Detailed Project for our Online Store

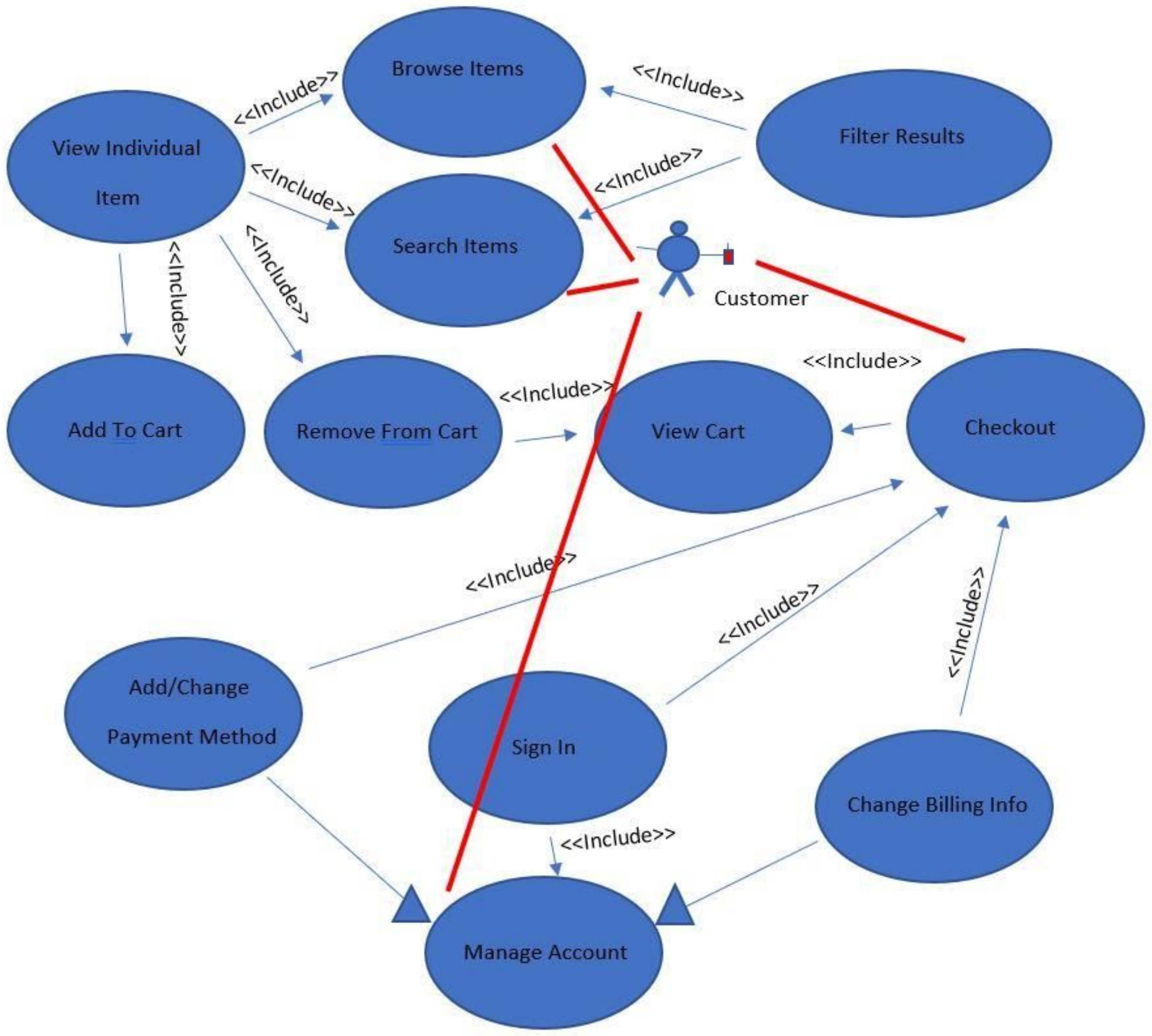
By: Matthew Heger, Payson Harris, Elan Canfield

Our project addresses the problems of sales, bookkeeping, and inventory tracking for an online store. The application facilitates sales by being an online storefront, allowing customers to browse, search, and purchase items. When a customer makes a purchase, the bookkeeping and inventory features of the application play their role. The bookkeeping side of the application will record the sale, including the amount of each item bought, as well as the total cost of the sale and the time and date of the sale. The owners will be able to see data, including revenue over a time period, and the most popular products in the store.

