Group Number: 1

Group Members: Katie Dao, Reynaldo Williams

Application: Shopping Cart

Description

This application is to mimic a shopping cart experience that mirrors that of online shopping platforms such as Amazon using the Swing GUI widget for Java. Basic functions such as browsing and adding items into a virtual shopping cart are created to allow the user to purchase these items. Other functions are created for the users who are sellers of this application, this includes but is not limited to list of inventory and calculation of sales. Detailed specifications are listed below.

Functional Specification

Login Specification

The users are provided with a login screen in which they will enter in their username and password credentials for their previously created accounts. If the user is a seller, the users are given a checkbox to indication such information.

Logout Specification

If and when the user decides they are done browsing our application, they are given a "Logout" button and if they have added products into their shopping cart, this information will be saved for when the user decides to log into the application again

Browsing and Review Products

After the user is able to successfully log into the application as a buyer, the window/frame will display a list of products for the users to browse. The users are able to see detailed information such as name, price, and quantity available of their product of interest.

Adding Products to Shopping Cart

When the user(buyer) has decided they would like to add their product of interest to their cart, the window/frame will give them the option to do so with a button labeled "Add to Cart". The user can add multiple of such products if they wish by continuously clicking the "Add to Cart" button. The user is able to review and see all the products they have added in the "Shopping Cart" window.

Editing Shopping Cart

Users(buyer) can edit what products and the quantity of the product that is in their shopping cart by simplifying clicking the "-" or "+" signs that are under the name of the product. This will alter the quantity in the cart and also change the total price of the shopping cart.

Purchasing Products

When the user(buyer) decides they are ready to check out, they are given a "Checkout Items" button. By clicking this button, the user will be brought to the purchase page and they are required to fill out their information such as name, number, address, credit card number, expiration date, and CCV number of the credit card. If the user does not fill in all these fields, the application will indicate there is missing information needed. When successful, the purchase page will indicate the purchase has gone through.

Editing Inventory (Seller)

The user(seller) is able to edit the inventory of their existing products in the sellers frame. The user can edit the quantity by simply clicking "-" or "+" under the desired product. When the seller edits this information and logs out, the buyers will instantly see this change.

Viewing Sales (Seller)

After the buyers have successfully completed their transaction, their information will be logged in the 'Metrics" on the sellers side. The sellers are able to see all the transactions by clicking "View Metrics" on the sellers frame.

Calculating Net Sales (Seller)

The seller can see their net profit by clicking the "View Metrics" button on the sellers frame and then clicking "View Profit" in which the application will do the calculations for the seller internally and display the profit in the lower left hand corner of the frame.

Use Cases

1) **User Login (Buyer)**
    a) User opens application
    b) User is met with login frame
    c) User enters username and password
    d) User clicks “login” button

   Variation 1:
    a) Continue at (d) in Use case 1
    b) Login frame informs user of invalid login credentials
    c) User enters correct credentials
    d) User clicks “login” button
2) **User Login (Seller)**
    a) Continue at (c) in Use Case 1
    b) User clicks “Seller” checkbox
    c) User clicks “login” button

   Variation 1:
    a) Continue at (d) in Use Case 1
    b) Login frame informs user of invalid login credentials
    c) User enters correct credentials
    d) User clicks “Seller” checkbox
    e) User clicks “login” button
3) **Add items to cart**
    a) Continue from (d) in Use Case 1
    b) User is met with products frame
    c) User selects product of interest by clicking “Add to Cart” button next to the wanted product
4) **Checkout**
    a) Continue with (d) from Use Case 1
    b) User navigates to “shopping cart”
    c) User is met with shopping cart frame
    d) User clicks “Checkout Items”
    e) User is met with purchase frame
    f) User enters full name, number, address, credit card number, expiration date, and CCV number
    g) User clicks “Submit” button
    h) Purchase frame displays “Purchase successful” message

   Variation 1:
    a) Continue with (e) from Use Case 4
    b) User enters required information and forgets some fields
    c) Purchase frame indicates the missing fields to the user

    d) User correct enters in required information
    e) User clicks “Submit” button
    f) Purchase displays “Purchase successful” message

