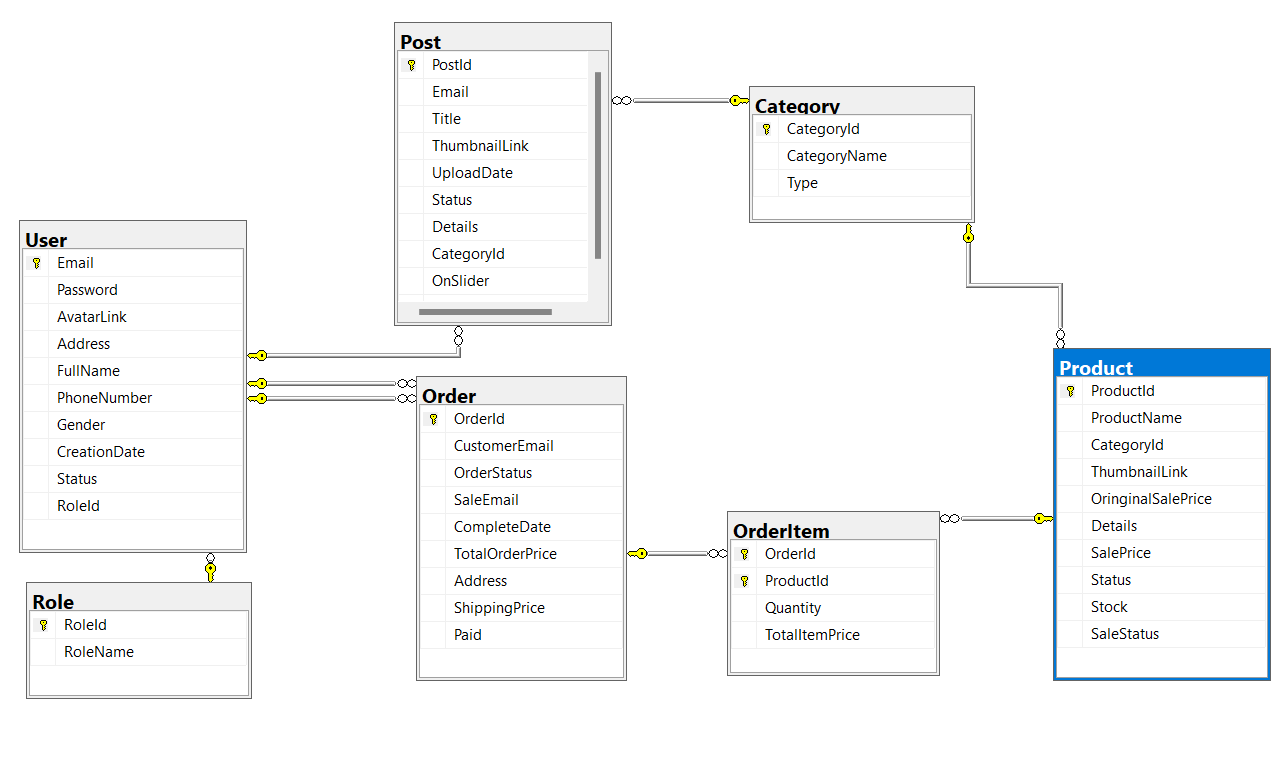


Click [here](https://lucid.app/lucidchart/3d5a0d8d-e2a2-4900-863f-10bc12f8762a/edit?viewport_loc=-388%2C-40%2C2384%2C1090%2C0_0&invitationId=inv_01d9c068-38d9-4aa8-8fd6-ffdc152f6c67) to view full diagram.

***Package descriptions***

|  |  |  |
| --- | --- | --- |
| Number | Package | Description |
| 1 | Controllers | This package contains all controllers for this MVC2 project and authentication or dispatcher filters  Servlet Naming convention: Naming with camelCase and starting with a verb. |
| 2 | Views | This package contains all views (HTML,CSS,JS,JSP file) for this project  HTML/CSS/JS/JSP file naming convention: first letter is a lowercase letter, using camelCase naming |
| 3 | Model | This package contains all products , users class for this project.  Naming convention: first letter is a lowercase letter, the name is using camelCase naming  This package .java files which allow developers to skip writing DAOs for their persistence objects when they usingJDBC. |
| 4 | Utils | (Utils) This package contains DBcontext.java file which is a bridge connection between database and programming. |