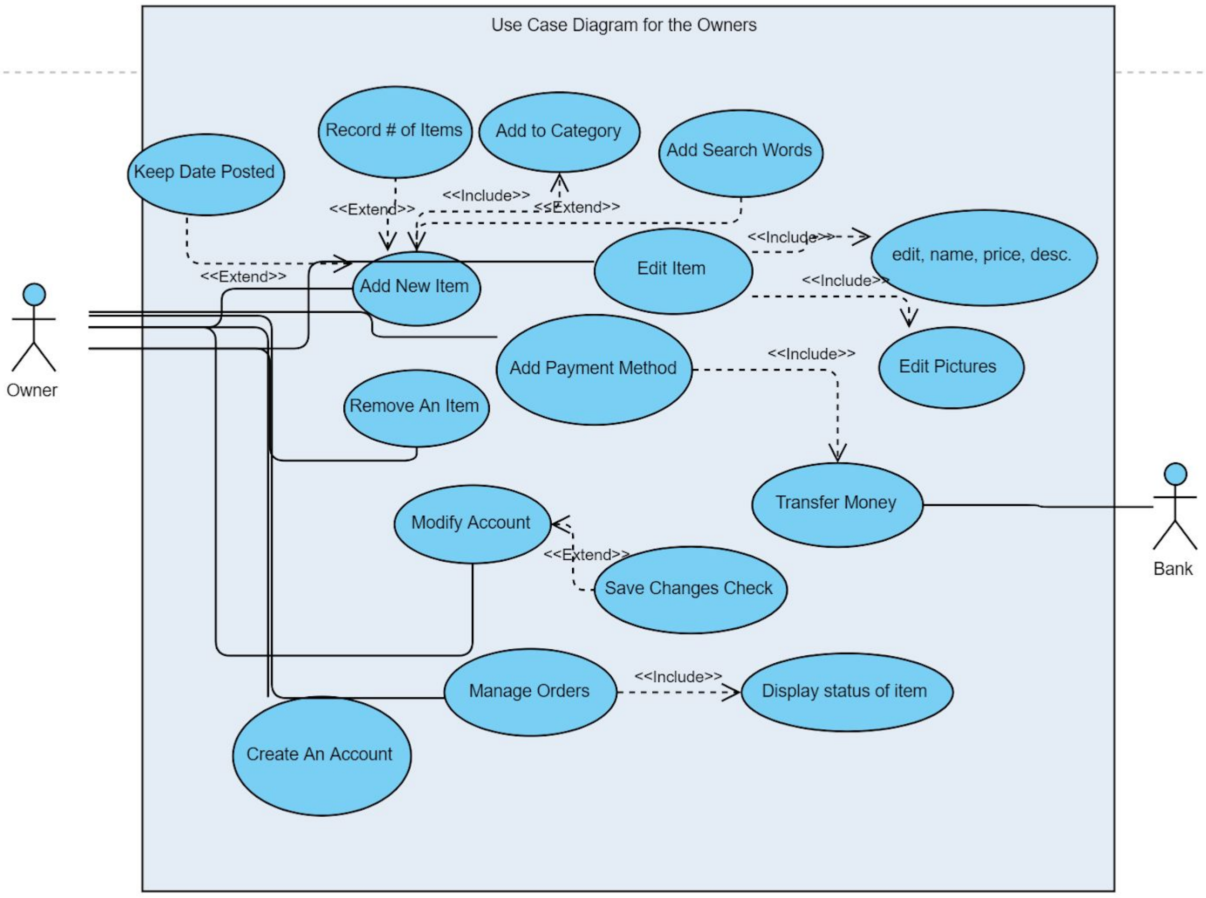
This store will be organized as an object-oriented system so it adheres to the object-oriented model. The store itself will be an object with an inventory and a record of orders. The inventory is an ArrayList of Item objects. Each item object within the inventory will have a name, description, price, and quantity. The quantity will be adjusted automatically when a user purchases the item, or the owner adds more of the item to the inventory. The record of orders is an ArrayList of Order objects, which contain data about a customer's order. This data includes a Customer object representing the customer, the items ordered, a timestamp, and a boolean value which keeps track of the item's shipping status. A customer object will have the customer's name, email address, and shipping address saved, and will also keep track of the items in the customer's cart.