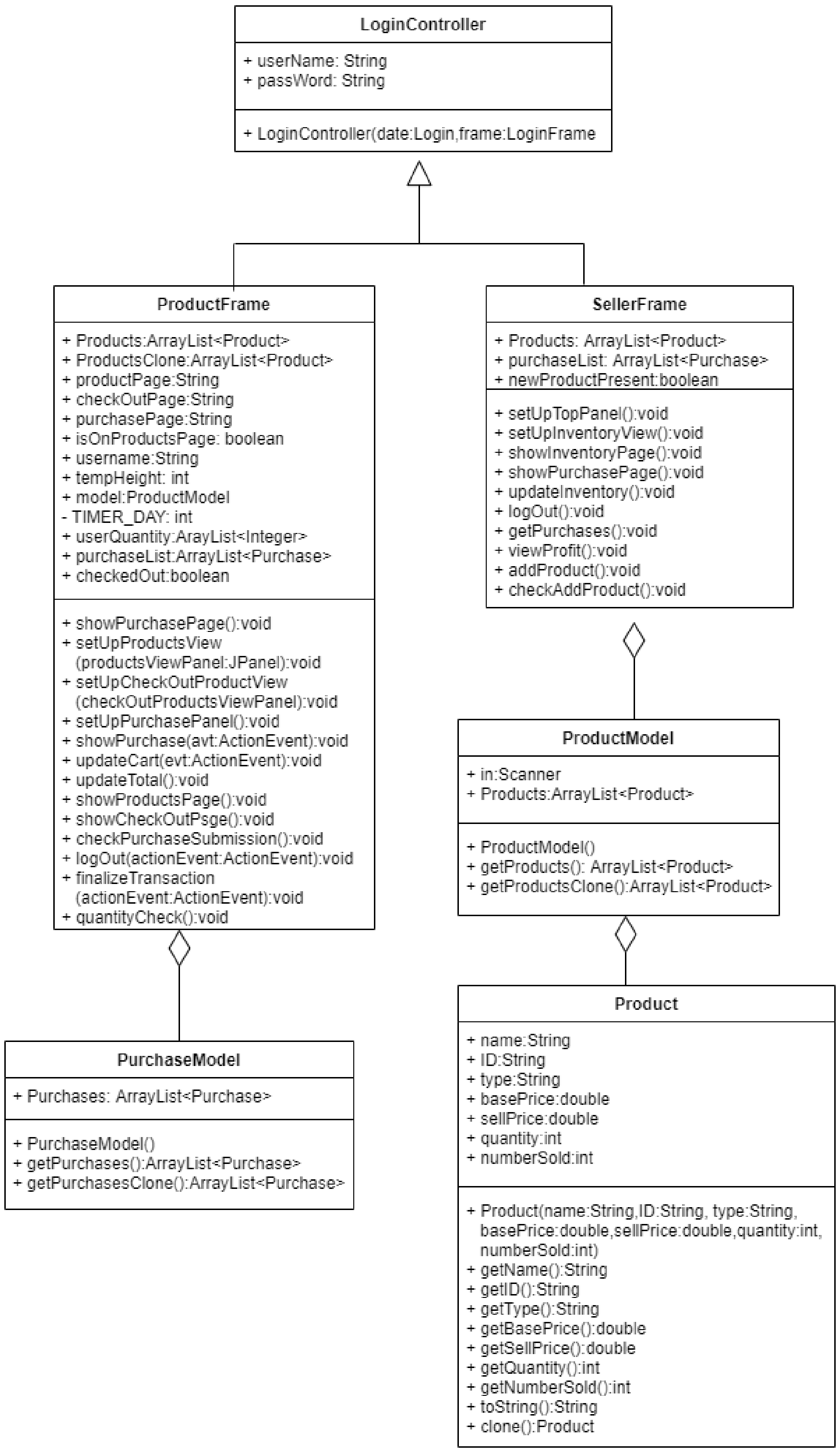
5) **User (Seller) accessing Sales Data**
    a) Continue with (c) from Use Case 2
    b) User is met with seller frame
    c) User clicks “View Metrics” button
    d) Seller frame displays sales data

6) **Calculating Net Profit (Seller)**
    a) Continue with (c) of Use Case 2
    b) User is met with seller frame
    c) User clicks “View Metrics”
    d) Seller frame displays sales data
    e) User clicks “View Profit”
    f) Frame calculates and displays the seller’s net profit

7) **User (Seller) adding new product**
    a) Continue with with (c) of Use Case 2
    b) User is met with the seller frame
    c) User clicks “Add Products”
    d) User fills in required information
    e) User clicks “Submit”

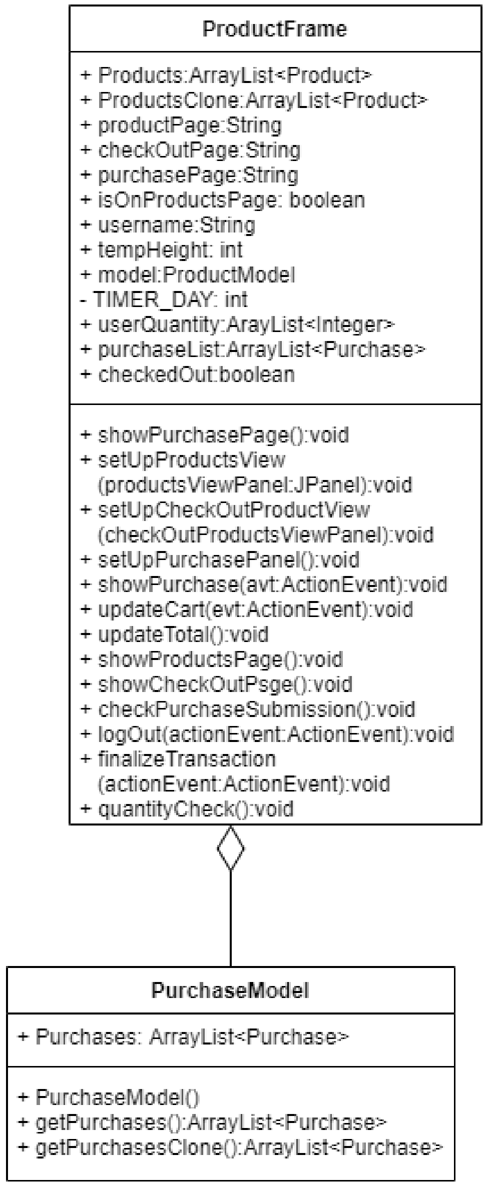
   Variation 1:
    a) Continue with (e) of Use Case 7
    b) Frame indicates there are missing fields
    c) User enters in missing fields
    d) User clicks “Submit”

