

**<<ONLINE SHOPPING>>**

**Software Design Document**

– Ho Chi Minh, May 2021 –

Table of Contents

**I.Overview3**

1) Code Packages/Namespaces 3

Package descriptions & package class naming conventions3

2) Database Schema and ERD4

Table descriptions & package class naming conventions 5

**II.** **Code Designs 7**

1) Class Diagram 7

2) Class Specification 7

a. UserDAO7

b. PostDAO8

c. OrderDAO9

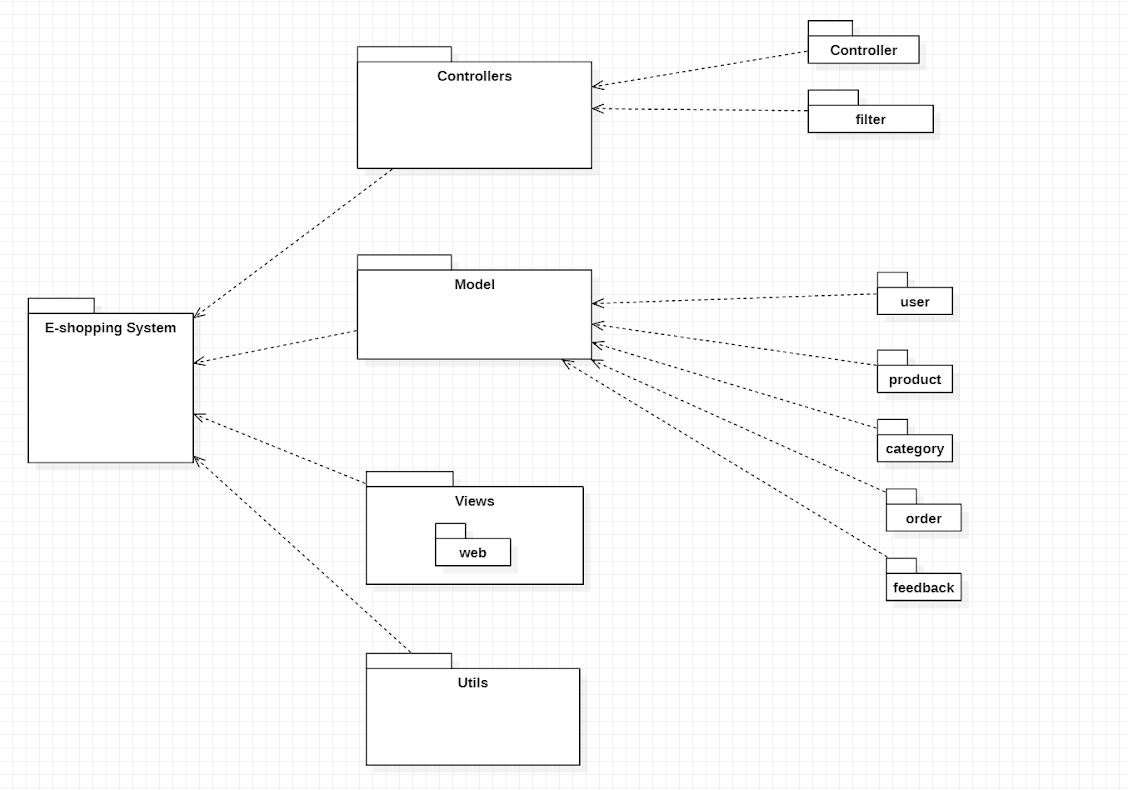
d. ProductDAO10