## 2. Database design

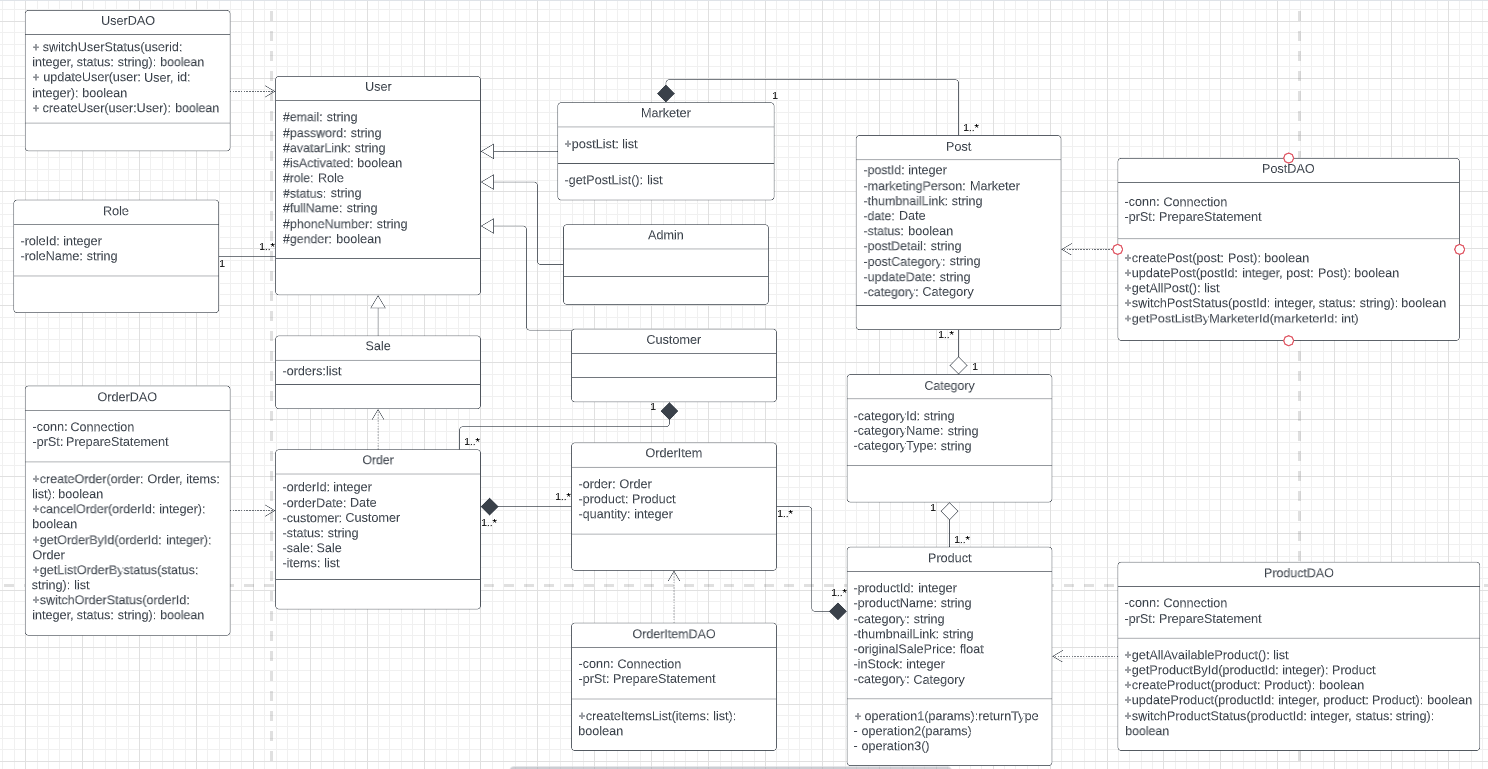


***Table descriptions***

|  |  |  |
| --- | --- | --- |
| **No** | **Table** | **Description** |
| 01 | User | Email: The User’s Email  Password: The User’s account password  AvatarLink: Link to this account’s Avatar  Address: The Customer’s Address  FullName: The Customer’s Full name  PhoneNumber: The Customer’s Phone number  Gender: The Customer’s Gender  CreationDate: Date when this account’s created  Status: The User Account’s status  RoleId: Id of the User’s role  - Primary keys: Email  - Foreign keys: RoleId |
| 02 | Role | RoleId: Role’s Id  Name: Role’s Name  - Primary keys: RoleId |
| 03 | Post | PostId: Post’s Id  Email: Email of the Post’s author  Title: Post’s title  Thumbnail Link: Link to the Post’s Thumbnail  Date: Date when Post is first created.  Status: Post’s status  Detail: Content of the Post  CategoryID: ID of the Post’s Category  UpdateDate: Date of the most recent time Post is updated  - Primary keys: PostId  - Foreign keys: Email, CategoryID |
| 04 | Category | CategoryId: Category’s Id  Name: Category’s Name  Type: Category’s Type  - Primary keys: CategoryId |
| 05 | Product | ProductId: Product’s Id  Name: Product’s Name  CategoryId: Id of the Product’s Category  Thumbnail Link: Link to the Product’s Thumbnail  OriginalSalePrice: The Product’s original price  Details: Product’s information  SalePrice: The Product’s price after apply Sale  Status: Product’s Status  Sale Status: Product is on Sale or not  Stock: The amount of Product still in Stock.  - Primary keys: ProductId  - Foreign keys: CategoryID |
| 06 | Order | OrderId: Order’s Id  Date: Date when Order is created  CustomerEmail: The customer of this Order’s Email  Status: Order’s Status  SaleEmail: Email of Sale in charge of this Order  CompleteDate: Date when Order is completed.  TotalOrderPrice: Price of the Order  Address: Order’s shipping address.  - Primary keys: OrderId  - Foreign keys: CustomerEmail, SaleEmail |
| 07 | OrderItem | OrderItemId: Id of the Order this item belongs to  ProductId: This item’s Id  Quantity: This item’s quantity  TotalOrderItemPrice: Total Price of the Order Item.  - Primary keys: OrderItemId |

## 3. Detail design

### 3.1. Class Diagram



Click [here](https://lucid.app/lucidchart/7ce26ae8-d0e6-4265-a535-65ecc4545932/edit?viewport_loc=-625%2C260%2C2987%2C1366%2C0_0&invitationId=inv_c2e26259-7f13-4b96-8539-86bfe5c3d5e5) to view full diagram

### 3.2. Class Specification

#### 3.2.1 UserDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *switchUserStatus* | *- Input: user Id (integer) , status (String)*  *- Output: User corresponding to user Id will have his/her current status switched to given status*  *- Description: This method  execute query to update a user’s status* |
| *02* | *createUser* | *- Input: user (User instance)*  *- Output: A boolean value determines if a new user is inserted into User table in the database successfully*  *- Description: This method execute query to insert a new user to database* |