Use Case Diagrams

Customer Use Case Diagram



Owner Use Case Diagram



Use Cases

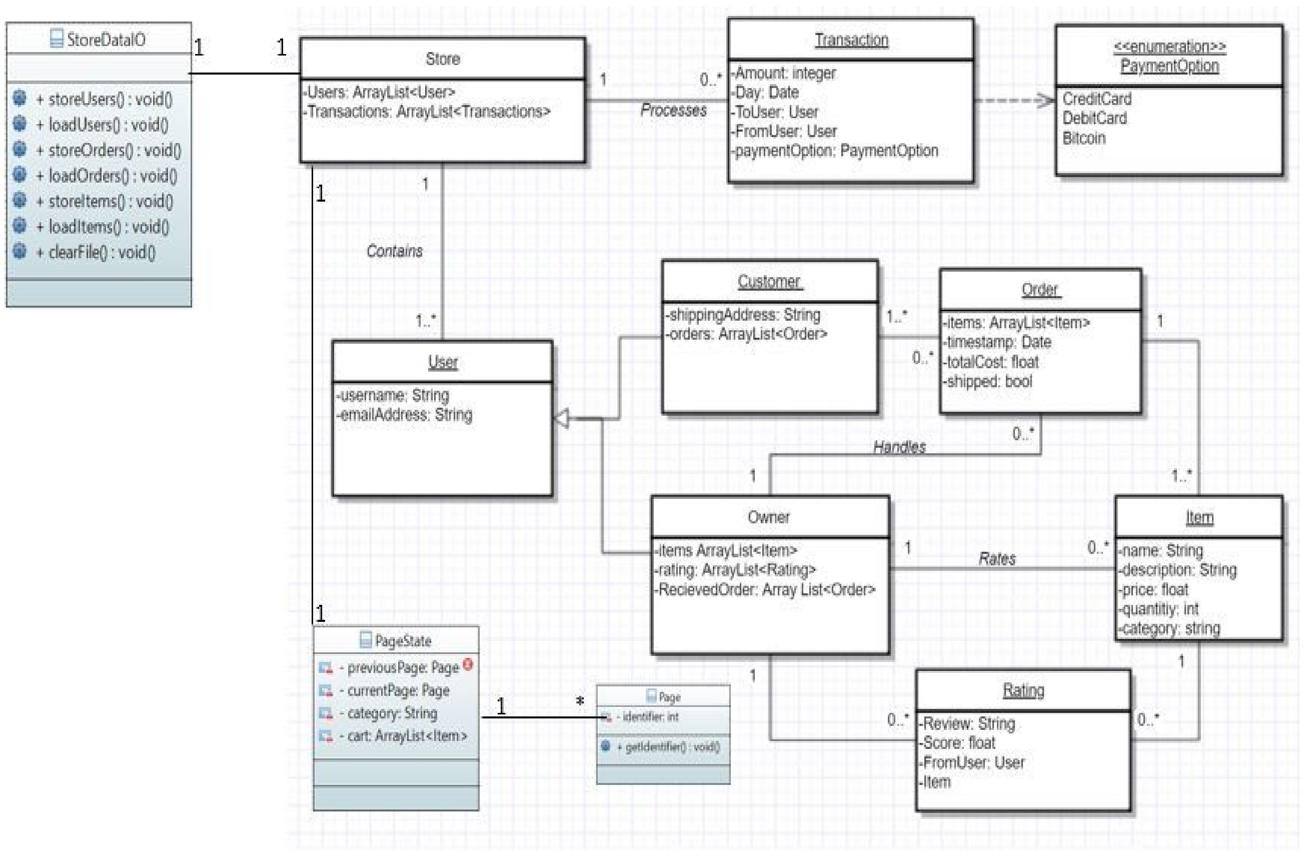
Use Case Name: Add New Item
Actors: Owner
Preconditions: Owner is on the inventory page, where the “Add New Item” button is.
Description: Owner is directed to an “Add New Item” page which prompts owner for product information. Owner sets name, description, price, and quantity for the new item. Owner can also upload images of the product that the customer can browse. Once the owner confirms the details of the new item, the item is added to the inventory and becomes available for purchase by customers. Also, the new item is added to its specific category and the posting date is recorded in the system.
Exceptions: ● Item with given name already exists ● Negative quantity entered ● Negative price entered
Postconditions: Owner is taken back to inventory page

Use Case Name: Add Payment Method
Actors: Owner and Customer
Preconditions: Owner is on the home screen and the system is waiting for an input.
Description: The owner will select between a few payment options credit, debit, or bitcoin. If credit or debit is selected, then the user will enter the following information into the text fields: the card number, expiration date, name on card, and security code. If bitcoin is selected, then the owner will select the bitcoin wallet they use and put in the transfer location information. There will be a spot to select whether the payment method will be used to send or recieve money.
Exceptions: ● Invalid card number or Bitcoin address is entered ● Expiration date on card has expired ● Invalid CVC ● Bank declined Transaction
Postconditions: A success display will show that the owner successfully added a valid payment method and it will take them back to the main menu.

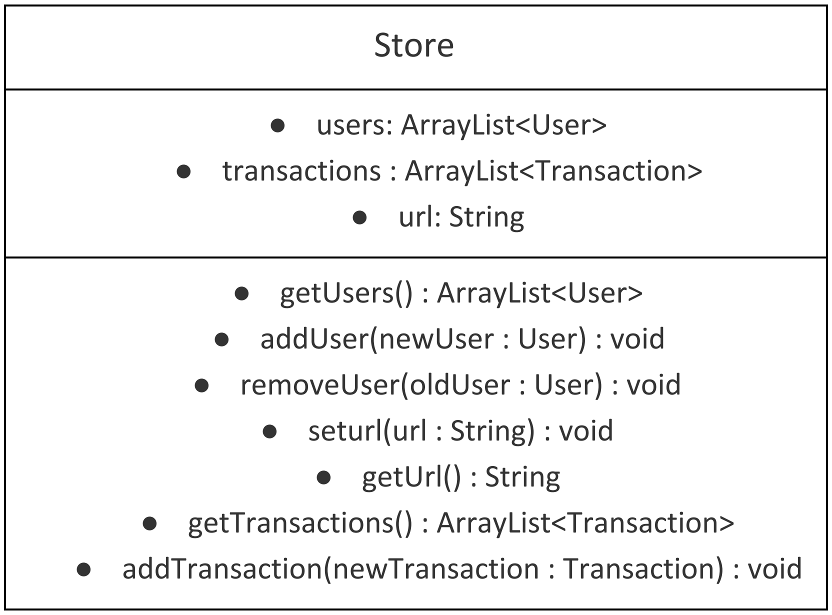
Use Case Name: Search Items
Actors: Customer, Owner
Preconditions: The system is waiting at the main menu.
Description: A customer or the owner may initiate a search simply by typing an item name or keyword into the search box, which should be located in the main menu, and hitting enter. The system will proceed to search through the list of items and display any items containing the search phrases.
Exceptions: ● No items match the search: If no items were found that contain any part of the search, then the following message is displayed. “Your search did not match any items”
Postconditions: A list of all items matching the search will be displayed.