e. OrderItemDAO11

3) Sequence Diagrams12

**I. Overview**

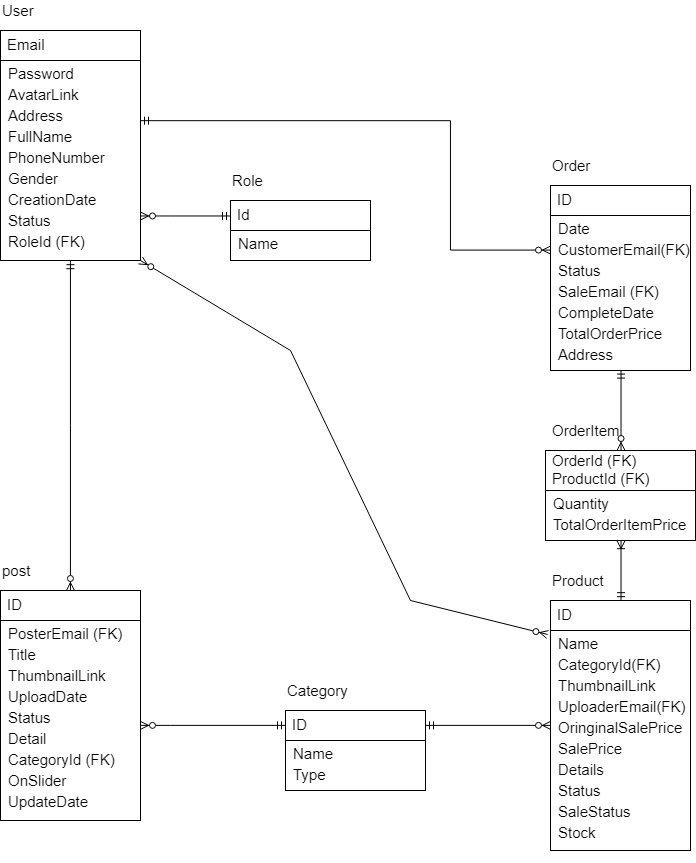
**1.** **Code Packages/Namespaces**

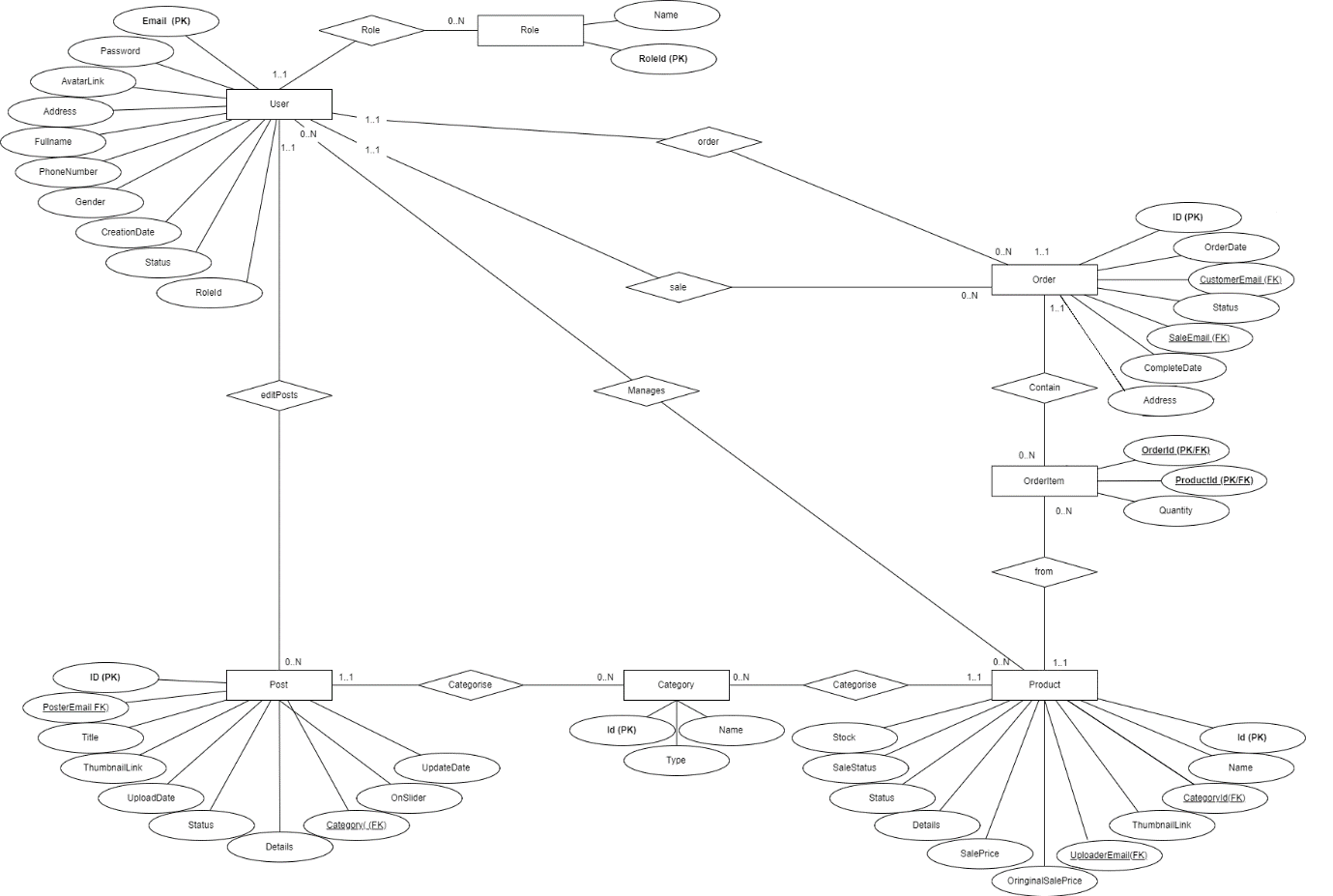


***Package descriptions & package class naming conventions***

|  |  |  |
| --- | --- | --- |
| 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. Ex: loginController, signUpController,... |
| 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 Schema and ERD**