UML Class Diagram

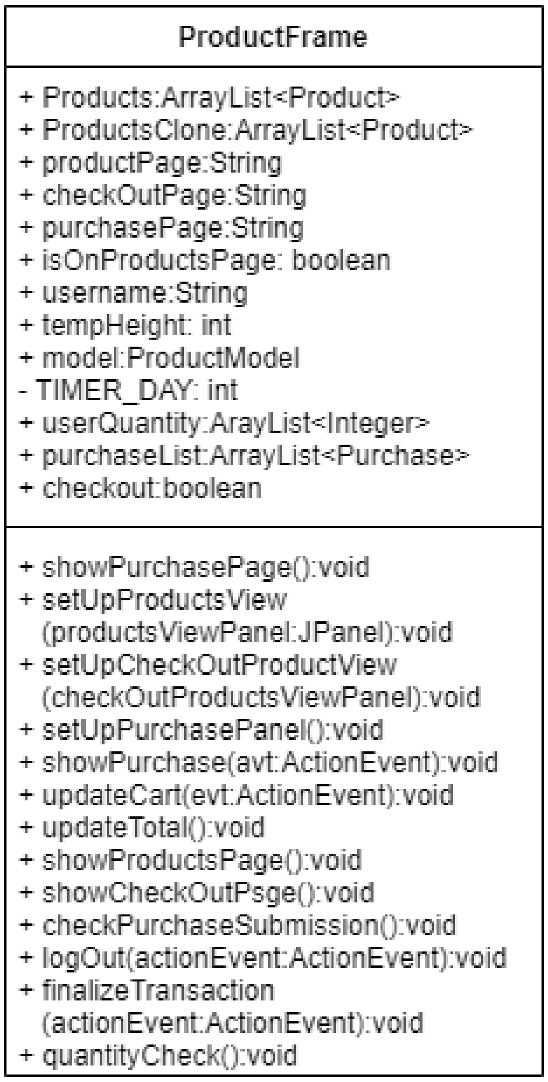

LoginController
+ userName: String
+ passWord: String
+ LoginController(date:Login,frame:LoginFrame
ProductFrame
+ Products:ArrayList<Product>
+ ProductsClone:ArrayList<Product>
+ productPage:String
+ checkOutPage:String
+ purchasePage:String
+ isOnProductsPage: boolean
+ username:String
+ tempHeight: int
+ model:ProductModel
- TIMER_DAY: int
+ userQuantity:ArayList<Integer>
+ purchaseList:ArrayList<Purchase>
+ checkedOut:boolean
+ showPurchasePage():void
+ setUpProductsView (productsViewPanel:JPanel):void
+ setUpCheckOutProductView (checkOutProductsViewPanel):void
+ setUpPurchasePanel():void
+ showPurchase(avt:ActionEvent):void
+ updateCart(evt:ActionEvent):void
+ updateTotal():void
+ showProductsPage():void
+ showCheckOutPsge():void
+ checkPurchaseSubmission():void
+ logOut(actionEvent:ActionEvent):void
+ finalizeTransaction (actionEvent:ActionEvent):void
+ quantityCheck():void
SellerFrame
+ Products: ArrayList<Product>
+ purchaseList: ArrayList<Purchase>
+ newProductPresent:boolean
+ setUpTopPanel():void
+ setUpInventoryView():void
+ showInventoryPage():void
+ showPurchasePage():void
+ updateInventory():void
+ logOut():void
+ getPurchases():void
+ viewProfit():void
+ addProduct():void
+ checkAddProduct():void
ProductModel
+ in:Scanner
+ Products:ArrayList<Product>
+ ProductModel()
+ getProducts(): ArrayList<Product>
+ getProductsClone():ArrayList<Product>
PurchaseModel
+ Purchases: ArrayList<Purchase>
+ PurchaseModel()
+ getPurchases():ArrayList<Purchase>
+ getPurchasesClone():ArrayList<Purchase>
Product
+ name:String
+ ID:String
+ type:String
+ basePrice:double
+ sellPrice:double
+ quantity:int
+ numberSold:int
+ Product(name:String,ID:String, type:String, basePrice:double,sellPrice:double,quantity:int, numberSold:int)
+ getName():String
+ getID():String
+ getType():String
+ getBasePrice():double
+ getSellPrice():double
+ getQuantity():int
+ getNumberSold():int
+ toString():String
+ clone():Product