| *03* | *updateUser* | *- Input: user (User instance) , id (integer)*  *- Output: A boolean value determines if an existing user is modified successfully*  *- Description: This method execute query to update information of a user having id equal to given id parameter* |

#### 3.2.2 PostDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *createPost* | *- Input: post (Post instance)*  *- Output: a boolean value determine if a post is created successfully*  *- Description: This method execute query to create a new post in the database* |
| *02* | *updatePost* | *- Input: post (Post instance), id (integer)*  *- Output: A boolean value determine if a post associated with the parameter id is updated*  *- Description: This method execute query to update a post having corresponding id* |
| *03* | *getAllPost* | *- Input: None*  *- Output: A list containing all post in the database*  *- Description: This method execute query to get all post exist in the database, store those on the list the return* |

#### 3.2.3 OrderDAO

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *createOrder* | *- Input: customer (Customer instance) , items (list of OrderItem instance)*  *- Output: A boolean value determine if an order is created successfully*  *- Description: This method execute query to insert an order as well as the list of order items corresponding to that order into database* |
| *02* | *cancelOrder* | *- Input: order id (integer)*  *- Output: A boolean value determine if an order is removed from the database*  *- Description: This method execute query to remove an order having similar id from database* |
| *03* | *getOrderById* | *- Input: order id (integer)*  *- Output: An order having similar order id*  *- Description: This method execute query to get an order with corresponding id* |
| *04* | *getListOrderByStatus* | *- Input: status (string)*  *- Output: An order list having similar status*  *- Description: This method execute query to get any order having given status, store as an array then return* |
| *05* | *getAllOrders* | *- Input: None*  *- Output: A list containing all orders  stored in the database*  *- Description: This method execute query to get all existing orders in the database, store as an array then return* |
| *06* | *switchOrderStatus* | *- Input: orderId(integer), status (string)*  *- Output: A boolean value determine if an order’s status is modified successfully*  *- Description: This method execute query to replace the current status of an order having the same id as the orderId parameter by a new status* |