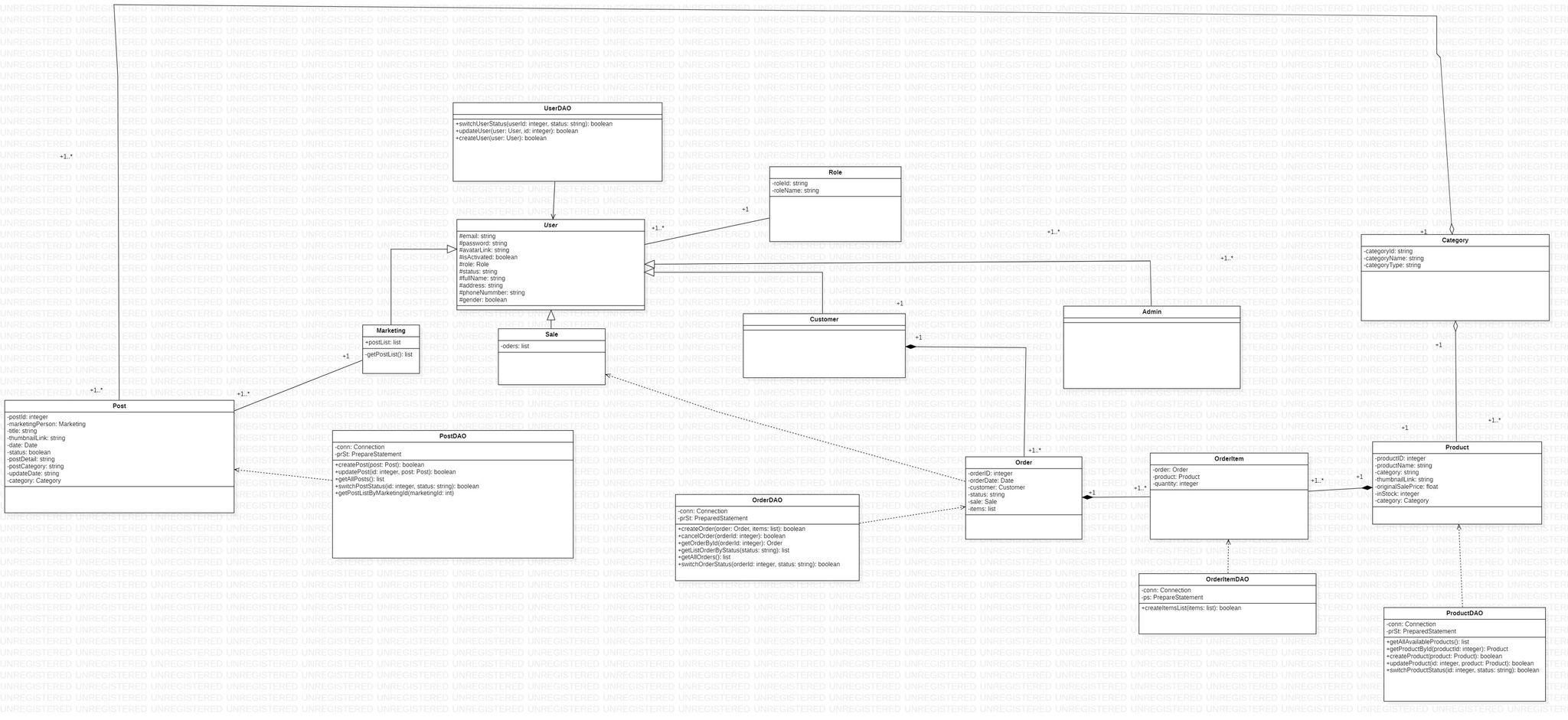


***Table descriptions & package class naming conventions are as below***

|  |  |  |
| --- | --- | --- |
| **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: Id  - Foreign keys: RoleId |
| 02 | Role | Id: Role’s Id  Name: Role’s Name  - Primary keys: Id |
| 03 | Post | Id: 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: Id  - Foreign keys: Email, CategoryID |
| 04 | Category | Id: Category’s Id  Name: Category’s Name  Type: Category’s Type  - Primary keys: Id |
| 05 | Product | Id: 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: Id  - Foreign keys: CategoryID |
| 06 | Order | Id: 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: Id  - Foreign keys: CustomerEmail, SaleEmail |
| 07 | OrderItem | OrderId: 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: Id |