Design Patterns

Decorator Design

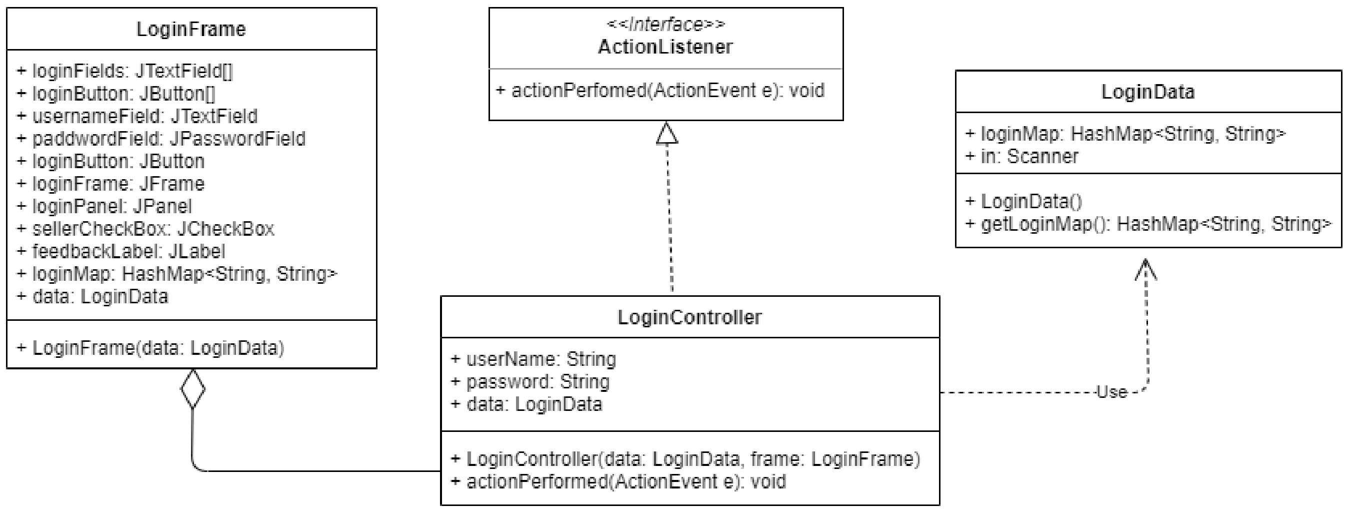

ProductFrame
+ Products:ArrayList<Product>
+ ProductsClone:ArrayList<Product>
+ productPage:String
+ checkOutPage:String
+ purchasePage:String
+ isOnProductsPage: boolean
+ username:String
+ tempHeight: int
+ model:ProductModel
- TIMER_DAY: int
+ userQuantity:ArayList<Integer>
+ purchaseList:ArrayList<Purchase>
+ checkedOut:boolean
+ showPurchasePage():void
+ setUpProductsView (productsViewPanel:JPanel):void
+ setUpCheckOutProductView (checkOutProductsViewPanel):void
+ setUpPurchasePanel():void
+ showPurchase(avt:ActionEvent):void
+ updateCart(evt:ActionEvent):void
+ updateTotal():void
+ showProductsPage():void
+ showCheckOutPsge():void
+ checkPurchaseSubmission():void
+ logOut(actionEvent:ActionEvent):void
+ finalizeTransaction (actionEvent:ActionEvent):void
+ quantityCheck():void
PurchaseModel
+ Purchases: ArrayList<Purchase>
+ PurchaseModel()
+ getPurchases():ArrayList<Purchase>
+ getPurchasesClone():ArrayList<Purchase>

Singleton Design


ProductFrame
+ Products:ArrayList<Product>
+ ProductsClone:ArrayList<Product>
+ productPage:String
+ checkOutPage:String
+ purchasePage:String
+ isOnProductsPage: boolean
+ username:String
+ tempHeight: int
+ model:ProductModel
- TIMER_DAY: int
+ userQuantity:ArayList<Integer>
+ purchaseList:ArrayList<Purchase>
+ checkout:boolean
+ showPurchasePage():void
+ setUpProductsView (productsViewPanel:JPanel):void
+ setUpCheckOutProductView (checkOutProductsViewPanel):void
+ setUpPurchasePanel():void
+ showPurchase(avt:ActionEvent):void
+ updateCart(evt:ActionEvent):void
+ updateTotal():void
+ showProductsPage():void
+ showCheckOutPsge():void
+ checkPurchaseSubmission():void
+ logOut(actionEvent:ActionEvent):void
+ finalizeTransaction (actionEvent:ActionEvent):void
+ quantityCheck():void

