ONLINE SHOPPING MALL MANAGEMENT SYSTEM

ROHIN GOYAL-19BIT0140
SUBHAM S. PANDA-19BIT0093
ADITYA-19BIT0139
AARUSH AGARWAL -19BIT0098

Date

28-02-2021

Course title

Object-Oriented Analysis and Design

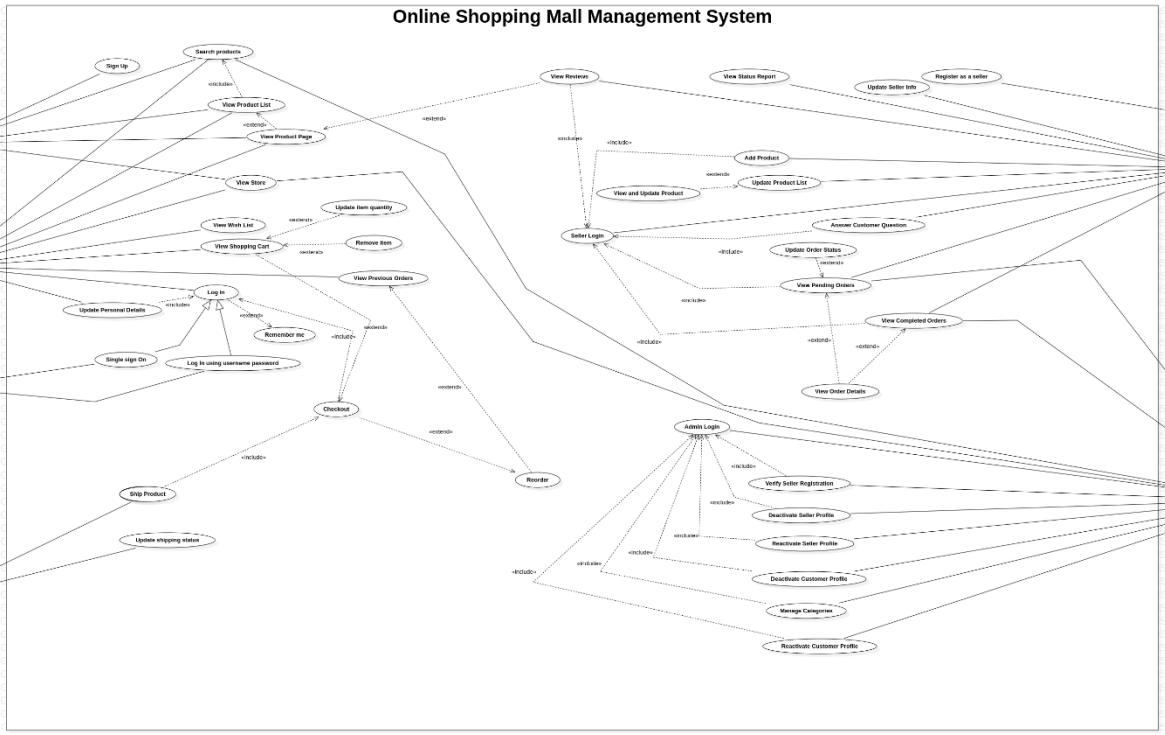
Teacher's name

Kiruba Thangam R

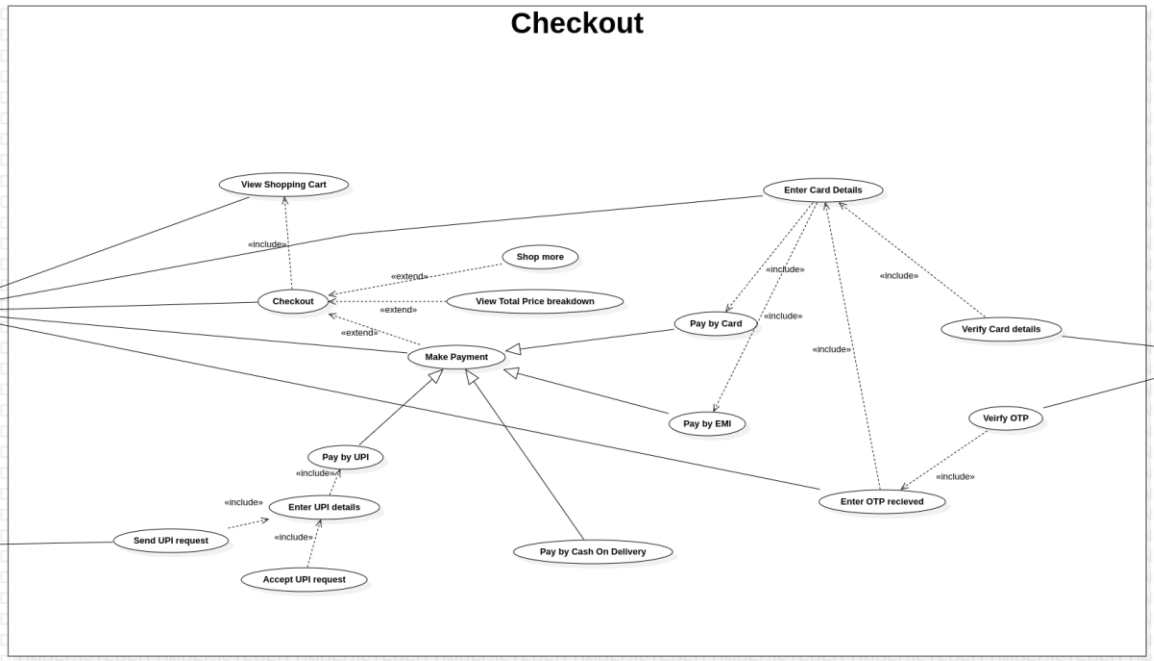
Table of Contents

Review 1:

Overall Key Usecase Diagram:

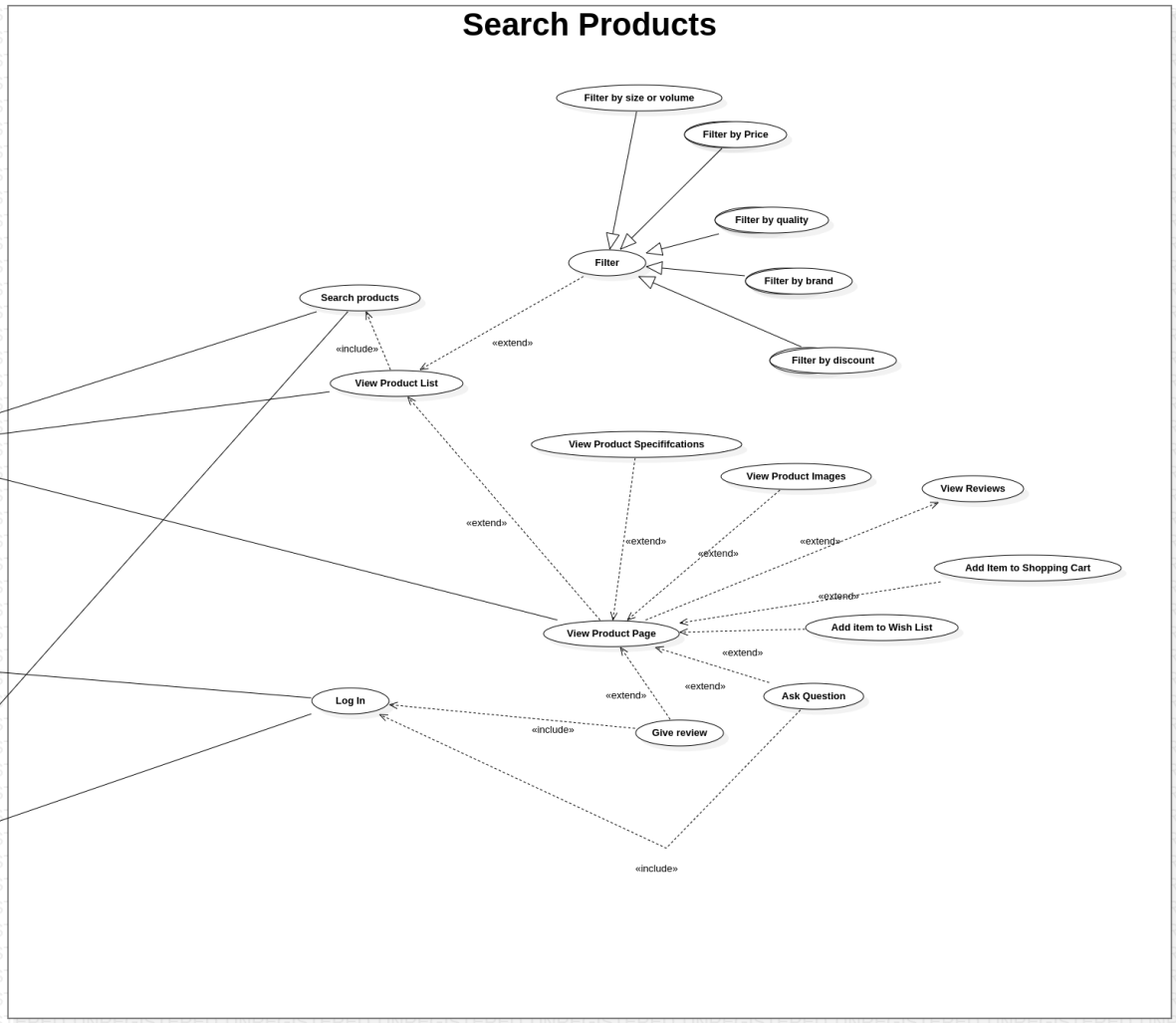


Checkout Key Usecase:



Use-case name	Checkout
Brief Description	After adding the items to the shopping cart the user now wants to buy the items, hence he uses the checkout option.
Basic Flow	1. The user selects to make Payment (or Shop More or View Total Price Breakdown) 2. The system fetches the payment methods available 3. The user selects the payment method 4. If the user chooses to pay by card/ pay by emi: a. The user enters card details b. The Card Verification Agent verifies the card details c. The Card Verification Agent sends verification OTP to user d. The User enters the OTP e. The Card verification Agent verifies the OTP f. Successful Transaction occurs 5. If the user chooses to pay by UPI: a. The user enters his UPI id b. The UPI service provider verifies the UPI id c. The UPI service provider sends a request to the user for transaction d. The user gives permission for transaction e. Successful transaction occurs 6. If the user chooses to pay by Cash on Delivery, required data notified to shipping agent.
Alternate flows	-
Special requirements	1. Secure payment Gateway 2.A successful payment.
Preconditions	Items must be present in the cart
Post conditions	1. The order should be saved in previous orders and must be visible to the user. 2. Payment confirmation sent to the user
Extension points	1a. Shop more 1b. View Total Price Breakdown

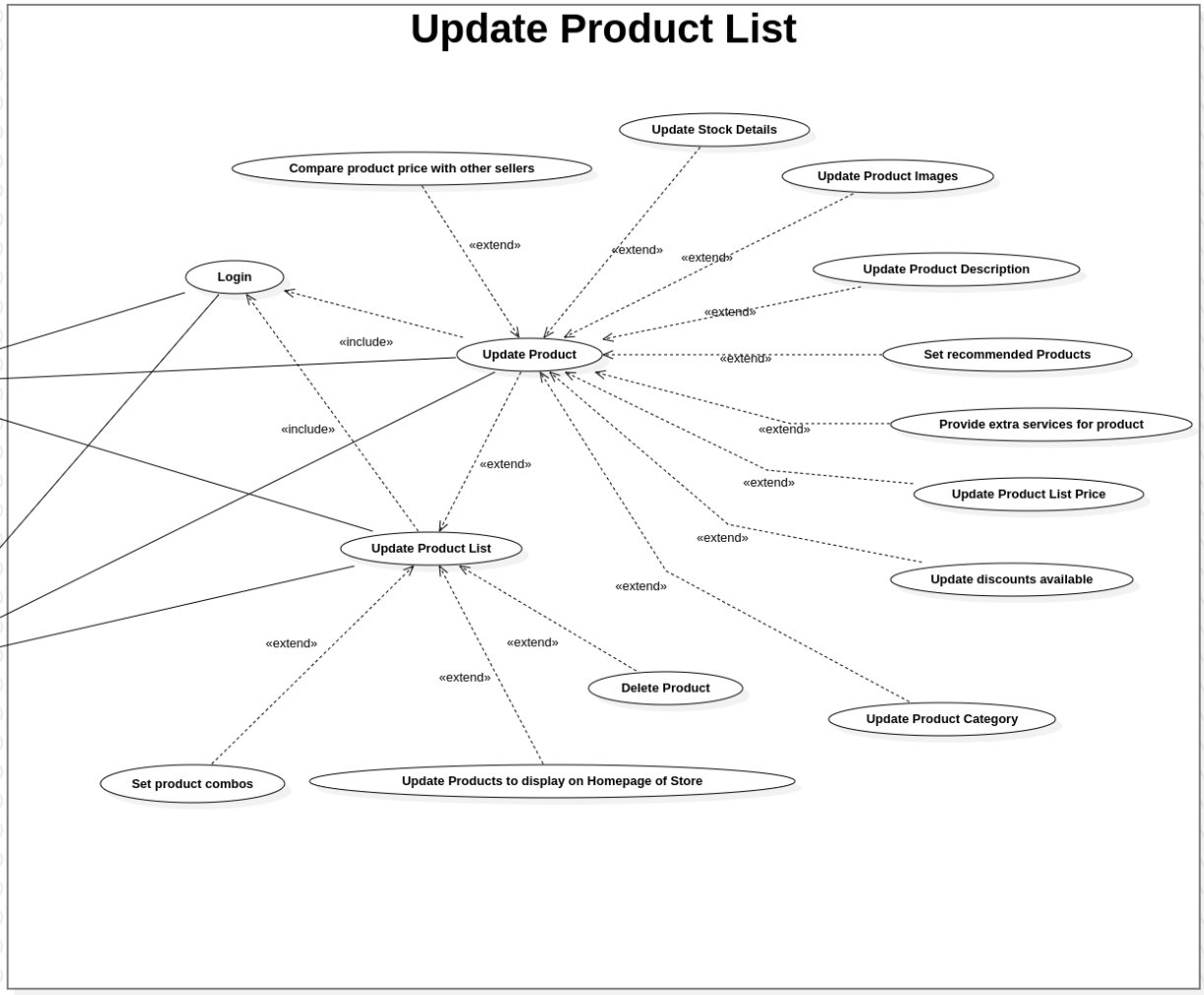
Search Products Key Usecase:



Use-case name	Search products
Brief Description	User uses the Search option to search for products he/she wishes to buy.
Basic Flow	1. The User enters his search query 2. The system fetches the product list 3. The system displays the product list 4. The user selects any products he want to view 5. The system fetches the product details 6. The system displays the product details

	7. The user can specifically view product images, product details and all product reviews. 8. The user can add the product to wish list or shopping cart
Alternate flows	3a. User applies filters (i) The user can apply filters to the fetched product list based on his preferences (ii) The system reconfigures the product list displayed based on filters (iii) Use case resumed at basic flow step 4 7a. If user is logged in (i) The user can give review (ii) The user can ask a question about the product to the seller
Special requirements	The required product is available.
Preconditions	The search bar is visible and easily navigable.
Post conditions	The complete detail page of the product is displayed , if the product is not found, the error page is displayed.
Extension points	3a. Filter by size or vokume, filter by price, filter by brand, filter by discount, filter by quality 7. View Product Specifications, View Product Images, View Reviews 7a. Give Review, Ask Question 8. Add Item to Wish List, Add item to Shopping Cart

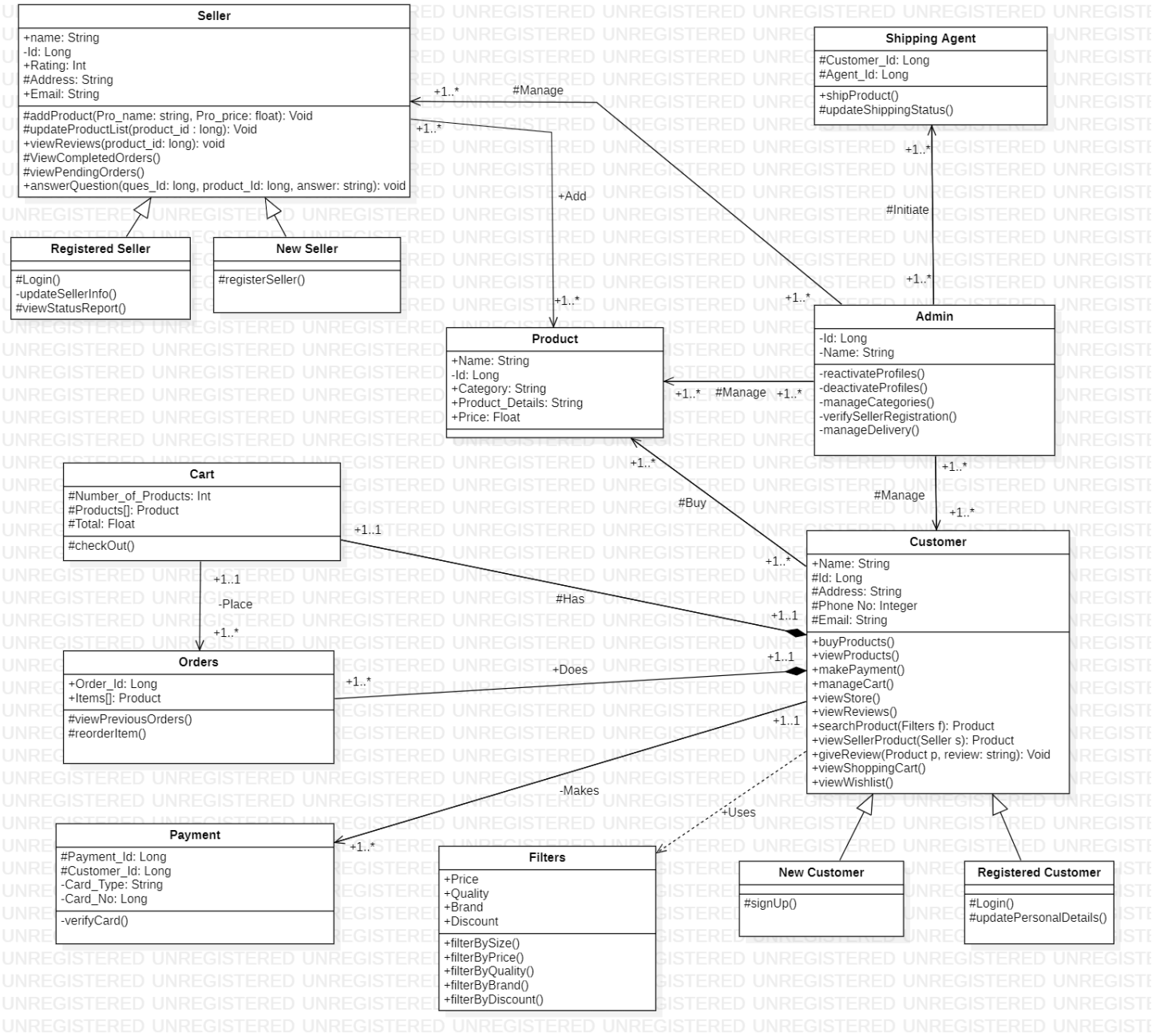
Update Product List Key Usecase:



Use-case name	Update product list
Brief Description	The seller manages his entire product catalogue. The seller can delete products, update product stocks, and set manual product recommendations.
Basic Flow	1. The system fetches the product list for the seller 2. The system displays the product list

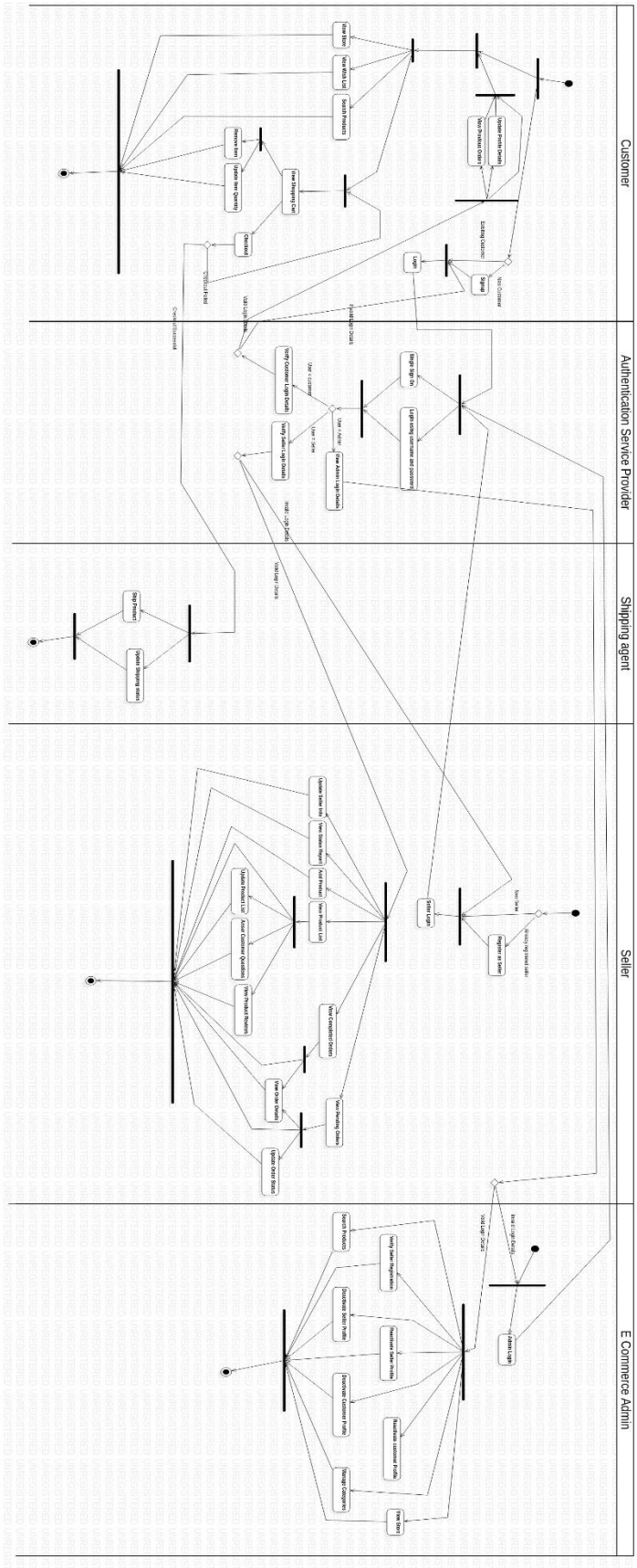
	3. The seller can select a specific product 4. The system fetches the product details 5. The system displays the product details 6. The seller can compare his product with other sellers selling the same product. The seller can update product details like stock, description, recommended products, extra services the seller provides with the product, list price, discount, category of product.
Alternate flows	2a. The seller can also make modifications on the product list level like, delete a product, update products to be displayed on the homepage of the seller store, set product combos by combining products.
Special requirements	The product list should not be empty.
Preconditions	The seller must be a registered seller and logged in to his account.
Post conditions	The updates made by the seller are updated on the shop and visible to the user.
Extension points	6. Compare product price with other sellers, Update Stock Details, Update Product Images, Update Product Description, Set Recommended Products, Provide extra services for product, Update Product List Price, Update Discounts Available, Update Product Category 2a. Set Product Combos, Update Products to display on homepage of store, Delete Product

Class Diagram:

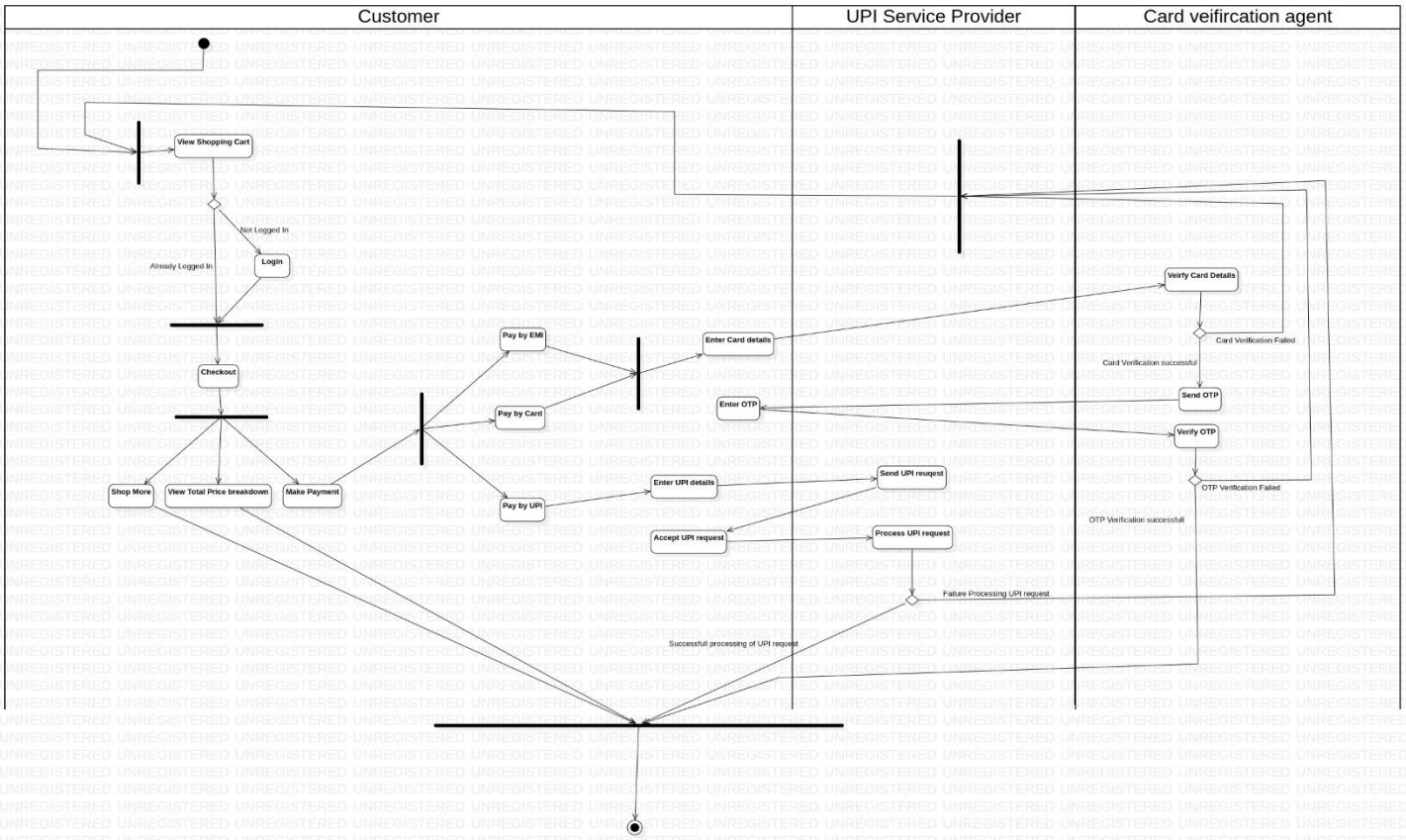


Review 2:

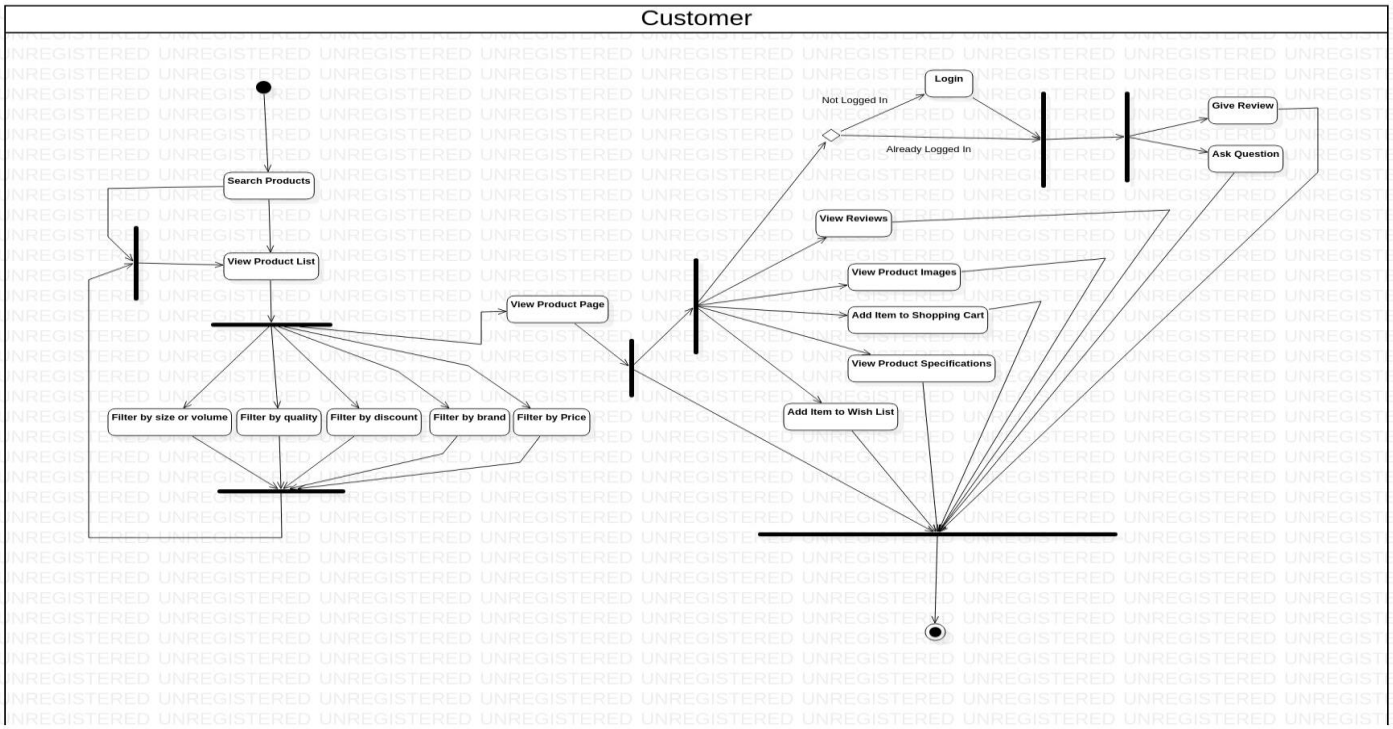
Overall Activity Diagram:


Customer
Authentication Service Provider
Shipping agent
Seller
E Commerce Admin

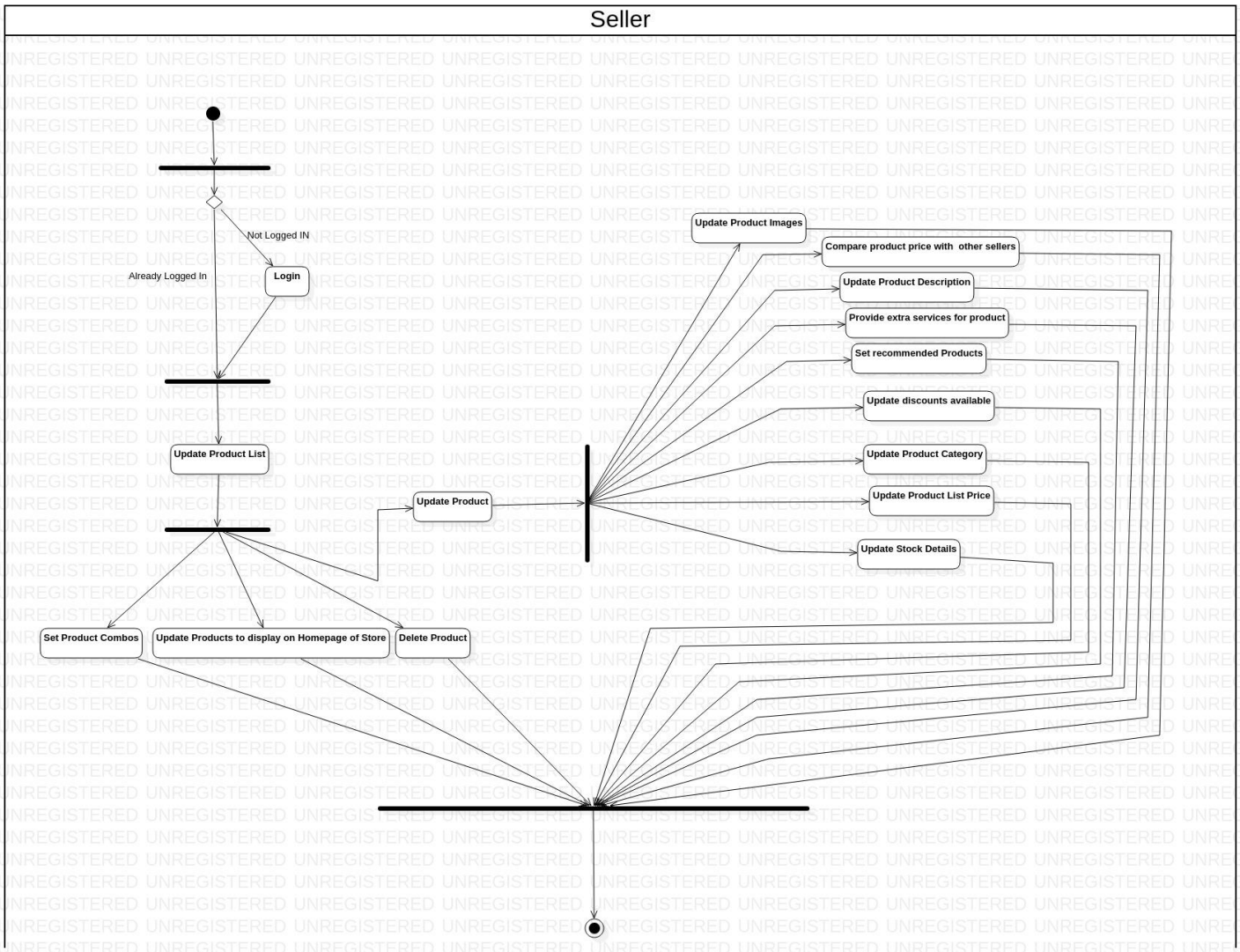
Activity Diagram for Checkout Key Usecase:



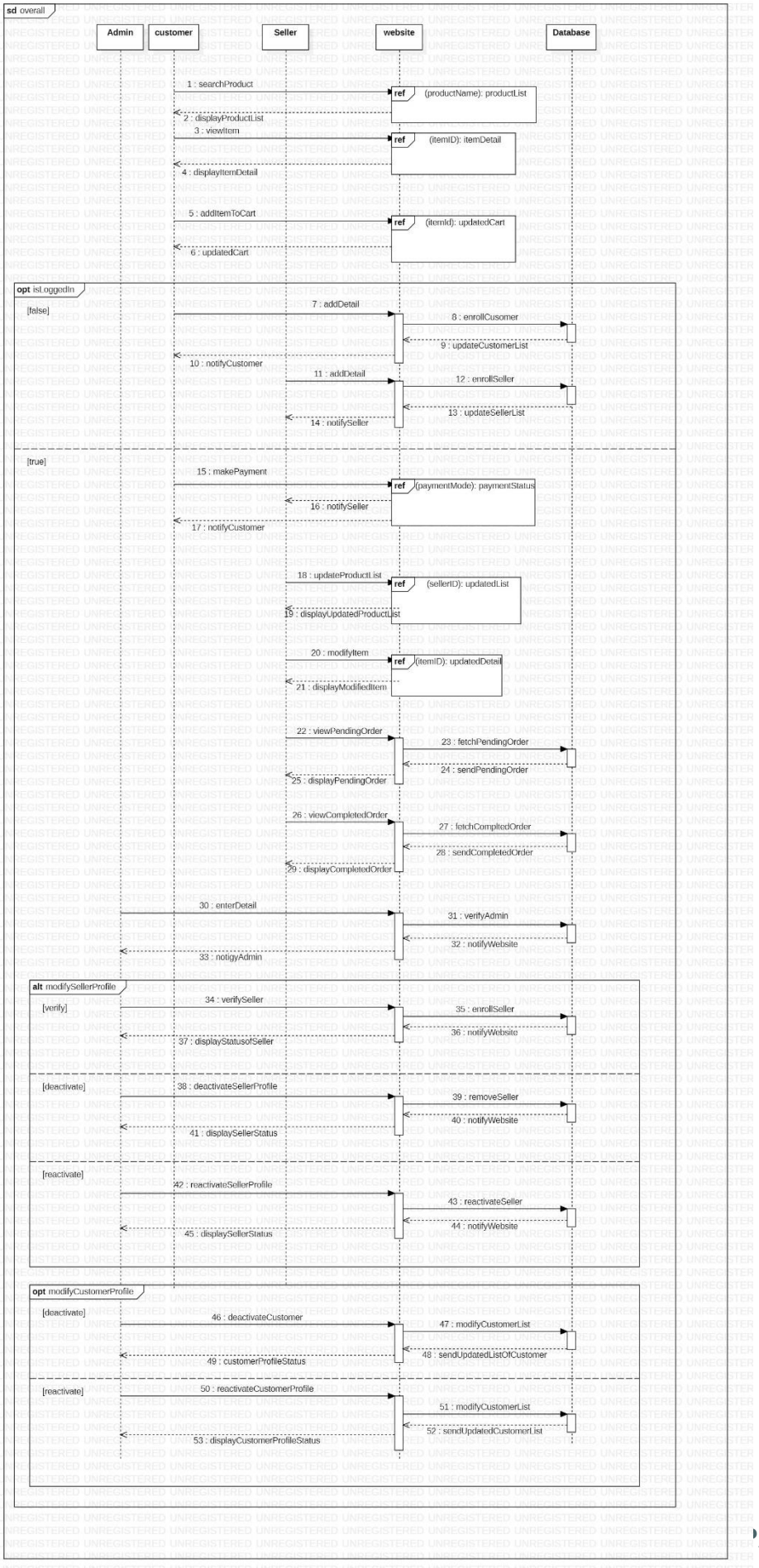
Activity Diagram for Search Product Key Usecase:



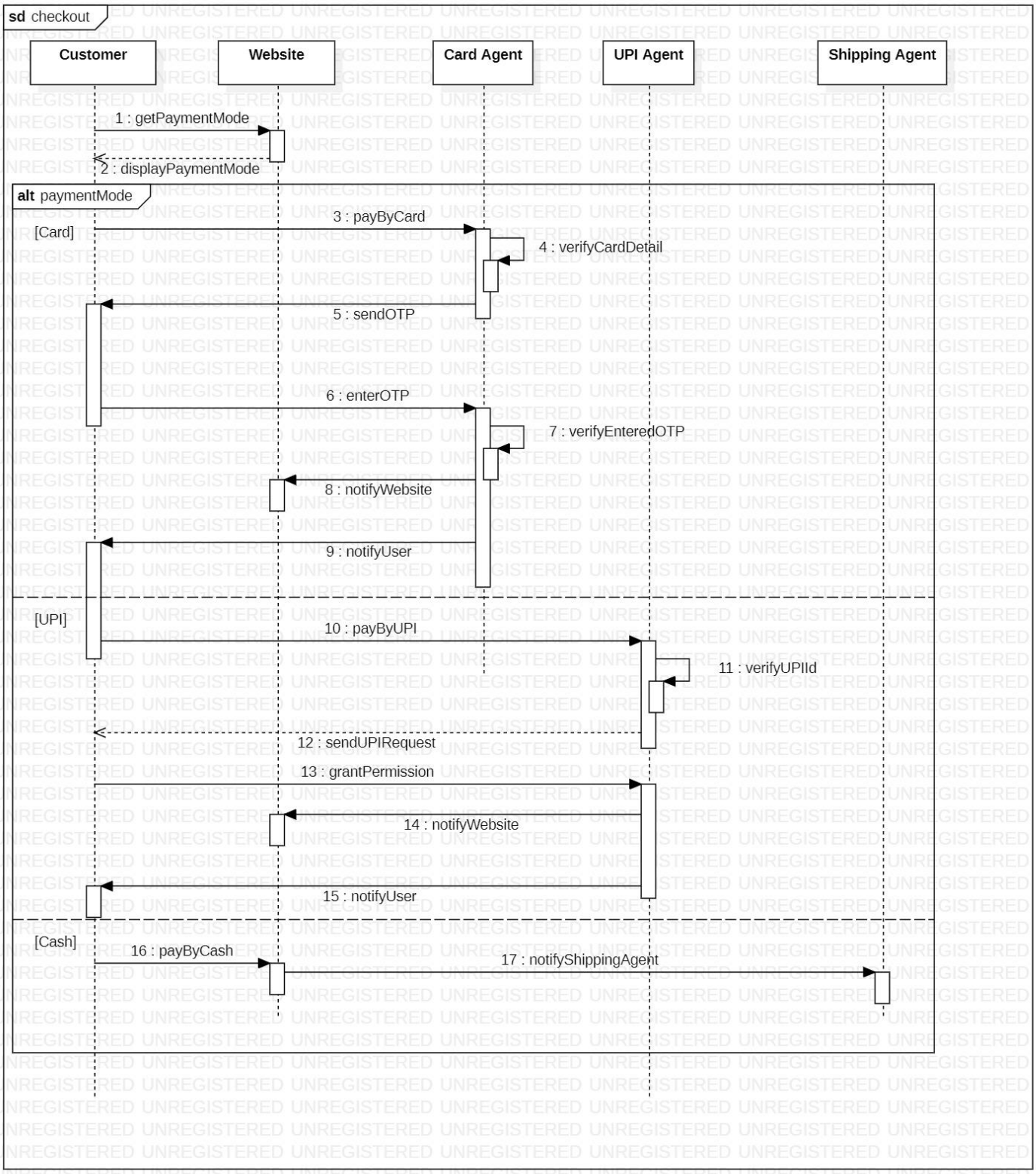
Activity Diagram for Update Product List Key Usecase:



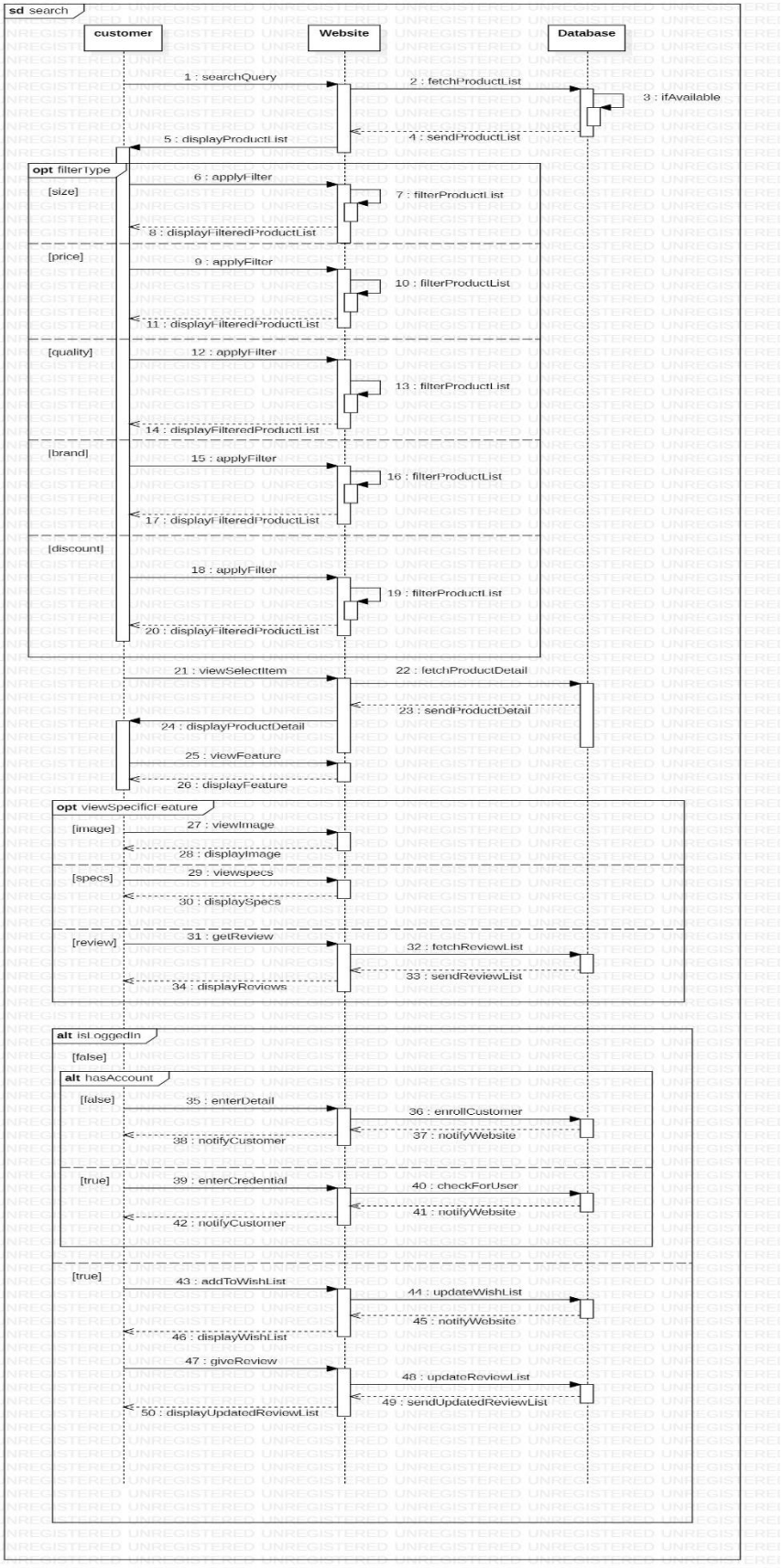
Overall Sequence Diagram:


sd overall
Admin
customer
Seller
website
Database
1 : searchProduct
ref (productName): productList
2 : displayProductList
3 : viewItem
ref (itemID): itemDetail
4 : displayItemDetail
5 : addItemToCart
ref (itemId): updatedCart
6 : updatedCart
opt isLoggedIn
[false]
7 : addDetail
8 : enrollCusomer
9 : updateCustomerList
10 : notifyCustomer
11 : addDetail
12 : enrollSeller
13 : updateSellerList
14 : notifySeller
[true]
15 : makePayment
ref (paymentMode): paymentStatus
16 : notifySeller
17 : notifyCustomer
18 : updateProductList
ref (sellerID): updatedList
19 : displayUpdatedProductList
20 : modifyItem
ref (itemID): updatedDetail
21 : displayModifiedItem
22 : viewPendingOrder
23 : fetchPendingOrder
24 : sendPendingOrder
25 : displayPendingOrder
26 : viewCompletedOrder
27 : fetchCompletedOrder
28 : sendCompletedOrder
29 : displayCompletedOrder
30 : enterDetail
31 : verifyAdmin
32 : notifyWebsite
33 : notigyAdmin
alt modifySellerProfile
[verify]
34 : verifySeller
35 : enrollSeller
36 : notifyWebsite
37 : displayStatusofSeller
[deactivate]
38 : deactivateSellerProfile
39 : removeSeller
40 : notifyWebsite
41 : displaySellerStatus
[reactivate]
42 : reactivateSellerProfile
43 : reactivateSeller
44 : notifyWebsite
45 : displaySellerStatus
opt modifyCustomerProfile
[deactivate]
46 : deactivateCustomer
47 : modifyCustomerList
48 : sendUpdatedListOfCustomer
49 : customerProfileStatus
[reactivate]
50 : reactivateCustomerProfile
51 : modifyCustomerList
52 : sendUpdatedCustomerList
53 : displayCustomerProfileStatus