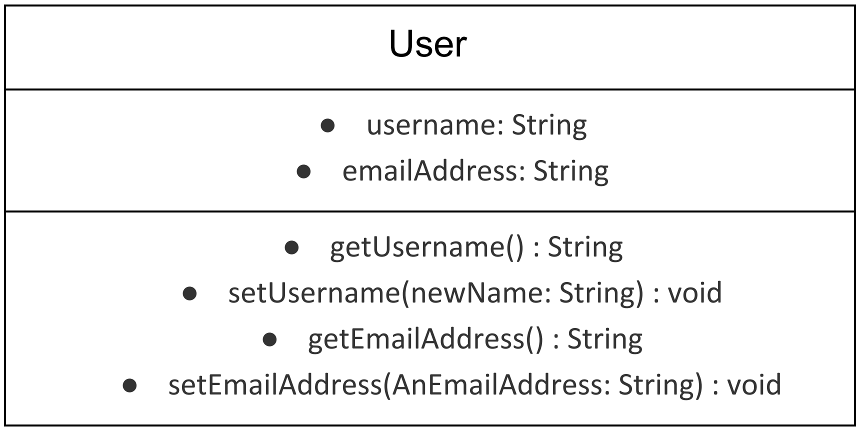
Use Case Name: Checkout
Actors: Customer
Preconditions: Customer is on the Cart page and presses “checkout” button.
Description: Customer is directed to a page which collects order information from the user. If the user is signed in, the data from the user’s Customer object will be used to fill name, email address, and shipping address fields. Otherwise, customer will have to enter this data manually. Then, the user will be taken to another page which displays the order breakdown and prompts the user for payment information. If the user already has payment information on file, they can use the previously entered data. Otherwise, the customer will have to enter payment information manually. Then, the user presses the confirm payment button.
Exceptions: ● Invalid email or shipping address ● Invalid payment information
Postconditions: Customer is taken to a “Thank you” page. From there, customer can choose to go back to the home page.

Domain Class Model



Detailed Individual Classes


Store
users: ArrayList<User>
transactions : ArrayList<Transaction>
url: String
getUsers() : ArrayList<User>
addUser(newUser : User) : void
removeUser(oldUser : User) : void
seturl(url : String) : void
getUrl() : String
getTransactions() : ArrayList<Transaction>
addTransaction(newTransaction : Transaction) : void


User
username: String
emailAddress: String
getUsername() : String
setUsername(newName: String) : void
getEmailAddress() : String
setEmailAddress(AnEmailAddress: String) : void

Owner extends User

- items : ArrayList<Item>
- ratings: ArrayList< Rating>
- recievedOrders : ArrayList<Order>

- Owner()
- Owner(newUsername: String)
- getReceivedOrders() : ArrayList<Order>
- addReceivedOrder(newOrder: Order) : void
- removeReceivedOrder(oldOrder: Order) : void
- getRatings() : ArrayList<Rating>
- addRating(newRating : Rating) : void
- removeRating(oldRating : Rating) : void
- getAverageRatingScore() : float

Customer extends User

- shippingAddress: String
- orders: ArrayList<Order>

- Customer()
- getShippingAddress() : String
- setShippingAddress(newShippingAddress: String) : void
- getNumberOfOrders() : int
- addOrder(newOrder:Order): void
- removeOrder(oldOrder:Order):void

Item

- name: String
- description: String
- price : float
- quantity: int

- Item()
- Item(newName: String)
- getName() : String
- setName(newName: String) : void
- getDescription() : String
- setDescription(adescription:String) : void
- getPrice() : float
- setPrice(newPrice: float) : void
- getQuantity() : int
- setQuantity(int aquantity): void
- incrementQuantity() : void
- decrementQuantity() : void

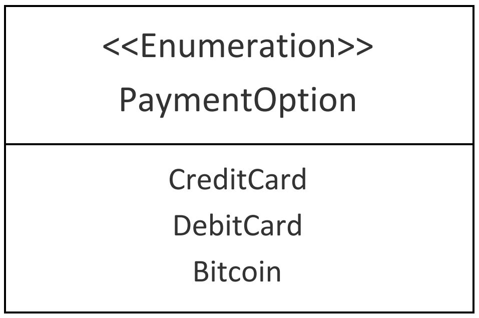
Order

- items: ArrayList<Item>
- timestamp : Date
- shipped : bool

- Order()
- getItems() : ArrayList<Item>
- addItem(newItem : Item) : void
- removeItem(oldItem : Item) : void
- getTimestamp() : Date
- setTimestamp(newTimestamp : Date) : void
- getTotalCost() : float
- setShipped(newShipped : bool) : void