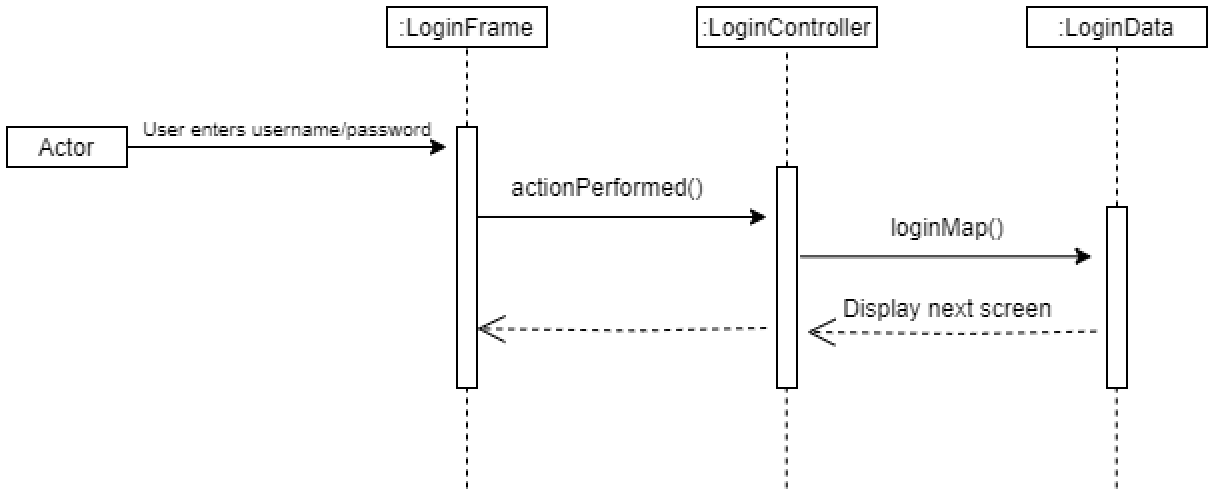
MVC UML Diagram

Login Diagram

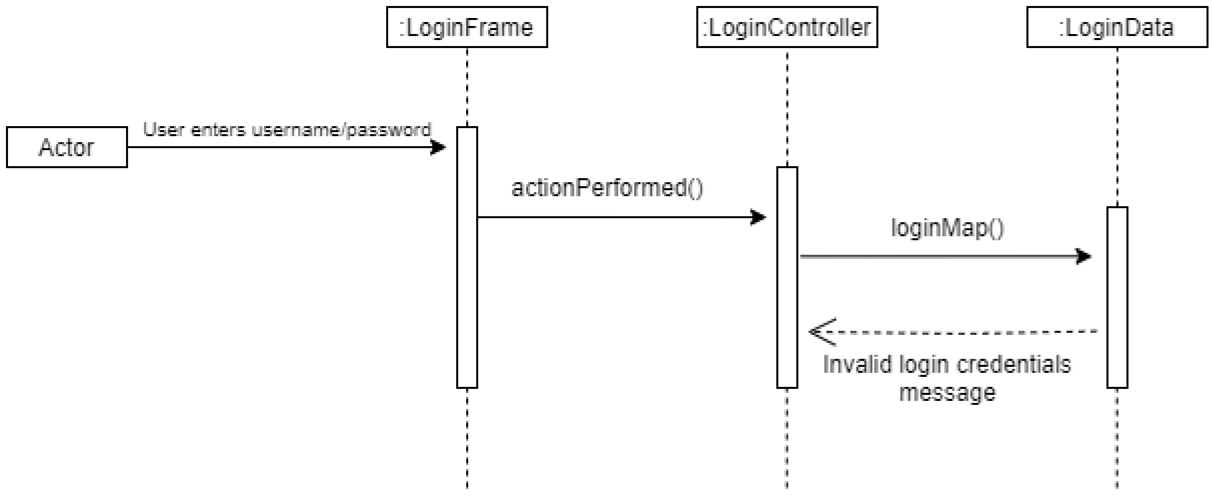

LoginFrame
+ loginFields: JTextField[]
+ loginButton: JButton[]
+ usernameField: JTextField
+ paddwordField: JPasswordField
+ loginButton: JButton
+ loginFrame: JFrame
+ loginPanel: JPanel
+ sellerCheckBox: JCheckBox
+ feedbackLabel: JLabel
+ loginMap: HashMap<String, String>
+ data: LoginData
+ LoginFrame(data: LoginData)
<<Interface>>
ActionListener
+ actionPerfomed(ActionEvent e): void
LoginData
+ loginMap: HashMap<String, String>
+ in: Scanner
+ LoginData()
+ getLoginMap(): HashMap<String, String>
LoginController
+ userName: String
+ password: String
+ data: LoginData
+ LoginController(data: LoginData, frame: LoginFrame)
+ actionPerformed(ActionEvent e): void
Use

Sequence Diagram

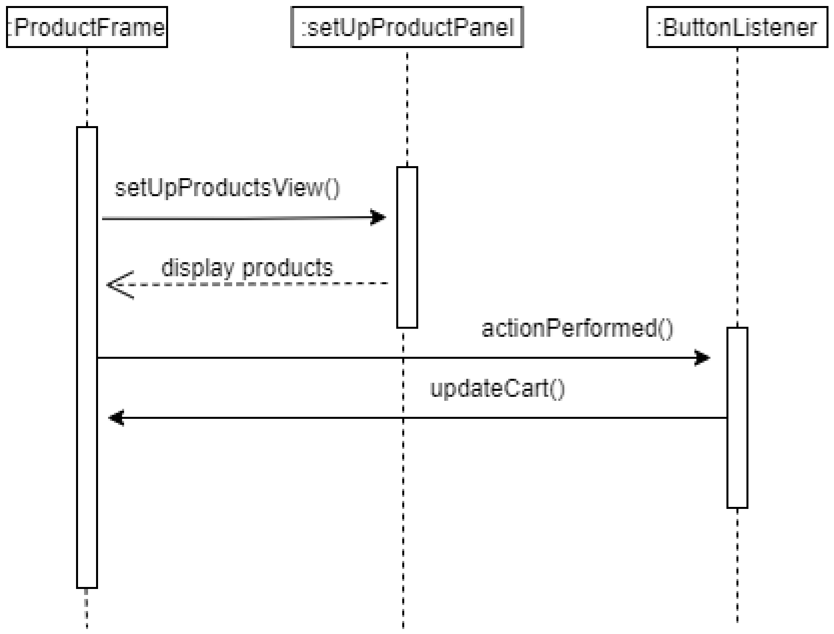
User Login (UC1)