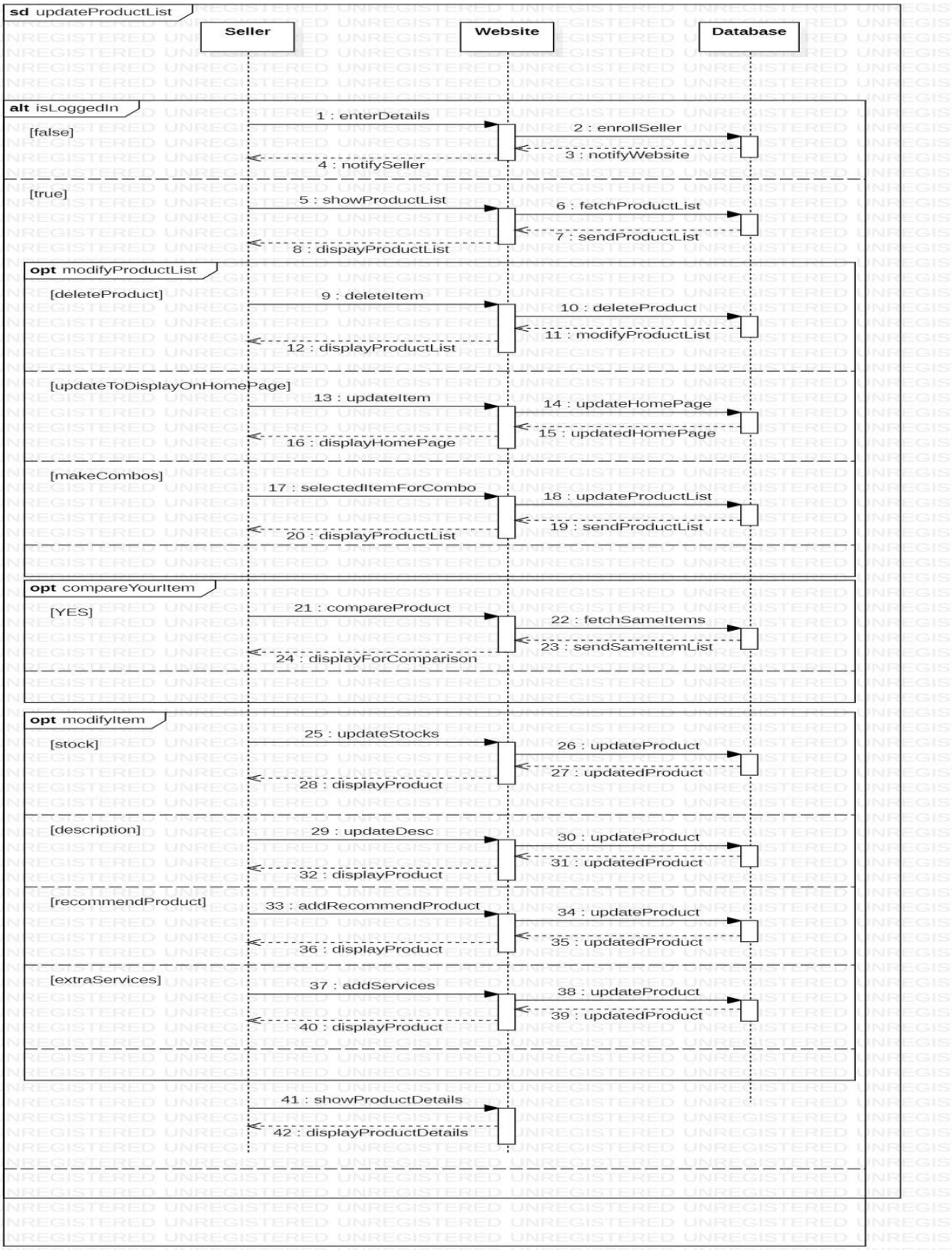
Sequence Diagram for Checkout Key Usecase:



Sequence Diagram for Search Product Key Usecase:

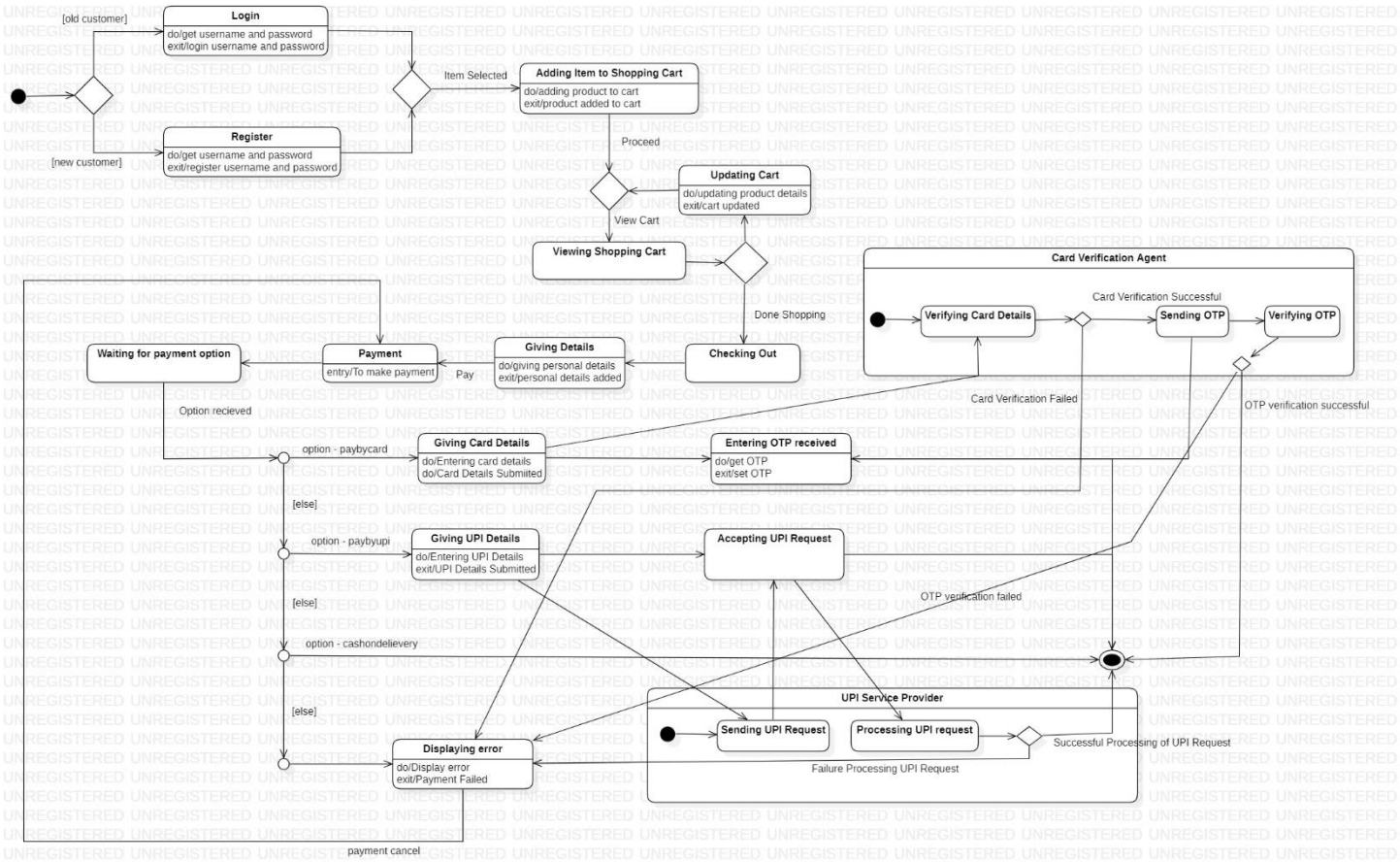

sd search
customer
Website
Database
1 : searchQuery
2 : fetchProductList
3 : ifAvailable
4 : sendProductList
5 : displayProductList
opt filterType
[size]
6 : applyFilter
7 : filterProductList
8 : displayFilteredProductList
[price]
9 : applyFilter
10 : filterProductList
11 : displayFilteredProductList
[quality]
12 : applyFilter
13 : filterProductList
14 : displayFilteredProductList
[brand]
15 : applyFilter
16 : filterProductList
17 : displayFilteredProductList
[discount]
18 : applyFilter
19 : filterProductList
20 : displayFilteredProductList
21 : viewSelectItem
22 : fetchProductDetail
23 : sendProductDetail
24 : displayProductDetail
25 : viewFeature
26 : displayFeature
opt viewSpecificFeature
[image]
27 : viewImage
28 : displayImage
[specs]
29 : viewSpecs
30 : displaySpecs
[review]
31 : getReview
32 : fetchReviewList
33 : sendReviewList
34 : displayReviews
alt isLoggedIn
[false]
alt hasAccount
[false]
35 : enterDetail
36 : enrollCustomer
37 : notifyWebsite
38 : notifyCustomer
[true]
39 : enterCredential
40 : checkForUser
41 : notifyWebsite
42 : notifyCustomer
[true]
43 : addToWishList
44 : updateWishList
45 : notifyWebsite
46 : displayWishList
47 : giveReview
48 : updateReviewList
49 : sendUpdatedReviewList
50 : displayUpdatedReviewList

Sequence Diagram for Update Product List Key Usecase:

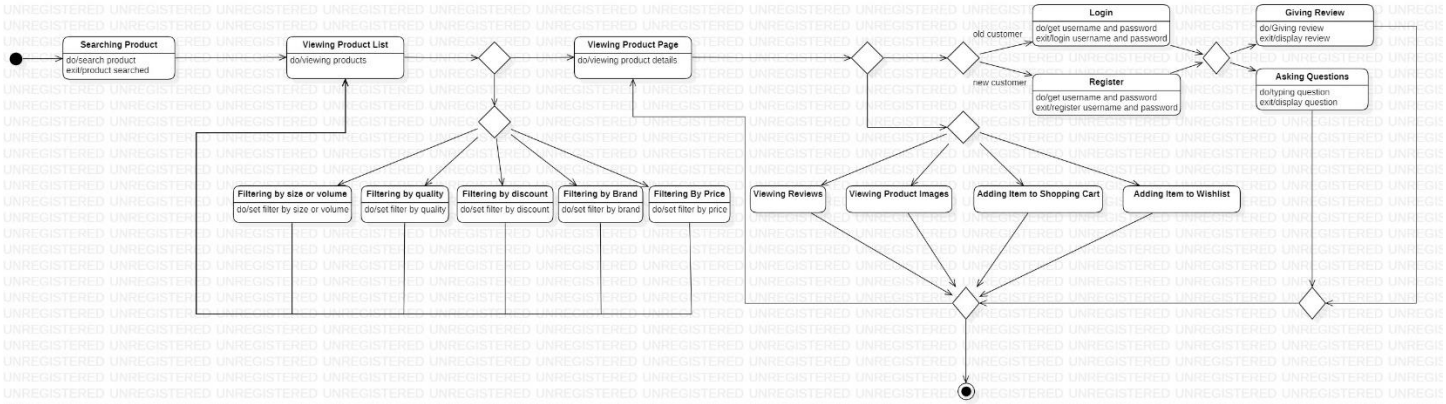


Review 3:

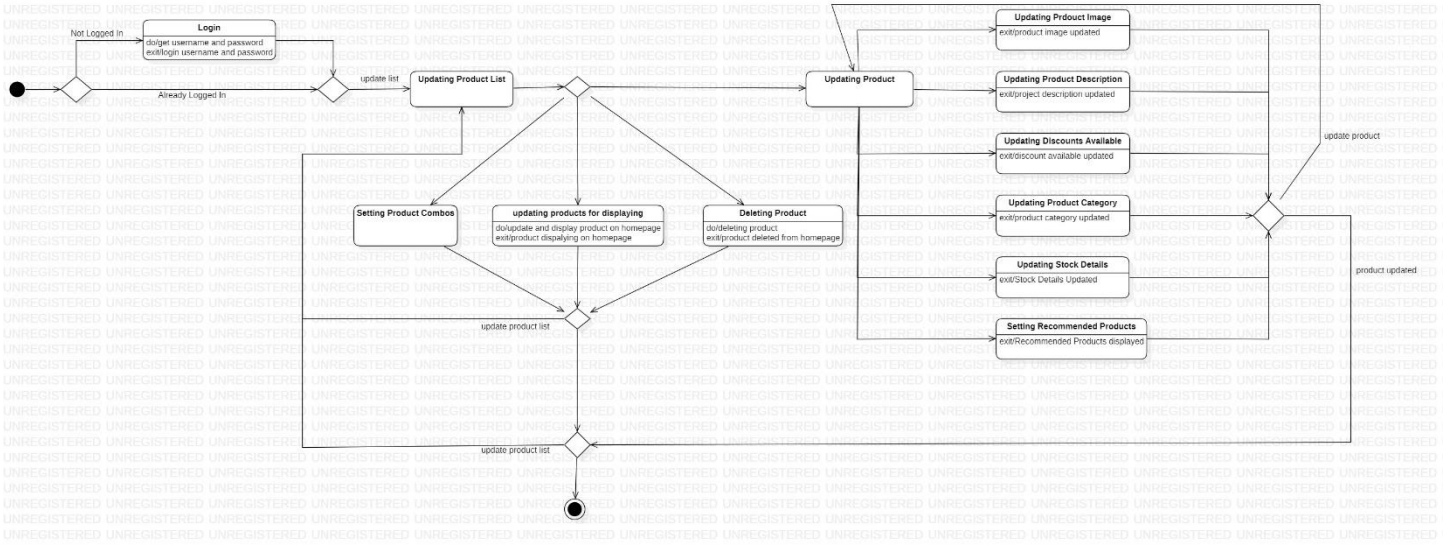
Statechart Diagram for Checkout Key Usecase:



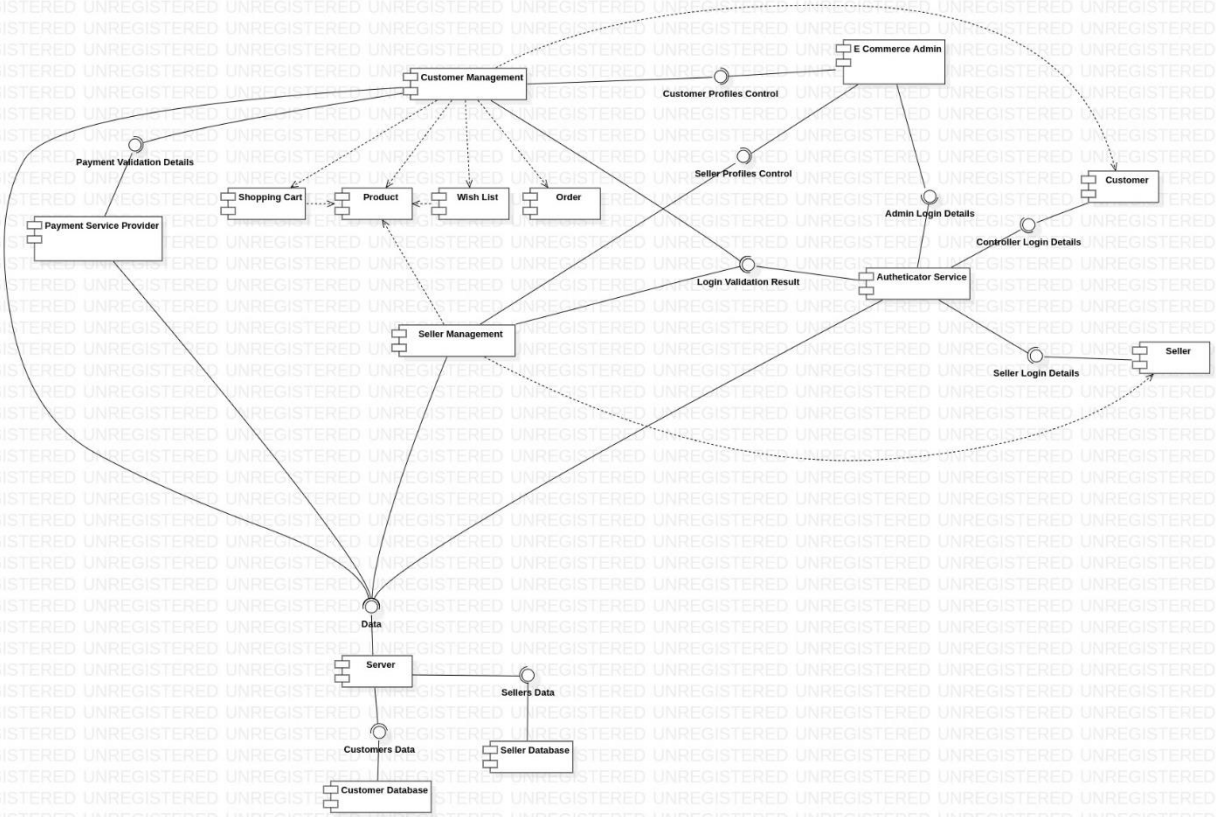
Statechart Diagram for Search Product Key Usecase:



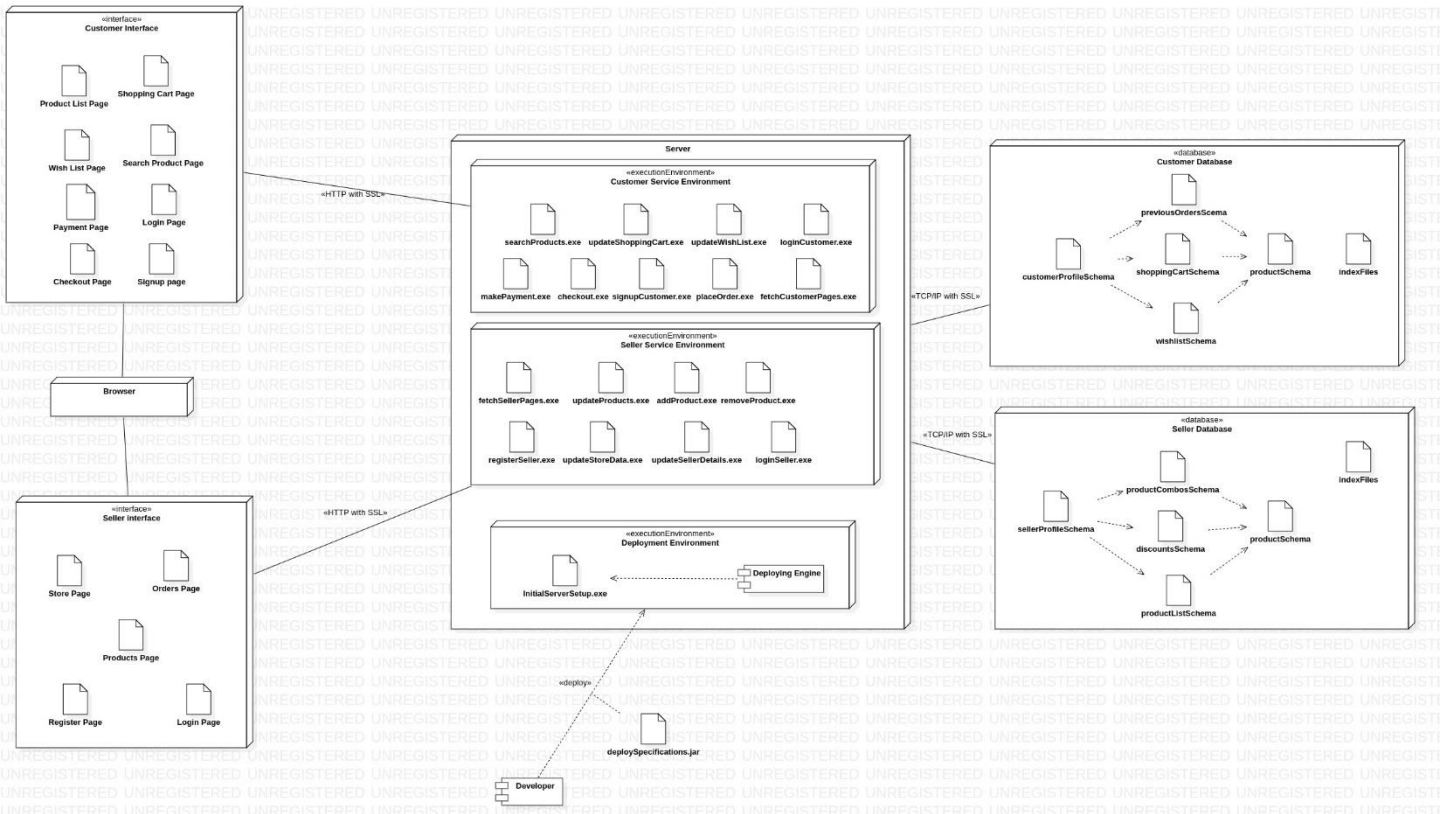
Statechart Diagram for Update Product List Key Usecase:



Component Diagram:



Deployment Diagram:



Code Generated:

Class Admin:

```
import java.util.*;

/**
 *
 */
public class Admin {

    /**
     * Default constructor
     */
    public Admin() {
    }

    /**
     *
     */
    private Long Id;

    /**
     *
     */
    private String Name;




    /**
     *
     */
    public Seller 1..*;

    /**
     *
     */
    public Shipping Agent 1..*;

    /**
     *
     */
    public Product 1..*;

    /**
     *
     */
    public Customer 1..*;

    /**
     *
```

```
     */
    private void reactivateProfiles() {
        // TODO implement here
    }

    /**
     *
     */
    private void deactivateProfiles() {
        // TODO implement here
    }

    /**
     *
     */
    private void manageCategories() {
        // TODO implement here
    }

    /**
     *
     */
    private void verifySellerRegistration() {
        // TODO implement here
    }

    /**
     *
     */
    private void manageDelivery() {
        // TODO implement here
    }

}
```

Class Cart:

```
import java.util.*;

/**
 *
 */
public class Cart {

    /**
     * Default constructor
     */
    public Cart() {
    }

    /**
     *
     */
    protected Int Number_of_Products;
```

```
    /**
     *
     */
    protected Product Products[];

    /**
     *
     */
    protected Float Total;


    /**
     *
     */
    public Orders 1..*;

    /**
     *
     */
    public Customer 1..1;

    /**
     *
     */
    protected void checkOut() {
        // TODO implement here
    }
}
```

Class Customer:

```
import java.util.*;

/**
 *
 */
public class Customer {

    /**
     * Default constructor
     */
    public Customer() {
    }

    /**
     *
     */
    public String Name;

    /**
     *
     */
    protected Long Id;

    /**
     *
     */
```

```
protected String Address;

/**
 *
 */
protected Integer Phone No;

/**
 *
 */
protected String Email;




/**
 *
 */
public Cart 1..1;

/**
 *
 */
public Payment 1..*;

/**
 *
 */
public Product 1..*;

/**
 *
 */
public Orders 1..*;


/**
 *
 */
public Cart 1..1;

/**
 *
 */
public Orders 1..*;

/**
 *
 */
public void buyProducts() {
    // TODO implement here
}

/**
 *
 */
public void viewProducts() {
```

```
    // TODO implement here
}

/**
 *
 */
public void makePayment() {
    // TODO implement here
}

/**
 *
 */
public void manageCart() {
    // TODO implement here
}

/**
 *
 */
public void viewStore() {
    // TODO implement here
}

/**
 *
 */
public void viewReviews() {
    // TODO implement here
}

/**
 * @param Filters f
 * @return
 */
public Product searchProduct(void Filters f) {
    // TODO implement here
    return null;
}

/**
 * @param Seller s
 * @return
 */
public Product viewSellerProduct(void Seller s) {
    // TODO implement here
    return null;
}

/**
 * @param Product p
 * @param review
 * @return
 */
public Void giveReview(void Product p, string review) {
    // TODO implement here
    return null;
}
```

```
    /**
     *
     */
    public void viewShoppingCart() {
        // TODO implement here
    }

    /**
     *
     */
    public void viewWishlist() {
        // TODO implement here
    }

}
```

Class Filters:

```
import java.util.*;

/**
 *
 */
public class Filters {

    /**
     * Default constructor
     */
    public Filters() {
    }

    /**
     *
     */
    public void Price;

    /**
     *
     */
    public void Quality;

    /**
     *
     */
    public void Brand;

    /**
     *
     */
    public void Discount;

    /**
     *
     */
    public void filterBySize() {
        // TODO implement here
```

```
    }

    /**
     *
     */
    public void filterByPrice() {
        // TODO implement here
    }

    /**
     *
     */
    public void filterByQuality() {
        // TODO implement here
    }

    /**
     *
     */
    public void filterByBrand() {
        // TODO implement here
    }

    /**
     *
     */
    public void filterByDiscount() {
        // TODO implement here
    }

}
```

Class New Customer:

```
import java.util.*;

/**
 *
 */
protected class New Customer extends Customer {

    /**
     * Default constructor
     */
    protected New Customer() {
    }

    /**
     *
     */
    protected void signUp() {
        // TODO implement here
    }

}
```

Class New Seller:

```
import java.util.*;

/**
 *
 */
public class New Seller extends New Seller {

    /**
     * Default constructor
     */
    public New Seller() {
    }




    /**
     *
     */
    protected void registerSeller() {
        // TODO implement here
    }

}
```

Class Orders:

```
import java.util.*;

/**
 *
 */
public class Orders {

    /**
     * Default constructor
     */
    public Orders() {
    }

    /**
     *
     */
    public Long Order_Id;

    /**
     *
     */
    public Product Items[];

    /**
     *
     */
    public Customer 1..1;
```

```
    /**
     *
     */
    protected void viewPreviousOrders() {
        // TODO implement here
    }

    /**
     *
     */
    protected void reorderItem() {
        // TODO implement here
    }

}
```

Class Payment:

```
import java.util.*;

/**
 *
 */
public class Payment {

    /**
     * Default constructor
     */
    public Payment() {
    }

    /**
     *
     */
    protected Long Payment_Id;

    /**
     *
     */
    protected Long Customer_Id;

    /**
     *
     */
    private String Card_Type;

    /**
     *
     */
    private Long Card_No;



    /**
     *
     */
```

```
    private void verifyCard() {
        // TODO implement here
    }

}
```

Class Product:

```
import java.util.*;

/**
 *
 */
public class Product {

    /**
     * Default constructor
     */
    public Product() {
    }

    /**
     *
     */
    public String Name;

    /**
     *
     */
    private Long Id;

    /**
     *
     */
    public String Category;

    /**
     *
     */
    public String Product_Details;

    /**
     *
     */
    public Float Price;



}
```

Class Registered Customer:

```
import java.util.*;
```

```
/**
 *
 */
public class Registered Customer extends Customer {

    /**
     * Default constructor
     */
    public Registered Customer() {
    }

    /**
     *
     */
    protected void Login() {
        // TODO implement here
    }

    /**
     *
     */
    protected void updatePersonalDetails() {
        // TODO implement here
    }

}
```

Class Registered Seller:

```
import java.util.*;

/**
 *
 */
public class Registered Seller extends Registered Seller {

    /**
     * Default constructor
     */
    public Registered Seller() {
    }

    /**
     *
     */
    protected void Login() {
        // TODO implement here
    }

    /**
     *
     */
    private void updateSellerInfo() {
        // TODO implement here
    }

    /**
     *
```

```
     */
    protected void viewStatusReport() {
        // TODO implement here
    }

}
```

Class Seller:

```
import java.util.*;

/**
 *
 */
public class Seller extends Seller {

    /**
     * Default constructor
     */
    public Seller() {
    }

    /**
     *
     */
    public String name;

    /**
     *
     */
    private Long Id;

    /**
     *
     */
    public Int Rating;

    /**
     *
     */
    protected String Address;

    /**
     *
     */
    public String Email;

    /**
     *
     */
    public Product 1..*;



    /**
     * @param Pro_name
     * @param Pro_price
```

```
     * @return
     */
    protected Void addProduct(string Pro_name, float Pro_price) {
        // TODO implement here
        return null;
    }

    /**
     * @param product_id
     * @return
     */
    protected Void updateProductList(long product_id ) {
        // TODO implement here
        return null;
    }

    /**
     * @param product_id
     * @return
     */
    public void viewReviews(long product_id) {
        // TODO implement here
        return null;
    }

    /**
     *
     */
    protected void ViewCompletedOrders() {
        // TODO implement here
    }

    /**
     *
     */
    protected void viewPendingOrders() {
        // TODO implement here
    }

    /**
     * @param ques_Id
     * @param product_Id
     * @param answer
     * @return
     */
    public void answerQuestion(long ques_Id, long product_Id, string answer) {
        // TODO implement here
        return null;
    }

}
```

Class Shipping Agent:

```
import java.util.*;

/**
```

```
 *
 */
public class Shipping Agent {

    /**
     * Default constructor
     */
    public Shipping Agent() {
    }

    /**
     *
     */
    protected Long Customer_Id;

    /**
     *
     */
    protected Long Agent_Id;


    /**
     *
     */
    public void shipProduct() {
        // TODO implement here
    }

    /**
     *
     */
    protected void updateShippingStatus() {
        // TODO implement here
    }

}
```