**II. Code Designs**

**1.** **Class Diagram**



**2. Class Specification**

### **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* |

### **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* |

### **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* |
| *07* | *assignOrder* | *- Input: order (Order instance)*  *- Output: A sale that has the least orders will be assigned incoming order*  *- Description: This method call getSaleWithLeastOrders method to figure out the least-occupied sale and assign him/her the order* |

### **ProductDAO**

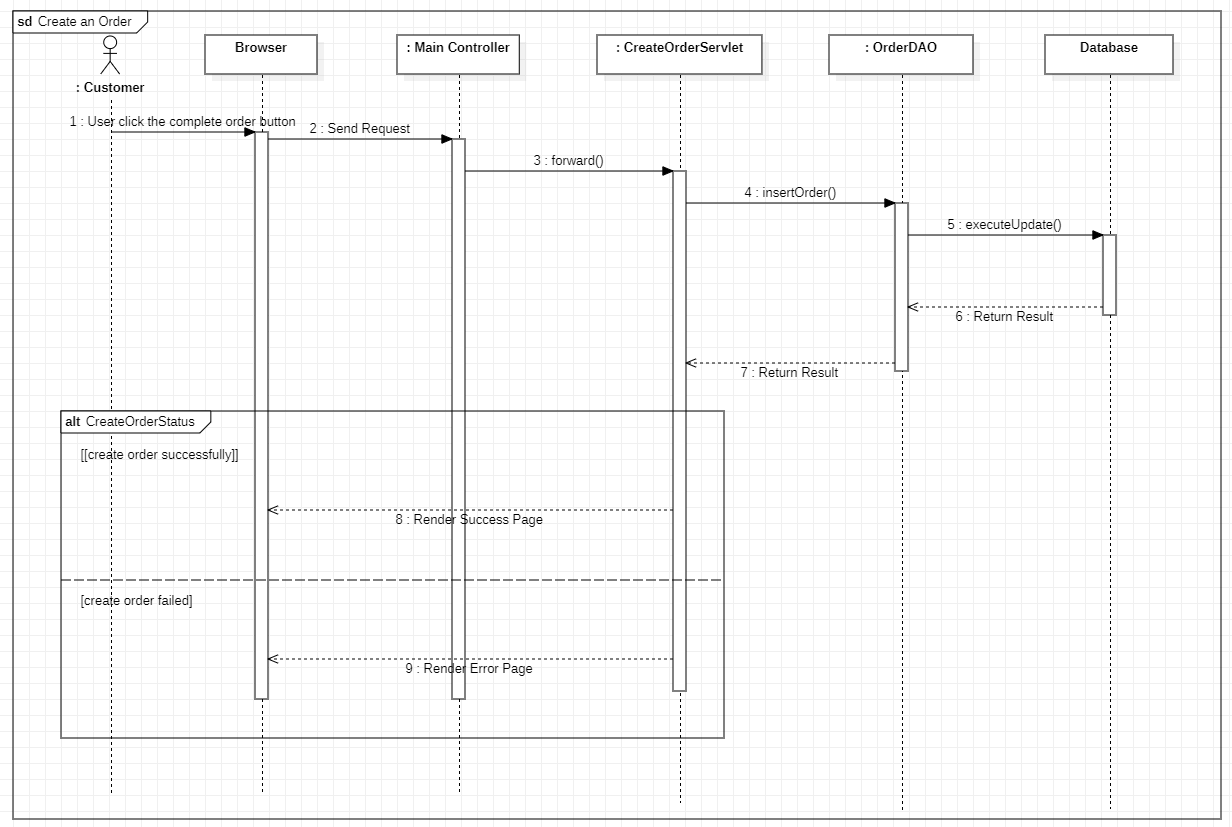
|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| *01* | *getAvailableProducts* | *- Input: none*  *- Output: The list of available products*  *- Description: This method execute query to get a list of available product, which exist in the database and have positive number for 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* |