#### 3.2.4 ProductDAO

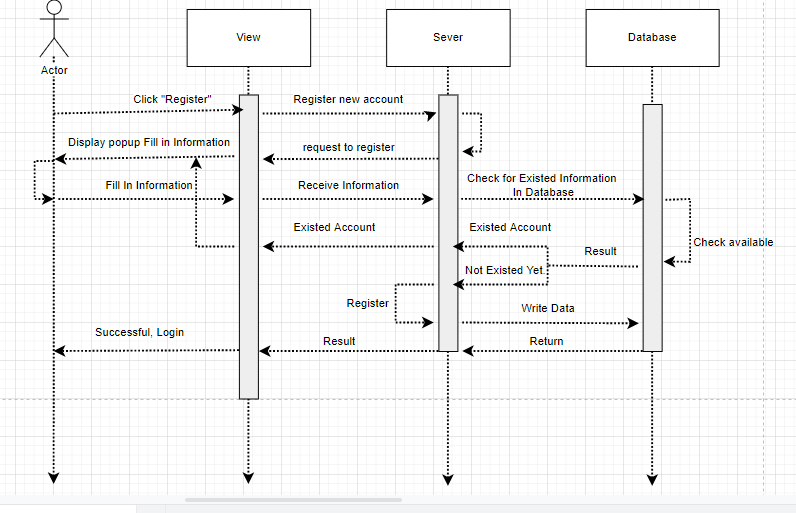
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAvailableProducts* | *- Input: none*  *- Output: The list of available products*  *- Description: This method execute query to get a list of available product, which exist in the database and have positive number for quantity* |
| *02* | *getProductById* | *- Input: productId (integer)*  *- Output: A product object having similar id*  *- Description: This method execute query to get a product according to the id parameter from the database, then return to client* |
| *03* | *createProduct* | *- Input: product (Product instance)*  *- Output: a boolean value determine if a product is created successfully*  *- Description: This method execute query to create a new product in the database* |
| *04* | *updateProduct* | *- Input: product(Product instance), id (integer)*  *- Output: A boolean value determine if a product associated with the parameter id is updated*  *- Description: This method execute query to update a product having corresponding id* |
| *05* | *switchProductStatus* | *- Input: id (integer) , status (string)*  *- Output: A particular product’s status is modified*  *- Description: This method execute query to replace a product’s current status by a new status* |

#### 3.2.5 OrderItemDAO

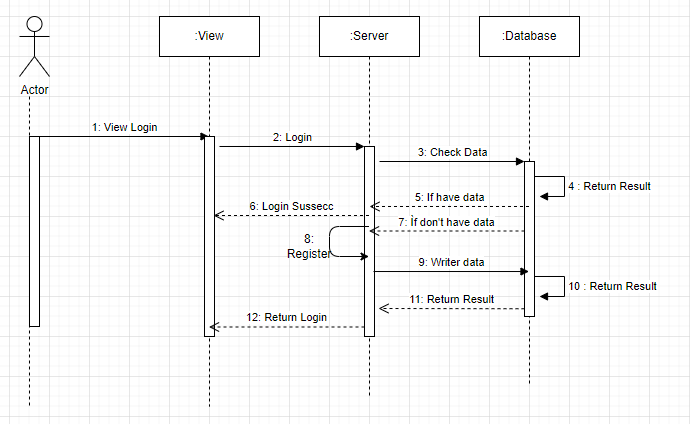
|  |  |  |
| --- | --- | --- |
| ***No*** | ***Method*** | ***Description*** |
| *01* | *createItemsList* | *-Input: List of OrderItem Instance*  *- Output: A boolean value indicate that creation is successful*  *- This method execute query to create a list of order items in the database* |