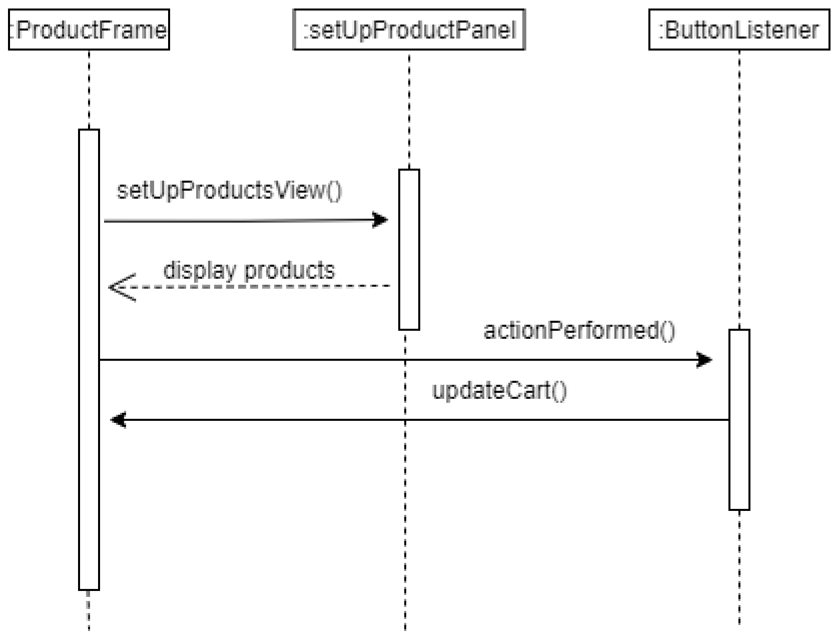
User Login - Failed (UC1 Variation 1)