e. OrderItemDAO

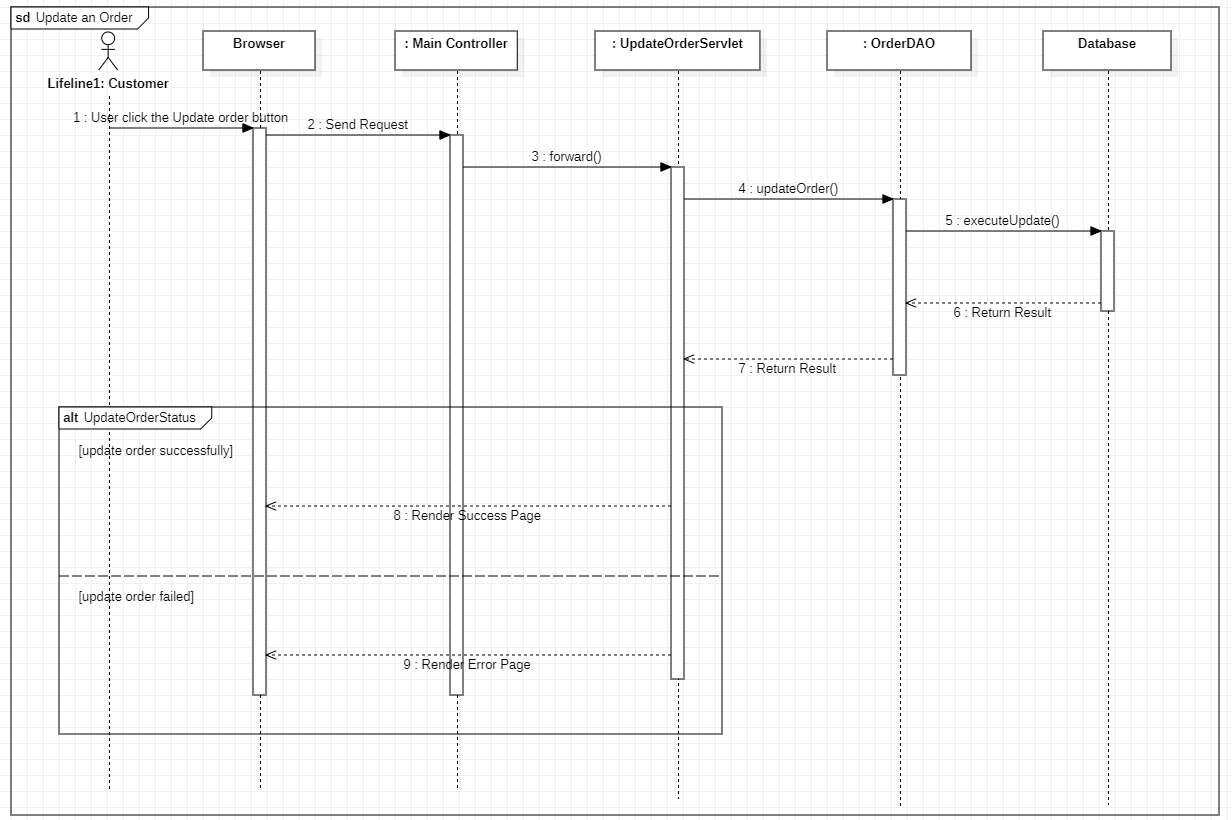
|  |  |  |
| --- | --- | --- |
| ***No*** | ***Method*** | ***Description*** |
| *01* | *createItemsList* | *-Input: List of OrderItem Instance*  *- Output: A boolean value indicate that creation is successful*  *- This method execute query to create a list of order items in the database* |