## 4 Sequence Diagram

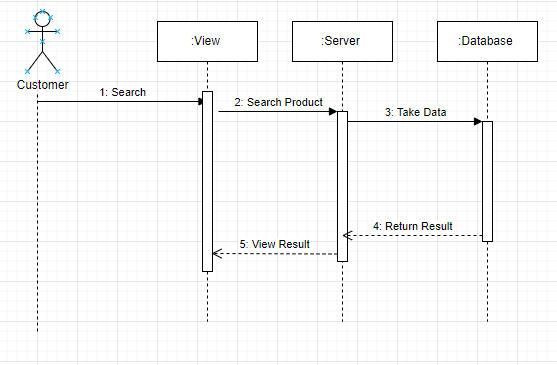
### 4.1 Register



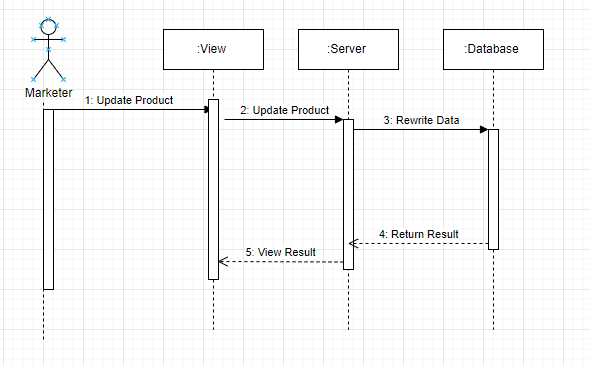
### 4.2 Login

****

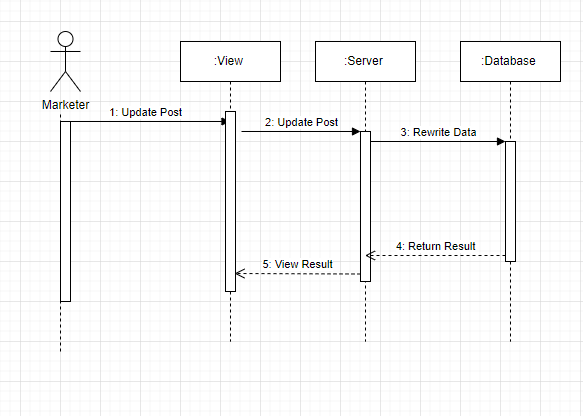
### 4.3 Search



### 4.4 Update Product



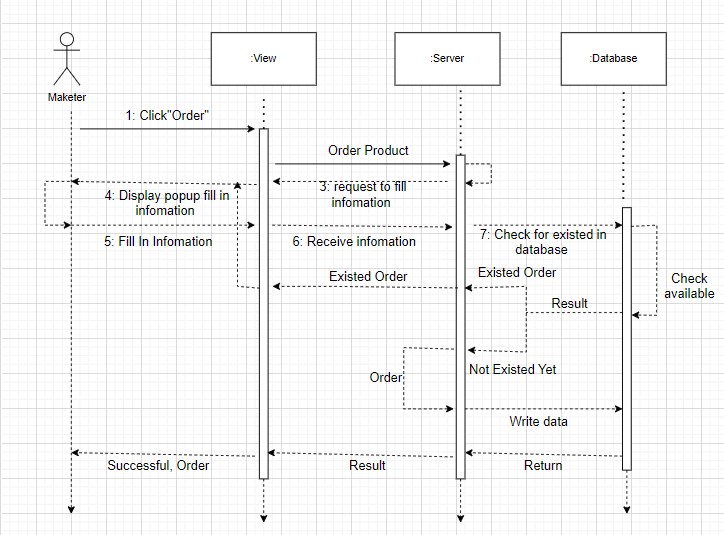
### 4.5 Update Post



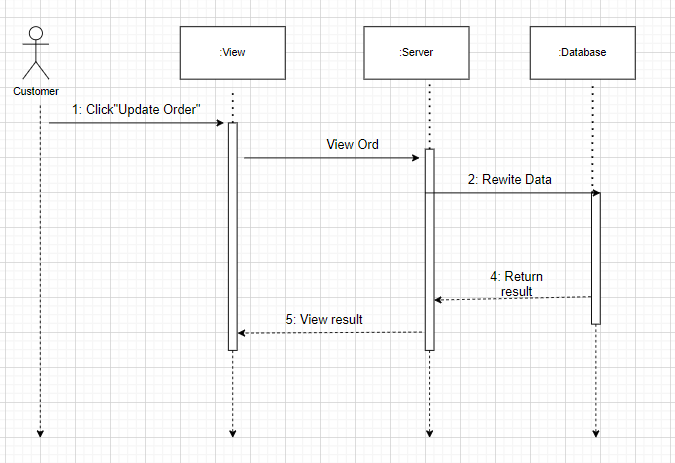