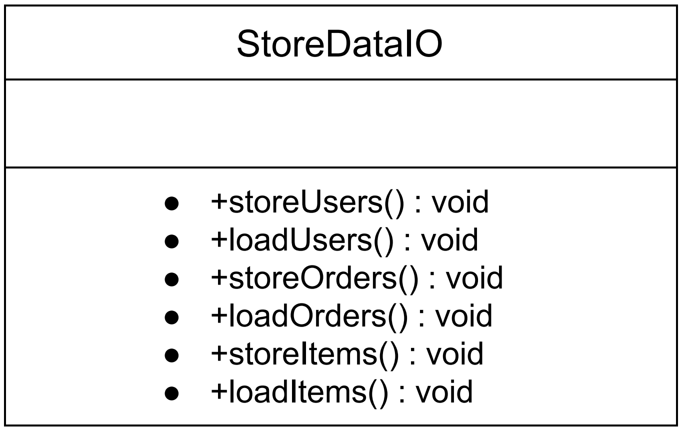
Rating
• review : String • score : float • Item: Item
• Rating() • Rating(review : String, score : float) • getReview() : String • setReview(newReview : String) : void • getScore() : float • setScore(newScore : float) : void • setItem(newItem : Item) : void • getItem(): Item

Transaction
• amount : int • date : Date • toUser : User • fromUser : User • paymentOption : PaymentOption
• Transaction() • setAmount(amount : int) : void • getAmount() : int • setDate(dateOfSale : Date) : void • getDate() : Date • setToUser(purchaser : User) : void • getToUser() : User • setFromUser(seller : User) : void • getFromUser() : User • setPaymentOption(payMethod : PaymentOption) : void • getPaymentOption() : PaymentOption



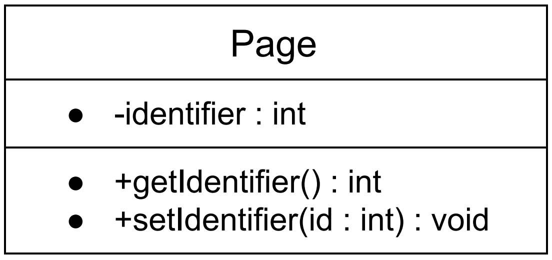
<<Enumeration>>
PaymentOption

CreditCard
DebitCard
Bitcoin



StoreDataIO

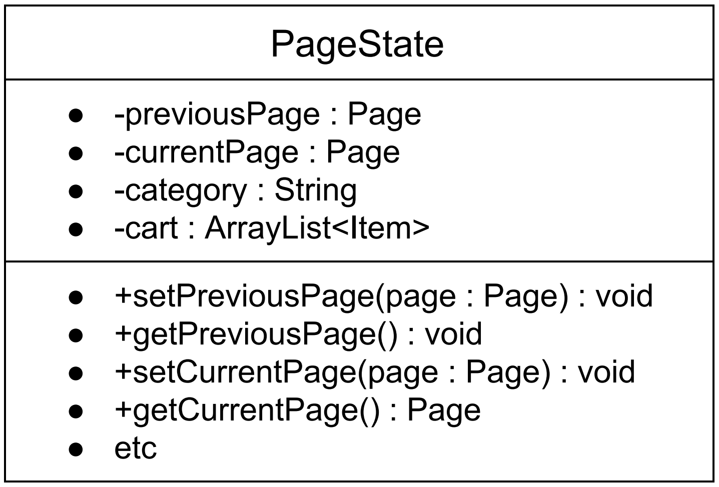
- +storeUsers() : void
- +loadUsers() : void
- +storeOrders() : void
- +loadOrders() : void
- +storeItems() : void
- +loadItems() : void