**3. Sequence Diagram**

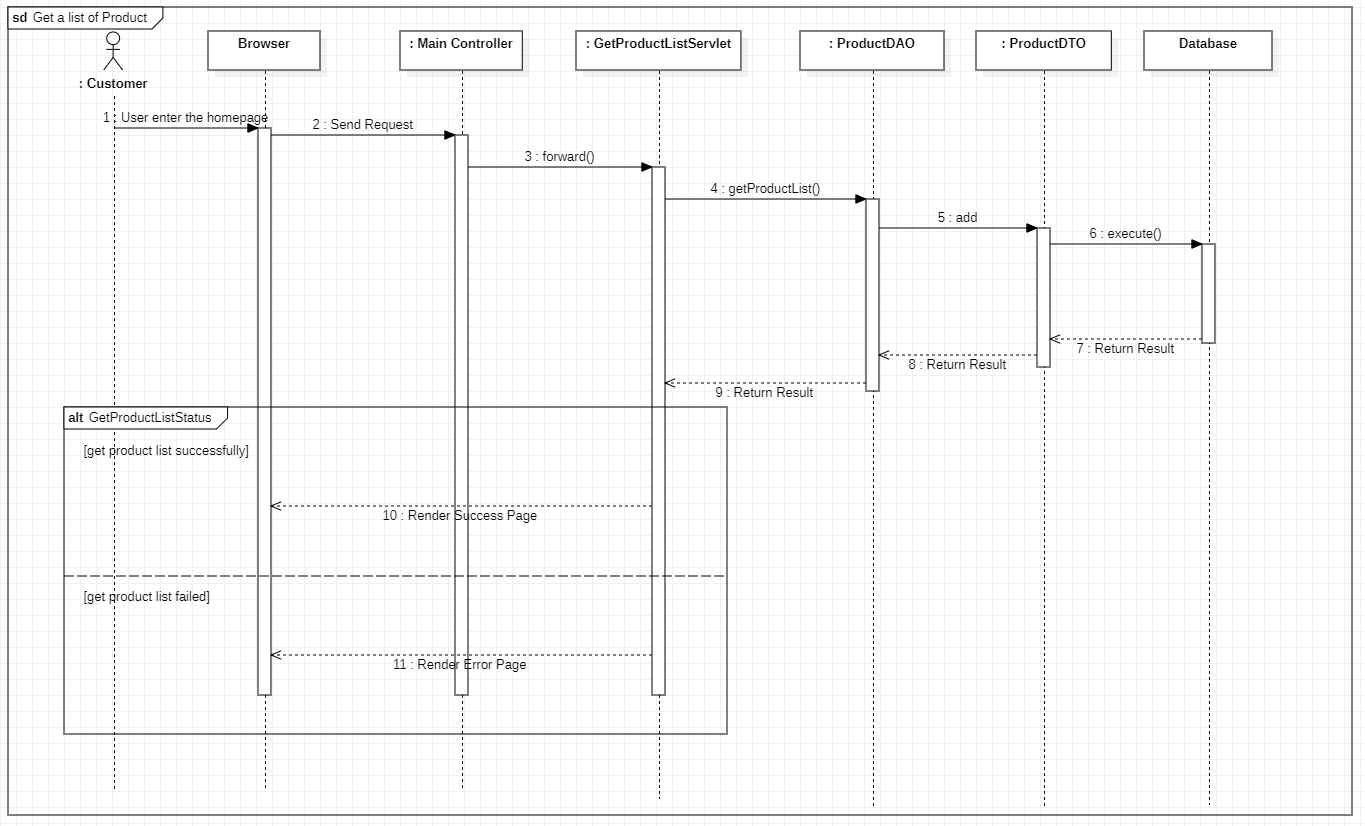
1. Create an Order



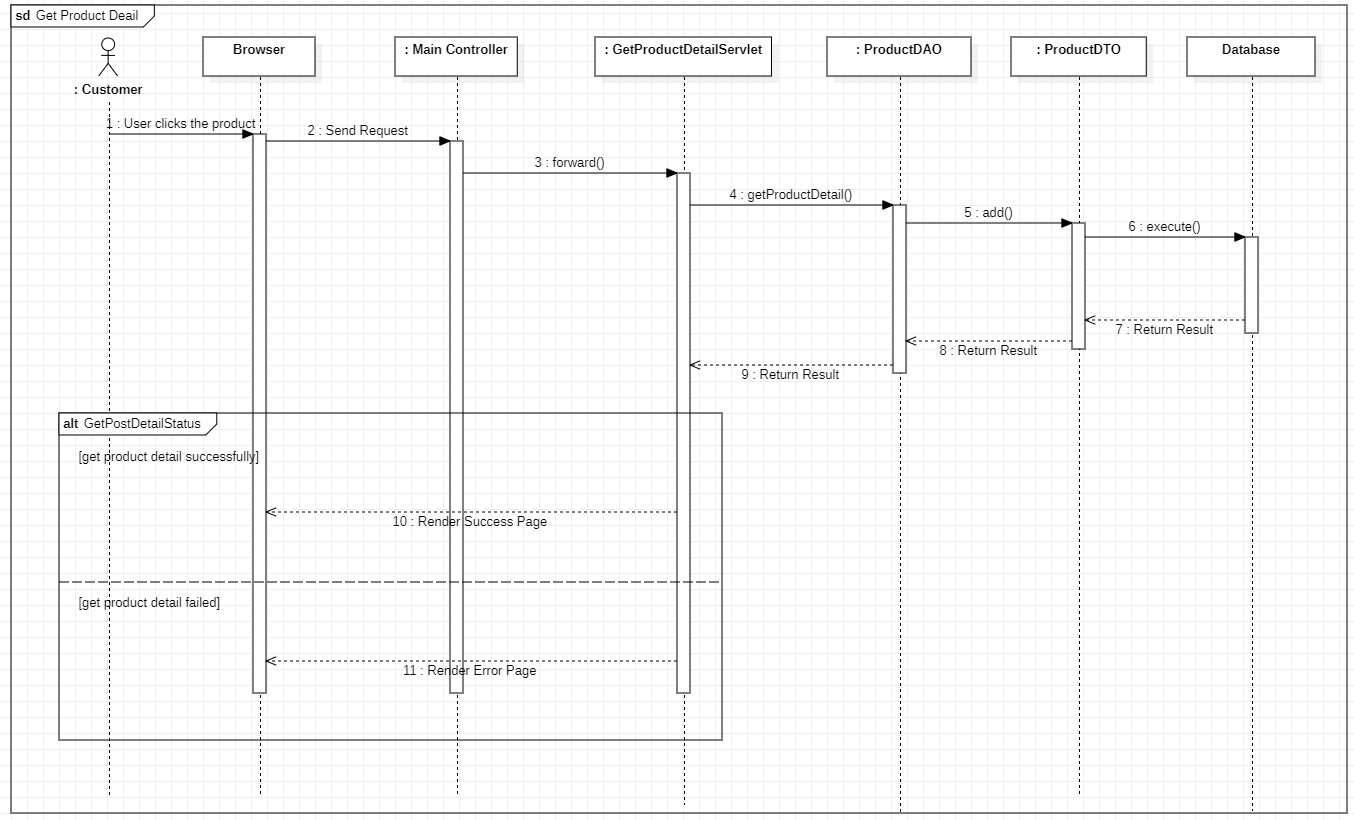