### 4.6 View Profile

### 4.7 Cancel order

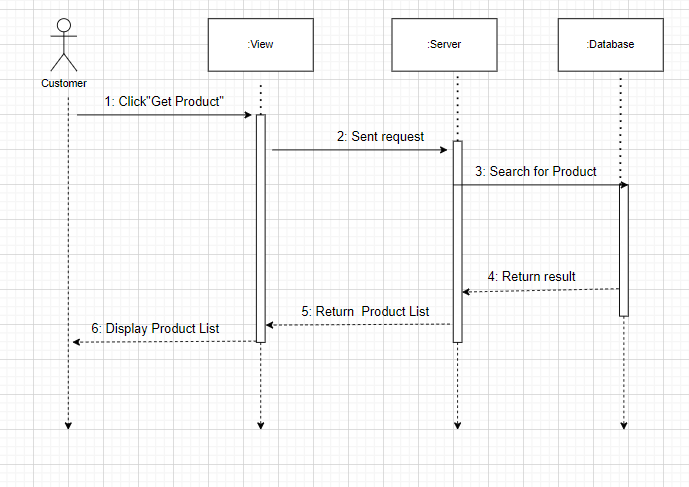
### 4.8 Create an Order



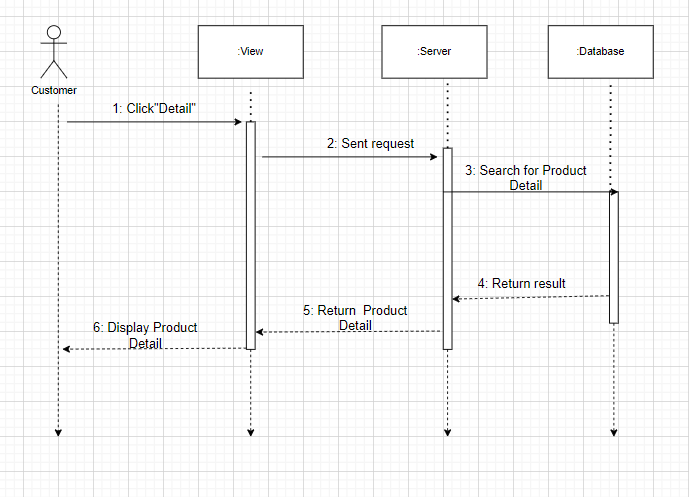
### 4.9 Update an Order



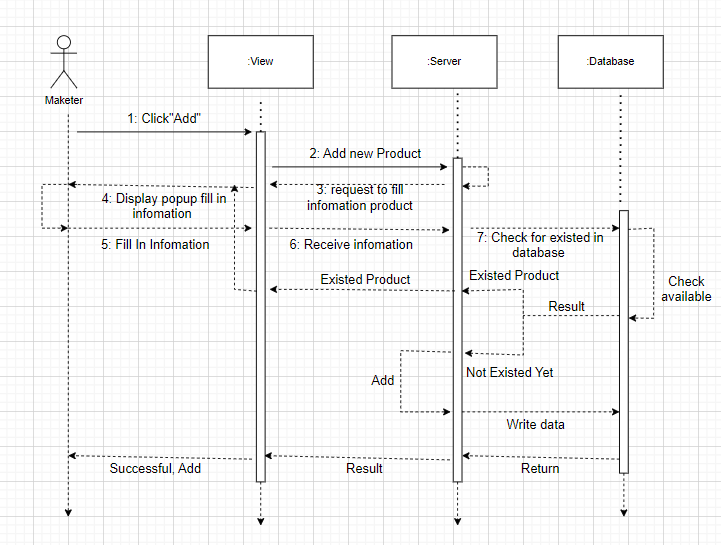
### 4.10 Get a List of Product



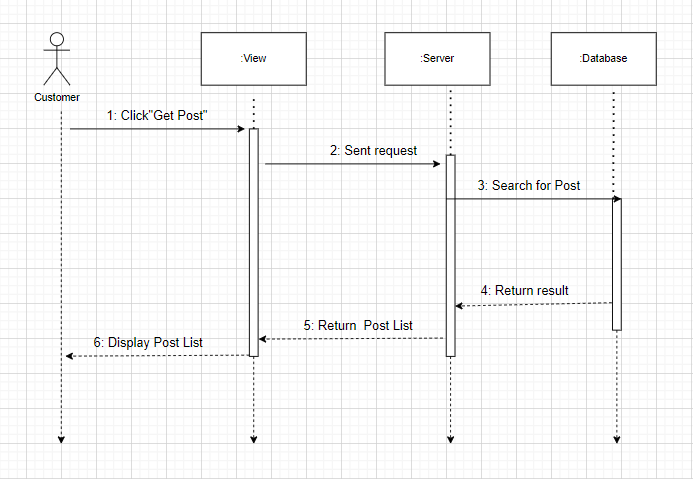
### 4.11 Get Product Detail



### 4.12 Add Product



### 4.13 Get Post List



### 4.14 Get Post Detail