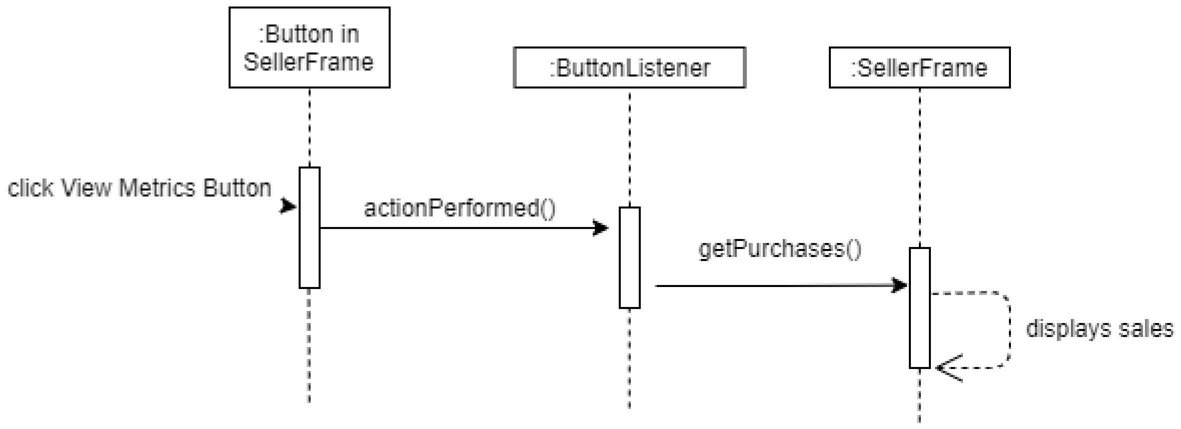
Add Items to Cart



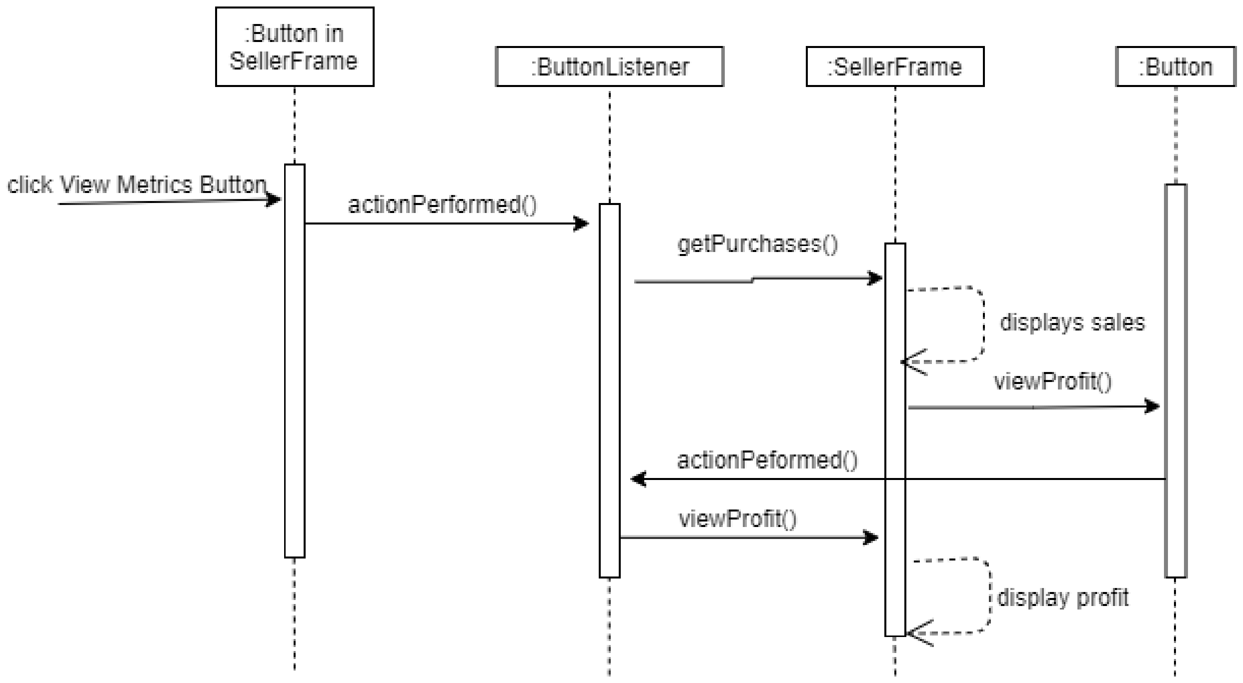
Checkout



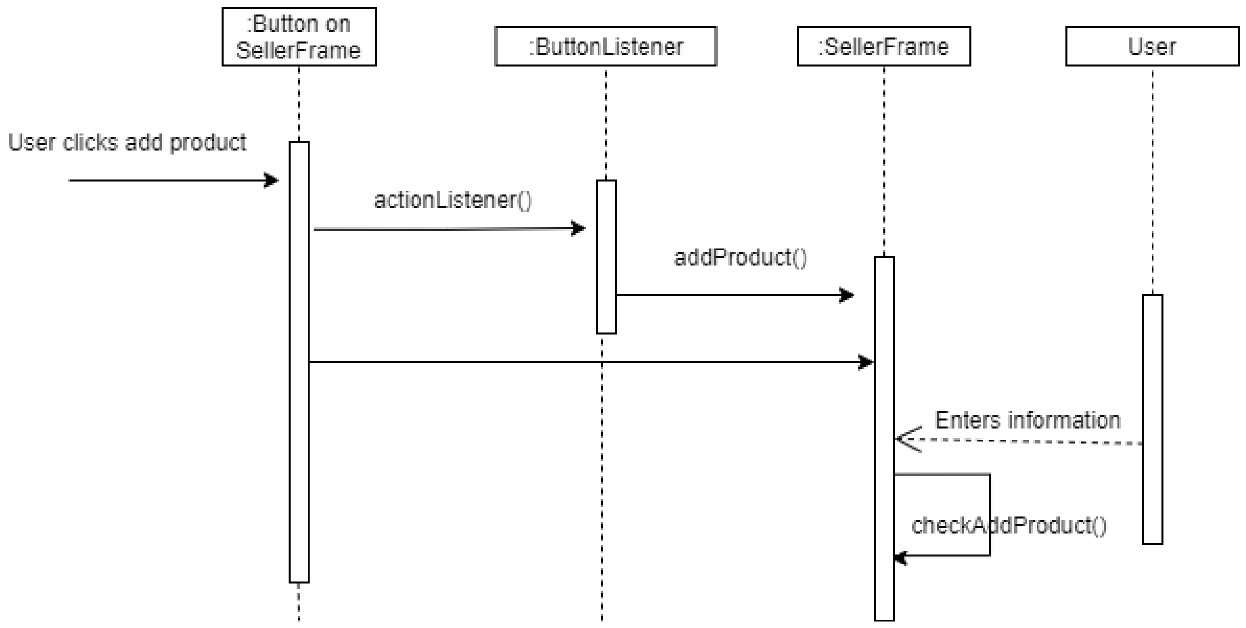
Accessing Sales Data



Viewing Profit

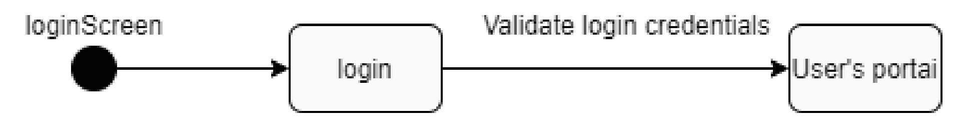


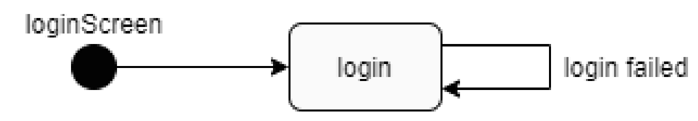
Adding Inventory



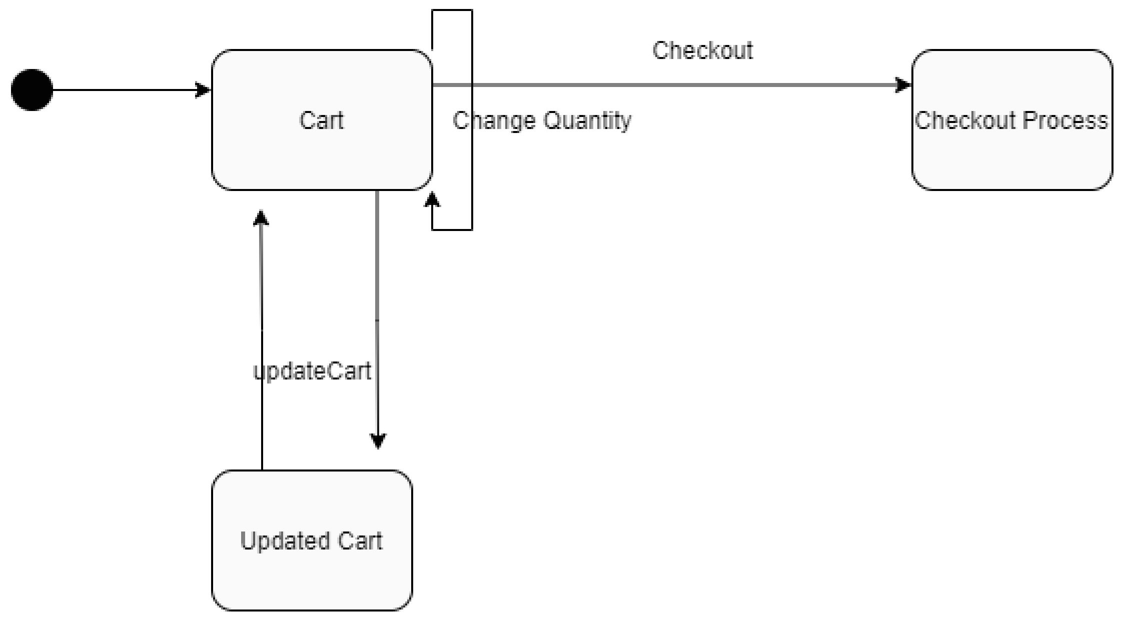
State Diagram

Login State