1. Update an Order



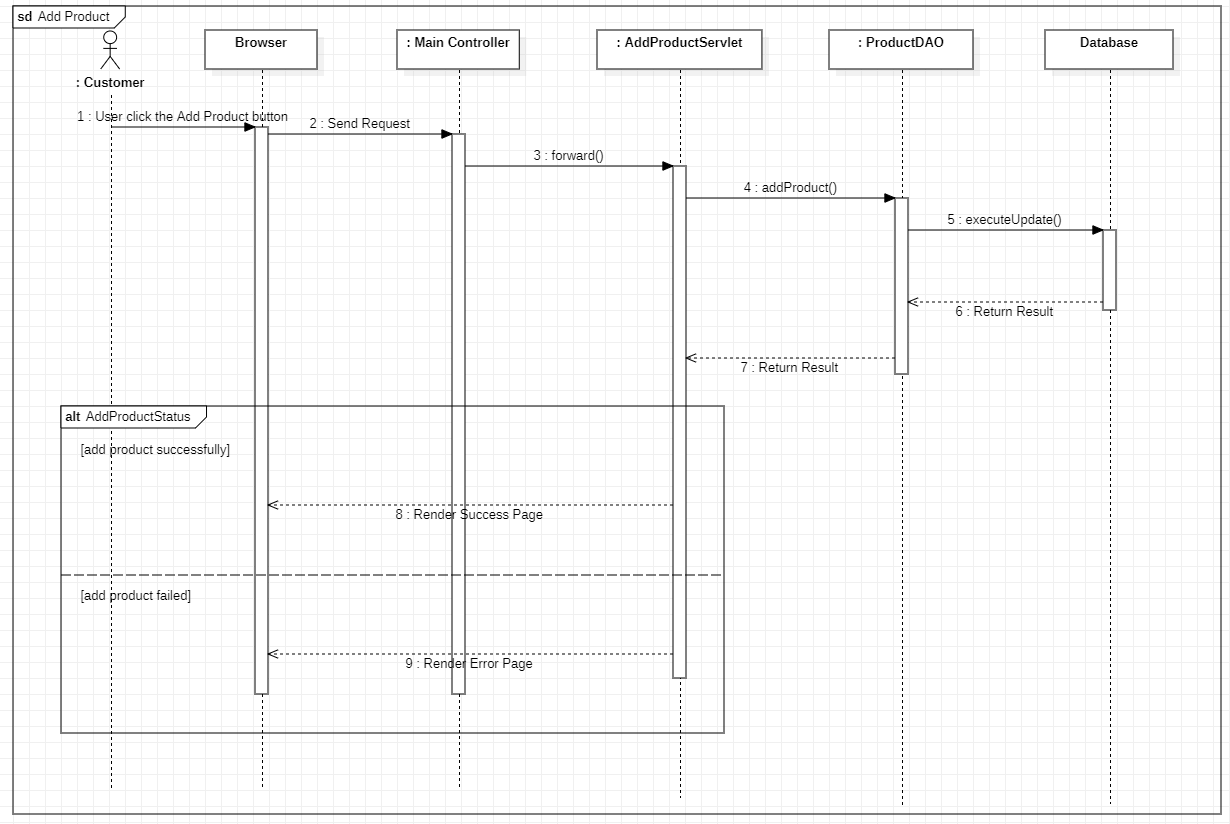
1. Get a List of Product