Page

- -identifier : int

- +getIdentifier() : int
- +setIdentifier(id : int) : void

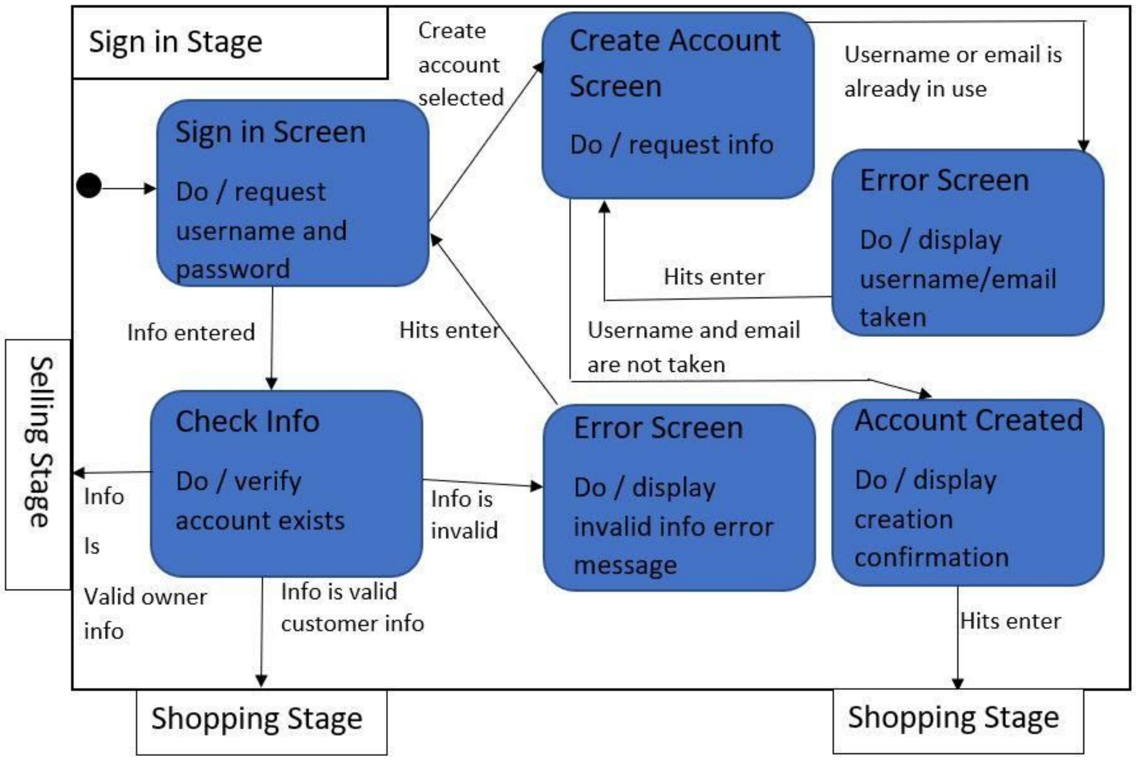


PageState

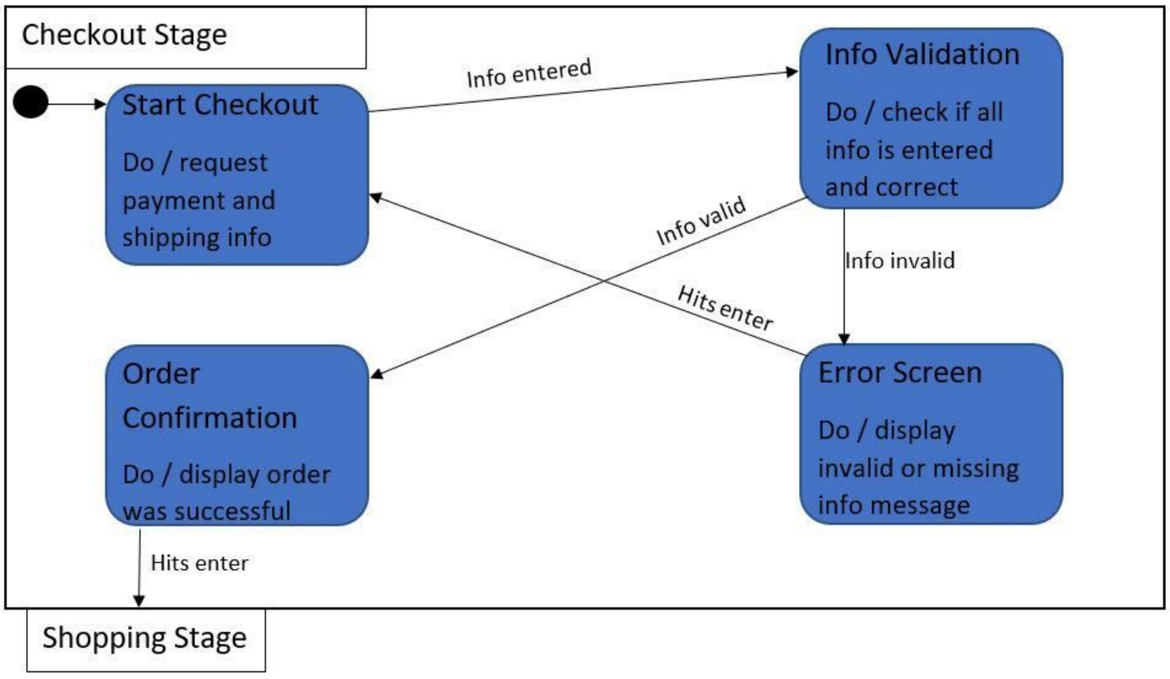
- -previousPage : Page
- -currentPage : Page
- -category : String
- -cart : ArrayList<Item>

- +setPreviousPage(page : Page) : void
- +getPreviousPage() : void
- +setCurrentPage(page : Page) : void
- +getCurrentPage() : Page
- etc