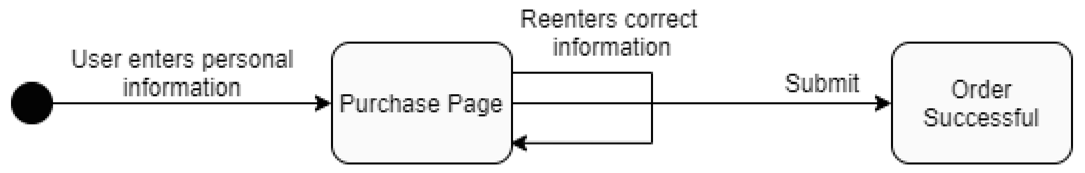




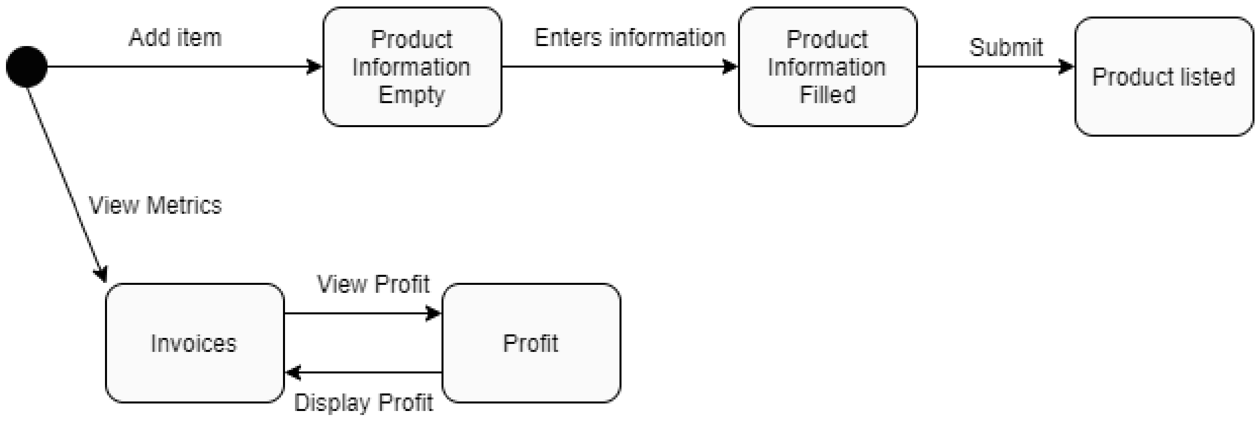
Shopping Cart State Diagram



Checkout State


User enters personal information
Purchase Page
Reenters correct information
Submit
Order Successful

Sellers Page State


Add item
Product Information Empty
Enters information
Product Information Filled
Submit
Product listed
View Metrics
Invoices
View Profit
Profit
Display Profit