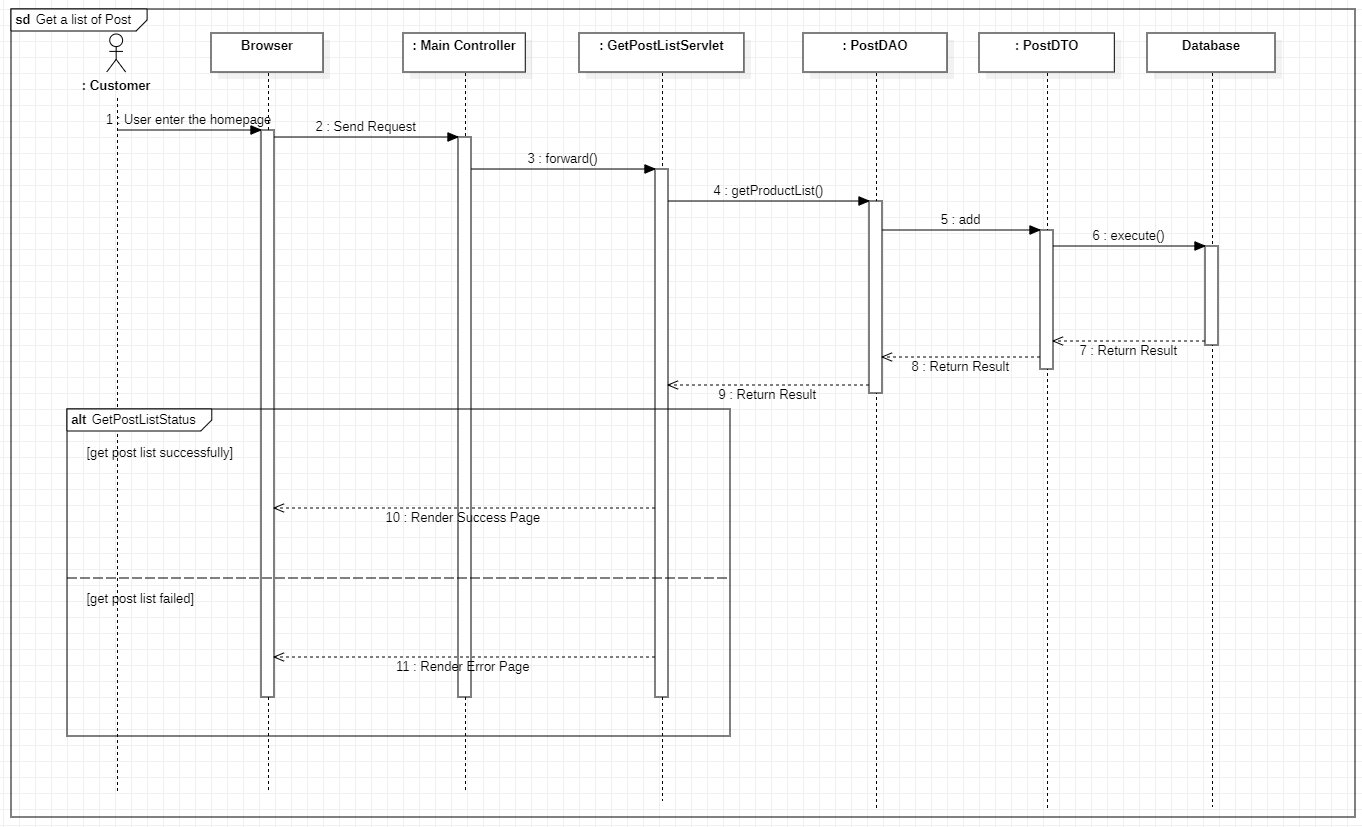
1. Get Product Detail



1. Add Product



1. Get Post List



1. Get Post Detail