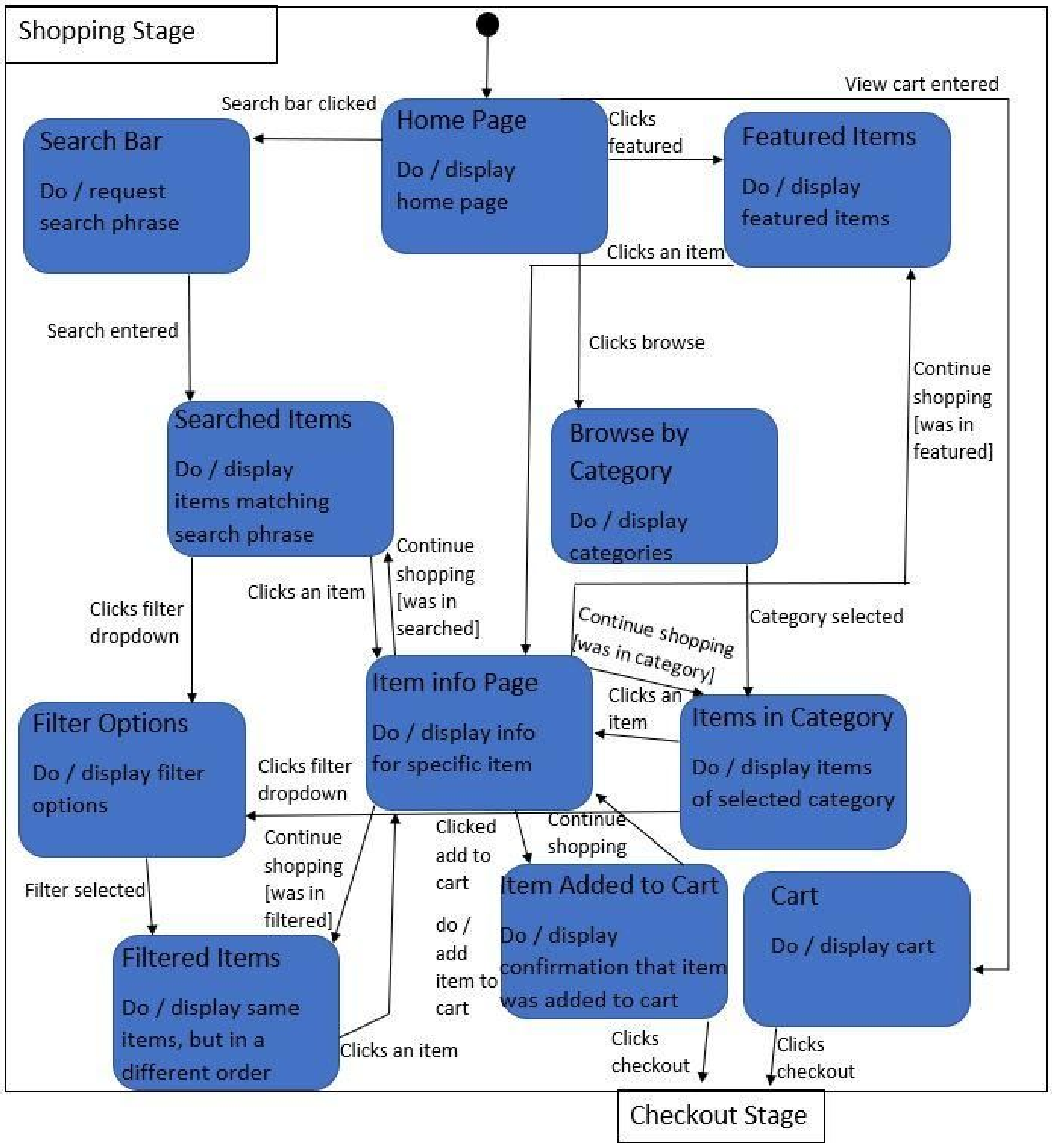
State Diagrams

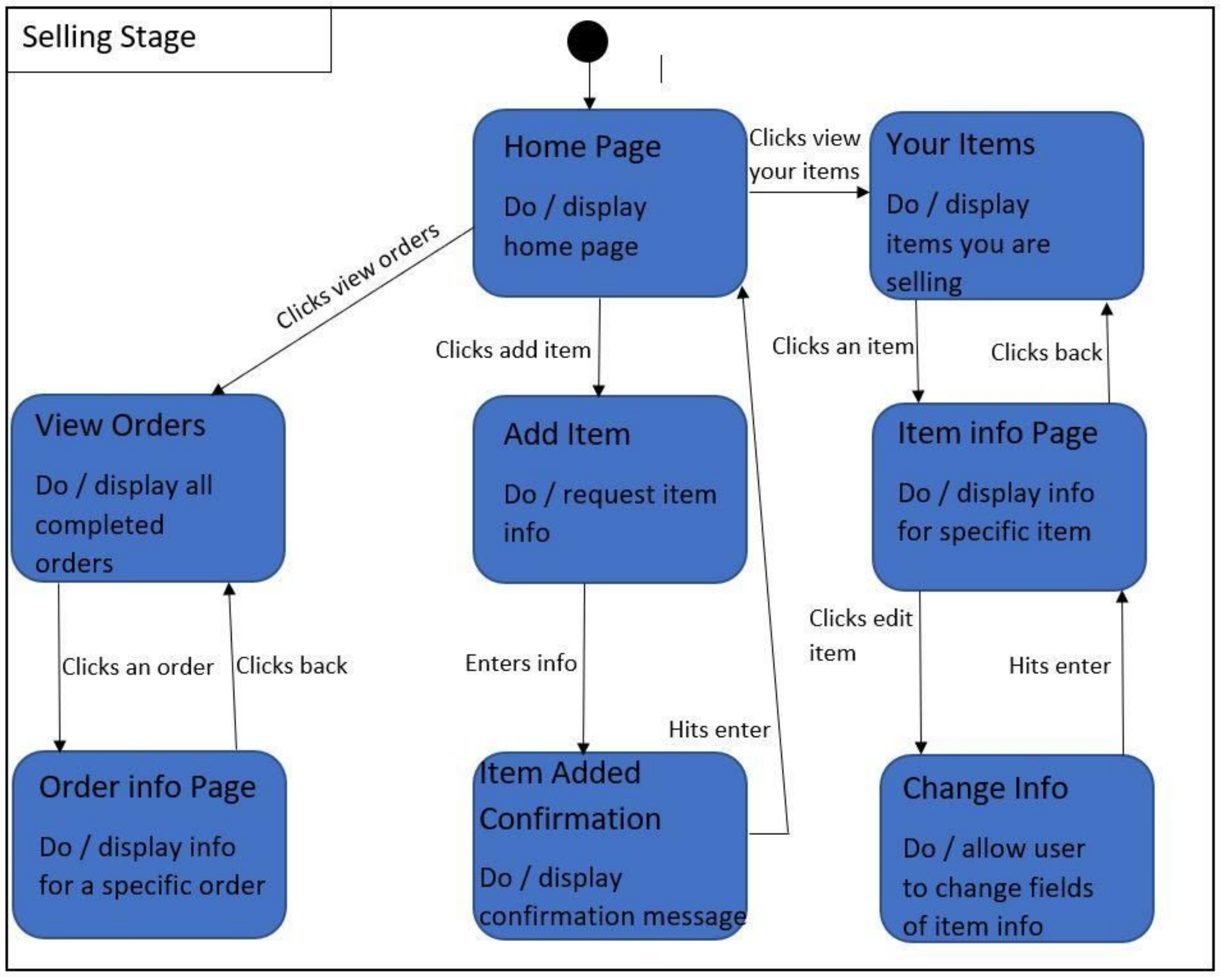


Sign in Stage
Sign in Screen
Do / request username and password
Create account selected
Create Account Screen
Do / request info
Username or email is already in use
Error Screen
Do / display username/email taken
Hits enter
Username and email are not taken
Info entered
Hits enter
Check Info
Do / verify account exists
Info is invalid
Error Screen
Do / display invalid info error message
Account Created
Do / display creation confirmation
Selling Stage
Info Is Valid owner info
Info is valid customer info
Hits enter
Shopping Stage
Shopping Stage



Checkout Stage
Start Checkout
Do / request payment and shipping info
Info entered
Info Validation
Do / check if all info is entered and correct
Info valid
Info invalid
Hits enter
Order Confirmation
Do / display order was successful
Error Screen
Do / display invalid or missing info message
Hits enter
Shopping Stage


Shopping Stage
Home Page
Do / display home page
Search bar clicked
Search Bar
Do / request search phrase
Clicks featured
Featured Items
Do / display featured items
View cart entered
Clicks an item
Clicks browse
Search entered
Continue shopping [was in featured]
Searched Items
Do / display items matching search phrase
Browse by Category
Do / display categories
Continue shopping [was in searched]
Clicks an item
Clicks filter dropdown
Continue shopping [was in category]
Category selected
Clicks an item
Item info Page
Do / display info for specific item
Items in Category
Do / display items of selected category
Filter Options
Do / display filter options
Clicks filter dropdown
Continue shopping [was in filtered]
Clicked add to cart
do / add item to cart
Continue shopping
Filter selected
Item Added to Cart
Do / display confirmation that item was added to cart
Cart
Do / display cart
Filtered Items
Do / display same items, but in a different order
Clicks an item
Clicks checkout
Clicks checkout
Checkout Stage


Selling Stage
Home Page
Do / display home page
Clicks view your items
Your Items
Do / display items you are selling
Clicks view orders
Clicks add item
Clicks an item
Clicks back
View Orders
Do / display all completed orders
Add Item
Do / request item info
Item info Page
Do / display info for specific item
Clicks edit item
Clicks an order
Clicks back
Enters info
Hits enter
Hits enter
Order info Page
Do / display info for a specific order
Item Added Confirmation
Do / display confirmation message
Change Info
Do / allow user to change fields of item info

Online Store Walkthrough video:
https://youtu.be/YYnsVw3aQWM